

Dynamic Movement Primitives

Abhishek Padalkar

Hochschule Bonn-Rhein-Sieg

abhishek.padalkar@smail.inf.h-brs.com

December 18, 2017

Trajectories for robots

- Robots need trajectories for manipulation and navigation.
- Trajectory is set of pairs $\{\text{position, velocity}\}$ at each point in time leading from initial position to the goal.
- Methods for generating trajectories :
 - Hand-crafting
 - Sampling based motion planning (e.g. OMPL)
 - Search based
 - Potential field
 - Learning

Model based vs Model free trajectory generation

- Model based[1]
 - Model based trajectory generations needs a fairly accurate model of robot as well as environment.
 - This framework uses internal simulations for validation of the generated trajectory. (e.g collision, reachability)
- Model free[1]
 - This method does not need any model.
 - A policy is learned and used for trajectory generation.
 - External framework is used for actual control of the robot.
- Learning a policy is often easier than learning accurate forward models.[1]
- Hence model-free policy is more widely used than model-based methods.[1]

Methods for policy representation

- Linear Policies:

- Linear controllers are the most simple time independent representation.
- Policies are represented as linear combination of known basis functions. [1]
- Specifying the basis functions by hand is typically a difficult task, and, hence, the application of linear controllers is limited to problems where appropriate basis functions are known. [1]

- Radial Basis Functions Networks:

- A typical nonlinear time independent policy representation is a radial basis function (RBF) network.
- RBF networks are powerful policy representations, they are also difficult to learn due to the high number of nonlinear parameters. [1]

- Dynamic Movement Primitives: Dynamic Movement Primitives (DMPs) are the most widely used policy representation in robotics. [1]

Advantages of Dynamic Motion Primitives

- It is model free learning approach.
- DMP learns trajectories in terms of attractor landscape of non-linear autonomous differential equations.
- Convergence of differential equation is guaranteed.
- Any arbitrary trajectory can be learned in end-effector space as well as in joint space.
- Here learning is linear regression, so it does not need large dataset. One trajectory is sufficient ideally.
- Trajectories can be scaled in space as well as in time.
- These learned motion primitives can be initialized anywhere in the attractor space which shows flexibility.

Advantages of Dynamic Motion Primitives

- Trajectory evolves as robot actually moves along the trajectory.
- Hence on-line modifications in trajectory are possible.
- These modifications can be realized by introducing vanishing coupling terms in the differential equations. (e.g. potential field around a obstacles [6])
- Perturbations can be handled robustly due to above reasons.
- Re-planning is not needed unless an event causing major disturbance in the environment occurs.

Available framework and resources

- Well established DMP theory [7] [3] [2]
- Methods for sequencing DMPs [5] [4]
- Work related to reinforcement learning and learning from demonstration using DMPs.
- DMP package in ROS.


Problem addressed

- This work will focus on combining simple dynamic motion primitives for accomplishing a complex task.
- A framework for knowledge base representation will be devised which will allow combination of dynamic motion primitives with minimum effort and maximum flexibility.
- Methods for modifying already learned trajectories will be explored.

- Enabling Jenny to make coffee by combining already learned primitives
- Navigating ROPOD platform along a corner regardless of right or left turn

Possible approach

- Learn trajectories in end-effector space.
- Design framework for knowledge base representation.
- Use external frameworks to realize use cases, in our case :
 - Use current setup (MoveIt!) on Jenny to realize above mentioned use cases.
 - MoveIt! provides reachability check and self-collision avoidance features.
- Navigation in case of ROPOD mobile platform will not suffer from self-collision and reachability problems.
- Robust navigation is possible using current software stack (move_base and direct_base_controller), once trajectory is generated.

- [1] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.
- [2] Auke J Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1547–1554, 2003.
- [3] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.
- [4] Rudolf Lioutikov, Oliver Kroemer, Guilherme Maeda, and Jan Peters. Learning manipulation by sequencing motor primitives with a two-armed robot. In *Intelligent Autonomous Systems 13*, pages 1601–1611. Springer, 2016.
- [5] Bojan Nemec and Aleš Ude. Action sequencing using dynamic movement primitives. *Robotica*, 30(5):837–846, 2012.
- [6] Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoid Robots, 2008*, 

Humanoids 2008. 8th IEEE-RAS International Conference on, pages 91–98. IEEE, 2008.

- [7] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.