March 2019

Robotic Motion Generation by Using Spatial-Temporal Patterns from Human Demonstrations

Yongqiang Huang

University of South Florida, dragonhyq@gmail.com

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the Artificial Intelligence and Robotics Commons, and the Robotics Commons

Scholar Commons Citation

Huang, Yongqiang, "Robotic Motion Generation by Using Spatial-Temporal Patterns from Human Demonstrations" (2019). USF Tampa Graduate Theses and Dissertations.
https://digitalcommons.usf.edu/etd/8373

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.
Robotic Motion Generation by Using Spatial-Temporal Patterns from Human Demonstrations

by

Yongqiang Huang

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Yu Sun, Ph.D.
Lawrence Hall, Ph.D.
Sudeep Sarkar, Ph.D.
Rajiv Dubey, Ph.D.
Chad Dube, Ph.D.

Date of Approval:
March 18, 2019

Keywords: learning from demonstration, trajectory generation, velocity generation

Copyright © 2019, Yongqiang Huang
DEDICATION

To God and my parents, whose love made all this possible.
ACKNOWLEDGMENTS

Most directly it is my advisor, Dr. Sun Yu. Can I ask for a better advisor? No. Patience, rigor, care, dedication, you name it. I feel so lucky and I cannot imagine a better advisor. Thank you SO much.

I would like to thank my committee members: Dr. Lawrence Hall, Dr. Sudeep Sarkar, Dr. Rajiv Dubey, and Dr. Chad Dube for providing insightful comments and encouragement.

Shanie Lightbourne, Gabriela Franco, and Laura Owczarek. Thank you for meeting my endless needs and being friendly to me.

My labmates! Lin Yun, Lin Bing Xiong, John Rippetoe, David Paulius, Troi Williams, David Bucanan, Jeanine Sam, James Robe, Jeanine Sam, Joseph Cox, Jason Melancon, Ahmad Jelodar, Roger Milton, Chen Tian Ze. It has been great working with you. Thank you very much.

Rui Ran, Zhao Wen, Guo Fei, Zhao Bi, Mou Cheng Cheng, Bu Kun, Liu Ren Hao, Cao Yan, Cao Yu Ting, Fang Song, Guo Lu Kai, Jin Yan, Li Hao, Li Yi, Liu Guo Jun, Shen Da Kun, Sun Xu Xue, Zhou Mu, Zuo Xiang, Zhou Yi, Zhao Nan, Lv Xing, Martin Llufriou, RaviKiran Krishnan, Santosh Aditham, Fellipe Souza, Zhang Fan, Huang Li Jie, Shen Xi, Xu Tian Shu, Chen Jin. Thank you for the fun, company, help, and kindness.

I would also like to thank the Chinese Christian Fellowship and the local Bible study group now led by Greg and Jaime, for accepting me and for peace.

My family, all of you, who supported me, believed in me, cared about me, worried about me. I feel like a prince.

Throughout all the difficulties, God and my parents are with me. A thank you is so insignificant that it almost becomes an insult. I am indebted, to say the least. The deepest
security, is what I gain from my God and my parents. Understanding, supporting, and saving. I merely received.

It has not been easy, and I would not have done this without anyone. The world is scary, but with you, it is less scary. The least I can say:

Thank you.
TABLE OF CONTENTS

LIST OF TABLES iv
LIST OF FIGURES v
ABSTRACT viii

CHAPTER 1 INTRODUCTION 1
  1.1 Related Work 3
    1.1.1 Motion Synthesis 3
    1.1.2 Robotic Trajectory Generation 4
  1.2 Data Necessity 6
  1.3 Contribution 7

CHAPTER 2 REVIEW OF OBJECT MANIPULATION DATASETS 8
  2.1 Note to Reader 8
  2.2 Introduction 8
  2.3 Datasets of Cooking Activity 9
    2.3.1 Slice&Dice 10
    2.3.2 CMU-MMAC 12
    2.3.3 GTEA 12
    2.3.4 Gaze and Gaze+ 13
    2.3.5 MPII Cooking, Cooking Composite, and Cooking 2 13
    2.3.6 50 Salad 14
    2.3.7 Actions for Cooking Eggs (ACE) 15
    2.3.8 YouCook 15
    2.3.9 Actions for Making Cereal 15
    2.3.10 Brown Breakfast 16
    2.3.11 FOON 16
    2.3.12 Epic Kitchen 17
    2.3.13 Summary 17
  2.4 Datasets of Activities of Daily Living (ADL) 21
    2.4.1 TUM Kitchen 22
    2.4.2 Rochester ADL 22
    2.4.3 OPPORTUNITY 23
    2.4.4 Cornell CAD-60 and CAD-120 23
LIST OF TABLES

Table 2.1 Publication year of datasets 9
Table 2.2 Number of instances provided by each dataset 10
Table 2.3 Links to datasets 11
Table 2.4 Modalities for cooking 18
Table 2.5 Annotated activities shared by at least 4 cooking datasets 21
Table 2.6 Modalities ADL 28
Table 2.7 Shared annotated ADLs. 32
Table 3.1 The count for each modality for each motion 39
Table 3.2 Modality coverage throughout the entire data. 40
Table 6.1 Accuracy for pouring water from different source containers 110
Table 6.2 Accuracy of pouring liquids with different viscosities 114
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Count of datasets for each modality</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Count of datasets for each modality</td>
<td>31</td>
</tr>
<tr>
<td>3.1</td>
<td>The structure that connects the tool, the FT sensor and the PO sensor</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>Tracking both the tool and the object with two PO sensors</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Examples of the tools that we have adapted</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>The tool’s base point is the center of the tooling side of the FT sensor</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Coordinate frames for the ground plane and the trackable of OptiTrack</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Coordinate frames of Patriot source and sensor placed on the same plane</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Coordinate frames of the FT and PO sensors</td>
<td>45</td>
</tr>
<tr>
<td>3.8</td>
<td>The structure of the dataset where the red text is verbatim.</td>
<td>47</td>
</tr>
<tr>
<td>3.9</td>
<td>The structure of a non-OptiTrack csv data file.</td>
<td>48</td>
</tr>
<tr>
<td>3.10</td>
<td>Formats of the columns for PO for one and two sensors</td>
<td>49</td>
</tr>
<tr>
<td>3.11</td>
<td>Axes and yaw-pitch-roll correspondence for the PO sensor</td>
<td>50</td>
</tr>
<tr>
<td>3.12</td>
<td>Visualizing the PO data</td>
<td>50</td>
</tr>
<tr>
<td>3.13</td>
<td>The organization of the pouring data where the red text is verbatim</td>
<td>52</td>
</tr>
<tr>
<td>4.1</td>
<td>The proposed approach consists of three steps</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustration of different eigenfunctions carrying different variation</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>We define the right arm of NAO as the kinematics chain.</td>
<td>63</td>
</tr>
<tr>
<td>4.4</td>
<td>First three eigenfunctions of dataset beat for each degree</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison between our method and OMPL and LSPB</td>
<td>65</td>
</tr>
</tbody>
</table>
Figure 4.6 Result of experiment *obstacle* with sufficient guidance from via points

Figure 4.7 Another result of experiment *obstacle* with sufficient via point-guidance

Figure 4.8 Result of experiment *clearing obstacle* with poor guidance from via points

Figure 5.1 Mechanism inside an LSTM unit, Zaremba’s version [1]

Figure 5.2 The architectures of (left) \( \text{frc} \), (middle) \( \text{vel} \), (right) \( \text{stp} \)

Figure 5.3 Initializing \( \text{frc}, \text{vel}, \text{stp} \)

Figure 5.4 3D printed adapter

Figure 5.5 Generalizing on an unseen cup.

Figure 5.6 Generalizing on an unseen container

Figure 5.7 Generalizing on an unseen material

Figure 5.8 Generalizing on an unseen cup and container

Figure 5.9 Generalizing on an unseen container and material

Figure 5.10 Generalizing on an unseen cup and material

Figure 5.11 Generalizing on an unseen cup, container, and material

Figure 6.1 The sequential pouring system

Figure 6.2 Mechanism of a peephole LSTM unit

Figure 6.3 Six physical quantities to obtain for pouring

Figure 6.4 Illustration of the data collection setup

Figure 6.5 All the source containers used in training and in the experiments

Figure 6.6 An example of the network with two LSTM layers and the final layer

Figure 6.7 Our network has 1 layer and 16 peephole LSTM units

Figure 6.8 The physical evaluation system

Figure 6.9 Actual-vs-target comparison of water pouring in experiments

Figure 6.10 Evaluating minimum accurate amount

Figure 6.11 Force sensor reading of a 1 lbf-weight taken during 300 seconds
| Figure 6.12 | Actual-vs-target comparison of one human subject pouring water | 117 |
| Figure 6.13 | Pairs of initial and target force in the training data and the experiments | 118 |
| Figure 6.14 | Pairs of height and diameter in the training data and the experiments | 119 |
| Figure 6.15 | Comparisons of normalized histograms of minimum static distances | 120 |
| Figure 6.16 | Comparisons of normalized histograms of minimum DTW distances | 120 |
| Figure 6.17 | Actual-vs-target comparison of pouring oil using the red cup. | 121 |
| Figure 6.18 | Actual-vs-target comparison of pouring syrup using the red cup. | 122 |
| Figure 6.19 | Pouring accuracy of liquids with different viscosity | 123 |
Robots excel in manufacturing facilities because the tasks are repetitive and do not change. However, when the tasks change, which happens in almost all tasks that humans perform daily, such as cutting, pouring, and grasping, etc., robots perform much worse. We aim at teaching robots to perform tasks that are subject to change using demonstrations collected from humans, a problem referred to as learning from demonstration (LfD).

LfD consists of two parts: the data of human demonstrations, and the algorithm that extracts knowledge from the data to perform the same motions. Similarly, this thesis is divided into two parts. The first part discusses what related datasets exist, how each dataset fits and does not fit our purpose, and how we collected our own dataset. The second part presents two approaches for generating robotic manipulation motions.

The first approach uses functional principal component analysis to break down a motion into simpler components, each component carrying a certain pattern of variation throughout the complete execution duration. New motions are built using those components, with certain constraints specified by the users. We used this approach to generate motions with the arm.

The second approach uses recurrent neural networks as its framework which solves the drawbacks that we identified in the first approach. The essence of the approach is a velocity generator that runs forward in time. A trajectory is generated through the execution of the velocities. We particularly used this approach to solve the problem of accurate pouring. We evaluated the approach on a physical system and achieved high accuracy when pouring water from different source containers.
CHAPTER 1

INTRODUCTION

Robots do well in manufacturing in which they carry out the same tasks by executing certain programs repeatedly. While at work, the robots are fixed in a position, the objects they interact with have predictable positions and orientations, and the environment in which the interactions take place does not change. With repetitive precision, robots have proved helpful in factories. Robots in assembly lines, however, cannot adapt to changes that are outside the definition of their program.

Humans carry out a variety of tasks daily. Day after day, a person may repeat the same set of tasks, or add to, or subtract a few tasks. It is worth noticing though, that even if a person carries out only one single task again and again, he or she would hardly do exactly the same thing every time, certainly not to the precision of a robot in an assembly line. We take slicing food as an example. In different trials, the food that is to be sliced can be a carrot, cucumber, celery, broccoli, eggplants, etc., the knife can have different lengths and different shapes of blades, and the chop board can vary in size, material, and thickness. Except for the physical identities, the person may want to slice to different shapes, from different angles, and fast or slowly. It is obvious that we have only named very few examples of variations. Numerous factors can vary, but one thing remains: the performed action which is slicing. It is curious, if not shocking, that the actual execution of the same task allows for so many, possibly infinite variations.
It is one of the goals in the robotic community that robots be able to execute tasks with the same flexibility as humans do. Exhaustive programming is infeasible because the number of cases (which can be infinite) that need to be covered is too big. We can only use finite, and possibly a small amount of data to teach the task to the robots wisely so that they can successfully execute the task even if the environment changes. As humans invent and master the tasks, data that are directly collected from humans are naturally invaluable. The class of methods which use data collected from humans to teach tasks to robots offline are referred to as Learning from Demonstration (LfD), Programming by Demonstration (PbD), or Imitation Learning (IL) [2, 3, 4]. The key criterion of a LfD algorithm is whether the learned skill can generalize to a new environment.

In this dissertation, we focus on trajectory level learning, which means the output of the algorithm is the execution trajectory of the task, either in joint space or in world space. The trajectory can be generated either as a whole which means all the points in the trajectory are generated at the same time, or it can be generated point by point in order. The first kind of trajectory generation is less flexible than the second kind because the entire trajectory has to change if any point in the trajectory needs to be changed, and it is also inconsistent with the fact that a task is executed gradually in time rather than in an instant. For the second kind of trajectory generation, the trajectory is essentially generated by accumulated velocities in time. The task can be represented as a dynamical system which maps the current state to a command that is to be executed. The mapping is also referred to as a policy, and thus the trajectory generation algorithm can be referred to as a policy generation algorithm.

The problem of learning the dynamical system is similar to what is targeted by Reinforcement Learning (RL) [5], in which the policies are learned through trial-and-error, i.e. letting the dynamical system execute certain policies in a certain environment and observing corresponding outcomes. LfD differs from RL in that LfD does not assume that it is possible to execute the policy while the learning is taking place.
1.1 Related Work

The problem of trajectory generation has found solutions in both graphics and robotics. In graphics, trajectory generation is referred to as motion synthesis, whose goal is to synthesize novel and naturalistic motions for animated characters. In robotics, a trajectory generation algorithm is intended to be deployed to a real physical systems. We review related work in both fields.

1.1.1 Motion Synthesis

One popular solution is Hidden Markov models (HMM) which is a popular temporal representation of human motion. In [6], HMM is used on the second level of the bi-level motion model. In [7], a multidimensional HMM is used to encrypt the styles of motions, and to generate new styles. Similar to [7], also focusing on variation/styles of motion, [8] learns the structure of a dynamic Bayesian network, which has the ability to synthesize both temporal and spatial variants of the original motion.

Another solution is linear dynamical systems (LDS). In [9], LDS is used for modeling motion textons and transition matrices for texton distribution, and synthesizes motion sequences with constrained LDS, while [10] presents motions with LDS, and generates new motions using an optimization technique which considers both Gaussian-modeled motion priors and user defined constraints.

A third tool, Gaussian Process (GP) is used in [11] to model the force field that exists in the motion, and combines it with a Newtonian dynamics model to synthesize new motions. In [12], GP is used to model transitions between morphable primitives, which is a model that encrypts geometric and time variation.

The use of graphs also provide a plethora of solutions. In [13], the motion is modeled as a directed graph where the edges represent clips of motion and the nodes represent their
connections. The branch and bound algorithm is used to search for a path that meets the user’s requirements. In [14], a hierarchical graph is built to represent motion sequences, and random search was used to look for paths that accommodate user constraints. On the basis of [13, 14], and [15], [16] builds a compressed interpolated motion graph, and uses ARA*, an anytime heuristic search algorithm to find the optimal or sub-optimal path in the graph that meets the path sketched by the user and also compute the interpolation weights.

1.1.2 Robotic Trajectory Generation

One popular solution for robotic trajectory generation is Gaussian mixture regression (GMR), which is based on Gaussian mixture models (GMM). First, each data point of a trajectory is augmented with the time stamp, thus the trajectory loses its nature as a trajectory and becomes a set of data points with one extra dimension which is the time stamp. Then we learn a GMM from the converted data. The learned GMM models the joint probability of the actual data and the time. To generate a new trajectory, a conditional distribution of the data conditioned on the time stamp is derived from the GMM, and which gives the distribution of a data point at any certain time stamp for the new trajectory.

Learning the GMM can be done in batch using Expectation Maximization (EM) [17] or incrementally [18]. GMR can be used to generate a new trajectory while considering both world space and configuration space constraints, which are connected through the Jacobian [19]. Instead of generating trajectories using specific time stamps, the GMM can learn the system state which includes position and velocity to model a dynamical system [20]. Task-parameterized GMR extends GMR to incorporate demonstrations observed from multiple frames [21].

Movement primitives (MP) is another class of trajectory generation approaches that include many variants. The first of the MPs, the dynamic movement primitives (DMP) is a non-linear dynamical system that consists of three components: a point attractor system
that guarantees the convergence of the system state to a goal state, a forcing function which contains the trajectory the system is expected to go through, and a canonical system that controls the temporal profile of the system [22]. In the original design, the point attractor system applies a strictly damped string model, the forcing function applies a basis expansion of the expected trajectory, and the canonical system is modeled using a first-order dynamical system. The choice of specific implementation of each component, however, is not fixed, and is free to change depending on the tasks at hand.

DMP is capable of modeling discrete movement such as swinging a tennis racket [23], playing table tennis [24] as well as rhythmic movement such as drumming [25] and walking [26]. Interactive primitives (IP) which is based on DMP learns the correlation between the actions of two interacting agents [27]. IP learns a conditional probability of the DMP parameters of the interacting agent conditioned on the incomplete trajectory of the observed agent and uses the probability to infer how to react.

Probabilistic movement primitives (ProMP) keeps two practices of DMP: 1) using basis functions to represent a reference trajectory, and 2) using a phase variable for temporal modulation [28]. Different from DMP, it does not involves a dynamical system but rather keeps a distribution of the parameters of the basis function expansion. The distribution can be updated in batch or incrementally when new demonstrated trajectories become available, and which then can be used to generate new trajectories with constraints in forms of via points of positions or velocities. The concept of IP can be applied to ProMP to learn the interaction between two agents [29]. The Interactive ProMP is extended by including GMM to learn multiple interactive patterns [30].

Principal Component Analysis (PCA) also proves useful for motion generation. Known as a dimension reduction technique used on the dimensionality axis of the data, PCA can be used on the time axis of motion trajectories instead to retrieve geometric variations [31]. Besides, PCA can be applied to find variations in how the motion progresses in time, which,
combined with the variations in geometry enables generating motions with more flexibility [32]. Functional PCA (fPCA) extends PCA by introducing continuous-time basis functions and treating trajectories as functions instead of collections of points [33, 34]. In [35], fPCA is applied for producing trajectories of gross motion such as answering a phone and punching, and for making the trajectories avoid obstacles with the guidance of quality via points. [36] uses fPCA for generating trajectories of fine motion such as pouring.

Designed to handle time sequences, recurrent neural networks (RNN) have recently been chosen more often for sequence generation [37]. RNN is capable of modeling general dynamical systems [38, 39], and in comparison to non-dynamical GMR, MP, and fPCA, it does not require temporal alignment of the demonstrations. As more manipulation datasets become available [40], it becomes feasible to learn a deep RNN. The dataset has been used to generate trajectories of pouring [41]. Linear dynamical systems, an approach that is closely related to RNN can handle time series as well [42].

1.2 Data Necessity

Datasets are valuable in various scientific fields because they are crucial for testing an algorithm. The demands for datasets follow the advancement of a field or the evolution of a problem, and new datasets never stopped being created. A good dataset may not only be used to verify or deny the correctness and effectiveness of an algorithm, but may also help expose the flaws or exemplify the strength of the algorithm. To choose a good dataset, one first needs to know what datasets are available, what they include, and how they differ. Then one can decide on whether any datasets would be useful and which one or several would best serve the research purpose. One may also decide that none of the datasets suits the purpose, and the reason on which that particular decision is made can be used to improve on the existing datasets and make new ones.
As we aim to devise algorithms that learn from human demonstration data, we need to find the proper datasets, or if we cannot find any, we need to create our own dataset.

1.3 Contribution

The contribution of this thesis include

1. Reviewed 28 datasets related to object manipulation, among which 15 are related to cooking the other 13 related to activities of daily living

2. Collected an object motion dataset which includes over 1,500 sequences of more than 30 types of motions of cooking and activities of daily living.

3. Presented motion harmonics as a trajectory generation approach, which can incorporate user-defined constraints and which has the potential of working with motion planners.

4. Presented the application of recurrent neural network as a framework for generating pouring trajectories.

5. Presented a pouring algorithm based on a recurrent neural network that poured liquid with high accuracy. Evaluated the accurate pouring algorithm using a real physical system. Showed that the proposed algorithm generalized.
CHAPTER 2

REVIEW OF OBJECT MANIPULATION DATASETS

2.1 Note to Reader

Portions of this chapter have been previously published in “Recent Data Sets on Object Manipulation: A Survey”, Big Data 2016 4:4, pp. 197-216, and have been reproduced with permission from Mary Ann Liebert, Inc.

2.2 Introduction

In this chapter, we review datasets that we consider useful for research on object manipulation. The introduced datasets were published no earlier than 2009. Object manipulation is the process of changing, in a controlled fashion, the position and orientation of an object in order to execute a specific task. In contrast to a gross motion such as waving and stretching, an object manipulation motion is a fine motion, and the body parts involved cover a much smaller physical space. We report on datasets that focus on object manipulation motion. Gross motions may be present in certain datasets, but do not play the dominant role.

We divide our review into two categories and present them separately: those that include mostly cooking activities, in Section 2.3 and those that include more general activities of daily living (ADL), in Section 2.4. All datasets are summarized in Table 2.1 that classifies the datasets according to the year that they were published. In Table 2.2, we list the number of instances provided in each dataset. When a dataset contains sequences, we report the number of sequences; otherwise, we report the number of data samples.
### Table 2.1. Publication year of datasets

<table>
<thead>
<tr>
<th>Year (20-)</th>
<th>09</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datasets</td>
<td>[43]</td>
<td>[44]</td>
<td>[45]</td>
<td>[46]</td>
<td>[47]</td>
<td>[48]</td>
<td>[49]</td>
<td>[50]</td>
<td>[51]</td>
</tr>
<tr>
<td></td>
<td>[52]</td>
<td>[53]</td>
<td>[46]</td>
<td>[54]</td>
<td>[55]</td>
<td>[56]</td>
<td>[58]</td>
<td>[59]</td>
<td>[60]</td>
</tr>
<tr>
<td></td>
<td>[62]</td>
<td>[63]</td>
<td>[64]</td>
<td>[65]</td>
<td>[66]</td>
<td>[67]</td>
<td>[68]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In each category, we present the datasets in ascending chronological order. For each dataset, we report on the modalities, the activities performed, and annotations, and we give our view on how each dataset relates to object manipulation. After reporting on the datasets one-by-one, we summarize them on the availability of modalities, object identifiability in annotated activities, and the forms of temporal segmentation of annotated activities. We also provide the lists of shared annotated activities for the ADL and cooking datasets, respectively.

For those who want to further examine the datasets covered in this work, we provide the links to all datasets in Table 2.3.

### 2.3 Datasets of Cooking Activity

In this section, we present 15 datasets of cooking activities. The interest in studying cooking activities is motivated by the large number of interactions with the objects and the external environment that human hands and body usually undergo. The datasets include common visual-based acquisition modalities such as RGB vision and depth vision, as well as modalities that are less common such as skin temperature and body heat. RGB vision is used by all datasets. We first present each dataset individually, describing the different characteristics; data type and size, modalities, equipment, annotations etc. Then, we compare
the datasets on their different descriptive fields and discuss their suitability and applicability for LfD.

2.3.1 Slice&Dice

Slice&Dice [43] features four instrumented utensils which include three knives of different sizes and a spoon. Each utensil embeds in its handle a 3-axis accelerometer. Twenty subjects participated and each subject prepared a salad or a sandwich freely using the ingredients provided by the experimenter. The acceleration data are accompanied by RGB videos. We consider embedding accelerometers inside objects a merit as, unlike vision based sensors,
<table>
<thead>
<tr>
<th></th>
<th>Links to datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>[52]</td>
<td><a href="http://kitchen.cs.cmu.edu/">http://kitchen.cs.cmu.edu/</a></td>
</tr>
<tr>
<td>[53]<a href="+">46</a></td>
<td><a href="http://www.cbi.gatech.edu/fpv/">http://www.cbi.gatech.edu/fpv/</a></td>
</tr>
<tr>
<td>[54]</td>
<td><a href="http://www.murase.m.is.nagoya-u.ac.jp/KSCGR/">http://www.murase.m.is.nagoya-u.ac.jp/KSCGR/</a></td>
</tr>
<tr>
<td>[59]</td>
<td><a href="http://web.eecs.umich.edu/~jjcorso/r/youcook/">http://web.eecs.umich.edu/~jjcorso/r/youcook/</a></td>
</tr>
<tr>
<td>[51]</td>
<td><a href="https://epic-kitchens.github.io/2018">https://epic-kitchens.github.io/2018</a></td>
</tr>
<tr>
<td>[57]</td>
<td><a href="https://ias.in.tum.de/software/kitchen-activity-data">https://ias.in.tum.de/software/kitchen-activity-data</a></td>
</tr>
<tr>
<td>[64]</td>
<td><a href="http://www.eng.yale.edu/grablab/humangrasping/">http://www.eng.yale.edu/grablab/humangrasping/</a></td>
</tr>
<tr>
<td>[61]</td>
<td><a href="https://sites.google.com/site/brainrobotdata/home/push-dataset">https://sites.google.com/site/brainrobotdata/home/push-dataset</a></td>
</tr>
<tr>
<td>[56]</td>
<td><a href="https://sites.google.com/site/brainrobotdata/home/grasping-dataset">https://sites.google.com/site/brainrobotdata/home/grasping-dataset</a></td>
</tr>
<tr>
<td>[50]</td>
<td><a href="https://github.com/jrl-umi3218/ManipulationKinodynamics">https://github.com/jrl-umi3218/ManipulationKinodynamics</a></td>
</tr>
</tbody>
</table>

[46](+) refers to both Gaze and Gaze+
they provide acceleration data that belong to a certain object alone, and are readily usable without running object recognition first.

2.3.2 CMU-MMAC

The CMU-MMAC dataset [52] contains multi-modal cooking activities for five recipes: brownie, eggs, pizza, salad, and sandwich. The modalities include RGB videos from static and wearable cameras, multi-channel audios, motion capture, inertial measurement units (IMU), RFID, etc. We are not positive on the number of subjects that were involved, but we infer that it is between thirty-nine and forty-five. Each subject prepared all the recipes. The dataset also specifically recorded anomalous accidental events that occurred while cooking. Certain modalities are incomplete for certain recipes performed by certain subjects. Annotations exist for sixteen subjects while preparing brownies and correspond to the videos captured by the wearable camera. The annotations apply the structure of “verb+objectOne+preposition+objectTwo”, whose components are assembled using grammar.

Except RFID tagging which merely reports the involvement of certain objects, all modalities are on humans, which is contrary to the Slice&Dice dataset [43]. The dataset is rich in data of upper arm motions because of the combined use of motion capture and IMUs, and therefore is suitable for 3D manipulation motion analysis.

2.3.3 GTEA

The GTEA dataset [53] includes egocentric videos of four subjects performing seven food/beverage preparing activities. The videos amount to 31,222 RGB images. Annotations consist of simple verbs (such as put, take, pour, etc.) and names of objects (cup, sugar, etc.). Object recognition or manually drawn bounding boxes on objects is required prior to analysis of the object motion.
2.3.4 Gaze and Gaze+

The Gaze dataset [46] contains RGB egocentric videos of fourteen subjects preparing meals using provided ingredients on a table. The videos were captured using an eye-tracking camera and therefore are accompanied by gaze data. The Gaze+ dataset [46] (later referred to as [46]+) is an upgrade to Gaze, and provides the two modalities in Gaze plus audio. The videos have higher resolution than Gaze, and were captured in an instrumented kitchen instead of on a simple table. Ten subjects were involved and each one of them prepared a set of seven dishes. Actions and objects were annotated in the same way as in Gaze. Compared to static images, egocentric images have much larger proportions of the image showing object manipulation specifically and contain more detail, which we consider a merit. Analyzing object motion, however, would assume that object tracking has been done.

2.3.5 MPII Cooking, Cooking Composite, and Cooking 2

MPII sequentially created three datasets related to cooking: the MPII Cooking dataset [58] which focuses on fine grained activity, the MPII Cooking Composite dataset [62] which focuses on composite activities composed of basic-level activities, and the MPII Cooking 2 dataset [49] which unifies and is an upgrade of both [58] and [62].

The MPII Cooking dataset involved twelve subjects each preparing one to six out of fourteen dishes, and contains forty-four RGB high-definition (HD) videos with a total length of over eight hours or 881,755 frames. The annotations include sixty-five activities, and 5,609 instances were identified.

The MPII Cooking Composite dataset included all the videos from the MPII Cooking dataset and added 212 newly-recorded videos. Eighteen more subjects than in the MPII Cooking dataset participated. Different from the MPII Cooking dataset, the MPII Cooking Composite dataset annotations include four categories: activities (e.g. verbs), ingredients,
tools, and containers, which combined are referred to as “attributes”. There exist 218 attributes in the dataset, among which seventy-eight are activities. A total of 49,258 attribute instances have been identified which belong to 12,642 annotated temporal segments.

As a refined superset of [58] and [62], the MPII Cooking 2 dataset contains 273 videos involving thirty subjects. The dataset contains fifty-nine dishes, which consist of fourteen diverse and complex dishes from [58], and forty-five shorter and simpler composite dishes from [62]. A total of 222 attributes exist, among which eighty-seven are activities. 54,774 attribute instances have been identified which belong to 14,105 temporal segments. For the above MPII datasets, the subjects were only told which dish to prepare, which lead to natural activities with much variability.

Of all the datasets we include in this work, the MPII datasets altogether have the largest number of HD videos and annotation instances. Objects and fine actions are annotated in great detail, and 2D poses of the upper body are also provided. For vision-based 2D object manipulation analysis, the amount of data and action variability of the MPII datasets can only be rivaled by the Brown breakfast dataset [55], if it is not unmatched.

2.3.6 50 Salad

The 50 Salad dataset [47] extends Slice&Dice [43] by using accelerometers on more utensils and by including depth videos in addition to RGB ones. Twenty-five subjects participated and each prepared a mixed salad twice, and in each run followed a specific sequence of tasks. The sequences were produced by a statistical activity diagram, which would theoretically enable the same number of samples for each task sequence.

The annotation includes three high-level activities: prepare dressing, cut and mix ingredients, and serve salad. Each high-level activity summarizes several low-level activities, and each low-level activity has -pre, -core, and -post phases, which were annotated respectively. 50 Salad inherits the merit of Slice&Dice [43], involves more subjects, enables 3D analysis.
with depth videos, and has finer annotations. In that regard, we recommend 50 Salad over Slice&Dice.

2.3.7 Actions for Cooking Eggs (ACE)

The ACE dataset [54] contains RGB-D videos of cooking activities for five egg menus, all of which were cooked by each of seven subjects. The labels contain only verbs: break, mix, bake, turn, cut, boil, season, and peel. We include this dataset because it provides fine object manipulation motion, but since objects are not identified in any way, using the dataset would rely on human and object tracking more heavily than other datasets.

2.3.8 YouCook

The YouCook dataset [59] consists of eighty-eight RGB cooking videos downloaded from Youtube. All the videos have a third person point of view. Although only seven actions labels are used, as many as forty-eight object labels spanning seven object categories exist, and object tracks are provided. We consider the richness of object labels and the availability of the objects tracks as the merits of the dataset, of which the latter facilitates analysis of fine motion in 2D.

2.3.9 Actions for Making Cereal

In [48] the data of eight subjects are included while preparing cereal. The dataset includes multiple modalities, including RGB-D videos, audios, estimated six degree-of-freedom (DOF) object pose trajectories, and object mesh models. We consider the object pose trajectories as the merit of the dataset. No other datasets that we include provide such a modality, and using the trajectories alone suffices to conduct analysis on 3D object manipulation.
2.3.10 Brown Breakfast

The Brown breakfast dataset [55] contains roughly seventy-seven hours of RGB videos involving fifty-two subjects captured at up to eighteen distinct kitchens. In total ten recipes were performed and each subject was reported to have performed all ten recipes, but available data for different subjects vary. Forty-eight coarse activity annotations exist and 11,267 annotation instances were identified. The statistics of the dataset makes it a possible rival of the MPII datasets. It has the largest number of video frames (non HD) among the datasets we include, more than the MPII datasets by 50%. The number of coarse annotation instances is not much lower than the MPII datasets, but the detail and richness of the annotations could not compete with MPII. The dataset does include fine activity annotations, but the statistics and the description of the formation of such annotations are not yet available. Compared with MPII, the dataset lacks 2D upper body pose annotations.

2.3.11 FOON

The functional object-oriented network (FOON) [68] is a knowledge database or representation of cooking motions. It is a bipartite graph that consists of multiple functional units. A functional unit is a small directed graph that represents the change of the states of certain objects as a result of a certain motion. A functional unit represents an atomic motion, and multiple functional units chained together represent complicated motions. Thus, FOON which contains various functional units connected together provides a multitude of recipes with every step in detail. Specifically, it contains the objects involved and the motion required for each minuscule step of a recipe.

FOON currently contains the knowledge from 65 cooking videos. The object states and motions in each video were annotated, made into functional units, and merged into FOON. Thus, FOON is a database where knowledge and information has been organized, rather
than a collection of raw data. FOON comes from RGB videos but does not contains videos per se. It is readily useful for cooking motion analysis and planning in robotics.

2.3.12 Epic Kitchen

The Epic Kitchen [51] is a dataset of egocentric videos of kitchen motions. The dataset involves 32 subjects and contains 11.5M frames. After completing the videos, each subject was asked to watch the videos and describe the actions that they took. The audio data of their descriptions were recorded and made into textual annotations in sentences as well as into verb and noun classes. Bounding boxes of objects for key frames were also provided. The dataset can be directly used to expand FOON. It is also valuable for object and action recognition tasks. However, the dataset is not readily usable for LfD which requires execution trajectories of actions. To obtain the spatial trajectories of the actions, certain 3D reconstructive procedures need to be applied.

2.3.13 Summary

Table 2.4 lays out the different modalities included in all the datasets in this category, and Fig. 2.1 shows in descending order the count of datasets for each modality. One can easily notice in Table 2.4 that [52] includes the highest number of acquisition modalities, most of which cannot be found in the other datasets. This is because the goal of [52] is to make the dataset multi-modal. In Table 2.4, we can notice that RGB vision is used in all fourteen datasets and is the sole acquisition modality of six datasets. The equipment required for recording RGB images is generally minimal and is easy to set up. Apart from evaluating certain vision-based algorithms, the recorded RGB video can also be used to verify that the data collection scene is properly set up, to spot any mistakes during the data collection process, and to segment the collected data. RGB images are matrices and carry much more information than other scalar modalities (such as acceleration) captured at a comparable
<table>
<thead>
<tr>
<th>Modalities</th>
<th>[43]</th>
<th>[52]</th>
<th>[53]</th>
<th>[46]</th>
<th>[46]+[58]</th>
<th>[62]</th>
<th>[49]</th>
<th>[47]</th>
<th>[54]</th>
<th>[59]</th>
<th>[48]</th>
<th>[55]</th>
<th>[51]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB video (image sequences)</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>depth image sequence</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>audio</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>RFID</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>motion capture</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D acceleration on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D rotational velocity on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>2D acceleration on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>body heat</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D magnetic field on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>skin temperature</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>skin conductance</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>gaze</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D acceleration on object</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>6D object pose track</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>object models</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>air temperature</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>light</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>moving camera</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td># subjects</td>
<td>20</td>
<td>&gt;39</td>
<td>4</td>
<td>14</td>
<td>10</td>
<td>12</td>
<td>30</td>
<td>30</td>
<td>25</td>
<td>7</td>
<td>n/a</td>
<td>8</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 2.4. Modalities for cooking
Figure 2.1. Count of datasets for each modality. Orange denotes the modalities of CMU-MMAC [52], and blue denotes the modalities of the other datasets. FOON is excluded because modality does not apply to FOON.
frequency. The pose of the object or the human estimated from RGB images has a lower accuracy than if it is directly measured by a motion capture system, but it usually suffices for action recognition.

Only three datasets include depth images, only two datasets provide information on 3D acceleration on object, and only one dataset provides sequences of estimated object poses. Despite the high accuracy it provides, a motion capture system is used only in [52], possibly because of its cost, the lack of portability and the effort required for the system setup. It is worth mentioning that one of the most envisioned applications of these datasets in robotics is for learning from demonstration (LfD). What is commonly done is to use movement sequences of objects as input for training, while testing is performed through physical manipulation of real objects by the robot, (e.g., see [70]). The particular class of applications motivated us to consider and evaluate which characteristics are important to create a dataset that is designed for LfD. Ideally, we would like to have readily available sequences of 3D object poses, which include positions and orientations as in [48]. Object 3D acceleration data can be also converted to 3D position as in [43], [47], and RGB-D images can be used to estimate 3D object pose as in [54]. Another important aspect to take into account in data collection for LfD is the environment, either real or lab-based. For example, among the datasets described in this section, [59], [46]+, and [55] were collected in real kitchens, and the other datasets were collected either on a table-top or in a lab kitchen. An important difference between a real and a lab kitchen lies in the amount of clutter in the background: real kitchens generally have more clutter, which increases the difficulty of object recognition, which may lower the accuracy of object recognition. Since object pose estimations are fed into LfD as input, a possibly lower accuracy of object recognition is undesirable.

Activity annotations can be useful for various purposes. For example, if the annotations are short sentences describing a video, natural language processing can be combined with vision to provide higher accuracy on action/object recognition, or generate more annotations
Since [49] supercedes [58] and [62], we only include [49] in the table.

Annotations in the form of words can be used to represent motion activity classes. We identified the annotated activities that are shared by at least 4 cooking datasets (> 1/4 of all cooking datasets), and list those datasets in Table 2.5.

### 2.4 Datasets of Activities of Daily Living (ADL)

In this section, we present ten datasets of activities of daily living (ADL), one dataset of grasps acquired using a camera, and two datasets of robot motion. The interest in studying ADLs is motivated by the extensive variety of the objects that human hands interact with daily, and the variety of the environments where these interactions take place. Compared with Section 2.3, this section introduces additional modalities such as 3D kinematics of objects, force and torque on objects and on joints of robotic arms, sequences of estimated human skeleton etc. Apart from action recognition, the application fields of the datasets include hand pose recognition for Human Machine Interaction, grasp analysis, and deep
learning, among others. Following the format in Section 2.3, we first review each dataset individually, and then we discuss the use of motion capture and we provide more details on dataset suitability for LfD.

2.4.1 TUM Kitchen

The TUM Kitchen dataset [57] contains multi-modal data of set-a-table activities. The modalities include RGB and raw Bayer pattern videos, motion capture, RFID, and reed sensor. Four subjects each transported certain objects from the cupboard, the counter, and the drawer, to a table, and then laid them out in a specified way. The subjects transported the objects one by one as a robot would do, and also several objects at a time as naturally done by a human. The dataset also includes repetitive activities of picking up and putting down objects. The annotations cover the entire duration of the set-a-table activity which starts with Reaching through ReleaseGraspOfSomething. The actions of the left hand, the right hand, and the trunk were annotated respectively.

Similarly to CMU-MMAC [52], the dataset identifies the objects involved during motion execution, and the availability of motion capture makes it a good candidate for 3D analysis on pick-and-place motion.

2.4.2 Rochester ADL

The Rochester ADL dataset [69] contains RGB videos of five subjects performing certain ADL and Instrumented ADL (IADL) activities which can be summarized as: using phone, writing, drinking and eating, and preparing food. Each video records one activity. Similar to the MPII datasets [58]-[49] and the Brown breakfast dataset [55], the Rochester ADL dataset would rely on human and object recognition to be useful for 2D fine motion analysis.
2.4.3 OPPORTUNITY

The OPPORTUNITY dataset [44] contains multi-modal data of five morning ADL runs and one Drill run for each of four subjects. Motion sensors were densely deployed on the human body, on the objects, and in the environment. The modalities on the human body include IMUs, 3D accelerometers, and 3D localizers. The modalities on the objects include 3D accelerometers and 2D rotational velocity sensors. The annotations consists of five “tracks”: locomotion, high-level activities, mid-level gestures, low-level actions, and objects for the left and the right hand, respectively.

The dataset distinguishes itself from others that we include by using accelerometers and rotational velocity sensors on both the hand and the objects. Since object manipulation analysis focuses on the interaction between hand and objects, data that include the motion of both the hand and the objects are desired. The dataset is comparable with 50 Salad [47], CMU-MMAC [52], and TUM Kitchen [57] in modality availability, although the last three target cooking scenarios. For the objects, the dataset includes 2D rotational velocity, which is unavailable in 50 Salad. For the human body, the dataset lacks motion capture, which is available in CMU-MMAC and TUM Kitchen, but alternatively provides 3D acceleration and 3D rotational velocity.

2.4.4 Cornell CAD-60 and CAD-120

The CAD-60 [45] and the CAD-120 [63] are both RGB-D video datasets. CAD-60 includes video sequences of four subjects performing twelve ADLs in five different indoor environments. Each sequence corresponds to one instance of a certain activity. The CAD-120 dataset recorded four subjects each performing ten high-level activities. Each subject performed every high-level activity multiple times with different objects. The annotations include ten low-level activities, and twelve object affordances.
CAD-60 and CAD-120 feature skeleton data, which include tracks of 3D position of all fifteen joints plus 3D orientation of eleven joints. The skeleton data in these datasets were generated using the NITE library that complements the PrimeSense sensors and were therefore estimated data. By comparison, the skeleton data collected using a motion capture system are actual physical measurements and therefore can be regarded as ground truth. Thus, the accuracy of the skeleton data in CAD-60 & 120 is lower than the accuracy of those collected with a motion capture system. Nevertheless, the skeleton data are directly usable for 3D fine motion analysis, a characteristic we consider as an advantage of these datasets.

2.4.5 First Person ADL

The First Person ADL dataset by Pirsiavash [66] contains RGB videos captured using a GoPro camera. It recorded twenty subjects performing eighteen ADLs. Forty-two objects were annotated by annotators with bounding boxes, tracks, and the status as to whether the object is being interacted with. Similar to Gaze(+) [46], with first person images, the working area of the hands is emphasized. However, since the dataset includes a single modality, using it for analysis on 2D fine motion would rely on object tracking.

2.4.6 Wrist-Worn Accelerometer

The wrist-worn accelerometer dataset [60] contains accelerometer data of sixteen subjects performing a total of fourteen ADLs. The accelerometers were attached to the right wrists of the subjects and the data were recorded at the subjects’ home. The dataset contains 979 trials. For fine motion analysis, wrist acceleration may be less ideal than hand acceleration, but it remains a readily usable modality.
2.4.7 UCI-EGO

The UCI-EGO or general-HANDS dataset [67] includes four sequences of object manipulation activities. Each sequence includes 1,000 RGB-D frames captured using an egocentric camera. Various objects were involved and manipulated, but since the dataset focuses on hand detection and pose estimation, the manipulation tasks performed with each object are relatively short. As with other vision oriented datasets, the use of UCI-EGO dataset for object manipulation analysis relies on object tracking.

2.4.8 Yale Human Grasping

The Yale human grasping dataset [64] contains 27.7 hours of RGB wide-angle videos of profession-related manipulation motion. Two machinists and two housekeepers participated. The dataset is intended for grasping analysis. The annotations were done on two levels. On the first level, the grasp type was annotated along with the corresponding task name and object name. The second level provided the properties of the object and the task. A total of 18,210 grasp instances have been annotated. The dataset includes prolonged videos of manipulation motion of machining and housekeeping alone, two categories that are not to be found in other datasets that we include.

2.4.9 Google Push and Grasping

To facilitate deep learning in robotics, Google Brain publicly shares two datasets of movements of robotic arms: Push [61] and Grasping [56].

The Push dataset contains about 59,000 sequences of multimodal data of robotic arms pushing objects. A bin which contained different objects was placed in front of a 7 DOF robotic arm, and the arm repeatedly pushed the objects in one out of two ways: either pushing randomly, or starting randomly from somewhere on the border of the bin and sweeping
towards the middle. A camera was mounted behind the arm facing the bin. The bin contained ten to twenty objects at a time, and the objects were swapped out for new ones after roughly 4,000 pushes. Ten robotic arms were used. The data include RGB images, recorded gripper pose ($x$, $y$, $z$, yaw, pitch), commanded gripper pose, robot joint position and external torques. The dataset provides two test sets each including 1,500 sequences. One test set contains two different subsets of objects from the training set, and the other test set includes two sets of objects absent from the training set.

The Grasping dataset is collected using a similar setup to that of Push. The dataset contains about 650,000 sequences of multimodal data of robotic arms grasping objects. The modalities include RGB-D images, recorded and commanded gripper pose (position in $x$, $y$, $z$ and orientation in quaternions), joint positions – velocities – external torques – and commanded torques.

Using Push or Grasping which involve robots only, one aims at learning to finish a task rather than learning to finish a task like a human. The absence of the retargeting problem [72] is an inherent convenience if the learned motion is to be executed by the same robot.

2.4.10 Manipulation Kinodynamics

The manipulation kinodynamics dataset [50] includes 3.2 hours of kinematics and dynamics information of objects grasped and manipulated by humans using five fingers. More specifically, the data of the object includes mass, inertia, linear and angular acceleration, angular velocity, and orientation. For each of the five fingers, the collected data include friction, force, contact point position, and the axes of a right-handed local coordinate frame ($x$, $y$, $z$), where axes $x$ and $y$ define the contact surface, and axis $z$ points towards the object. The dataset does not include images or videos. The objects are custom made and can vary in mass distribution, friction, and shape. The performed motions vary in speed, direction, and task (e.g., emulating pouring). In total 193 different combinations were recorded.
[50] provides a full suite of kinematics and dynamics data. It was created for investigating the mapping relationship between the kinematics features (velocity, acceleration, etc.) of a manipulated object and the underlying manipulating force, which is something similar to a Newtonian physical law. Both the cause of manipulation (the force) and the corresponding result (the kinematics) were measured and both were of the object, and no extra processing or estimation is needed. Therefore, we consider the dataset as invaluable for manipulation research, although including RGB-D images would have made the dataset more approachable to the computer vision community.

2.4.11 RPAL Tool Manipulation

The dataset [65] features tool manipulation by humans and is still in the process of being created. The dataset contains multimodal sequential data of subjects using different tools. The tool consists of four components from front to back: a swappable tooltip, a 6 DOF force-and-torque (FT) sensor, a universal handle, and a 6 DOF position-and-orientation (PO, x, y, z, yaw, pitch roll) tracker. When possible, another PO tracker is mounted on the object which interacts with the tool. Modalities recorded besides FT and PO data are top view RGB videos and depth sequences of the scene, and finger flexure. Currently available data are hosted at http://rpal.cse.usf.edu/imd/. Since FT and PO data are of the tool, they can be used directly for manipulation learning, without the need of feature extraction which is necessary for images.

2.4.12 Summary

Similar to what we do for the cooking datasets, here we lay out the modalities in all the datasets in Table 2.6, and we show in Fig. 2.2 the count of datasets for each modality in descending order.
Table 2.6. Modalities ADL

<table>
<thead>
<tr>
<th>Modalities</th>
<th>[57]</th>
<th>[69]</th>
<th>[44]</th>
<th>[45]</th>
<th>[63]</th>
<th>[66]</th>
<th>[60]</th>
<th>[64]</th>
<th>[61]</th>
<th>[56]</th>
<th>[65]</th>
<th>[50]</th>
<th>[67]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB video (image sequences)</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>depth image sequence</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>RFID</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>motion capture</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>Bayer pattern video</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D acceleration on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D rotational velocity on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D orientation on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D location on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D magnetic field on human</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>finger flexure</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>skeleton</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>joint angles</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D acceleration on object</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D rotational velocity and acceleration on object</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>2D rotational velocity on object</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>3D acceleration on furniture</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>reed switch on furniture</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>force on fingertip</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>in-tool force</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>in-tool torque</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>gripper pose</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>commanded gripper pose</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>robot joint position</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>robot joint velocity</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>robot joint commanded torque</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>robot joint external torque</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>moving camera</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td># subjects</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>20</td>
<td>16</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>
We can see from the figure that, RGB vision is the most commonly used modality and is provided in twelve datasets excluding only [44], [60] and [50]. In fact, [44] did collect RGB videos but did not publish them. Motion capture data are very accurate and can be found in [57] which uses a markerless system, and in [65] which uses both an optical marker-based system and an electromagnetic system. When an object is being manipulated, its orientation may change significantly (for example, when a spatula is used to flip a slice of bread), challenging the reliability of an optical marker-based system. Moreover, objects vary in shape and can be small, which limits the maximum amount of markers that can be used. Also, during the execution of a task, a manipulated object generally has certain contact with another object or certain material (such as water), which makes the contact surface unavailable for mounting markers and the available mounting surface even smaller. The above reasons drove [65] to switch from an optical marker-based system to an electromagnetic (EM) alternative. The EM motion capture system consists of at least one source which defines the world frame and acts as the origin, and one tracker which senses its position and orientation with respect to the source. The source and tracker are both connected to a processing station with cables. Since the EM system uses cables, it does not require an unconstrained line-of-sight as in an optical marker-based system, and therefore a significant object pose change cause occlusions and does not affect measurement accuracy. However, continuously rotating motion such as using a screwdriver has a possibility of finally putting stress on the cable, and therefore requires extra attention.

Unique to this section, [50], [65], [61], and [56] introduced the provision of force and torque. The [50] and [65] data belong to the object and the [61] and [56] data belong to the joints of the robotic arms. Including force and torque enables modeling feedback, which makes the learning of object manipulation more physically realistic and helps with performing a learned task with a real object.
The dataset described in [51], is intended for learning the relationship between kinematic features and manipulating force during a manipulation motion in general, and not for a particular manipulation task. In simpler words, the dataset focuses on manipulation rather than task. As a consequence, the dataset falls short of the requirement for Learning from Demonstration [70], which focuses on manipulation tasks. Since [50] used 3D printed objects, modifying [50] to make it suitable for LfD would require to change the current 3D object models to enable interaction with other objects while keeping the kinodynamics sensors from interfering with the manipulation tasks, which may be non-trivial. In comparison, [65] focuses on recording data of tasks and is suitable for LfD, although it provides less fine-grained dynamics data than [50].

As for the cooking datasets, we identified the annotated activities that are shared by multiple ADL datasets, and we list those datasets in Table 2.7. We combine similar annotations and specify each in the cells. For example, on the first row of Table 2.7, the annotated activity is summarized as “use phone”, whereas [69] specifically uses “answer phone” and “dial on a phone”, and [45] specifically uses “talk on the phone”.

2.5 Discussion

Research in object manipulation might find 3D object poses very useful. Explicit or readily usable recordings of object poses are available in [65]. Poses of the robot end effector are provided in [61] and [56]. [48] provides estimated object pose trajectories. Object poses may be computed using acceleration and rotational velocity, and object motions that are simpler than poses can be obtained if a sensor actively takes samples and is attached to an object. Datasets with such setup include

1. [43] and [47] where objects were equipped with accelerometers,
Figure 2.2. Count of datasets for each modality
Table 2.7. Shared annotated ADLs.

<table>
<thead>
<tr>
<th>Activities</th>
<th>[57]</th>
<th>[69]</th>
<th>[44]</th>
<th>[45]</th>
<th>[63]</th>
<th>[66]</th>
<th>[60]</th>
</tr>
</thead>
<tbody>
<tr>
<td>use phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>write on whiteboard</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drink</td>
<td>■</td>
<td></td>
<td>sip</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eat</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chop/cut</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reach</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>release</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>comb hair</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brush teeth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use computer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stir</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>open</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We only consider low-level annotations for [44].
2. [44] where objects were equipped with accelerometers and rotational velocity sensors. Furniture and appliances were equipped with reed switches and accelerometers,

3. [57] where doors were equipped with reed switches.

The shared activities demonstrate a consensus among different authors on what activities should be performed and annotated. For example, certain grasping taxonomies are often adopted and such directions can be helpful for one who tries to create a new dataset. However, not being a commonly shared activity does not necessarily mean an activity is not important. Therefore, we also provide the complete list of annotated activities at http://rpal.cse.usf.edu/motiondatasetreview/index.htm, for cooking and ADL, respectively. The shared activities can also help us utilize more than one dataset. If one wants to study a certain shared activity, one could use several datasets that include this activity in order to access more modalities and have higher variability. Objects that are involved in an activity may also be helpful for activity analysis. For all datasets except [54], objects are identifiable in the annotated activities through

1. being separately annotated: [62], [49], [59], [63], [66], [64], [53], [68], [51],

2. being part of the annotation phrases: [43], [52], [46], [46]+, [58], [47], [48], [55], [57], [69], [45], [60], [53], [51],

3. being equipped with sensors

   (a) accelerometers: [43], [47], [44],

   (b) rotational velocity sensors: [44],

   (c) reed switches: [57], [44],

   (d) RFID: [52], [57].
Temporal segmentation of annotated activities is also important for activity analysis. [64] does not include temporal segmentation because it focuses on grasp instances. All other datasets include temporal segmentation, in the following forms

1. video subtitle: [43], [48],
2. explicit video time: [46]+, [66],
3. frame number: [52], [46], [58], [62], [49], [54], [59], [55], [57], [63], [53], [51],
4. timestamp: [47], [44], [51],
5. implicit: [69], [45], [60].

We are aware of the existence of other related datasets, however, to keep this work focused we do not include them. Examples of the excluded datasets are

1. [73], and [74], [75], which are datasets that do not include object manipulation motions, or if they do, the object manipulation motions are sparse.
2. [76], [77], and [78], which are datasets of objects that are typically involved in manipulation, rather than datasets of motion.

Most datasets are intended for action recognition. However, researchers who work on learning from demonstration (LfD) [70] intend to reproduce human actions rather than recognize them. Thus, we suggest in addition to choosing from the modalities we have reviewed, a more ideal dataset for LfD should also aim to provide readily usable data that are more closely related to dynamic and kinematic motion execution. Examples of suggested modalities include trajectories of object poses, joint poses of human upper body, hand posture, torque, force between hand and object, etc.

Finally, an important specification for creating useful datasets that can be used in robotics applications is to facilitate benchmarking. One interesting example is provided in [78], where
the objects used for manipulation were chosen to cover different aspects of the manipulation problem and object characteristics, and RGB-D object scans, physical properties and geometric models are also provided together with protocol examples and physical object delivery.

2.6 Conclusions

In this chapter, we reviewed 28 datasets on object manipulation. We reported the characteristics and modalities of each dataset individually, we gave our view on the relation between each dataset and object manipulation, and we compared and summarized all of them together.

The datasets were created to serve their own purposes and many of them are unique. Therefore different modalities were used. The modalities range from popular video recording to rarely used air temperature and light. Many datasets were collected with numerous subjects, while some were collected with only one subject. The survey provides a “map” for researchers in choosing the right existing dataset(s) for their own research purposes. If the right datasets are not found, the researchers may decide on creating new datasets that will supplement the existing datasets. For example, we have not come across a dataset that includes interactive force or torque.

Observing the diversity of the datasets, we understand that trying to get a unique standard for the different types of datasets is clearly a daunting and challenging task. However, moving towards a common standardization that defines common data formats for common working areas as well as acquisition protocols would enable efficient data re-usage and sharing, fostering collaborations, and creating large datasets that allow big-data-driven approaches such as deep learning. It has been discussed recently in many conferences and workshops of the robotics community as one of several important initiatives.
This survey does not include datasets that, although are introduced in publications, are not openly available. Many of them were presented in the Workshop on Grasping and Manipulation Datasets that was organized under the International Conference on Robotics and Automation (ICRA) in May 2016. The workshop’s report [79] provides a survey of those works and datasets.
CHAPTER 3

DATA COLLECTION

In this chapter, we present our own dataset of interactive manipulation. The data collection originated from our need for 3-dimensional motion data of objects involved in fine manipulation motion as well as data that represent the interaction, which we need to learn the dynamic system of the manipulation. The related datasets which have been reviewed in Chapter 2 are less than ideal in that 1) calculating the position trajectory using the acceleration may be inaccurate due to accumulated error, 2) the motions of objects are not always emphasized or even available, and 3) all the activities are not fine manipulations that serve to finish tasks. Having identified those deficiencies, we collected a dataset ourselves that includes 3-dimensional “position and orientation, force and torque” data of tools/objects being manipulated to fulfill certain tasks.

The dataset focuses on position, orientation, force, and torque of objects manipulated in daily tasks. The dataset includes 1,593 trials of 32 types of daily motions and 1,596 trials of pouring alone, as well as helper code.

3.1 Overview

We recorded daily performed fine motion in which an object was manipulated to interact with another object. We refer to the person who executes the motion as subject, the manipulated object as tool, and the interactive object as object. We focus on recording the motion of the tool. In some cases, we also record the motion of the object.
The dataset consists of two parts. The first part contains 1,593 trials that cover 32 types of motions. We choose fine motions that people commonly perform in daily life which involve interaction with a variety of objects. We reviewed existing motion-related datasets [80, 81, 79] to help us decide which motions to collect.

The second part contains the pouring motion alone. We collected it to help with motion generalization to different environments. We chose pouring because 1) pouring is found to be the second most frequently executed motion in cooking, right after pick-and-place [68] and 2) we can vary the environment setup of the pouring motion easily by switching different materials, cups, and containers. The pouring data contains 1,596 trials of pouring 3 materials from 6 cups into 10 containers.

We collected the two parts of the data using the same system. We specifically describe the pouring data in Sec. 3.9.

The dataset aims to provide position and orientation (PO) and force and torque (FT), nevertheless, it also provides RGB and depth vision with a smaller coverage. Table 3.1 shows the number of trials and the counts of each modality for each motion. The minimum number of trials for each motion is 25. Table 3.2 shows the coverage of each modality throughout the entire data, where the coverage has a range of (0, 1], and a coverage of 1 means the modality is available for every trial. The lower coverage of the vision modality is due to filming permission restrictions.

3.2 Hardware

On a desk surface, we use blue masking take to enclose a rectangular area which we refer to as the working area, and within which we perform all the motions. We make a PrimeSense RGB+depth camera aim at the working area from above.

We started collecting PO data using the OptiTrack motion capture (mocap) system and soon afterwards replaced OptiTrack with the Patriot mocap system. Both systems provide
Table 3.1. The count for each modality for each motion. Each motion is coded $mx$, where $x$ is an integer.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Total</th>
<th>PO</th>
<th>FT</th>
<th>vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2</td>
<td>stir with spatula</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m3</td>
<td>sprinkle, shake pepper</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>m4</td>
<td>spread/oil</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m6</td>
<td>vertical cut</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m7</td>
<td>use spoon to pick up</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>35</td>
</tr>
<tr>
<td>m8</td>
<td>pizza wheel</td>
<td>25</td>
<td>25</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>m10</td>
<td>use black brush</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m11</td>
<td>spear object using fork</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>m12</td>
<td>stir water using spoon</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m13</td>
<td>fasten screw with screwdriver</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>m14</td>
<td>loosen screw with screwdriver</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>m15</td>
<td>unlock lock with key</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>75</td>
</tr>
<tr>
<td>m16</td>
<td>fasten nut with wrench</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td>m17</td>
<td>use paint brush to dip and spread</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m18</td>
<td>use hammer to hammer in nail</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m19</td>
<td>brush teeth</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>m20</td>
<td>use file to file wooden thing</td>
<td>125</td>
<td>125</td>
<td>125</td>
<td>25</td>
</tr>
<tr>
<td>m21</td>
<td>comb hair</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m22</td>
<td>scrape substrate from surface</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m23</td>
<td>peel cucumber/potato</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>m24</td>
<td>slice cucumber</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m25</td>
<td>flip bread</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>74</td>
</tr>
<tr>
<td>m26</td>
<td>use spoon to scoop and pour</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m27</td>
<td>shave object</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>m28</td>
<td>use roller to roll out dough</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>m30</td>
<td>loosen nut with wrench</td>
<td>46</td>
<td>46</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>m31</td>
<td>scoop and pour with measuring spoon/cup</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>m32</td>
<td>insert peg into pegboard</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>m33</td>
<td>brush powder across grey tray</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>m34</td>
<td>insert straw through to-go cup lid</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>m35</td>
<td>m34 with eyes closed</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>m36</td>
<td>m31 without pour</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>
3-dimensional PO data regardless of their differences in technology. Patriot includes a source and a sensor. The source provides the reference frame, with respect to which the PO of the sensor is calculated. We use an ATI Mini40 force and torque (FT) sensor together with the Patriot PO sensor. To attach both the FT sensor and the PO sensor to a tool, we use a cascading structure that can be represented as: (tooltip + adapter + FT sensor + universal handle + PO sensor), where “+” means “connect”. The end result is shown in Fig. 3.1. A tool in general consists of a tooltip and a handle. We disconnected the tooltip from the stock handle, inserted the tooltip into a 3D-printed adapter, and glued them together. Then we connected the adapter with the tooling side of the FT sensor using screws. We 3D-print a universal handle and connect it with the mounting side of the FT sensor using screws. At the end of the universal handle we mount the PO sensor using screws. In some cases, we track the object in addition to the tool, and to do that we put a second PO sensor on the object, as shown in Fig. 3.2.

Each tooltip is provided with a separate adapter. Since the tooltip and the adapter are glued together, a tool is equivalent to “tooltip + adapter”. Fig. 3.3 shows the tools that we have adapted.

### 3.3 Coordinate Frames

To track a tool using OptiTrack, we need to define the ground plane and define the tool as a trackable. The ground plane is set by aligning a right-angle set tool to the bottom left corner of the working area. The trackable is defined from a set of selected markers, and is assigned the same coordinate frame, with the origin being the centroid of the markers. This is shown in Fig. 3.5.
Figure 3.1. The structure that connects the tool, the FT sensor and the PO sensor

Figure 3.2. Tracking both the tool and the object with two PO sensors
Patriot contains a source that supports up to two sensors. The source provides the reference frame for the sensors as shown in Fig. 3.6. We define the base point of the tool to be the center of the tooling side of the FT sensor, as shown in Fig. 3.4. The translation from the PO sensor to the base point of the tool is $[14.3, 0, 0.7]$, in the frame of the PO sensor, unit centimeter.

The FT sensor and the PO sensor are connected through the universal handle. The groove on the universal handle is orthogonal to both the $x - y$ plane of the FT sensor and the $y - z$ plane of the PO sensor. The relationship between the local frames of the FT sensor and the PO sensor is shown in Fig. 3.7.
Figure 3.4. The tool’s base point is the center of the tooling side of the FT sensor
Figure 3.5. Coordinate frames for the ground plane and the trackable of OptiTrack

Figure 3.6. Coordinate frames of Patriot source and sensor placed on the same plane. ✗ means into the paper plane.
3.4 Calibrate FT

**Definition 1** The level pose of the universal handle is a pose in which the groove of the handle faces up, and in which the $y - z$ plane of the FT sensor or equivalently the $x - y$ plane of the PO sensor is parallel to the desk surface.

**Definition 2** An average sample is the average of 500 FT samples.

The FT sensor has non-zero readings when it is static with the tool installed on it. We calibrate the FT sensor, or make the readings zeros, before we collect any data. We hold the handle in a level pose (Definition 1), and take an average sample (Definition 2) which we set as the bias $FT_b$. We subtract the bias from each FT sample before saving the sample: $FT_t \leftarrow FT_t - FT_b$. We calibrate the FT sensor each time we switch to a new tool.
3.5 Modality Synchronization

Different modalities run at different frequencies and therefore need synchronization, which we achieve by using time stamps. We use Microsoft QueryPerformanceCounter (QPC) to query time stamps with millisecond precision.

When we start the collection system, we query the time stamp and set it as the global start time $t_0$. Then we start each modality as an independent thread, so that they run simultaneously and do not affect each other. For each sample, a modality queries the time stamp $t$ through QPC, and sets the difference between $t$ and $t_0$, i.e. the elapsed time since $t_0$ as the time stamp for that sample:

$$t ← t - t_0.$$  \hfill (3.1)

3.6 Data Format

The data are organized in a “motion → subject → trial → data files” hierarchy, as shown in Fig. 3.8, where the prefixes for motion, subject, and trial directories are m, s, and t, respectively.

RGB videos save as .avi, depth images save as .png, and the rest data files save as .csv. Both RGB and depth have a resolution of 640×480, and are collected at 30Hz.

The .csv files excluding those of OptiTrack follow the same structure as shown in Fig. 3.9. The first row contains the global start time and is the same in all the .csv files that belong to the same trial. Starting with the second row, each row is a data sample, of which the first column is the time stamp (Eq. (3.1)), and the rest of the columns are data specific to a certain modality. The OptiTrack .csv file differs in that it contains a single-column row between the start-time row and the data rows, which contains the number of defined
trackables (1 or 2). In the following we explain the data part of a row for each different csv file.

The FT sensor outputs 6 columns: \((f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)\), where \(f_x\) and \(\tau_x\) are the force and torque in the \(+x\) direction, respectively. FT can be sampled at a very high frequency but we set it to be 1 kHz. The force has units pound (lbf) and the torque has units pound-foot (lbf-ft).

For the RGB videos and depth image sequences, we provide the time stamp for each frame in a csv file. The data part has one column, which is the frame index.

The PO data contain the tool, and may also contain the object. With two PO capture systems, and with or without the object, four different formats exist for the PO data, which are listed in Fig. 3.10. Patriot expresses the orientation using yaw-pitch-roll (w-p-r) which is depicted in Fig. 3.11, and OptiTrack uses unit quaternion \((q_x, q_y, q_z, q_w)\). If we only use one trackable but have defined two in OptiTrack, we disable the inactive one by setting all 7 columns for that trackable to be -1, i.e., the 8 columns for the inactive trackable would be \((1, -1, -1, -1, -1, -1, -1)\).
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>534227475973</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>-0.0967903</td>
<td>-0.02329654</td>
<td>0.0024319</td>
<td>0.0387435</td>
<td>0.00469243</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>96</td>
<td>0.0401328</td>
<td>-0.0974332</td>
<td>-0.021754</td>
<td>0.006532</td>
<td>0.040663</td>
<td>0.00372569</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>91</td>
<td>0.040193</td>
<td>-0.0973107</td>
<td>-0.023307</td>
<td>0.0065814</td>
<td>0.0385548</td>
<td>0.00517069</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>92</td>
<td>0.0385129</td>
<td>-0.0961003</td>
<td>-0.021756</td>
<td>0.0791507</td>
<td>0.0354935</td>
<td>0.00422668</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>94</td>
<td>0.0402371</td>
<td>-0.0961254</td>
<td>-0.0155676</td>
<td>0.0798316</td>
<td>0.0350909</td>
<td>0.00285439</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>96</td>
<td>0.0401114</td>
<td>-0.0973935</td>
<td>-0.0186508</td>
<td>0.0814023</td>
<td>0.0400738</td>
<td>0.00370263</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>97</td>
<td>0.0419745</td>
<td>-0.0973811</td>
<td>-0.0202471</td>
<td>0.0787946</td>
<td>0.0391888</td>
<td>0.00375471</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>0.0420114</td>
<td>-0.0961895</td>
<td>-0.0155945</td>
<td>0.0790353</td>
<td>0.0362022</td>
<td>0.00286058</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>101</td>
<td>0.042859</td>
<td>-0.0947612</td>
<td>-0.0156433</td>
<td>0.0767126</td>
<td>0.0404779</td>
<td>0.00431475</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>102</td>
<td>0.0438027</td>
<td>-0.0941987</td>
<td>-0.0171841</td>
<td>0.076847</td>
<td>0.0382818</td>
<td>0.00383793</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>103</td>
<td>0.0428785</td>
<td>-0.0926901</td>
<td>-0.0187007</td>
<td>0.0771571</td>
<td>0.0394843</td>
<td>0.00238794</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>104</td>
<td>0.0447166</td>
<td>-0.0947541</td>
<td>-0.0202906</td>
<td>0.0758502</td>
<td>0.0389333</td>
<td>0.00283306</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>105</td>
<td>0.0446055</td>
<td>-0.0940316</td>
<td>-0.0125835</td>
<td>0.0749259</td>
<td>0.0411111</td>
<td>0.00269877</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>106</td>
<td>0.0455434</td>
<td>-0.0933383</td>
<td>-0.023406</td>
<td>0.0752631</td>
<td>0.0426463</td>
<td>0.00270353</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>107</td>
<td>0.0465295</td>
<td>-0.0915837</td>
<td>-0.0187333</td>
<td>0.0738329</td>
<td>0.0396612</td>
<td>0.00269636</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>108</td>
<td>0.0464398</td>
<td>-0.0916779</td>
<td>-0.0156732</td>
<td>0.0747225</td>
<td>0.0411832</td>
<td>0.00292103</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>109</td>
<td>0.0455238</td>
<td>-0.090121</td>
<td>-0.018731</td>
<td>0.0759328</td>
<td>0.0418498</td>
<td>0.00290459</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>110</td>
<td>0.0464424</td>
<td>-0.0907289</td>
<td>-0.0172558</td>
<td>0.072309</td>
<td>0.043665</td>
<td>0.00338405</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>111</td>
<td>0.0509114</td>
<td>-0.093437</td>
<td>-0.0198491</td>
<td>0.0717216</td>
<td>0.043721</td>
<td>0.00296243</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>112</td>
<td>0.0481834</td>
<td>-0.0926369</td>
<td>-0.0203377</td>
<td>0.0744771</td>
<td>0.0437614</td>
<td>0.00383071</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>113</td>
<td>0.050044</td>
<td>-0.0929157</td>
<td>-0.0219222</td>
<td>0.0727203</td>
<td>0.0439312</td>
<td>0.00381968</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>114</td>
<td>0.050009</td>
<td>-0.090752</td>
<td>-0.0203561</td>
<td>0.0731716</td>
<td>0.0433789</td>
<td>0.00338468</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>115</td>
<td>0.0508601</td>
<td>-0.0925335</td>
<td>-0.0203883</td>
<td>0.0736765</td>
<td>0.0486728</td>
<td>0.00320939</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>116</td>
<td>0.0508684</td>
<td>-0.0903999</td>
<td>-0.0204099</td>
<td>0.0706129</td>
<td>0.0476692</td>
<td>0.00434006</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>117</td>
<td>0.049619</td>
<td>-0.0882958</td>
<td>-0.0142029</td>
<td>0.0708334</td>
<td>0.0470715</td>
<td>0.00343806</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>118</td>
<td>0.0508729</td>
<td>-0.093245</td>
<td>-0.0172912</td>
<td>0.0717008</td>
<td>0.0471731</td>
<td>0.00342319</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.9. The structure of a non-OptiTrack csv data file.
Patriot:

One sensor:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
<th>p</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>w</td>
<td>p</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
<th>p</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>w</td>
<td>p</td>
</tr>
</tbody>
</table>

Two sensors:

| 1 | x | y | z | w | p | r | 2 | x | y | z | w | p | r |

OptiTrack:

One trackable:

| 1 | x | y | z | qx | qy | qz | qw |

Two trackables:

| 1 | x | y | z | qx | qy | qz | qw | 2 | x | y | z | qx | qy | qz | qw |

Figure 3.10. Formats of the columns for PO for one and two sensors

Patriot samples at 60 Hz, its $x - y - z$ has unit centimeter and its yaw-pitch-roll has units degree. OptiTrack samples at 100 Hz, and its $x - y - z$ has unit meter.

3.7 Using the Data

We provide MATLAB code that visualizes the PO data for OptiTrack as well as Patriot, as shown in Fig. 3.12. The visualizer displays the trail of the base point of the tool (Fig. 3.4) and the object if applicable as the motion is played as an animation in 3D. The user can also manually slide through the motion forward or backward and go to a particular frame.

The FT and PO csv files have multiple formats, and we provide Python code that extracts FT and PO data from each trial given the path of the root folder. Although we have explained the format of the csv files of the FT and PO data in Sec. 3.6, we highly recommend using our code to get the FT and PO data to avoid error.
Figure 3.11. Axes and yaw-pitch-roll correspondence for the PO sensor

Figure 3.12. Visualizing the PO data
Each modality is sampled at a unique frequency, and using multiple modalities requires using the time stamps. One or more modalities need upsampling or downsampling.

3.8 Known Issue

The PO data recorded using OptiTrack contains occasional flickering and stagnant frames. This is caused by the dependency of OptiTrack on the line of sight. This issue is not present in the data collected with Patriot.

3.9 The Pouring Data

We want to learn to perform a type of motion from its PO and FT data, and generalize it, i.e., execute it in a different environment. Thus, we need data that shows how the motion varies in multiple different environments. We realize that since pouring is the second most frequently executed motion in cooking [68], it is worth learning. Also, collecting pouring data that contain a different environment setup is easy thanks to the convenience of switching material, cups, and containers. Therefore, we collected the pouring data.

The pouring data includes FT, Patriot PO, and RGB videos (no depth). We collected the data using the same system as described above. In the following, we explain what has not been covered and what differs from above.

The physical entities involved in a pouring motion include the material to be poured, the container from which the material is poured which we refer to as cup, and the container to which the material is poured which we refer to as container. The pouring data contain 1,596 trials of pouring water, ice, and beans from six different cups to ten different containers. Cups are considered tools and are installed on the FT sensor through 3D-printed adapters.

A second PO sensor is taped on the outer surface of the container just below the mouth.
We collect the FT data differently from above. When the cup is empty, we hold the handle in a level pose (Definition 1), and take an average sample (Definition 2) which we call “FT_empty”. Then we fill the cup with the material to an amount we desire, hold the handle in a level pose, and take an average sample which we call “FT_init”. Then we pour, during which we take however many samples (not average samples) which we call “FT”. After we finish pouring, we hold the handle in a level pose, and take an average sample which we call “FT_final”. In summary, we save four kinds of FT data files – three contain an average sample each: FT_empty, the FT_init, FT_final, and one contains regular samples: FT. We do not consider bias.

The organization of the data is shown in Fig. 3.13.

The pouring data can be used to learn how to pour in response to the sensed force of the cup. The force is a non-linear function of the physical properties of the cup and the material, the speed of pouring, the current pouring angle, the amount of remaining material in the cup, as well as other possibly related physical quantities. [?] shows an example of modeling
such function using a recurrent neural network and generalizing the pouring skills to unseen cups and containers.

3.10 Conclusions and Future Work

In this chapter, we presented our own dataset of daily interactive manipulations. The dataset includes 32 types of motions, and provides position and orientation, and force and torque for every motion trial. In addition, to support motion generalization to different environments, we chose the pouring motion and collected corresponding data. The dataset will be expanded with more types of motions and more modalities in the future.
CHAPTER 4

TRAJECTORY GENERATION USING MOTION HARMONICS

4.1 Note to Reader

Portions of this chapter have been previously published in "Generating manipulation trajectory using motion harmonics," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 4949-4954, and have been reproduced with permission from IEEE.

4.2 Overview

In this chapter, we present our first attempt at learning to generate trajectories using human demonstrations. We consider the shape of the trajectories and analyze how it varies from one trial to another. We consider a trajectory as the linear combination of a set of eigen-trajectories, where the effect of each eigen-trajectory on the resulting trajectory is represented by its weight. We build a new trajectory by determining the weight for each eigen-trajectory. We intend to learn from this experience, understand the problem more deeply, and gain some insights on which direction to take to achieve a better solution.

4.3 Approach

Fig. 4.1 shows the procedures of the proposed motion generation approach. First, human motion trajectories in the world space are collected. Since the trajectories may not lie completely within the workspace of the robot, we adapt them and meanwhile perform inverse
Figure 4.1. The proposed approach consists of three steps. First, the data go through temporal and spatial preprocessing and switch to a different space, and motion harmonics are extracted from the data. Then the approach takes task constraints, and uses the motion harmonics to generate new motion through optimization.

Kinematics to convert them to the robot’s configuration space or joint space. Instead of discrete vectors, we use continuous functions to represent the trajectories. Based on the data’s functional representation, a series of continues motion harmonics are obtained through functional analysis. To generate a new trajectory for a task with novel constraints that could be related to a new environment or a new goal, an optimal composition of the motion harmonics is computed to find a trajectory with the goal of resembling the demonstrated trajectories and minimizing the distance between the trajectory and the given constraints. The generated trajectory is in the robot’s joint space and can be directly used as joint control inputs. The algorithm is presented to work in the joint space, but it also applies to the world space.

4.3.1 Data Preprocessing

We preprocess the data before extracting the motion harmonics. First, we align all trajectories in time using batch DTW [82]. If the robot’s workspace is smaller than demonstrated workspace, the trajectories are then adapted to fit in the robot’s workspace by iterative
downscaling and translation. The adaptation returns the final scale and translation of the
entire set of trajectories, and the inverse kinematics of each single trajectory.

4.3.2 Representing Trajectories with Functions

In the real world, physical quantities change continuously in time. In contrast, most data
people collect are discrete in time due to the limited sampling rate of measuring devices.
Coming from physical quantities, data are intrinsically continuous and therefore should be
treated accordingly. Let $\mathbf{x} = [x_1, x_2, ..., x_T]^T \in \mathbb{R}^{T \times 1}$ denote some human motion data
collected uniformly in time where $T$ is the total number of samples. Using $\mathbf{x}$ directly as
a $T$-vector in analysis fails to preserve the very essential characteristics of human motion
which is continuous. To remedy that, we consider $\mathbf{x}$ as being driven by a function $x(t)$, and
$\mathbf{x}$ as discrete samples of $x(t)$ collected along axis $t$ with measurement noise.

A general function that does not have an explicit expression can be expressed using a
basis expansion. Denote by $\{\phi_k(t)\}, k = 1, ..., K$, a general functional basis system, and
$\{c_k\}$, the corresponding coefficients. A general function $f(t)$ can be expressed as

$$
 f(t) = \sum_{k=1}^{K} c_k \phi_k(t). \tag{4.1}
$$

To represent $\mathbf{x}$ using a basis expansion as in Eq. (4.1), both $\{\phi_k(t)\}$ and $\{c_k\}$ must be
determined.

$\{\phi_k(t)\}$ should be determined according to the characteristics of data $\mathbf{x}$. For open-ended
non-periodic data, the spline basis offers modeling flexibility through the choice of order and
design of breakpoints. For data that exhibits periodic patterns, the Fourier basis is a natural
candidate.

With a chosen basis system $\{\phi_k(t)\}$, $c_k$’s are computed by fitting the basis $\{\phi_k(t)\}$ to
the data $\mathbf{x}$. Since $\phi_k(t)$ is defined within time interval $[1, T]$, the basis $\phi_k(t)$ can be sampled
as $\phi_k = [\phi_{k,1}, \phi_{k,2}, ..., \phi_{k,T}]^T \in \mathbb{R}^{T \times 1}$ within $[1, T]$. We define $\Phi = [\phi_1, \phi_2, ..., \phi_K]$ and $c = [c_1, c_2, ..., c_K]^T$, then the data $\hat{x} \in \mathbb{R}^{T \times 1}$ can be proximated by

$$\hat{x} = \Phi \hat{c},$$

(4.2)

where

$$\hat{c} = \arg \min_c (x - \Phi c)^T W (x - \Phi c),$$

(4.3)

and $W \in \mathbb{R}^{T \times T}$ is a symmetric weighting matrix that accounts for non-uniform variance along time range $[1, T]$. $\hat{c}$ specifies the approximated driving function $\hat{x}(t) = \sum_k \hat{c}_k \phi_k(t)$.

4.3.3 Functional Analysis

After adaptation, the demonstrated motion trajectories are converted into the joint space. Consider a set of joint space trajectories represented by functions: $q_i(t)$, where $i = 1, \ldots, N$, and $t \in [1, T]$. $N$ is the number of trajectories.

The motion harmonics are the eigenfunctions of $\{q_i(t)\}$. We explain the acquisition of motion harmonics using the simplest case where $q_i(t)$ is one dimensional. To get the motion harmonics, we first calculate the mean motion

$$q_0(t) = \frac{1}{N} \sum_{i=1}^{N} q_i(t),$$

(4.4)

and use it to center all the trajectories:

$$q_i^*(t) = q_i(t) - q_0(t).$$

(4.5)
The covariance function is defined as

$$v(t, s) = \frac{1}{N} \sum_{i=1}^{N} q_i^*(t) q_i^*(s), \quad (4.6)$$

and the eigenfunctions $g(t)$ are determined by solving

$$\int_1^T v(t, s) g(s) ds = \lambda g(t), \quad (4.7)$$

where $\lambda$ is the eigenvalue corresponding to $g(t)$. Different eigenfunctions carry different variations in the data, and a simple example is shown in Fig. 4.2.

We select the $M$ eigenfunctions $g_m(t)$, $m = 1, ..., M$, with the largest eigenvalues, and refer to them as motion harmonics.

### 4.3.4 Constructing Trajectories using Motion Harmonics

Using $M$ eigenfunctions $g_m(t)$, $m = 1, ..., M$, a new trajectory can be constructed by

$$q(t) = q_0(t) + \sum_{m=1}^{M} c_m g_m(t), \quad (4.8)$$

where $c_m$ is the coefficient of $g_m(t)$. Since $\{g_m(t)\}$ comes from the data, by using them one can only generate trajectories that lie within the range of variation in the data. To allow shifting of new trajectories, we extend $\{g_m(t)\}$ and add a constant basis $g_{M+1}(t) = 1$:

$$\{g_m(t)\}_{m=1}^{M'} = \{g_m(t)\}_{m=1}^{M} \cup \{g_{M+1}(t)\}, \quad (4.9)$$

where $M' = M + 1$. Thus, a new trajectory is constructed by

$$q(t) = q_0(t) + \sum_{m=1}^{M'} c_m g_m(t). \quad (4.10)$$
Figure 4.2. Illustration of different eigenfunctions carrying different variation. In each subfigure, the mean function is shown as black solid line, the mean function plus the eigenfunction is shown as blue dash-dotted line, and the mean function minus the eigenfunction is shown as red dashed line.
The construction of a new trajectory is determined by the coefficient \( \{c_m\}, m = 1, \ldots, M' \).

### 4.3.5 Incorporating Constraints

For the learned task to be performed in a novel environment or for new goals, a set of \( N_c \) constraints can be specified and are denoted as \( \{e_i\}, i = 1, 2, \ldots, N_c \). The constraints specify the joint-space configuration at time instants \( \{t_i\} \in [1, T], i = 1, 2, \ldots, N_c \).

The joint space trajectories \( \{q_i(t)\}, i = 1, \ldots, N \), can be approximated using the motion harmonics \( \{g_m(t)\}_{m=1}^{M} \) (i.e., without the constant basis \( g_{M+1}(t) = 1 \)), with certain coefficients \( \{c_{m,i}\} \). The coefficients are obtained by computing

\[
c_{m,i} = \int_{T}^{1} \left( q_i(t) - q_0(t) \right) g_m(t) dt,
\]

whose mean is

\[
\bar{c}_m = \frac{1}{N} \sum_{i=1}^{N} c_{m,i} \quad m = 1, \ldots, M.
\]

We want to construct a new trajectory in a way that takes the constraints into consideration and also respects the demonstrated trajectories. The new trajectories are constructed by solving

\[
\min_{c_m} \frac{1}{N_c} \sum_{i=1}^{N_c} \left[ e_i - q_0(t_i) - \sum_{m=1}^{M'} c_m g_m(t_i) \right]^2
+ \frac{\alpha}{M} \sum_{m=1}^{M} (c_m - \bar{c}_m)^2,
\]

where \( \alpha \) is the weighting factor that balances between making the new trajectory stay close to the demonstration and making it meet the constraints.
Since \( q(t) \) must lie within the joint space so that the robot can physically perform it, we add the constraints of joint angle range

\[
q_l \leq q_0(t) + \sum_{m=1}^{M} c_m g_m(t) \leq q_u, \quad t \in [1, T]
\] (4.14)

where \( q_l \) and \( q_u \) are the lower and upper bound, respectively, for the joint.

The problem posed by Eq. (4.13) and (4.14) can be solved by quadratic programming.

### 4.3.6 Dissimilarity Measure

A dissimilarity measure is defined for the evaluation of our motion generation approach. It measures how dissimilar a trajectory is to a set of other trajectories.

A newly generated world-space trajectory \( y \) is compared with every non-preprocessed demonstrated world-space trajectory \( x_i, i = 1, ..., N \), where \( y \) and \( x_i \) are both discretely sampled time series. We use DTW to quantify the distance between \( y \) and \( x_i \) because DTW is a main benchmark of similarity measures for time series and very few similarity measures have been reported to systematically outperform DTW [83, 84]. However, since DTW is distance based, the accumulated distance matrix generated using two trajectories that are far away and with different scales may lead to incorrect alignment or incorrect normalized distance between the trajectories. To avoid that potential problem, we scale and translate each \( x_i \) before comparing it with \( y \), using the final scale \( s_{\text{final}} \) and final translation \( d_{\text{final}} \) returned by adaptation. Let trajectory \( x_i \) be specifically expressed as a time series \( x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,T}\} \), where \( T \) is the number of samples. The center of \( x_i \) is \( \bar{x}_i = \sum_{t=1}^{T} x_{i,t} \), and \( x_i \) that is scaled and translated is

\[
x_i^* = (s_{\text{final}}(x - \bar{x}_i) + \bar{x}_i) + d_{\text{final}}.
\] (4.15)
We define the distance between $x_i^*$ and $y$ as the normalized minimum distance between $x_i^*$ and $y$ computed by DTW, and denote it as $DTW(x_i^*, y)$. The dissimilarity between $y$ and $\{x_i\}$ is the average distance between $y$ and $\{x_i^*\}$. We assume the dissimilarity is always measured between the data $\{x_i\}$ and the new trajectory $y$ that uses the data, and therefore we omit $\{x_i\}$ when we talk about dissimilarity and only mention $y$:

$$\text{dissimilarity}(y) = \frac{1}{N} \sum_{i=1}^{N} DTW(x_i^*, y). \quad (4.16)$$

### 4.3.7 Error Measure

We measure how well trajectories generated by our approach meet the timed constraints by defining the average world-space error:

$$\text{error}(y) = \frac{1}{NN_c} \sum_{i=1}^{N} \sum_{j=1}^{N_c} |y(t_j) - f(e_j)|, \quad (4.17)$$

where $y$ is the generated trajectory and $e_j$ is the $j$-th constraint at time $t_j$.

### 4.4 Experiments and Evaluation

We tested our approach using five sets of data taken from [85]: beat, hand over, answer phone, pull gun, and punch. Among those tasks, answer phone and pull gun are performed by the same person, and each of the rest three is performed by a different person. Each task is repeated a number of times, and there are a total of 63 repetitions/trials.

The robot on which we tested our approach is NAO H25 v3.3. We set the hand as the end effector and consider the right arm as the kinematics chain (Fig. 4.3). Thus, the root joint for human is $R_{\text{collar}}$ [85] and for NAO it is $R_{\text{ShoulderPitch}}$. We include three more joints for NAO: $R_{\text{shoulderRoll}}$, $R_{\text{ElbowYaw}}$, and $R_{\text{ElbowRoll}}$. Thus our robot model has a degree of freedom (DoF) of four, which severely limits the options of orientation when it reaches
Figure 4.3. We define the right arm of NAO as the kinematics chain.

certain positions. Hence, in this paper, we only use the position information \((x, y, z)\), and thus, each motion trial is a three-dimensional trajectory. The distance in the world space is measured in millimeters.

Our implementation of DTW uses the Sakoe-Chiba local constraint with a slope range of \([0.5, 2]\) and which implies the Itakura global constraint.

Being an iterative process, the adaptation is affected by the initial scale and inter-iteration scaling factor. A large initial scale and a small scaling factor makes the algorithm runs slower, but gives it a higher probability to converge.

Ramsay’s FDA Matlab package [33] is used to represent trajectories with functions and obtain motion harmonics. The eigenanalysis considers all the joints together. Fig. 4.4 shows the first three eigenfunctions for dataset \(beat\). Twenty B-spline basis functions are
First, we compare with the classical Linear Segment with Parabolic Blend approach (LSPB) [87]. Let \( \{x_i\} \) be the world-space trajectories returned by adaptation, where each \( x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,T}\} \) is a time series. The mean trajectory \( \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_T\} \). We select the start point \( x_s = \bar{x}_1 + 20 \cdot [r_1, r_2, r_3]^T \), and the end point \( x_e = \bar{x}_T + 20 \cdot [r_1, r_2, r_3]^T \),
where $r_1, r_2, r_3 \sim U(0, 1)$. Then we compute their configuration $q_s = f^{-1}(x_s)$ and $q_e = f^{-1}(x_e)$, where $f^{-1}()$ represents inverse kinematics. We specify two timed configuration constraints: $\{ e_i \} = \{ q_s, q_e \}$ with time $\{ t_i \} = \{ 1, T \}$. Our approach produces different trajectories given different weighting factors $\alpha$. We choose 14 values of $\alpha$ from $[0, 100]$, and for each $\alpha$, we run the approach 20 times. Thus, there are $14 \times 20 = 280$ different sets of constraints.

Figure 4.5. Comparison between our method and OMPL and LSPB. The left column shows the error of our method, and the right column shows the dissimilarity of OMPL, LSPB, and our method. Each row corresponds to one dataset: (A) beat (B) handover (C) answer phone (D) pull gun and (E) punch.
4.4.2 Comparing with the Open Motion Planning Library

Second, we compare with trajectories generated by control-based planners through the Open Motion Planning Library (OMPL) [88].

The left column of Fig. 4.5 shows the average world-space error of constraints \( q_s, q_e \). As \( \alpha \) increases, the generated trajectory leans towards the demonstrated data and cares less about the constraints, and the average error increases. Conversely, as shown in the right column of Fig. 4.5, the dissimilarity of the generated trajectory goes down as \( \alpha \) increases. When \( \alpha \) reaches a certain point, the dissimilarity of our trajectories becomes lower than the dissimilarity of both the LSPB and the OMPL trajectories, and stays low thereafter. In addition, a range of \( \alpha \) is observed for which both the average error and the dissimilarity are low. The \( \alpha \)'s in such range may be considered desirable for automatic motion generation.

The paper is accompanied by a video that shows the NAO robot executing trajectories of the experimented tasks generated by the compared approaches.

4.4.3 Avoiding Obstacles with Guidance of Via Points

Third, in addition to start and end points, via points are added to guide the trajectory. We test if the trajectory can clear an obstacle in the configuration space. The start and end points are inherited from the last two experiments, and the via points are the optimal path states generated by OMPL which clear the obstacle.

Fig. 4.6 shows sufficient guidance provided by the via points. When \( \alpha \) is small, the trajectory weighs the via points more and by which avoids the obstacles. As \( \alpha \) becomes larger, the demonstrated data shows more influence, and the trajectory hits the obstacle. Fig. 4.7 shows similar phenomenon. In contrast, when the guidance provided by the via points is poor, the trajectory hits the obstacle even if it strictly adheres to the via points. This is shown in Fig. 4.8, where the via point resides too close to the obstacle.
Figure 4.6. Result of experiment *obstacle* with sufficient guidance from via points
Figure 4.7. Another result of experiment obstacle with sufficient via point-guidance
Figure 4.8. Result of experiment *clearing obstacle* with poor guidance from via points
The approach as presented does not specifically deal with obstacles, but as the results show, it can generate obstacle-clearing trajectories if quality via points are provided from motion planners.

4.5 Summary

In this work, we explored the approach of representing a trajectory as the linear combination of a set of eigen-trajectories with each eigen-trajectory representing a different pattern of variation. The approach successfully generates new trajectories that meet various user-defined constraints. The trajectory generation process is not causal, contradicting the fact that motions are executed causally in the real world. The entire trajectory must be generated again if any point on the trajectory needs changing. The computation is global and is therefore unnecessarily costly, and may produce a sub-optimal solution. Before extracting the eigen-trajectories, the data must be temporally aligned, which may lead to information loss. We think that a desired approach should generate a trajectory point by point, one after another, agreeing with the real world. It should avoid tempering with the temporal information of a trajectory before processing it.
CHAPTER 5

POURING TRAJECTORY GENERATION USING RNN

5.1 Note to Reader

Portions of this chapter have been previously published in ”Learning to pour,” 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 7005-7010, and have been reproduced with permission from IEEE.

5.2 Introduction

In this chapter, we propose an approach to trajectory generation based on recurrent neural networks, which solved the problems that existed in the algorithm presented in the previous chapter (Chap. 4). The approach generates a trajectory one point at a time, and keeps the temporal pattern of the data intact in learning. In this work, we focus our effort on generating the trajectories of pouring. Pouring is one of the most frequently executed motions in cooking scenarios, second only to pick-and-place [36, 89, 90], and therefore, the skill of pouring will prove useful once mastered. Also, pouring is a relatively simple task, which enables us to concentrate on the method rather than studying the physical mechanism of the task.

5.3 Methodology for Pouring Trajectory Generation

In this section, we describe in detail our algorithm for generating a pouring trajectory which builds on long short-term memory. To explain why we choose RNN as the building
block, prior to the system description, we review the basics of traditional RNNs, and of one particular structure, the long short-term memory.

### 5.3.1 Recurrent Neural Network

A recurrent neural network (RNN) conducts its computation one step at a time, and at any step its input consists of two parts: a given input, and its own output from the previous time step. The idea is shown in Eq. (5.1) where $x_t$ is the given input, $h_{t-1}$ and $h_t$ are output from the previous and at the current step. The weight $W$ and bias $b$ can be learned using Backpropagation Through Time [91].

$$h_t = \tanh(W[h_{t-1}, x_t]^T + b) \quad (5.1)$$

In theory, by including its past output in its input, an RNN takes the entire history of given inputs into account when it conducts computation, and therefore is inherently suitable for handling sequential data. However, the traditional RNN as shown in Eq. (5.1) is difficult to train and has the vanishing gradients problem, and therefore is inadequate for problems involving long-term dependency [92, 93]. Long short-term memory (LSTM) is a specific RNN design that overcomes the vanishing gradient problem [93]. We use a version of LSTM
whose working mechanism is described by [1]:

\[
i = \text{sigm} \left( W_i [h_{t-1}, x_t]^T + b_i \right) \tag{5.2}
\]

\[
o = \text{sigm} \left( W_o [h_{t-1}, x_t]^T + b_o \right) \tag{5.3}
\]

\[
f = \text{sigm} \left( W_f [h_{t-1}, x_t]^T + b_f \right) \tag{5.4}
\]

\[
g = \tanh \left( W_g [h_{t-1}, x_t]^T + b_g \right) \tag{5.5}
\]

\[
c_t = f \odot c_{t-1} + i \odot g \tag{5.6}
\]

\[
h_t = o \odot \tanh(c_t) \tag{5.7}
\]

where \( i, o, f \) are the input, output, and forget gates respectively, \( c \) is the cell, \( \text{sigm} \) is short for sigmoid, and \( \odot \) represents element-wise multiplication. Fig. 6.2 gives an illustration.

We identify RNN, and specifically LSTM, as the architecture with which we build our pouring system. The reasons include:

1. The structure of RNN makes it inherently fit for handling sequences.
2. An RNN is capable of modeling dynamical systems. Since a dynamical system is powered by velocity (or acceleration), it has the ability to react to changes of the environment.

3. RNN’s have a proven ability to generate both categorical and continuous-valued sequences.

4. An RNN eliminates the needs for temporally aligning sequences before modeling, and therefore preserves the dynamics in a sequence.

5. LSTM supercedes the traditional RNN, and has the proven ability to handle long-term dependency.

5.3.2 Generating Pouring Trajectories

The pouring system predicts the velocity of rotation using the force feedback produced by the cup, which is shown as (middle) in Fig. 5.1.

We assume $n$ trials of a pouring motion are available. The data of trial $i$ are represented by $(\theta_1...T_i, f_1...T_i, z)^{(i)}$, where $\theta_1...T_i$ is the sequence of cup rotation, $T_i$ is the sequence length, $f_1...T_i$ is the sequence of sensed force, and $z$ represents static data that characterize the trial. For simplicity, we assume $\theta, f, z$ are all one-dimensional.

We refer to the system that predicts the velocity of rotation as $\text{vel}$. The actual velocity is computed by

$$\omega_t = \theta_{t+1} - \theta_t, \quad t = 1...T_i - 1.$$ (5.8)

At step $t$, $\text{vel}$ takes $[\theta_t, f_t, z]^\top$ as input, and generates predicted velocity $\hat{\omega}_t$:

$$\mathbf{h}_t = \text{LSTM}([\theta_t, f_t, z]^\top)$$ (5.9)

$$\hat{\omega}_t = \text{fc}(\mathbf{h}_t)$$ (5.10)
Figure 5.2. The architectures of (left) frc, (middle) vel, (right) stp. LSTM16 refers to 16 LSTM units. FC refers to fully connected.
where ‘fc’ is short for ‘fully connected’. The loss is defined using mean squared error:

$$L_{\text{vel}} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T_i - 1} \sum_{t=1}^{T_i-1} (\omega_t^{(i)} - \hat{\omega}_t^{(i)})^2.$$  \hspace{1cm} (5.11)

In order to automatically stop the generation process after the pouring task has completed, we create a stopping system. We refer to the system that stops the pouring motion as $\text{stp}$ shown as (right) in Fig 5.2, which is a binary classifier. At step $t$, $\text{stp}$ takes $[\theta_t, f_t, z]$ as input, and outputs a 2-vector $r_t$. We define class 0 as ‘continue’, and class 1 as ‘stop’.

$$h_t = \text{LSTM}([\theta_t, f_t, z]^T)$$ \hspace{1cm} (5.12)

$$r_t = \text{fc}(h_t)$$ \hspace{1cm} (5.13)

$$s_t = \text{softmax}(r_t)$$ \hspace{1cm} (5.14)

Let the target be represented by a trivial one-hot vector $s'_t = [s'_{t,1}, s'_{t,2}]^T$, where $s'_{t,1}, s'_{t,2} \in \{0, 1\}$ and $s'_{t,1} + s'_{t,2} = 1$. The loss is defined using cross entropy:

$$L_{\text{stp}} = - \sum_{i=1}^{n} \sum_{t=1}^{T_i} (s'_{t,1} \ln s_{t,1}^{(i)} + s'_{t,2} \ln s_{t,2}^{(i)})$$ \hspace{1cm} (5.15)

The initial state of LSTM includes $c_0$ and $h_0$, which are obtained by

$$c_0 = \text{fc}([\theta_1, f_1, z]^T),$$ \hspace{1cm} (5.16)

$$h_0 = \text{tanh}(c_0),$$ \hspace{1cm} (5.17)

as shown in Fig. 5.3.

The trajectory is generated by first initializing $\text{vel}$ and $\text{stp}$, and then a process of generating and executing rotational velocities. Specifically, the trajectory generation process is described in Alg. 1.
**Algorithm 1** Trajectory Generation

1: Initialize vel and stp using $[\theta_1, f_1, z]^T$

2: $t \leftarrow 1$

3: while True do

4: $\omega_t \leftarrow \text{vel}([\theta_t, f_t, z]^T)$

5: $\theta_{t+1} \leftarrow \theta_t + \omega_t$

6: $s \leftarrow \text{argmax stp}([\theta_t, f_t, z]^T)$

7: $t \leftarrow t + 1$

8: if $s == 1$ then

9: Break

10: end if

11: end while

Figure 5.3. Initializing frc, vel, stp
5.4 Data Preparation and Training

The equipment for data collection includes six different cups, ten different containers, one ATI mini40 force and torque (FT) sensor, and one Polhemus Patriot motion tracker. We refer to the pour-from container as cup and the pour-to container as container. All cups are different from one another and so are all the containers. The FT sensor records \((f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)\) at 1KHz. The motion tracker records \((x, y, z, \text{yaw}, \text{pitch}, \text{roll})\) at 60Hz. The cup, the force sensor, and the motion tracker are connected by 3D printed adapters, shown in Fig. 5.4. The materials that are poured include water, beans, and ice.

We obtain the empty reading by keeping an empty cup in a level position, taking 500 FT samples (which takes 0.5 seconds), and then taking the average. Similarly, for each trial, we obtain the initial reading right before the trial with material in the cup, and the final reading right after the trial with or without material in the cup depending on the trial.

We define the sensed force as

\[
    f = \sqrt{f_x^2 + f_y^2 + f_z^2}. \tag{5.18}
\]
In total we collected 1,138 trials which involved 3 subjects. Each trial is represented by a sequence \(\{\mathbf{a}_t\}_{t=1}^{T_i}\) where \(\mathbf{a}_t \in \mathbb{R}^{10}\) and which includes

1. \(\theta_t\), rotation angle at time \(t\) (degree)
2. \(f_t\), sensed force at time \(t\) (lbf)
3. \(f_{\text{init}}\), sensed force before pouring (lbf)
4. \(f_{\text{empty}}\), sensed force while cup is empty (lbf)
5. \(f_{\text{final}}\), sensed force after pouring (lbf)
6. \(d_{\text{cup}}\), diameter of the cup (mm)
7. \(h_{\text{cup}}\), height of the cup (mm)
8. \(d_{\text{ctn}}\), diameter of the container (mm)
9. \(h_{\text{ctn}}\), height of the container (mm)
10. \(\rho\), material density / water density (unitless)

We pad all the sequences to the maximum length in the data: \(T_{\text{max}} = \max(\{T_i\})\). For \textit{vel}, we pad using zero because zero padding makes it easy to compute the original length of a sequence during training. For \textit{stp}, we pad using the end value of the sequence because \textit{stp} is intended to be used on generated motions which will not have zero padding.

In this work we aim to learn and test the system’s ability to generalize to unseen pouring situations. Therefore, we extract certain pouring situations from the data and use them as the test set (Sec. 5.5 provides the list of those situations). We shuffle the rest of the data, which exclude those pouring situations, using a fixed seed for the random number generator. Then we use the first 80% of the shuffled data for training and the rest 20% for validation.
For training and validation, the system applies Alg. 1 which uses the force available in the
data. For testing, the system applies Alg. 2 which generates the force by itself.

We train using the Adam optimizer [94] and set the learning rate to 0.01. We trained
each system for a fixed number of epochs: 4,000 for \texttt{vel}, and 2,000 for \texttt{stp}. The training
error for \texttt{vel} ranges from 0.002 to 0.005 (mm), and the accuracy of \texttt{stp} ranges from 0.9 to
0.98.

5.4.1 Training Force Estimation

In order to run our approach in simulation, we need to have force feedback after we have
arrived at a new rotation. Real force feedback is not applicable in simulation. The movement
of the liquid during pouring forms a complex dynamical system and is difficult to calculate
analytically. Thus, to get force feedback, we decided to generate the force by ourselves. To
that end, we learn from data the mapping relationship from rotation angles to force, and
then use the learned model to estimate the force corresponding to current rotation.

Thus, we need to train a new system. We refer to the system that estimates the sensed
force from rotation as \texttt{frc}, shown as (left) in Fig. 5.2. At step \texttt{t}, \texttt{frc} takes $[\theta_t, z]^{\top}$ as input,
and produces estimated force $\hat{f}_t$:

$$h_t = \text{LSTM}([\theta_t, z]^{\top})$$  \hspace{1cm} (5.19)

$$\hat{f}_t = fc(h_t)$$  \hspace{1cm} (5.20)

The loss is defined using mean squared error:

$$L_{\text{frc}} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T_i} \sum_{t=1}^{T_i} (f_t^{(i)} - \hat{f}_t^{(i)})^2.$$  \hspace{1cm} (5.21)
The initialization of \( \mathbf{frc} \) includes
\[
c_0 = \mathbf{fc}([\theta_1, z]^\top), \tag{5.22}
\]
\[
h_0 = \tanh(c_0), \tag{5.23}
\]
as shown in Fig. 5.3.

The data preparation for \( \mathbf{frc} \) uses zero padding. We train the \( \mathbf{frc} \) with a fixed 2000 epochs, and the error ranges between 0.002 to 0.003 (lbf).

With \( \mathbf{frc} \), the trajectory generation process needs modification. Force can no longer be assumed to be available, but must be produced explicitly by \( \mathbf{frc} \). The modified trajectory generation process is shown in Alg. 2.

```
Algorithm 2 Trajectory generation for simulation
1: Initialize \( \mathbf{frc} \) using \([\theta_1, z]^\top\)
2: \( f_1 \leftarrow \mathbf{frc}([\theta_1, z]^\top) \)
3: Initialize \( \mathbf{vel} \) and \( \mathbf{stp} \) using \([\theta_1, f_1, z]^\top\)
4: \( t \leftarrow 1 \)
5: while \( t < T_{\text{max}} \) do
6: \( \omega_t \leftarrow \mathbf{vel}([\theta_t, f_t, z]^\top) \)
7: \( \theta_{t+1} \leftarrow \theta_t + \omega_t \)
8: \( s \leftarrow \text{argmax} \ \mathbf{stp}([\theta_t, f_t, z]^\top) \)
9: \( f_{t+1} \leftarrow \mathbf{frc}([\theta_{t+1}, z]^\top) \)
10: \( t \leftarrow t + 1 \)
11: if \( s == 1 \) then
12: Break
13: end if
14: end while
```

5.5 Experiment on Generalization

We evaluate the generalization ability of the system and see if it can generate pouring motion in unseen situations. Given a test sequence, we extract \( \theta_1 \) and \( z \), and generate a sequence using Alg. 2. The evaluation is conducted in simulation.
We test the system using unseen

1. cup,
2. container,
3. material,
4. cup and container,
5. container and material,
6. cup and material,
7. cup and container and material.

5.5.1 Identifying Success

We evaluate the generalization ability of the pouring system using dynamic time warping (DTW) \[95\], which gives the minimum normalized distance between two trajectories.

We provide a set of test sequence which include an element that is unseen during training and see if the system is able to adapt to the changes. Let the set of test sequences be \(\{x_i\}_{i=1}^m\).

We first compute the distance between each pair of test sequences and draw a histogram:

\[
h_1 = \text{hist}(\{\text{dtw}(x_i, x_j)\}_{i \neq j}) \quad i, j = 1, 2, \ldots, m.
\]  \hspace{1cm} (5.24)

Each \(x_i\) can be used to generate a new trajectory \(x'_i\). We compute the distance between \(x'_i\) and every test sequence \(x_j\) and draw another histogram.

\[
h_2 = \text{hist}(\{\text{dtw}(x'_i, x_j)\}) \quad i, j = 1, 2, \ldots, m.
\]  \hspace{1cm} (5.25)
Both histograms are normalized. We visually compare the similarity between $h_1$ and $h_2$. If they are similar, then it means the generated trajectories are similar to the trajectories executed by humans, which indicates that the generalization succeeds. The system fails to generalize if otherwise.

### 5.5.2 Results

The results for the seven cases of unseen elements of the pouring characteristics are shown in Fig. 5.5 to 5.11. Generalization on cup, or container, or material alone is successful because the pairing histograms are similar (Fig. 5.5, 5.6 and 5.7). Generalizing on cup and container (Fig. 5.8) and container and material (Fig. 5.9) can be considered successful because of the similarity in the concentration of the small-distances, despite the difference on mid to high-valued distance parts, which occupy only a small portion of all the distances. Generalizing on cup and material fails as well as on cup and container and materials, as shown in Fig. 5.10 and Fig. 5.11. For cup and container and material, only 8 test sequences are available, which may partly contribute to the difference between the two histograms.

### 5.6 Discussion

The system successfully generalizes when either a cup, a container, or the material changes, and starts to stumble when changes of more than one element are present. Since the total size of data does not change, the more that is left out for testing (more unseen elements), the less there is available for training. Thus, the system accepts a weaker set of training data and after which it faces more demanding challenges. The observed results of degrading performance with increasing generalization difficulty was expected.
5.7 Summary

In this chapter, we present an algorithm for pouring trajectory generation based on RNN. The algorithm consists of three RNNs, of which the velocity generator and the force estimator are essential. Using two models inside an algorithm requires both models to be trained well. The algorithm exhibits potential but it was not evaluated using a real physical system. The true ability of the algorithm must be evaluated on a physical system. Our next step is to test the algorithm partly or in full on a physical system.
Figure 5.6. Generalizing on an unseen container
Figure 5.7. Generalizing on an unseen material
Figure 5.8. Generalizing on an unseen cup and container
Figure 5.9. Generalizing on an unseen container and material
Figure 5.10. Generalizing on an unseen cup and material
Figure 5.11. Generalizing on an unseen cup, container, and material
CHAPTER 6

ACCURATE LIQUID POURING: LEARNING AND GENERALIZATION

In the previous chapter (Chap. 5), we discussed generating the trajectories of pouring in simulation. In this chapter, we go further in two aspects:

1. we design an algorithm that pours accurately,

2. we evaluate the algorithm using a physical system.

The work in this chapter differs in multiple aspects from the work in the previous chapter. Therefore, we discuss it in full and only leave out the basics of RNN which we have covered in the last chapter.

6.1 Related Work on Accurate Pouring

Pouring is a task commonly seen in people’s daily lives and is also useful in casting factories. One important ability of a pouring algorithm is its accuracy. In casting factories where molten metal is poured into molds, accurate pouring is required. [96] proposes predicting the residual pouring quantity of the liquid to increase the accuracy of pouring. [97] introduces predictive sequence control which suppresses the increase of error when the pouring amount increases. Factory-specific pouring algorithms achieve high accuracy but cannot be applied to different source containers.

Besides accuracy, the other ability of a pouring algorithm is its adaptability, i.e. that of pouring from different source containers, pouring to different receiving containers, and
pouring liquids with different physical properties. If the algorithm bases itself on learning, its adaptability is usually referred to as generalizability, i.e., being able to perform tasks that were not taught during learning. In [98], the authors proposed warping the point cloud of known objects to the shape of a new object, which enables pouring gel balls from one new source cup to three different receiving containers. The algorithm shows adaptability, but not accuracy.

It is more desirable that an algorithm exhibits both accuracy and adaptability. In [99] they use a deep neural network to estimate the volume of liquid in a cup from raw visual data and uses PID controllers to control the rotation of the robot arm. In 30 pours the average error was 38 milliliter (mL). Three different receiving containers were tested, for which the robot performed approximately the same. However, the authors did not claim that the algorithm can generalize to different target containers. [100] uses RGB-D point cloud of the receiving cup to determine the liquid height and PID controller to control the rotating angle of the source cup. The pouring action is programmed and is stopped as soon as the desired height is achieved. The mean error of pouring water to three different cups is 23.9 mL, 13.2mL, and 30.5mL respectively. The algorithm does not involve learning and can pour both transparent and opaque liquid. [101] uses reinforcement learning to learn the policy of pouring water in simulation and tested the policy in actual robots. In the test, the poured height is estimated from RGB-D images. The algorithm averaged a 19.96mL error over 40 pours, and it generalized to milk, orange juice and apple juice but not to olive oil. The algorithm did not consider using different source or receiving containers.

6.2 Problem Description and Approach

The amount of liquid can be represented using either weight or volume. Volume can be perceived visually, is commonly used for liquid and is intuitive for measuring liquid. In this
work, for explaining the theory and presenting the experimental results, we use volume to represent the amount of liquid.

We define the task of accurate pouring as pouring the requested volume accurately from a source container to a receiving container. Initially there is certain volume of liquid in the source container. If the source container is full then as soon as it starts rotating, the liquid will come out. If the source container is not full then there is a time period during which the source container is rotating but no liquid comes out. After the liquid comes out, it goes into the receiving container where it then stays and therefore the poured volume can only increase and can never decrease. Depending on the liquid volume inside the source container, to stop the liquid from coming out, the source container has to either stop rotating or rotate back to a certain angle. However, even if the source container has stopped rotating or has been rotating back, the liquid may keep coming out and as a result the poured volume increases.

The pouring process is sequential and the poured volume is determined by the trajectory of the rotation velocities of the source container. The pouring process as described above can be modeled as a discrete time series as shown in Alg. 3. where $t_1$ is the initial time instant, $\Delta t$ is the time interval, $\theta(t)$ and $\omega(t)$ are the rotation angle and angular velocity

![Diagram](image)

**Figure 6.1.** The sequential pouring system. The input is the current angular velocity and the current poured volume and the output is the poured volume for the next time step.
Algorithm 3 Pouring Model

1: for $i$ in $(1, 2, \ldots)$ do
2: \hspace{1em} $t = t_1 + (i - 1)\Delta t$
3: \hspace{1em} $\theta(t + \Delta t) = \theta(t) + \omega(t)\Delta t$
4: \hspace{1em} $\text{vol}(t + \Delta t) = F((\omega(\tau))_{\tau=t_1}^t)$
5: end for

of the source container, respectively, $\text{vol}(t)$ is the poured volume, $F(\cdot)$ denotes the pouring system. $(\omega(\tau))_{\tau=t_1}^t = (\omega(t_1), \ldots, \omega(t))$ is the sequence of velocities. The effect of the velocity $\omega(t)$ executed at time $t$ is observed at the next time step, time $t + \Delta t$, and the effects are the next rotation angle $\theta(t + \Delta t)$ and the next poured volume $\text{vol}(t + \Delta t)$. The rotation angle $\theta(t + \Delta t)$ is the numerical integration of the sequence of velocities $(\omega(\tau))_{\tau=t_1}^t$. The poured volume $\text{vol}(t + \Delta t)$ is the result of the sequence of velocities $(\omega(\tau))_{\tau=t_1}^t$ acted through the pouring system $F(\cdot)$.

The pouring system $F(\cdot)$ is a complicated nonlinear time-variant system that can be affected by many factors including factors that change with time and static factors. For example, the pouring system can be

$$\text{vol}(t + \Delta t) = F((\omega(\tau))_{\tau=t_1}^t, d(t), H, s(h)|_{h=0}^H, \text{vol}_{2\text{pour}}, \text{vol}_{\text{total}}, T, \rho, \mu) \quad (6.1)$$

where

- $d(t)$ is the translation vector from the tip of the source container to the center of the mouth of the receiving container at time $t$
- $H$ is the height of the source container
- $s(h)|_{h=0}^H$ is the evolution of $s(h)$ from $h = 0$ through $h = H$ where $s(h)$ is the shape of the cross-section of the source container at height $h$
• \( vol_{\text{total}} \) is the total volume of liquid in the source container before pouring

• \( vol_{\text{pour}} \) is the volume to pour, i.e. the target volume

• \( T \) is the temperature

• \( \rho \) is the density of the liquid at temperature \( T \)

• \( \mu \) is the viscosity of the liquid at temperature \( T \)

Among the factors, \((\omega(\tau))_{\tau=t_1}^t\) and \(d(t)\) change with time and the others are static. This example is far from capturing all the possible factors that affect pouring.

The angular velocity \( \omega(t) \) is the action that pushes the pouring process forward. To perform pouring, we need to generate the angular velocity \( \omega(t) \). The generator needs to take the target volume as input. It also needs to be sequential. At any time step during pouring, the generator should take the current poured volume as input, compare it with the target volume, and adjust the velocity accordingly. The generator is represented as

\[
\omega(t) = G((\omega(\tau))_{\tau=t_1}^t, vol(t), vol_{\text{pour}}),
\]

(6.2)

where \( G(\cdot) \) denotes the generator and \( vol_{\text{pour}} \) is the target volume. With the velocity generator represented, the pouring process is written again in Alg. 4.

Algorithm 4 Pouring Model With Velocity Generator

1: for \( i \) in \( (1, 2, \ldots) \) do
2: \( t = t_1 + (i - 1)\Delta t \)
3: \( \omega(t) = G((\omega(\tau))_{\tau=t_1}^t, vol(t), vol_{\text{pour}}) \)
4: \( \theta(t + \Delta t) = \theta(t) + \omega(t)\Delta t \)
5: \( vol(t + \Delta t) = F((\omega(\tau))_{\tau=t_1}^t) \)
6: end for
6.2.1 RNN for Velocity Generation

A natural solution for velocity generation of pouring is Model Predictive Control (MPC) [102], which optimizes control inputs based on their corresponding predicted future outcomes. However, using MPC for pouring requires that we know the pouring system \( F(\cdot) \) so that we can perform predictions of future outcomes of candidate velocities. Since an accurate \( F(\cdot) \) is difficult to obtain, we cannot readily use MPC and need to turn to other solutions.

We intend to identify a model for velocity generation and learn the parameters of the model from human demonstrations. We seek two properties from the candidate model for velocity generation:

1. The model should be inherently capable of dealing with sequences because all data are sequences.

2. The model should be able to learn effectively with variable lengths of sequences because human demonstrations vary in lengths.

We use RNN to model the velocity generator. RNN is a class of neural networks that is designed to process its inputs in order. Previously we discussed the mechanism of plain RNN and plain LSTM in Section 5.3.1. Peepholes were added to plain LSTM to enable the access of all gates to the memory cell [103]. The mechanism of peephole LSTM is illustrated
in Fig. 6.2 and is written as:

$$i = \text{sigm} \left( W_i [h(t-1), x(t)]^T + b_i + p_i \odot c(t-1) \right) \quad (6.3)$$

$$f = \text{sigm} \left( W_f [h(t-1), x(t)]^T + b_f + p_f \odot c(t-1) \right) \quad (6.4)$$

$$g = \tanh \left( W_g [h(t-1), x(t)]^T + b_g \right) \quad (6.5)$$

$$c(t) = f \odot c(t-1) + i \odot g \quad (6.6)$$

$$o = \text{sigm} \left( W_o [h(t-1), x(t)]^T + b_o + p_o \odot c(t) \right) \quad (6.7)$$

$$h(t) = o \odot \tanh(c(t)) \quad (6.8)$$

where $i$, $o$, and $f$ are the input, output, and forget gates respectively. $c$ is the memory cell. $p_i$, $p_o$, and $p_f$ are the peephole connection weights for gate $i$, $o$ and $f$, respectively. Sigm represents the sigmoid function and is used to implement gates. $\odot$ represent element-wise multiplication. In this work, we specifically use peephole LSTMs to model the velocity generator.
6.2.2 Input Features

We need to decide the input features to the RNN at any time step. Each feature corresponds to a type of data. We write Eq. (6.2) again below for convenience:

$$\omega(t) = G((\omega(\tau))_{\tau=t_1}^{t-\Delta t}, vol(t), vol_{2pour})$$  \hspace{1cm} (6.9)

The first feature is the sequence of velocities $\omega(\tau)_{\tau=t_1}^{t-\Delta t}$. $\theta(t)$ is the numerical integration of the sequence of velocities and therefore we identify $\theta(t)$ as the first feature. The second feature is the current poured volume $vol(t)$. The third feature is the target volume $vol_{2pour}$. Thus we have set all three parameters in Eq. (6.2) as features.

Corresponding to the target volume $vol_{2pour}$, the initial volume of liquid in the source container $vol_{total}$ can be set as a feature. We can also have features that describe the shape of the source container. We model the source container as a cylinder and set both the height $h$ and the body diameter $d$ as features.

The four static features $vol_{2pour}$, $vol_{total}$, $h$, and $d$ describe a pouring task and distinguishes one task from another. The two sequential features $\theta(t)$ and $vol(t)$ are the results of executing the task described by the four static features. Fig. 6.3 illustrates the six input features.

6.3 Data Collection for Training

We wanted to collect all the input features that we have identified for the network and we needed to decide how to measure volume. Intuitively, the volumes $vol_{total}$, $vol_{2pour}$ can be measured using a measuring cup. However, obtaining $vol(t)$ using a measuring cup requires a real-time video stream of the measuring cup and a computer vision algorithm that extracts the volume from the video stream.

To simply the problem that we have to solve, we decided that we would not include the above vision problem in our solution, and instead we computed the volume from other
Figure 6.3. Six physical quantities to obtain for pouring. \( vol_{2\text{pour}} \) and \( vol_{\text{total}} \) is the target and initial volume. \( d \) and \( h \) are the diameter and height of the source container. \( \theta(t) \) and \( vol(t) \) are the sequences of rotation angle and of the poured volume.

quantities. The volume is mass \( m \) divided by density \( \rho \), i.e. \( v = \frac{m}{\rho} \). We consider weight which is the gravitational force acted on an object that keeps the object in place. The weight \( f \) is the product of mass \( m \) and gravitational acceleration \( g \), i.e. \( f = mg \). Thus volume can be calculated from weight:

\[
v = \frac{f}{\rho g}, \tag{6.10}
\]

and therefore can be represented by weight. We represent \( vol_{\text{total}} \) by its corresponding weight \( f_{\text{total}} \), \( vol_{2\text{pour}} \) by weight \( f_{2\text{pour}} \), and similarly the current poured volume \( vol(t) \) by weight \( f(t) \).

Fig. 6.4 illustrates the setup for our data collection. We collected data of pouring water from 9 different source containers into the same receiving container. The 9 source containers are shown as the left half of Fig. 6.5. We measured \( h \) and \( d \) of each source container in millimeter using a ruler. We 3D-printed a handle where the source container was mounted on one end, and a Polhemus Patriot motion tracker was mounted on the other end. The motion tracker recorded the rotating angles \( \theta(t) \)'s of the source container in degrees. We placed an ATI mini40 force/torque sensor under the receiving container to record the raw
force reading \( f_{\text{raw}}(t) \)'s in pound-force (lbf). We obtained \( f_{\text{total}} \) and \( f_{\text{2pour}} \) from \( f_{\text{raw}}(t) \). In each trial, \( f_{\text{total}} > f_{\text{2pour}} \), that is, there was water left in the source container after pouring.

\( \theta(t) \)'s were recorded at 60Hz and \( f_{\text{raw}}(t)'t \) were recorded at 1KHz. The collected pouring data is part of RPAL daily interactive manipulation dataset [104].

### 6.4 Implementation

The network can have multiple layers and each layer can contain multiple peephole LSTM units. RNN Dropout [1] is applied between layers. The final layer is a fully connected layer with linear activation which generates the angular velocity. The mechanism of the network with \( L \) layers at time \( t \) is represented in Alg. 5, where LSTM(\( \cdot; n_{\text{unit}} \)) means LSTM block

\[
\text{Algorithm 5 Velocity Generator Network Structure}
\]

1: \( h_0(t) = x(t) \)
2: for \( i = (1, 2, \ldots, L) \) do
3: \( h_i(t) = \text{LSTM}(h_{i-1}(t); n_{\text{unit}}) \)
4: \( h_i(t) = \text{Dropout}(h_i(t); p_{\text{keep}}) \)
5: end for
6: \( \hat{y}(t) = W_y h_L(t) + b_y \)

with \( n_{\text{unit}} \) units, and Dropout(\( \cdot; p_{\text{keep}} \)) means dropout with a keep probability of \( p_{\text{keep}} \). Fig. 6.6 illustrate the network with two LSTM layers and the final layer.

To feed the input features into the network, we group them into a vector
\[
x(t) = [\theta(t), f(t), f_{\text{total}}, f_{\text{2pour}}, h, \kappa]^\top \quad \text{for} \quad t = 1, \ldots, T - 1, \quad \text{where} \quad T \quad \text{is the length of the trial and}
\]

1. \( \theta(t) \) is the rotating angle of the source container.
2. \( f(t) \) is the weight of the poured liquid.
3. \( f_{\text{total}} \) is the weight of the initial amount of liquid present in the source container before pouring.

100
Figure 6.4. Illustration of the data collection setup. The source container is connected to the motion tracker through a 3-D printed adapter. The force sensor is placed underneath the receiving container.
Figure 6.5. All the source containers used in training and in the experiments. The left half labeled as red were used for training and the right half labeled as green were used for experiments. The red cup in the middle was used both for training and for experiments.
Figure 6.6. An example of the network with two LSTM layers and the final layer

4. \( f_{\text{pour}} \) is the weight of the target liquid amount.

5. \( h \) is the height of the source container.

6. \( \kappa \) is the body curvature of the source container.

The body curvature \( \kappa \) of the source container is calculated from the body diameter, \( d \):

\[
\kappa = \frac{2}{d} \quad (6.11)
\]

The angular velocities \( \omega(1:T-1) \) are computed from \( \theta(1:T) \):

\[
\omega(t) = (\theta(t+1) - \theta(t))f_s, \quad t = 1, 2, \ldots, T-1 \quad (6.12)
\]
where $f_s$ is the sampling frequency of $\theta(t)$. For each trial, at time $t \in [1, 2, \ldots, T - 1]$, the input $x(t)$ and target $y(t)$ of the network are

$$x(t) = [\theta(t), f(t), f_{\text{total}}, f_{\text{pour}}, h, \kappa]^T$$

(6.13)

$$y(t) = \omega(t)$$

(6.14)

The output of the network is denoted by $\hat{y}(t)$. Assume we have $N$ trials in total, and each trial has length $T_i$, $i \in [1, 2, \ldots, N]$. The loss function is defined as

$$c = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i - 1} \sum_{t=1}^{T_i-1} (\hat{y}_i(t) - y_i(t))^2.$$  

(6.15)

### 6.5 Data Preparation and Training

We set the sampling frequency $f_s = 60Hz$ since it is the lower one between the frequencies of $\theta(t)$ and $f_{\text{raw}}(t)$. We kept the recorded $\theta(t)$'s intact and downsampled $f_{\text{raw}}(t)$ to 60Hz. We obtain $f(t)$ by filtering the raw reading from the force sensor $f_{\text{raw}}(t)$, specifically

$$f_m(1 : t) \leftarrow \text{median\_filter}(f_{\text{raw}}(1 : t)), \quad \text{window}\_\text{size} = 5,$$

(6.16)

$$f(t) \leftarrow \text{Gaussian\_filter}(f_m(1 : t)), \quad \sigma = 2.$$  

(6.17)

We normalize each input dimension independently using the mean and standard deviation of that dimension.

Our network had 1 layer and 16 LSTM units. We trained models with different numbers of layers and LSTM units and we found the model with 1 layer and 16 units had a simple structure and performed well. We set the keep probability of dropout to be 0.5. Specifically,
The computation for time step $t$ is represented as:

\[
    h(t) = \text{LSTM} (h(t - 1), x(t)) \\
    h_d(t) = \text{Dropout} (h(t)) \\
    \hat{y}(t) = W_y h_d(t) + b_y
\]  

The network is shown in Fig. 6.7.

Learning the network involved 284 trials in total, among which 221 were for training and 63 for validation. Each iteration is an epoch, in which the entire training and validation data were traversed. We trained the network for 2000 iterations/epochs, and picked the model that has the lowest validation loss. We used the Adam optimizer and set the initial learning rate to be 0.001. The code was written using TensorFlow.
6.6 Physical System for Evaluation

To evaluate our approach, we made a physical system that consists of the trained network, a Dynamixel MX-64 motor and the same force sensor with which we collected the data. The motor was placed at a certain height above the surface, and the force sensor was placed on the surface close by. The source container was attached to the motor and the receiving container was placed on top of the force sensor. We placed the receiving container (along with the force sensor) properly according to the particular source container used so that there was little spilling. Fig. 6.8 (Left) shows the setup of the physical system.

The physical system runs at 60Hz, same as the data collection. The time between consecutive time steps is $\Delta t = 0.016$ seconds. Before performing each separate pouring trial, we obtain the four static features which we denote by $z = [f_{total}, f_{2pour}, h, \kappa]$. During the trial, at time step $t$, we obtain $\theta(t)$ from the motor and $f(t)$ from the force sensor, and we feed the input features $x(t) = [\theta(t), f(t), z]^{\top}$ to the network. The network generates velocity $\omega(t)$. The motor executes the velocity. The above process repeats at time step $t + \Delta t$. Fig. 6.8 (Right) shows the working process of the physical system at time $t$.

In the same way as was done in training, the physical system:

1. normalized every input dimension.

2. obtained $f(t)$ by filtering the raw force reading.

6.7 Experiments and Evaluation

We evaluated our system by testing it on pouring certain kinds of liquid from certain source containers. The difficulty of the task changes when the liquid and the source container changes. For each pair of liquid and source container, the system poured 15 times, each time with arbitrary $vol_{total}$ and $vol_{2pour}$ where $vol_{total} > vol_{2pour}$. We show the pouring accuracy of a pair of liquid and source container using a figure, in which we plot the actual poured
Figure 6.8. The physical evaluation system. (Left) The physical system consists of a motor that executes the generated velocity command and a force sensor that monitors the poured amount. The source containers are attached to the motor through a 3-D printed adapter. (Right) Before pouring, we obtain the static features $z = [f_{\text{total}}, f_{\text{pour}}, h, \kappa]$. At time step $t$, the physical system obtains $\theta(t)$ and $f(t)$, combine them with $z$, and send to the network. The network generates velocity command $\omega(t)$ which is executed by the motor.
volume against the target volume for all 15 trials. We also compute the mean and standard deviation of the pouring error: $\mu_e$ and $\sigma_e$ in milliliters and show them together with the liquid type in the figure. By the side of the actual-vs-target figure, we show the source container that was used.

Computing the volume from force requires the density of the liquid $\rho$ and the gravitational acceleration $g$. We used 0.997g/mL for the density for water and 9.80665 m/s$^2$ for gravitational acceleration.

We started with the task that has the lowest difficulty and tested the system on pouring water from the red cup that has been used for training. Fig. 6.9 (a) shows that the accuracy was high, indicating that the learning was successful. Then we increased the difficulty of the tasks and tested the system on pouring water from five different source containers that have not been used for training. The accuracy is shown in Fig. 6.9 (b) through (f), which we show in an increasing order of the error mean $\mu_e$. Compared with the accuracy of using the red cup, the accuracy of using the five unseen source containers is lower, which is within expectation. It is worth noting that although lower than the accuracy of the red cup, the accuracy of the slender bottle (Fig. 6.9 (b)) is still high and is comparable with that of the red cup.

Table 6.1 summarizes the mean and standard deviation of the errors, $\mu_e$ and $\sigma_e$, in milliliters of the system pouring water from different source containers. The table is ordered in an increasing order of the error mean $\mu_e$.

Having evaluated the accuracy of system pouring different but relatively large amount of water, and we would like to know the minimum volume that the system could pour accurately. Therefore we made the system use the red cup to pour 20mL and 15mL, respectively, each for 15 times, and Fig. 6.10 shows the accuracy. Both $\mu_e$ and $\sigma_e$ for pouring 20mL are lower than those of pouring larger volume with the red cup (Fig. 6.9 (a)). The accuracy of pouring
15mL is much lower than that of pouring 20mL and those of pouring larger volume. Thus, 20mL was the minimum volume that the system was able to pour accurately.

In Fig. 6.11, we plot the reading of the force sensor for a 1.0-lbf weight during 300 seconds. In the figure we also show the water volume converted from force. For a 1.0-lbf weight, the force sensor has a nonlinearity error of around 0.01 lbf, which is 1% of 1.0 lbf. The corresponding error in volume is around 5mL.

To have a sense of how accurately the system pours compared with human, we had a human subject pour water to variable target volume more than 15 times and the accuracy is shown in Fig. 6.12. We found that for pouring water with the red cup (Fig. 6.9 (a)), with the slender bottle (Fig. 6.9 (b)), and with the bubble cup (Fig. 6.9 (c)), our system achieved a higher accuracy than human.
<table>
<thead>
<tr>
<th>cup</th>
<th>cup in training</th>
<th>$\mu_e$ (mL)</th>
<th>$\sigma_e$ (mL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>yes</td>
<td>3.71</td>
<td>3.88</td>
</tr>
<tr>
<td>slender bottle</td>
<td>no</td>
<td>4.12</td>
<td>4.29</td>
</tr>
<tr>
<td>bubble</td>
<td>no</td>
<td>6.77</td>
<td>5.76</td>
</tr>
<tr>
<td>glass</td>
<td>no</td>
<td>7.32</td>
<td>8.24</td>
</tr>
<tr>
<td>measuring cup</td>
<td>no</td>
<td>11.29</td>
<td>12.82</td>
</tr>
<tr>
<td>fat bottle</td>
<td>no</td>
<td>12.35</td>
<td>8.88</td>
</tr>
</tbody>
</table>

### 6.8 Generalization Analysis

In the previous section, we showed that our system was able to pour with source containers that were not included in training, which showed that the system could generalize the pouring skill to different source containers.

In this section, we analyzed the generalization of the system by comparing each experiment with the data. To be specific, we compared the records of the static features and of the dynamic features of each experiment with those of the data, respectively.

First we compare the variation of the static features directly. In Fig. 6.13 and 6.14 we show $v_{2\text{pour}}$-vs-$v_{\text{total}}$ and $d$-vs-$h$ contained in the training data and all the experiments. In Fig. 6.13, all the records reside below the diagonal line which represents emptying the source container. This verifies what we said previously that $v_{2\text{pour}} < v_{\text{total}}$ for both data and experiments. Fig. 6.14 shows that in the experiment the heights lie within the range of the data but the diameter reaches a high value (above 120 mm) that the data do not contain.

Then we investigated the static features more thoroughly. The four static input features, $f_{\text{total}}$, $f_{2\text{pour}}$, $h$, and $\kappa$ specify how each time of pouring as a task differs from others. The four static features can each be normalized as done in training and put together as a 4-vector. We refer to the normalized 4-vector as the static vector. A trial is specified by a single static
vector $v \in \mathbb{R}^4$. We use the L2 norm between the static vectors of two trials to quantify the
difference between the two trials, which we refer to as the static distance.

We want to compare the training data with the experiment of each pouring container to see how much the system generalizes. We represent each trial in the training data by

$$a_i = \min\left(\{||v_i - v_j||_2\}_{j=1...N, j \neq i}\right), \quad i = 1, \ldots, N$$

(6.21)
i.e. the minimum among the static distances between trial $i$ and all the other trials, $N$ is the number of trials in the training data. We represent each trial in an experiment by

$$b_i = \min(\{||v_i - v_j||\}_{j=1...N}), \quad i = 1, \ldots, M$$

(6.22)
i.e. the minimum among the static distances between trial $i$ and all the trials in the training data, $M$ is the number of trials in the experiment.

For each experiment, we plot the histogram of $a_i$’s for the training data together with the histogram of $b_i$’s for the experiment. We show the histograms in Fig. 6.15 (a) through (f). In Fig. 6.15 (a), (b), and (c), the histogram of the experiment is within the histogram of the data, which means the tasks have been learned and the system did not generalize. In Fig. 6.15 (d), the histogram of the experiment has a small overlap with that of the data but also extends to a minimum static distance as far as twice the width of the distance coverage in the data. The system generalized to a certain degree. In Fig. 6.15 (e) and (f), the experiment histogram is outside that of the data and reached to a certain distance from the data. The system was able to execute tasks that have not been learned, and therefore generalized.

Similarly to the static features, we compare the two sequential features $\theta(t)$ and $f(t)$ of the data with those of the experiments. The $\theta(t)$’s and $f(t)$’s of a trial result from executing a
task that is specified by a particular static vector $v$, and they represent the specific solution given by the system to fulfill that task. We consider $\theta(t)$ and $f(t)$ as a single sequence $s = \{s(1), s(2), \ldots, s(T)\}$ where $s(t) = [\theta(t), f(t)]^\top \in \mathbb{R}^2$ and $T$ is the length of the sequence. We normalize both $\theta(t)$ and $f(t)$ as was done for the training. Corresponding to the static vector, we refer to the normalized sequence $s$ as the dynamic sequence.

We represent the distance between two sequences $s_i$ and $s_j$ using the normalized distance computed by dynamic time warping (DTW) [95], denoted as $d_{\text{DTW}}(s_i, s_j)$. Similarly to the static vectors, for dynamic sequences, we represent each trial in the training data by

$$p_i = \min\{d_{\text{DTW}}(s_i, s_j)\}_{j=1\ldots N, j \neq i}, \quad i = 1, \ldots, N$$

(6.23)

i.e. the minimum among the normalized DTW distances between trial $i$ and all the other trials, $N$ is the number of trials in the training data. We represent each trial in an experiment by

$$q_i = \min\{d_{\text{DTW}}(s_i, s_j)\}_{j=1\ldots N}, \quad i = 1, \ldots, M$$

(6.24)

i.e. the minimum among the normalized DTW distances between trial $i$ and all the trials in the training data, $M$ is the number of trials in the experiment.

For each experiment, we plot the histogram of $p_i$’s for the training data together with the histogram of $q_i$’s for the experiment. We show the histogram in Fig. 6.16 (a) through (f). In Fig. 6.16 (a) and (b), the histogram of the experiment is within and similar to the histogram of the data: the system repeated what it learned from data. In Fig. 6.16 (c) and (d), the histogram of the experiment is within that of the data but has a different shape. In Fig. 6.16 (e), the histogram of the experiment has a similar shape to that of the data but its DTW distances exhibit a shift to the higher values. The above four experiments shows certain generalization. In Fig. 6.16 (f), the histogram of the experiment has a different shape from that of the data and it also has little overlap with the histogram of the data.
According to Fig. 6.15 (f), the static vectors or the task specifications differ from those in the data, and to perform the tasks Fig. 6.16 (f) shows that to perform the tasks the system did something different from the data. Fig. 6.15 (f) together with Fig. 6.16 (f) show that the system generalized.

6.9 The Effect of Viscosity

We wanted to find out if our system was able to generalize to liquid with different viscosity from water. Therefore, we tested the system on pouring cooking oil and syrup with the red cup, respectively. The red cup was used for training but the data only included it being used for pouring water. Therefore, pouring oil and syrup for the red cup is generalizing. Fig. 6.17 shows the accuracy of pouring oil, which is lower than but comparable with that of pouring water (Fig. 6.9 (a)). It may be because the density and viscosity is not significantly different from water. Fig. 6.18 shows that syrup is always over-poured. It may be because it took longer for the syrup in the air to reach the receiving container which delayed the response of the system. The density of oil and syrup we used for computing the volume from force is 0.92g/mL and 1.37g/mL respectively, in comparison to that of water which is 0.997g/mL.

We speculated that viscosity played a big role in the accuracy of pouring different kinds of liquid. Therefore, in Fig. 6.19 we show the error bars of pouring water, oil, and syrup with the red cup versus their viscosity. The three types of liquid have very different viscosities. We use 1 centipoise (cps) as the viscosity for water, 65 centipoise for oil, and 2000 for syrup. We plotted the viscosities in logarithm scale. Equivalently but in another form, Table 6.2 lists the accuracy of pouring liquids with different viscosities. Fig. 6.19 and Table 6.2 show that error mean \( \mu_e \) increases as the viscosity increases and the relationship is neither linear nor exponential.
Table 6.2. Accuracy of pouring liquids with different viscosities

<table>
<thead>
<tr>
<th>liquid</th>
<th>viscosity (cps)</th>
<th>$\mu_e$ (mL)</th>
<th>$\sigma_e$ (mL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>water</td>
<td>1</td>
<td>3.71</td>
<td>3.88</td>
</tr>
<tr>
<td>oil</td>
<td>65</td>
<td>4.11</td>
<td>4.80</td>
</tr>
<tr>
<td>syrup</td>
<td>2000</td>
<td>15.66</td>
<td>3.43</td>
</tr>
</tbody>
</table>

6.10 Summary

In this chapter, we further explore the utility of the algorithm based on recurrent neural network and test it using a physical system. The algorithm demonstrated high accuracy and certain generalizability. The algorithm fails when pouring with big and tall source containers. It also fails with syrup and solid materials. Here “fail” refers to low accuracy. In the cases of failure, the algorithm cannot handle the way in which the material goes into the receiving container. The algorithm lacks a mechanism with which it can drastically adjust the motion according to how the material goes into the receiving container. The immediate future work is to add an outcome-driven component to the algorithm, so that it can actively adjust itself once it encounters unusual cases with the temporal progression of the outcome.
Figure 6.10. Evaluating minimum accurate amount. Actual-vs-target comparison of pouring 15mL and 20mL of water using the red cup.

15mL: $\mu_e=9.68\text{mL}$, $\sigma_e=7.96\text{mL}$

20mL: $\mu_e=2.83\text{mL}$, $\sigma_e=3.33\text{mL}$
Figure 6.11. Force sensor reading of a 1 lbf-weight taken during 300 seconds. The bottom subfigure shows the volume converted from force.
Figure 6.12. Actual-vs-target comparison of one human subject pouring water
Figure 6.13. Pairs of initial and target force in the training data and the experiments
Figure 6.14. Pairs of height and diameter in the training data and the experiments
Figure 6.15. Comparisons of normalized histograms of minimum static distances

Figure 6.16. Comparisons of normalized histograms of minimum DTW distances
Figure 6.17. Actual-vs-target comparison of pouring oil using the red cup.

cooking oil, $\mu_e=4.11\text{mL}$, $\sigma_e=4.80\text{mL}$
Figure 6.18. Actual-vs-target comparison of pouring syrup using the red cup.

syrup, $\mu_e=15.66\text{mL}$, $\sigma_e=3.43\text{mL}$
Figure 6.19. Pouring accuracy of liquids with different viscosity. x-axis plotted in logarithm.
CHAPTER 7

CONCLUSION AND FUTURE WORK

This dissertation focuses on generating trajectories for task-oriented object manipulation by learning from demonstrations. We covered two subjects in the thesis. The first subject is the data that are needed for model learning and the second subject is the trajectory generation approaches we have developed. In this chapter, we summarize the contents covered in this thesis, point out certain limitations in our approaches, and broadly discuss ideas relating to the advancement of the field.

7.1 Data

We reviewed 28 object-manipulation related datasets. We divided the datasets into two categories: cooking and activities of daily living (ADL). For each dataset, we reviewed the size, modalities, activities, annotations, and how well it fits the need for Learning from Demonstration (LfD). We summarized the similarity and differences of certain aspects among different set of datasets, including equipment used, annotation types, and temporal segmentations. We provided the links to all the datasets, list of years in which each dataset was published, list of size and type of each dataset, and the lists of shared annotated activities for the ADL and cooking datasets, respectively.

The existing related datasets we reviewed did not meet our need, and therefore we created our own datasets. We collected more than 1,500 trials of more than 30 types of motions, plus more than 1,500 trials of pouring alone. Our data include RGB and Depth vision, position...
and orientation of the manipulated objects, and force and torque sensed at the base of the tool tips. In detail, we explained the equipment setup, tool preparation, force calibration, synchronization, as well as output file formats. We also presented our data visualization tool. The dataset keeps expanding as we are still collecting new data.

7.2 Approaches

We first presented an approach that utilizes functional analysis and constrained optimization. We used functional analysis to extract motion harmonics from the motion data, and we used constrained optimization to find the optimal weights of the motion harmonics so that the resulting trajectory strike a balance between resembling the demonstrated motions and meeting user-defined constraints. The approach is novel in treating the motion data as samples from continuous sources, and using the motion harmonics as building blocks for new trajectories. Since the approach generates trajectories as a whole, it compromises flexibility. Any change to any part of the trajectory requires re-generation of the entire trajectory. This may be computational expensive. Also, in reality what has been executed cannot be altered, which contradicts the process of trajectory re-generation. The approach also requires aligning the data temporally before learning, which may alter the velocity profile of the data in ways we do not know. The development of this approach pushes us to seek and devise an approach that 1) allows keeping the data intact during learning, 2) allows point-by-point generation.

We identified recurrent neural network (RNN) to be our next candidate for trajectory generation, because it meets the two requirements we set for our new approach. We built a pouring trajectory generation system that includes three parts: the velocity generation model, the force estimation model, and the stopping model. We compared the resulting trajectories with the data and found out that the approach generalizes when the change
to the pouring situation is small, but fails to generalize when the changes to the situation becomes large.

The three-system design relies significantly on how well the learning goes. If any sub-system fails to learn successfully, the whole system might fail. Therefore, as the next step, we make the velocity generation system the sole component of the pouring system, with force read directly from a physical force sensor. This requires testing the system on a real physical system, which we made. The new system contains a single model to learn and therefore fewer components that might fail. The new system achieved high accuracy on the physical system and it also generalized to different source containers. The system still broke when pouring from certain source containers, but it showed robustness for pouring with various regularly-shaped source containers.

### 7.3 Future Work

Learning from Demonstration falls into the category of supervised learning. Despite the increasing popularity of reinforcement learning, the success of our pouring algorithm made us believe that LfD approaches still had potentials to exploit. Neural networks originated from psychology and physiology, and they may keep updating with the advancement of the two fields. The study of human motion, particularly how human use motion to reach goals and how human define task goals, from the viewpoint of biology, may provide insight to artificial intelligence. Specific mechanism of the interaction between motions and the real world now seems too complex to clearly specify, and which imposes great difficulty on learning a model that can be truly effectively deployed. Because of the extreme lack of knowledge about how the real world works, simulations have been put into use instead, and encouraging results have been achieved. However, simulation can never replace the real world, and therefore is in essence insufficient. Also, a huge amount of simulation is needed for learning, while humans
can learn successfully with a few examples. The problem is difficult but interesting at the same time. We eagerly look forward to the days of flexible task-fulfilling capability.
LIST OF REFERENCES


APPENDIX A

COPYRIGHT PERMISSIONS

The permission below is for the use of material in Chapter 2.

Big data

<table>
<thead>
<tr>
<th>Order detail ID:</th>
<th>71861594</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order License Id:</td>
<td>4558841265916</td>
</tr>
<tr>
<td>ISSN:</td>
<td>2167-6461</td>
</tr>
<tr>
<td>Publication Type:</td>
<td>Journal</td>
</tr>
<tr>
<td>Volume:</td>
<td></td>
</tr>
<tr>
<td>Issue:</td>
<td></td>
</tr>
<tr>
<td>Start page:</td>
<td></td>
</tr>
<tr>
<td>Publisher:</td>
<td>Mary Ann Liebert, Inc</td>
</tr>
<tr>
<td>Author/Editor:</td>
<td>Mary Ann Liebert, Inc.</td>
</tr>
</tbody>
</table>

Permission Status: √ Granted
Permission type: Republish or display content
Type of use: Thesis/Dissertation

Requestor type: Academic Institution
Format: Electronic
Portion: chapter/article
The requesting person/organization: University of South Florida
Title or numeric reference of the portion(s): Recent Data Sets on Object Manipulation: A Survey
Title of the article or chapter the portion is from:
Editor of portion(s): n/a
Author of portion(s): Yongqiang Huang
Volume of serial or monograph: n/a
Page range of portion: 2016
Publication date of portion: 2016
Rights for: Main product
Duration of use: Life of current edition
Creation of copies for the disabled: no
With minor editing privileges: no
For distribution to:
In the following language(s): Original language of publication
With incidental promotional use: no
Lifetime unit quantity of new product: Up to 499

Title: Robotic Motion Generation by Using Spatial-Temporal Patterns from Human Demonstrations
Institution name: University of South Florida
Expected presentation date: Mar 2019

142
The permission below is for the use of material in Chapter 4 and 5.

**Thesis / Dissertation Reuse**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.