From Perception over Anticipation to Manipulation

A dissertation submitted towards the degree
Doctor Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by
Wenbin Li, M.Sc.

Saarbrücken, 2018

Day of Colloquium	25/04/2018
Dean of the Faculty	Univ.-Prof. Dr. Sebastian Hack
Chair of the Committee	Prof. Dr. Dietrich Klakow
First Reviewer, Advisor	Dr. Mario Fritz
Second Reviewer	Prof. Dr. Aleš Leonardis
Third Reviewer	Prof. Dr. Philipp Slusallek
Academic Assistant	Dr. Michael Xuelin Huang

Abstract

From autonomous driving cars to surgical robots, robotic system has enjoyed significant growth over the past decade. With the rapid development in robotics alongside the evolution in the related fields, such as computer vision and machine learning, integrating perception, anticipation and manipulation is key to the success of future robotic system. In this thesis, we explore different ways of such integration to extend the capabilities of a robotic system to take on more challenging real world tasks.

On anticipation and perception, we address the recognition of ongoing activity from videos. In particular we focus on long-duration and complex activities and hence propose a new challenging dataset to facilitate the work. We introduce hierarchical labels over the activity classes and investigate the temporal accuracy-specificity trade-offs. We propose a new method based on recurrent neural networks that learns to predict over this hierarchy and realize accuracy specificity trade-offs. Our method outperforms several baselines on this new challenge.

On manipulation with perception, we propose an efficient framework for programming a robot to use human tools. We first present a novel and compact model for using tools described by a tip model. Then we explore a strategy of utilizing a dual-gripper approach for manipulating tools – motivated by the absence of dexterous hands on widely available general purpose robots. Afterwards, we embed the tool use learning into a hierarchical architecture and evaluate it on a Baxter research robot.

Finally, combining perception, anticipation and manipulation, we focus on a block stacking task. First we explore how to guide robot to place a single block into the scene without collapsing the existing structure. We introduce a mechanism to predict physical stability directly from visual input and evaluate it first on a synthetic data and then on real-world block stacking. Further, we introduce the target stacking task where the agent stacks blocks to reproduce a tower shown in an image. To do so, we create a synthetic block stacking environment with physics simulation in which the agent can learn block stacking end-to-end through trial and error, bypassing to explicitly model the corresponding physics knowledge. We propose a goal-parametrized GDQN model to plan with respect to the specific goal. We validate the model on both a navigation task in a classic gridworld environment and the block stacking task.

Zusammenfassung

Von autonom fahrenden Autos bis zu chirurgischen Robotern haben Robotersysteme in den letzten zehn Jahren ein beträchtliches Wachstum erfahren. Mit der rasanten Entwicklung in der Robotik und der Entwicklung in den verwandten Bereichen, wie Computer Vision und Machine Learning, ist die Integration von Wahrnehmung, Antizipation und Handhabung der Schlüssel zum Erfolg zukünftiger Robotersysteme. In dieser Arbeit untersuchen wir verschiedene Möglichkeiten einer solchen Integration, um die Fähigkeiten eines Robotersystems zur Bewältigung anspruchsvollerer Aufgaben in der realen Welt zu erweitern.

Im Bereich der Antizipation und Wahrnehmung beschäftigen wir uns mit der Erkennung laufender Aktivitäten aus Videos. Insbesondere konzentrieren wir uns auf lang andauernde und komplexe Aktivitäten und schlagen somit einen neuen anspruchsvollen Datensatz vor, um die Arbeit zu erleichtern. Wir führen hierarchische Label über die Aktivitätsklassen ein und untersuchen die zeitlichen Zielkonflikte zwischen Genauigkeit und Spezifität. Wir schlagen eine neue auf rekurrenten neuronalen Netzen basierende Methode vor, die lernt, über diese Hierarchie vorherzusagen und Zielkonflikte zwischen Genauigkeit und Spezifität zu erkennen. Unsere Methode übertrifft mehrere Baselines bei dieser neuen Herausforderung.

In Bezug auf Handhabung mit Wahrnehmung schlagen wir ein effizientes System für die Programmierung eines Roboters zur Verwendung von menschlichen Werkzeugen vor. Wir stellen zunächst ein neuartiges und kompaktes Modell für die Verwendung von Werkzeugen, die durch ein Werkzeugspitzenmodell beschrieben werden, vor. Dann untersuchen wir die Strategie, einen Doppelgreifer-Ansatz für die Handhabung von Werkzeugen zu verwenden - motiviert durch das Fehlen von geschickten Händen bei allgemein verfügbaren Allzweckrobotern. Anschließend betten wir das Tool-Use-Learning in eine hierarchische Architektur ein und werten es auf einem Baxter-Forschungsroboter aus.

Schließlich konzentrieren wir uns bei der Kombination von Wahrnehmung, Antizipation und Handhabung auf eine Blockstapelaufgabe. Zuerst untersuchen wir, wie man Roboter anleitet, einen einzelnen Block zu platzieren, ohne dass die bestehende Struktur zusammenbricht. Wir führen einen Mechanismus ein, um die physikalische Stabilität direkt aus der visuellen Eingabe vorherzusagen und bewerten ihn zunächst auf Grundlage von synthetischen Daten und dann auf Grundlage einer

Blockstapelaufgabe aus der realen Welt. Außerdem führen wir die Stapelaufgabe mit Vorgabe ein, bei der der Agent Blöcke mit dem Ziel stapelt, einen in einem Bild gezeigten Turm zu reproduzieren. Um dies zu erreichen, erstellen wir eine synthetische Blockstapelumgebung mit Physiksimulation, in der der Agent das Stapeln von Blöcken durchgehend durch Versuch und Irrtum lernen kann, um das entsprechende physikalische Wissen explizit zu modellieren. Wir schlagen ein zielparametriertes GDQN-Modell vor, um in Bezug auf das spezifische Ziel zu planen. Wir validieren das Modell sowohl für eine Navigationsaufgabe in einer klassischen Gridworld-Umgebung als auch für die Blockstapelaufgabe.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Mario Fritz for giving me the opportunity to work under his guidance in the Scalable Learning and Perception group at the Max Planck Institute for Informatics. His vision for research and full support make it possible for me to explore across multiple fronts between computer vision, machine learning and robotics that I can never imagine before.

I would like to thank Prof. Aleš Leonardis for agreeing to serve as an external reviewer and Prof. Philipp Slusallek for being part of the thesis committee. I would also like to thank Prof. Dietrich Klakow for being the chair of the thesis committee and Dr. Michael Xuelin Huang for being the academic assistant.

Also, I would like to express my sincere gratitude to all my collaborators. Thanks to Seyedmajid Azimi for running the initial CNN classifier in the visual stability prediction work. Special thanks to Prof. Aleš Leonardis for the discussion throughout the same project and helping out at the ICRA interactive presentation session. I am also truly grateful to Prof. Jeannette Bohg for being part of the target stacking project with her constant insightful feedback.

Furthermore, I would like to thank all the members in D2 department. In particular, I would like to thank Prof. Bernt Schiele, for his lead and efforts of creating such a great research environment for the whole department. I also appreciate countless conversations and discussions with Mateusz Malinowski and Wei-Chen Chiu in the Scalable Learning and Perception group. Moreover, I would like to thank all my former office mates, Leonid Pishchulin, Hosnieh Sattar and Yang He for creating such a wonderful working atmosphere. Special thanks to our department secretary Connie Balzert for her numerous help during my stay at the institute and the assistance for the German translation of the thesis' abstract.

Last but not least, I would like to thank all my friends in Saarbrücken along the years and I can never thank my parents enough for their always understanding and never-ending support in my life.

Contents

Chapter 1

Introduction

Contents

1.1 Perspective

The field of Robotics enjoyed a significant growth over the past decade. With the rapid development in sensing, communication and computing technology, robotic system is getting closer to our daily life, from the autonomous driving cars to the deployment of surgical robots. To further advance the study in robotics and take on more challenging real world tasks, it is important to extend the capabilities of the current robotic system to act in complex and dynamic environments.

In this dissertation, we base on three important modules in robotic system, namely perception, manipulation and anticipation, bringing them together to deal with complex tasks.

Perception This refers to the process that transforms the raw sensory information into the internal representation. It is a major topic in general pattern recognition: in speech recognition, an audio signal is converted into corresponding text; in computer vision, an image can be labeled by the categories of objects inside it, or regions of distinct semantic concepts.

Manipulation This refers to the operation that interacts with objects to achieve a certain goal. It is a major topic in robotic application: from pick-and-place to automatic assembly in production line.

Anticipation This refers to the mechanism that predicts the future consequence. It can take different forms and develop into various interesting problems (c.f. section 2.1.2).

These modules are closely connected with each other, and our work mainly explores three ways of integrations with one another as shown in Figure 1.1:

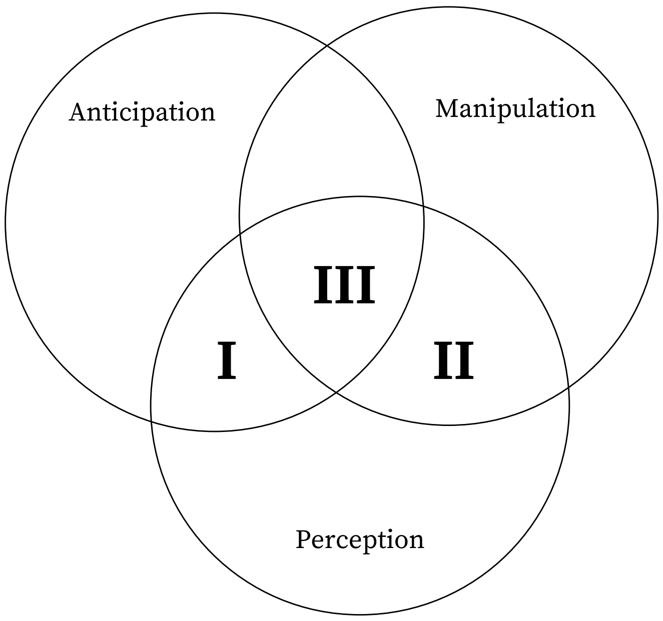


Figure 1.1: Our work explores different ways of integrations with perception, anticipation and manipulation modules in intelligent systems.

I. From Perception to Anticipation There is a close tie between perception and anticipation. Perception generally provides a description of what the current state is or in other words, an understanding of the current state. This often serves as a good base for anticipation to extrapolate to the future. In return, anticipation can strengthen the current perception and decision-making. There are a lot of robotic applications that can benefit from the anticipation mechanism. In autonomous driving, observing the surrounding traffic flow can hint on nearby vehicles' incoming behavior, the autonomous vehicle can hence take precautions of others' misbehaviors. In assisted living, recognizing personnel's current activity can suggest his or her possible follow up, the service robot can thus provide necessary assistance.

The key challenge along this direction is how to relate the future with the presence. Depending on specific problems, the model can take different forms. For example, to predict the trajectory in the near future, it can assume certain consistency between consecutive time frames; to predict the physical consequence, it can assume some form of physical knowledge. In section 2.1.2, we will see more of related examples.

Our work in chapter 3 tackles the recognition of ongoing activity from videos, i.e. given the observation so far, anticipating the activity class for the whole sequence.

From assisted living and many other applications such as surveillance system, crime prevention and human robot interaction can make use of such mechanism to infer which activity is currently going on and hence being able to behave proactive or context-aware.

II. From Perception to Manipulation In robotics literature, it is common to couple planning with perception. For instance, in classic pick-up task, the robot needs to first detect the object and then figures out where and how to reach and pick it up. In a modern unmanned aerial vehicle, various sensors provide it with information about local environment to navigate without collisions.

Yet when it comes down to the specific domain of robotic manipulation, such applications have not gone too far from the aforementioned pick-up and some other low-level primitives like grasping. There are various reasons for this situation such as the disagreement between the information demanded for planning and the noisy and insufficient input from the perception module.

Another important reason is that in general, it is nontrivial to formulate a planning model with perception for a complex manipulation which often includes multiple steps of operations. Our work in chapter 5 explores the possibility along this line and proposes a framework for the general tool using process. With further introduction of the dynamic movement primitive method, the robot can learn to use tool with one or a few expert demonstrations.

III. From Perception over Anticipation to Manipulation Finally, to achieve higher level of intelligence for a robotic system, it is necessary to pursue a more advanced integration of both perception and anticipation into manipulation.

In chapter 7 and chapter 8, we focus on a block stacking task. In the former, we attempt the explicit learning of intuitive physics into anticipation to guide stacking a single block into the scene without collapsing the existing structure; in the latter, we explore the end-to-end learning of stacking multiple blocks to achieve a specified target structure.

1.2 Contributions

This thesis contributes along the three major areas that we have just discussed in the previous section with the following major works:

Recognition of Ongoing Complex Activities This work lies at the intersection of perception and anticipation. It takes place in the domain of video analysis and aims to anticipate activity class from video data. In this work:

- We propose the first dataset of complex activities that have a substantial time span and are organized in a semantic hierarchy.

- We introduce a new representation that models the uncertainty over different time frames across different levels of the semantic hierarchy.
- We study how prediction can be performed from partial observation of ongoing activities with a semantic hierarchy back-off strategy.
- We propose a performance measure that evaluates the gain in specificity over time as more observation becomes available.
- Our new method outperforms several baselines as well as techniques that we adopt from the literature on accuracy specificity trade-offs for object recognition.

This work was presented at WACV (Li and Fritz, 2016).

Teaching Robots the Use of Human Tools This work integrates perception and manipulation. Here we want to find a faster way to program robot in a manipulation task within a perception-manipulation loop. Specially, we focus on the task of tool using. In this work:

- We present a novel and compact model for using tools that can be described by a tip model. With this model, we can decompose the tool using procedures into manipulation primitives.
- We explore a strategy of utilizing a dual-gripper approach for manipulating tools – motivated by the absence of dexterous hands on today's most widely deployed general purpose robots.
- We formulate a hierarchical architecture to embed the tool use in a learning from demonstration framework. At a high-level, we learn temporal orders for dual-arm coordination and at lower-level, we learn DMPs for manipulation primitives.
- The approach is evaluated on a Baxter research robot with learning and operation of three human tools, including an electric tacker, an electric drill and a hot-glue pen.

This work was presented at Humanoids (Li and Fritz, 2015).

Visual Stability Prediction for Robotic Manipulation This work combines perception, anticipation and manipulation. Here we search for a mechanism to predict physical stability directly from visual input, it can then be used in the block stacking manipulation. In this work:

- In contrast to existing approaches, we bypass explicit 3D representations and physical simulation and learn a model for visual stability prediction from data.

- We evaluate our model on a range of conditions including variations in number of blocks, size of blocks and 3D structure of the overall tower, reflecting the challenges of inference with growing complexity of the structure.
- We conduct a human subject study on a subset of our synthetic data and show that our model achieves comparable or even better results than humans in the same setting.
- We investigate the discriminative image regions found by the model and spot a correlation between such regions and the initial collapse area in the structure, providing additional insights into our obtained model.
- We apply our approach to a block stacking setting and guide a robot for the block placement with the stability predictions of the future states.

This work started as a tech report (Li *et al.*, 2016) focusing on visual stability prediction. It was later extended with the content of robotic manipulation and accepted as an oral presentation at 2016 NIPS workshop on Intuitive Physics. The final paper is published at ICRA (Li *et al.*, 2017b).

Acquiring Target Stacking Skills This work extends the previous one by tackling the task of stacking multiple blocks. Anticipation for a single stacking step is no longer sufficient in such task, instead it requires more a complex planning while still encoding the essential knowledge and additional constraints, such as physics and the specified target shape in our experiment. In this work:

- We create a synthetic block stacking environment with physics simulation in which the agent can learn block stacking end-to-end through trial and error, bypassing an explicit modeling of the corresponding physics knowledge.
- We introduce a target stacking task where the agent stacks blocks to reproduce a tower shown in an image. The task presents a distinct type of challenge requiring the agent to reach a given goal state that may be different for every new trial.
- We propose a goal-parametrized GDQN model to plan with respect to the specific goal, allowing better generalization across different goals.
- We validate the model on both a navigation task and the target stacking task itself. Our proposed model shows good performances on both tasks.

This work was presented in a tech report (Li *et al.*, 2017a).

1.3 Outline

This dissertation is composed of three major parts:

Part I chapter 3. In this part, we explore anticipations from the perception and specifically we propose a learning framework to anticipate in video data.

Part II chapters 4 and 5. In this part, we integrate the perception module into the manipulation. The first system builds its actions directly on the visual information for a picking task in an automatic warehouse scenario whereas the latter plans further with respect to experts' demonstrations.

Part III chapters 6 to 8. This part attempts the integration of both the perception and the anticipation into the manipulation. We specifically focus on a block stacking scenario and employ both synthetic and real-world experiments.

A brief summary of the content in each chapter is as follows:

Chapter 2: Related Work. We review related works and provide the readers with a background knowledge to better understand the content in later chapters.

Chapter 3: Recognition of Ongoing Complex Activities. Observing an ongoing complex activity, how can the machine effectively predict its activity class before seeing the whole sequence? In this work, we present a framework to predict activities over a hierarchical label space, making trade-off between the accuracy over the specificity.

Chapter 4: Participation in Amazon Picking Challenge We describe our system participated in the first Amazon Picking Challenge and introduce the robotics platform used in later chapters.

Chapter 5: Teaching Robots the Use of Human Tools. Conventional industrial robots require experts to carefully script each step of a fixed procedure, here we show an alternative way to this paradigm with learning from demonstration. By one or a few expert's demonstrations, we successfully taught the robot to use a few human tools.

Chapter 6: Simulation Environment We discuss the details on both design and implementation for the environments used in the next two chapters.

Chapter 7: Visual Stability Prediction for Robotic Manipulation. Given a physical structure, human has a sense of feeling if it is stable. This is an example of intuitive physics. In this work, we show it is possible to learn a model to mimic such capability to guide the robot for a one-block stacking task.

Chapter 8: Acquiring Target Stacking Skills. Going beyond the one-block stacking task in the previous chapter, we attempt to build an agent to stack multiple blocks to a specified target shape. The task motivates us to extend the the DQN with goal-parameterized learning.

Chapter 9: Conclusions and future perspectives. This chapter concludes the thesis and offers insights of some interesting directions for future research.

Chapter 2

Related Work

Contents

In this chapter, we provide a more general context for the work in this thesis. The contents are arranged in four topics corresponding to the later chapters.

In section 2.1, we start with the research trend of pursuing deeper understanding of video data and then focus on one particular direction, i.e. the anticipation model. This directly connects to the idea in chapter 3 on Recognition of Ongoing Complex Activities, where we anticipate activity class from videos with only partial observation.

In section 2.2, we discuss the general paradigm of learning from demonstration and its applications. Further we review the framework of the dynamic movement primitives and supply readers with more detailed insights into it. All of these help readers to better understand our work in chapter 5 on Teaching Robots the Use of

Human Tools.

In section 2.3, we revisit the idea of intuitive physics and sketch a broader landscape across the research in cognitive science and development psychology. This connects to the work in chapter 7 where we apply the similar idea to learn physics from synthetic data.

In section 2.4, we go through some basics of reinforcement learning and then examine in more details on the recent developed deep Q-Network. This supply readers with a better background knowledge to understand our work in chapter 8 on Acquiring Target Stacking Skills.

2.1 Anticipation from Video

From recognition of simple activities in short video clips to anticipating long and complex activities, research in video analysis has made significant progress over the past decade. In this section, we first give a brief review over the recent progress in related areas on video. Then we focus on the sub-area of anticipation from video data which closely connects to our work of recognition of ongoing activities described in chapter 3. Afterwards we also discuss some developments in video data representation, in particular providing more detailed information about the trajectory feature used in our work described in chapter 3.

2.1.1 Deeper Understanding of Videos

A picture is worth a thousand words, and a minute long video-clip easily contains more than a thousand pictures [1]. Naturally, videos can provide significant more dynamic contexts than static images. This in return also presents some unique challenges on the analysis of videos.

Identifying and localizing objects can provide a basic level of information about the video. Established techniques in computer vision such as semantic segmentation and object detection can be applied to address this task. However, it is often not sufficient to describe a video by the objects alone. This leads to another unique element in videos, the **motion**. The objects can actively move in the video frames and interact with each other. Techniques such as optical flow and tracking can partially address issues from this perspective. Yet, even with the addition of motion information, it still seems inadequate to express the video as the information obtained so far only offers a plain summary about the video. Hence the third essential element is needed as the **event**. An event is a succinct and meaningful abstraction of the video, which can embody in the form of simply an event or an activity label in the classification task, a temporal localization of event's occurrence in the event detection or a compatible textual description as in the video captioning task. In a sense, with

[1]With a simple calculation in a common video frame rate of $24FPS$, a minute long video contains $24 \times 60 = 1440$ image frames.

the introduction of event, we achieve a deeper understanding of videos as shown in Figure 2.1.

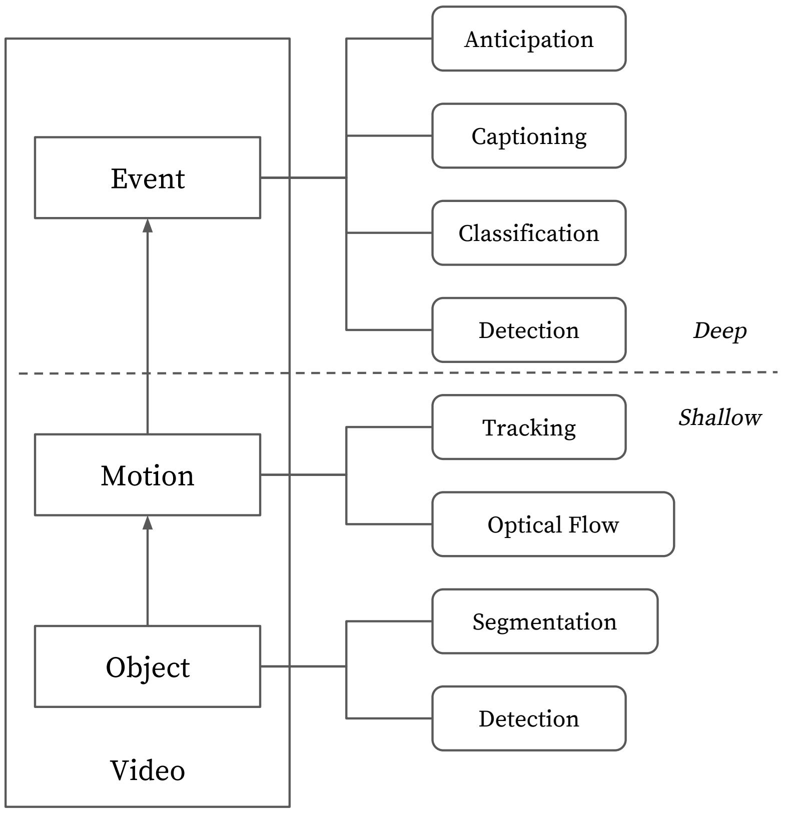


Figure 2.1: A hierarchy of video analysis.

Along the years, there are several trends in pursuing a deeper understanding of videos. For starters, the studied videos have become more and more complex. Early datasets in the literature including the KTH dataset (Schuldt *et al.*, 2004) and the Weizmann dataset (Blank *et al.*, 2005) feature simple activities like walking and jumping. Later datasets such as the UCF Sports include activities like golf swinging and the MPII cooking dataset (Rohrbach *et al.*, 2012) on making different dishes. This can be attributed to the advancement of research in video representation and machine learning. In the section 2.1.3, we will discuss more specifically on the development of research in video representation.

The second trend is that datasets are collected from more and more diverse sources. There are datasets from movies such as the Hollywood2 Marszalek *et al.* (2009), surveillance footage like the VIRAT (Oh *et al.*, 2011) and daily living recordings like the TUM Kitchen (Tenorth *et al.*, 2009). We also see ego-centric datasets (Fathi *et al.*, 2011; Stein and McKenna, 2013). With the growing machinery to analyze video data, researchers are inspired to apply different techniques to solving problems in more practical scenarios.

Another development from a different perspective is the rise of anticipation model. While most of the aforementioned works focus on elaborating the facts in the observation so far, researchers now become more interested in reflecting the knowledge from the observation and applying it to anticipate the unseen future. We will briefly review some recent works along this line in the following subsection.

2.1.2 Anticipation from Visual Data

An anticipation model generally makes certain predictions about the future based on the observation so far. The prediction can take different forms and develops into various interesting specific problems:

The anticipated object can be qualitative (**qualitative anticipation**), such as Ryoo (2011) and Li and Fritz (2016) to predict activity class based on a partial observation as shown in Figure 2.2. If we further include images into the observation, such as the models proposed in Lerer *et al.* (2016) and Li *et al.* (2016), it can anticipate the possible physical outcomes.

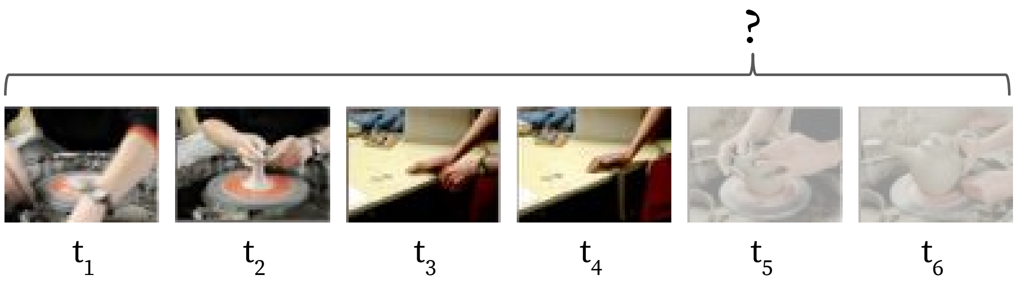


Figure 2.2: Example of qualitative anticipation: recognition of ongoing activity (Li and Fritz, 2016).

The anticipated object can also be quantitative (**quantitative anticipation**, such as the model in Hoai and De la Torre (2012) to predict pedestrian's walking path from the surveillance video footage, or the one in Mottaghi *et al.* (2016), Fragkiadaki *et al.* (2016) and Bhattacharyya *et al.* (2018) to predict the objects' path under physical law as shown in Figure 2.3.

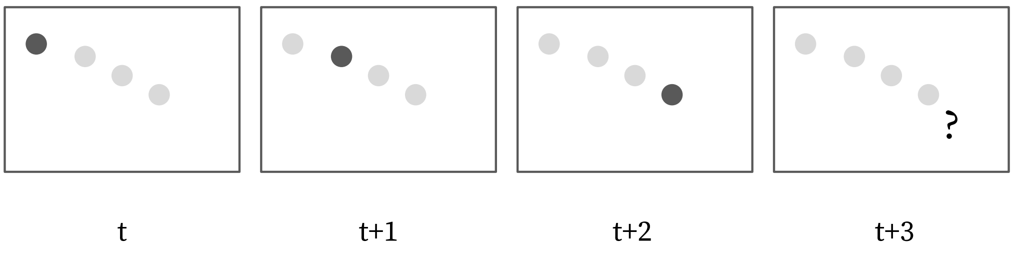


Figure 2.3: Example of quantitative anticipation: predicting the objects' trajectory

A quite a few aforementioned works on anticipation model are built on the physical understanding from observation which related to another important topic in this thesis, namely the *intuitive physics*. We will discuss more on this in section 2.3.

2.1.3 Video Representation

Significant progress has been made for video representation in recent years, in particular with the rise of deep learning methods. In earlier research, a common way to do feature representation is based on image descriptors located at spatio-temporal interest points, such as Laptev (2005).

A later work (Wang *et al.*, 2011) has advanced the idea to track those spatio-temporal key points across multiple image frames and accumulate bag-of-words descriptors aligned with these trajectories. More specifically, the key points are first densely sampled on a image pyramid across multiple scales, as shown in Figure 2.4, based on the sampled key points, trajectories are tracked separately for each spatial scale with respect to the frame-wise optical flow. Local descriptors including the histograms of oriented gradients (HOG) (Dalal and Triggs, 2005), histograms of optical flow (HOF) and motion boundary histogram (MBF) (Dalal *et al.*, 2006) located at the key points are accumulated to represent the whole video stream. The proposed representation (Wang *et al.*, 2011) has shown state-of-art performance by the time of its publication.

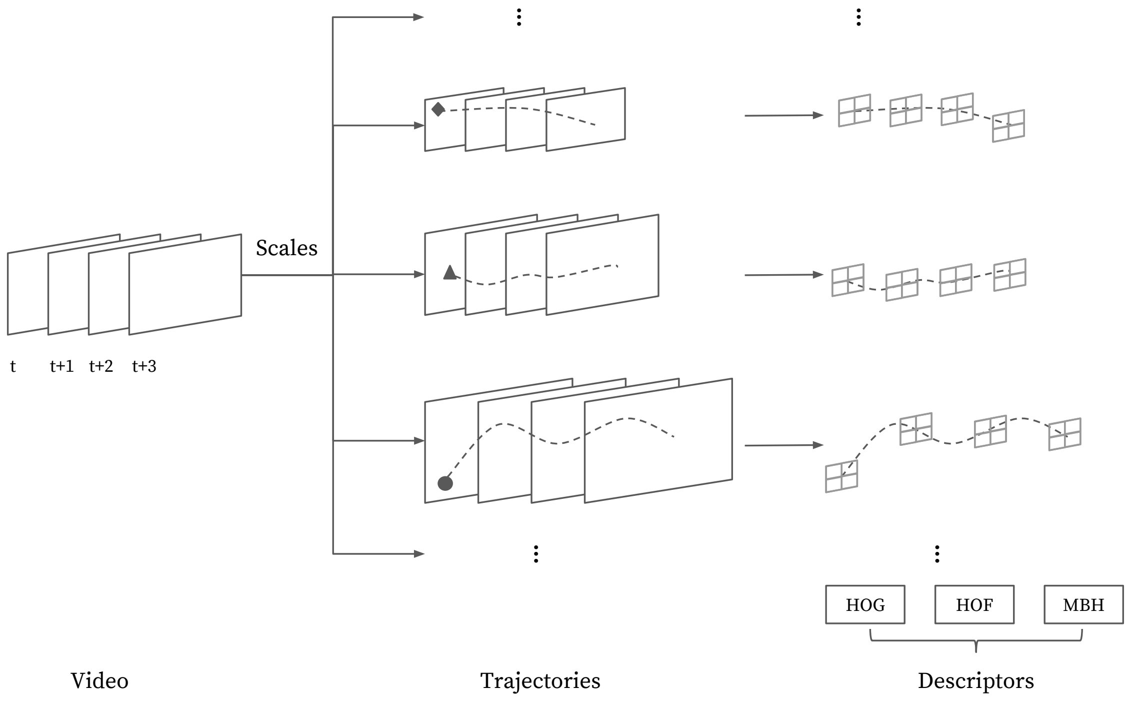


Figure 2.4: The architecture for the improved trajectory feature for video data.

Interestingly, at the similar time frame, with the increasing popularity of deep learning, in particular with some initial success of deep neural network in image domain (Krizhevsky *et al.*, 2012), researchers started to explore its application in video domain. A closely related work following the dense trajectory feature by Wang *et al.* (2015) presented a way to interchange the hand-crafted descriptors with convolutional neural network. This is a hybrid approach with the trajectories still coming from the optical flow.

A later work (Yue-Hei Ng *et al.*, 2015) proposed an end-to-end descriptor based completely on deep neural networks as shown in Figure 2.5, where the frame-wise feature comes from the convolutional neural network and the overall descriptor is formed from a long-short term memory network (Hochreiter and Schmidhuber, 1997) over the frame feature.

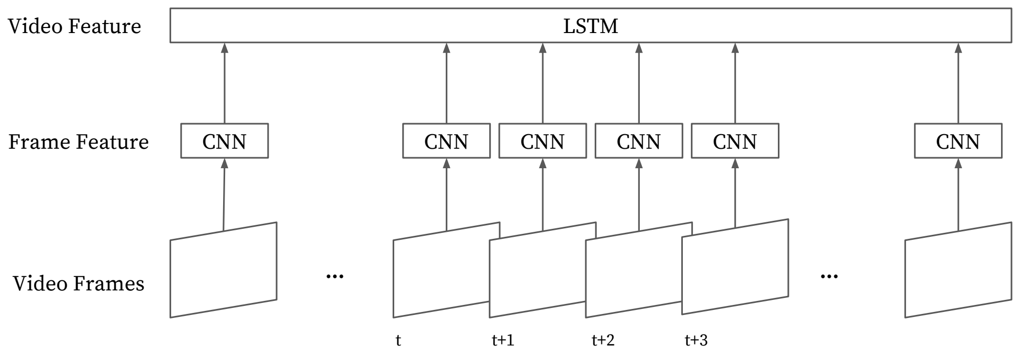


Figure 2.5: The architecture for the deep learning based representation for video data.

2.2 Learning from Demonstration

In this section, we recap the learning from demonstration paradigm which is used later in chapter 5. Robot learning from demonstration enables the robot to automatically derive control schemes on a task from the observation of an expert's performance on it. There are a variety of methods in the learning from demonstration literature, a comprehensive survey on the topic in robotics can be found in Argall *et al.* (2009) and a general overview article is available on-line by Billard and Grollman (2013). Here we first give a brief overview of some basics in learning from demonstration and then revisit the Dynamic Movement Primitives (DMP), a particular framework of learning from demonstration used chapter 5. Afterwards, we cover some related example applications.

2.2.1 Basic Concepts

With the introduction of the **policy**, we can more formally formulate the idea of control scheme for a robot. A policy π is defined as a mapping from state s to a corresponding action a: $\pi : s \rightarrow a$. The policy is one of the most important concepts in the literature of robot control. As we shall see later in the related work, it is also a key ingredient in the reinforcement learning. A traditional way to obtain the policy for the robot control, e.g. to program a robot arm, is to analytically decompose and manually code the desired task execution. However this approach is in general very demanding and the obtained the policy is often not easy to adapt to novel cases. To tackle these issues, learning from demonstration builds its model for policy learning from example executions. More specifically, as shown in Figure 2.6, given the same environment of the task, a **teacher** first performs sample executions as the **demonstrations**, in the form of a series of interactions observing the state s and act a. The demonstrations are provided to the robot to derive its own policy π, under which given a state s' it selects an action a'.

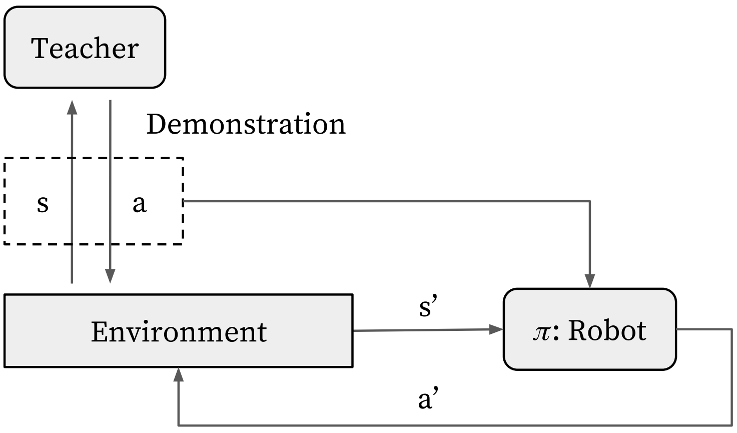


Figure 2.6: Learning from demonstration paradigm.

From one hand, there are different ways to perform demonstrations, which in return decide how the demonstrator's policy can differ from the ones used by the robot. For example, in the most straightforward case, the demonstration can be done with the robot itself through teleoperations by human teachers with the observation of its own sensor. The policy applied in demonstration can be very similar to the one at test as both subject to the comparable constraints, such as the mechanical constraints from the robot. The other extreme case is that the demonstration is done by a drastically different body and provided to the robot through external observations, e.g. the robot observes a human demonstrator to perform sample executions. The policy applied in the demonstration can be significantly different from what will be used at test time, considering the operations are achieved under different mechanical structures.

From the other hand, the learned policy itself should be rich enough so that it can be applied to novel cases other than only the ones seen during demonstration, namely the policy should have the capacity for **generalization**. There are various approaches to allow generalizations, including the Dynamic Movement Primitive (Ijspeert *et al.*, 2002), which we will discuss in more details in the following subsection.

2.2.2 Dynamic Movement Primitives

The name of dynamic movement primitives (DMPs) is inspired by the biological concept of motor primitives designed to be used and modulated in real time for generating complex movements (Ijspeert *et al.*, 2013). In particular, the term of "dynamic" comes from the use of nonlinear dynamical system to formalize the movement primitives.

Formally speaking, a **dynamical system** defines a rule for time evolution on a state space. The **state space** describes the state at any instant, and the **dynamical rule** specifies the immediate future of all state variables given only the present values of those same state variables (Meiss, 2007).

In DMPs, the state space is formalized with a **kinematic representation**, i.e. location, velocity and acceleration. The dynamical system encodes the desired kinematic motor behaviors with the trajectories in the state space as shown in Figure 2.7, and the kinematic behaviors are later converted into motor commands (such as the torque from the robot's joint) with some inverse dynamic controller.

The dynamical rule for DMPs is intended to generate the motion plan with respect to a specified goal state where the planned trajectory resembles the ones seen in the demonstrations. This naturally leads to two criteria for the dynamical rule:

1. Reach a specified goal state.

2. The generated path is adjustable to imitate the ones in the demonstrations.

To meet criteria (1) a base point attractive system is formulated based on a damped spring model to converge to the goal state:

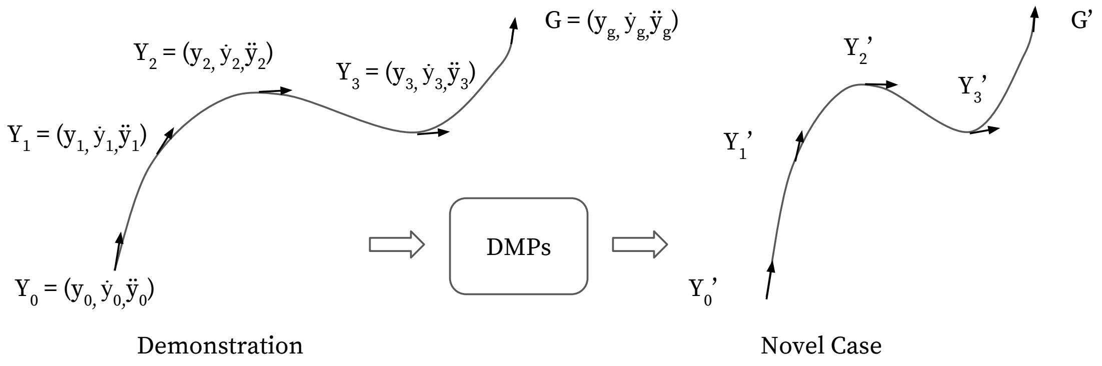


Figure 2.7: Movement generation with DMPs.

$$\tau \ddot{y} = \alpha_z(\beta_z(g - y) - \dot{y} + f$$

or more commonly as written in first-order notation in the related work:

$$\begin{aligned}\tau \dot{z} &= \alpha_z(\beta_z(g - y) - z + f \\ \tau \dot{y} &= z\end{aligned}$$

where τ is a global time scaling parameter and α_z and β_z are positive constants. g is the known goal state, also known as the **point attractor**. f is the forcing term and the component to drive the system to imitate paths seen in the demonstration for criteria (2). Note the particular form of first-order notation was taken from a more recent publication (Ijspeert *et al.*, 2013), as pointed out in it, earlier works took a slightly different form and turned out to be analytically less favorable. Further the **nonlinearity** is enforced by the forcing term f representing arbitrary nonlinear functions as a normalized linear combination of basis functions:

$$f(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{i=1}^{N} \Psi_i(t)}$$

where Ψ_i are the fixed basis functions and w_i are the adjustable weights. This formulation still renders the system of explicit time dependence, also known as **nonautonomous**, which does not allow straightforward coupling with other dynamical systems and the coordination of multiple DOF in one dynamical system as discussed in Ijspeert *et al.* (2013). The solution is to reformulate it by introducing the **canonical system**:

$$\tau \dot{x} = -\alpha_x x$$

The $x = x(t)$ acts as a phase variable, given some arbitrarily chosen initial state x_0 such as $x_0 = 1$, the state x converges monotonically to zero: $x = 1$ indicates the start of the trajectory and $x \to 0$ indicates the goal state g has been achieved. Then the forcing term f and corresponding basis function Ψ_i are defined as:

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0)$$
$$\Psi_i(x) = exp(-\frac{1}{2\sigma_i^2}(x - c_i)^2)$$

where σ_i and c_i are constants for the basis functions. Comparing the new formulation with the previous one, we note:

- The resulted system no longer depends explicitly on time but on the phase variable x and hence becomes **autonomous** as desired.
- Additional term of $x(g-y_0)$ is introduced to modulate the forcing term with the phase variable x to control the amplitude (as $x \to 0$ to reach the goal state, the force also diminishes to 0) and the movement amplitude $g-y_0$ (assuming $g \neq y_0$ preserves useful **spatial scaling properties**, i.e. the generated trajectory is scaled to it. Reader can find more detailed discussion in Ijspeert *et al.* (2013).

The eventual learning process is formulated in a supervised learning framework. Given one or multiple desired trajectories in terms of the kinematics representation as $(y_demo(t), \dot{y}_{demo}(t), \ddot{y}_{demo}(t))$ where $t \in [0, ..., P]$ with:

$$g = y_{demo}(t = P)$$
$$y_0 = y_{demo}(t = 0)$$
$$\tau = P$$

The weights parameters w_i are learned with **locally weighted regression** (Schaal and Atkeson, 1998) by minimizing its loss with respect to the forcing term f for each kernel function Ψ_i:

$$L_i = \sum_{t=0}^{P} \Psi_i(t)(f_{target}(t) - w_i x(t)(g - y_0))^2$$

where f_{target} is obtained from the demonstrated trajectory:

$$f_{target} = \tau^2 \ddot{y} - \alpha_z(\beta_z(g - y_{demo}) - \tau \dot{y}_{demo})$$

To generalize to novel cases, e.g. a different goal state, one only needs to plug in the new goal state g for the movement amplitude $g - y_0$ in the obtained dynamic system and can then derive motion plan in kinematic representation. As discussed in Ijspeert *et al.* (2013), the weights parameters w_i and the other constants α_z,β_z define qualitatively a particular behavior, i.e. movement primitive, hence are kept constant for the very kind of behavior. There are many more details, such as the stability properties of the system and an alternative formulation for rhythmic pattern generator which we have left out here for the sake of space, readers can find detailed discussion in the aforementioned paper (Ijspeert *et al.*, 2013).

2.2.3 Applications

There are a wide range of applications of learning from demonstration in robotics. As for DMPs, early works taught a 30 degrees-of-freedom humanoid robot for performing tennis swings and drumming movements(Ijspeert *et al.*, 2002). Extensions with coupling of additional sensor reading has been applied for grasping. Our work of teaching the robot to using human tools also built up on the DMPs which we will go into details in chapter 5.

As the learning from demonstration process often involves the participation of human teacher, it naturally become an interesting topic in the field of human-robot interaction. One perspective is that human can be more actively in the learning process, for example in **active imitation learning** (Shon *et al.*, 2007), the robot can request helps from the teacher.

Another initiative from learning from demonstration is that the demonstration can be used in different ways other than the aforementioned supervised learning in the DMPS, for instance, when combining with reinforcement learning, it can further guide the exploration in search for potentially better policy than the ones shown in demonstration. We have seen applications, such as aerobatic helicopter flight with **apprenticeship learning** as described in Abbeel (2008).

2.3 Intuitive Physics

In this section, we introduce the research in intuitive physics. **Intuitive Physics** generally refers to the commonsense knowledge for human to understand physical environment and interaction. This is in contrast with the thorough and rigorous physics study, such as the **Newtonian Physics**. Yet intuitive physics works well enough for most situations in our daily life. It is intriguing what exactly represents such a knowledge, where it comes from and how we can possibly build it into artificial intelligent systems. Along the years, research have been conducted from many different perspectives across psychology, cognitive science and artificial intelligence. In particular, with the recent advance of deep learning, new computational model has been proposed to do physics inference from direct visual observations. We will continue the discussion of such approach later in our visual stability prediction work in 7.

Here we briefly review relevant research on this topic in a more broader context. In particular, the research in development psychology how infant acquire knowledge of intuitive physics is one of the main inspirations for our work in 7. Readers can also find a recent review on intuitive physics by Kubricht *et al.* (2017).

2.3.1 Origin

The concept of intuitive physics aims to describe the knowledge enables human to understand physical environment and interact accordingly. In particular, the "intuitive" part emphasize the knowledge is a part of commonsense to normal people not reliant on specialized training in physics. Intuitive physics is ubiquitous in guiding humans' action in daily life, such as where to put a cup stably and how to catch a ball.

Historically, Hayes first used the term **Navie Physics** to describe his approach to developing a "large-scale formalism" of commonsense knowledge about the world (Hayes *et al.*, 1978), pioneering the study to model the intuitive physics. The use of the word "naive" reveals its nature including **commonsense knowledge** that is normally taken for granted by "the man in the street" other than specialized-trained physicists. Through his seminal "Naive Physics Manifesto" (Hayes *et al.*, 1978) and its revision in the "Second Naive Physics Manifesto" (Hayes, 1985), he argued such knowledge should:

- attempt reasonable completeness - describing a significant portion of the way how human understand the world rather than just small pieces
- include both commonsense normally taken for granted in formal physics and elements outside what is considered to be the field of physics

Hayes' work pioneered the study to formally model human's knowledge of intuitive physics and influence later work along the line of research. A succinct discussion about this can be found in the thesis' chapter by Blackwell (1988).

2.3.2 Research in Cognitive Science

As pointed out by Hayes, the intuitive physics can also include elements outside what is considered to be the field of physics and may hence also choose to describe the phenomena in a way that is familiar to "the man in the street", but would not be considered appropriate to a physicist (Blackwell, 1988). In other words, it can sometimes deviate from physical reality due to certain cognition bias. Researchers in cognitive science have addressed this issue, i.e. the **misconception** in intuitive physics. A notable such example is the "impetus" theory of motion discussed by McCloskey (1983). As shown in Figure 2.8, where the object is released while the car is moving, the subjects are asked to predict which trajectory the object will follow. 49% of people in the experiment considered the trajectories B and 6% considered C are correct, and only 45% picked the the actual right trajectory A. The large portion of misconception about trajectory B resembles the pre-Newtonian theory of impetus that motion must have a cause – as long as the object is released there is nothing keeps it moves and hence trajectory B is considered correct.

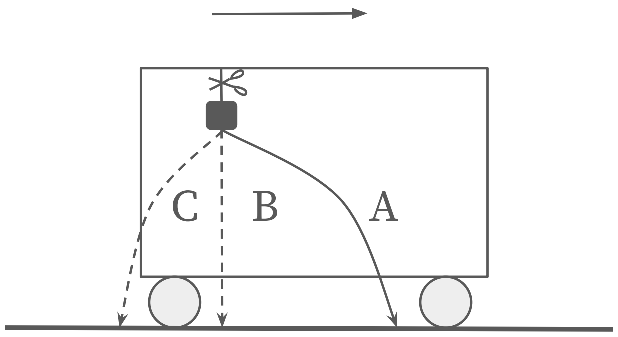


Figure 2.8: Example of misconception about motion. The object is released while the car is moving, the subjects are asked to predict which trajectory the object will follow. The solid line A is the actual trajectory for the object, however according to the study by McCloskey (1983), a significant portion of people consider B and C as the answer.

A different line of research is to explore possible models to explain the behaviors from intuitive physics, accounting for not only the success part but also aforementioned failure cases. Specifically, **computational models** focusing on the mathematical formulation have becoming increasingly popular. A symposium of computation model of intuitive physics has been held to discuss recent progress from several different perspectives, an overview can be found in Battaglia *et al.* (2012).

One recent idea is that human have an internal probabilistic physics engine, also known as the **Intuitive Physics Engine** (IPE) to support physical inference (Battaglia *et al.*, 2013). As shown in Figure 2.9, given a scene, the IPE first generates its internal state based on the scene's true state from a certain view. Note a Gaussian noise is injected here to represent the uncertainty during the generation process.

Afterwards, a simulation is carried out on the internal state and arrives at its outcome. As the scene can be observed from multiple views, each view can lead to slightly different internal states, the final judgement for the scene's physics can be an aggregation based on the simulated outcomes from theses corresponding internal states. In the experiments, along with the predictions from IPE, human subjects were tested with same set of different blocks scenes and required to predict, such as if a specific structure will fall and in which direction it will fall. Results have been found to show high correlation between IPE's and people's average judgments.

Interestingly, the authors further explored the possibility that people's judgments do not involve any mental simulation. So the alternative explanation for people's physical scene understanding is that they use the model-free methods that depend heavily on their experienced interactions with the world. In a sense, it is the widely adopted data-driven approach in computer vision. In experiments, they used a multivariate regression based on a set of predefined geometric features, such as tower's height and height of the tower's center of mass to represent the alternative model. They found it consistently worse at predicting people's judgments than the IPE model and hence concluded that it is not viable as a general-purpose alternative to the IPE model. Additionally, they gave several possible reasons why a purely model-free account is inefficient, including:

- it has to be compact so that it can be learned from finite experience.
- it has to be rich enough so that it can generalize to novel scenes.

The analysis actually inspired us to start our research in visual stability prediction described in chapter 7. For starters, human may actually come across more data than "finite", as we will describe in the following subsection, research in development psychology sheds some light on how the infant can develop its understanding on physics at an early stage through the interaction and the observation. Moreover, other than the predefined geometric features, there can be much richer representation for the visual information to allow generalization to novel instances, in particular with the rise of deep learning methods, learner has been shown to perform very well on large-scale real world image dataset (Krizhevsky *et al.*, 2012). Though we are not arguing the data-driven approach can explain better about human's judgement, we would like definitely to show the high feasibility of it.

2.3.3 Research in Development Psychology

In contrast to seeking what makes the mechanism for intuitive physics, related research in development psychology focuses more on answering *if* and *how* human develop intuitive physics. The chapter by Baillargeon (2002) has summarized the progress on the very topic by the time of its publication. First and foremost, infants do acquire rules about physical event. The rules specify for them what are the likely outcomes of events and hence can be understood as the intuitive physics. This provides a positive answer to the if question. Infants' rules become more sophisticated

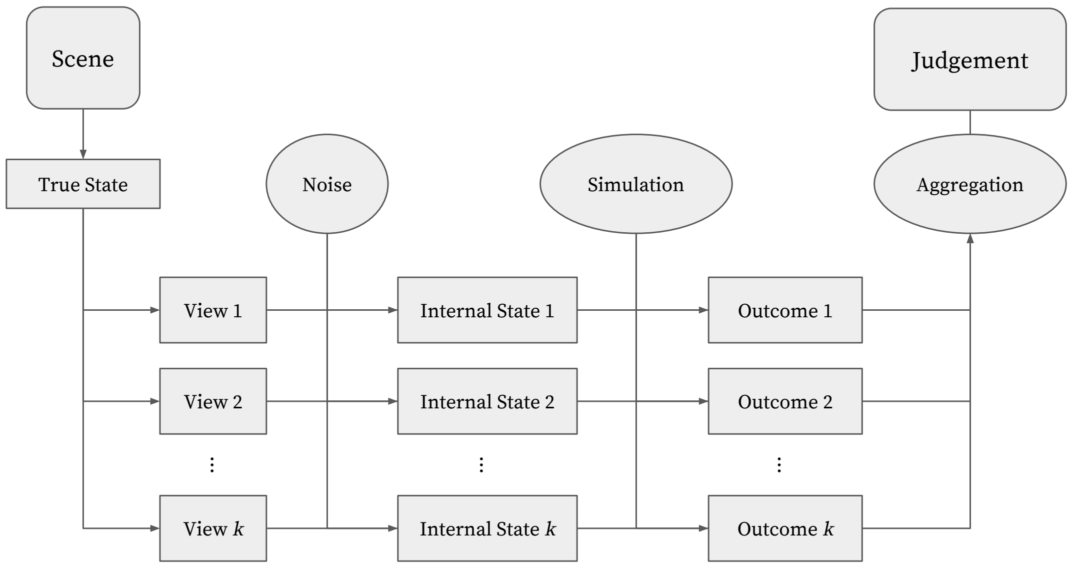


Figure 2.9: The architecture of intuitive physics engine proposed by Battaglia *et al.* (2013).

over time. The rules about different events all seem to develop according to the same general pattern, i.e. first forming an initial concept on all-or-none distinction and then gradually identifying a sequence of variables to refine the understanding. This provides leads on the how question.

These findings are based on a series of experiments across multiple different physical events, such as the **support event** where a box was held in one of several positions relative to the platform, and the infants judged if the box remains stable when released as shown in Figure 2.10, the **collision event** where objects with different weights roll down from the ramp and collide with the object at the bottom, infants reasoned about heavier and lighter objects as shown in Figure 2.11. The key to identify the learning or development process is through the observation of violation-of-expectation when infants watch these events. More specifically, as the intuitive physics specify for them the likely outcome of events (*expectation*), when faced with events inconsistent with their expectation, they typically are surprised or puzzled and most often look reliably longer at the *violation*. In the initial stages of learning, infants often err in predicting the outcome of events suggesting their understanding about physical events is likely rather primitive or incomplete. However, that is not the end of story, the physical knowledge develops over time. Take the support event as an example, in the first few months, infants only formed an initial concept of support centered on a simple contact/no-contact distinction, namely they expect the box to remain stable if released in contact with the platform, and to fall otherwise. In the following months, infants identify a sequence of variables to refine their initial concept: from the type of contact between the box and the platform to the amount of contact between the box and the platform till finally the proportional distribution of the box (Baillargeon, 2002).

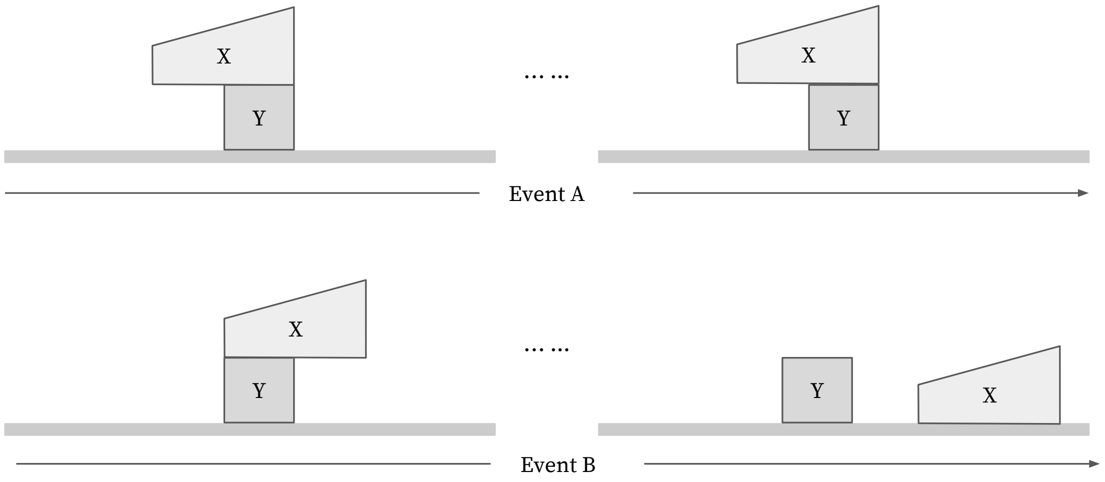


Figure 2.10: Support event.

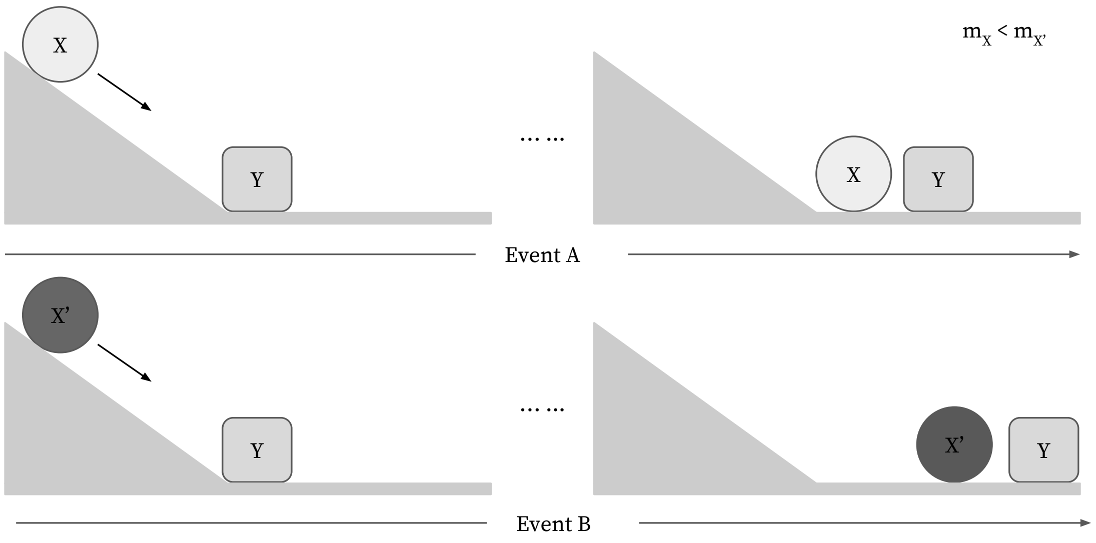


Figure 2.11: Collision event.

2.4 Reinforcement Learning

In this section, we give a brief overview on reinforcement learning which we used later to formulate learning the framework for acquiring stacking skill of blocks. We start with introducing a generic agent-environment interface for reinforcement learning problems, then dive into the widely-used Markov Decision Process (MDP) formulation. Afterwards, we briefly look back into some classical methods in solving reinforcement learning problems under the MDP formulation and how some of them connects to more recent advanced techniques in the related research areas, in particular the Deep Q-Network (DQN), which we extended later in chapter 8. The readers can find more systematic and detailed description on reinforcement learning in the monograph by Sutton and Barto (1998). A newer version of the book draft is kindly provided on-line [2] by the author with updated contents and discussion of recent advancement on the topic.

2.4.1 Basic Elements in Reinforcement Learning

The agent-environment interface is a generic way to describe reinforcement learning paradigm as shown in Figure 2.12. The agent observes the state from the environment it acts on, performs action and receives **reward** signal, the environment then transit into a new state, and the agent continues the process. If the process naturally breaks into subsequences (episodes), each terminates after finite steps, it is usually referred to as being **Episodic**. Otherwise, if it goes on continually without limit, it is then referred to as being **Continuous**. In this thesis, we focus on the episodic settings.

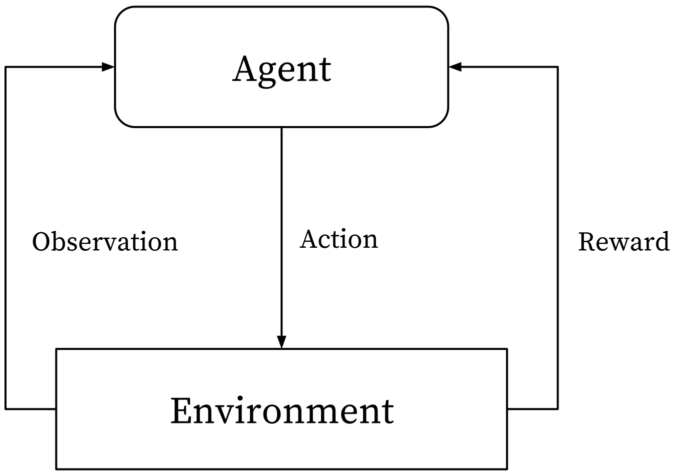


Figure 2.12: Agent Environment Interface for Reinforcement Learning.

The strategy for the agent on how to act at a given observation is generally referred to as the **policy** π. The policy can be either deterministic or stochastic.

[2]http://incompleteideas.net/book/the-book.html

The objective of reinforcement learning is to seek an optimal policy for the agent to maximize the overall rewards (this will become more formal in the formulation introduced in the next subsection).

An important concept arised in the learning process is the trade-off between the exploration and the exploitation. Since the policy is in general refined through the interactions with the environment, the agent faces the choice either to **exploit** the estimated optimal policy so far or to **explore** other possible actions beyond the current estimate so that it may find an overall better policy.

2.4.2 Markov Decision Process

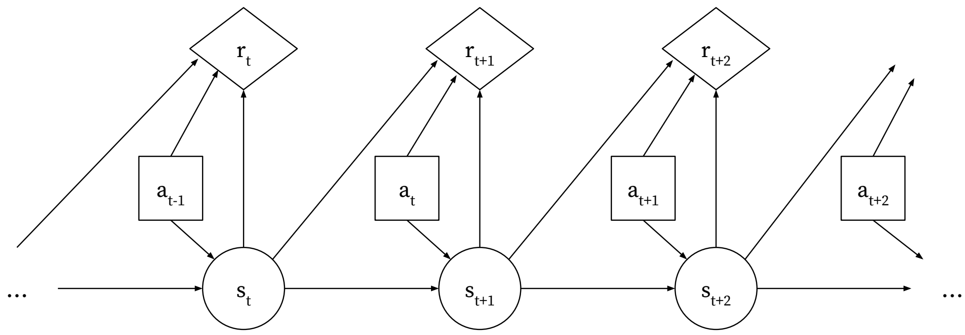


Figure 2.13: Markov Decision Process

In the literature, under certain assumption, namely the **Markov property**, reinforcement learning is usually mathematically formulated into a **Markov Decision Process** (MDP) as shown in Figure 2.13. Given a state s_t, the action taken at time t as a_t, the next state as $s_t + 1$ and the received reward as r_t, the key dynamics of the environment is defined as:

$$p(s_{t+1}, r_{t+1} | s_t, a_t)$$

Note in the most general case, $s_{t+1}, r_{t+1} \backsim s_0, a_0, r_1, ..., r_t, s_t, a_t$, however as already mentioned earlier it is reduced to this form under Markov assumption. With this formulation, the state-transition probability is defined over the two consecutive states as:

$$p(s_t + 1 | s_t, a_t)$$

This in a sense assumes a **model** for the environment. Methods with explicit formulation on the state-transition model are sometimes referred to as **Model-based**, the counter part without explicit transition model are referred to as **Model-free**.

The policy π is defined as a function over the current state:

$$\pi(s_t) = p(a_t|s_t)$$

As discussed earlier, pi can be either deterministic or stochastic. In the former case, given s_t, the agent performs a specific action A with $p(a_t = A|s_t) = 1$, the pi is then simply a lookup table to specify which action to take under different states. In the latter case, pi defines a probability distribution of actions over states, and given a state s_t, the agent samples an action with respect to pi.

One unique concept that distinguishes reinforcement learning from other learning paradigms like supervised learning and unsupervised learning is the **reward**. When an agent interacts with an environment, the reward signal returned from the environment regulates its decision for the actions. Under the MDP formulation, a commonly used generic formulation for reward is the **reward function**, i.e. the current reward $r_{t+1}$is defined as a function of the previous state s_t, the action taken before a_t and the resulted current state s_{t+1}:

$$r_{t+1} = r(s_t, a_t, s_t + 1)$$

Often the agent only receives direct reward at the end of the episode. However we can introduce additional intermediate rewards with **reward shaping** methods to speed up the learning process. The reward shaping should be designed in a way to maintain the policy invariance, i.e. the optimal policy remains unchanged under the new reward settings. The reader can find more detailed discussion in Ng *et al.* (1999).

In general, an agent does not act by only greedily maximizing the current reward but instead seeks to maximize the cumulative reward received in the long run. Under the episodic settings, the accumulative reward is defined as the **return**, which is the sum of rewards afterwards till the end of the episode:

$$R_t = \sum_{i=0}^{T} r_{t+i+1}$$

By introducing $\gamma (0 \leq \gamma \leq 1)$ as the **discounting factor**, we can control the overall strategy for the agent, i.e. how much the agent weights the decision based on current reward over the ones in the future: with $\gamma = 0$, the agent is "myopic", degrading to a greedy decision for immediate reward; with γ approaches 1, the agent is "farsighted", weighting the future rewards more strongly. The **discounted return** is defined as:

$$R_t = \sum_{i=0}^{T} \gamma^{t+i+1} r_{t+i+1}$$

Since the agent makes decisions before it actually observes the return, the objective can be hence formulated in terms of an estimate for the return as the **expected return** $\mathbb{E}_\pi[R_t]$ subjected to a specified policy pi. Hence the overall learning objective is defined as:

$$\max \mathbb{E}_\pi[R_t]$$

The expected return can be defined for each state. This leads to the concept of **state-value function for policy** π as $v_\pi(s)$, defining the **value** of a given state s, i.e. the expected return after state s following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s]$$

A similar concept of **action-value function for policy** π as $q_\pi(s,a)$ is also defined for each state-action pair (s,a), being the expected return starting from a given state s, taking action a and following the policy π afterwards:

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a]$$

Both v_π and q_π subject to the associated policy π. Naturally, different policies can lead to different values. The **optimal policy** π_* is defined as the one with the maximum value, or more formally put as:

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S$$
$$q_*(s,a) = \max_\pi q_\pi(s,a), \forall s \in S, a \in A(s)$$

The S denotes the state space that the agent acts on and $A(s)$ refers to the set of all possible actions given a state s.

By now, solving a reinforcement learning can be reduced to a search problem for π_*. Yet with various representations for the problems of interest, there are different ways to solve the corresponding formulation. For example, one can represent the problem based on the value function (**value-function based method**). The **iterative methods** are often used, such as the notion of **generalized policy iteration** (GPI) by Sutton and Barto (1998). As shown in Figure 2.14, π_0 is initialized first for the policy and then it is evaluated for the associated value function v_0 (**policy evaluation**), then **greedy** choices based on v_0 is used to refine the policy as π_1, and this process is repeated until it ultimately converges to π_* and v_*. An alternative paradigm is to directly model the policy and search for the optimal solution in policy space (**policy-based method**). The DQN used in our work is an example of value-based method for which we will describe in more details in the following subsection.

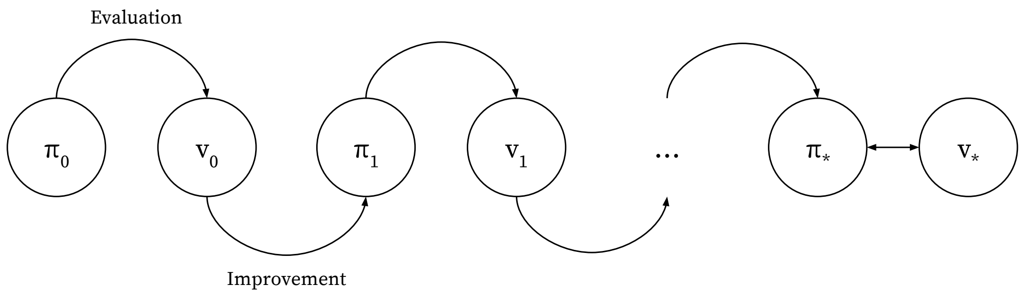


Figure 2.14: Generalized Policy Iteration

2.4.3 Q-Learning and Deep Q-Network

One important framework to solve reinforcement learning is the **temporal-difference** (TD) learning. It is one of the value-function based methods and fits in the GPI framework as we just discussed in the previous subsection. The name of 'temporal difference' was first used by Sutton (1988) and derives from its usage of updating value function estimates based on the difference between temporally successive predictions.

Consider a simple example, $TD(0)$, as discussed in Sutton and Barto (1998) is defined by:

$$V(s_t)|_{t+1} \leftarrow V(s_t)|_t + \alpha[r_{t+1} + \gamma V(s_{t+1})|_t - V(s_t)|_t]$$

Here α is a constant step-size (can be also understood as the learning rate), γ is the discount factor, $V(s_t), V(s_{t+1})|_t$ are the estimates at time t for $V(s_t)$ and $V(s_{t+1})$, and the difference between the two contributes to the update to the estimate at time $t+1$ for $V(s_t)$.

Q-learning (Watkins and Dayan, 1992) is an prominent example of TD methods. As the name indicates, its estimate is based on Q-value (action-value) function and is defined as:

$$Q(s_t, a_t)|_{t+1} \leftarrow Q(s_t, a_t)|_t + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)|_t - Q(s_t, a_t)|_t]$$

The update is repeated until it converges, i.e. the TD-update $r_{t+1} + \gamma \max_a Q(s_{t+1}, a)|_t - Q(s_t, a_t)|_t$ approaches 0 or some other specified threshold. The full procedures are sketched as below.

```
Initialize $Q(s, a)$,$\forall s, a$
while not converge criteria do
    // Repeat episodes
    Initialize $S$
    while $S$ is not terminal do
        // Repeat steps
        Choose A from S w.r.t. $Q$ // e.g.$\epsilon$-greedy
        Take action $A$, observe $R$, $S'$
        Update $Q(S, A)$
        $S = S'$
    end
end
```

Algorithm 1: Q-Learning

Note at each step, ϵ-greedy is often used for selecting actions for the balance between exploration and exploitation with respect to the current estimate of Q-value. It is defined as:

$$A \leftarrow \begin{cases} \arg\max_a Q(s, a) & \text{with probability } 1 - \epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

One obvious limitation of the original Q-learning is that it does not scale to real-world problem with arbitrarily large state space. One idea to overcome this is to explore some parametrized function approximation to the Q-value function so that it can represent all different states in a more compact way other than simply memorizing.

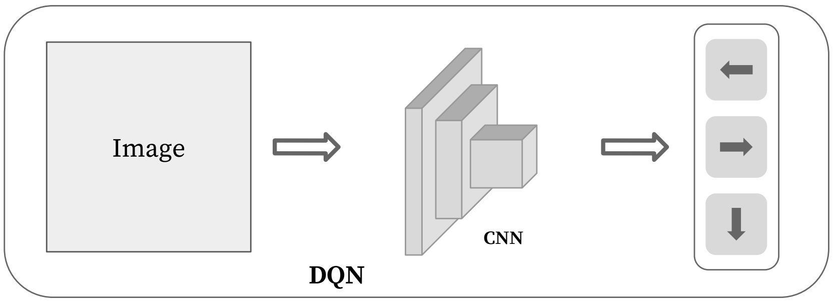


Figure 2.15: The DQN by Mnih *et al.* (2015).

One recent success along this line is the **Deep Q-Network** (DQN) by Mnih *et al.* (2015), which achieves human-level performance across a variety of Atari games. Several techniques contributed to the working system. For starters, the Q-value function is now represented by a deep **Convolutional Neural Network** (CNN), which has been shown great success on image classification task (Krizhevsky *et al.*, 2012). The CNN is parameterized by θ, receives the observed images as the input, and outputs the predicted q-value $Q(s, a; \theta)$ for all possible actions a as shown in Figure 2.15.

The learning follows the same idea of the original Q-learning but takes slightly different forms as the update now takes place in the parameter space θ of Q-value function other than Q-value directly. Hence the learning objective now is to minimize a loss function in terms of the temporal difference:

$$L(\theta_t) = [r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t) - Q(s_t, a_t; \theta_t)]^2$$

The resulted update can therefore be expressed in the gradient-descent form as:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} L(\theta_t)$$

Here α is the learning rate for the gradient-descent step. In practice, α is set using the RMSProp (Tieleman and Hinton, 2012).

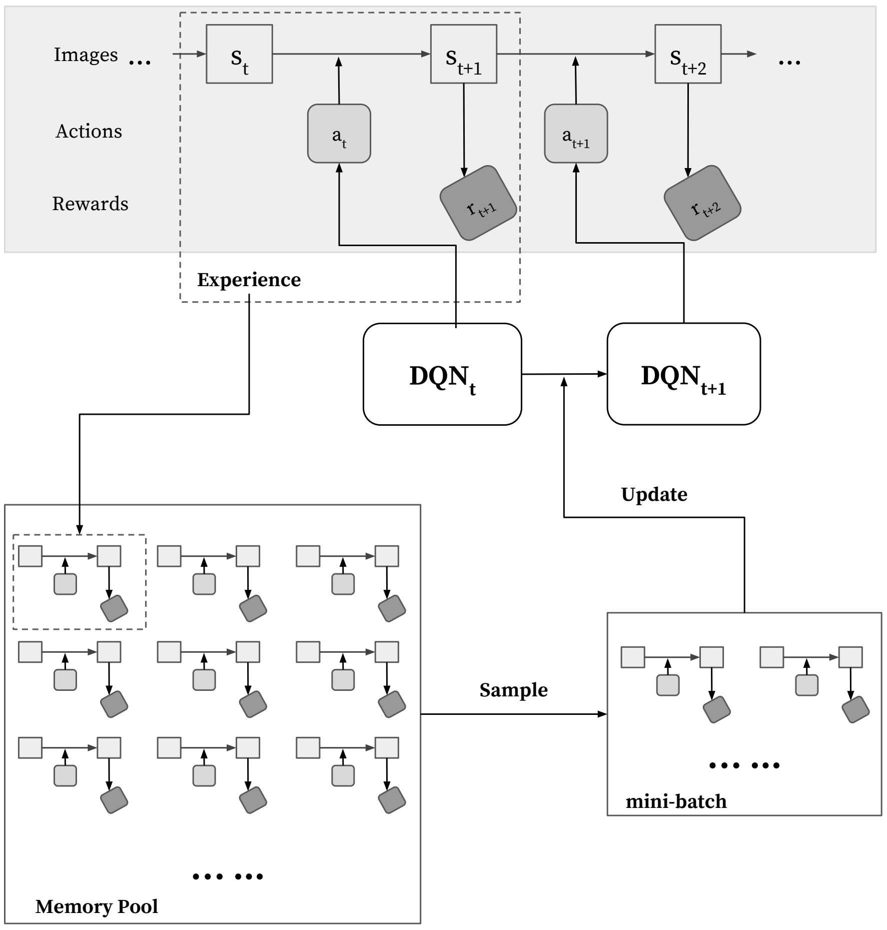


Figure 2.16: Experience replay used in DQN.

Along with the CNN modeling, **experience replay** (Lin, 1992) is also used. As shown in Figure 2.16, at each time-step t, the sequence of current observation s_t, the action a_t, the resulted next observation s_{t+1}, the reward r_{t+1} are collected as an **experience** into the memory pool, and θ is then updated by a mini-batch sampled from it. The memory pool is usually set to a fixed size M queue structure, whenever a new experience enters the pool, the 'oldest' experience leaves the pool, i.e. the lifespan of an experience in the pool is M. In practice, M is often set to be a large number so that a single experience can stay in the memory for a long time and is likely to be sampled several times. In this way, the **data efficiency** is increased, as experience can be reused for the update. Also, since each update is done by randomly sampled data, it removes the dependence among successive experiences on the current weights, hence improves the learning in general.

Noticeably, the memory pool is first filled from empty to full and then the old experience leaves whenever new experience enters. In actual implementation, since it often pre-allocates space for the full memory pool with certain ways of initialization, mini-batch should only be sampled among the actual experience in the pool other than blindly from all the entries in it. This is because those initialized entries are not generated from interactions in the environment and can hence deteriorate the learning.

Another important technique used in the work is the **skip-frame** (Bellemare *et al.*, 2012). More specifically, the agent sees and selects actions on every k-th frame ($k = 4$ is used) instead of every frame, and its last action is repeated on the skipped frames. Skip-frame is used together with the **stack-frame** representation. Note, in the actual DQN working system, each state is not the single image for the current time-step but a stack of consecutive l frames ($l = 4$ is used). Overall, single update spans across $k \times l$ frames as shown in Figure 2.17.

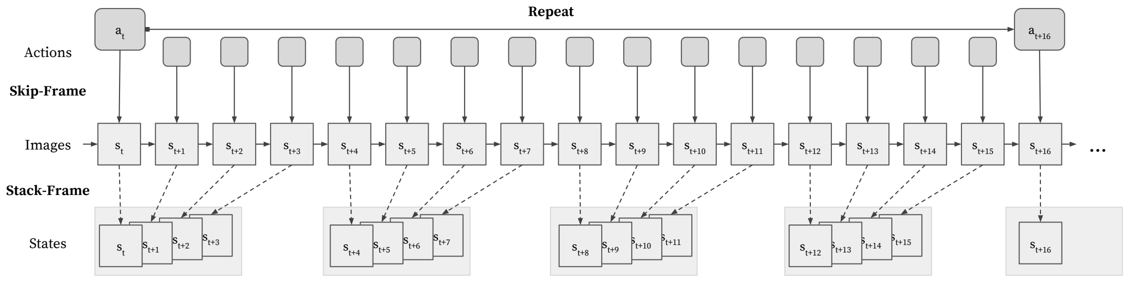

Repeat
Actions
Skip-Frame
Images
Stack-Frame
States
a_t
a_{t+16}
s_t s_{t+1} s_{t+2} s_{t+3} s_{t+4} s_{t+5} s_{t+6} s_{t+7} s_{t+8} s_{t+9} s_{t+10} s_{t+11} s_{t+12} s_{t+13} s_{t+14} s_{t+15} s_{t+16}
...

Figure 2.17: Skip-frame used in DQN.

Following the DQN, several variants have been proposed to further improve the results, including the dueling network (Wang *et al.*, 2016), double DQN (Van Hasselt *et al.*, 2016) and prioritized replay (Schaul *et al.*, 2016).

2.4.4 Outlook

The DQN has been shown to be a great success in beating the Atari games, yet the games themselves are still limited in terms of the visual richness and control schemes. Therefore, researchers now are working on tasks with more complex visual input and more challenging control schemes, examples include the Vizdoom (Kempka *et al.*, 2016) and Starcraft II (Vinyals *et al.*, 2017). Further, (Brockman *et al.*, 2016) set up a user-friendly platforms for benchmarking different reinforcement learning algorithms for a larger variety of tasks, including ones for continuous control and many more. Another notable recent highlight in reinforcement learning is the AlphaGo (Silver *et al.*, 2016) has outplayed top human players.

Part I
From Perception to Anticipation

Vision is a wide-spread modality for robotic perception, in computer vision, research has made great progress on tasks describing the content in the current image or video. For example, object detection to localize object in the frame, recognition to identify object class. To pursue higher level of understanding, it is necessary to expand the analysis for more complex tasks. Anticipation is one of such tasks, making prediction over the future based on the current observation, which can facilitate robotic systems to act in complex and dynamic environments.

In chapter 3 we tackle the recognition of ongoing activity from videos. Given the observation so far, we want to anticipate the activity class for the whole sequence.

Chapter 3

Recognition of Ongoing Complex Activities by Sequence Prediction over A Hierarchical Label Space

Contents

Human activity recognition from full video sequence has been extensively studied. Recently, there has been increasing interest in early recognition or recognition from partial observation. In general, it is very challenging to make a fine grained prediction of an activity given only a small fraction of the observation. To this end, we propose a method to predict ongoing activities over a hierarchical label space. We approach this task as a sequence prediction problem in a recurrent neural network where we predict over a hierarchical label space of activities. Our model learns to realize accuracy-specificity trade-offs over time by starting with coarse labels and proceeding to more fine grained recognition as more evidence becomes available in order to meet a prescribed target accuracy.

In order to study this task we have collected a large video dataset of complex activities with long duration. The activities are annotated in a hierarchical label space from coarse to fine. By directly training a sequence predictor over the hierarchical label space, our method outperforms several baselines including prior work on accuracy specificity trade-offs originally developed for object recognition.

This work has been published in WACV (Li and Fritz, 2016).

3.1 Introduction

Strong progress has been achieved in the area of human activity recognition over the last decade ranging from coarse actions (Schuldt *et al.*, 2004) to fine-grained activities (Rohrbach *et al.*, 2012). The majority of these techniques focus on what has happened. However, many application in surveillance system, crime prevention, assistive technology for daily living, human computer interaction interface, human robot interaction would like to infer which activity is currently going on in order to enable a system to behave proactive or context-aware.

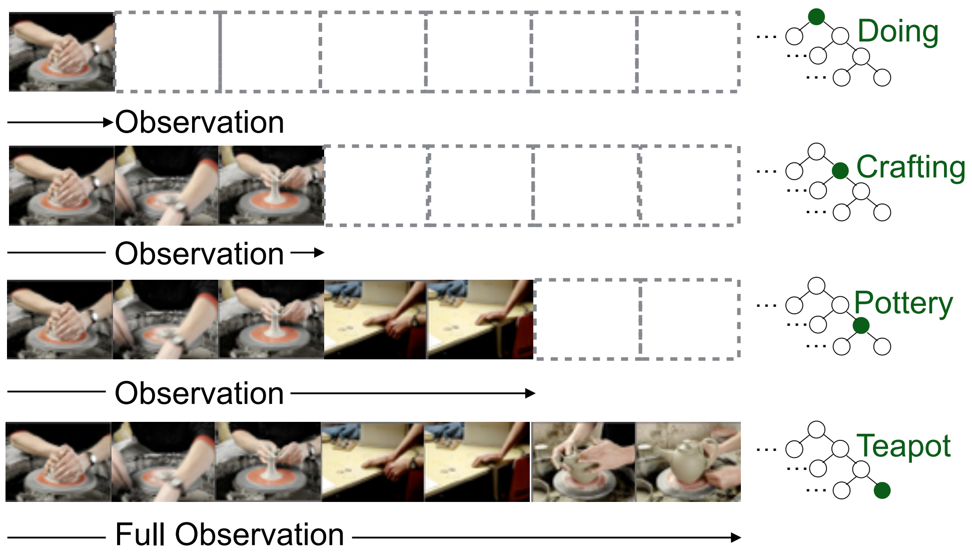


Figure 3.1: An overview of our approach to inferring goal during the recognition process of a complex activity by predicting with semantic abstraction.

The current systems that address early recognition and also activity recognition in general have a relatively short temporal reach. Typically the investigated activities are on the order of a minute – with a few exceptions like cooking activities. This limits the systems in the way they can act and assist to a more immediate, anticipatory response. In order to increase the temporal reach of our systems we have to study a richer set of longer term activities.

While this goal is desirable, it raises the question how much can be inferred about a complex activity from a very short observation? The natural concern is that in many cases the information might be quite limited until a more substantial fraction is being observed. Therefore we argue that recognition of complex activities from partial observation has to go hand in hand with a mechanism to predict at different semantic granularity in order to deliver a useful prediction at anytime. In this sense, we have to bring coarse and fine-grained recognition together in a single approach.

In order to study this challenging problem, we propose the first dataset of complex activities that have a substantial time span and that are organized in a semantic hierarchy. We study how prediction can be performed from partial observation of ongoing activities in this setting by employing of a semantic back-off strategy. We introduce a new representation that models the uncertainty over different time frames across different levels of the semantic hierarchy. More specifically, we propose a recurrent neural network formulation that learns to directly predict on this hierarchical label space from coarse to fine in order to achieve an accuracy specificity trade off. A performance measure is proposed that evaluates the gain in specificity over time as more observations become available. Our new method outperforms several baselines as well as techniques that we adopt from the literature on accuracy specificity trade-offs for object recognition.

3.2 Related Work

Early computer vision research in recognizing human motion and activities can be date back in early 1990's (Aggarwal and Ryoo, 2011). Researchers have explored various approaches to tackle the task and excellent surveys are available (Moeslund *et al.*, 2006; Poppe, 2010; Aggarwal and Ryoo, 2011; Weinland *et al.*, 2011). In this section, we first review related datasets for activity recognition and then discuss some related approaches for analysis from partial observation.

Activity Datasets There is a large number of datasets proposed for activity recognition or detection with various levels of complexity. Early efforts to construct such datasets include the KTH dataset (Schuldt *et al.*, 2004) and the Weizmann dataset (Blank *et al.*, 2005) which feature simple activities like walking and jumping.

More complex datasets are introduced later with various backgrounds, slightly longer duration and more importantly, with additional interaction of objects or peoples. Typical examples are answering phone (human-object interaction) in the Hollywood2 (Marszalek *et al.*, 2009), golf swinging (human-object interaction) in the UCF Sports (Rodriguez *et al.*, 2008) and hand shaking (human-human interaction) in the UT-Interaction (Ryoo and Aggarwal, 2010). However, activities covered in these datasets usually consist only very few types of (in most cases only one) interactions, hence their structure are still relatively simple and individual video's length is rather short.

In contrast to this, there are also long-duration datasets collected from surveillance video like VIRAT (Oh *et al.*, 2011) or daily living recordings like TUM Kitchen (Tenorth *et al.*, 2009), CMU-MMAC (De la Torre *et al.*, 2008), URDAL (Messing *et al.*, 2009), TRECVID (Over *et al.*, 2010), ego-centric datasets (Fathi *et al.*, 2011;

Dataset	#Class	Avg. #Frames per Seq.
KTH	6	91
UT interaction	6	84
UCF sports	9	58
HMDB51	51	93
MPI cooking	65	157
Youtube action	11	163
Olympics sports	16	234
Hollywood2	12	285
VIRAT	12	357
KSCGR	8	786
Our dataset	48	8.7K

Table 3.1: An overview of action/activity datasets in the literature.

Stein and McKenna, 2013), and MPII Cooking (Rohrbach *et al.*, 2012). Videos from these datasets show complex activities that comprise multiple interactions of different type. Considering the example of making a dish in the cooking dataset (Rohrbach *et al.*, 2012), one needs to interact with different tools and ingredients to complete a recipe.

One limitation about current long-duration datasets is that they are mostly used for recognition of elementary activities within the long sequence instead of the recognizing the overall process. In addition, they are limited to a particular domain and do not provide a hierarchical label space. Our dataset exactly aims to fill this gap by providing dataset of complex activities across multiple domains that are organized in a semantic label hierarchy. A related dataset was built in Fabian Caba Heilbron and Niebles (2015) where activities are grouped with respect to social interactions and places where the activity usually takes places. There are several recent works collecting videos from YouTube (Liu *et al.*, 2009; Kuehne *et al.*, 2011; Sergio *et al.*, 2013). We also collect our dataset from web sources, as it is a practical way to collect a sizable dataset of realistic videos. A more detailed overview of our dataset together with the aforementioned datasets is shown in Table 3.1.

Analysis from Partial Observation There are a few recent works focusing on analysis from partial observation. Early recognition (Ryoo, 2011) aims to recognize activity from the forepart of videos. The authors use accumulated histogram with respect to the observed frames to represent the activity of interest, and performs sequence matching with templates averaged from training dataset. Following the similar idea of sequence matching, Cao *et al.* (2013) relaxes the location of observed part and addresses recognition from partial observation and applies sparse coding for better representation of the matching likelihood. A slightly different line of work (Hoai and De la Torre, 2012) focuses on early detection, deciding the temporal location of the activity, yet as shown in Cao *et al.* (2012) the algorithm does not work so well in practical recognition task. A more recent work (Lan *et al.*, 2014) builds the early activity prediction on the human detection and uses the resulted tracks to form a hierarchical representation, yet in real world scenario, the detection and tracking are usually expensive for complex scene involving multiple people and can often be unreliable, not mention in situations like ego-centric video where it is impossible to obtain a detection result for the person doing the activity.

Beyond predicting a label for the sequence, Kitani *et al.* (2012) poses a more challenging task to predict activity where a MDP is used for the distribution over all possible human navigation trajectories. The walking path addressed in Kitani *et al.* (2012) is still relatively simple, hence Koppula and Saxena (2013) further explores indoor activities which involves more complex interactions between human and object. They represent the distribution over incoming states with a set of particles sampled from an augmented CRF. The activities exploited in Koppula and Saxena (2013), like taking medicine, already meets our definition of complexities, yet it only applies to a very limited number of objects in very simple scenes under controlled lab conditions. It is not clear how it scale up to real world application and longer time scales.

Prediction vs. Early Recognition Here we want to distinguish two different types of problems addressed in related literature, i.e. prediction for unseen action/activity or early recognition on the current observation. This is illustrated in Figure 3.3 with the example of making a salad which involves multiple activities including peeling, cutting and mixing. As one watches the activity moving on, prediction is to forecast what is the next move after mixing whereas early recognition is to tell the main theme of the activity as cooking regardless of only partial observation of the whole process. Kitani *et al.* (2012) and Koppula and Saxena (2013) are examples of prediction and Ryoo (2011), Cao *et al.* (2013) and Lan *et al.* (2014) are for early recognition.

Indeed predicting real world activity is a very difficult task, it generally requires various components including but not limited to pose estimation, object detection and temporal segmentation where each task along is challenging already in real world application not to mention it is nontrivial to combine them for a reasonable solution. Our work focuses on early recognition which due to the long temporal duration of our investigated activities, also extend to the future.

Activity Recognition with Recurrent Neural Networks Baccouche *et al.* (2011) applies a different variant of RNN to action recognition task. Our work differs from it in that we focus on the early recognition and prediction over a hierarchical label space with a learned accuracy-specificity tradeoff. Our results show the joint training of the sequence model w.r.t. to the accuracy specificity trade-off is key to our performance improvements.

3.3 Method

In this section, we first discuss different ways of video representation and formulate the task of early recognition in video sequence. Our goal is to predict at any point in the sequence over a hierarchical label space in order to realize accuracy-specificity trade-offs. In order to have a temporal integration of information over time and learn classifier and the hierarchical trade-off jointly, we employ recurrent neural networks for modeling videos and extend it to labeling in a hierarchy, and learning accuracy-specificity trade-off in early activity recognition in videos.

3.3.1 Video Representation

Given a specific descriptor x, a video sequence can then be represented as $[x_1, x_2, ..., x_T]$, where T denotes the frame index. There are several ways to obtain a compact representation for sequences as shown in Figure 3.2: (1) pooling over the whole sequence; (2) partition the video into separated temporal segments, and combine the representation from the segments into a temporal sequence representation $[x_{1:t}, x_{t+1:2t}, ...x_{T-t+1:T}]$, where t is the number of frames for each segment. This approach has been applied in Cao *et al.* (2012); Tang *et al.* (2012); (3) sample clips (also temporal segments) from the video to represent the video, each clip is trained as a single instance in the video class, i.e. $[x_{t_1:t_1+t}] \cup [x_{t_2:t_2+t}] \cup ...$, where $t_1, t_2, ...$ are the time stamps when the clips are sampled, and final prediction on the video is based on the majority vote from the sampled clips. This approach is applied in the large-scale video recognition work (Karpathy *et al.*, 2014).

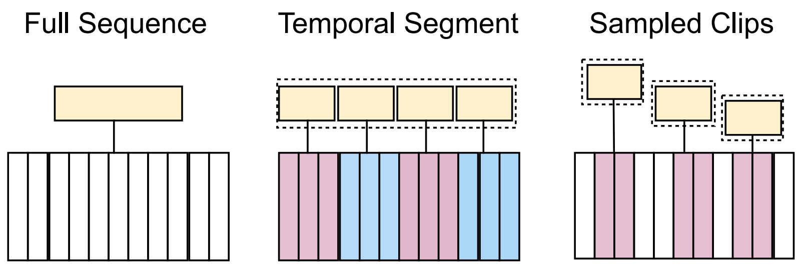


Figure 3.2: Different ways to represent individual video sequence: full sequence (left), use one representation for the full sequence; temporal segment (middle), partition full sequence into temporal segments and combine representation for individual segments into one representation; sampled clips (right), sample video clips from the sequence to form several representations.

We use improved dense trajectory feature (Wang *et al.*, 2011) as the basic feature to encode spatio-temporal information across frames in the videos with a code book of size 4000 obtained via k-means clustering. It uses a dense representation of extracted trajectories and combines with trajectory-aligned features, including HOG (Dalal and Triggs, 2005), HOF (Laptev *et al.*, 2008), motion boundary histograms (MBF) (Dalal *et al.*, 2006). The descriptor itself has been shown to achieved the state-of-art performances on several public datasets.

3.3.2 Early Recognition

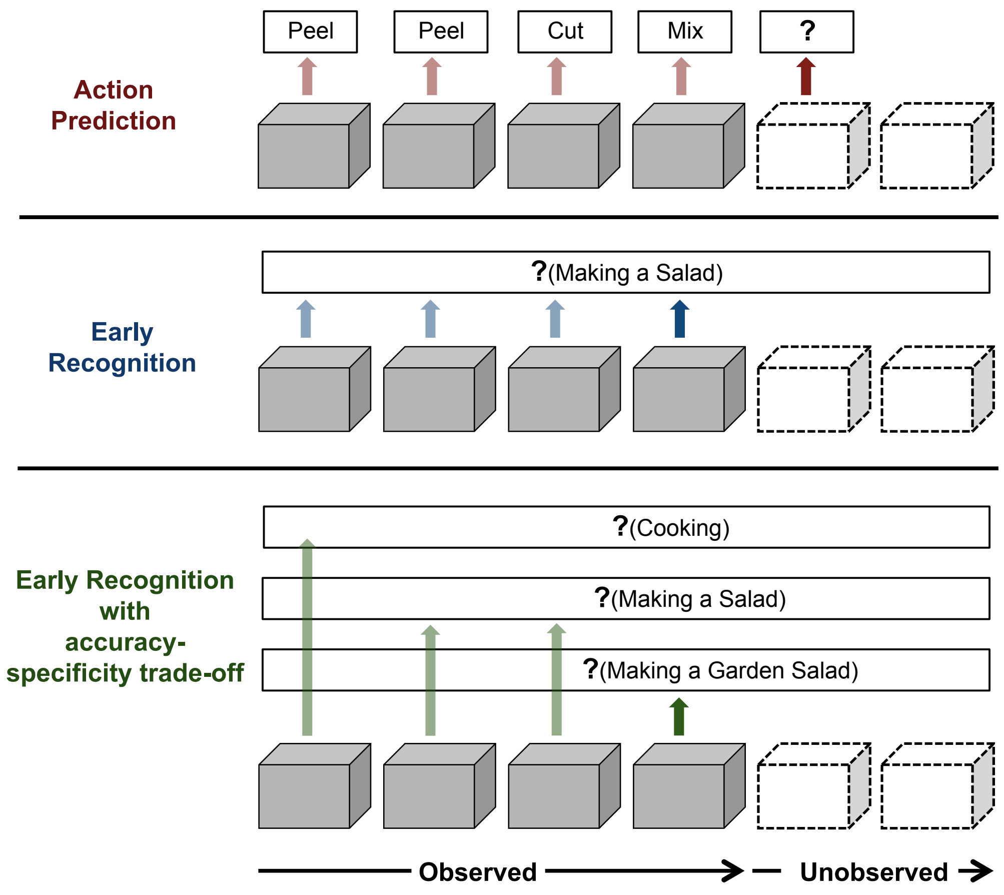


Figure 3.3: Example to show the difference between action prediction and early recognition. Given a streaming video sequence, (top) action prediction aims to give a label to a incoming local temporal segment (mix or cut); (middle) standard early recognition gives a label for the global video sequence (make a salad); (bottom) our proposed framework to predict in the hierarchy with accuracy-specificity trade-off. The solid arrows mark when the decision is made.

Given a video sequence $[x_1, x_2, ..., x_T]$, where T is the last frame index of the video. Early recognition generally refers to the task of classification:

$$y = f([x_1, x_2, ..., x_{\tilde{t}}]) \quad , \tilde{t} \leq T$$

The standard full video classification can be seen as a special case of this formulation where $\widetilde{t} = T$. It is important to point out the difference between early recognition of activity (Ryoo, 2011) and human action prediction/anticipation (Kitani *et al.*, 2012; Koppula and Saxena, 2013; Lan *et al.*, 2014) whereas the former is essentially a sequence classification problem that maps sequence in a single label

$$X : [x_1, x_2, ..., x_T] \mapsto y$$

and the latter is a temporal classification problem that maps a input sequence into a target sequence.

$$X : [x_1, x_2, ..., x_T] \mapsto [y_1, y_2, ..., y_L]$$

A specific example is shown in Figure 3.3 for ongoing video stream of making salad. Assume we already observe the person in video peeled the cucumber, cut it into slice, mix it with other ingredients, early recognition generally aims to output a global label for the sequence based on the available observation, in this case, the label should be 'making salad' while the action prediction tries to predict the specific local action label for the incoming video frames, in this case, the correct label would be 'put the salad in plate'.

3.3.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a class of artificial neural network that allows connections between units to form a directed cycle.

We consider an architecture with one self-connected hidden layer, which can be unrolled in time as shown in Figure 3.4. One notable merit for RNN is that the recurrent connections allow a 'memory' of previous inputs to persist in the network's internal state which can then be used to influence the network output (Graves, 2012). This characteristic makes it suitable for sequence analysis, especially in our case, with long duration and complex video sequences.

Given a video sequence composed of temporal segments $x_1, x_2, ..., x_T$, each in R^d, the network computes a sequence of hidden states $h_1, h_2, ..., h_T$, each in R^m and predictions $y_1, y_2, ..., y_T$, each in R^k. The hidden unit integrates the information from the arrived observation and those propagated from previous blocks of the networks:

$$\begin{aligned} \widetilde{x_i} &= W_{xh} x_i + W_{hh} h_{i-1} + b_h \\ h_i &= \tanh(\widetilde{x_i}) \end{aligned}$$

Each prediction unit represents the class label up to the current observation and is activated by the hidden unit via a softmax function:

$$\begin{aligned} \widetilde{h_i} &= W_{hy} h_i + b_y \\ y_i &= \mathrm{softmax}(\widetilde{h_i}) \end{aligned}$$

where W_{xh}, W_{hh}, W_{hy} are the weight matrices and b_h, b_y are the biases. The training is done by BackPropagation Through Time (BPTT) (Rumelhart *et al.*, 1985) and the parameter is learned by conjugate gradient descent method.

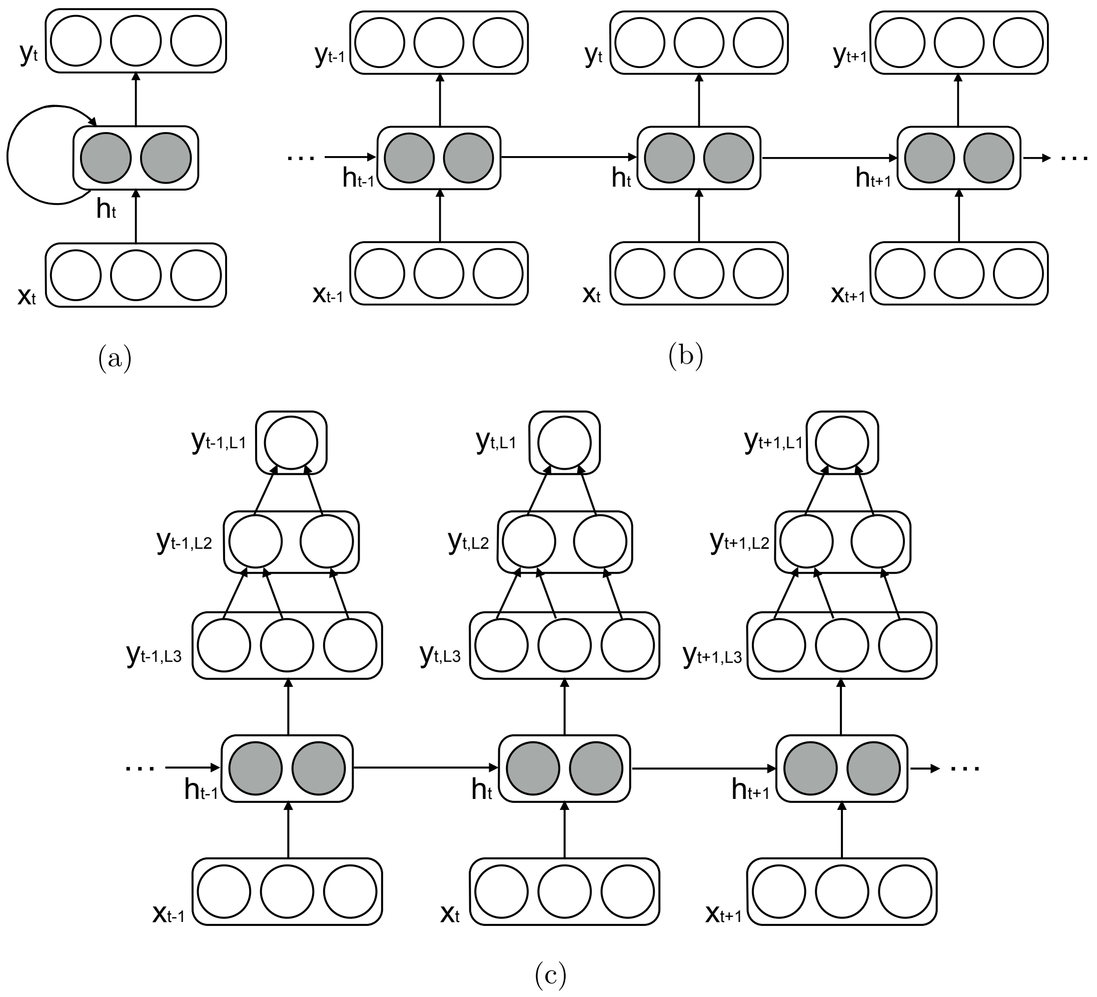


Figure 3.4: Graphical model for recurrent neural networks. (a) The recurrent neural networks with a single, self-connected hidden layer. (b) The unrolled model with respect to discrete time steps from (a). (c) The recurrent network with structural output over the hierarchy.

3.3.4 Early Recognition in a Semantic Hierarchy

Deng *et al.* (2012) first introduce the concept of optimizing an accuracy-specificity trade-off for hierarchical image classification with the DARTS algorithm. Here we briefly recap the formulation of the concept and discuss how we extend it to our settings.

Optimizing Accuracy-Specificity Trade-Offs The key idea behind the accuracy-specificity trade-off is to make cost-sensitive prediction over the hierarchy, where predictions at upper level (fine-grained level categories) of the hierarchy get penalized more than those at lower level (coarse level categories). By optimizing over both the specificity and accuracy an optimal trade-off is found. More formally, given a classifier $f : X \mapsto Y$, with accuracy $\Phi(f)$ defined as,

$$\Phi(f) = E[f(X) \in \pi(Y)]$$

where $\pi(Y)$ is the set of all possible correct predictions, $[P]$ is the Iverson bracket, i.e., 1 if P is true and 0 otherwise. The preference for specific class labels over general class labels at each node v, i.e. the specificity is encoded with information gain (decrease in entropy) by

$$r_v = log_2|Y| - log_2 \sum_{y \in Y} [v \in \pi(y)]$$

The total reward for the classifier f is hence defined as

$$R(f) = E(r_f(X)[f(X) \in \pi(Y)])$$

The optimal trade-off between accuracy and specificity is then formulated as maximizing the reward given an accuracy guarantee $1 - \epsilon \in (0, 1]$:

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & R(f) \\ \text{subject to} \quad & \Phi(f) \geq 1 - \epsilon \end{aligned}$$

Accuracy-specificity Trade-off Over Time Motivated by ideas from incremental feature computation and anytime recognition (Karayev *et al.*, 2012, 2014), we extend this concept with a temporal dimension to early recognition and model the decision process when analyzing an ongoing video stream. At each time step, we have to predict a label in the hierarchy and hereby trade-off between accuracy and specificity.

$$R(f, t) = E(r_f(X_t)[f(X_t) \in \pi(Y)])$$

The intuition behind this is that when only observing a small portion of the video, we have little evidence to accurately predict at a fine-grained level but may still be able to give a sensible coarse-level class label. By considering the total cost of possible wrong prediction at fine-grained level and probably correct prediction at coarse-level, together with our preference to predict at different levels, we are likely to give prediction at coarse-level given little observed data. When observing more and more parts of the video, we become more certain about our prediction at the fine-grained level, and by the same mechanism, we start to predict at a more fine-grained level. Figure 3.3 shows a concrete example. By summing up the term over time T,

$$\sum_t^T R(f,t) = \sum_t^T E(r_f(X_t)[f(X_t) \in \pi(Y)])$$

we can evaluate the efficiency for accuracy-specificity from a classifier f.

Structured Output RNN for Predictions Over Hierarchical Label Spaces We propose an RNN optimizing the objective from above by predicting over a structured output space – our hierarchy H. We denote labels at top-layer, middle-layer and bottom layer in the hierarchy as Y_1, Y_2, Y_3 (coarse to fine). As shown in Figure 3.4, our RNN model directly predicts an output layer $y_{3,i}$ representing the posterior probabilities over the fine grained labels in the bottom layer.

$$y_{3,i} = p(Y_3 = i|x), i = 1, ..., K_3$$

where K_3 is the number of classes within the layer. On top of this layer, we introduce an additional layer to represent the middle layer, where the connections between these two layers are defined according to the hierarchy H, i.e. if class i in bottom layer belongs to class j in middle layer, node i and j are connected or $(i,j) \in H$. Accordingly, the middle layer activations are defined as follows:

$$\begin{aligned} y_{2,j} &= p(Y_2 = j|x), j = 1, ..., K_2 \\ &= \sum_i p(Y_2 = j|Y_3 = i)p(Y_3 = i|x) \end{aligned}$$

Similarly we define another layer above this layer to represent top layer prediction (coarse labels):

$$\begin{aligned} y_{1,k} &= p(Y_1 = k|x), k = 1, ..., K_1 \\ &= \sum_j p(Y_1 = k|Y_2 = j)p(Y_2 = j|x) \\ &= \sum_j \sum_i p(Y_1 = k|Y_2 = j)p(Y_2 = j|Y_3 = i)p(Y_3 = i|x) \end{aligned}$$

The complete model is shown in Figure 3.4. Based on the this prediction over the hierarchy we define a structured loss in order to optimize for the desired accuracy specificity trade-off:

$$\begin{aligned} Loss(\theta, D) = -\sum_i &\log p(Y_1 = y_1^{(i)}|x^{(i)}, \theta) \\ &+ \log p(Y_2 = y_2^{(i)}|x^{(i)}, \theta) \\ &+ \log p(Y_3 = y_3^{(i)}|x^{(i)}, \theta) \end{aligned}$$

where D denotes the training set$\{X, Y_1, Y_2, Y_3\}$, and i indexes the i-th data instance in D.

3.4 Experiments

We first present our new dataset of complex activities with a hierarchical label space and afterwards perform a quantitative comparison of our proposed method and compare to several baselines.

3.4.1 Datasets

We explore various complex activities composed of multiple interactions. Recording videos for a large set of diverse classes is difficult, considering we need to find experts in different fields to perform the activities and capture multiple videos for a single class. In addition, this would eliminate the challenge of different capture devices. Hence, we build our dataset on videos from web to create our dataset. In the following part, we briefly discuss how we collect videos, pre-process the data and tackle the associated challenges of building such a dataset.

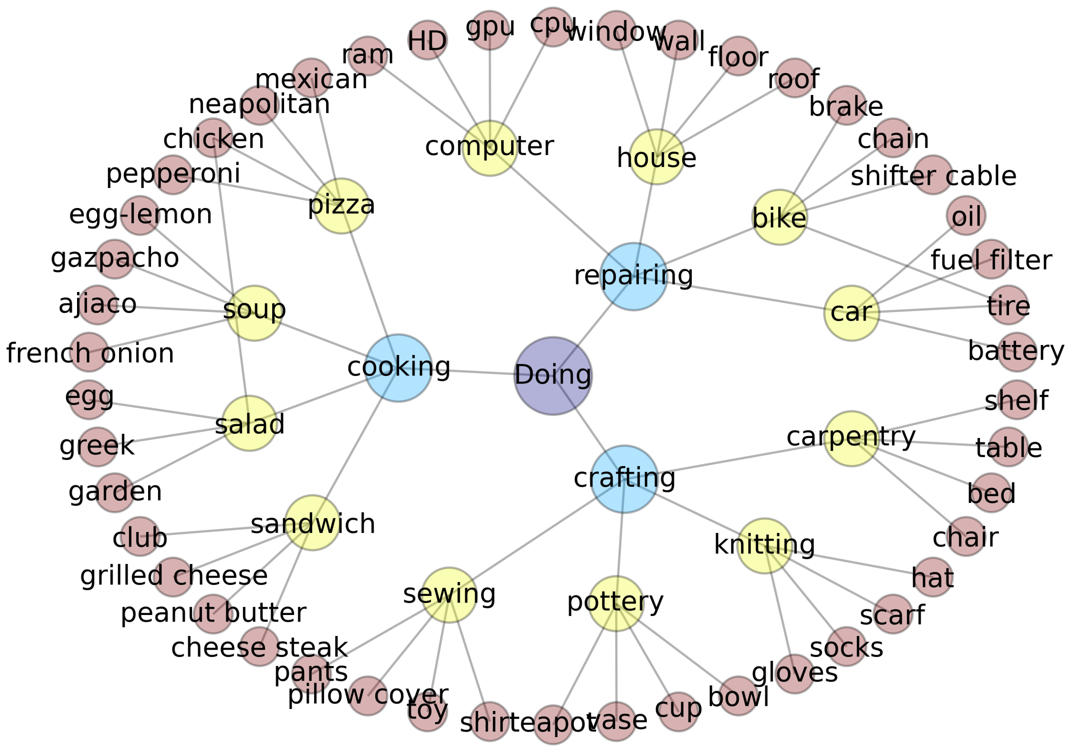


Figure 3.5: 3-layer hierarchy defined in our dataset.

Data Collection We begin by defining a 3-layer semantic hierarchy (we count the root node of "Doing something" as layer 0) with 3 nodes in the first layer (cooking, crafting, repairing), then for each node, we select 4 specific derived categories as nodes to form the following layer, and for each node in the middle layer, we select 4 more specific classes as leaf nodes to form the bottom layer. For example, we include making "pizza", "soup", "salad" and "sandwich" as the second layer for "cooking", and for "salad", we consider making 4 different kinds of salads, namely "egg salad", "garden salad", "chicken salad" and "Greek salad". Overall, we obtain a tree for complex activities with total $3 \times 4 \times 4 = 48$ activity classes as leaf nodes. The full hierarchy is shown in Figure 3.5. 10 videos are collected for each leave node from YouTube and eHow, which sum up to 480 video clips with total length of more than 41 hours or more than 4.18 million frames. The dataset is available online[3]. Sample frames of the videos are shown in Figure 3.6. We use half number of videos for training and the rest for testing.

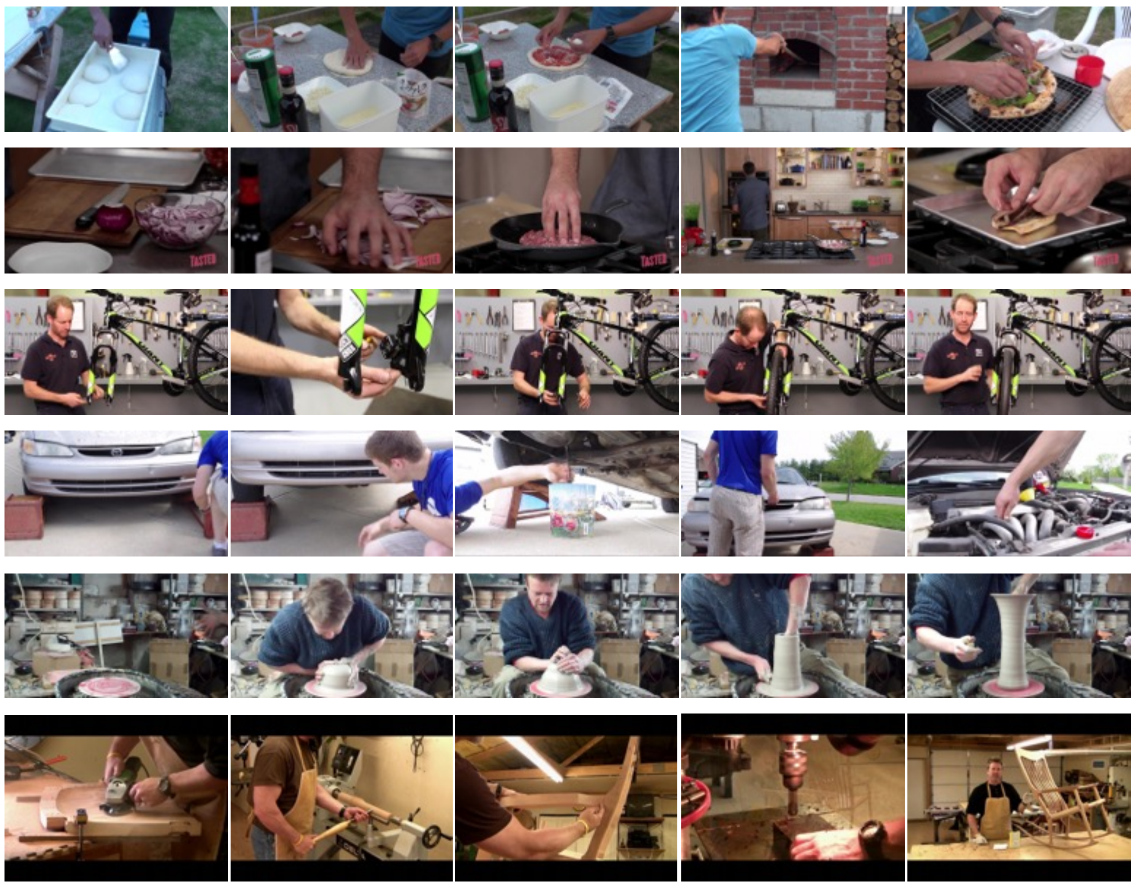


Figure 3.6: Some sample frames from our dataset.From top to bottom: make Neapolitan pizza, make cheese steak sandwich, repair bike brake, change car oil, make vase, build chair.

These videos from the web differ from the ones recorded from lab: while the latter record the whole process for each activity with good controlled conditions, the

[3]http://www.mpii.de/ongoing-activity

former are often edited (adding head leader, tail leader, titles, flashback, etc) from the uploaders under various conditions (different point of view, shooting skills, etc). Hence such data is very noisy and exposes many of the aforementioned challenges of realistic videos. We rescale video into 360p for further processing (640×360).

3.4.2 Full Video Recognition

First we consider a setting where we classify full video sequences. This is particularly interesting since our collected videos are significant more complex than previous datasets on both the temporal scale and internal structure. This provides a relative measure of difficulty for activity recognition on our database. We train and predict on full sequences with a SVM classifier. While a χ^2 kernel is usually applied to integrate different descriptors in a multi-channel fashion as in Wang *et al.* (2011), we find out that a linear kernel gets slightly better results on our dataset. Therefore we use the linear SVM for all our experiments. As can be seen from Table 3.2, the performance reaches 25.7% at layer-3 (fine-grained), which suggests that we have established indeed a very difficult task at this detailed level.

An alternative to training on full sequence is to use sampled clips to represent the whole video (Karpathy *et al.*, 2014). Accordingly, we randomly sample 20% of each video in the training set for training, and test on the full video sequences in the test set. Note here we predict directly on the entire video sequence instead of the average on the prediction of sampled clips from each test sequence. As shown in 3.2, this approach is also valid on our dataset and is only slightly worse than training on full video sequence.

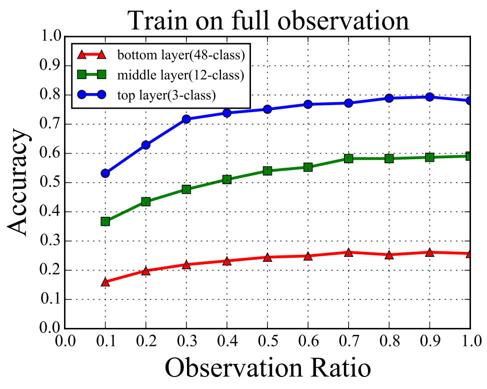
Layer	Clips-training(%)	Full-training(%)
bottom-layer	25.3	25.7
middle-layer	52.3	59.1
top-layer	76.4	78.1

Table 3.2: Classification results on full video sequence

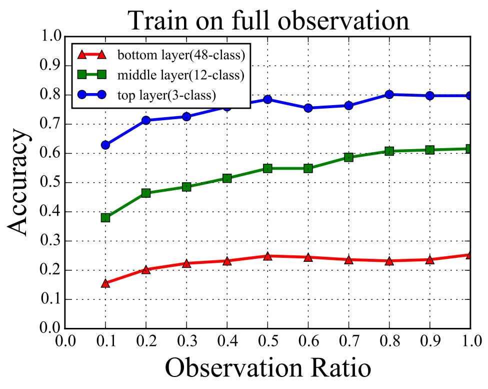
3.4.3 Recognition from Partial Observation

We proceed by examining the case where the video is only partially observed. We simulate two types of video segments to represent incremental arrival of video frames, (i) frames from the beginning with different observation ratios $[10\%, 20\%, ..., 100\%]$ where we explore different strategies for early recognition (ii) continuous frames taking up 10% of the full observation starting at different temporal location.

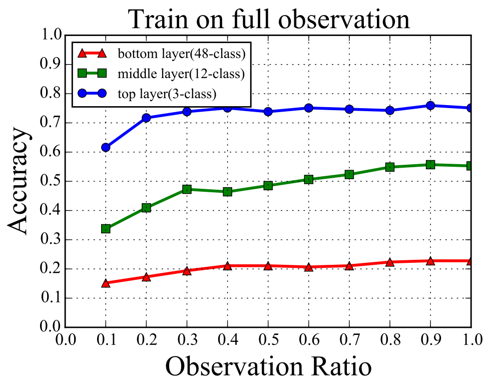
As a result a video is represented as a temporal sequence of these frame segments $[0-10\%, 10-20\%, ..., 90-100\%,]$. This setting simulates the practical setting where the observer starts from arbitrary position and perceive a part of videos and wants to infer the current activity class. We refer to this as *online recognition*.



(a)



(b)

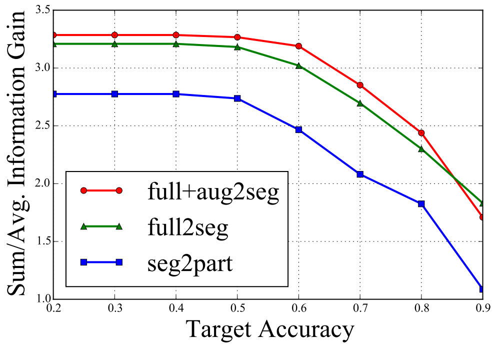


(c)

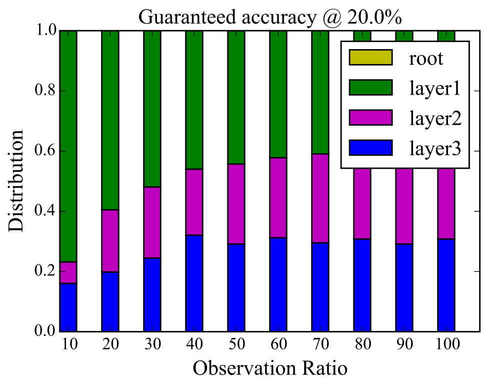
Figure 3.7: (a): train SVM model from full video and predict at different observation ratios. (b): train model from full video sequences and augmented dataset.(c): train model from sampled clips.

Early Recognition We evaluate the following strategies for early recognition: (1) train single model on full video and predict at different observation ratios; (2) train with augmented data i.e the combination of full video and video segments of different observation ratios and make predictions; (3) train on sampled video segments and test on partial observation, this is inspired by the result from our full-video recognition experiment and we would like to investigate how models trained on sampled clips can discriminate activity classes based on partial observation. We use the same pipeline as in the full video recognition and also test on differ layers in the label hierarchy. The results are shown in Figure 3.7. As we see from the plots, while the training on sampled clips gets slightly worse results than the other two settings, i.e. training on full sequence and on both full sequence and augmented data is still feasible. Comparing training on full sequence with and without the augment dataset, we observe improvement on upper two layer. At the lower-level, it helps when the observation ratio is below 60% but degrades the performance slightly afterwards.

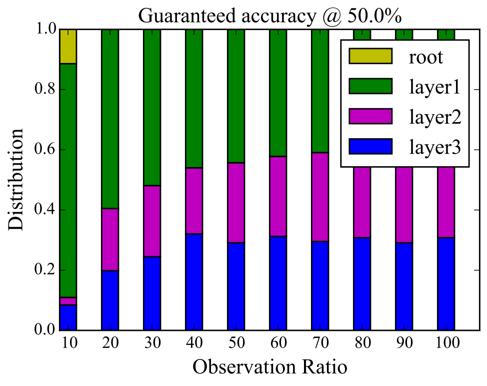
In addition to early recognition setting, we also evaluate the accuracy-specificity trade-off over time as shown in Figure 3.8. We compare to our adaption of the DARTS algorithm as described above as a baseline. As we can see from the expected information gain at fixed target accuracy, training with full sequence and augment data (red curve) most of the time achieve the best reward over the other two, i.e. training on full sequence (green curve) and training on clips (blue curve). To help better understand the concept, Figure 3.8 also shows examples of prediction a distribution over the hierarchy and observation ratios at several target accuracy. The proportion of predictions at lower level grows with time, which means the classifier gets more certain about the activity class over time. When specifying a lower target accuracy, there are more predictions at lower levels in the hierarchy, with higher target accuracy the other way around. In order to reach the target accuracy, the prediction has to move to higher layer that has better confidence.



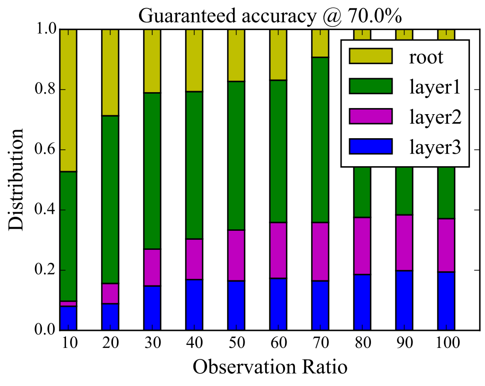
(a)



(b)



(c)



(d)

Figure 3.8: (a): the expected information over time given specific accuracy. (b),(c),(d): example of prediction distribution over time based on the optimization over accuracy-specificity trade-off for train on full and augment segments.

Online Recognition We evaluate online recognition using our proposed RNN model that performs a structured prediction over the label hierarchy (structRNN) and compare it to the DARTS algorithm and a plain RNN. For RNN and struct RNN, we use 50 units for the hidden layer and use a L_2 regularizer. Parameters were selected based on the validation set. We perform dimensionality reduction by using the decision value from linear classifiers trained over different layers in order to reduce the raw feature vector of $20K$ dimension. In a preliminary study we have observed that this generally gives better performance on the expected reward for accuracy-specificity trade-off. The final results are show in Figure 3.9. Compared to the baseline DARTS algorithm, the RNN achieves better performance. This is due to the connectivity between the hidden units that improve the propagation of information along time. The structRNN further improves the RNN performance as we enforce the structural loss with respect to the hierarchy. In addition, we note that our structRNN even outperforms the previously investigated early recognition settings. Figure 3.10 shows some example predictions for videos.

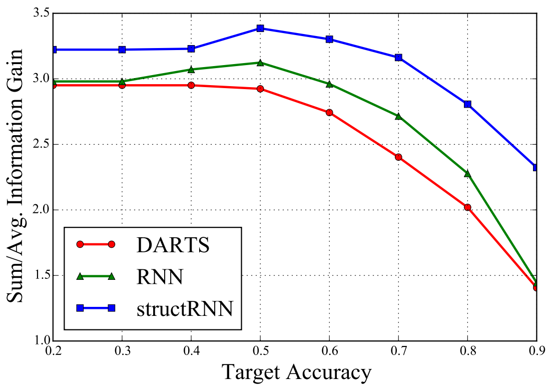
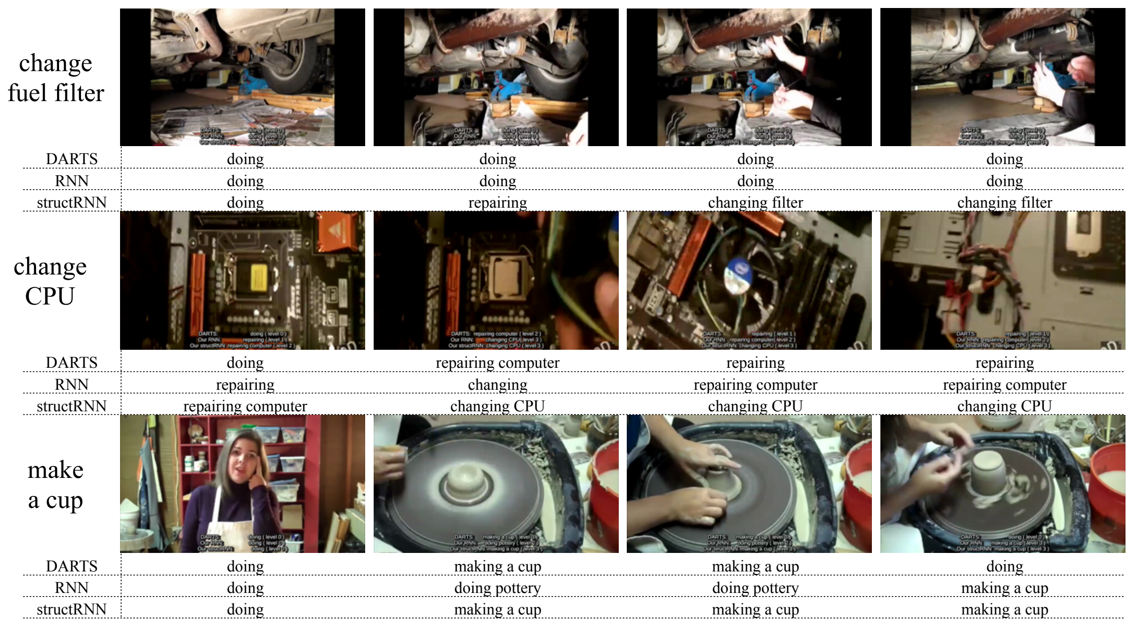


Figure 3.9: Results for online recognition.

3.5 Conclusion

In this paper, we proposed a new challenging dataset with hierarchical labels to study the recognition of long-duration, complex activities and temporal accuracy-specificity trade-offs. We propose a new method based on recurrent neural networks that learns to predicts over this hierarchy and realizes accuracy specificity tradeoffs. Our method outperforms several baselines on this new challenge including an adaptation of hierarchical prediction from object recognition.



change fuel filter				
DARTS	doing	doing	doing	doing
RNN	doing	doing	doing	doing
structRNN	doing	repairing	changing filter	changing filter
change CPU				
DARTS	doing	repairing computer	repairing	repairing
RNN	repairing	changing	repairing computer	repairing computer
structRNN	repairing computer	changing CPU	changing CPU	changing CPU
make a cup				
DARTS	doing	making a cup	making a cup	doing
RNN	doing	doing pottery	doing pottery	making a cup
structRNN	doing	making a cup	making a cup	making a cup

Figure 3.10: Example prediction over three activities in the video, from top to bottom: change car filter, change a CPU and make a cup.

Part II
From Perception to Manipulation

In robotics research, it is a common practice to couple planning with perception. Chapter 4 presents a such set-up for a classic pick-up task grounded in a modern automated warehouse scenario and introduces the robot platform also used in later chapters.

Chapter 5 tackles a more complex manipulation task of tool using. In general, it is nontrivial to formulate a planning model with perception for complex manipulation tasks which often include multiple steps of operations. Our work explores the possibility along this line and proposes a framework for the process. With further introduction of the dynamic movement primitive method, the robot can learn to use tool with one or few expert's demonstrations.

Chapter 4

Participation in Amazon Picking Challenge

Contents

We participated in the first Amazon Picking Challenge in 2015 and qualified for the first stage of the competition. This section starts with a brief description of the challenge itself and then goes through the design of our system for the competition.

After the competition, we have used the same robot setup in later projects discussed in chapter 5 on Teaching Robots the Use of Human Tools and chapter 7 on Visual Stability Prediction for Robotic Manipulation where some of the submodules used for the competition, such as the reaching movement, were also reused there.

This chapter also provides a compact overview of some basic background knowledge on the robotic system, such as the ROS system and motion planning. We integrate related contents in the description of the system architecture.

4.1 Amazon Picking Challenge

The Amazon Picking Challenge (APC) [4] was initiated by Amazon.com to further improve their automated warehouse system. Figure 4.1 sketches a basic set-up for

[4]The challenge began in 2015 and continued in 2016. In 2017, it was renamed to Amazon Robotics Challenge.

current warehouse system in Amazon. In contrast to the traditional warehouse, the items are stored on the shelf system that can be transported by thousands of mobile robots. Upon receiving the order from a customer, the shelves that contain the items within the order can navigate to the picking station where a human operator can pick off corresponding ones into the package, the package then enters the delivery process. Note the process of picking items from shelves into the package still relies on human labor, hence it is essential to come up with a automatic picking mechanism in order to further automate the whole system.

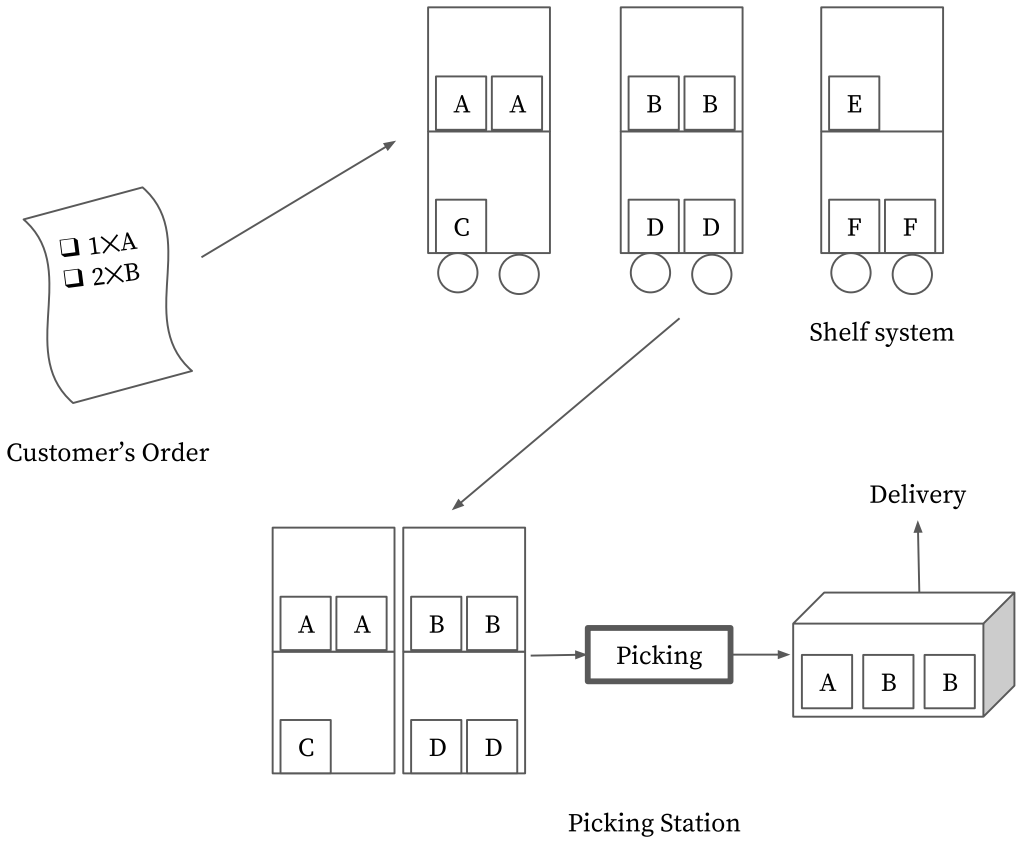


Figure 4.1: Pipeline for processing order: from receiving the customers' order to delivering the package of the ordered the items.

Besides the commercial motivation, the competition also presents some unique challenges to robotics research, not only the system needs to integrate various submodules like object recognition, grasp planning, compliant manipulation and motion planning but also it should be both robust, fast and efficient.

The first APC carried on in multiple stages. In the first stage, the participated team submitted a demo video for their initial system, and qualified teams were awarded with both a shelf unit and a set of practice inventory that will be used in the field competition. In the next stage, the teams brought their robot system to the competition held during two days at the 2015 IEEE International Conference on Robotics and Automation (ICRA) in Seattle, Washington. Each team needs to perform the picking under the rule of the competition and is scored by both the quantity and quality of the picking operations. A review of the challenge can be found in Correll *et al.* (2016). We participated and qualified in the first stage of the competition. From over 30 global submissions only 20 awards were given out to those who showed the most progress and focus towards completing the challenge goals.

4.2 System Architecture

In this section, we first give a brief overview of our system. Then we elaborate on the Baxter research robot from both the hardware and the software perspective. Afterwards, we describe in more details about two major components in our system, namely the segmentation for the perception module and the action primitive for the actuation module.

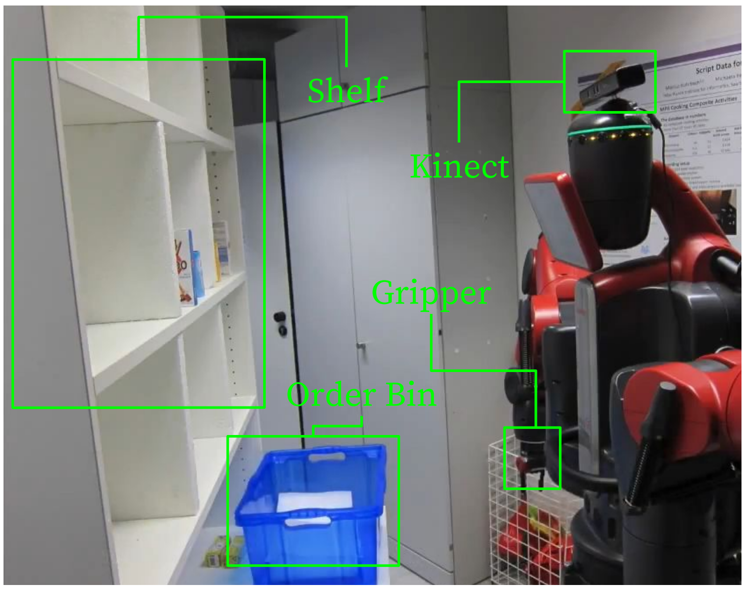


Figure 4.2: Setup for our system in APC.

4.2.1 System Overview

An overview of the set-up for our system is shown in Figure 4.2. More specifically, a wood shelf is used for the **shelf** unit and a blue plastic basket is used for the **order bin** which holds the item picked off from the shelf. For the operator, we use a Baxter research robot equipped with a head-mounted first generation **Kinect**. The Kinect is used for the perception to detect and locate items on the shelf and the robot's parallel **gripper** performs the actual picking operation.

The goal for our system by the time of submission is to correctly localize items and perform reach, pick-and-place operations. This is done without the recognition of specific items and the planning of picking a series of items as the full picking process discussed earlier shown in Figure 4.1. Our main purpose is to showcase that we can implement a relative complete system to achieve most parts of the whole pipeline.

4.2.2 Baxter Research Robot

Baxter is a humanoid, anthropomorphic robot produced by Rethink Robotics. It is equipped with two seven degree-of-freedom arms as shown in Figure 4.3, each with the sensor readings of force, torque and position at every joint. Moreover, there are three built-in cameras: one on the head, one of each on the wrist per arm. All these components are connected to the built-in computer inside the torso. To control the robot, we have an additional workstation connected to the computer. Other external sensors such as the Kinect and any other cameras or sensors can be connected to the workstation with communication to the robot control system.

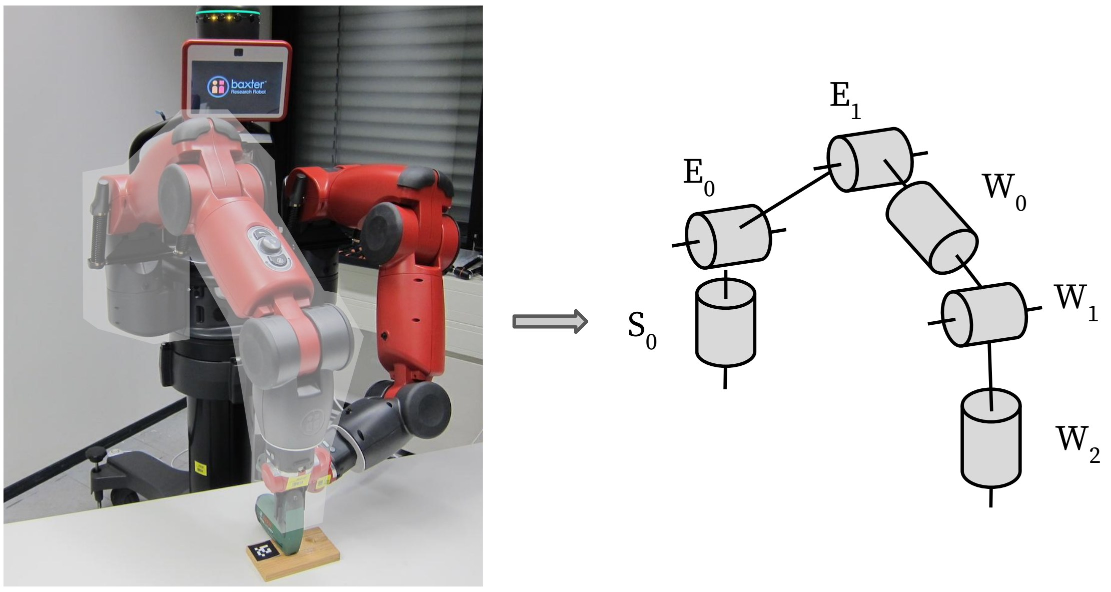


Figure 4.3: The arm joints in Baxter robot.

The communication between different components (including the built-in computer and the external workstation) are done via the **Robot Operating System** (ROS) as a peer-to-peer network. Each component is a **node** in the network where the built-in computer acts as the **master node** providing lookup for nodes to find each other. The overall topology of the network for Baxter robot is shown in Figure 4.4. While there still exists some proprietary programming frameworks for certain robots, more and more robots nowadays are integrated with ROS support.

To better understand how the communication is done in the network, a few more important concepts need to be introduced, namely the messages, the topics and the services. As shown in Figure 4.5, information circulates in the network in the form

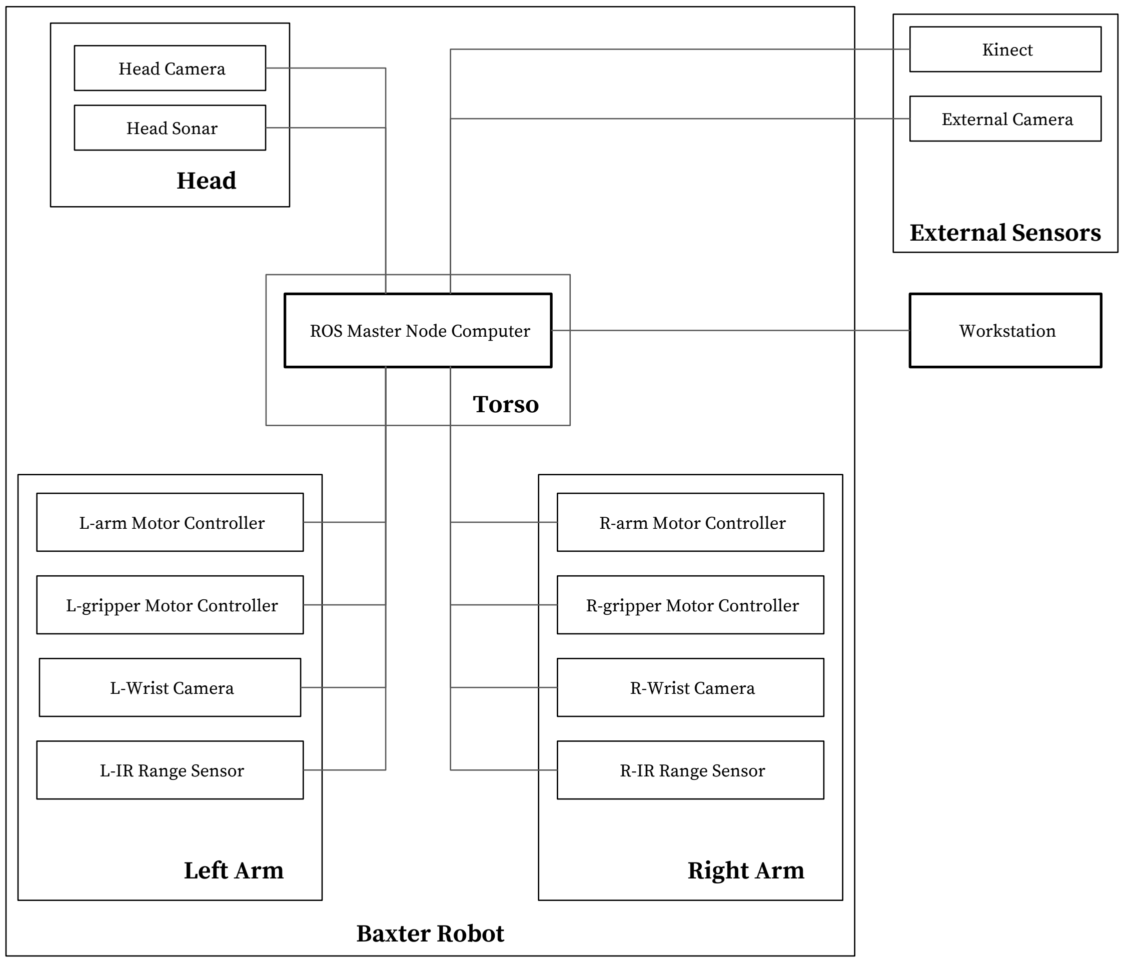


Figure 4.4: Components in Baxter robot.

of a **message**, it can be a reading from the sensor, such as the point cloud from the Kinect, or it can also be the control command to a arm-joint to move by a certain angle. A node can **publish** its message to a given topic, the master node provides information of available published topics, and then other node that is interested in data of specific topic can **subscribe** to it as shown in Figure 4.5a. Although the topics model is very flexible, it is not very appropriate for request/reply interactions. Hence the **service** model is created. As shown in Figure 4.5b, a providing node offers a service, the master node provides information on available services, and then the other client nodes can request it on demand. One such example is the inverse kinematics planner, which is implemented as a service for the Baxter robot. Given a desired location of the gripper, the service computes the rotation angle for each arm joint.

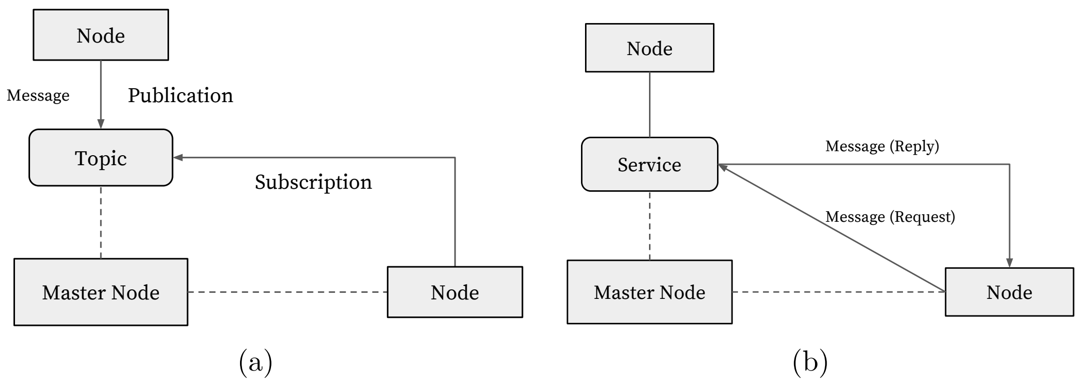


Figure 4.5: (a): ROS topics model. (b): ROS services model.

4.2.3 Perception Module

The perception is implemented by a first generation Kinect mounted on the head of the robot as shown in Figure 4.2. The objective for the module is to detect and localize the object in 3D coordinates so that the actuation module can plan pick-and-place operation afterwards. An overview of the perception module is shown in Figure 4.6.

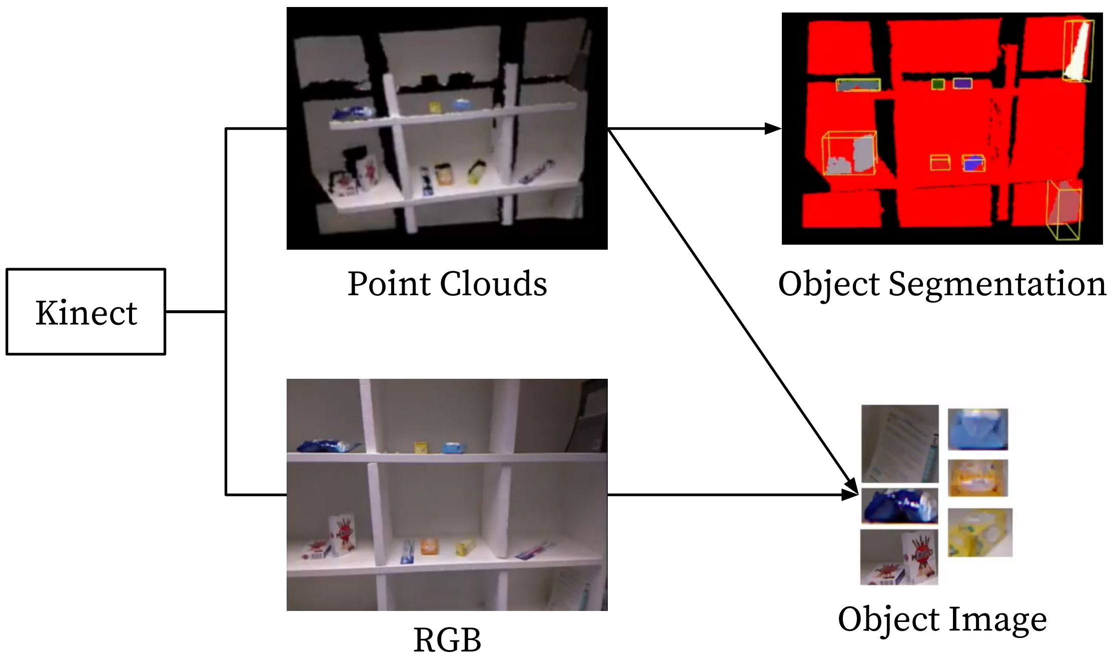


Figure 4.6: An overview for the perception module.

The module is implemented with Point Cloud Library (PCL) [5] with ROS integration. Both the RGB image and the corresponding point clouds from the Kinect are

[5]http://pointclouds.org/

used for the process. At the beginning we capture the scene of the shelf without any items as the *background*. Then upon a picking scene with actual items on the shelf, we can perform background subtraction to get the point clouds for the foreground objects, i.e. the items on the shelf. Afterwards, we extract clusters from the foreground point clouds with the `pcl::EuclideanClusterExtraction` class. Note due to the noise in the point cloud, it is often necessary to filter out some small clusters so that each of the obtained clusters approximately corresponds to a different item on the shelf. From these clusters, we can estimate the item's volume by a 3D bounding box with the `pcl::getMinMax3D`. The pick-and-place movement is then planned with respect to the estimated 3D bounding box. Meanwhile, with the correspondence of the point clouds and RGB image, we can trace the extracted point cloud clusters to their corresponding regions in the RGB image. Combined with image-based classifier, we can further recognize the category of the item.

4.2.4 Actuation Module

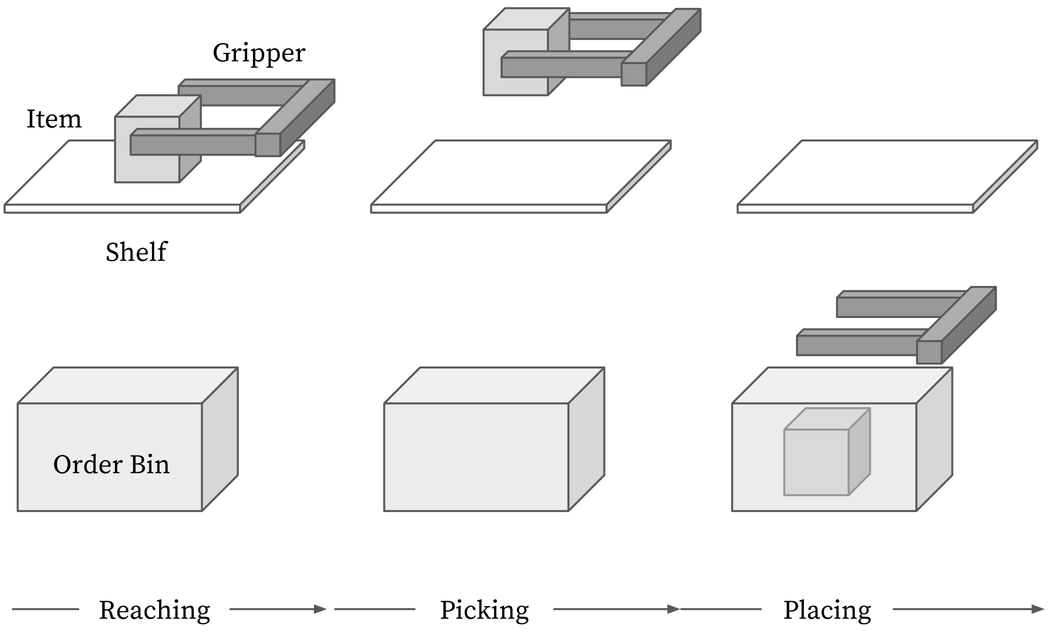
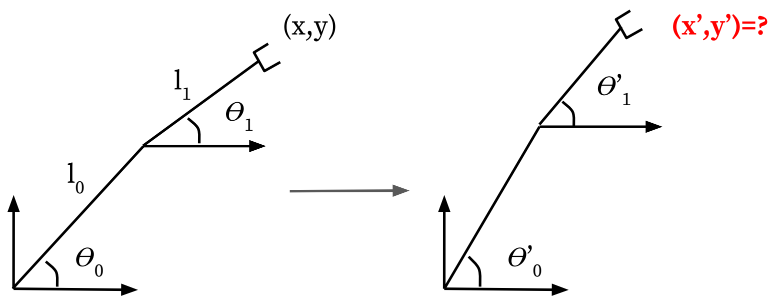


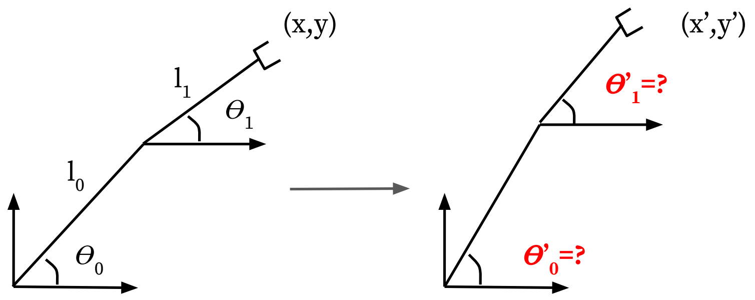
Figure 4.7: An overview for the actuation module.

Once the perception system obtained the bounding box of an item, the actuation module is relatively straightforward. The movement is decomposed into three stages: 1) reach a pre-grasp position where gripper can firmly grasp the item, 2) close the gripper to pick the item and lift it up, 3) move over the order bin, release the gripper to put the item into it. An overview of the process is shown in Figure 4.7.

Reaching. One key component to accomplish the module is given a location and the robot arm should plan its movement for individual joints so that the end-effector, i.e. the gripper can reach the specified location. This applies throughout the process, such as the reaching the pre-grasping location and moving to above the order bin. This is done by requesting the Baxter's built-in `IKService` (Inverse Kinematics Service). Solving **Inverse Kinematics** is an important task in robotics, it refers to the procedure we have just described, i.e. given a desired position to derive the moving angle for each joint. The opposite direction that given the moving angles for each joint to derive the resulted position is called **forward kinematics**. Figure 4.8 shows a simple example of 2-joints robot arm in 2D. The arm's link lengths are l_0, l_1. The initial position of the gripper is at (x, y) and initial joint angles are θ_0, θ_1. In forward kinematics as shown in Figure 4.8a, given new joint angles θ'_0, θ'_1, it derives the corresponding new location at (x', y'). In inverse kinematics shown in Figure 4.8b, given the goal position at (x', y'), it derives the corresponding angles of (θ'_0, θ'_1).



(a)



(b)

Figure 4.8: (a): Forward kinematics. (b): Inverse kinematics.

Picking. As described in the perception module, each object is represented by a 3D bounding box of its volume, the gripper is then aligned with the center of the bounding box after the reaching step. To pick the object, the robot simply closes the grippers and lifts the arm. Note, the grippers is set to a specific grip force level so that they will not pull harder once the object is held tight enough to the specified force level. This intends to avoid potentially damage to the picked object or the mechanical structure of the gripper.

Placing. To put the picked object into the order bin, the same control module is applied to the arm, only this time, the target location is a predefined location relative to the order bin. Once reaching the location, the grippers are released so that the object is dropped into the bin.

An example of the complete actuation process is shown in Figure 4.9.

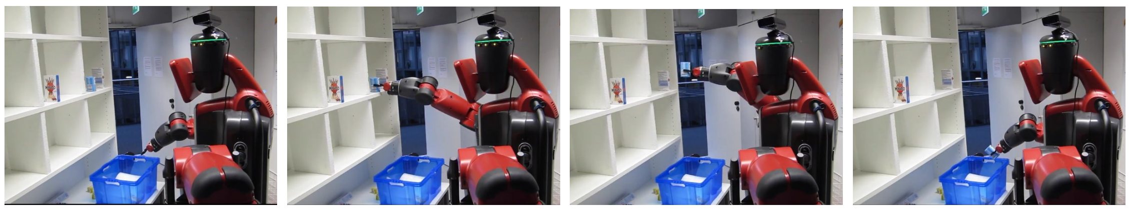


Figure 4.9: Example picking process.

4.2.5 Limitations

The main limitations for the current system come from the approximated representation of the 3D bounding box for each individual object through the Kinect sensor. First of all, the relatively low resolution of the Kinect sensor makes it very difficult to detect small objects, such as a single pencil or eraser. Further, with current method, it is infeasible to detect individual object with the presence of clutter, i.e. there are multiple objects occlude each others in the view. In addition, the variation of objects' sizes can also challenge the system, for example, if the object is wider than the gripper can open, then it is impossible to pick the object.

To overcome the aforementioned limitations, the system needs to adopt both a more advanced perception module and a more capable actuation mechanism. As discussed earlier for the picking challenge, the setup itself presents some unique challenges in robotics, and it is an ongoing effort to construct a practical system.

Chapter 5

Teaching Robots the Use of Human Tools from Demonstration with Non-Dexterous End-Effectors

Contents

In the previous chapter, we went over our set-up for the classic pick-and-place task, however to cope with more complex manipulation task, we need to further devise models with respect to its characteristics. Hence in this chapter, we explore along this direction by focusing on a tool using scenario.

Commercial, affordable and general-purpose robots like the PR-2, Baxter and UBR-1 robots can take over a wide range of tasks or assist human workers in a mixed human-robot environment. However, end-effectors on these robots are usually restricted to low-cost, non-dexterous grippers which constrains the application scenarios. In this work, we levitate this limitation via a dual-gripper strategy in replacement of the much less widely deployed dexterous hand for tool manipulation.

We present a novel and compact model for the use of human tools and propose a hierarchical architecture to embed tool use in a learning from demonstration framework, learning temporal order for dual-arm coordination at higher level and Dynamic Movement Primitives at lower level for a multi-step execution. The approach is demonstrated and evaluated on a Baxter research robot for three human tools.

This work has published in Humanoids (Li and Fritz, 2015).

5.1 Introduction

Humans use tools to extend their reach, to amplify their physical strength, and achieve many daily tasks, making tool use a very important aspect of human life. Being able to use tools is generally interpreted as a sign of intelligence. In contrast, even with the advent of commercial, affordable, general purpose robots, that start to penetrate human work places, their use cases are often restricted to simpler activities like pick-and-place actions. On the other hand, industrial robots use highly customized tools for much more sophisticated tasks like welding, cutting and painting and generally require experts to carefully script each step of a fixed procedure. We aim at narrowing this gap by exploring the possibility to teach affordable, general-purpose robots to use human tools in an easy way.

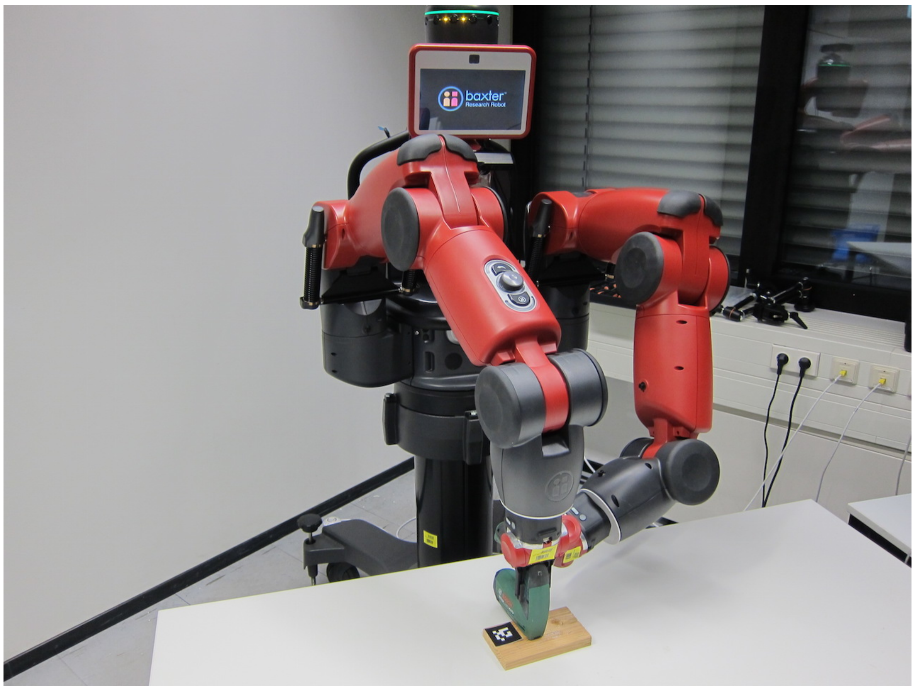


Figure 5.1: Example of manipulating a human tool: Our approach enables the Baxter research robot to use an electric tacker.

We envision a system that allows the end-users to intuitively and easily teach robots, which is key in facilitating wide spread deployment of robots into real world application. We approach this challenge by the *Learning from Demonstration (LfD)* framework (Billard *et al.*, 2008; Argall *et al.*, 2009): the robotic system observes a teacher's demonstration, automatically derives a representation of the activity and applies them to novel situation afterwards. One of the challenges of applying LfD to learning tool use is that manipulating a tool is generally a multiple-step process. It is non-trivial to separate the individual steps and properly model the stepwise operation. Building on recent work of *Dynamic Movement Primitives (DMPs)* (Ijspeert *et al.*, 2002; Pastor *et al.*, 2009), we propose a formulation to embed tool use in a Learning from Demonstration framework.

Following the LfD framework, we need to encode the interaction patterns between the tools and human hands and transfer them to the robot's end-effectors. Dexterous hands featured on robots such as Honda's Asimo and NASA's Robonaut could follow this approach, since little changes are required to transfer the interaction pattern. Yet the hardware cost and limited deployment of these robots makes this approach less accessible. In contrast, the mechanical grippers equipped on low-cost robots are much more affordable and more widely deployed. Hence, we explore a strategy of utilizing dual-grippers to replace the dexterous hand for tool manipulation. We believe that this approach will significantly increase the range of tasks that can be performed by today's most widely deployed general purpose robots.

In the following, we first review related work about tool use and tool use in robotics and LfD and its applications. Then we discuss our novel approach to modeling tool use and express it in a LfD framework. We test and evaluate our approach on a Baxter research robot by learning and using three tools.

5.2 Related Work

Early work on tool use in animals can be found in the work by Beck (1980). Yet until recently, there are few studies of autonomous robotic tool use. In Stoytchev (2005), a behavior-grounded approach is proposed for tool representation by connecting the tool's affordance with feasible exploratory behaviors. While the idea is appealing, it is only tested in a simplified setting, i.e. like scoring with a hockey puck and different shapes of sticks. Kemp and Edsinger (2006) address the modeling of the tool's tip as the task relevant feature. Later work (Edsinger and Kemp, 2007) from the same group combines tip detection and tracking to model tool use, and test it on a brushing task. However, there are clearly more circumstances where the procedures of tool use are beyond the pure movement of tool tip including examples, like screwdriver and hot-glue gun used for tip detection in Kemp and Edsinger (2006). Our tool use model works with the similar type of tools and can tackle more complex instances by only using non-dexterous grippers.

There is a large body of work on LfD to program robots. Contrary to traditional approaches to robot control with domain dynamics and mathematically derived policies, LfD typically acquires the polices from demonstration, opening the policy development to non-robotics-experts. An overview of the topic can be found in Billard *et al.* (2008) and Argall *et al.* (2009). Among many approaches in LfD, DMPs

(Ijspeert *et al.*, 2002) for motor skill learning is widely used due to its compactness and efficiency. Most related to our work, Tamosiunaite *et al.* (2011) demonstrated how to learn to pour water using DMPs and in Niekum *et al.* (2013), a variant of HMM and DMPs are applied for LfD to assembling an Ikea table. Our work also builds on a DMP formulation and explores extensions to model tool use.

Learning complex tasks with a single policy can be challenging. A common approach to handle this issue is to segment the complex task into multiple simpler sub-tasks where each sub-task can then be formulated in a established LfD framework. While it is appealing to automate the segmentation process and a number of attempts have been made towards this direction (Grollman and Jenkins, 2010; Chiappa and Peters, 2010; Konidaris *et al.*, 2011; Niekum *et al.*, 2012), the unsupervised nature of these proposed methods cannot guarantee they will spot the exactly transition states which are often crucial to structured complex tasks. Hence in our work, we align different sub-tasks with the activation states' changes between the robot' arms and model each sub-task in a DMPs framework.

A line of relevant work are described in Aksoy *et al.* (2011) and Aksoy *et al.* (2015) where the authors segmented tasks with Semantic Event Chain (SEC) by touching-nontouching relation. It is similar to our tool-tip model to decompose tool-use process by contact states, yet we aligned this with states' change between the dual-arm coordination, and together we present a efficient strategy for robot with dual-gripper to use human tools over dexterous hand.

5.3 Method

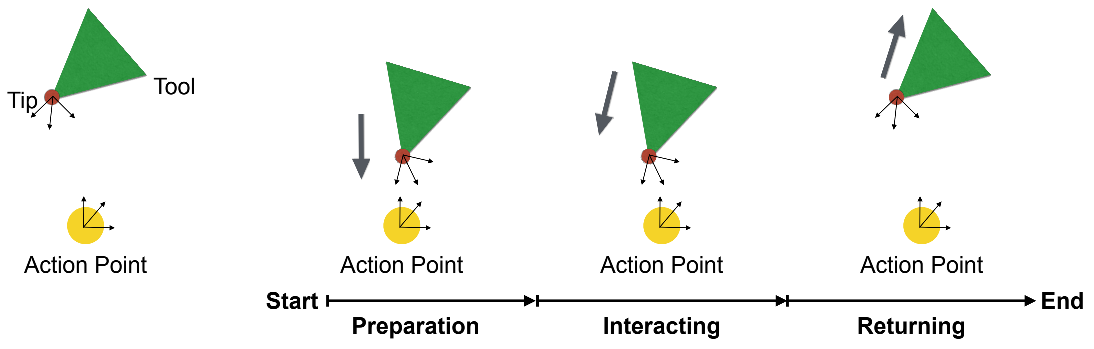


Figure 5.2: Illustration of our tool-tip modeling where it decompose the process into three stages: preparation, interacting and returning. The arrow sketches the moving direction of the tool tip.

5.3.1 Compact Modeling of Human Tools

While there are various kinds of common tools, a large body of them can be characterized by interaction patterns between the world and a tool tip (Radwin *et al.*, 1996; Kemp and Edsinger, 2006). The general process to operate such tools can be decomposed into three stages, including preparation, interaction and returning to standby as illustrated in Figure 5.2. In the *preparation* stage, the tool is moved to align the *tip* and the *action point* on the object. In the interaction stage, the tip of tool makes contact with the action point, triggering related mechanisms with respect to the functions of different tools. In the returning to standby stage, the tool tip is detached from the action point. This decomposition is consistent with the widely used definition of tool use from Beck (1980), for example the preparation stage includes the requirement that "user is responsible for the proper and effective orientation of the tool" and the interaction stage is a compact representation for this type of tools as "an unattached environmental object to alter more efficiently the form, position, or condition of another object, another organism...".

5.3.2 Robot Manipulation of Human Tools

Although industrial robots use tools, those tools are in general highly customized and the operating procedure is typically carefully scripted and fixed. There are various issues making programming a robot to manipulate human tools very challenging: (1) as discussed in related work, the process of manipulating a tool often goes beyond a single primitive movement. To construct a complete model for tool use, one approach is to apply the model-based approach built on the tool's affordance. Yet the modeling of affordance itself is non-trivial and often such modeling relies on the perception of the tool where the uncertainty of perception could bring about additional difficulties. (2) the design of human tools are usually optimized for human hands which are extremely dexterous.

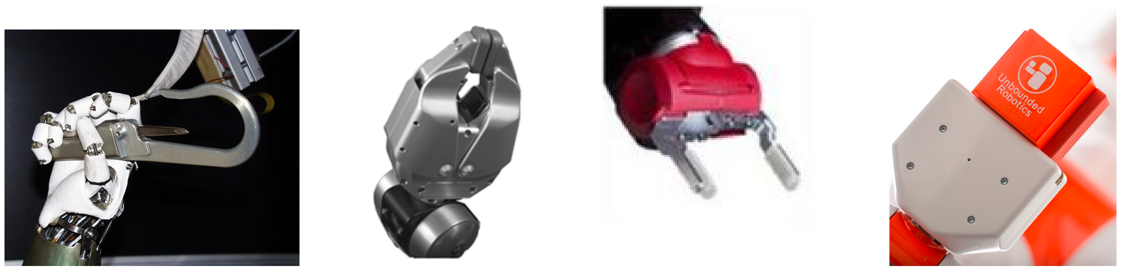


Figure 5.3: End-effectors on different robots, from left to right: dexterous hand on NASA's robonaut, PR2's jaw gripper, Baxter's parallel plate gripper and UBR1 parallel jaw gripper

The most straightforward way to transfer the operating manner from human to robot would be a close and complete mapping of finger-palm movement to a dexterous hand end-effector on a robot like the NASA robonaut (Figure 5.3). Nevertheless such end-effectors are extremely expensive and not widely deployed to date.

To cope with the aforementioned issues, we make carefully design choice for our method as follows:

5.3.2.1 Model-free Approach for Tool Use

Unlike the tool-specific affordance modeling as in the model-based approaches, we turn to a general, tool-agnostic paradigm for tool use modeling. With our tool-tip model, the action point P and the state of the tip T are tracked. Note we do not actually track the tip of the tool as in Kemp and Edsinger (2006) but the end-effector with the tool under the assumption that the tool is held firmly by the end-effector with the same configuration. Hence the process of using a tool X is represented as sequence of state changes in time:

$$\text{ToolUse}_X := \{P, T\}_t$$

where t denotes time stamp for each state, P for the pose of the action point, T for the state of the end-effector, including the pose and other information like the closure of the gripper depending on the type of the end-effector.

5.3.2.2 Dual-Gripper Coordination for Complex Manipulation

With the rapid development in robotics, there is a trend that has supplied affordable robots to industry and research communities. Typical examples like the PR-2, Baxter and UBR-1 robot are equipped with reasonably low-cost grippers as shown in Figure 5.3. Although the much more expensive dexterous robot hands offer more flexibility, we can actually decouple the manipulation primitives and propose to achieve the complete operation using two arms with simple, non-dexterous grippers, i.e. with one gripper to hold the tool and the other to do additional operations, like pushing button. One such example is shown in Figure 5.4. Note that this can introduce additional step in the preparation phase like in the case of the electric tacker in Figure 5.4. While the first end-effector is holding the tacker, the second end-effector needs extra alignment step in order to finish the preparation stage.

Similar to the required coordination of different fingers in the dexterous hand system, the dual-gripper also needs proper coordination. This coordination reflects the temporal order in manipulations between different grippers which is essential to tool use and is typically asymmetric as discussed in Guiard (1987); Zöllner *et al.* (2004). Yet explicit condition/event modeling as in Zöllner *et al.* (2004) is not needed in our approach, as the temporal order between the two gripper can be included in the model-free formulation by replacing the single end-effector T with the gripper pair T_1, T_2. Hence we formalize the overall process as:

$$\text{Tool Use}_X := \{P, T_1, T_2\}_t$$

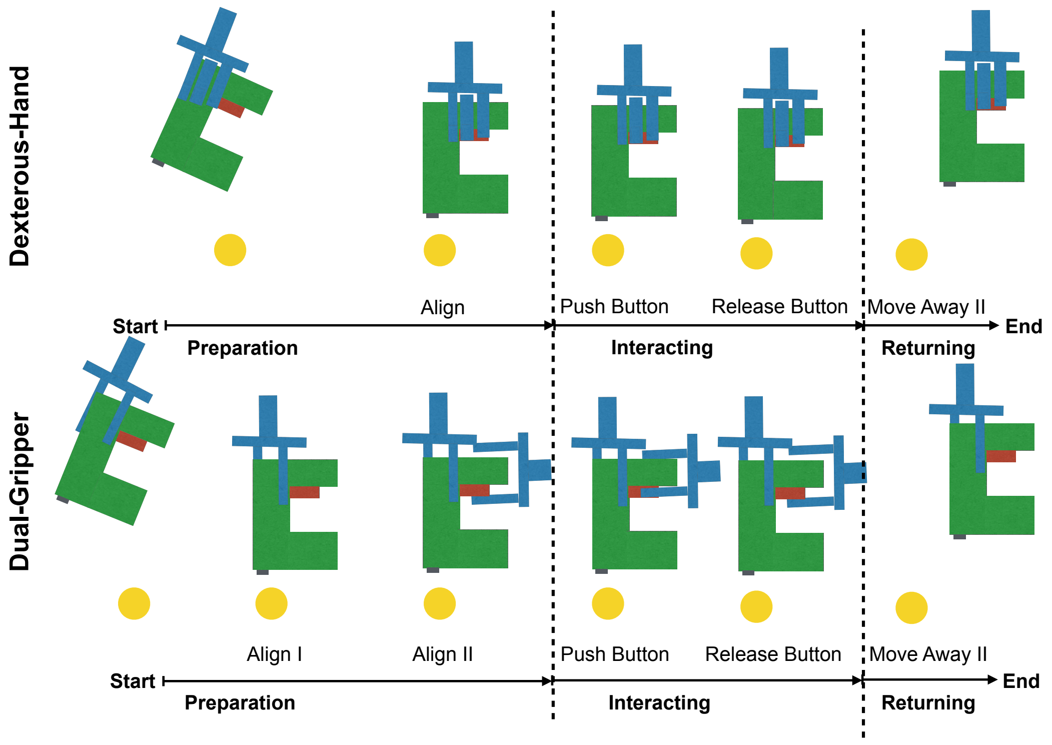


Figure 5.4: Example of manipulating an electric tacker with dexterous hand and a dual-gripper.

5.3.3 Learning Tool Use from Demonstration

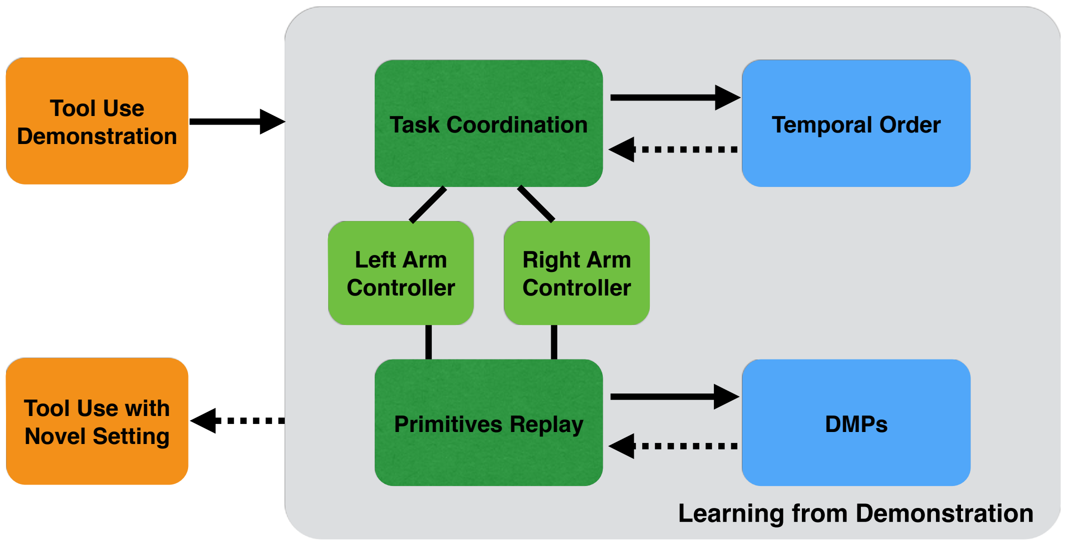


Figure 5.5: Overview of our approach. The solid arrow denotes the learning process from demonstration, the dashed arrow for the process of replaying on novel task.

We apply a hierarchical architecture to embed tool use in a learning from demonstration framework: on a higher level, temporal order for dual-arm coordination is learned and on the lower level, primitives are learned by constructing DMPs from exemplars. The pipeline is shown in Figure 5.5.

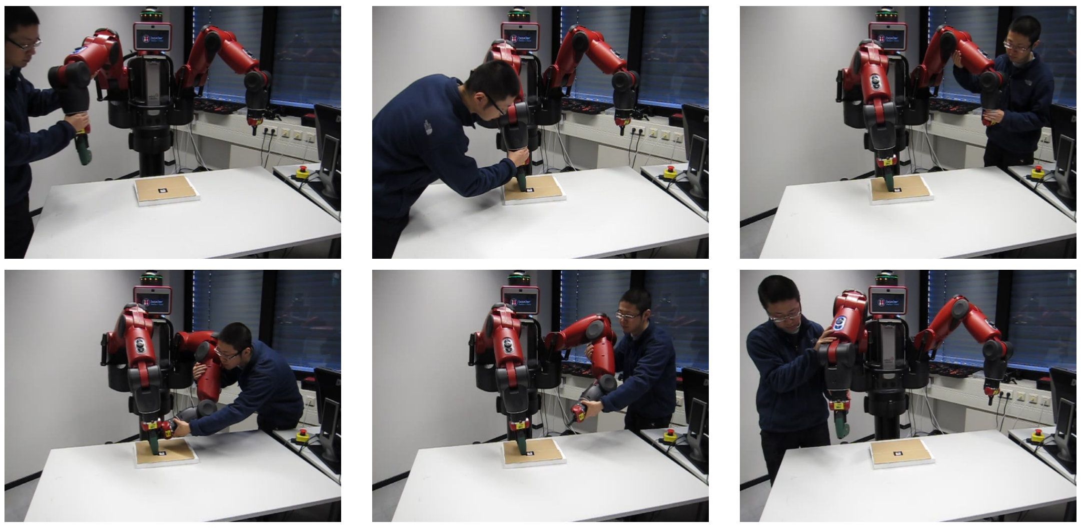


Figure 5.6: Kinesthetic demonstration to use an electric tacker.

5.3.3.1 Temporal Segmentation for Manipulation Primitives

We teach the robot to learn the use of tools from kinesthetic demonstration, i.e. the teacher physically moves the robot's arm to perform manipulation as shown in Figure 5.6. In addition to the state of end-effectors T_1, T_2 and action point P, we track the activation state of each arm as S_1, S_2 respectively. The activated periods for each S_1, S_2 naturally segment the whole process into *Manipulation Primitive (MP)*:

$$\mathrm{MP}_{S_i,\tau}, i \in \{1, 2\}$$

where τ denotes the temporal interval of the corresponded activated period for MP. S_1, S_2 together encode the temporal order of manipulative primitives between two arms. We assume only one arm is activated at a time. An example is shown in Figure 5.7. The MPs are in line with our tool use model shown in Figure 5.2 except for the part of additional alignment step introduced by the dual-gripper design.

5.3.3.2 DMPs for Manipulation Primitives

Dynamic Movement Primitives (DMPs) (Ijspeert *et al.*, 2002) describe the evolution of dynamical systems over time using a system of non-linear differential equations.

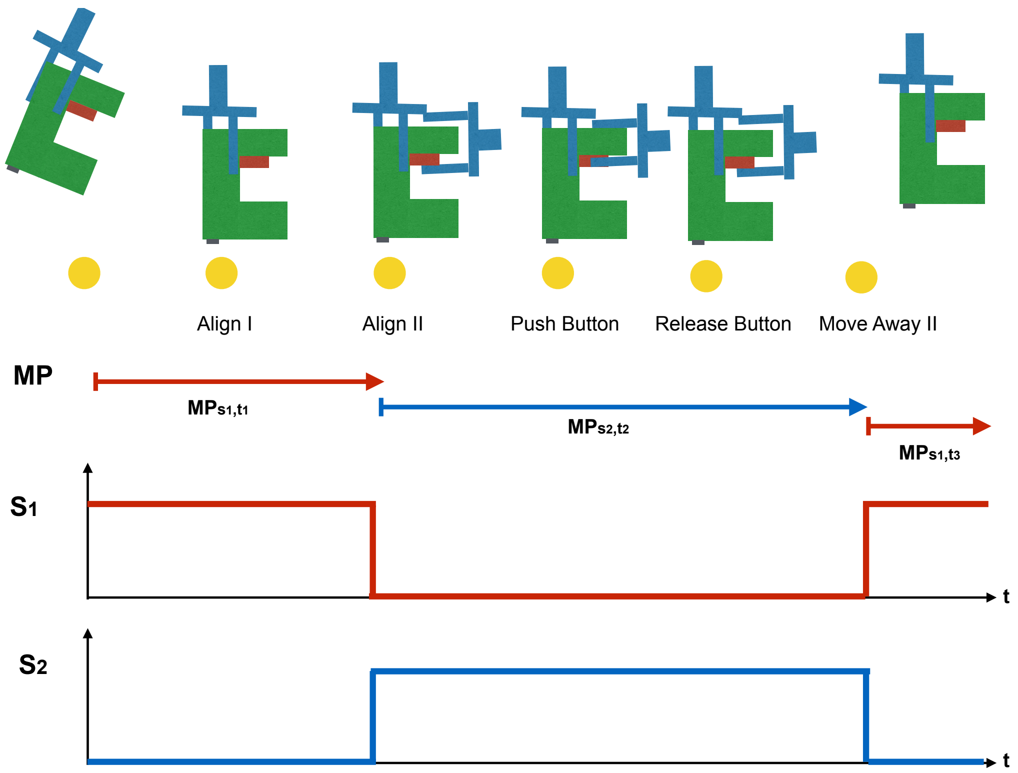


Figure 5.7: An illustration of the LfD approach. S_1, S_2 represent the activated states for end-effector 1 and 2, MP for Manipulation Primitive.

In general it is nontrivial to represent the whole tool use process with a single DMP since it involves muti-stage operations, each with a distinct contraint between the stage-transition, yet we can encode the Manipulation Primitive at each stage with a DMP. In our study, we are only interested in the formulation for discrete movements and apply an improved version (Pastor *et al.*, 2009) formulated as follows:

$$\begin{aligned}\tau\dot{v} =& K(g-x) - Dv - K(g-x_0)s + Kf(s)\\ \tau\dot{x} =& v\\ \tau\dot{s} =& -\alpha s\end{aligned}$$

where x and v are the position and velocity of the system, x_0 and g are the start and goal position, τ is the temporal scaling factor, K is a spring constant, s a phase variable, and D a damping term. The non-linear function $f(s)$ defines the shape of the movement and is approximated by a weighted set of basis functions $\phi_i(s)$. Compared to the original DMPs formulation, it better adapts the movement to a new goal position by changing the goal parameter g (Pastor *et al.*, 2009). We use Scott Niekum's DMP implementation [6], where $f(s) = \sum_{i=1}^{N} w_i\phi_i(s)s$ is approximated

[6]https://github.com/sniekum/dmp

by the univariate Fourier basis (Niekum *et al.*, 2012) and the target function is formulated as:

$$f_{target}(s) = \frac{-K(g - x(s)) + D\dot{x}(s) + \tau\ddot{x}(s)}{g - x_0}.$$

Given a demonstration trajectory $\{x(t), \dot{x}(t), \ddot{x}(t)\}$, we can then learn a set of values for the weights w_i (Ijspeert *et al.*, 2002). The spring and damping constants are set to ensure critical damping.

In the case of tool use, the goal positions $T = \{T_1, T_2\}$ of the two end-effectors are coded in the coordinate frame of the action point P and in order to execute the DMPs in a novel situation, the goals are then shifted based on the coordinate frame of the new action point P' as $T' = \{T_1', T_2'\}$ as shown in Figure 5.8. Let $M(i, T_i')_\tau$ be the model learned on the temporal interval τ for end-effector i over T' (recall the notation S_i for the activation signal for end-effector i) then the overall tool use process in LfD framework is defined as:

$$\text{ToolUse}_X := \{M(i, T_i')_\tau, S_i\}$$

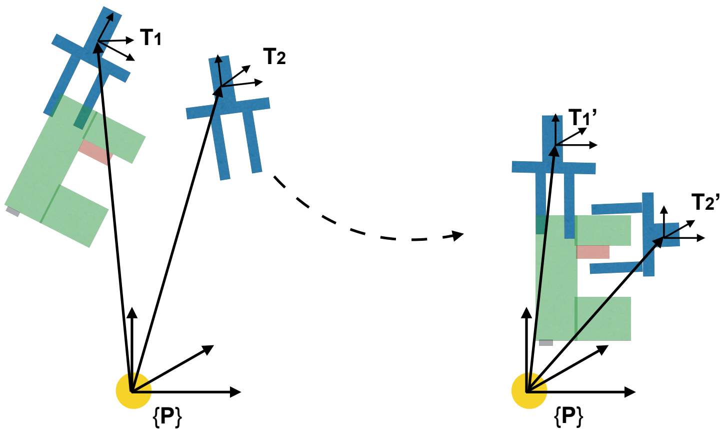
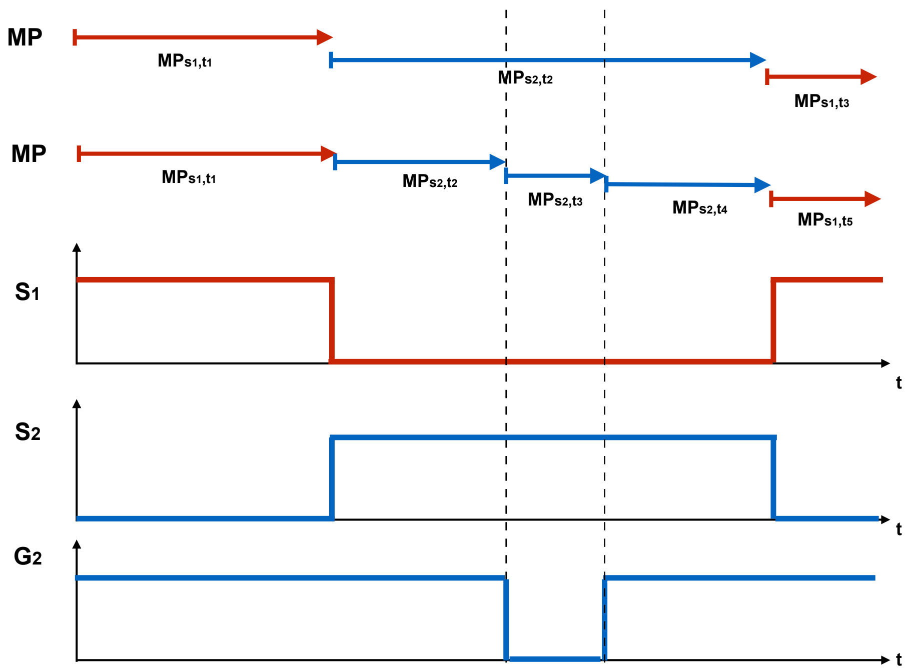


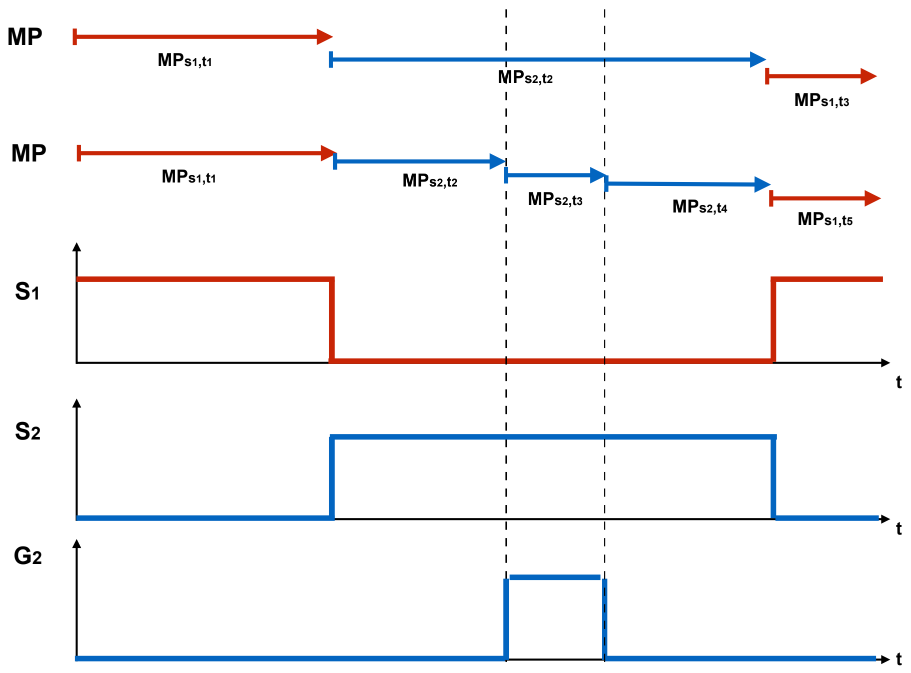
Figure 5.8: The geometry of the tool use model. The trajectory of both end-effectors T_1, T_2 is transformed into the coordinate system of the action point $\{P\}$, encoding the goal position $T' = \{T_1', T_2'\}$ in our tool use model.

Further, the second manipulation primitive that interacts with the tool has to be segmented into three primitives, i.e. 'before interaction', 'during interaction' and 'after interaction', in the case of using the tacker, these correspond to 'move the second end-effector to the button', 'push the button' and 'release the button $\rightarrow$ move away the second end-effector' [7].

[7]the second primitive 'during interaction' is a trivial one since it only models how long the gripper opens or closes; we keep it as a constant in our experiments



(a) gripper changes from ‘open’ to ‘close’, then ‘open’.



(b) gripper changes from ‘close’ to ‘open’, then ‘close’

Figure 5.9: Additional segmentation of manipulation primitives introduced by the state changes of the gripper compared to the original segmentation shown in Figure 5.7. S_1, S_2 are the activation signal for end-effector $1, 2$, G_2 is the gripper’s open/close state for end-effector 2.

This is due to the nature of the DMP model — DMPs allow the generalization of different paths between the starting and goal position. If it is applied to model the whole primitive from ‘move the second end-effector’ to ‘move away the end-effector’, there is no guarantee that the end-effector will be pushed at the same location relative to the tool as in the demonstration. In our settings, this segmentation is made by the identifying the state changes of the gripper during demonstration as shown in Figure 5.9, including two pattens, i.e. the gripper changes from ‘close’ to ‘open’ and from ‘open’ to ‘close’.

5.4 Experiments

We implement our system in ROS (Quigley *et al.*, 2009) and test it on a Baxter research robot by learning to use three different tools: a tacker, a glue-pen and a drill as shown in Figure 5.10.

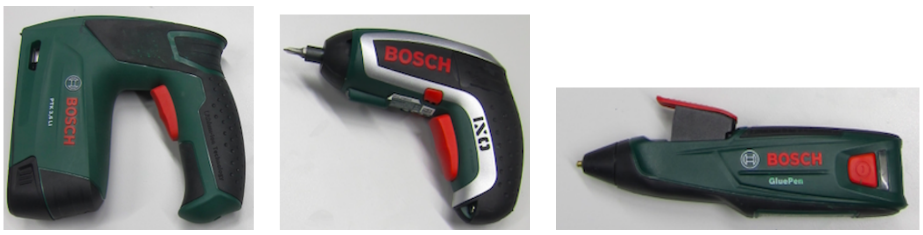


Figure 5.10: Tools used in our experiments, (left) an electric tacker, (middle), an electric drill and (right) a hot-glue pen.

Example tasks are provided to the robot via kinesthetic demonstration, in which the teacher physically moves the robot’s arm in zero-gravity mode to perform the task and uses the button on the cuff to set the closure of the grippers. On pushing the button on a arm, the recording begins, the teacher starts to move the same arm to perform manipulation. When the manipulation is done, the teacher presses again the button to pause the recording. To continue the manipulation with another hand and the recording, the teacher simply repeats the steps. The signals of arm activation and the grippers’ state during the demonstration are recorded to segment the tool use process into sequential manipulation primitives, where each primitive is characterized by a starting pose, an ending pose of the actuated end-effector and the sequence of the poses during the primitive. The primitives are learned via the DMPs framework. These primitives and the sequencing of the primitives constitute the model for tool use.

At test time, the tool use model is replayed on novel configurations, generating a sequence of primitives, where each primitive’s starting pose and ending pose are adjusted according to the action point. For both demonstration and test time, action points are tracked with AR tag as in Niekum *et al.* (2013) using an ASUS Xtion Pro Live sensor mounted on the robot.

5.4.1 Experiment 1: Learning to use a hot-glue pen

We first evaluated our system on learning to use a hot-glue pen. During the demonstration, the teacher moves one of the robot's arms holding the glue pen to the action point T, until there is only a small distance vertically between the tip and T. Then the other arm is moved to reach the button of the glue pen, and presses it with the gripper to release the hot glue. Afterwards, the arm pressed the button releases the gripper and returns to a neutral position. At test time, given an action point, the robot is required to (1) position the glue pen, (2) correctly press the button, (3) release button and (4) return to the neutral position. A failure case is counted whenever the robot fails to finish the whole process, for example, one arm fails to reach the button and hence cannot press it to use the tool. One successful run and corresponding signal sequences are shown in Figure 5.11.

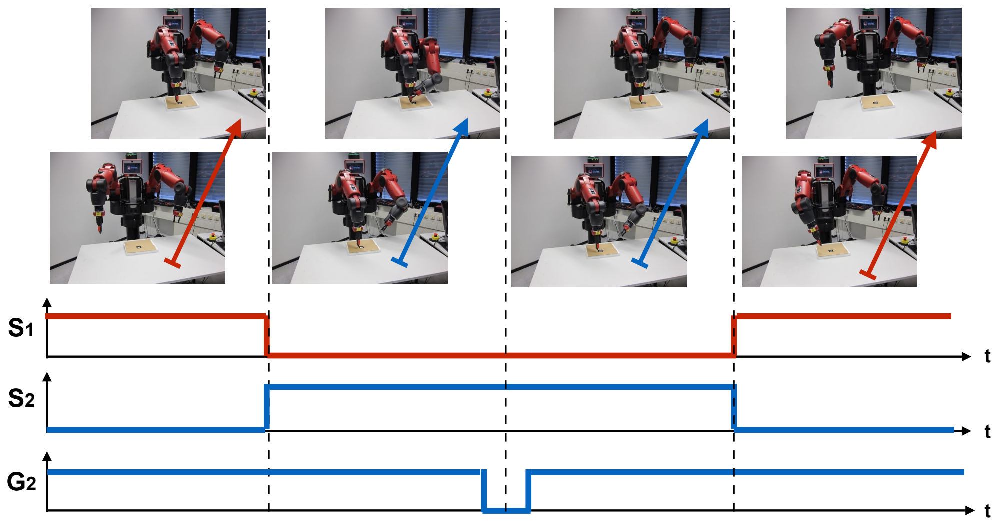


Figure 5.11: Steps of robot using a gluepen after learning. S_1, S_2 are the activation signals for robot's arm $1, 2$, G_2 is the gripper's open/close state for arm 2.

5.4.2 Experiment 2: Learning to use an electric drill

Next, we evaluated our approach for an electric drill. Ideally, one would expect the tip of the drill to go into the contact surface, yet for the purpose of our experiments we refrain from penetrating the target. Thereby we make an alternative design for this experiment, similarly to the glue pen, we aim the tip of the drill to be correctly positioned right above the action point. At test time, given an action point, the robot is required to (1) position the drill, (2) correctly press the button, (3) release the button and (4) return to the neutral position. A failure case is counted when the robot fails to finish the whole process. One successful run and the corresponding signal sequences are shown in Figure 5.12.

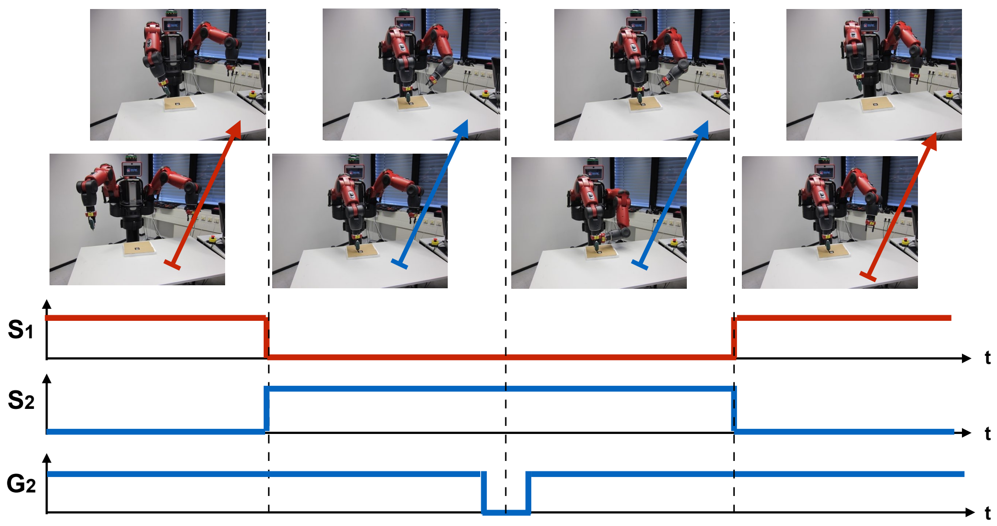


Figure 5.12: Steps of robot using a drill after learning. S_1, S_2 are the activation signals for robot's arm $1, 2$, G_2 is the gripper's open/close state for arm 2.

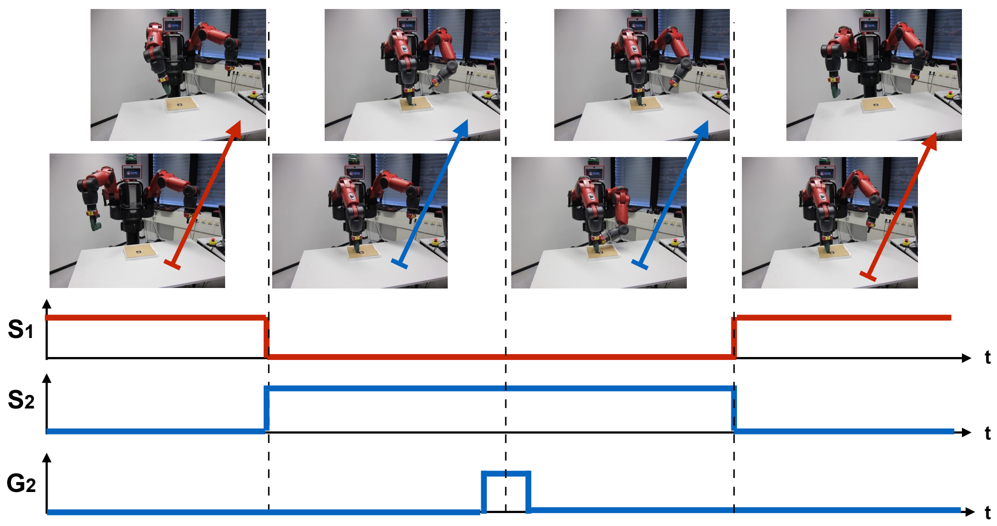


Figure 5.13: Steps of robot using an electric tacker after learning. S_1, S_2 are the activation signals for robot's arm $1, 2$, G_2 is the gripper's open/close state for arm 2.

5.4.3 Experiment 3: Learning to use an electric tacker

We also evaluate our approach for learning to use an electric tacker. During the demonstration, the teacher moves one of the robot's arms holding the tacker to the action point T. Then the other arm is moved to reach the button of the tacker, and press it with the gripper. Afterwards, the arm returns to a neutral position. A successful run requires the robot to (1) position the tacker, (2) correctly press the button, (3) release the button and (4) return to the neutral position. Whenever the robot fails to finish the whole process, a failure case is counted. Note that different from using the glue pen and the drill, the second gripper starts with the 'close' state and changes to 'open' state to press the button in this process for better maneuvering the tacker. One successful run and the corresponding signal sequences are shown in Figure 5.13.

5.4.4 Evaluation

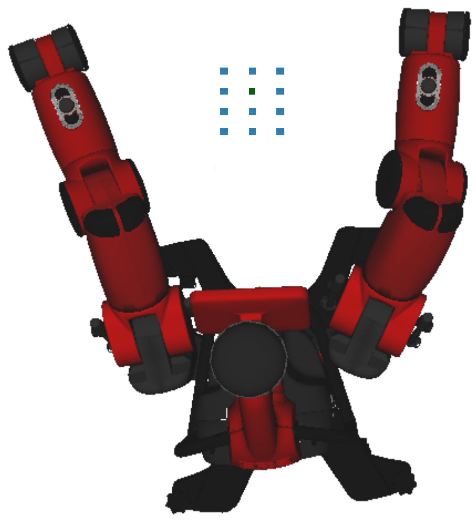


Figure 5.14: Configuration tested in our experiment (top view of the robot), where the green dot denotes the position seen in the demonstration, blue dots for the novel locations.

For all three tools we have tested 11 novel configurations together with the one seen in the demonstration as shown in Figure 5.14. The range of the configurations was chosen based on the reachable space of the robot holding the tool.

As shown in Figure 5.15, alternatively to applying DMPs for each manipulation primitive, one can also use an end-to-end model from start to end position relative to the action point, then query the inverse kinematic solver for a plan. However, this approach tends to neglect the kinematic pattern, some of which reflect the physical constraint of the tool that can be crucial for the end-effector's operation. Two such failure cases of direct planning in contrast to the employment of DMPs are given in Figure 5.16.

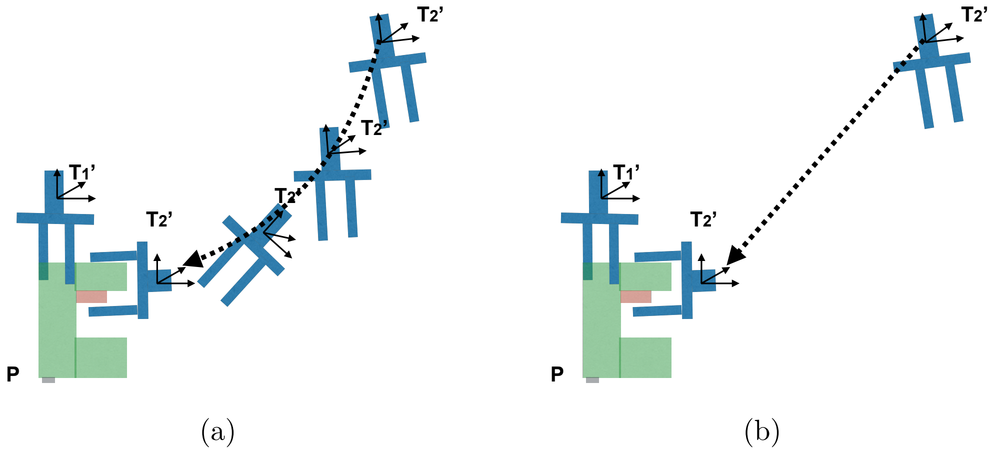


Figure 5.15: Two different approaches for modeling the manipulation primitive: (a): the second gripper fails to reach the proper position for the gluepen. (b): the second gripper fails to reach proper position for the drill.

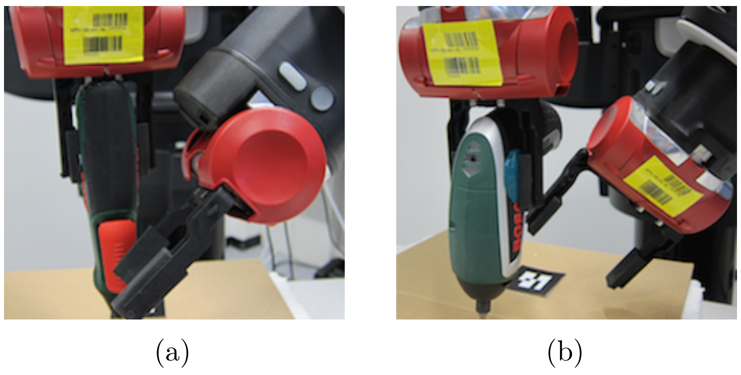


Figure 5.16: Examples of failure cases for end-to-end modeling: (a) the second gripper fails to reach the proper position for the gluepen. (b) the second gripper fails to reach proper position for the drill.

	Sequential MPs	Sequential MPs + DMPs
Glue-pen	66.7%(8/12)	100.0% (12/12)
Drill	33.3%(4/12)	66.7%(8/12)
Tacker	66.7%(8/12)	91.7% (11/12)

Table 5.1: Success rate of tool use in our experiments

A quantitative comparison of results for our sequential *Manipulation Primitives* with and without DMPs for individual primitives is shown in Table 5.1. For simpler tasks, like the glue pen and tacker, sequential MPs with simple end-to-end modeling for each primitive achieves a reasonable success rate. Overall, DMPs significant improve the success rate for using all the three tools.

5.5 Conclusions

In this work, we first present a novel and compact model for using tools that can be described by a tip model. Then we explore a strategy of utilizing a dual-gripper approach for manipulating tools – motivated by the absence of dexterous hands on today's most widely deployed general purpose robots. Afterwards, we describe and formulate our hierarchical architecture to embed tool use in a learning from demonstration framework. At a high-level, we learn temporal orders for dual-arm coordination and at lower-level, we learn DMPs for manipulation primitives. The approach is tested and evaluated on a Baxter research robot. Learning and operation of three human tools, including an electric tacker, an electric drill and a hot-glue pen are shown.

Part III
From Perception over Anticipation to Manipulation

To achieve complex, intelligent behavior of an autonomous system, it is necessary to pursue a more advanced integration of both perception and anticipation into manipulation.

In this part, we focus on a block stacking task. In chapter 6, we first introduce the simulation environments used in the following two chapters and discuss some details on their design and implementation. In chapter 7, we explore an explicit learning of intuitive physics into anticipation to guide stacking single block into the scene without collapsing the existing structure. In chapter 8, we employ an end-to-end learning of stacking multiple blocks to achieve a specified target structure.

Chapter 6

Simulation Environment

Contents

Modern machine learning techniques build on data. With the rise of deep learning models, the need of data becomes even more prominent. One of the key elements for the recent success of deep learning model in domains like generic image and speech recognition is the abundance of data in related fields. However, when it comes to a specific domain, it often runs short of data and in reality, collecting data is still an expensive process. One remedy for this issue is to exploit the domain knowledge to synthesize data and utilize the generated data for learning. This also happens to be true for our cases as there is no obvious source of clean and sufficient data to learn physics from. Hence, we make use of the game engine with physics engine to create a simulation environment. Within the environment, we can efficiently collect data to meet our specific needs.

In this chapter,we start with the introduction of the Panda3D game engine which we used to build the simulation environments for the projects in the next two chapters. Then we go through some detailed design in the environments for chapter 7 on Visual Stability Prediction for Robotic Manipulation and chapter 8 on Acquiring Target Stacking Skills respectively.

6.1 Introduction to Panda3D

Panda3D [8] is an open source game engine for Python and C++ programs. It was originally developed by the Disney's VR studio to be "flexible enough to support everything from realtime graphics applications to the development of high-end virtual reality theme park attractions or video games" (Goslin and Mine, 2004) and it has evolved significantly along the years.

As a game engine, the Panda3D provides additional capabilities besides 3D rendering including the physics system with integration of different physics engines. Asides from its own built-in basic physics engine, it also supports more advanced ones including the Open Dynamics Engine (ODE) [9] (Smith *et al.*, 2005) and the Bullet [10] (Coumans, 2010) for physics simulation back-end. We used the Bullet in both of our simulation environments.

The workflow of Panda3D builds on the concept of scene graph. The **Scene graph** is a general data structure to represent and organize a graphical scene. In Panda3D, the scene graph is maintained as a tree of objects to be rendered. The tree consists of objects of class `PandaNode`. As shown in Figure 6.1, the root node is called the `render` and the rest define different perspectives of the scene with various attributes. For example, the `LensNode` controls the camera such as the perspective, the `LightNode` manages the lighting in the scene such as the color and type of the lighting and the `ModelNode` encodes the 3D object in the frame.

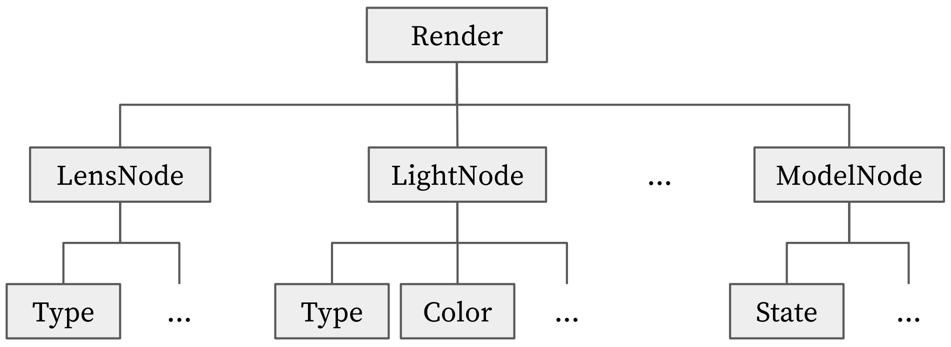


Figure 6.1: An example of scene graph in Panda3D.

Another important concept is the **task**. Tasks are functions called by Panda3D at every frame or for every specified amount of time. Together with **event handler** which is called upon special conditions (**events**) occur, update can be made to the scene in Panda3D between rendering steps as shown in Figure 6.2. For instance, the task can be the simulation subroutine that updates the states of objects in the scene caused by physics.

[8] https://www.panda3d.org/
[9] http://www.ode.org/
[10] http://bulletphysics.org/wordpress/

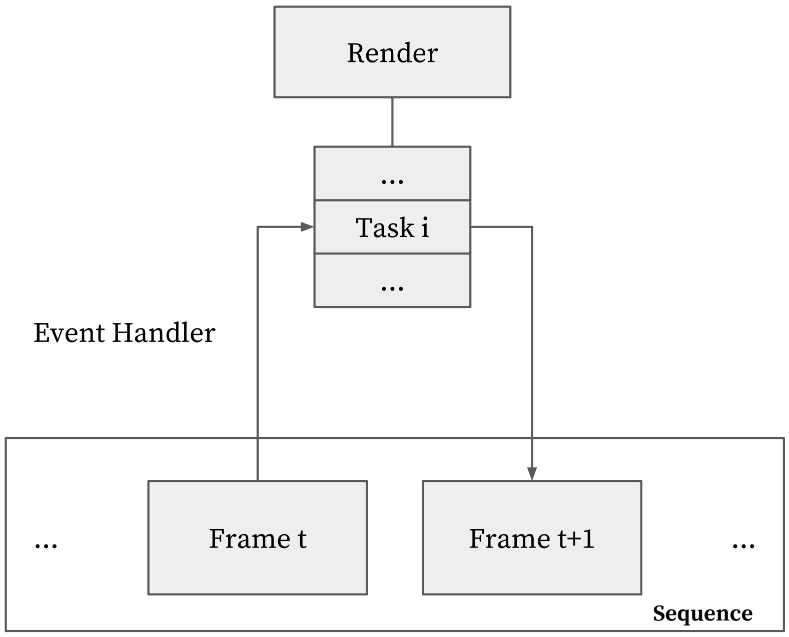


Figure 6.2: The render process in Panda3D.

6.2 Data Generator for Visual Stability Prediction

In chapter 7, in order to learn a model to predict physical stability directly from mono-view image of a block structure, we need a lot of images for the structures and their corresponding stability labels. Collecting data in real world requires tedious efforts of setting up the scene for different structures, capturing their images, observing and recording the stability outcomes. Instead, we refer to the game engine to build a simulation environment to do all of these procedures in an automatic way.

Figure 6.3 gives an overview of the data generator. The first key component is the **tower generation**. It automatically generate a large number of different blocks structures (tower) under the *scene parameters*, including the number of blocks, stacking depth and block size (for more details, please refer the description in chapter 7). These towers are recorded as scene files which only track all the locations of blocks in the scene.

The next component is the **stability simulation**. It loads the stored scene files and simulates their stability with physics engine. The images for the towers before running the physics engine are captured as the scene images, the stability labels from the simulation are automatically determined and recorded for the corresponding towers. Both the scene images and the obtained stability labels are later put into the deep convolutional neural network to learn the visual stability classifier.

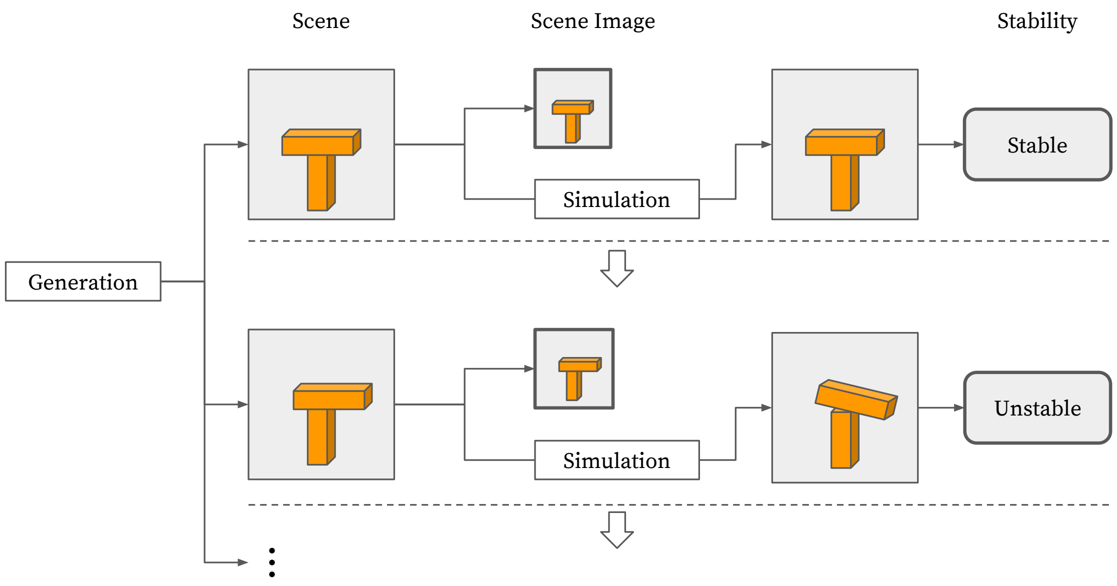


Figure 6.3: Overview of the data generator for visual stability prediction. The scene images and their corresponding stability labels are collected.

6.2.1 Tower Generation

An example of the scene setup is shown Figure 6.4. The tower is placed on a plane, and a camera is positioned at the front-facing location of which the elevation is adjusted for the towers of different heights. For different scenes, the towers are generated differently, and the scene images are captured through the camera. To make the scene images more realistic, wood texture is added to all the blocks.

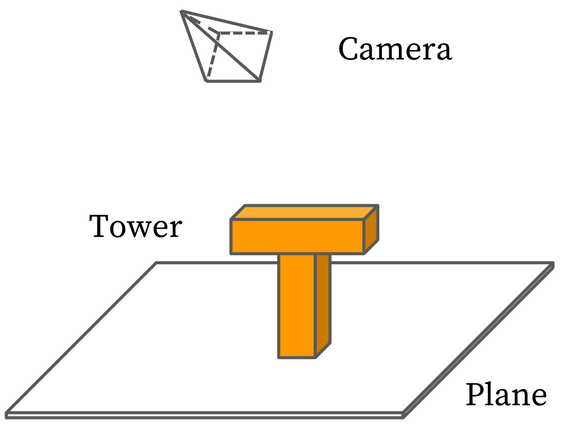
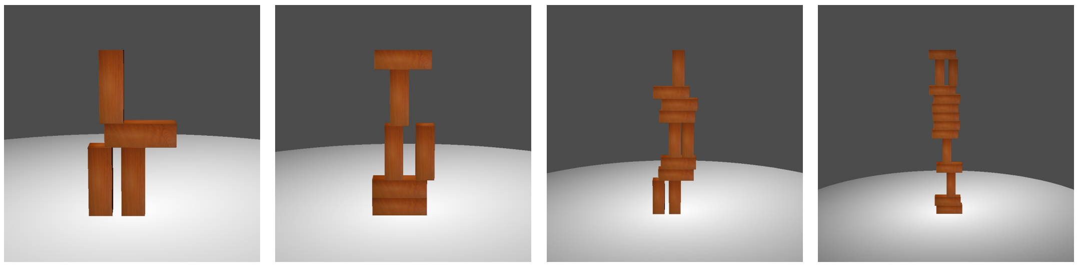


Figure 6.4: Example set-up of the scene for the data generator.

The basic tower generation system is based on the framework by Battaglia *et al.* (2013). Given a specified number of total blocks in the tower, the blocks are sequentially added into the scene under the geometrical constraints, such as no collision between blocks. Some examples of obtained scene images are shown in Figure 6.5.



(a) 4 Blocks (b) 6 Blocks (c) 10 Blocks (d) 14 Blocks

Figure 6.5: Example of generated scene images with different total number of blocks.

6.2.2 Stability Simulation

During the stability simulation, we set the simulation time universally to 2 seconds for all the scenes. We count the displacements of all the blocks in the scene before and after the simulation. If the displacement for any block is above a threshold, it is deemed to be unstable, otherwise stable. This is necessary as the blocks in stable towers can generate small displacement during the simulation process, simply using zero displacement to determine the stability can lead to lots of erroneous cases where the stable towers are labeled as unstable. In practice, we picked the threshold based on the evaluation of a small set of towers.

6.3 Stacking Environment for Skill Learning

An overview of the stacking environment is shown in Figure 6.6. For each episode, a target structure is randomly picked from a pool of structures and then the blocks are spawned according to the target. The agent moves the spawned block in the scene until the placement is done. During this process, if the agent's move causes the block to (1) move out over the boundary of the scene or (2) collide with existing structure (3) collapse the structure with the placement, the episode is terminated and a new episode can start. A new block is spawned upon the previous one is placed successfully till reaching the total number of blocks in the target. When all the blocks are placed in the scene, the environment determines if the agent achieves the assigned target structure.

Compared to the environment for visual stability prediction as a data generator, the stacking environment is a much more engaging component for the learning process with which the agent closely interact with and hence is more complex. The environment implements a more dynamic system where different moves from the agent can lead to different consequences. For example, when a block (not the last one for the episode) is placed stably in the scene, a new block should be spawned into the environment whereas if the block collapses the existing structure, the current episode is terminated so that a new one can be started. The key to react correctly under these different conditions is to detect them effectively. We will describe the mechanism for collision and stability detection in the later subsection. Furthermore, with the rise of interest in deep reinforcement learning, researchers may want to benchmark their own agent's implementation on the standardized environment, so we make additional efforts to formalize the interface design as discussed in the last subsection.

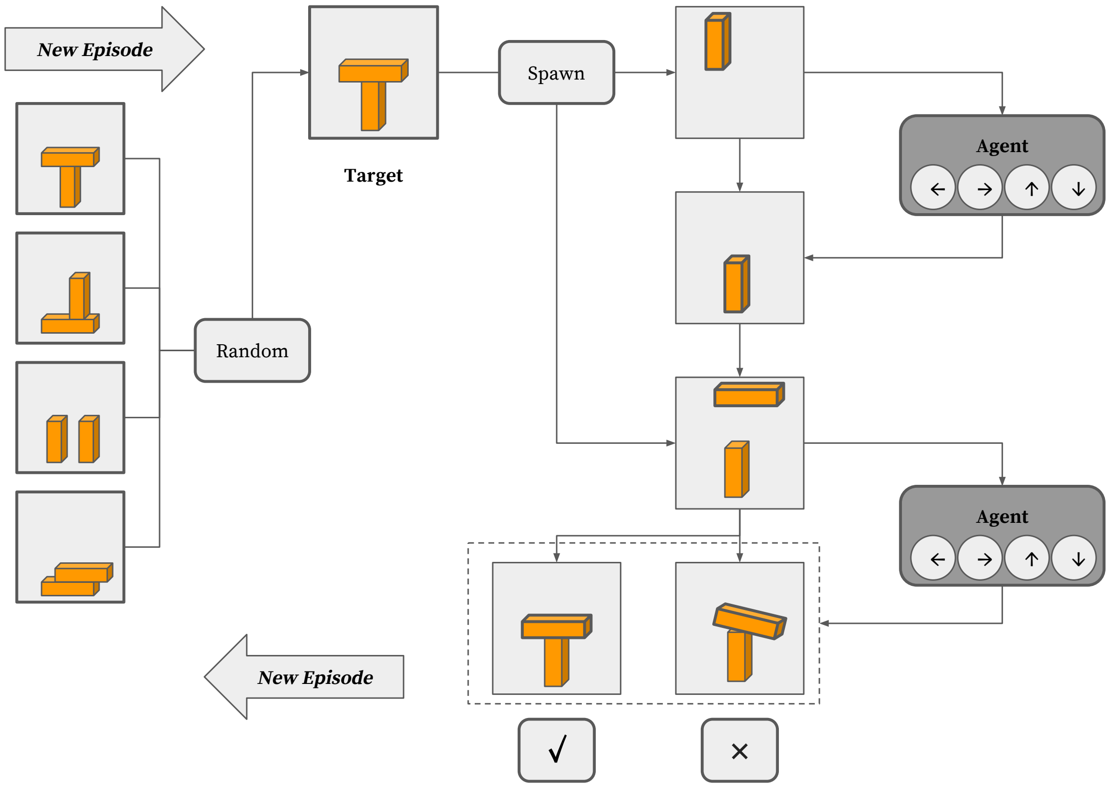


Figure 6.6: Overview of the environment for target stacking.

6.3.1 State Representation

To further reduce the appearance difference caused by varying perspective, the state in the stacking environment is rendered using orthographic projection. It is implemented by the `OrthographicLens` in Panda3D where parallel lines stay parallel and don't converge as shown in Figure 6.7b. This is in contrast with regular perspective camera used in the visual stability prediction environment as shown in Figure 6.7a.

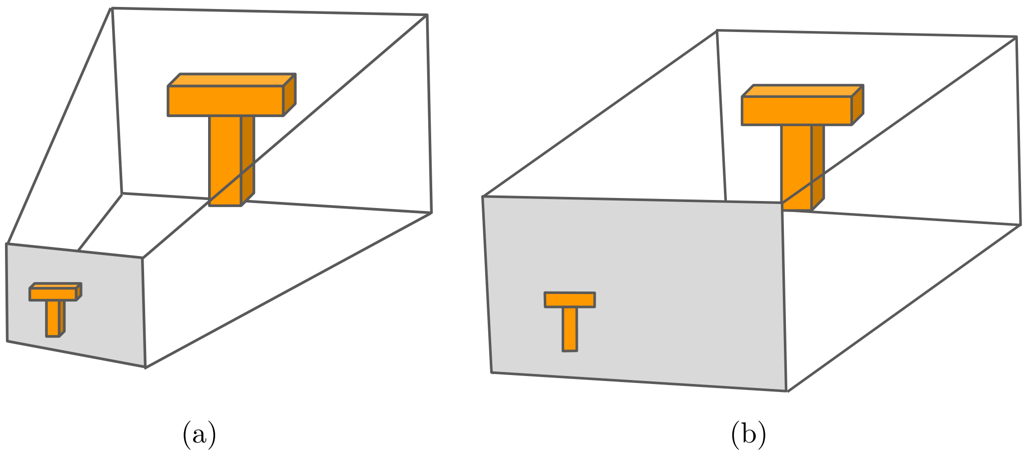


Figure 6.7: Different perspective lens used in this thesis.(a): Perspective projection. (b): Orthographic projection.

Additionally, as we consider the action in discrete space, namely {left,right,down} for the task, the state representation can also be formalized correspondingly as shown in Figure 6.8. The individual block size follows a ratio of $l : w : h = 5 : 2 : 1$, where l,w,h denote length, width and height respectively. With the aforementioned orthographic camera, only length and height are preserved on the projected image with a ratio of $l : h = 5 : 1$. Each action moves the block by a displacement of the block's height along its direction. Hence the state can be represented in a grid map of unit square cells with length of the block's height (u in the image space). In actual implementation, the blocks have to be set with coordinates in continuous values, so a separate binary matrix is used to maintain the discrete representation of the state where 1 denotes the block presence in the cell and 0 otherwise.

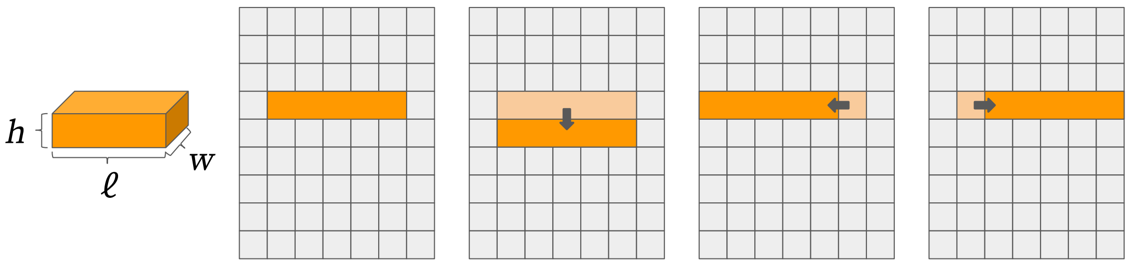


Figure 6.8: State representation in the stacking environment

The environment can query both the rendered image from the camera and the grid map for the state's representation, though in our current experiments in chapter 8, we use mostly the grid map for a simplified representation. Another benefit from the grid map representation is that it can be used as a occupancy map which is crucial for our implemented collision detection mechanism discussed in the next subsection.

6.3.2 Collision and Stability Detection

As the block is maneuvered in the environment, it will eventually make contact with the scene. The occurrence of contact marks an important transition of the state for the environment. As shown in Figure 6.9:

If the contact is made by a block from above (vertical), it suggests a placement[11], then physics simulation is called to obtain the stability. The stability detection is implemented the same way as the one used in the data generator for visual stability prediction. Here the simulation only activates upon vertical contact is detected to save runtime. If the block remains stable in the scene and is not the last block for the episode, a new block is then spawned at the top of the scene with random horizontal location. If the block is the final one and remains stable or it collapses the structure in the scene, then the current episode is terminated and a new episode can be started.

If the contact is made by a horizontal movement, it will not directly affect the current episode. If an additional move causes the collision, then the episode will be terminated, otherwise the block can be moved further until either vertical contact happens, it goes into the simulation and stability-detection branch or collision and terminated.

Note if the block is about to move out of the view of the camera, it is also considered a sub-type of collision. This is implemented straightforwardly by comparing the block's incoming location with the boundary of the scene.

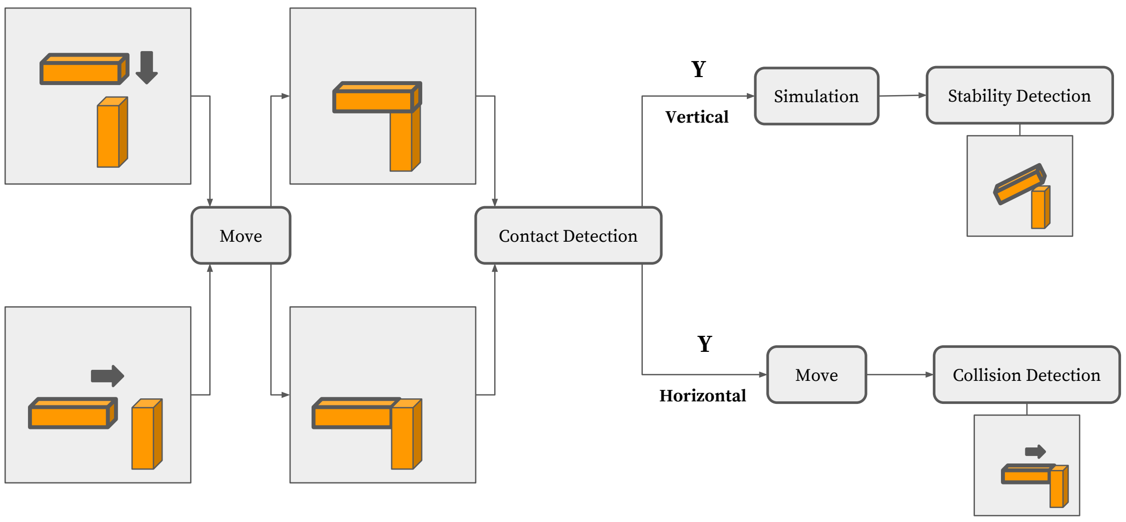
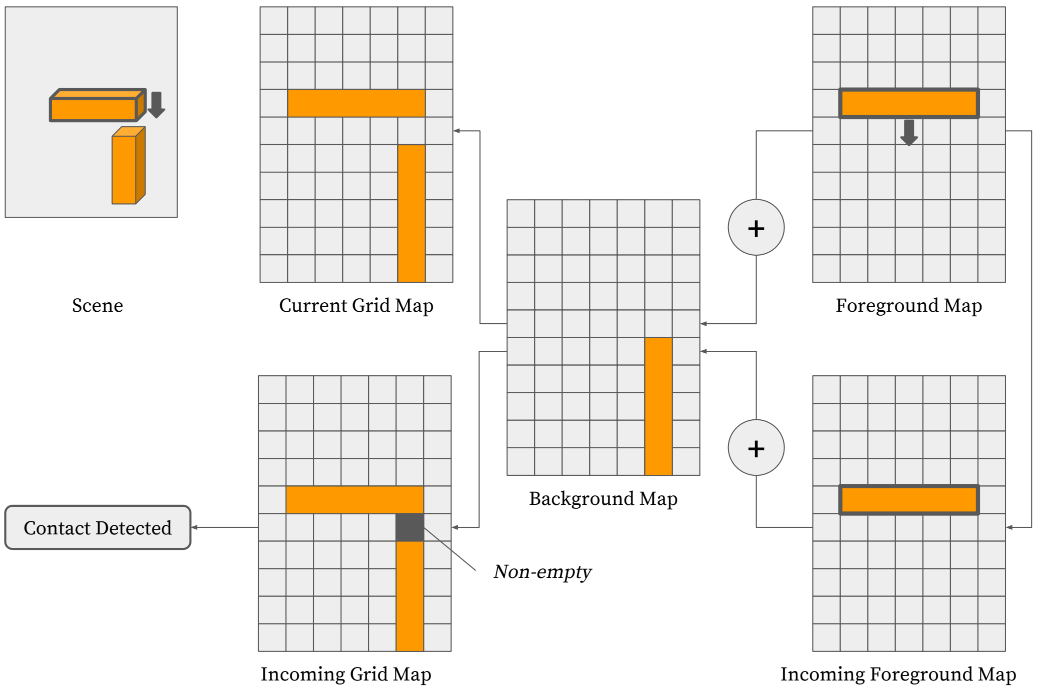


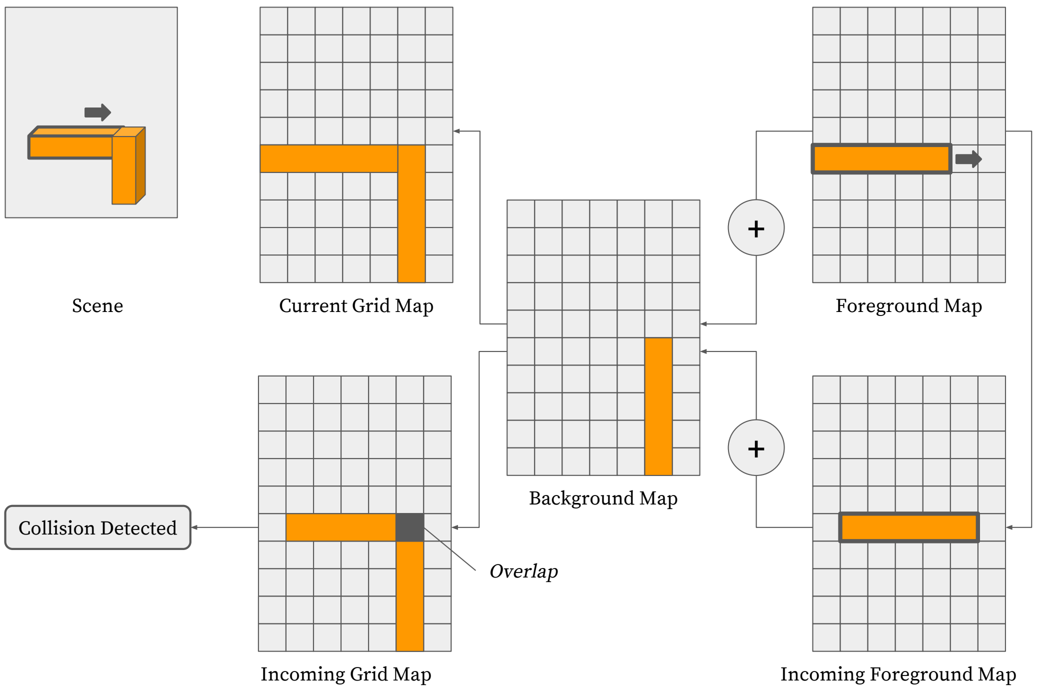
Figure 6.9: Process of collision and stability detection

The contact and collision detection are implemented with the grid map representation described in the previous subsection as shown in Figure 6.10.

[11]In our environment, we simplify the release movement from the real world block placement, i.e. the block is deemed to be released once it makes vertical contact with the scene, either the ground or the structure in the scene.



(a)



(b)

Figure 6.10: Internal representation to detect contact and collision conditions: (a) Detect vertical contact. (b) Detect collision.

We keep two grid map for each episode, one for the existing structure in the scene (**background map**) and the other for the moving block (**foreground map**). A state (grid map) is internally represented as the summation of the background map and foreground map. Given a state and a specified action, the system will compute the incoming foreground map after the action, if the incoming foreground map overlaps with the background map, then collision is detected as shown in Figure 6.10b; if any of the adjacent cells below the block in the incoming foreground map is non-empty in the incoming grid map, then a vertical contact is detected as shown in Figure 6.10a.

6.3.3 Interface Design

With recent progress in reinforcement learning, there is an increasing need for the standardized benchmarks for different learning agents. To this end, OpenAI recently introduced Gym (Brockman *et al.*, 2016) to provide access to a standardized set of environments, such as Atari games and board games. By encapsulating the environment's internal mechanism from the agent, different agents can be evaluated on the environment with minimal modification.

We also adopt a similar design in our environment. The diagram in Figure 6.11 shows part of the design of the environment with interactions with an agent. The agent's `act` method takes the current state from the environment, decides its action and pass the decision to the environment. The `step` method receives the action from the agent and updates the environment's state. The `reward_shape` method evaluates the current state and transforms the reward with specific designs. Then the pair of states before and after the action, together with the reward, are input to the `update` method of the agent to update its parameter depending on different learning algorithms.

As a side note, the environment for the toy navigation task described in chapter 8 is implemented with the same design, so that the same learning agent can be applied to both tasks without further adaptation.

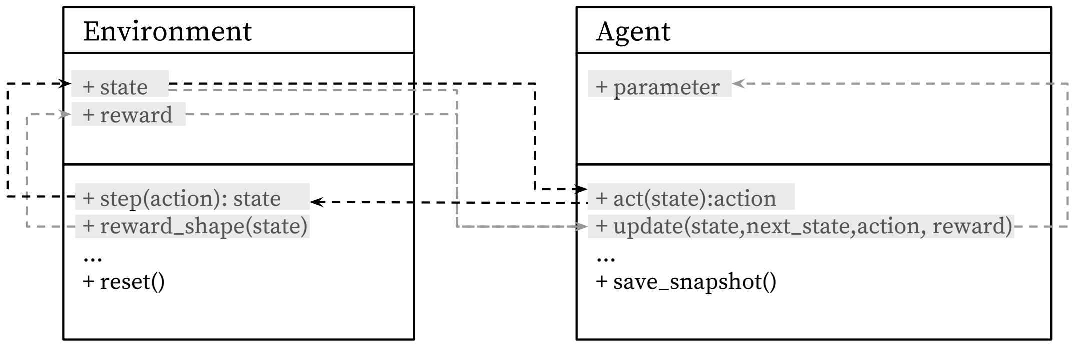


Figure 6.11: Interface design for the environment with interaction of the agent.

Chapter 7

Visual Stability Prediction for Robotic Manipulation

Contents

In this chapter, we start to tackle the manipulation task of block stacking. The goal for the task in this work is to place a single block into the scene without collapsing the existing structure. For this purpose, we introduced a mechanism to predict physical stability directly from visual input. This relates to the concept of *intuitive physics* as we discussed earlier in the relate work in section 2.3. It is a key competence that enables humans and animals to act and interact under uncertain perception in previously unseen environments containing novel objects and their configurations.

We present a learning-based approach based on simulated data that predicts stability of towers comprised of wooden blocks under different conditions and quantities related to the potential fall of the towers. We first evaluate the approach on synthetic data and compared the results to human judgments on the same stimuli. Eventually, we extend this approach to reason about future states of such towers that in return enables successful stacking (Figure 7.1).

This work started as a tech report (Li *et al.*, 2016) focusing on the visual stability prediction on the synthetic data. Later it was extended with the robotic manipulation and accepted as an oral presentation at 2016 NIPS workshop on Intuitive Physics. The final paper is published at ICRA (Li *et al.*, 2017b).

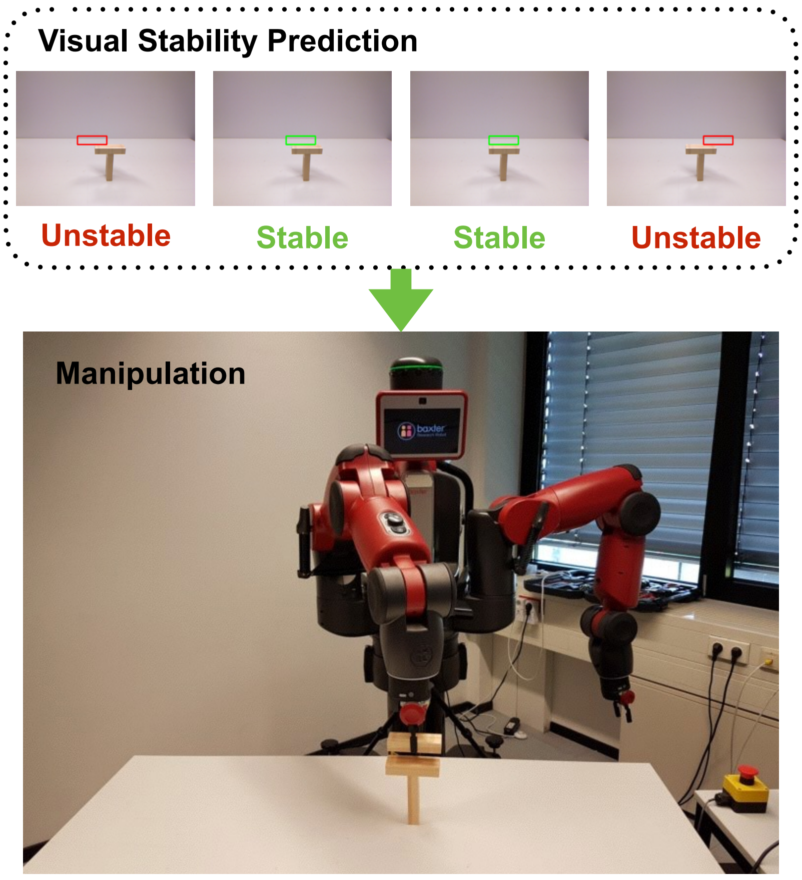


Figure 7.1: Given a wood block structure, our visual stability classifier predicts the stability for future placements, the robot then stacks a block among the predicted stable placements.

7.1 Introduction

Scene understanding requires, among others, understanding of relations between the objects. Many of these relations are governed by the Newtonian laws and thereby rule out unlikely or even implausible configurations for the observer. They are ubiquitous in our everyday visual data which helps us *interpret* the configurations of objects correctly and accurately. Although objects simply obey these elementary laws of Newtonian mechanics, which can very well be captured in simulators, uncertainty in perception makes exploiting these relations challenging in artificial systems.

In contrast, humans understand such physical relations naturally, which e.g., enables them to manipulate and interact with objects in unseen conditions with ease. We build on a rich set of prior experiences that allow us to employ a type of commonsense understanding that, most likely, does not involve symbolic representations of 3D geometry that is processed by a physics simulation engine. We rather build on what has been coined as "naïve physics" (Smith and Casati, 1994) or "intuitive physics" (McCloskey, 1983), which is a good enough proxy to make us operate successfully in the real-world.

It has not yet been shown how to equip machines with a similar set of physics commonsense – and thereby bypassing a model–based representation and a physical simulation. In fact, it has been argued that such an approach is unlikely due to e.g., the complexity of the problem (Battaglia *et al.*, 2013). Only recently, several works have revived this idea and reattempted a fully data driven approach to capturing the essence of physical events via machine learning methods (Mottaghi *et al.*, 2016; Wu *et al.*, 2015; Fragkiadaki *et al.*, 2016; Bhattacharyya *et al.*, 2018).

In contrast, studies in developmental psychology (Baillargeon, 1994) have shown that infants acquire knowledge of physical events by observation at a very early age, for example, support, how an object can stably hold another object; collision, how a moving object interact with another object. According to their research, the infant with some innate basic core knowledge (Baillargeon, 2008) gradually builds its internal model of the physical event by observing its various outcomes. Amazingly, such basic knowledge of physical event, for example the understanding of support phenomenon can make its way into relatively complex operations to construct structures. Such structures are generated by stacking up an element or removing one while retaining the structure's stability primarily relying on effective knowledge of support events in such toy constructions. In our work, we focus on exactly this support event and construct a model for machines to predict object stability.

We revisit the classic setup of Battaglia *et al.* (2013) and explore to which extent machines can predict physical stability events directly from appearance cues. We approach this problem by synthetically generating a large set of wood block towers under a range of conditions, including varying number of blocks, varying block sizes, planar vs. multi-layered configurations. We run those configurations through a simulator (*only at training time!*) in order to generate labels whether the tower would fall or not. We show for the first time that the aforementioned stability test can be learned and predicted in a purely data driven way—bypassing traditional model-based simulation approaches. Further, we accompany our experimental study with human judgments on the same stimuli.

We also apply the approach to guide the robot to stack blocks based on the stability prediction. To circumvent the domain shift between the synthesized images and the real world scene images, we extract the foreground masks for both synthesized and captured images. Given a real world block structure, the robot uses the model trained on the synthesized data to predict the stability outcome across possible candidate placements, and performs stacking on the feasible locations afterwards. We evaluate both the prediction and manipulation performance on the very task.

7.2 Related Work

As humans, we possess the ability to judge from vision alone if an object is physically stable or not and predict the objects' physical behaviors. Yet it is unclear: (1) How we make such decisions and (2) how we acquire this capability. Research in development psychology (Baillargeon, 1994, 1995, 2002) suggests that infants acquire the knowledge of physical events at very young age by observing those events, including support events and others. This partly answers Question 2, however, there seems to be no consensus on the internal mechanisms for interpreting external physical events to address Question 1. Battaglia *et al.* (2013) proposed an intuitive physics simulation engine for such a mechanism and found that it resembles behavior patterns of human subjects on several psychological tasks. Historically, intuitive physics is connected to the cases where people often hold erroneous physical intuitions (McCloskey, 1983), such as they tend to expect an object dropped from a moving subject to fall vertically straight down. It is rather counter-intuitive how the proposed simulation engine in Battaglia *et al.* (2013) can explain such erroneous intuitions.

While it is probably illusive to fully reveal the human's inner mechanisms for physical modeling and inference, it is feasible to build up models based on observation, in particular the visual information. In fact, looking back to the history, physical laws were discovered through the observation of physical events (MacDougal, 2012). Our work is in this direction. By observing a large number of support event instances in simulation, we want to gain deeper insights into the prediction paradigm.

In our work, we use a game engine to render scene images and a built-in physics simulator to simulate the scenes' stability behavior. The data generation procedure is based on the platform used in Battaglia *et al.* (2013), however as discussed before, their work hypothesized a simulation engine as an internal mechanism for human to understand the physics in the external world while we are interested in finding an image-based model to directly predict the physical behavior from visual channel. Learning from synthetic data has a long tradition in computer vision and has recently gained increasing interest (Li and Fritz, 2012; Rematas *et al.*, 2014; Peng *et al.*, 2015; Rematas *et al.*, 2016) due to data hungry deep-learning approaches.

Understanding physical events also plays an important role in scene understanding in computer vision. By including additional clues from physical constraints into the inference mechanism, mostly from the support event, it has further improved results in segmentation of surfaces (Gupta *et al.*, 2010), scenes (Silberman *et al.*, 2012) from image data, and object segmentation in 3D point cloud data (Zheng *et al.*, 2013).

Only very recently, learning physical concepts from data has been attempted. Mottaghi *et al.* (2016) aim at understanding dynamic events governed by laws of Newtonian physics, but use proto-typical motion scenarios as exemplars. In Fragkiadaki *et al.* (2016), they analyze billiard table scenarios, learning the dynamics from observation with explicit object notion. An alternative approach based on boundary extrapolation (Bhattacharyya *et al.*, 2018) addresses similar settings without imposing any object notion. Wu *et al.* (2015) aims to understand physical properties of objects. They again rely on an explicit physical simulation. In contrast, we only use simulation at training time and predict for the first time visual stability directly from visual inputs of scenes containing various towers with a large number of degrees of freedom.

In Lerer *et al.* (2016), the authors present their work similar to our setting. Yet the focus of their work is different from ours, namely predicting outcome and falling trajectories for simple 4 block scenes, whereas we significantly vary the scene parameters, investigating if and how the prediction performance from image trained model changes according to such changes, and further we examine how the human's prediction adapt to the variation in the generated scenes and compare it to our model.

To shed more light on the capabilities of our model, we explore how it can be used in a robotic manipulation task, i.e., stacking a wood block given a block structure. In the past, we have seen researchers perform tasks with wood blocks, like playing Jenga from different perspectives. Kröger *et al.* (2006) demonstrated multi-sensor integration by using a marker-based system with multiple cameras and sensors: a random block is first chosen in the tower, then the robot arm will try to pull the very block, if the force sensor detects large counter force or the CCD cameras detect large motion of tower, then it will stop pulling and try other block. Wang *et al.* (2009) improved on Kröger *et al.* (2006) by further incorporating a physics engine to initialize the candidates for pulling test. A different line of research is Kimura *et al.* (2010) where physical force is explicitly formulated with respect to the tower structure for planning. In our work, we do not do explicit formation of contact force as in Kimura *et al.* (2010), nor do we perform trials on-site for evaluating the robot's operation. We only use physics engine to acquire synthesized data for training the visual-physics model. At test time, the planning system for our robot mainly exploits the knowledge encoded in the visual-physics model to evaluate the feasibility of individual candidates and performs operations accordingly.

7.3 Recognition

In order to tackle a visual stability test, we require a data generation process that allows us to control various degrees of freedom induced by the problem as well as generation of large quantities of data in a repeatable setup. Therefore, we follow the seminal work on this topic (Battaglia *et al.*, 2013) and use a simulator to setup and predict physical outcomes of wood block towers. Afterwards, we describe the method that we investigate for visual stability prediction. We employ state-of-the-art deep learning techniques, which are the de-facto standard in today's recognition systems. Lastly, we describe the setup of the human study that we conduct to complement the machine predictions with a human reference. An overview of our approach is shown in Figure 7.2.

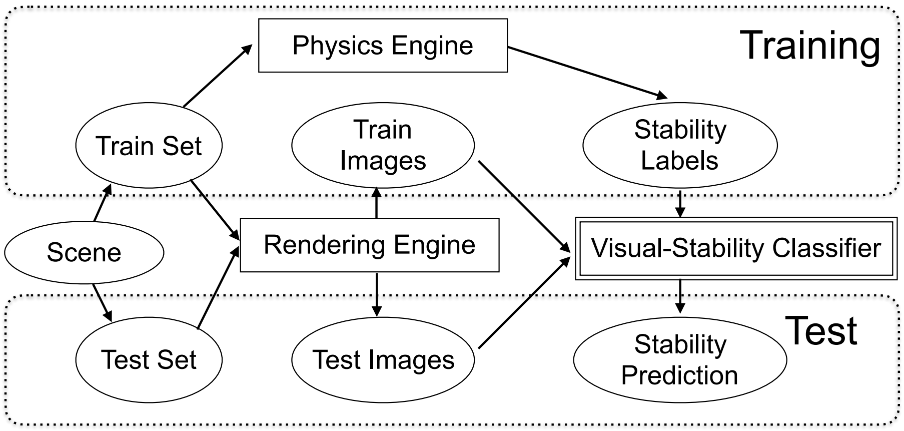


Figure 7.2: An overview of our approach for learning visual stability. Note that physics engine is only used during training time to get the ground truth to train the deep neural network while at test time, only rendered scene images are given to the learned model to predict the physical stability of the scenes.

7.3.1 Synthetic Data

Based on the scene simulation framework used in Hamrick *et al.* (2011) and Battaglia *et al.* (2013), we also generate synthetic data with rectangular cuboid blocks as basic elements. The number of blocks, blocks' size and stacking depth are varied in different scenes, to which we will refer as *scene parameters*.

Numbers of Blocks We expect that varying the size of the towers will influence the difficulty and challenge the competence of "eye-balling" the stability of a tower in humans and machine. While evidently the appearance becomes more complex with the increasing number of blocks, the number of contact surfaces and interactions equally make the problem richer. Therefore, we include scenes with four different number of blocks, i.e., 4 blocks, 6 blocks, 10 blocks and 14 blocks as $\{4B, 6B, 10B, 14B\}$.

Stacking Depth As we focus our investigations on judging stability from a monocular input, we vary the depth of the tower from a one layer setting which we call $2D$ to a multi-layer setting which we call $3D$. The first one only allows a single block along the image plane at all height levels while the other does not enforce such constraint and can expand in the image plane. Visually, the former results in a single-layer stacking similar to Tetris while the latter ends in a multiple-layer structure as shown in Table 7.1. The latter most likely requires the observer to pick up on more subtle visual cues, as many of its layers are heavily occluded.

Block Size We include two groups of block size settings. In the first one, the towers are constructed of blocks that have all the same size of $1 \times 1 \times 3$ as in Battaglia *et al.* (2013). The second one introduces varying block sizes where two of the three dimensions are randomly scaled with respect to a truncated Normal distribution $N(1, \sigma^2)$ around $[1-\delta, 1+\delta]$, σ and δ are small values. These two settings are referred to as $\{\mathrm{U}ni, \mathrm{N}onUni\}$. The setting with non-uniform blocks introduces small visual cues where stability hinges on small gaps between differently sized blocks that are challenging even for human observers.

Scenes Combining these three scene parameters, we define 16 different scene groups. For example, group 10B-2D-Uni is for scenes stacked with 10 Blocks of same size, stacked within a single layer. For each group, 1000 candidate scenes are generated where each scene is constructed with non-overlapping geometrical constraint in a bottom-up manner. There are 16K scenes in total. For prediction experiments, half of the images in each group are for training and the other half for test, the split is fixed across the experiments.

Rendering While we keep the rendering basic, we like to point out that we deliberately decided against colored bricks as in Battaglia *et al.* (2013) in order to challenge perception and make identifying brick outlines and configurations more challenging. The lighting is fixed across scenes and the camera is automatically adjusted so that the whole tower is centered in the captured image. Images are rendered at resolution of 800×800 in color.

Physics Engine We use Bullet (Coumans, 2010) in Panda3D (Goslin and Mine, 2004) to perform physics-based simulation for 2s at 1000Hz for each scene. Surface friction and gravity are enabled in the simulation. The system records the configuration of a scene of N blocks at time t as $(p_1, p_2, ..., p_N)_t$, where p_i is the location for block i. The stability is then automatically decided as a Boolean variable:

$$S = \bigvee_{i=1}^{N} (\Delta((p_i)_{t=T} - (p_i)_{t=0}) > \tau)$$

where T is the end time of simulation, δ measures the displacement for the blocks between the starting point and end time, τ is the displacement threshold, $\bigvee$ denotes the logical Or operator, that is to say it counts as unstable $S = \text{True}$ if any block in the scene moved in simulation, otherwise as stable $S = \text{False}$.

7.3.2 Stability Prediction from Still Images

Inspiration from Human Studies Research in Hamrick *et al.* (2011) and Battaglia *et al.* (2013) suggests the combinations of the most salient features in the scenes are insufficient to capture people's judgments, however, contemporary study reveals human's perception of visual information, in particular some geometric feature, like critical angle (Cholewiak *et al.*, 2013, 2015) plays an important role in the process.

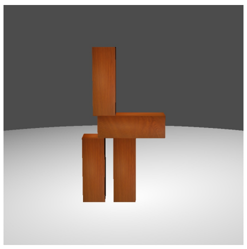
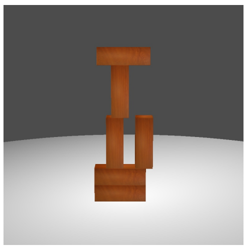
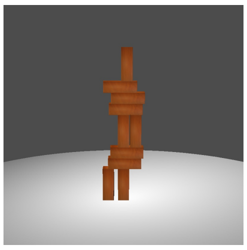
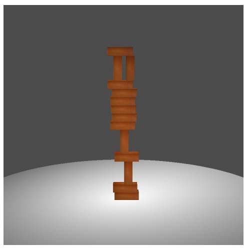
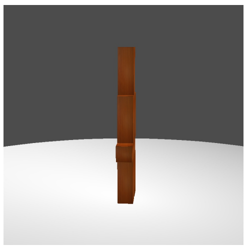
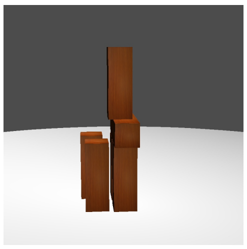
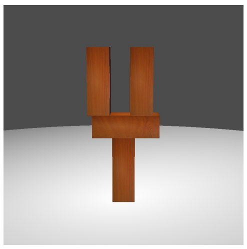
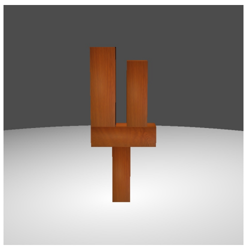
Block Numbers			
			
(a) 4 Blocks	(b) 6 Blocks	(c) 10 Blocks	(d) 14 Blocks
Stacking Depth		Block Size	
			
(e) 2D-stack	(f) 3D-stack	(g) Size-fix	(h) Size-Vary

Table 7.1: Overview of the scene parameters in our rendered scenes. There are 3 groups of scene parameters across number of blocks, stacking depth and block size.

Regardless of the actual inner mechanism for humans to parse the visual input, it is clear there is a mapping f involving visual input I to the stability prediction P.

$$f : I, * \to P$$

Here, $*$ denotes other possible information, i.e., the mapping can be inclusive, as in Hamrick *et al.* (2011) using it along with other aspects, like physical constraint to make judgment or the mapping is exclusive, as in Cholewiak *et al.* (2013) using visual cues alone to decide.

Image Classifier for Stability Prediction In our work, we are interested in the mapping f exclusive to visual input and directly predicts the physical stability. To this end, we use deep convolutional neural networks as it has shown great success on image classification tasks (Krizhevsky *et al.*, 2012). Such networks have been shown to be able to adapt to a wide range of classification and prediction task (Razavian *et al.*, 2014) through re-training or adaptation by fine-tuning. Therefore, these approaches seem to be adequate methods to study visual prediction on this challenging task with the motivation that by changing conventional image classes labels to stability labels the network can learn “physical stability salient” features.

In a pilot study, we tested on a subset of the generated data with LeNet (LeCun *et al.*, 1995), a relatively small network designed for digit recognition, AlexNet (Krizhevsky *et al.*, 2012), a large network and VGG Net (Simonyan and Zisserman, 2015), an even larger network than AlexNet. We trained from scratch for the LeNet and fine-tuned for the large network pre-trained on ImageNet (Deng *et al.*, 2009). VGG Net consistently outperforms the other two, hence we use it across our experiment. We use the Caffe framework (Jia *et al.*, 2014) in all our experiments.

7.3.3 Prediction Performance

In this part of the experiments, the images are captured before the physics engine is enabled, and the stability labels are recorded from the simulation engine as described before. At the training time, the model has access to the images and the stability labels. At test time, the learned model predicts the stability results against the results generated by the simulator.

We divide the experiment design into 3 sets: the intra-group, cross-group and generalization. The first set investigates influence on the model's performance from an individual scene parameter, the other two sets explore generalization properties under different settings.

7.3.3.1 Intra-Group Experiment

In this set of experiments, we train and test on the scenes with the same scene parameters in order to assess the feasibility of our task.

Number of Blocks (4B, 6B, 10B, 14B) In this group of experiment, we fix the stacking depth and keep the all blocks in the same size but vary the number of blocks in the scene to observe how it affects the prediction rates from the image trained model, which approximates the relative recognition difficulty from this scene parameter alone. The results are shown in Table 7.2. A consistent drop of performance can be observed with increasing number of blocks in the scene under various block sizes and stacking depth conditions. More blocks in the scene generally leads to higher scene structure and hence higher difficulty in perception.

Block Size (Uni. vs. NonUni.) In this group of experiment, we aim to explore how same size and varied blocks sizes affect the prediction rates from the image trained model. We compare the results at different number of blocks to the previous group, in the most obvious case, scenes happened to have similar stacking patterns and same number of blocks can result in changes visual appearance. To further eliminate the influence from the stacking depth, we fix all the scenes in this group to be 2D stacking only. As can be seen from Table 7.2, the performance decreases when moving from 2D stacking to 3D. The additional variety introduced by the block size indeed makes the task more challenging.

Num.of Blks	Uni.		NonUni.
	$2D$	$3D$	$2D$
$4B$	93.0	99.2	93.2
$6B$	88.8	91.6	88.0
$10B$	76.4	68.4	69.8
$14B$	71.2	57.0	74.8

Table 7.2: Intra-group experiment by varying scene parameters.

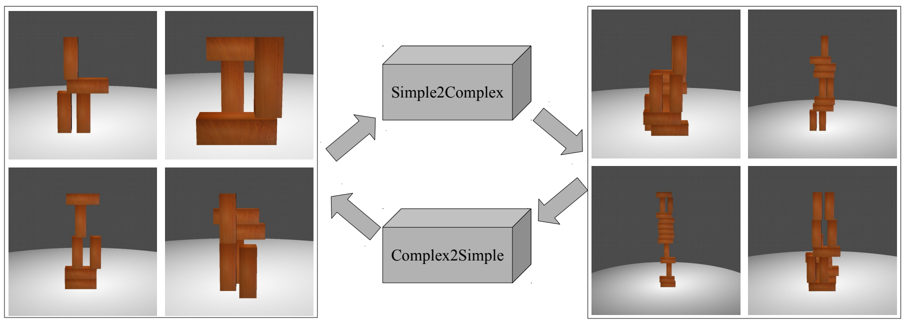
Stacking Depth (2D vs. 3D) In this group of experiment, we investigate how stacking depth affects the prediction rates. With increasing stacking depth, it naturally introduces ambiguity in the perception of the scene structure, namely some parts of the scene can be occluded or partially occluded by other parts. Similar to the experiments in previous groups, we want to minimize the influences from other scene parameters, we fix the block size to be the same and only observe the performance across different number of blocks. The results in Table 7.2 show a little inconsistent behaviors between relative simple scenes (4 blocks and 6 blocks) and difficult scenes (10 blocks and 14 blocks). For simple scenes, prediction accuracy increases when moving from $2D$ stacking to $3D$ while it is the other way around for the complex scene. Naturally relaxing the constraint in stacking depth can introduce additional challenge for perception of depth information, yet given a fixed number of blocks in the scene, the condition change is also more likely to make the scene structure lower which reduces the difficulty in perception. A combination of these two factors decides the final difficulty of the task, for simple scenes, the height factor has stronger influence and hence exhibits better prediction accuracy for $3D$ over $2D$ stacking while for complex scenes, the stacking depth dominates the influence as the significant higher number of blocks can retain a reasonable height of the structure, hence receives decreased performance when moving from $2D$ stacking to $3D$.

7.3.3.2 Cross-Group Experiment

In this set of experiment, we want to see how the learned model transfers across scenes with different complexity, so we further divide the scene groups into two large groups by the number of blocks, where a *simple scene* group for all the scenes with 4 and 6 blocks and a *complex scene* for the rest of scenes with 10 and 14 blocks. We investigate in two-direction classification, shown in the figure in Table 7.3:

1. Train on simple scenes and predict on complex scenes: Train on 4 and 6 blocks and test on 10 and 14 blocks

2. Train on complex scenes and predict on simple scenes: Train on 10 and 14 blocks and test on 4 and 6 blocks



Setting	Simple → Complex	Complex → Simple
Accuracy (%)	69.9	86.9

Table 7.3: The upper figure shows the experiment settings for Cross-group classification where we train on simpler scenes and test on more complex scenes. The lower table shows the results.

As shown in Table 7.3, when trained on simple scenes and predicting on complex scenes, it gets 69.9%, which is significantly better than random guess at 50%. This is understandable as the learned visual feature can transfer across different scene. Further we observe significant performance boost when trained on complex scenes and tested on simple scene. This can be explained by the richer feature learned from the complex scenes with better generalization.

7.3.3.3 Generalization Experiment

In this set of experiment, we want to explore if we can train a general model to predict stability for scenes with any scene parameters, which is very similar to human's prediction in the task. We use training images from all different scene groups and test on any groups. The Result is shown in Table 7.4. While the performance exhibits similar trend to the one in the intra-group with respect to the complexity of the scenes, namely increasing recognition rate for simpler settings and decreasing rate for more complex settings, there is a consistent improvement over the intra-group experiment for individual groups. Together with the result in the cross-group experiment, it suggests a strong generalization capability of the image trained model.

7.3.3.4 Discussion

Overall, we can conclude that direct stability prediction is possible and in fact fairly accurate at recognition rates over 80% for moderate difficulty levels. As expected, the 3D setting adds difficulties to the prediction from appearance due to significant occlusion for towers of more than 10 blocks. Surprisingly, little effect was observed for small tower sizes switching from uniform to non-uniform blocks - although the appearance difference can be quite small. To better understand our results, we

Num.of Blks	Uni.		NonUni.	
	$2D$	$3D$	$2D$	$3D$
4B	93.2	99.0	95.4	99.8
6B	89.0	94.8	87.8	93.0
10B	83.4	76.0	77.2	74.8
14B	82.4	67.2	78.4	66.2

Table 7.4: Results for generalization experiments.

further discuss the following two questions:

How does the model performs compared to human? To answer this, we conduct a human subject test. We recruit human subjects to predict stability for give scene images. Due to large number of test data, we sample images from different scene groups for human subject test. 8 subjects are recruited for the test. Each subject is presented with a set of captured images from the test split. Each set includes 96 images where images cover all 16 scene groups with 6 scene instances per group. For each scene image, subject is required to rate the stability on a scale from $1-5$ without any constraint for response time:

1. Definitely unstable: definitely at least one block will move/fall
2. Probably unstable: probably at least one block will move/fall
3. Cannot tell: the subject is not sure about the stability
4. Probably stable: probably no block will move/fall
5. Definitely stable: definitely no block will move/fall

The predictions are binarized, namely 1) and 2) are treated as unstable prediction, 4) and 5) as stable prediction, "Cannot tell" will be counted as 0.5 correct prediction.

The results are shown in Table 7.5. For simple scenes with few blocks, human can reach close to perfect performance while for complex scenes, the performance drops significantly to around 60%. Compared to human prediction in the same test data, the image-based model outperforms human in most scene groups. While showing similar trends in performance with respect to different scene parameters, the image-based model is less affected by a more difficult scene parameter setting, for example, given the same block size and stacking depth condition, the prediction accuracy decreases more slowly than the counter part in human prediction. We interpret this as image-based model possesses better generalization capability than human in the very task.

Does the model learn something explicitly interpretable? Here we apply the technique from Zhou *et al.* (2016) to visualize the learned discriminative image

Num.of Blks	Uni.		NonUni.	
	$2D$	$3D$	$2D$	$3D$
4B	79.1/**91.7**	93.8/**100.0**	72.9/**93.8**	92.7/**100.0**
6B	78.1/**91.7**	83.3/**93.8**	71.9/**87.5**	89.6/**93.8**
10B	67.7/**87.5**	72.9/72.9	66.7/**72.9**	71.9/68.8
14B	71.9/**79.2**	68.8/66.7	71.9/**81.3**	59.3/**60.4**

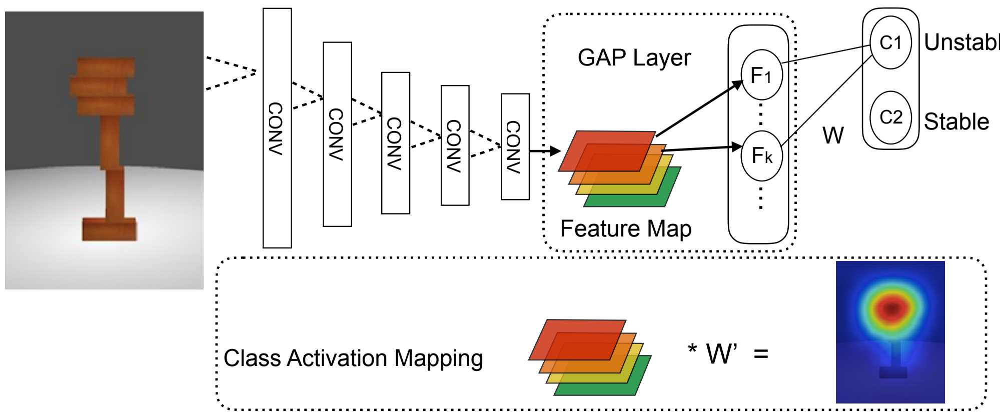
Table 7.5: Results from human subject test a and corresponded accuracies from image-based model b in format a/b for the sampled data.

regions from CNN for individual category. The approach is illustrated in Figure 7.3a. With Global Average Pooling (GAP), the resulted spatial average of the feature maps from previous convolutional layers forms fully-connected layer to directly decides the final output. By back-projecting the weights from the fully-connected layer from each category, we can hence obtain Class Activation Map (CAM) to visualize the discriminative image regions. In our case, we investigate discriminative regions for unstable predictions to see if the model can spot the weakness in the structure. We use deep flow (Weinzaepfel *et al.*, 2013) to compute the optical flow magnitude between the frame before the physics engine is enabled and the one afterwards to serve as a coarse ground truth for the structural weakness where we assume the collapse motion starts from such weakness in the structure. Though not universal among the unstable cases, we do find significant positive cases showing high correlation between the activation regions in CAM for unstable output and the regions where the collapse motion begins. Some examples are shown in Figure 7.3b.

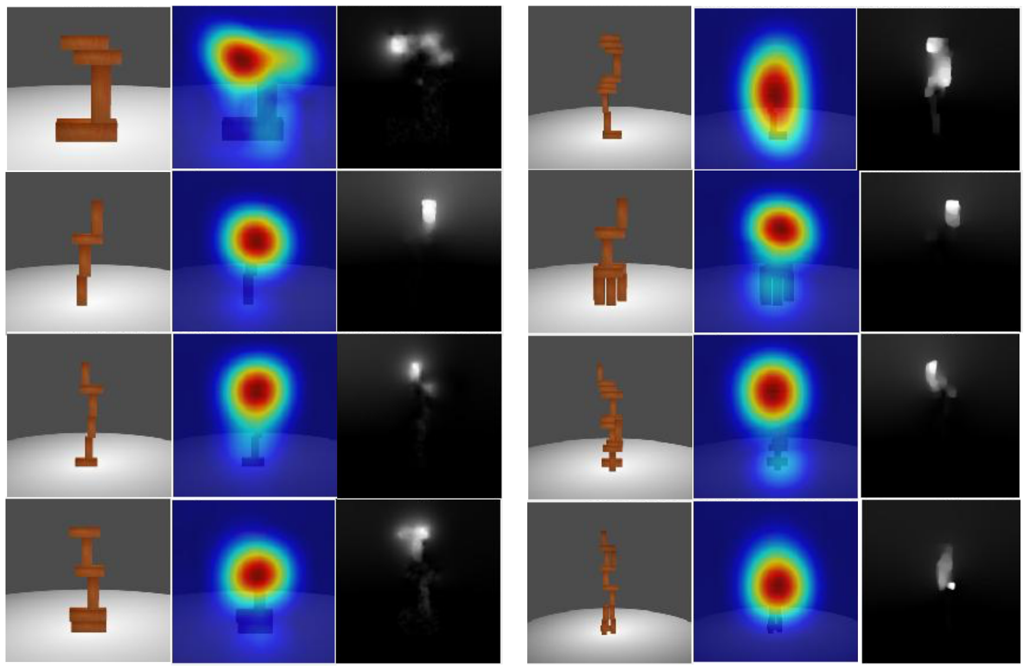
7.4 Manipulation

In the previous section, we have shown that an appereance-based model can predict physical stability relatively well on the synthetic data. Now we want to further explore if and how the synthetic data trained model can be utilized for a real world application, especially for robotic manipulation. Hence, we decide to set up a testbed where a Baxter robot's task is to stack one wood block on a given block structure without breaking the structure's stability as shown in Figure 7.1. The overview of our system is illustrated in Figure 7.4. In our experiment, we use Kapla blocks as basic unit, and tape 6 blocks into a bigger one as shown in Figure 7.5a. To simplify the task, adjustments were made to the free-style stacking:

- The given block structure is restricted to be single layer as the $2D$ case in the previous section. For the final test, we report results on the 6 scenes as shown in Table 7.6.



(a) By introducing the GAP layer directly connected to the final output, the learned weights can be backprojected to the feature map for each category to construct the CAM. The CAM can be used to visualize the discriminative image regions for individual category.



(b) Examples of CAM showing the discriminative regions for unstable prediction in comparison to the flow magnitude indicating where the collapse motion begins. For each example, from left to right are original image, CAM and flow magnitude map.

Figure 7.3: We use CAM to visualize the results for model interpretation.

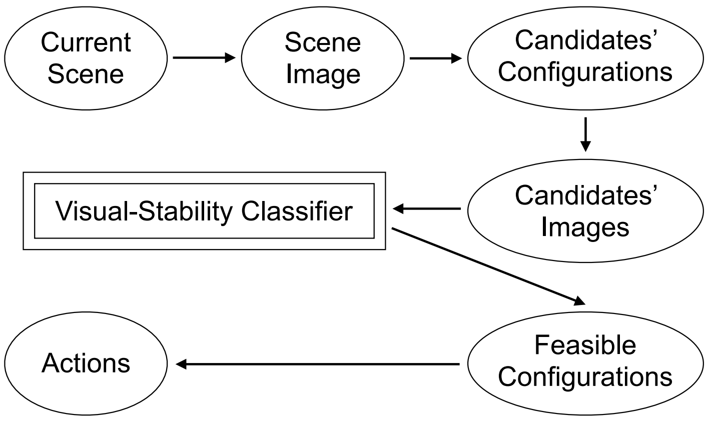
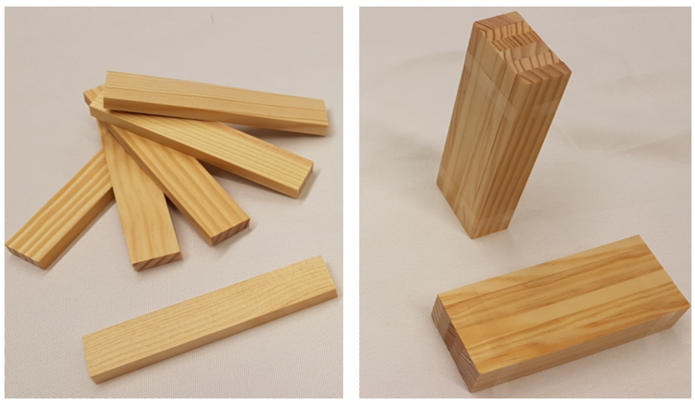
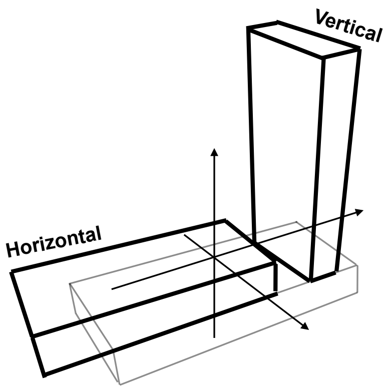


Figure 7.4: An overview of our manipulation system. Our visual-stability classifier is integrated to recognize feasible candidate placement to guide manipulation.



(a) Kapla block (left), block in test (right).



(b) Allowed configurations in test.

Figure 7.5: Blocks used in our experiment.

- The block to be put on top of the given structure is limited two canonical configurations {vertical, horizontal} as shown in Figure 7.5b. and assumed to be held in hand of robot before the placement.

- The block is constrained to be placed on the top-most horizontal surface (stacking surface) in the given structure.

- The depth of the structure (perpendicular distance to the robot) is calibrated so that we only need to decide the horizontal and vertical displacements with respect to the stacking surface.

7.4.1 Prediction on Real World Data

Considering there are significant difference between the synthesized data and real world captured data, including factors (not limited to) as texture, illumination condition, size of blocks and accuracy of the render, we performed a pilot study to directly apply the model trained on the RGB images to predict stability on the real data, but only got results on par with random guessing. Hence we decided to train the visual-stability model on the binary-valued foreground mask on the synthesized data and deal with the masks at test time also for the real scenes. In this way, we significantly reduce the effect from the aforementioned factors. Observing comparable results when using the RGB images, we continue to the approach on real world data.

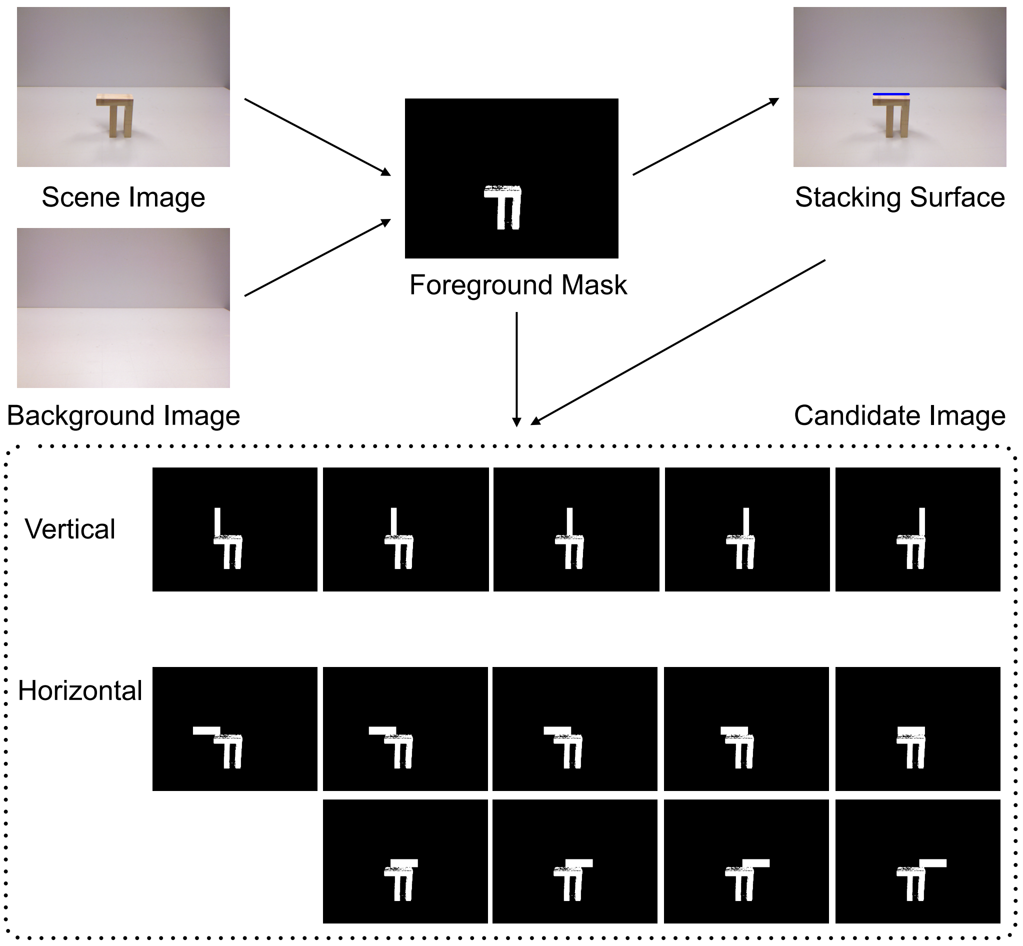


Figure 7.6: The procedure to generate candidates placement images for a give scene in our experiment.

At test time, a background image is first captured for the empty scene. Then for each test scene (shown in Table 7.6), an image is captured and converted to foreground mask via background subtraction. The top-most horizontal boundary is detected as the stacking surface and then used to generate candidate placements: the surface is divided evenly into 9 horizontal candidates and 5 vertical candidates, resulting in 84 candidates. The process is shown in Figure 7.6. Afterwards, these candidates are put to the visual-stability model for stability prediction. Each generated candidate's actual stability is manually tested and recorded as ground truth. The final recognition result is shown in Table 7.6. The model trained with synthetic data is able to predict with overall accuracy of 78.6% across different candidates in real world.

7.4.2 Manipulation Test

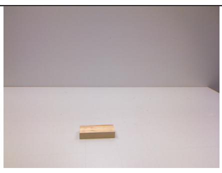
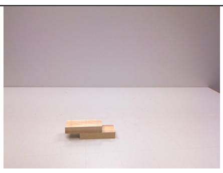
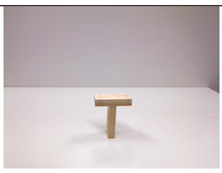
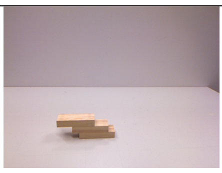
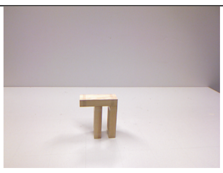
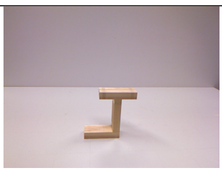
Id.		1		2		3
Scene						
Pred.(%)	66.7	100.0	66.7	60.0	88.9	100.0
Mani.(%)	80.0(4/5)	100.0(5/5)	66.7(2/3)	100.0(3/3)	66.7(2/3)	100.0(1/1)
Placement	H	V	H	V	H	V
Id.		4		5		6
Scene						
Pred.(%)	77.8	80.0	100.0	40.0	66.7	60.0
Mani.(%)	66.7(2/2)	66.7(2/3)	100.0(3/3)	25.0(1/4)	0.0(0/3)	0.0(0/1)
Placement	H	V	H	V	H	V

Table 7.6: Results for real world test. "Pred." is the prediction accuracy. "Mani." is the manipulation success rate with counts for successful placements/all possible stable placements for each scene. "H/V" refer to horizontal/vertical placement.

At test time, when the model predicts a give candidate placement as stable, the robot will execute routine to place the block with 3 attempts. We count the execution as a success if any of the attempt works. The manipulation success rate is defined as:

$$\frac{\#\{\text{successful placements}\}}{\#\{\text{all stable placements}\}}$$

where #{successful placements} is the number of successful placements made by the robot, and #{all stable placements} is the number of all ground truth stable placements.

As shown in Table 7.6, the manipulation performance is good across most of the scenes for both horizontal and vertical placements except for the 6-th scene where the classifier predicts all candidates as unstable hence no attempts have been made by the robot.

7.4.3 Discussion

Comparing to the work in block manipulation (Wang *et al.*, 2009), we do not fit 3D models or run physics simulation at test time for the given scene but instead use the scene image as input to directly predict the physics of the structure. Simply putting the block along the center of mass (COM) of the given structure may often be a feasible option, yet, there are two limitations to this approach: first, it is nontrivial to compute the COM of a given structure; second, it only gives one possible stable solution (assuming it actually stay stable). In comparison, our method does not rely the COM of the structure and provide a search over multiple possible solutions.

7.5 Conclusion

In this work, we answer the question if and how well we can build up a mechanism to predict physical stability directly from visual input. In contrast to existing approaches, we bypass explicit 3D representations and physical simulation and learn a model for visual stability prediction from data. We evaluate our model on a range of conditions including variations in number of blocks, size of blocks and 3D structure of the overall tower. The results reflect the challenges of inference with growing complexity of the structure. To further understand the results, we conduct a human subject study on a subset of our synthetic data and show that our model achieves comparable or even better results than humans in the same setting. Moreover, we investigate the discriminative image regions found by the model and spot correlation between such regions and initial collapse area in the structure. Finally, We apply our approach to a block stacking setting and show that our model can guide a robot for placements of new blocks by predicting the stability of future states.

Chapter 8

Acquiring Target Stacking Skills by Goal-Parameterized Deep Reinforcement Learning

Contents

In chapter 7, we have addressed the manipulation task for the single block's placement without collapsing the existing structure. Here, we want to go further along the task and seek solutions to place multiple blocks. More specifically, we are interested in a setting that resembles how children play wood block, where for every trial, a (different) target is given, the agent should be able to stack back to it.

Rather than explicitly modeling physical knowledge within the policy as in chapter 7, we take a different approach in this work. We investigate how an artificial agent can autonomously acquire this intuition through interaction with the environment. To this end, we created a synthetic block stacking environment with physics simulation in which the agent can learn a policy end-to-end through trial and error.

We propose a deep reinforcement learning framework that learns policies which are parametrized by a goal and validate the model on a toy example navigating in a grid world with different target positions and in a block stacking task with different target structures of the final tower. This work was presented in a tech report (Li *et al.*, 2017a).

8.1 Introduction

Understanding and predicting physical phenomena in daily life is an important component of human intelligence. This ability enables us to effortlessly manipulate objects in unseen conditions. It is an open question how this kind of knowledge can be represented and what kind of models could explain human manipulation behavior (Yildirim *et al.*, 2017). In this paper we are concerned with the question of how an artificial agent can autonomously acquire physical interaction skills through trial and error.

Until recently, researcher have attempted to build computational models for capturing the essence of physical events via machine learning methods from sensory inputs (Mottaghi *et al.*, 2016; Wu *et al.*, 2015; Fragkiadaki *et al.*, 2016; Bhattacharyya *et al.*, 2018; Lerer *et al.*, 2016; Li *et al.*, 2016). Yet there is little work to investigate how the knowledge captured by such a model can be directly applied for manipulation.

In this work, we aim to learn block stacking through trial-and-error, bypassing to explicitly model the corresponding physics knowledge. For this purpose, we build a synthetic environment with physics simulation, where the agent can move and stack blocks and observe the different outcomes of its actions. We apply deep reinforcement learning to directly acquire the block stacking skill in an end-to-end fashion.

While previous work focuses on learning policies for a fixed task, we introduce goal-parameterized policies that facilitate generalization of the learned skill to different targets. We study this problem in the aforementioned block stacking task in which the agent has to reproduce a tower as shown in an image. The agent has to stack blocks into the same shape while retaining physical stability and avoiding pre-mature collisions with the existing structure.

In particular, we aim to learn a single model to guide the agent to build different shapes on request. This is generally not intended in conventional reinforcement learning formulations where the policy is typically optimized to reach a specific goal. In our learning framework, the varying goals are given to the agent as input. We first validated this extended model on a toy example where the agent has to navigate in a gridworld. Both, the location of the start and end point are randomized for each episode. We observed good generalization performance.

Then we apply the framework to the block stacking task. We show that execution depends on the desired target structure and observe promising results for generalization across different goals.

8.2 Related Work

Humans possess the amazing ability to perceive and understand ubiquitous physical phenomena occurring in their daily life. There is research in psychology that seeks to understand how this ability develops. Baillargeon (2002) suggest that infants acquire the knowledge of physical events at a very young age by observing those events, including support events and others. Interestingly, in a recent work (Denil *et al.*, 2017), the authors introduce a basic set of tasks that require the learning agent to estimate physical properties (mass and cohesion combinations) of objects in an interactive simulated environment and find that it can learn to perform the experiments strategically to discover such hidden properties in analogy to human's development of physics knowledge.

Battaglia *et al.* (2013) proposes an intuitive physics simulation engine as an internal mechanism for such type of ability and found close correlation between its behavior patterns and human subjects' on several psychological tasks.

More recently, there is an increasing interest in equipping artificial agents with such an ability by letting them learn physical concepts from visual data. Mottaghi *et al.* (2016) aim at understanding dynamic events governed by laws of Newtonian physics and use proto-typical motion scenarios as exemplars. Fragkiadaki *et al.* (2016) analyze billiard table scenarios and learn dynamics from observation with explicit object notion. An alternative approach based on boundary extrapolation Bhattacharyya *et al.* (2018) addresses similar settings without imposing any object notion. Wu *et al.* (2015) aims to understand physical properties of objects based on explicit physical simulation. Mottaghi *et al.* (2017) proposes to reason about containers and the behavior of the liquids inside them from a single RGB image.

Moreover, Lerer *et al.* (2016) propose using a visual model to predict stability and falling trajectories for simple 4 block scenes. Li *et al.* (2016) investigate if and how the prediction performance of such image-based models changes when trained on block stacking scenes with larger variety. They further examine how the human's prediction adapts to the variation in the generated scenes and compare to the learned visual model. Each work requires significant amounts of simulated, physically-realistic data to train the large-capacity, deep models.

Another interesting question that has been explored in psychology is how knowledge about physical events affects and guides human's actual interaction with objects (Yildirim *et al.*, 2017). Yet it is not clear how machine model trained for physics understanding can be directly applied into real-world interactions with object and accomplish manipulation tasks. Li *et al.* (2017b) makes a first attempt along this direction by extending their previous work (Li *et al.*, 2016) on stability classification. They task a robot to place a wooden block on an existing structure while maintaining stability. Placement candidates are first generated and then evaluated through the visual stability classifier, so that only predicted stable placements are executed on the robot.

In this paper, reinforcement learning is used to learn an end-to-end model directly from the experience collected during interaction with a physically-realistic environment. The majority of work in reinforcement learning focuses on solving task with a single goal. However, there are also tasks where the goal may change for every trial. It is not obvious how to directly apply the model learned towards a specific goal to a different one. An early idea has been proposed by Kaelbling (1993) for a maze navigation problem in which the goal changes. The author introduces an analogous formulation to the Q-learning by using shortest path in replacement of the value functions. Yet there are two major limitations for the framework: 1) it is only formulated in tabular form which is not practical for application with complex states 2) the introduced shortest path is very specific to the maze navigation setting and hence cannot be easily adapt to handle task like different targets stacking, serving as a general solution to this type of problem. In contrast, we propose a goal-parameterized model to integrate goal information into a general learning-based framework that facilitates generalization across different goals. The model has been shown to work on both a navigation task and target stacking.

8.3 Learning

We introduce a new manipulation task: *target stacking*. In this task, an image of a target structure made of stacked blocks is provided. Given the same number of blocks as in the target structure, the goal is to reproduce the structure shown in the image. The manipulation primitives in this task include moving and placing blocks. This is inspired by the scenario where young children learn to stack blocks to different shapes given an example structure. We want to explore how an artificial agent can acquire such a skill through trial and error.

8.3.1 Task Description

For each task instance, a target structure is generated and its image is provided to the agent along with the number of blocks. Each of these blocks has a fixed orientation. The sequence of block orientations is such that reproducing the target is feasible. The agent attempts to construct the target structure by placing the blocks in the given sequence. The spawning location for each block is randomized along the top boundary of the environment. A sample task instance is shown in Figure 8.1.

8.3.2 Task Distinction

The following characteristics distinguish this task from other tasks commonly used in the literature.

Goal-Specific A widely-used benchmark for deep reinforcement learning algorithm are the Atari games (Bellemare *et al.*, 2013) that were made popular by Mnih *et al.* (2013). While this game collection has a large variety, the games are defined by a single goal or no specific goal is enforced at a particular point in time. For example

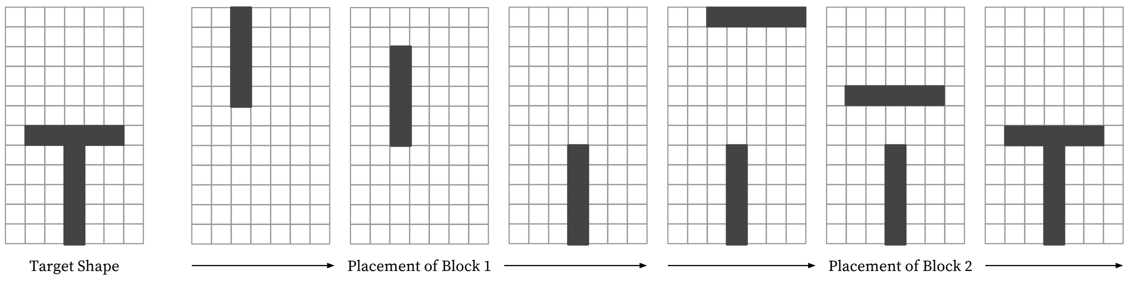


Figure 8.1: Target stacking: Given a target shape image, the agent is required to move and stack blocks to reproduce it.

in Breakout, the player tries to bounce off as many bricks as possible. In Enduro, the player tries to pass as many cars as possible while simultaneously avoiding cars.

In the target stacking task, each task instance differs in the specific goal (the target structure), and all the moves are planned towards this goal. Given the same state, moves that were optimal in one task instance are unlikely to be optimal in another task instance with a different target structure. This is in contrast to games where one type of move will most likely work in similar scenes. This argument also applies to AI research platforms with richer visuals like VizDoom (Kempka *et al.*, 2016).

Longer sequences Target stacking requires looking ahead over a longer time horizon to simultaneously ensure stability and similarity to the target structure. This is different from learning to poke (Agrawal *et al.*, 2016) where the objective is to select a motion primitive that is the optimal next action. It is also different from the work by Li *et al.* (2017b) that reasons about the placement of one block.

Rich Physics Bounded Besides stacking to the assigned target shape the agent needs to learn to move the block without colliding with the environment and existing structure and to choose the block's placement wisely not to collapse the current structure. The agent has no prior knowledge of this. It needs to learn everything from scratch by observing the consequence (collision, collapse) of its actions.

8.3.3 Environment Implementation

A deep reinforcement learning agent requires to learn from a larger number of samples. To enable this, we build a simulated environment for the agent to interact with physical-realistic task instances. While we keep the essential parts of the task, at its current stage the simulated environment remains an abstraction of a real-world robotics scenario. This generally requires an integration of multiple modules for a full-fledged working system, such as Toussaint *et al.* (2010), which is out of scope of this paper.

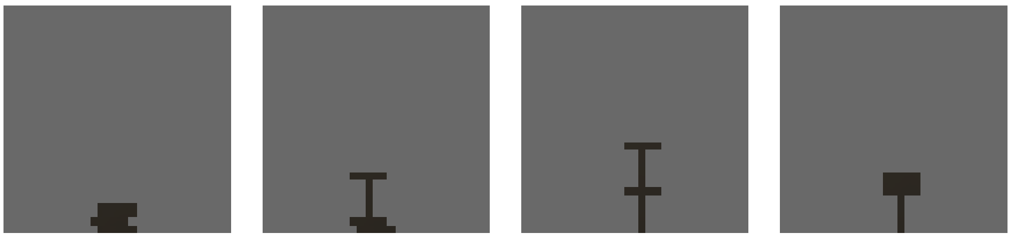


Figure 8.2: Example scenes constructed by the learned agent.

In detail, the simulated stacking environment is implemented in Panda3D (Goslin and Mine, 2004) with bullet (Coumans, 2010) as physics engine. The block size follows a ratio of $l : w : h = 5 : 2 : 1$, where l,w,h denote length, width and height respectively. We ignore the impact during block placement and focus on the resulting stability of the entire structure. Once the block makes contact with the existing structure, it is treated as releasing the block for a placement. In each episode, if the moving block collides with the environment boundary or existing structure, it will terminate the current episode. Further, if the block placement causes the resulting new structure to collapse, it will also end the episode. Stability is simulated similar to Li *et al.* (2017b) by comparing the change of displacement across all the blocks to a pre-selected small threshold within a fraction of time. If all of the blocks' displacements are below this threshold, the structure is deemed stable, otherwise unstable. To simplify the setting, we further constrain the action to be $\{\text{left}, \text{right}, \text{down}\}$.

The physics simulation runs at $60Hz$. However considering the cost of simulation we only use it when there is contact between the moving block and the boundary or the existing structure. Otherwise, the current block is moving without actual physics simulation. To further reduce the appearance difference caused by varying perspective, the environment is rendered using orthographic projection. Figure 8.2 shows example images. The environment provides a user-friendly Python interface (similar to Gym (Brockman *et al.*, 2016)) so that it can be used to test different reinforcement learning agents. At time of publication we will release our implementation of the environment.

8.4 Goal-Parameterized Deep Q Networks

As one major characteristic of this task is that it requires goal-specific planning: given the same or similar states under different objectives, the optimal move can be different. To this end, we extend the typical reinforcement learning formulation to incorporate additional goal information.

8.4.1 Learning Framework

In a typical reinforcement learning setting, the agent interacts with the environment at time t, observes the state s_t, takes action a_t, receives reward r_t and transits to a new state s_{t+1}. A common goal for a reinforcement learning agent is to maximize the cumulative reward. This is commonly formalized in form of a value function as the expected sum of rewards from a state s, $\mathbf{E}[\sum\limits_{i=0}^{\infty}\gamma^i r_{t+i+1}|s_t = s, \pi]$ when actions are taken with respect to a policy $\pi(a|s)$, with $0 \leq \gamma \leq 1$ being the discount factor. The alternative formulation to this is the action-value function $Q^\pi(s,a) = \mathbf{E}[\sum\limits_{i=0}^{\infty}\gamma^i r_{t+i+1}|s_t = s, a_t = a]$.

Value-based reinforcement learning algorithms, such as Q-learning (Watkins and Dayan, 1992) directly search for optimal Q-value function. Recently by incorporating deep neural network as a function approximator for Q-function, the DQN (Mnih *et al.*, 2015) has shown impressive results across a variety of Atari games.

DQN For our task, we apply a *Deep Q Network* (DQN) which uses a deep neural network for approximating the action-value function $Q(s,a;\theta)$, mapping from an input state s and action a to Q values. In particular, two important improvements have been proposed by Mnih *et al.* (2015) for the learning process, including (1) experience replay, the agent stores observed transitions in a memory buffer for some time, and uniformly samples from the memory to update the network (2) the target network, agent maintains two networks for the loss function — one for the current estimator of Q function and one for the surrogate of the true Q function. For the current estimator, the parameters are constantly updated. For the surrogate, the parameters are only updated for every certain number of steps from the current estimator network otherwise kept fixed.

Learning Goal-Parameterized Policies To plan with respect to the specific goal, we can parametrize the agent's policy π by the goal g:

$$\pi(s, g, a) \tag{8.1}$$

Since in this work, we applies DQN as value-based method, this corresponds to the update to original Q function with the additional goal information. The new Q-value function is hence defined as:

$$Q^\pi(s,g,a) = \mathbb{E}[\sum_{i=0}^{\infty}\gamma^i r_{t+i+1}|s_t = s, g, a_t = a] \tag{8.2}$$

As shown in Figure 8.3, in contrast to the original DQN model, where state and action are used to estimate Q-value, the new model further include the current goal into the network to produce the estimate. We call this model as Goal-Parametrized Q Network (GDQN).

The resulted loss function is as:

$$L_Q = \mathbb{E}[(R + \gamma \max_a' Q^\pi(s', g, a'; \theta^-) - Q(s, g, a; \theta))^2] \tag{8.3}$$

where θ^- are the previous parameters and the optimization is with respect to θ.

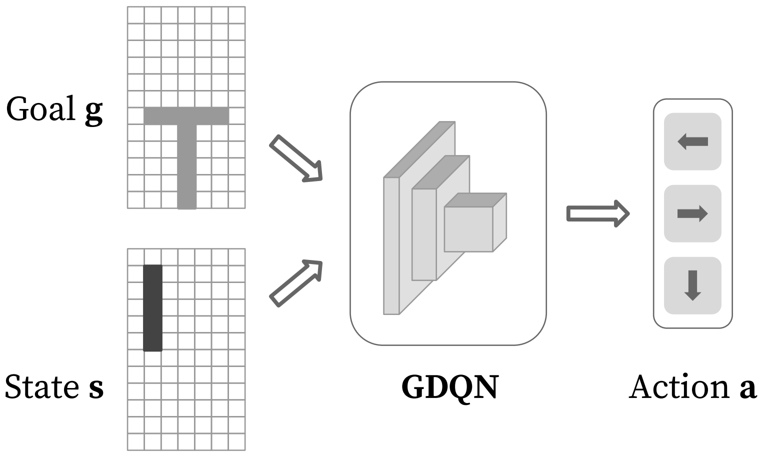


Figure 8.3: Our proposed model GDQN which extends the Q-function approximator to integrate goal information.

8.4.2 Implementation Details

The DQN agent is implemented in Theano and Keras to adapt to the settings in our experiment, while we use a 2 hidden layer (each with 64 hidden units and rectified linear activation) multilayer perceptron (MLP) for most cases, we additionally swap the MLP with the CNN and follow the reported parameter settings as in the original paper (Mnih *et al.*, 2015) to ensure our implementation can reach similar performance.

Note we don't apply the frame-skipping technique (Bellemare *et al.*, 2012) used for Atari games (Mnih *et al.*, 2015) allowing the agent sees and selects actions on every kth frame where its last action is repeated on skipped frames. It does not suit our task, in particular when the moving block is getting close to the existing structure, simply repeating action decided from previous frame can cause unintended collision or collapse.

Reward In the target stacking task, the agent gets reward +1 when the episode ends with complete reproduction of the target structure, otherwise 0 reward.

Further, we explore reward shaping (Ng *et al.*, 1999) in the task providing more prompt intermediate reward. Two types of reward shaping are included: overlap ratio and distance transform.

For the overlap ratio, for each state s_t under the same target g_i, an overlap ratio is counted as the ratio between the intersected foreground region (of the current state and the target state) and the target foreground region (shown in Figure 8.4a):

$$o(s_t, g_i) = \frac{s_t \cap g_i}{g_i} \tag{8.4}$$

For each transition (s_t, a_t, s_{t+1}), the reward is defined by the change of overlap ratio before and after the action:

$$r_t = \begin{cases} 1, & \text{if } \Delta o_{t \to t+1} = o(s_{t+1}) - o(s_t) > 0 \\ -1, & \text{if } \Delta o_{t \to t+1} = o(s_{t+1}) - o(s_t) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.5}$$

The intuition is that actions increasing the current state to become more overlapped with the target scene should be encouraged.

For the distance transform (Fabbri *et al.*, 2008), it generates a map D whose value in each pixel p is the smallest distance from it to a target object O:

$$D(p) = \min\{\text{dist}(p, q) | q \in O\} \tag{8.6}$$

where dist can be any valid distance metric, like Euclidean or Manhattan distance.

For each state s_t under the same target g_i, a distance to the goal is the sum of all the element-wise distance in s_t to g_i under D_{g_i} (shown in Figure 8.4b) as:

$$d(s_t, g_i) = \sum_j D_{g_i}(s_t^j), s_t^j \in s_t \tag{8.7}$$

For each transition (s_t, a_t, s_{t+1}), the reward is defined as:

$$r_t = \begin{cases} 1, & \text{if } \Delta d_{t \to t+1} = d(s_{t+1}) - d(s_t) < 0 \\ -1, & \text{if } \Delta d_{t \to t+1} = d(s_{t+1}) - d(s_t) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.8}$$

The intuition behind this is that action decreasing the distance between the current state and the target scene should be encouraged.

8.5 Experiments

We evaluate the proposed GDQN model on both a navigation task and target stacking and compare it to the base DQN model which does not integrate goal information. In addition, we include the result from GDQN model with different ways of reward shaping in the target stacking task.

8.5.1 Toy Example with Goal Integration

As a toy example, we introduce a type of navigation task in the classic grid-world environment. The locations for the starting point and goal are randomized for each episode. The agent needs to reach the goal with four possible actions

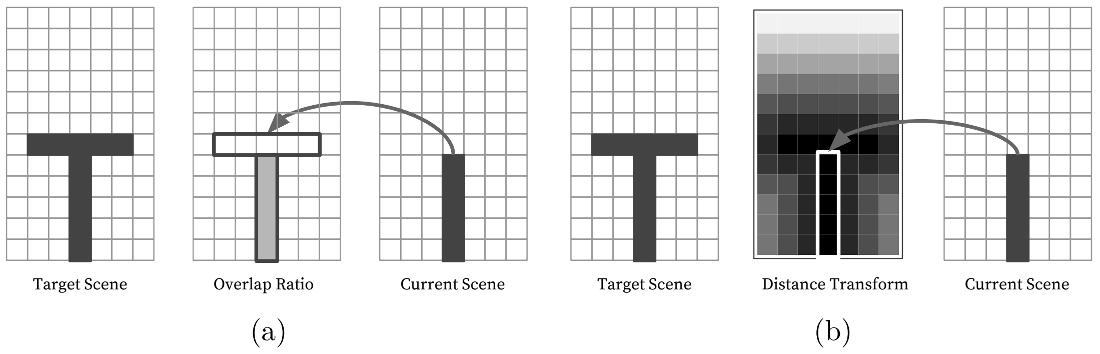


Figure 8.4: Reward shaping used in target stacking. (a): overlap ratio to the target. The gray area in the middle figure denotes the intersected foreground region between current and target scene, and the overlap ratio is the ratio between the areas of the two. (b): distance under the distance transform of the target. The middle figure denotes the distance transform under the target shown in the left. The distance from current scene to the target is the sum of distances masked by the current scene in the distance transform.

Grid Size	DQN	GDQN
5×5	0.67	0.97
7×7	0.67	0.95

Table 8.1: Results from navigation task.

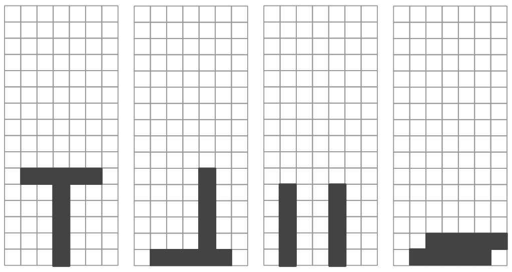
$\{\text{left}, \text{right}, \text{up}, \text{down}\}$. Action that will make the agent go off the grid will leave it stay in the same location. The episode only terminates once the agent reaches the goal. The agent only receive reward +1 when reaching the current goal. Two different sizes of gridworld are tested at 5×5 and 7×7.

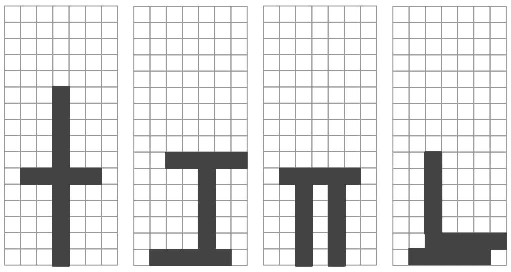
The training epoch size is 1000 in steps for the smaller gridworld and 3000 for the larger one, the test sizes are the same for both at 100. All the agents run for 100 epochs and the ϵ for ϵ-greedy anneals linearly from 1.0 to 0.1 over the first 20 epochs, and fixed at 0.1 thereafter. The memory buffer size is set the same to the annealing length, i.e. for the smaller gridworld, the buffer size equals to the length 20 epochs in training with 20000 steps whereas for the larger one, the buffer size is 30000 steps. We measure the proportion of episodes in the test epoch that reaches the goal in shortest distance as the success ratio. The results are shown in Table 8.1 for the best agents throughout the training process.

As in this simple task with relative small state space, DQN gets some performance due to running an average policy across all the the goals, but this is not addressing the task we set out to do. In contrast, GDQN parametrized specifically to include goal information achieves significant better results on both sizes of the environment.

8.5.2 Target Stacking

We set up 3 groups of target structures consisted of different number of blocks $\{2, 3, 4\}$ in the scene as shown in Figure 8.5. Within each group of target shapes, a random target (with the accompanied orientation order) is picked at the very beginning for individual episode. Each training epoch consists of 10000 steps and each test epoch with 1000 steps. Similar to the setting in the toy example, all the agents run for 100 epochs and the ϵ anneals for the first 20 epochs, and the memory buffer size is set as long as the annealing steps at $200K$ steps.


(a)


(b)

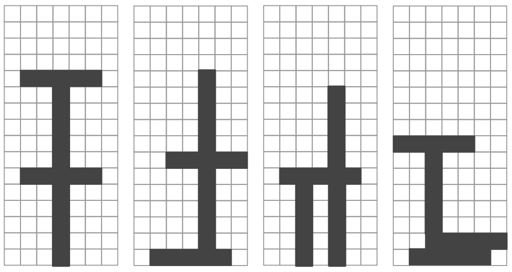

(c)

Figure 8.5: a: Targets for 2 blocks.b: Targets for 3 blocks. c: Targets for 4 blocks.

We computed both average overlap ratio (OR) and success rate (SR) for the finished stacking episodes in each test epoch. Here overlap ratio is the same as defined in the reward shaping in Equation 8.4, but simply measures the end scene over the assigned target scene. This tells the relative completion of the stacked structure in comparison to the assigned target structure, the higher the value is, the better completion it is to the target. At the maximum of 1, it suggests completely reproduction of the target. The success rate counts the ratio how many episodes complete the exact same shape as assigned over the total number of episodes finished in the test epoch. This is the absolute metric counting overall successful stacking. The results are shown in Table 8.2 for the best agents throughout the training process.

Over all groups on both metrics, we observe GDQN outperforms DQN, showing the importance of integrating goal information. In general, the more blocks in the task, the more difficult it becomes. When there are only small number of blocks (2

Num. of Blks.	DQN		GDQN		GDQN + OR		GDQN + DT	
	OR	SR	OR	SR	OR	SR	OR	SR
2	0.70	0.70	0.82	0.72	0.84	0.77	**0.88**	**0.78**
3	0.43	0.43	0.76	**0.67**	**0.86**	0.63	0.83	0.65
4	0.03	0.0	0.41	0.17	0.73	0.55	**0.79**	**0.56**

Table 8.2: Results for target stacking. For "GDQN + X", X denotes different ways for reward shaping as described in previous section, OR for overlap ratio, DT for distance transform. For metrics, OR stands for average overlap ratio, SR for average success rate.

blocks and 3 blocks) in the scene, the single policy learned by DQN averages over the few target shapes can still work to some extent. However when introducing more blocks into the scene, it becomes more and more difficult for this averaged model to handle. As we can see from the result, there is already a significant decrease of performance (success rate drops from 0.70 to 0.43) when increasing the blocks number from 2 to 3, whereas GDQN's performance only decreases slightly from 0.72 to 0.67. In 4 blocks scene, the DQN can no longer reproduce any single target (0.0 for success rate, 0.03 for overlap ratio) while GDQN parametrized specifically to include goal information can still do. Though the success rate (absolute completion to the target) for the basic GDQN is relatively low at 0.17 but the average overlap ratio (relative completion to the target) still holds up pretty well at 0.41. Also we see reward shaping can further improves GDQN model, in particular distance transform can boost the performance more than overlap ratio.

8.6 Conclusion

We create a synthetic block stacking environment with physics simulation in which the agent can learn block stacking end-to-end through trial and error, bypassing to explicitly model the corresponding physics knowledge.

We introduce a target stacking task where the agent stacks blocks to reproduces a tower shown in an image. The task presents a distinct type of challenge requiring the agent to reach a given goal state that may be different for every new trial. Therefore we propose a goal-parametrized GDQN model to plan with respect to the specific goal, allowing better generalization across different goals. We validate the model on both a navigation task in a classic gridworld environment with different start and goal positions and the block stacking task itself with different target structures. Our proposed model shows good performance on both tasks.

Chapter 9

Conclusions and future perspectives

Contents

In this thesis, we have explored different ways of integrations with perception, anticipation and manipulation for robotic systems. In this chapter, we conclude the thesis by summarizing our key contributions and discuss potential future directions.

9.1 Conclusions

9.1.1 From Perception to Anticipation

In chapter 3, we addressed the task on recognition of ongoing activity from videos. In particular we focused on long-duration and complex activities and proposed a new challenging dataset to facilitate the work.

We introduced hierarchical labels over the activity classes and investigated the temporal accuracy-specificity trade-offs. We propose a new method based on recurrent neural networks that learns to predicts over this hierarchy and realizes accuracy specificity tradeoffs. Our method outperforms several baselines on this new challenge including an adaptation of hierarchical prediction from object recognition.

9.1.2 From Perception to Manipulation

In chapter 5, we explored efficient frameworks for programming robot to use tools. We first present a novel and compact model for using tools that can be described by a tip model. Then we explore a strategy of utilizing a dual-gripper approach for manipulating tools – motivated by the absence of dexterous hands on today's most widely deployed general purpose robots. Afterwards, we describe and formulate our hierarchical architecture to embed tool use in a learning from demonstration framework: at a high-level, we learn temporal orders for dual-arm coordination and at lower-level, we learn DMPs for manipulation primitives. The approach is evaluated on a Baxter research robot. We showed its promising application on learning and operation of three human tools, including an electric tacker, an electric drill and a hot-glue pen.

9.1.3 From Perception over Anticipation to Manipulation

In chapter 7 and chapter 8, we tackled the manipulation task of block stacking. In chapter 7, we explored how to guide robot to place a single block into the scene without collapsing the existing structure. For this purpose, we introduced a mechanism to predict physical stability directly from visual input and evaluated it on a synthetic data. To better understand the model, we conducted accompanied human subject test and attempted a model interpretation. Finally, we validated it on a real-world block stacking task.

In chapter 8, we introduced the target stacking task where the agent stacks blocks to reproduces a tower shown in an image. The task presents a distinct type of challenge requiring the agent to reach a given goal state that may be different for every new trial. To do so, we created a synthetic block stacking environment with physics simulation in which the agent can learn block stacking end-to-end through trial and error, bypassing to explicitly model the corresponding physics knowledge.

We proposed a goal-parametrized GDQN model to plan with respect to the specific goal, allowing better generalization across different goals. We validated the model on both a navigation task in a classic gridworld environment with different start and goal positions and the block stacking task itself with different target structures.

Overall, bringing together perception, anticipation and manipulation has shown to deal with complex tasks and that will facilitate future research on acting in complex and dynamic environments.

9.2 Future Perspectives

Now that the discussion of my work is finished, I will give a perspective of future research along the proposed contributions. More specifically:

9.2.1 Higher Level Intelligence

This applies to both: (1) the growing model capability, from shallowly perceiving the data to inferring the implication based on the data and (2) the increasing model interpretability, from merely searching for a working model to designing towards a more tractable model.

Deeper understanding from visual data. More research can be done in the direction of anticipation from the data. Starting from the very topic of anticipation for complex activity from videos in chapter 3, one can go into **fine-grained modeling** over the sub-activities inside individual complex activity and anticipate the incoming sub-activity. Now that our work has shown that one can infer the activity class for the whole sequence with partial observation, this can be further used to reduce the search space for the anticipation. In addition, one can explore the combination of the fine-grained modeling and **objectness representation** and exploit anticipation with richer and more semantic-meaningful description. For example, in the cooking video, the model can predict what tools is the person going to use in the following sub-activity or what ingredient the person needs in the next procedure. This in return will be directly useful in a service robot for daily assistance.

Another interesting extension is to perform anticipation over **group** or **inter-person activity**, such as sports activity or other social activities. Focusing on the interaction between the individuals in the observation, it is crucial to adopt more dedicated representation to capture the characteristics of the scenarios. This can be the quantitative anticipation discussed in section 2.1.2 built on trajectory analysis, for example, to predict the direction for a specific player will move in a soccer game, or can be the qualitative anticipation built on holistic representation over the incurred interaction.

A more relevant idea related to our intuitive physics modeling discussed in chapter 7 is to integrate **common sense** or **prior knowledge** into the anticipation over the visual data. For example, the prior knowledge of stability can hint the possible placement locations for objects in a manipulation task, the awareness of behavioral social cues can suggest potential directions for the following interactions.

Better understanding of the model. A major reservation holds the deep learning model back from real world applications is the lack of interpretability. In our work in chapter 7, we made an initial attempt to seek insight into the deep model we learned for the visual stability prediction. Yet, for the target stacking task in chapter 8, we haven't been able to pull off explicit connections between the placement decision and the physics. Exploration of model interpretation for deep learning model will be an important topic among the community in the perceivable future.

The straightforward approach is to trace the intermediate computation results in the deep-nets, but an alternative can be to directly couple the network with prior knowledge or intuition so that we can inspect the obtained model more easily.

9.2.2 Higher Level Integration

In this dissertation, we have explored the integration of perception, anticipation and manipulation in intelligent system. Integration of information from different channels and conforming to more sophisticated decision mechanism will be a continuous pursuit in both academic and industrial world.

Towards Real World Application. A most relevant application from the thesis is the mobile manipulator, which is still at its primitive stage. Most production robots aimed at household service still fall into the category of robot arm for simple pick-and-place with basic object detection modules. To take up real world tasks with higher autonomy in the system, it is essential to be capable of processing information in a more complex and dynamic environment.

In chapter 7 and chapter 8, we have demonstrated how a system plans its manipulation with physics understanding. Yet to put this mechanism to real world application, further groundings of the task need to be considered. A very simple example task would be placing the tablewares, though the system can always find feasible placement for the plate on top of the bowl but it is preferable for the other way around. In such scenario, task groundings, together with physics should be accounted into the planning loop.

Moreover, it is appealing to further include the context information. In applications of human-robot interactions, not only should the robot understand the task and plan its actions by the task's constraints but also be able to anticipate the behavior or intention of humans as we discussed in chapter 3 to provide precise and timely assistance.

Synergy of data and learning. Even in an age often characterized by "big data", obtaining sufficient quality annotation is always expensive. With simulation and data synthesis, one can more easily and relatively cheaply generate data for specific needs. This has been particular successful in games (Mnih *et al.*, 2015; Silver *et al.*, 2016). In our work in both chapter 7 and chapter 8, we used synthetic data for our studies.

While simulation engines and game engines, such as the Gazebo[12], the Unreal Engine[13] and the Panda3D used in our work provide a generic platform to write environment to different needs. It is still very time-consuming to customize an environment to a specific tasks and limits its use from a broader application. Hence, an interesting direction would be to develop a user-friendly generic platform with easy customization.

Further, in domains where the real world data are significantly more complex and different than the simulation can capture, such as the manipulation task in the real world, it is often nontrivial to adopt the model that works on the synthetic data for the real world counter-part. Hence, how to bridge the gap between the synthetic

[12]http://gazebosim.org/
[13]https://www.unrealengine.com/

data and real world data will be a constant topic in the field. In chapter 7, we chose the representation to reduce the discrepancy between the real world and synthetic data. An alternative is to train an initial model on the synthetic data and fine-tune it on the real world data. Various possibilities can be explored here.

List of Figures

List of Tables

Bibliography

P. Abbeel (2008). *Apprenticeship learning and reinforcement learning with application to robotic control*, Stanford University. Cited on page 19.

J. Aggarwal and M. S. Ryoo (2011). Human activity analysis: A review, *ACM Computing Surveys (CSUR)*. Cited on page 39.

P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine (2016). Learning to poke by poking: Experiential learning of intuitive physics, in *Advances in Neural Information Processing Systems 2016*. Cited on page 119.

E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter (2011). Learning the semantics of object–action relations by observation, *The International Journal of Robotics Research*, p. 0278364911410459. Cited on page 70.

E. E. Aksoy, F. Wörgötter, A. Ude, *et al.* (2015). Probabilistic semantic models for manipulation action representation and extraction, *Robotics and Autonomous Systems*, vol. 65, pp. 40–56. Cited on page 70.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning (2009). A survey of robot learning from demonstration, *Robotics and autonomous systems*, vol. 57(5), pp. 469–483. Cited on pages 15 and 69.

M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt (2011). Sequential deep learning for human action recognition, in *Human Behavior Understanding 2011*, Springer. Cited on page 41.

R. Baillargeon (1994). How do infants learn about the physical world?, *Current Directions in Psychological Science*. Cited on pages 99 and 100.

R. Baillargeon (1995). A model of physical reasoning in infancy, *Advances in infancy research*. Cited on page 100.

R. Baillargeon (2002). The acquisition of physical knowledge in infancy: A summary in eight lessons, *Blackwell handbook of childhood cognitive development*. Cited on pages 22, 23, 100, and 117.

R. Baillargeon (2008). Innate ideas revisited: For a principle of persistence in infants' physical reasoning, *Perspectives on Psychological Science.* Cited on page 99.

P. Battaglia, T. Ullman, J. Tenenbaum, A. Sanborn, K. Forbus, T. Gerstenberg, and D. Lagnado (2012). Computational Models of Intuitive Physics, in *Proceedings of the Cognitive Science Society 2012*. Cited on page 21.

P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum (2013). Simulation as an engine of physical scene understanding, *Proceedings of the National Academy of Sciences.* Cited on pages 21, 23, 90, 99, 100, 101, 102, 103, 117, and 133.

B. B. Beck (1980). *Animal tool behavior: the use and manufacture of tools by animals*, Garland STMP Press. Cited on pages 69 and 71.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents, *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279. Cited on page 118.

M. G. Bellemare, J. Veness, and M. Bowling (2012). Investigating Contingency Awareness Using Atari 2600 Games., in *AAAI 2012*. Cited on pages 32 and 122.

A. Bhattacharyya, M. Malinowski, B. Schiele, and M. Fritz (2018). Long-Term Image Boundary Prediction, in *Association for the Advancement of Artificial Intelligence (AAAI) 2018*. Cited on pages 12, 99, 100, 116, and 117.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal (2008). Robot programming by demonstration, in *Springer handbook of robotics 2008*, pp. 1371–1394, Springer. Cited on page 69.

A. Billard and D. Grollman (2013). Robot learning by demonstration, *Scholarpedia*, vol. 8(12), p. 3824. Cited on page 15.

A. F. Blackwell (1988). Spatial reasoning for robots: a qualitative approach. Cited on pages 20 and 21.

M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri (2005). Actions as Space-Time Shapes, in *ICCV 2005*. Cited on pages 11 and 39.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba (2016). OpenAI gym, *arXiv preprint arXiv:1606.01540*. Cited on pages 33, 96, and 120.

L. Cao, Y. Mu, A. Natsev, S.-F. Chang, G. Hua, and J. R. Smith (2012). Scene aligned pooling for complex video recognition, in *ECCV 2012 2012*. Cited on pages 40 and 42.

Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. M. Siskind, and S. Wang (2013). Recognizing Human Activities from Partially Observed Videos, in *CVPR 2013*. Cited on pages 40 and 41.

S. Chiappa and J. R. Peters (2010). Movement extraction by detecting dynamics switches and repetitions, in *Advances in neural information processing systems 2010*. Cited on page 70.

S. A. Cholewiak, R. W. Fleming, and M. Singh (2013). Visual perception of the physical stability of asymmetric three-dimensional objects, *Journal of vision*. Cited on pages 103 and 104.

S. A. Cholewiak, R. W. Fleming, and M. Singh (2015). Perception of physical stability and center of mass of 3-D objects, *Journal of vision*. Cited on page 103.

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*. Cited on page 59.

E. Coumans (2010). Bullet physics engine, *Open Source Software: http://bulletphysics.org*, vol. 1. Cited on pages 88, 103, and 120.

N. Dalal and B. Triggs (2005). Histograms of oriented gradients for human detection, in *CVPR 2005*. Cited on pages 13 and 42.

N. Dalal, B. Triggs, and C. Schmid (2006). Human detection using oriented histograms of flow and appearance, in *ECCV 2006*. Cited on pages 13 and 43.

F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran (2008). Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Cited on page 39.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). Imagenet: A large-scale hierarchical image database, in *CVPR 2009*. Cited on page 105.

J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei (2012). Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition, in *CVPR 2012*. Cited on page 45.

M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. Battaglia, and N. de Freitas (2017). Learning to perform physics experiments via deep reinforcement learning, in *International Conference on Learning Representations 2017*. Cited on page 117.

A. Edsinger and C. C. Kemp (2007). Toward robot learning of tool manipulation from human demonstration, Technical report, Citeseer. Cited on page 69.

R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno (2008). 2D Euclidean distance transform algorithms: A comparative survey, *ACM Computing Surveys (CSUR)*, vol. 40(1), p. 2. Cited on page 123.

B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles (2015). ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding, in *CVPR 2015*. Cited on page 40.

A. Fathi, A. Farhadi, and J. M. Rehg (2011). Understanding egocentric activities, in *ICCV 2011*. Cited on pages 11 and 39.

K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik (2016). Learning Visual Predictive Models of Physics for Playing Billiards, in *ICLR 2016*. Cited on pages 12, 99, 100, 116, and 117.

M. Goslin and M. R. Mine (2004). The Panda3D graphics engine, *Computer*, vol. 37(10), pp. 112–114. Cited on pages 88, 103, and 120.

A. Graves (2012). *Supervised sequence labelling with recurrent neural networks*, Springer. Cited on page 44.

D. H. Grollman and O. C. Jenkins (2010). Incremental learning of subtasks from unsegmented demonstration, in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on 2010*. Cited on page 70.

Y. Guiard (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model, *Journal of motor behavior*, vol. 19(4), pp. 486–517. Cited on page 72.

A. Gupta, A. A. Efros, and M. Hebert (2010). Blocks world revisited: Image understanding using qualitative geometry and mechanics, in *ECCV 2010*. Cited on page 100.

J. Hamrick, P. Battaglia, and J. B. Tenenbaum (2011). Internal physics models guide probabilistic judgments about object dynamics, in *Proceedings of the 33rd annual conference of the cognitive science society 2011*. Cited on pages 102, 103, and 104.

P. J. Hayes (1985). The second naive physics manifesto. Cited on page 20.

P. J. Hayes *et al.* (1978). The naive physics manifesto. Cited on page 20.

M. Hoai and F. De la Torre (2012). Max-margin early event detectors, in *CVPR 2012*. Cited on pages 12 and 40.

S. Hochreiter and J. Schmidhuber (1997). Long short-term memory, *Neural computation*, vol. 9(8), pp. 1735–1780. Cited on page 14.

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(2), pp. 328–373. Cited on pages 16, 17, 18, and 19.

A. J. Ijspeert, J. Nakanishi, and S. Schaal (2002). Learning Attractor Landscapes for Learning Motor Primitives, in *Advances in Neural Information Processing Systems 2002*. Cited on pages 16, 19, 69, 70, 74, and 76.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell (2014). Caffe: Convolutional architecture for fast feature embedding, in *Proceedings of the ACM International Conference on Multimedia 2014*. Cited on page 105.

L. P. Kaelbling (1993). Learning to achieve goals, in *IJCAI 1993*. Cited on page 118.

S. Karayev, T. Baumgartner, M. Fritz, and T. Darrell (2012). Timely Object Recognition, in *NIPS 2012*. Cited on page 46.

S. Karayev, M. Fritz, and T. Darrell (2014). Anytime Recognition of Objects and Scenes, in *CVPR 2014*. Cited on page 46.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). Large-scale video classification with convolutional neural networks, in *CVPR 2014*. Cited on pages 42 and 50.

C. C. Kemp and A. Edsinger (2006). Robot manipulation of human tools: Autonomous detection and control of task relevant features, in *Int. Conf. Development and Learning 2006*. Cited on pages 69, 71, and 72.

M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning, in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on 2016*. Cited on pages 33 and 119.

S. Kimura, T. Watanabe, and Y. Aiyama (2010). Force based manipulation of Jenga blocks, in *IROS 2010*. Cited on page 101.

K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert (2012). Activity forecasting, in *ECCV 2012*. Cited on pages 41 and 44.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto (2011). Robot learning from demonstration by constructing skill trees, *The International Journal of Robotics Research*, p. 0278364911428653. Cited on page 70.

H. S. Koppula and A. Saxena (2013). Anticipating human activities using object affordances for reactive robotic response. Cited on pages 41 and 44.

A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks, in *NIPS 2012*. Cited on pages 14, 22, 30, 104, and 105.

T. Kröger, B. Finkemeyer, S. Winkelbach, S. Molkenstruck, L. Eble, and F. M. Wahl (2006). Demonstration of multi-sensor integration in industrial manipulation (poster), in *ICRA 2006*. Cited on page 101.

J. R. Kubricht, K. J. Holyoak, and H. Lu (2017). Intuitive physics: Current research and controversies, *Trends in cognitive sciences*, vol. 21(10), pp. 749–759. Cited on page 20.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre (2011). HMDB: a large video database for human motion recognition, in *ICCV 2011*. Cited on page 40.

T. Lan, T.-C. Chen, and S. Savarese (2014). A Hierarchical Representation for Future Action Prediction, in *ECCV 2014*. Cited on pages 41 and 44.

I. Laptev (2005). On space-time interest points, *International Journal of Computer Vision*, vol. 64(2-3), pp. 107–123. Cited on page 13.

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld (2008). Learning realistic human actions from movies, in *CVPR 2008*. Cited on page 43.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, *et al.* (1995). Comparison of learning algorithms for handwritten digit recognition, in *International conference on artificial neural networks 1995*. Cited on page 105.

A. Lerer, S. Gross, and R. Fergus (2016). Learning Physical Intuition of Block Towers by Example, in *International Conference on Machine Learning 2016*. Cited on pages 12, 101, 116, and 117.

W. Li, S. Azimi, A. Leonardis, and M. Fritz (2016). To fall or not to fall: A visual approach to physical stability prediction, *arXiv preprint arXiv:1604.00066*. Cited on pages 5, 12, 97, 116, and 117.

W. Li, J. Bohg, and M. Fritz (2017a). Acquiring Target Stacking Skills by Goal-Parameterized Deep Reinforcement Learning, *arXiv preprint arXiv:1711.00267*. Cited on pages 5 and 116.

W. Li and M. Fritz (2012). Recognizing Materials from Virtual Examples, in *ECCV 2012*. Cited on page 100.

W. Li and M. Fritz (2015). Teaching robots the use of human tools from demonstration with non-dexterous end-effectors, in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on 2015*. Cited on pages 4 and 68.

W. Li and M. Fritz (2016). Recognition of ongoing complex activities by sequence prediction over a hierarchical label space, in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on 2016*. Cited on pages 4, 12, 38, and 133.

W. Li, A. Leonardis, and M. Fritz (2017b). Visual Stability Prediction for Robotic Manipulation, in *IEEE International Conference on Robotics and Automation 2017*. Cited on pages 5, 98, 117, 119, and 120.

L.-H. Lin (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching, *Machine learning*, vol. 8(3/4), pp. 69–97. Cited on page 32.

J. Liu, J. Luo, and M. Shah (2009). Recognizing realistic actions from videos “in the wild”, in *CVPR 2009*. Cited on page 40.

D. W. MacDougal (2012). Galileo’s Great Discovery: How Things Fall, in *Newton’s Gravity 2012*, Springer. Cited on page 100.

M. Marszalek, I. Laptev, and C. Schmid (2009). Actions in context, in *CVPR 2009*. Cited on pages 11 and 39.

M. McCloskey (1983). Intuitive physics, *Scientific american*. Cited on pages 21, 99, 100, and 133.

J. Meiss (2007). Dynamical systems, *Scholarpedia*, vol. 2(2), p. 1629. Cited on page 16.

R. Messing, C. Pal, and H. Kautz (2009). Activity recognition using the velocity histories of tracked keypoints, in *ICCV 2009*. Cited on page 39.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller (2013). Playing Atari With Deep Reinforcement Learning, in *NIPS Deep Learning Workshop 2013*. Cited on page 118.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.* (2015). Human-level control through deep reinforcement learning, *Nature*. Cited on pages 30, 121, 122, 130, and 133.

T. B. Moeslund, A. Hilton, and V. Krüger (2006). A survey of advances in vision-based human motion capture and analysis, *Computer vision and image understanding*. Cited on page 39.

R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi (2016). Newtonian scene understanding: Unfolding the dynamics of objects in static images, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*. Cited on pages 12, 99, 100, 116, and 117.

R. Mottaghi, C. Schenck, D. Fox, and A. Farhadi (2017). See the Glass Half Full: Reasoning About Liquid Containers, Their Volume and Content, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. Cited on page 117.

A. Y. Ng, D. Harada, and S. Russell (1999). Policy invariance under reward transformations: Theory and application to reward shaping, in *ICML 1999*. Cited on pages 27 and 122.

S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto (2013). Incremental Semantically Grounded Learning from Demonstration, in *Robotics: Science and Systems 2013 2013*. Cited on pages 70 and 78.

S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto (2012). Learning and generalization of complex tasks from unstructured demonstrations, in *IEEE/RS Int. Conf. Intelligent Robots and Systems (IROS) 2012*. Cited on pages 70 and 76.

S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, *et al.* (2011). A large-scale benchmark dataset for event recognition in surveillance video, in *CVPR 2011*. Cited on pages 11 and 39.

P. Over, G. Awad, J. G. Fiscus, B. Antonishek, M. Michel, W. Kraaij, A. F. Smeaton, and G. Quénot (2010). TRECVID 2010 - An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics, in *TRECVID 2010*. Cited on page 39.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal (2009). Learning and generalization of motor skills by learning from demonstration, in *IEEE Int. Conf. Robotics and Automation (ICRA) 2009*. Cited on pages 69 and 75.

X. Peng, B. Sun, K. Ali, and K. Saenko (2015). Learning Deep Object Detectors from 3D Models, in *ICCV 2015*. Cited on page 100.

R. Poppe (2010). A survey on vision-based human action recognition, *Image and vision computing*. Cited on page 39.

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, in *ICRA workshop on open source software 2009*. Cited on page 78.

R. G. Radwin, J. T. Haney, A. I. H. Association, E. Committee, *et al.* (1996). *An ergonomics guide to hand tools*, American Industrial Hygiene Association. Cited on page 71.

A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson (2014). CNN features off-the-shelf: an astounding baseline for recognition, in *CVPR workshops 2014*. Cited on page 104.

K. Rematas, T. Ritschel, M. Fritz, E. Gavves, and T. Tuytelaars (2016). Deep Reflectance Maps, in *CVPR 2016*. Cited on page 100.

K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars (2014). Image-based Synthesis and Re-Synthesis of Viewpoints Guided by 3D Models, in *CVPR 2014*. Cited on page 100.

M. D. Rodriguez, J. Ahmed, and M. Shah (2008). Action MACH a spatio-temporal Maximum Average Correlation Height filter for action recognition, in *CVPR 2008*. Cited on page 39.

M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele (2012). A database for fine grained activity detection of cooking activities, in *CVPR 2012*. Cited on pages 11, 38, and 40.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1985). Learning internal representations by error propagation, Technical report, DTIC Document. Cited on page 45.

M. Ryoo (2011). Human activity prediction: Early recognition of ongoing activities from streaming videos, in *ICCV 2011*. Cited on pages 12, 40, 41, and 44.

M. Ryoo and J. Aggarwal (2010). *UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities(SDHA)*. Cited on page 39.

S. Schaal and C. G. Atkeson (1998). Constructive incremental learning from only local information, *Neural computation*, vol. 10(8), pp. 2047–2084. Cited on page 18.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver (2016). Prioritized Experience Replay, in *International Conference on Learning Representations 2016*. Cited on page 32.

C. Schuldt, I. Laptev, and B. Caputo (2004). Recognizing human actions: a local SVM approach, in *ICPR 2004*. Cited on pages 11, 38, and 39.

G. Sergio, N. Krishnamoorthy, G. Malkarnenkar, T. Darrell, R. Mooney, and K. Saenko (2013). YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-Shoot Recognition, in *ICCV 2013*. Cited on page 40.

A. P. Shon, D. Verma, and R. P. Rao (2007). Active imitation learning, in *AAAI 2007*. Cited on page 19.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus (2012). Indoor segmentation and support inference from RGBD images, in *ECCV 2012*. Cited on page 100.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.* (2016). Mastering the game of Go with deep neural networks and tree search, *Nature*, vol. 529(7587), pp. 484–489. Cited on pages 33 and 130.

K. Simonyan and A. Zisserman (2015). Very deep convolutional networks for large-scale image recognition, in *ICLR 2015 2015*. Cited on page 105.

B. Smith and R. Casati (1994). *Naive Physics: An Essay in Ontology*, Philosophical Psychology. Cited on page 99.

R. Smith *et al.* (2005). Open dynamics engine. Cited on page 88.

S. Stein and S. J. McKenna (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities, in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing 2013*. Cited on pages 11 and 40.

A. Stoytchev (2005). Behavior-grounded representation of tool affordances, in *IEEE Int. Conf. Robotics and Automation (ICRA) 2005*. Cited on page 69.

R. S. Sutton (1988). Learning to predict by the methods of temporal differences, *Machine learning*, vol. 3(1), pp. 9–44. Cited on page 29.

R. S. Sutton and A. G. Barto (1998). *Introduction to reinforcement learning*, vol. 135, MIT Press Cambridge. Cited on pages 25, 28, and 29.

M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter (2011). Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives, *Robotics and Autonomous Systems*, vol. 59(11), pp. 910–922. Cited on page 70.

K. Tang, L. Fei-Fei, and D. Koller (2012). Learning latent temporal structure for complex event detection, in *CVPR 2012*. Cited on page 42.

M. Tenorth, J. Bandouch, and M. Beetz (2009). The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition, in *ICCV Workshops 2009*. Cited on pages 11 and 39.

T. Tieleman and G. Hinton (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, *COURSERA: Neural networks for machine learning*, vol. 4(2), pp. 26–31. Cited on page 31.

M. Toussaint, N. Plath, T. Lang, and N. Jetchev (2010). Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference, in *Robotics and Automation (ICRA), 2010 IEEE International Conference on 2010*. Cited on page 119.

H. Van Hasselt, A. Guez, and D. Silver (2016). Deep Reinforcement Learning with Double Q-Learning., in *AAAI 2016*. Cited on page 32.

O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, *et al.* (2017). StarCraft II: a new challenge for reinforcement learning, *arXiv preprint arXiv:1708.04782*. Cited on page 33.

H. Wang, A. Klaser, C. Schmid, and C.-L. Liu (2011). Action recognition by dense trajectories, in *CVPR 2011*. Cited on pages 13, 42, and 50.

J. Wang, P. Rogers, L. Parker, D. Brooks, and M. Stilman (2009). Robot Jenga: Autonomous and strategic block extraction, in *IROS 2009*. Cited on pages 101 and 114.

L. Wang, Y. Qiao, and X. Tang (2015). Action recognition with trajectory-pooled deep-convolutional descriptors, in *Proceedings of the IEEE conference on computer vision and pattern recognition 2015*. Cited on page 14.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas (2016). Dueling Network Architectures for Deep Reinforcement Learning, in *International Conference on Machine Learning 2016*. Cited on page 32.

C. J. Watkins and P. Dayan (1992). Q-learning, *Machine learning*, vol. 8(3-4), pp. 279–292. Cited on pages 29 and 121.

D. Weinland, R. Ronfard, and E. Boyer (2011). A survey of vision-based methods for action representation, segmentation and recognition, *Computer Vision and Image Understanding*. Cited on page 39.

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid (2013). Deepflow: Large displacement optical flow with deep matching, in *ICCV 2013*. Cited on page 109.

J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum (2015). Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning, in *NIPS 2015*. Cited on pages 99, 100, 116, and 117.

I. Yildirim, T. Gerstenberg, B. Saeed, M. Toussaint, and J. Tenenbaum (2017). Physical problem solving: Joint planning with symbolic, geometric, and dynamic constraints, *arXiv preprint arXiv:1707.08212*. Cited on pages 116 and 117.

J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici (2015). Beyond short snippets: Deep networks for video classification, in *Proceedings of the IEEE conference on computer vision and pattern recognition 2015*. Cited on page 14.

B. Zheng, Y. Zhao, J. Yu, K. Ikeuchi, and S.-C. Zhu (2013). Beyond point clouds: Scene understanding by reasoning geometry and physics, in *CVPR 2013*. Cited on page 100.

B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba (2016). Learning Deep Features for Discriminative Localization., *CVPR*. Cited on page 108.

R. Zöllner, T. Asfour, and R. Dillmann (2004). Programming by demonstration: dual-arm manipulation tasks for humanoid robots., in *IEEE/RS Int. Conf. Intelligent Robots and Systems (IROS) 2004*. Cited on page 72.