# Project 2: Pouring Dynamics Estimation Using Recurrent Neural Network

Project TA: Juan Wilches, `jwilches@usf.edu`

## 1 Introduction

To be useful for our daily living tasks, robots need to be able to perform many different manipulations. Among them, pouring is the second most frequently executed motion (after pick-and-place) in cooking scenarios [1, 2]. Programming or controlling a robot to perform pouring water, we would usually need to model the water's dynamics so that the program or controller can predict the water's behavior. In pouring, it is how much and how fast water comes out of a pouring container at a certain rotation angle and rotation velocity.

Robotic pouring became a popular research topic recently. [3] uses a deep neural network to estimate liquid volume in a cup and a pouring PID controller, while [4] estimates liquid height from an RGB-D point cloud and apply a PID controller to control the pouring cup. [5] learns the pouring policy in simulation using reinforcement learning and tested the policy with an actual robot.

Like many other daily-living manipulation tasks, it is very difficult to build an accurate pouring dynamics model since it is difficult to model liquid dynamics. However, humans can pour water regularly and fairly accurately without explicitly modeling pouring dynamics. Humans learned to pour at their very early age by observing other people first and then practicing pouring along with other manipulation skills frequently. Therefore, we have developed a robot learning approach that learns pouring skills from human's demonstration [6]. The human demonstration data is from an openly available daily interactive manipulation dataset [7]. A comprehensive review of object manipulation datasets could be found at [8].

The manipulation learning uses Recurrent neural networks (RNN) that take an input and a hidden state from the last time step, and produce an output for that time step. The mechanism of RNN makes it inherently suitable for learning the behavior of a dynamic system [9, 10]. RNN has been very successful in processing languages and handwriting. For example, [11] generates English writing trajectories by predicting pen-tip locations. [12] applies a similar strategy to generate Chinese characters.

To perform simulation, the water behavior should be modeled during pouring so that the simulation system could produce an accurate simulation of pouring outcomes for various pouring velocities, pouring amounts, and different pouring containers. This project is mainly to design an RNN that can learn water behavior in pouring so that it could be as be used to predict the water behavior. The predicted water behavior could be used in a model-predicted control framework to generate accurate robotic pouring [13].

## 2 Project Goal

In this project, students will design a recurrent neural network to estimate the response to a sequence of manipulation actions. The motion in this project is pouring (represented with the angle and velocity of the pouring cup) and the response is the change of the amount of water in the pouring cup (represented with weight). The students are encouraged to read our previous work on how to model liquid behavior with RNN [13]. and model pouring behavior using RNN [6, 14, 15].

## 3 Data Set

The data set includes 688 motion sequences (executed by a pouring robot) and their corresponding weight measurements. The data have shape `[num_sequence, max_len, feature_dim]`. `max_len` is the maximum length of all sequences. Any sequence whose actual length is less than `max_len` is padded with zeros at the end for all features. The padded zeros are not actual collected data and therefore should not be included when you compute loss. The sampling rate for data collection was 60 Hz. Each motion sequence has seven feature dimensions, i.e. `feature_dim=7` which, for each time stamp of a motion sequence, are

[

| | |
|---|---|
| $\theta(t)$ | rotation angle at time t (degree) |
| $f(t)$ | weight at time t (lbf) |
| $f_{init}$ | weight before pouring (lbf) |
| $f_{target}$ | weight aimed to be poured in the receiving cup (lbf) |
| $h_{cup}$ | height of the pouring cup (mm) |
| $d_{cup}$ | diameter of the pouring cup (mm) |
| $\dot{\theta}(t)$ | velocity of the pouring cup (rad/s) |

]

Only $\theta(t)$, $\dot{\theta}(t)$, and f(t) change with time, the other four dimensions are constant throughout the entire sequence. You are suggested to plot the data for yourself to understand it better. The training data is a Numpy file with name '`Robot_Trials_DL.npy`' (stored at `https://drive.google.com/file/d/1hYNcadYeR5Vv5iTKotwTdDbjRxj6AKps/view?usp=sharing` which include all seven dimensions. You are expected to extract the weight $f(t)$ from the training data and use it as target. You are allowed to use whichever else dimension(s) as input data. You are supposed to divide the training data into training-, validation-, and testing-set, respectively, by yourselves. You are suggested to do any preprocessing you see fit, such as making the range of each dimension comparable. The data we provided does not include some samples we held out for testing. We will also use the samples generated from a cup, which is not in the training data, and thus test the generalization ability of the model. We withhold the testing data and will distribute it to you once we receive your trained model. The detailed data collection process could be found in [1].

## 4 Inputs and Outputs

You are free to determine the input to your network as long as you believe it is the best possible input you can think of to estimate $f(t)$. However, all possible inputs do not have the same effects on training. For example, not including $\theta(t)$ in the input will result in difficulty in training your network. The output of the network should have 1 dimension: $f(t)$.

## 5 Network Structure and Training

It is up to you to decide on the structure of the RNN. [6] has one solution as an example. You can download the paper at `https://ieeexplore.ieee.org/document/8206626/`. The loss function could be defined as the average squared difference per time stamp, or per sequence, or something else. You can start with building 4 layers, where each layer includes 16 LSTM cells. You can also use Gated Recurrent Unit (GRU) instead of LSTM. Dropout is recommended and its theoretical detail can be found at `https://arxiv.org/abs/1409.2329`. In short, you should regularize only the connections that do not convey time related information.

You are allowed to use Python and related machine learning library and Keras and TensorFlow.

## 6 Report

You are required to write a technical report that shows what *you* know and have done. To that end, you can try describing your model in detail and the rationale behind your decision, showing how it learns, and explaining your results. You are suggested to use numbers to be concise and precise and use figures to facilitate the explanation. You are encouraged to present any insights or findings regarding the problem. You are expected to write the report in the format of a technical paper. You could find an acceptable paper template at `http://ras.papercept.net/conferences/support/files/ieeeconf.zip` (latex) or `http://ras.papercept.net/conferences/support/files/ieeeconf_letter.dot` (Word) Your paper should contains the following sections (or similar sections):

- Abstract
- Introduction
- Data and Preprocessing
- Methodology
- Evaluation and Results
- Discussion

- Reference

## 7 Due Dates

| | |
|---|---|
| Network design and evaluation | March 21 |
| Report | March 28 |

## 8 Grading Rubric

| | |
|---|---|
| 40% | Network Design & Training: architecture design and modifications, loss function, training options, tuning. |
| 10% | Results and Analysis: Present your results clearly and analyze your results. Provide an explaining about the good and problems in your results. |
| 30% | Your model's performance on the testing data. |
| 20% | Report: Structure of the paper should be logically sound, The description of the approach should be clear. The figures should illustrate the network and help in presenting the results. The results should be thorough and clearly presented. Results should be analyzed and discussed through graphs and figures. The paper should be easy to read. |

## References

[1] David Paulius, Yongqiang Huang, Roger Milton, William D Buchanan, Jeanine Sam, and Yu Sun. Functional object-oriented network for manipulation learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2655–2662. IEEE, 2016.

[2] David Paulius, Ahmad B Jelodar, and Yu Sun. Functional object-oriented network: Construction & expansion. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[3] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2629–2636, May 2017.

[4] Chau Do and Wolfram Burgard. Accurate pouring with an autonomous robot using an RGB-D camera. *CoRR*, abs/1810.03303, 2018.

[5] Chau Do, Camilo Gordillo, and Wolfram Burgard. Learning to pour using deep deterministic policy gradients. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[6] Yongqiang Huang and Yu Sun. Learning to pour. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7005–7010. IEEE, 2017.

[7] Yongqiang Huang and Yu Sun. A dataset of daily interactive manipulation. *International Journal of Robotics Research*, 2019.

[8] Yongqiang Huang, Matteo Bianchi, Minas Liarokapis, and Sun Yu. Recent data sets on object manipulation: A survey. *Big Data*, 4(4):197–216, December 2016.

[9] Min Han, Zhiwei Shi, and Wei Wang. Modeling dynamic system by recurrent neural network with state variables. In *Advances in Neural Networks - ISNN 2004*, 2004.

[10] Adam P. Trischler and Gabriele M.T. DEleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *Neural Networks*, 80:67 – 78, 2016.

[11] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[12] Xu-Yao Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, and Yoshua Bengio. Drawing and recognizing chinese characters with recurrent neural network. *CoRR*, abs/1606.06539, 2016.

[13] Tianze Chen, Yongqiang Huang, and Yu Sun. Accurate pouring using model predictive control enabled by recurrent neural network. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[14] Juan Wilches, Yongqiang Huang, and Yu Sun. Generalizing learned manipulation skills in practice. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9322–9328, 2020.

[15] Yongqiang Huang, Juan Wilches, and Yu Sun. Robot gaining accurate pouring skills through self-supervised learning and generalization. *Robotics and Autonomous Systems*, 136:103692, 2021.