CS294-40 Learning for Robotics and Control Lecture 24 - 11/25/2008

Learning to Walk in 20 Minutes-Techake, Zhang, Seung

Lecturer: Pieter Abbeel *Scribe: Jared Wood*

1 Dynamic Walking

Dynamic walking: iff ground projection of center of mass leaves the convex hull of the ground contact points during some portion of the walking cycle.

Dynamic walking is challenging because bipeds can only control the trajectory of their center of mass through the unilateral, intermittent, uncertain force contacts with the ground. Accurate modeling is difficult, so learning/fine tuning on the real robot is important.

Assume there's always one foot/leg in contact with the gound. There are 3 degrees of freedom (DOF) for orientation and 6 internal DOF, resulting in a total of 9 DOF. There are 4 control inputs, so the robot is underactuated.

The continuous-time control problem is to find some $u = \pi(\hat{q}, \dot{\hat{q}})$, for the dynamics $\ddot{q} = f(q, \dot{q}, u, d)$. Alternatively the robot could be modeled in discrete-time with a Poincare return map when the left foot touches the ground. With this approach the transition probability is

$$P(\hat{x}_{n+1} = x' | \hat{x}_n = x, \pi)$$

where $x = \{q, \dot{q}\} \backslash \theta_{roll}$. The reward function per step is

$$R(x_n) = -\frac{1}{2} \parallel x_n - x* \parallel^2$$

where $x*$ is from the gait down a 0.03 radians slope. The policy/control is

$$u = \pi_w(\hat{x}) = \sum_i w_i \phi_i(\hat{x})$$

which is a weighted average of the features of the state estimate. At the nth step, the weights are updated as $w_n \longleftarrow z$ where $z \sim N(0, \Sigma)$ and

$$\begin{aligned}
\delta(n) &= R(\hat{x}_n) + \gamma \hat{V}(\hat{x}_{n+1}) - \hat{V}(\hat{x}_n) \\
e_i(n) &= \gamma e_i(n-1) + b_i(n) z_i(n) \\
\Delta w_i(n) &= \eta_w \delta(n) e_i(n) \\
\Delta v_i(n) &= -\eta_v \delta(n) \psi_i(\hat{x}_n)
\end{aligned}$$

Here, η_w and η_v are learning rates; $b_i(n)$ is boolean and equals 1 if $\phi_i(\hat{x}) > 0$ any time during the step; and $\hat{V} = \sum_i v_i \psi_i(\hat{x})$. Notice that another set of features is used for $\hat{V}$. Note that w should not be too drastically updated at each step because this will cause poor algorithm performance.

In practice the dimensionality is reduced by decomposeing roll/pitch. For roll, $\pi_{roll}(\hat{x}) \in R$, where there are two servos per foot and both feet are identically actuated. The policy can then have the form $\pi : (\theta_{roll}, \dot{\theta}_{roll}) \rightarrow R$. Figure 1 shows how a passive walker will not achieve the desired roll rate, whereas Figure 2 shows how after learning, the desired roll rate is obtained.

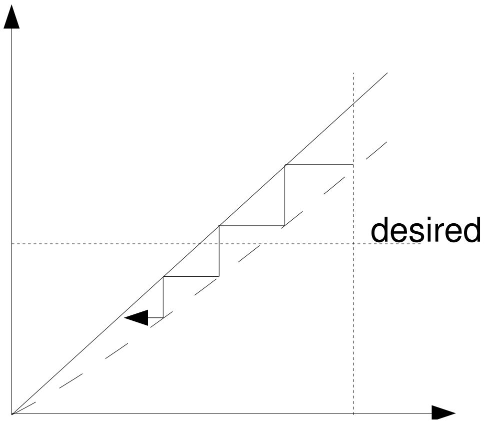

Figure 1: Return map for passive walker.

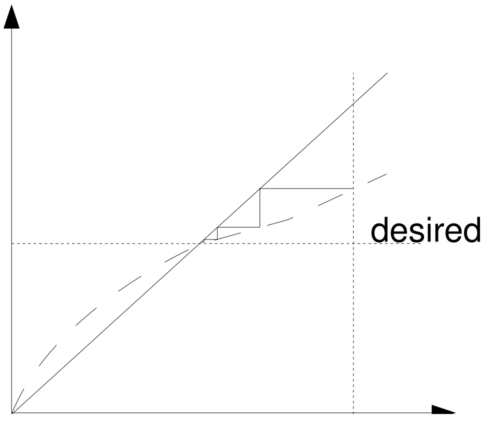

Figure 2: Return map after learning.