Adaptive Control and Reinforcement Learning (16-899C, S09) Lecture #5 (01/29/2009)

CHOMP: Gradient Optimization Techniques for Efficient Motion Planning

Lecturer: Nathan Ratliff, Matt Zucker *Scribe: Don Burnette*

1 Introduction

Covariant Hamiltonian Optimization for Motion Planning (CHOMP), is a method for generating and optimizing trajectories, without requiring that the input path be obstacle free. The algorithm consists of three main features:

- In order to encourage smoothness we must measure the size of an update to our hypothesis in terms of the amount of a particular dynamical quantity (such as total velocity or total acceleration) it adds to the trajectory.
- Measurements of obstacle costs should be taken in the workspace so as to correctly account for the geometrical relationship between the robot and the surrounding environment.
- The same geometrical considerations used to update a trajectory should be used when correcting any joint limit violations that may occur.

2 The CHOMP Algorithm

2.1 Covariant Gradient Descent

The cost of a trajectory using two terms: an obstacle term f_{obs}, which measures the cost of being near obstacles, and a prior term f_{prior}, which measures dynamical quantities of the robot such as smoothness and acceleration. The cost function can be written as

$$C(\xi) = f_{prior}(\xi) + f_{obs}(\xi). \tag{1}$$

The f_{prior} term is merely the sum of the squared derivatives, which can be expressed in the simple quadratic form

$$f_{prior}(\xi) = \frac{1}{2}\xi^T A\xi + \xi^T b + c, \tag{2}$$

for suitable matrix, vector, and scalar constants A, b, c. When constructed as defined above, A will always be symmetric positive definite for all d. The goal is to improve the trajectory at each iteration by minimizing a local approximation of the function that suggests only smooth perturbations to the trajectory. At iteration k, within a region of our current hypothesis ξ_k, we can approximate our cost using a first-order Taylor expansion:

$$C(\xi) = C(\xi_k) + g_k^T(\xi - \xi_k), \tag{3}$$

where $g_k = \nabla C(\xi_k)$. Using this expansion, our update can be written formally as

$$\xi_{k+1} = \underset{\xi}{\operatorname{argmin}} \left\{ C(\xi_k) + g_k^T(\xi - \xi_k) + \frac{\lambda}{2}||\xi - \xi_k||_M^2 \right\}, \tag{4}$$

where the notation $||\delta||_M^2$ denotes the norm of the displacement $\delta = \xi - \xi_k$ taken with respect to the Riemannian metric M equal to $\delta^T M \delta$. Setting the gradient of the right hand side of equation 4 to zero and solving for the minimizer results in the following more succinct update rule:

$$\xi_{k+1} = \xi_k - \frac{1}{\lambda} M^{-1} g_k \tag{5}$$

This update rule serves to ensure that the trajectory remains smooth after each trajectory modification.

2.2 Obstacle avoidance

If obstacles are static, it becomes advantageous to simply precompute a signed distance field $d(x)$ which stores the distance from a point $x \in \Re^3$ to the boundary of the nearest obstacle. Values of $d(x)$ are negative inside obstacles, positive outside, and zero at the boundary. There are a few key advantages of using the signed distance field to check for collisions.

- Collision checking is very fast, taking time proportional to the number of voxels occupied by the robots skeleton.
- Since the signed distance field is stored over the workspace, computing its gradient via finite differencing is a trivial operation.
- Because we have distance information everywhere, not just outside of obstacles, we can generate a valid gradient even when the robot is in collision, a particularly difficult feat for other representations and distance query methods.

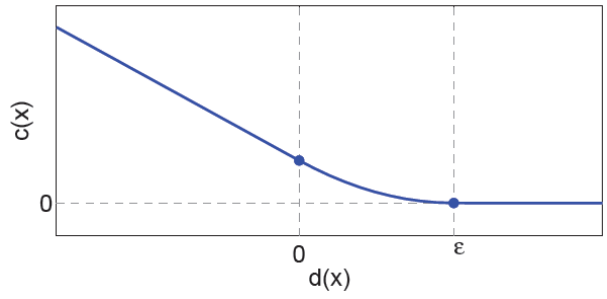

Figure 1: Potential function for obstacle avoidance

Now we can define the workspace potential function c(x), which penalizes points of the robot for being near obstacles. A good choice, shown in Figure 1, is given by

$$c(x) = \begin{cases} -d(x) + \frac{1}{2}\epsilon, & \text{if } d(x) < 0 \\ \frac{1}{2\epsilon}(d(x) - \epsilon)^2, & \text{if } 0 \leq d(x) \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{6}$$