Haoying Zhou U48070579

Master Thesis Proposal

Imitation Learning From Human Motions With Dynamic Movement Primitives

I. Problem Formalization

My idea is to imitate some actions given some datasets extracted from real human motions; for instance, grasp a bottle and hold it up. To avoid tuning parameters manually and worrying about instabilities, I will use dynamic movement primitives as the control strategy.

The basic equations will be shown in “theory and background” part with specific meanings of every parameter.

The input will be vectors which include coordinates, angles and corresponding time. The output will be trajectory points and angles with time in $\mathbb{R}^{4\times4}$ form ($\begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$). However, I’ll transfer them into some force signals, then pass it to the plant or device. That will be mainly the robotic internal solver’s work.

Compared to the past work, I’ll use the human motions’ datasets rather than assisting the robotic arm to accomplish some specific motions and collecting the dataset for later usage.

The difficulty of the project will be included but limited as following:
1. Mirror symmetry when extracting data from human motion
2. solver simplification or development
3. multiple degrees of freedom

II. Theory and background

1 Dynamic Movement Primitives

Dynamic movement primitives (DMPs) are a method of trajectory control or planning from Stefan Schaal’s lab. They were presented way back in 2003[1], and then updated in 2013 by Auke Ijspeert[2]. This work was motivated by the desire to find a way to represent complex motor actions that can be flexibly adjusted without manual parameter tuning or having to worry about instability.

Formally, if dynamic system one obeys $\dot{x} = f(x)$, and system two yields $\dot{y} = g(y)$, then the existence of an orientation preserving homeomorphism h: $[x \quad \dot{x}] \overset{h}{\rightarrow} [y \quad \dot{y}]$ and $[x \quad \dot{x}] \overset{h^{-1}}{\longleftarrow} [y \quad \dot{y}]$ proves topological equivalence.

And DMPs retain their qualitative behavior if translated and if scaled in both space and time.

For this project, I'll just use discrete DMP model.

1.1 Positions with DMPs

The basic equations for positions are shown following[2]:

$$\ddot{y} = \tau^2[\alpha_y(\beta_y(g-y) - \dot{y}) + f]$$
$$\dot{x} = \tau(-\alpha_x x)$$

where

$$f(x,g) = \frac{\sum_{i=1}^{N}\psi_i\omega_i}{\sum_{i=1}^{N}\psi_i}x(g-y_0)$$
$$\psi_i = e^{-h_i(x-c_i)^2}$$

which is a Gaussian function centered at c_i and h_i is the variance

According to the Schaal's paper's conclusion, it can derive the solution of weights[3]:

$$\omega_i = \frac{s^T\psi_i f_d}{s^T\psi_i s}$$

where

$$s = \begin{bmatrix} x_{t_0}(g-y_0) \\ \vdots \\ x_{t_N}(g-y_0) \end{bmatrix}, \qquad \psi_i = \begin{bmatrix} \psi_i(t_0) & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \psi_i(t_n) \end{bmatrix}$$
$$f_d = \ddot{y}_d - \alpha_y(\beta_y(g-y_d) - \dot{y}_d)$$

The parameters' meanings are: y is our system state as well as the output, g is the goal state, and α and β are gain terms, where y_0 is the initial position of the system, ω_i is a weighting for a given basis function ψ_i. x is the diminishing term, f is the force term. And the footnote d means corresponding parameters are from or calculated from the desired trajectory.

τ is the temporal term, for accuracy, its default is 1 and generally will not change.

And for properly activating Gaussian functions, the centers of Gaussians and the variances of Gaussians have following relationship[4]:

$$h_i = \frac{number\ of\ basis\ functions^{\frac{3}{2}}}{\alpha_x \cdot c_i}$$

1.2 Orientations with DMPs

The orientation imitation and position imitation are extremely similar. And the basic equations for angles or rotation matrix are[5]:

$$\tau\dot{\eta} = \alpha_z\left[\beta_z \log\left(R_g R^T\right) - \eta\right] + f_0(x)$$
$$\tau\dot{R} = [\eta]_\times R$$
$$\dot{x} = \tau(-\alpha_x x)$$

where[6]

$$[\eta]_\times = \begin{bmatrix} 0 & -\eta_z & \eta_y \\ \eta_z & 0 & -\eta_x \\ -\eta_y & \eta_x & 0 \end{bmatrix}$$
$$f_0(x) = D_0 \frac{\sum_{i=1}^{N} \psi_i(x)\omega_i}{\sum_{i=1}^{N} \psi_i\,(x)} x$$
$$D_0 = diag(\log\left(R_g R^T\right))$$

f_0 and corresponding ψ_i, ω_is are just similar to position's, the only difference is the dimension.

Also[7],

$$\log(R) = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, & R = I \\ \omega = \theta n, & otherwise \end{cases}$$

where

$$\theta = \arccos\left(\frac{trace(R) - 1}{2}\right), \qquad n = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \ (R \epsilon \mathbb{R}^{3\times 3})$$

The parameters' meanings are: R_g denotes the goal orientation, η is our system state as well as the scaled angular velocity, R is the output, and α and β are gain terms, ω_i is a weighting for a given basis function ψ_i. x is the diminishing term, f is the force term.

τ is the temporal term, for accuracy, its default is 1 and generally will not change.

2. Locally Weighted Regression

2.1 Introduction

Locally weighted regression (LWR)[8] is a class of function approximation techniques, where a prediction is done by using an approximated local model around the current point of interest

It can overcome the disadvantage of global method --- sometimes no parameter can provide a sufficient good approximation.

The computational costs for some tasks are extremely high. Local algorithm can avoid the high computational costs.

2.2 Basic algorithm

Generally, in an optimization problem, the question can be denoted as:

$$y = f(x) + \varepsilon$$

where $f(x)$ is the input, y is the output and ε is the noise.
The basic cost of LWR is defined as:

$$\sum_{i=1}^{n} \omega_i(x_q)(y_i - x_i\beta_q)^2$$

where x_q is the query point, which means the point where we want the prediction, it can also be named as the point of interest.
The whole algorithm is[9]:
Given:
(1) query point x_q
(2) a set of training points
Prediction:
(1) Build matrix $X = (\widehat{x_1}, \widehat{x_2}, \dots, \widehat{x_n})^T$ where $\hat{x_i} = \begin{bmatrix} {x_i}^T \\ 1 \end{bmatrix}$
(2) Build vector $y = (y_1, y_2, \dots, y_n)^T$
(3) Compute diagonal weight matrix W:

$$\omega_{i,i} = e^{-\frac{1}{2}(x_i - x_q)^T D(x_i - x_q)}$$

(4) Calculated Regression coefficient:

$$\beta_q = (X^T W X)^{-1} X^T W_y$$

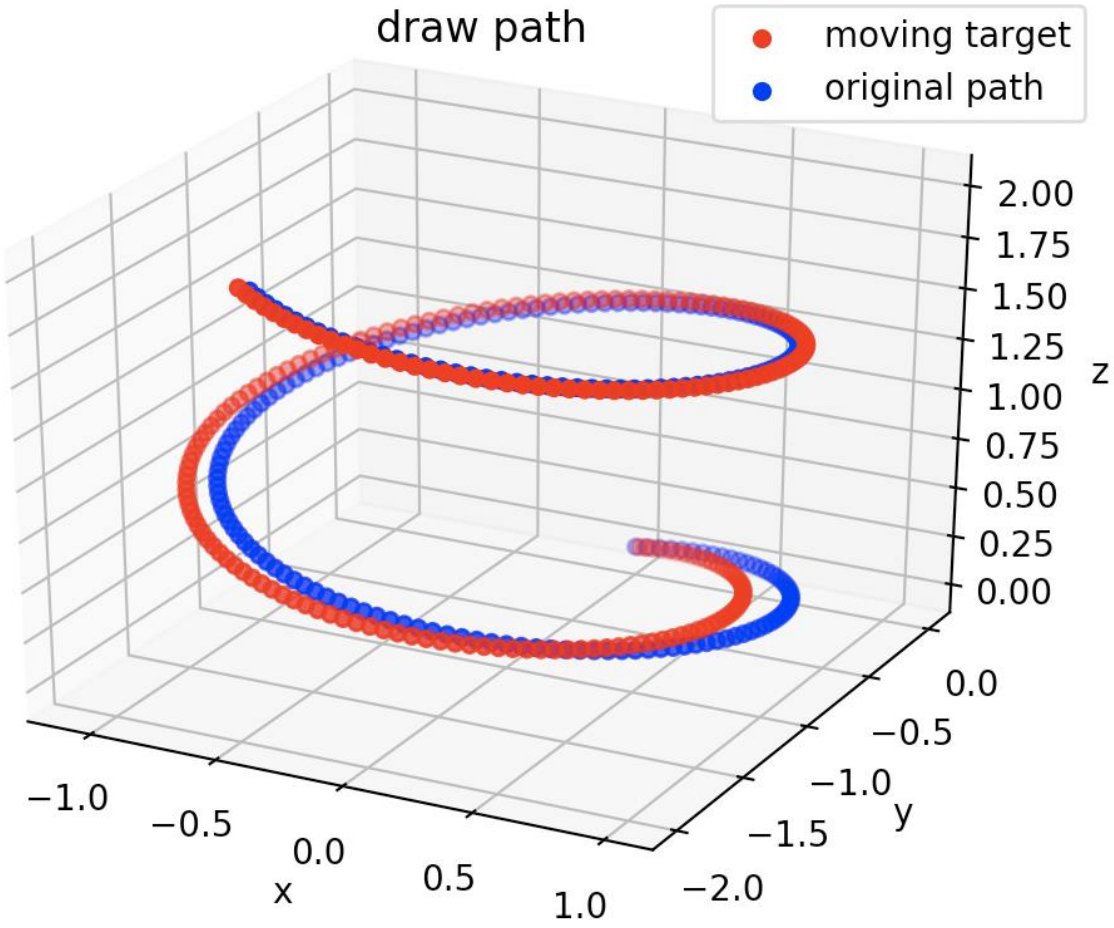
(5) Predict

$$\widehat{y_q} = \begin{bmatrix} {x_q}^T & 1 \end{bmatrix} \beta_q$$

III. Completed work

1 Coordinate imitation

I've successful completed coordinate imitation utilizing package "pydmp"[10] with some modification in Python:



2 Working on orientation imitation

I've accomplished transfer input angles to rotation matrix and currently substitute them into above equations.

IV. Timeline

Date	Purposes
2019.07.31	Accomplish the position and orientation imitation simulation for at least one trial trajectory
2019.08.31	Make the simulation work with multiple degrees of freedom
2019.09.31	Solve the mirror symmetry problem and get trajectories from real human motions
2019.10.31	Make the simulation work in vrep or corresponding software with the trajectory extracted from human motions
2019.11.30	Make the real robotic arm-hand system work with given human motions
2019.12.31	Accomplish all experiments (no later than 2020.1.29)
2020.3.10	First draft of thesis
2020.3.26	Final draft handed in
2020.4.9	Trial Defense
2020.4.15	Final Defense (date may change)

V. References

1. Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert; “Control, Planning, Learning, and Imitation with Dynamic Movement Primitives”, *IROS*, pp. 1-21, 2003
2. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, “Dynamic movement primitives: Learning attractor models for motor behaviors”, *Neural Computations*, vol. 25, no. 2, pp. 328-373, 2013
3. Stefan Schaal, Christopher G. Atkeson and Sethu Vijayakumar, “Scalable Techniques from Nonparametric Statistics for Real Time Robot Learning”, Applied Intelligence, vol.17, Issue 1, pp. 49-60, 2002
4.https://github.com/studywolf/pydmps/blob/master/pydmps/dmp_discrete.py
5. Aleš Ude, Bojan Nemec, Tadej Petrič and Jun Morimoto, “Orientation in Cartesian Space Dynamic Movement Primitives”, *IEEE*, DOI: 10.1109/ICRA.2014.6907291, 2014
6. R. M. Murray, Z. Li, and S. S. Sastry, “A Mathematical Introduction to Robotic Manipulation”, Boca Raton, New York: CRC Press, 1994
7. N. Ayache, “Artificial Vision for Mobile Robotics: Stereo Vision and Multisensory Perception”, Cambridge, Mass: MIT Press, 19915.
8. http://abr.ijs.si/upload/1523530180-Generalization.pdf , “Generalization, Locally Weighted Regression, Gaussian Process Regression”, Andrej Gams, Jožef Stefan Institute Ljubljana, Slovenia, 2018
9. Peter Englert, “~~Locally Weighted Learning”,~~ ALS 2012
10. https://github.com/studywolf/pydmps

For further background of dmp, you may go through:
https://studywolf.wordpress.com/2013/11/16/dynamic-movement-primitives-part-1-the-basics/