CHOMP: Covariant Hamiltonian Optimization for Motion Planning

In CHOMP, the objective function $U(\xi) = f(F_{obs}[\xi], F_{smooth}[\xi])$

complementary:

- $F_{smooth}[\xi]$ → Captures dynamics of trajectory independent of environment; governs timing of trajectory
- $F_{obs}[\xi]$ → path integral; (no effect of dynamics, cost independent of velocity) captures the requirement to avoid obstacles; governs the shape of trajectory; computed in workspace (signed distance fields):

Why Covariant?

"Covariant": Trajectory representation/parametrization does not inherently affect the optimization process in CHOMP. By making the optimization invariant to parametrization and optimizing directly in the space of trajectories, this makes the gradient descent operation covariant to reparametrization.

Why Hamiltonian?

"Hamiltonian": Gradient descent methods tend to local minima if the objective function is non-convex (which is usually the case). CHOMP uses HMC to perturb the local minima to restart the process for a better solution.

~ a momentum kick making this perturbation invariant to parametrization again.

Additional Pro: Trajectory wide constraints can be specified

Caveat: May not work for every environment

↳ Issues with maze-like environments

↳ May be stuck in collision in highly cluttered environments (no explicit obstacle avoidance)

CHOMP's objective function measures two complementary notions:

$$U[\xi] = F_{obs}[\xi] + \lambda F_{smooth}[\xi]$$

CHOMP considers $F_{smooth}[\xi] = \frac{1}{2}\int_0^1 \left\| \frac{d}{dt}\xi(t) \right\|^2 dt$

Although, any higher order dynamical objectives (acceleration, jerk etc.) can be considered as well.

$$F_{obs}[\xi] = \int_0^1 \int_B c(x(\xi(t), u)) \left\| \frac{d}{dt} x(\xi(t), u) \right\| du\, dt$$

$F_{obs}[\xi]$ aims to capture the cost of moving in the obstacle field. It integrates the cost of every point on the body $u \in B$ over time $t \in [0,1]$. The cost is computed based on distances in workspace. The function $x(\cdot,\cdot)$ maps the point on the trajectory $\xi(t) \in$ C-space and the body point 'u', to cartesian space point.

The multiplier $\left\| \frac{d}{dt} x(\cdot,\cdot) \right\|$ in the integrand ensures that the path cannot avoid accumulating cost by traversing faster in the obstacle field, it must instead aim to avoid them.

As before, Gradient descent step:

$$\xi_{i+1} = \xi_i - \eta_i \bar{\nabla} U(\xi_i), \quad \text{where}$$

$$\bar{\nabla} U(\xi) = \bar{\nabla} F_{obs}[\xi] + \lambda \bar{\nabla} F_{smooth}(\xi)$$

$$\bar{\nabla} F_{obs}[\xi] = \int_B J^T \|x'\| \left[(I - \hat{x}'\hat{x}'^T) \nabla c - c\kappa \right] du \quad \left[\text{using Euler-Lagrange equation}\right]$$

$\kappa = \|x'\|^{-2}(I - \hat{x}'\hat{x}'^T)x''$ is the curvature vector along the workspace trajectory traced by a body point,

x', x'' are the velocity and acceleration of a body point,
$\hat{x}'$ is the normalized velocity vector,
J is the kinematic Jacobian at that body point.

Note, the dependencies on time 't' and body point 'i' are suppressed for simplified notations.

The matrix $(I - \hat{x}'\hat{x}'^T)$ is a projection matrix that projects workspace gradients orthogonally to the trajectory's direction of motion; so that the trajectory's speed profile is not directly manipulated.

CHOMP Practical Considerations:

(a) Termination Condition: $\bar{\nabla} U[\xi] < \text{threshold}$

Note that CHOMP can terminate with a trajectory that is in collision. Neither does it report failure on finding / terminating with trajectory in collision

$\Rightarrow$ We need another higher level planner/ procedure that verifies the validity of the trajectory.

Note this is common to CHOMP and TrajOpt and in general this class of algorithms, since they don't explicitly avoid obstacles.

(b) Practical consideration in cost functional:

Early iterations → weigh $f_{obs}[\xi]$ heavily to find collision-free potentially non-smooth paths

Weigh $f_{smooth}[\xi]$ progressively to terminate with a smooth trajectory.

(c) Modeling the Robot and Obstacle:

↳ for efficient computation of $F_{obs}[\theta]$

Consider a robot's body to be a series of primitives (most commonly spheres) for which swept-volume as well as signed distance fields can be easily computed. Similarly, the obstacles in the environment are also modeled as a set of primitives

Pro: computational efficiency

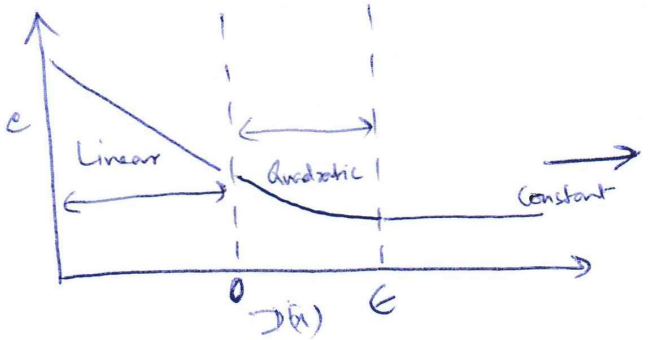
Caveat: Conservative movements due to spherical approximations

(Is it really that bad?
↳ higher level planner can verify with detailed models)

(d) Obstacle Cost formulation:

CHOMP penalizes distance from obstacles as follows:

$$c(x) \begin{cases} -D(x) + \frac{1}{2}\epsilon & D(x) < 0 \\ \frac{1}{2\epsilon}(D(x) - \epsilon)^2 & 0 < D(x) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$



$c(x)$ can however be modified for other user-specified requirements in the trajectory.

(e) Dealing with Local Minima: Hamiltonian Monte Carlo (HMC)

Given an objective $U(\xi_t)$, we can construct an associated probability distribution:

$$p(\xi_t, \alpha) \propto \exp\{-\alpha U(\xi_t)\}$$

$\Rightarrow$ High cost trajectories have low probability of being sampled

$\Rightarrow$ Low cost trajectories " high " " " "

$\alpha > 0$ adjusts how flat the distribution is.

as α tends to ∞, distribution becomes increasingly peaked around minimum. (hard to sample)

$\rightarrow$ Slowly increase α.

Intuitively, the concept behind HMC:

Imagine scaled objective $\alpha U(\xi_t)$ as a landscape.

Then, ξ_t is a single point (ball) in that landscape.

Optimization procedure

(i) Throw the ball with a momentum

(ii) Gradually reduce energy of the ball
↳ would have visited many local minima before settling in a deep basin.

Hamiltonian (H) = U (potential energy) + K (kinetic energy)

Potential energy = $U(\xi_t)$

Kinetic energy = $K(\gamma) = \frac{1}{2}\gamma^T\gamma$ where γ is the momentum.

Desired: $p(\xi_t) \leftarrow$ samples from

But, instead, we sample from $p(\xi_t, \gamma) \propto e^{-H(\xi,\gamma)}$
$\propto e^{-U(\xi)} e^{-K(\gamma)}$

and throw away the momentum samples.

We Knew from physics:

$$\begin{cases} \frac{d\xi}{dt} = \gamma & \text{(rate of change of position} \rightarrow \text{momentum)} \\ \frac{d\gamma}{dt} = -\bar{\nabla} U(\xi) & \text{(rate of change of momentum (force)} \rightarrow \text{change in potential energy)} \end{cases}$$

Conservation of energy principle:

$H(\xi(t), \gamma(t))$ same independent of t.

↳ Isocontours of Hamiltonian

So, simulating the Hamiltonian dynamics may move a point from $(\xi(0), \gamma(0))$ to $(\xi(t), \gamma(t))$ where $\xi(i)$ can be very different from $\xi(0)$ but $p(\xi, \gamma) \propto \exp\{-H(\xi, \gamma)\}$ is same (conservation of energy).

HMC procedure for optimization

(1) Sample initial trajectory momentum γ from Gaussian $p(\gamma) \propto e^{-\frac{1}{2}\gamma^T\gamma}$

(2) Simulate the system to obtain ξ_{t+1} from ξ_t: Practically, use the leap-frog method of numerical integration

$$\begin{cases} \gamma_{t+\epsilon/2} = \gamma_t - \epsilon/2 \, \bar{\nabla} U(\xi_t) \\ \xi_{t+\epsilon} = \xi_t + \epsilon \gamma_{t+\epsilon/2} \\ \gamma_{t+\epsilon} = \gamma_{t+\epsilon/2} - \epsilon/2 \, \bar{\nabla} U(\xi_{t+\epsilon}) \end{cases}$$

↳ Reversibility property (running it forward for T iterations and backward for T iterations returns the same point)

(3) Reject if the final point is less originally probable than initial point with $p(\xi_{t+1}, \gamma_{t+1}) / p(\xi_t, \gamma_t)$ (due to errors in numerical integration)

Otherwise accept it without question.

CHOMP and TrajOPT

Key differences in

(a) Collision checking (dealing with obstacles) and penalizing

(b) Numerical Optimization Scheme
↳ Sequential Convex Optimization

In practice, Section (d) [Computation of $c(x)$] is precomputed in CHOMP to efficiently compute ∇f_{obs} during planning.