Imitation Learning
using
Dynamic Movement Primitives

USC
UNIVERSITY
OF SOUTHERN
CALIFORNIA

Peter Pastor Sampedro
Computational Learning and Motor Control Lab
University of Southern California

Willow Garage

Intern Presentation
December 10th, 2009

Outline

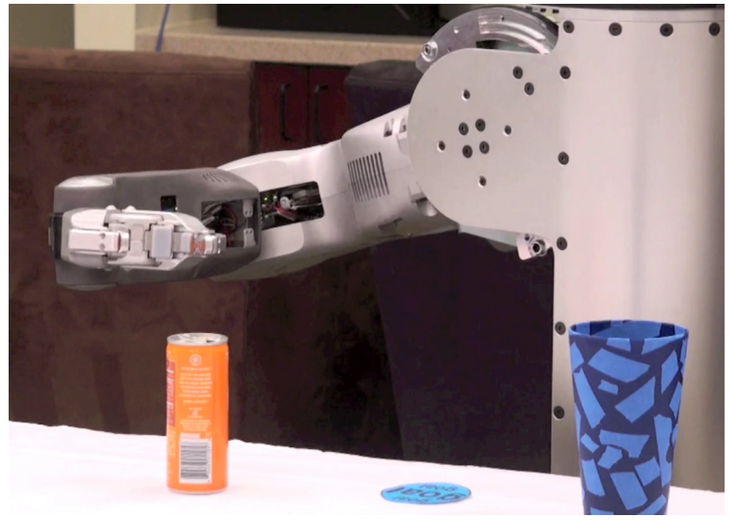
1) Motivation

2) Dynamic Movement Primitives

3) System overview

Motivation

Robots will do certain tasks over and over again.
For example: reaching a door handle, grasping a cup,...



For these tasks, it would be nice to have pre-computed solutions that can be retrieved from a library.

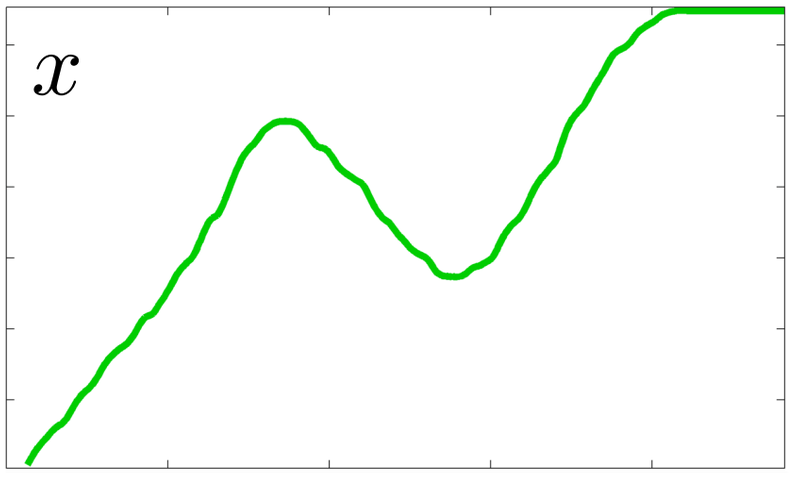
Feature wish list:

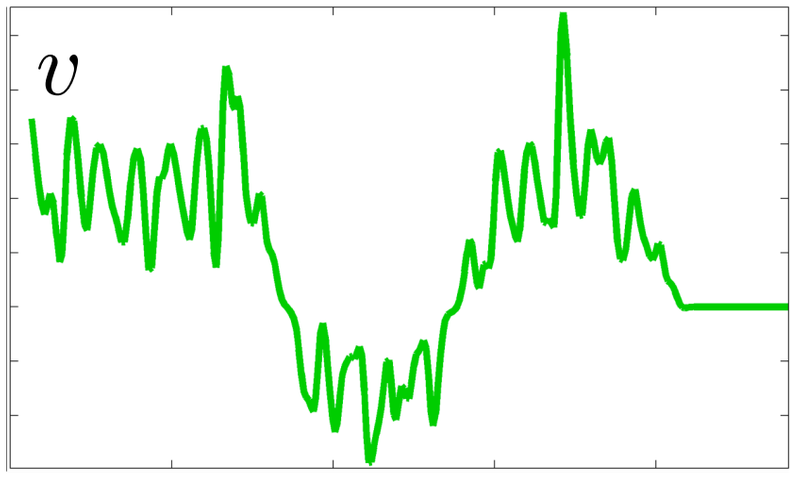
1) Ability to generalize to new situations
   Adapting individual movement primitives to new targets
   Sequencing of primitives to accomplish complex movements.

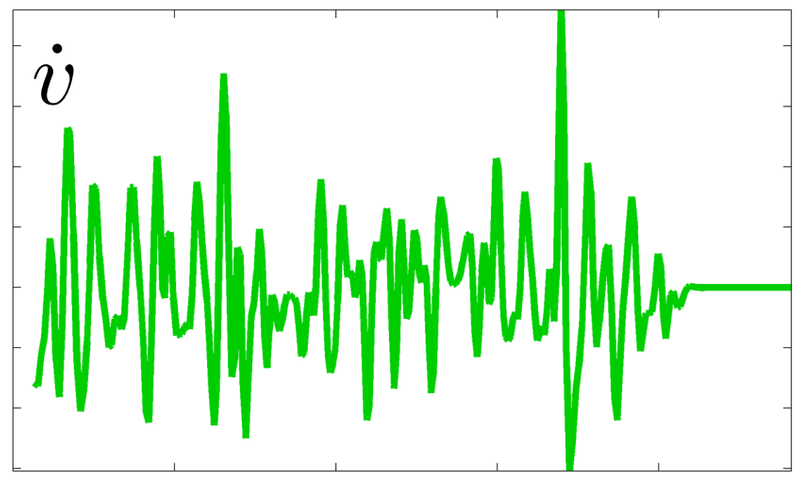
2) Robust against perturbation
   Ability to online adapt to external signals.

3) Easy programming through (one-shot) learning from demonstration

Dynamic Movement Primitives (DMPs)







$x, v, \dot{v}$

canonical system

$$\tau \dot{s} = -\alpha s$$

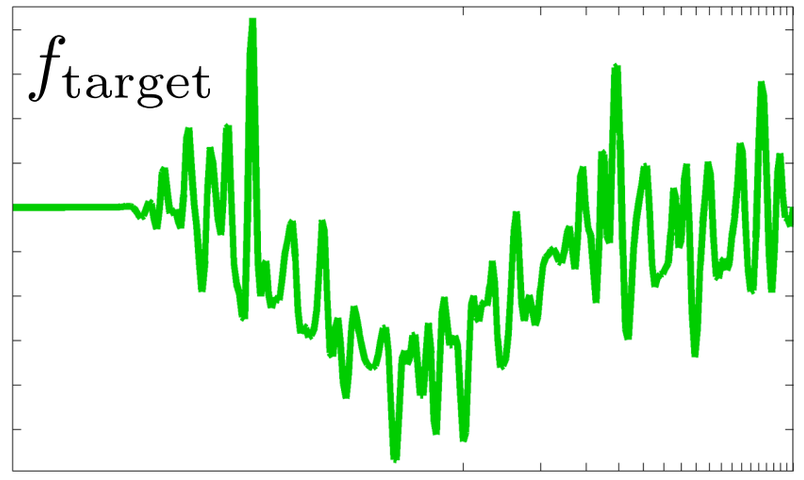
nonlinear function

$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

transformation system

$$\tau \dot{v} = K(g - x) - D\, v + (g - x_0) f$$
$$\tau \dot{x} = v$$

Dynamic Movement Primitives (DMPs)



$x, v, \dot{v}$

canonical system

$$\tau \dot{s} = -\alpha s$$

→

nonlinear function

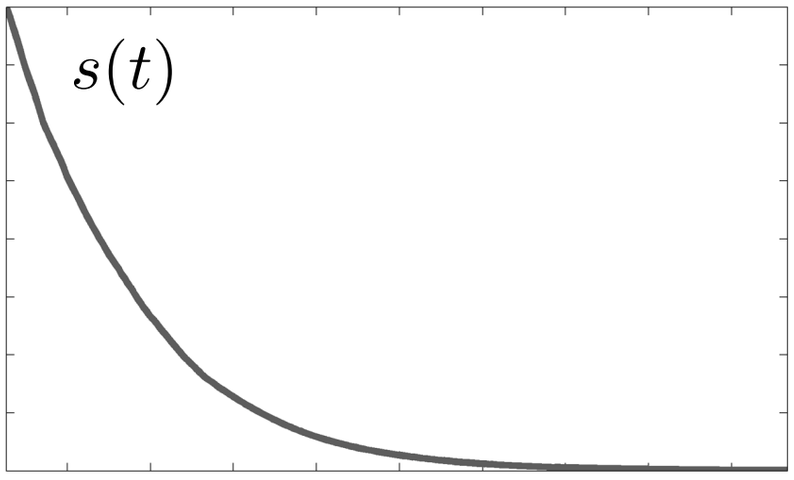
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

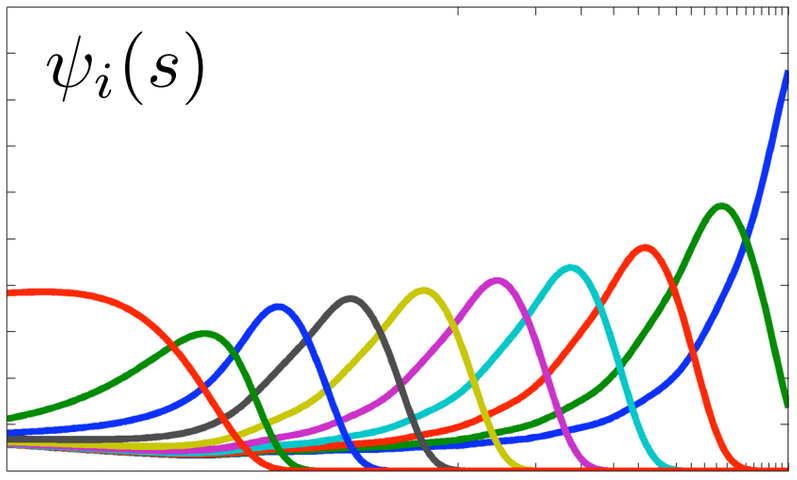
←

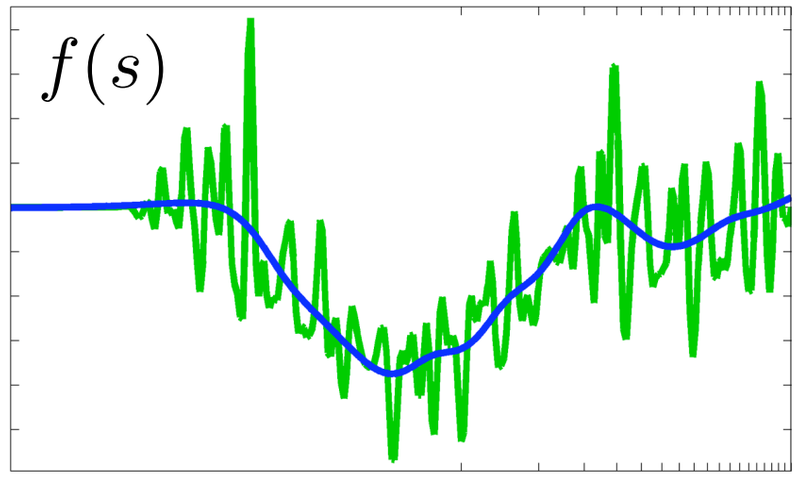
transformation system

$$\tau \dot{v} = K(g - x) - D\, v + (g - x_0) f$$
$$\tau \dot{x} = v$$

Dynamic Movement Primitives (DMPs)







w

Locally weighted learning

$x, v, \dot{v}$

canonical system

$$\tau \dot{s} = -\alpha s$$

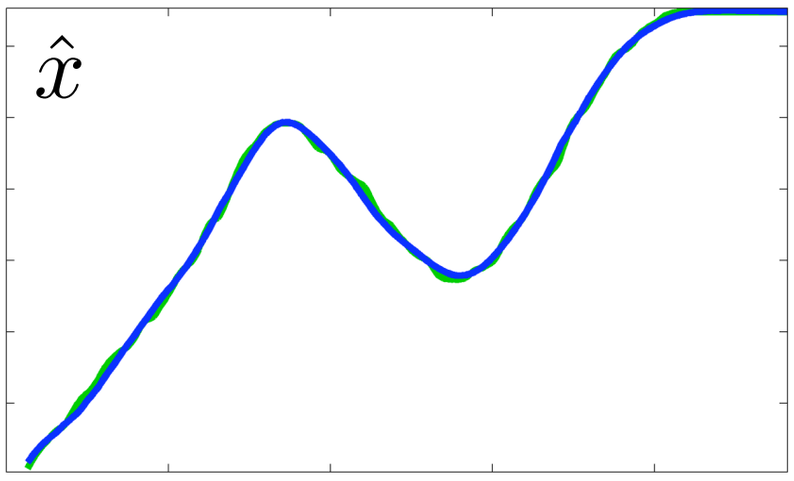
nonlinear function

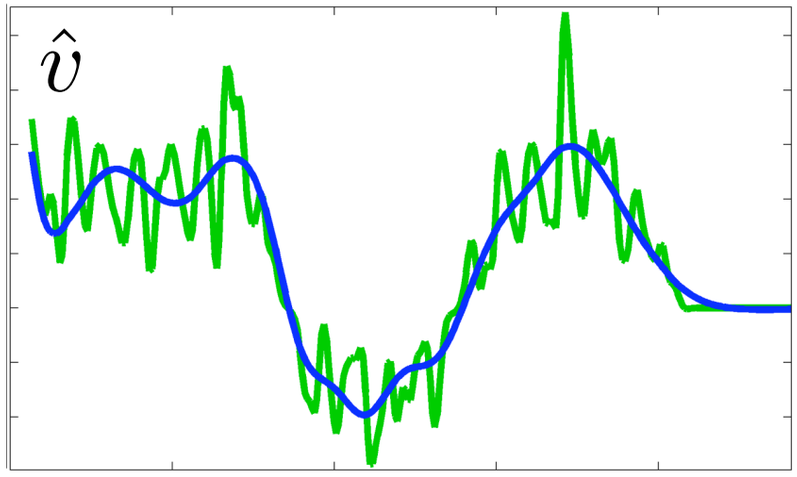
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

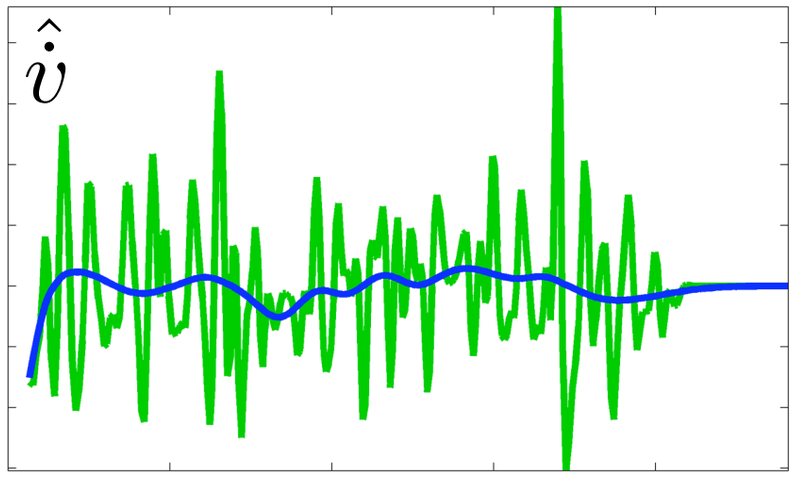
transformation system

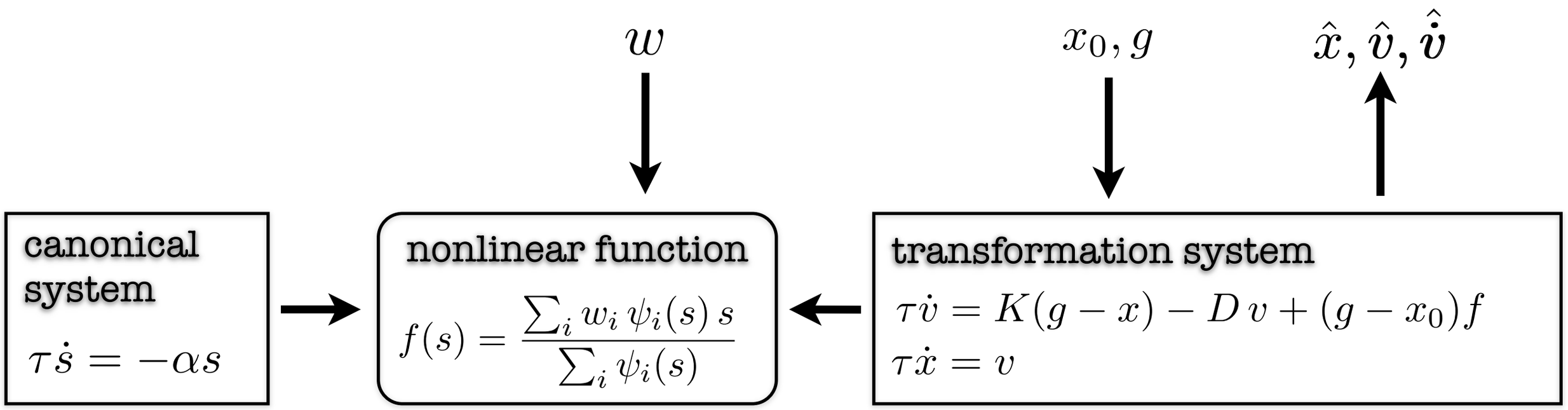
$$\tau \dot{v} = K(g - x) - D\, v + (g - x_0) f$$
$$\tau \dot{x} = v$$

Dynamic Movement Primitives (DMPs)

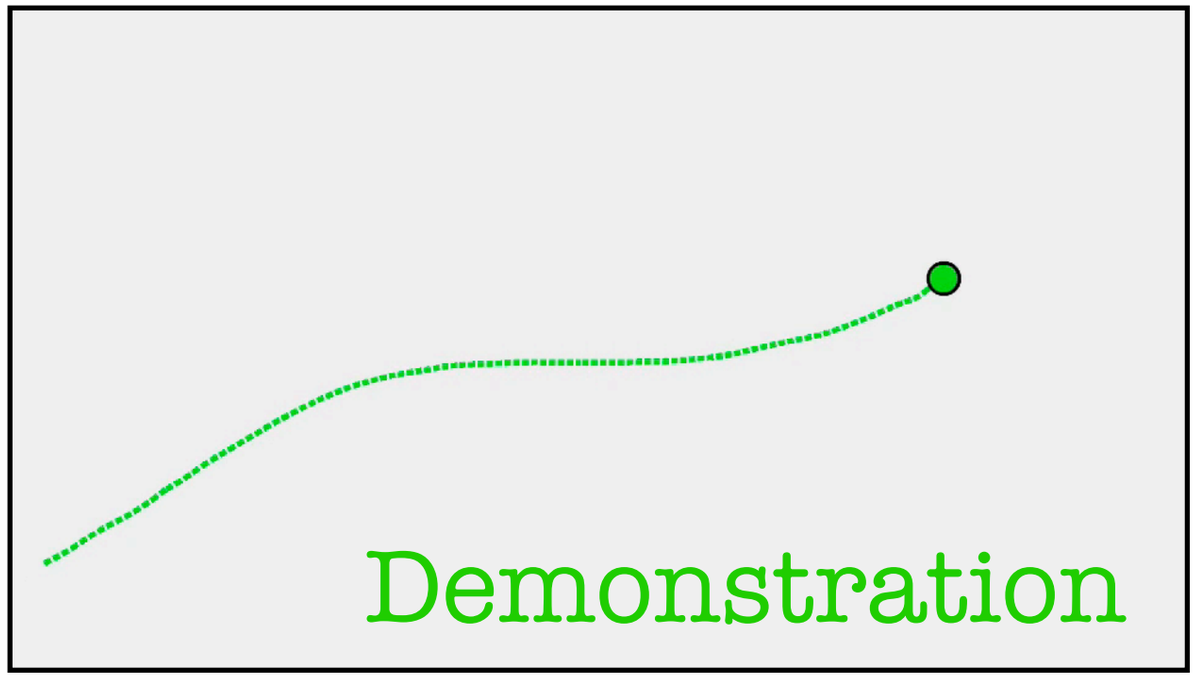


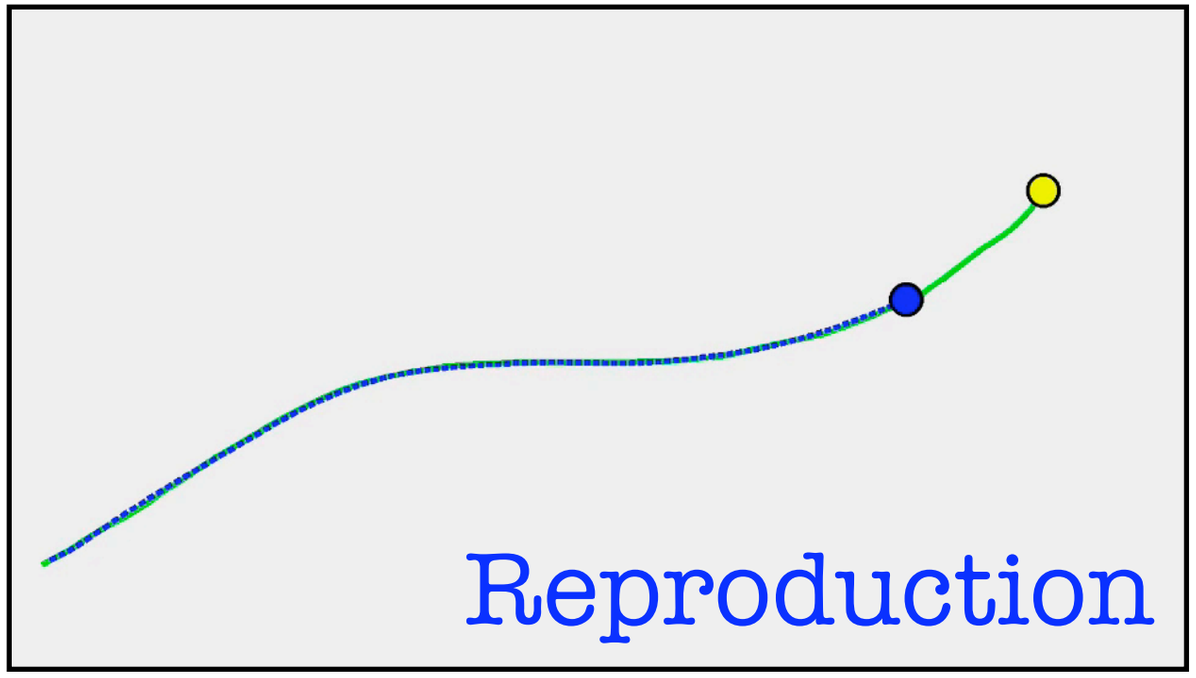






Robustness against perturbation





canonical system

$$\tau \dot{s} = -\alpha s$$

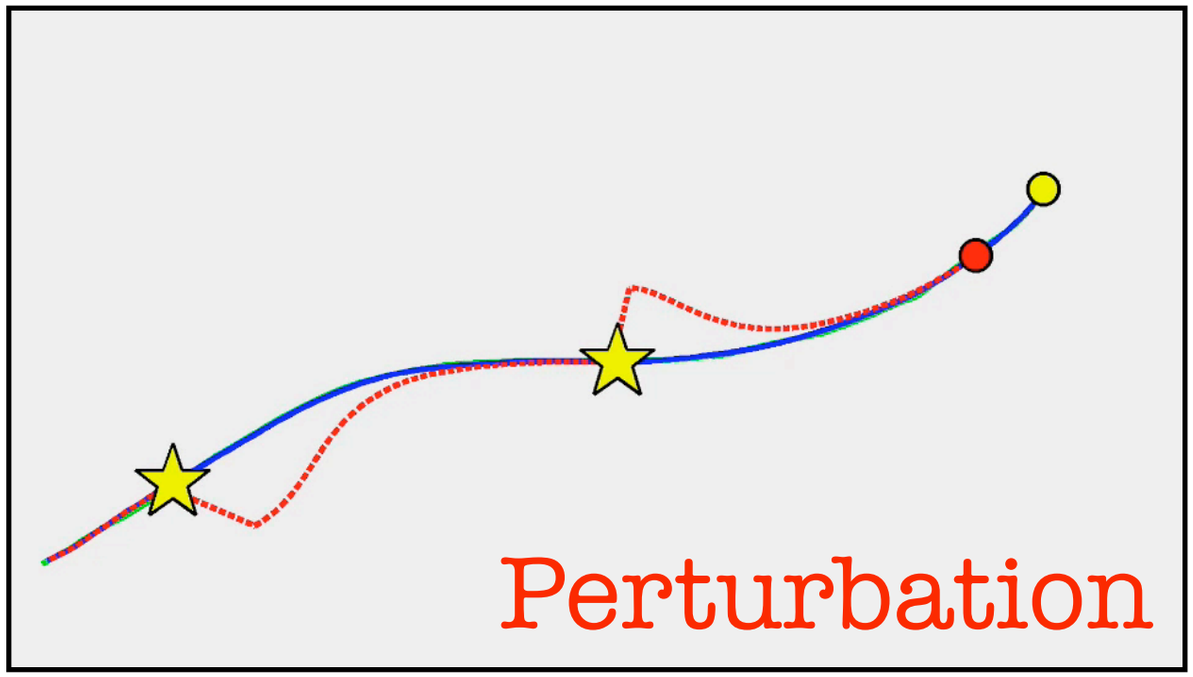
nonlinear function

$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

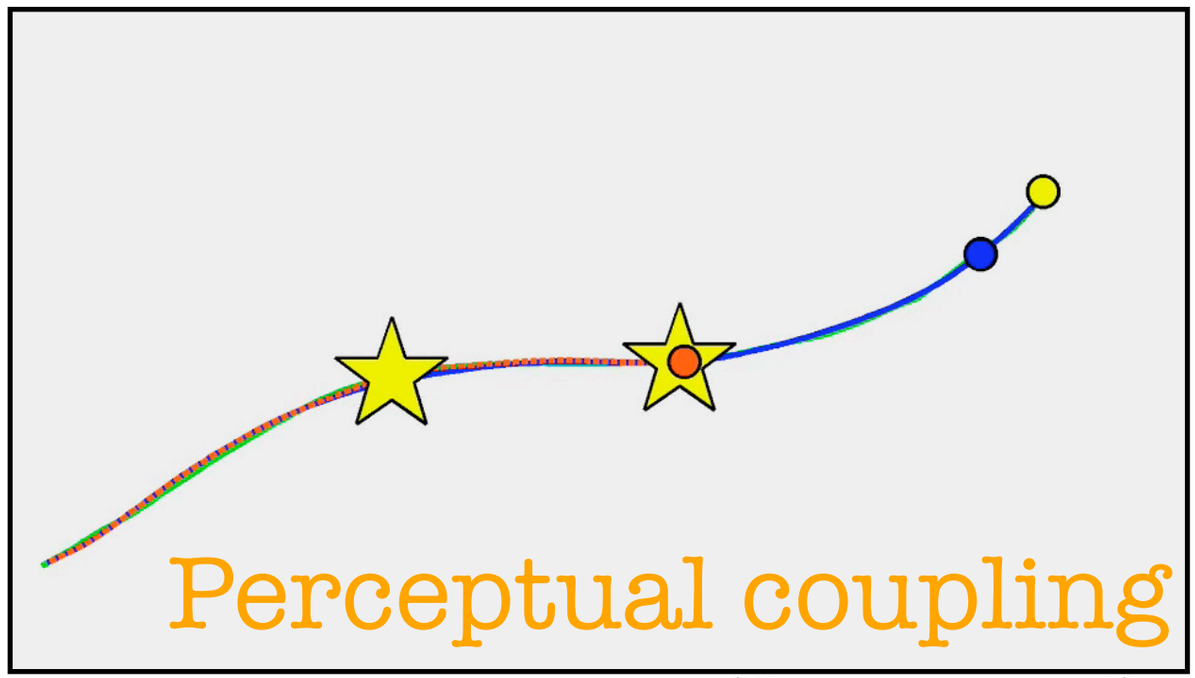
transformation system

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f$$
$$\tau \dot{x} = v$$

Robustness against perturbation



Perturbation



Perceptual coupling

(Ijspeert et. al)

canonical system

$$\tau \dot{s} = \frac{-\alpha s}{1+\alpha_s |\tilde{x} - x|}$$

nonlinear function

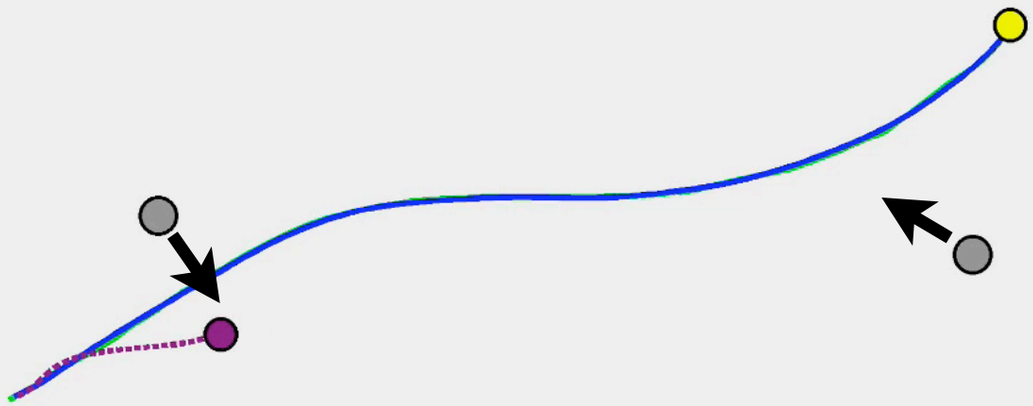
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

transformation system

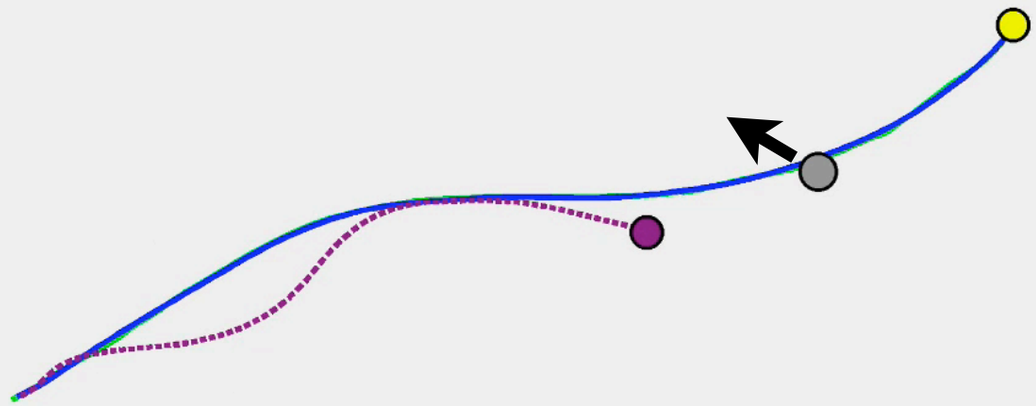
$$\tau \dot{v} = K(g - x) - Dv + (g - x_0) f$$
$$\tau \dot{x} = v + \alpha_x |\tilde{x} - x|$$

Robustness against perturbation

Obstacle avoidance



Obstacle avoidance



canonical system

$$\tau \dot{s} = -\alpha s$$

nonlinear function

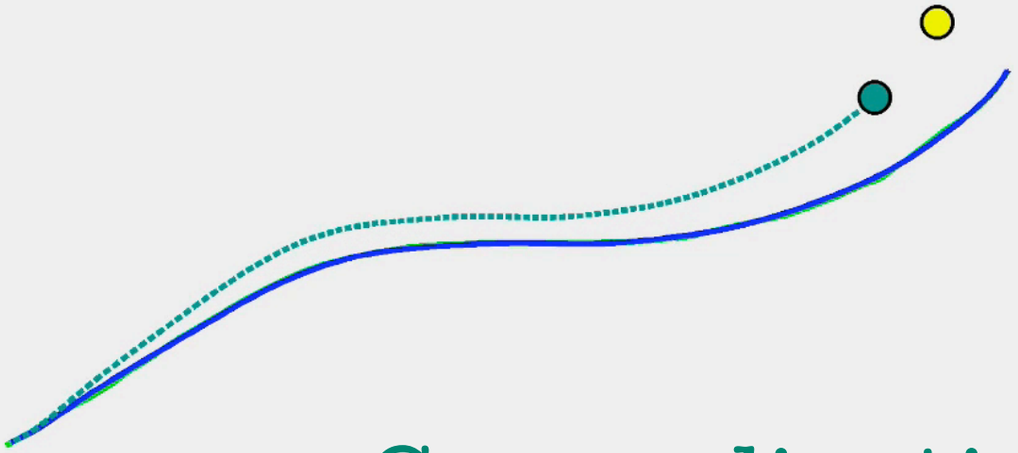
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

transformation system

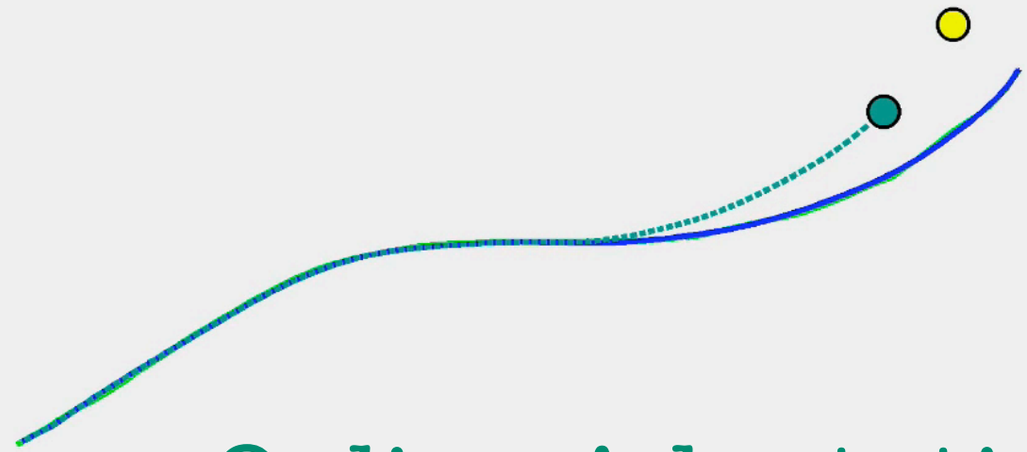
$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f + p(x, v)$$
$$\tau \dot{x} = v$$

Hoffmann, H., Pastor, P., Park, D.-H., Schaal, S. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance, ICRA2009.

Movement generalization



Generalization



Online Adaptation

canonical system

$$\tau \dot{s} = -\alpha s$$

nonlinear function

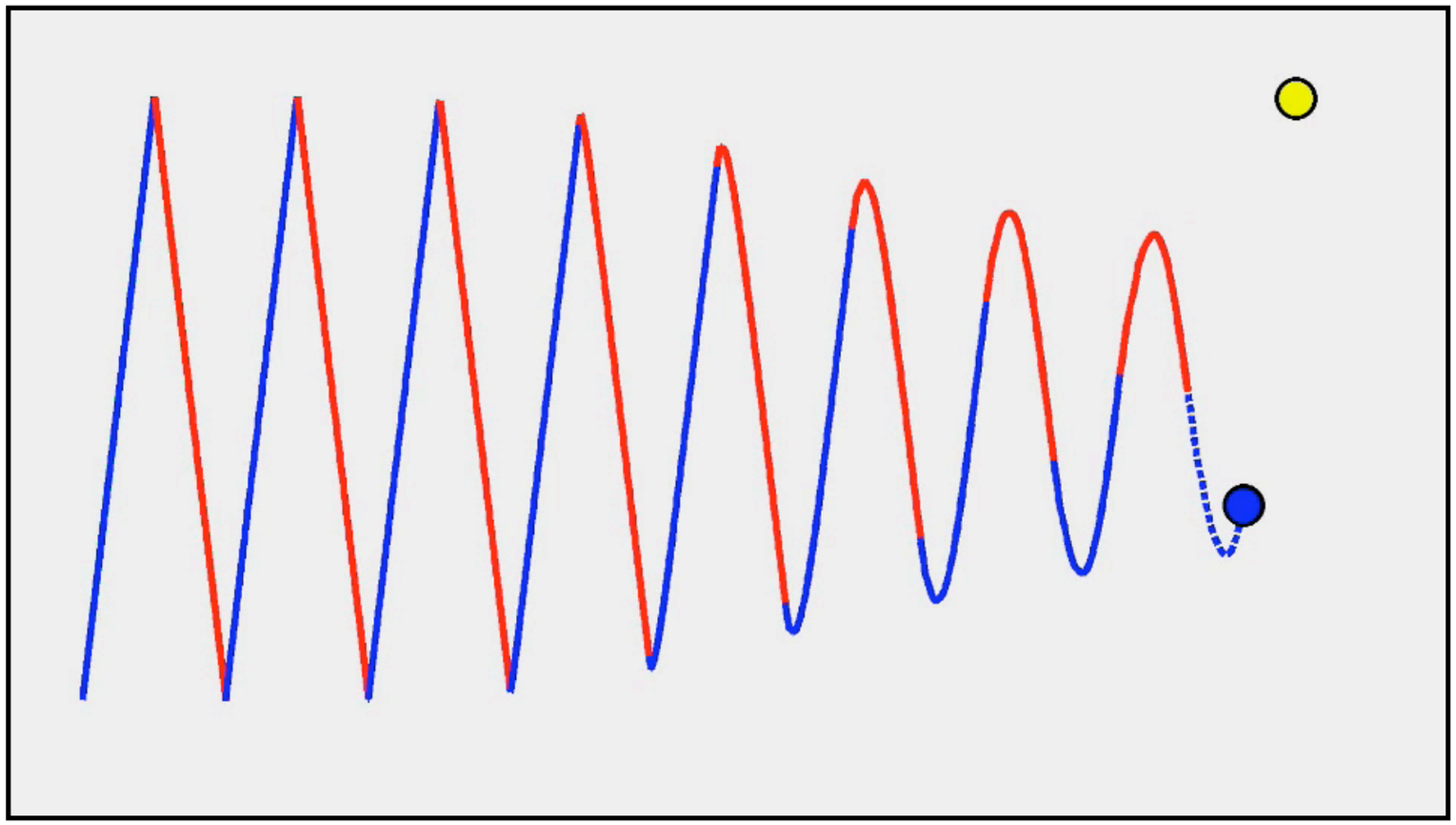
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

transformation system

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f$$

$$\tau \dot{x} = v$$

Movement sequencing



canonical system

$$\tau \dot{s} = -\alpha s$$

nonlinear function

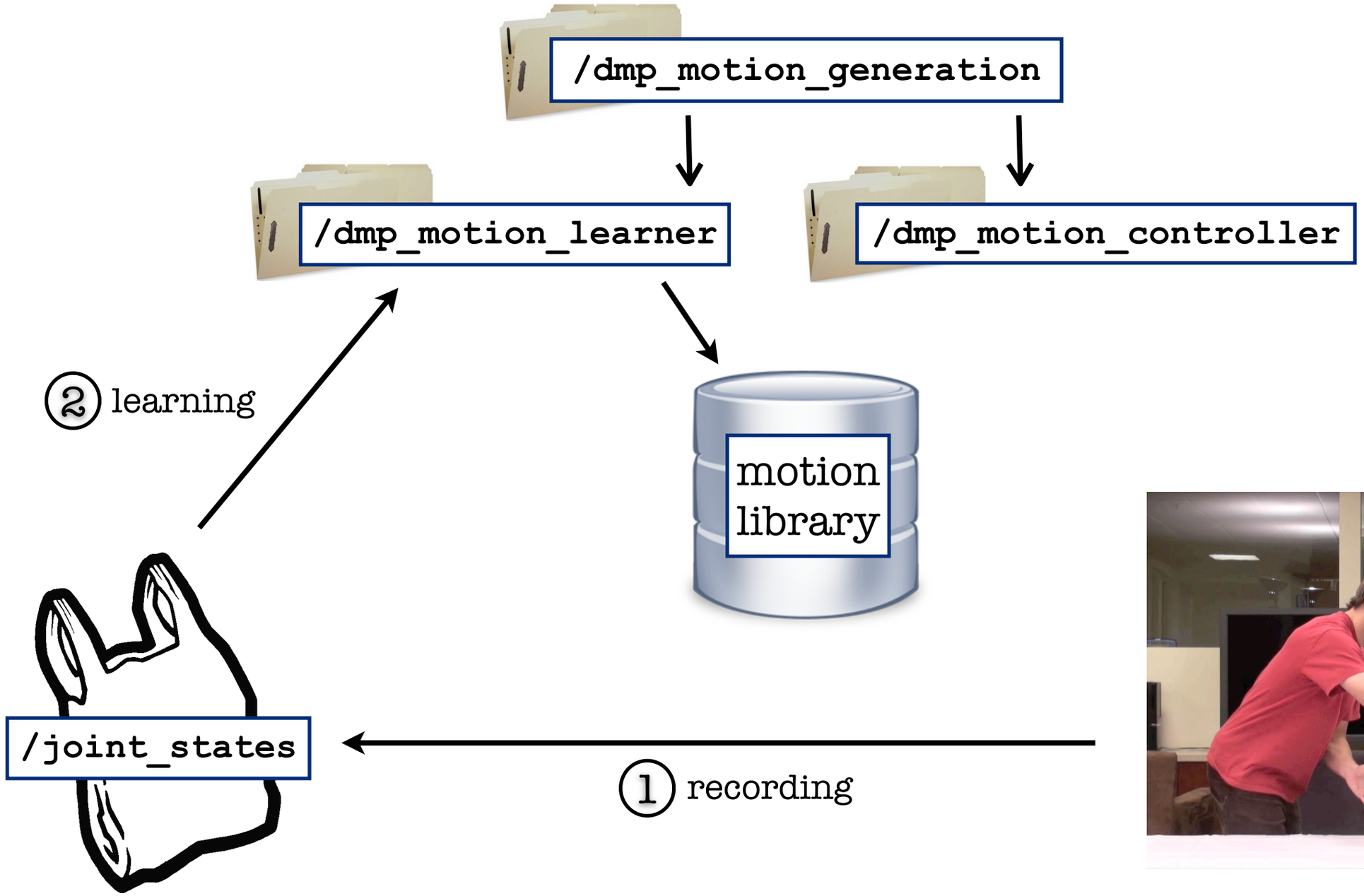
$$f(s) = \frac{\sum_i w_i \, \psi_i(s) \, s}{\sum_i \psi_i(s)}$$

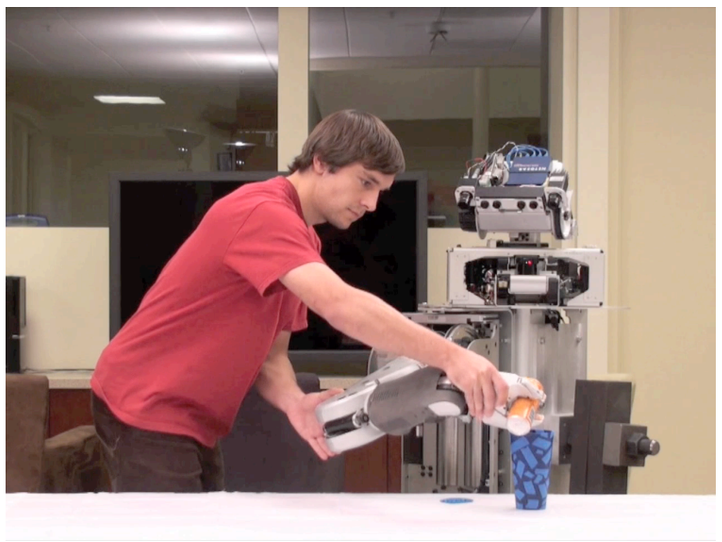
transformation system

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f$$

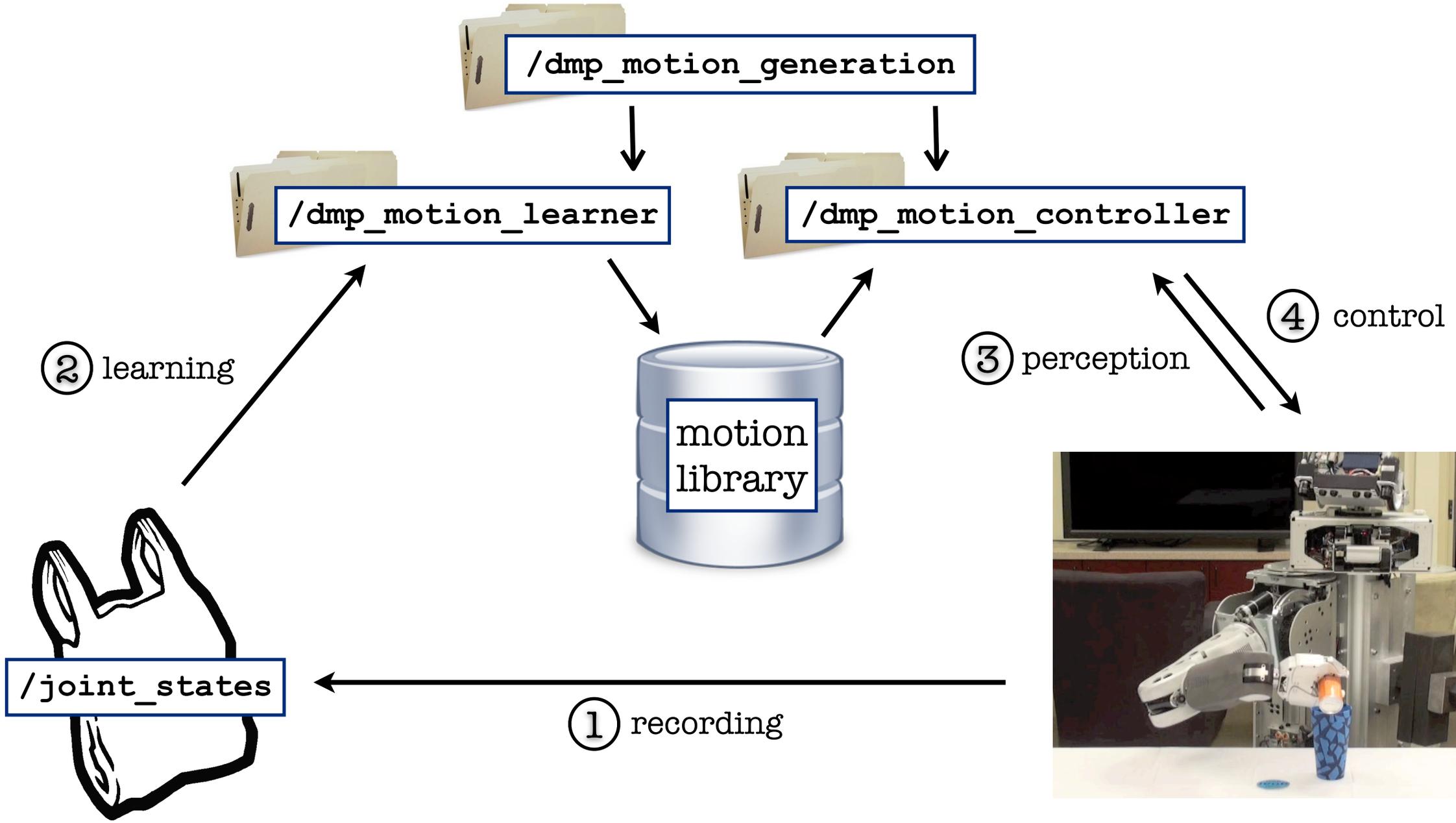
$$\tau \dot{x} = v$$

System overview





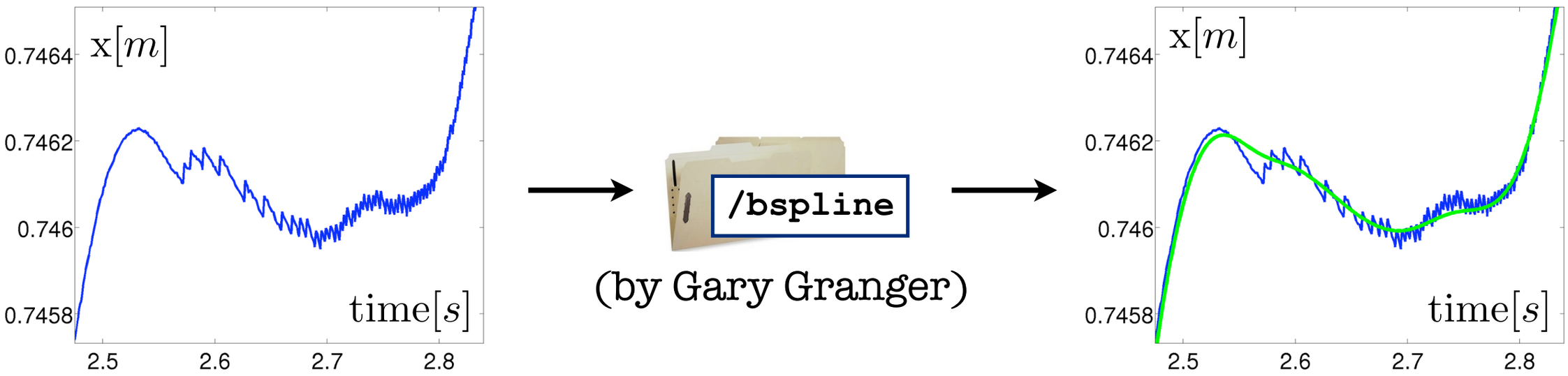
System overview



① recording

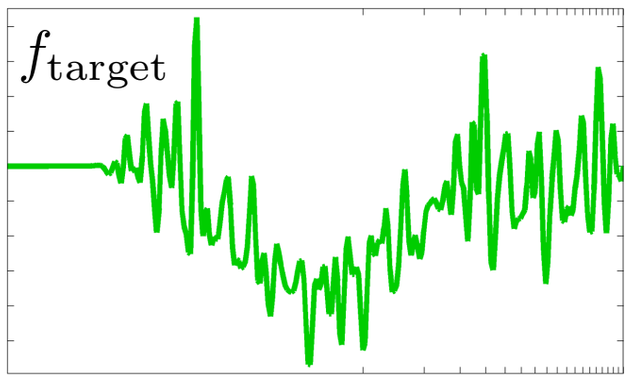
```
$ rosrecord /joint_states
```

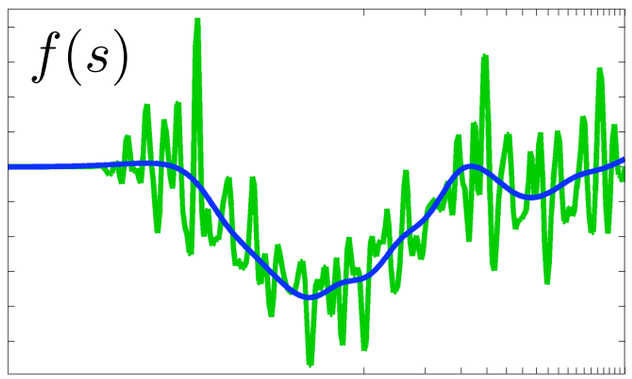
Problems:
1) variance on **dt**
2) **realtime_publisher** does not quite publish at 1000Hz



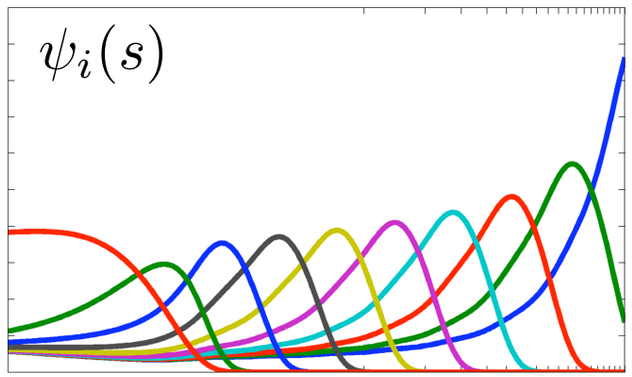
② learning

Locally Weighted Projection Regression (LWPR)





(by Stefan Klanke, Sethu Vijayakumar, and Stefan Schaal)



LWPR algorithm is implemented in plain ANSI C, with wrappers and bindings for C++, Matlab/Octave, and Python (using Numpy).

3 perception

/wge100_camera

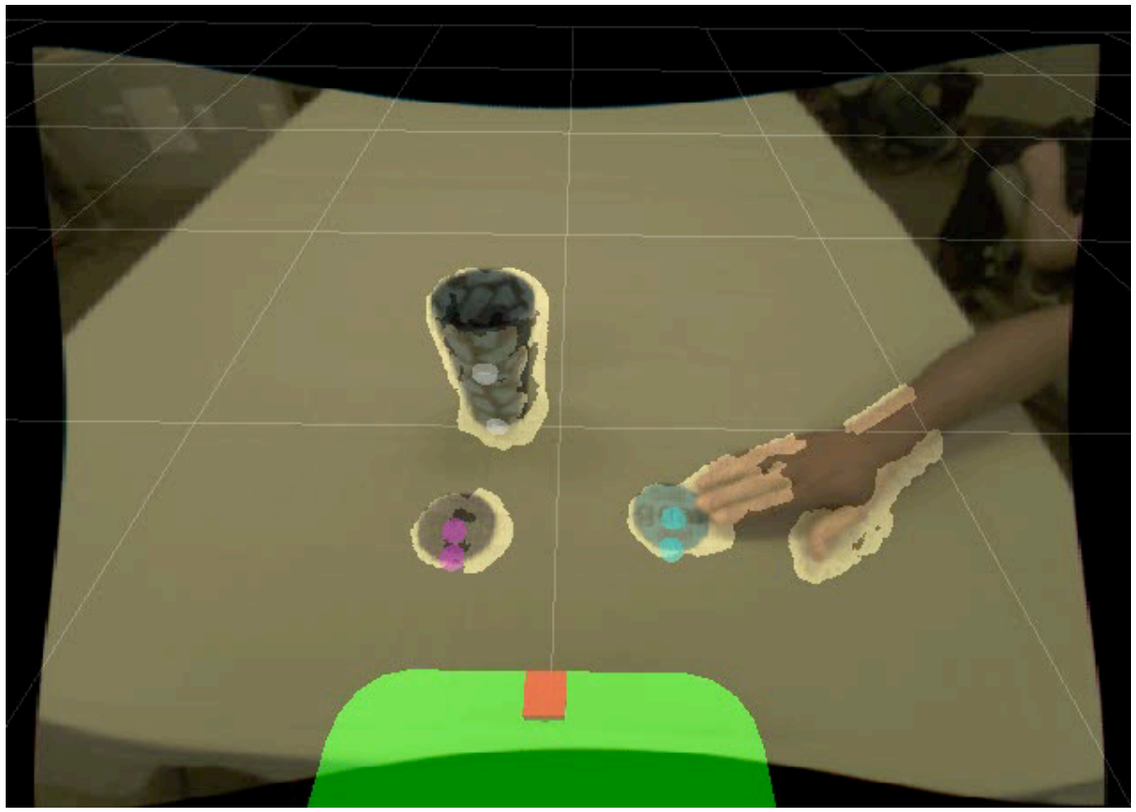
+

/stereo_image_proc

3D point cloud

corresponding image coordinates

time stamps



camera_offsetter_node

cloud_filter_node

1) compute closest match in image coordinates

2) get 3D projection from the point cloud

3) filter and compute velocities

perception_interface_node

/cmvision

color blob tracker (thanks to Nate)

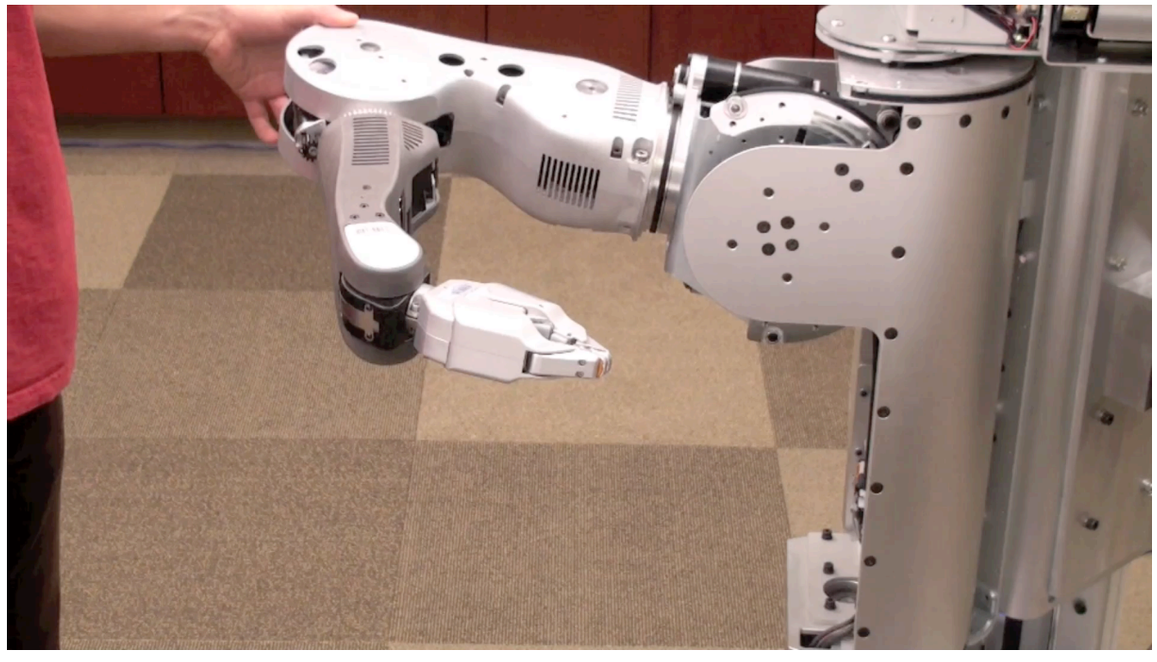
image coordinates of color blobs

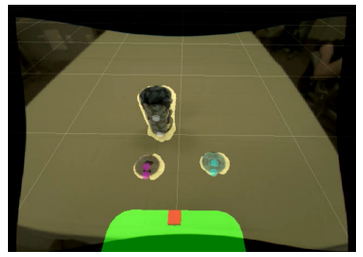
time stamps

④ control

Problems:
1) no control of the position of the elbow.

2) reproduction of demonstrated movements may run into singularities.





DMP ↓ ↑ id

position ↓ ↑ color

`r_arm_dmp_controller`

$\downarrow \boldsymbol{x}_\mathrm{d}, \dot{\boldsymbol{x}}_\mathrm{d}$ (desired end-effector pose and twist)

`r_arm_cartesian_pose_twist_controller`

$\downarrow \dot{\boldsymbol{x}}_\mathrm{r} = \dot{\boldsymbol{x}}_\mathrm{d} + (\boldsymbol{x}_\mathrm{d} - \boldsymbol{x})K_\mathrm{p} + (\dot{\boldsymbol{x}}_\mathrm{d} - \dot{\boldsymbol{x}})K_\mathrm{d}$

`r_arm_cartesian_wrench_controller`

$\downarrow \boldsymbol{q}_\mathrm{r} = \boldsymbol{J}^\mathrm{T}\boldsymbol{x}_\mathrm{r}$

τ

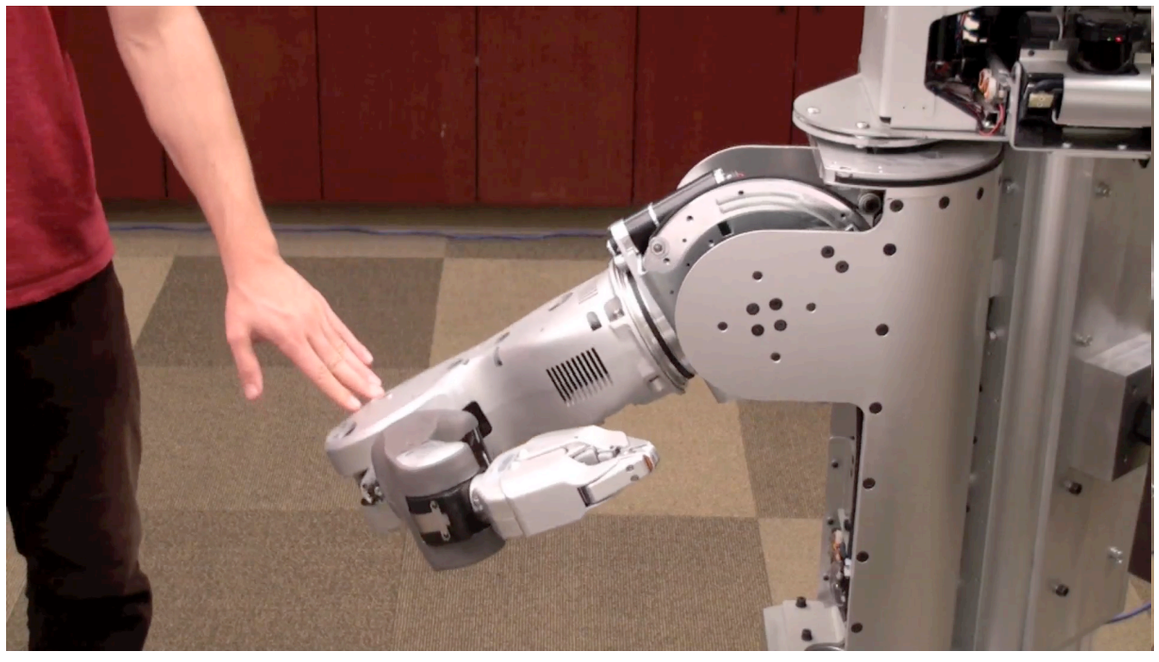
`r_arm_effort_controller`

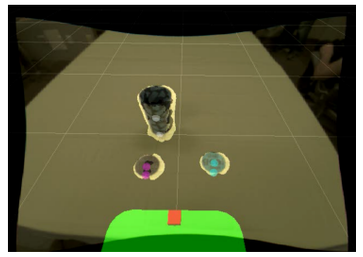
④ control

Problems:

1) no control of the position of the elbow. ✔

2) reproduction of demonstrated movements may run into singularities. ✔





DMP ⇅ id

position ⇅ color

`r_arm_dmp_controller`

$\boldsymbol{x}_\mathrm{d}, \dot{\boldsymbol{x}}_\mathrm{d}$ (desired end-effector pose and twist)

$\boldsymbol{q}_\mathrm{d}$ (desired joint angle configuration)

`r_arm_cartesian_twist_ik`

$$\dot{\boldsymbol{x}}_\mathrm{r} = \dot{\boldsymbol{x}}_\mathrm{d} + (\boldsymbol{x}_\mathrm{d} - \boldsymbol{x})K_\mathrm{p} + (\dot{\boldsymbol{x}}_\mathrm{d} - \dot{\boldsymbol{x}})K_\mathrm{d}$$

$$\dot{\boldsymbol{q}}_\mathrm{r} = \boldsymbol{J}^\dagger \dot{\boldsymbol{x}}_\mathrm{r} + (\boldsymbol{I} - \boldsymbol{J}^\dagger \boldsymbol{J})(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q})$$

τ

`r_arm_effort_controller`

Thank you !

Sachin for the support with this kind of research.

Jeremy for replacing broken hard discs.

Nate for answering my “stupid questions”.

Eric for the Jacobians.

Scott for Thanksgiving dinner.

Steffi for the psychological support.

Eitan for the parties up in the city.

Wim for answering ALL my “quick questions”.

Tully for TF + left over beers.

Curt, Derek, and Dallas for being my late night buddies.

Josh for rviz.

Matei, Brian, and Josh2 for being my soccer buddies.

Tim for rxplay + letting me win playing pool.

Kevin for the BBQ.

Ethan for pronouncing my name correctly (Peeeta).

Guenter for the lectures in control.

Vijay, btw is the robot calibrated ?

Stu for the real_time publisher.

Melonee for giving me slack with my intern doc :)

Ken for the bike competition.

Blaise for wge100_camera.

Everybody else, for the good time.