Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization

John Schulman, Alex Lee, Ibrahim Awwal, Henry Bradlow, Pieter Abbeel [*]

Trajectory optimization algorithms have two roles in robotic motion planning. First of all, they can be used to smooth and shorten trajectories generated by some other method. Second of all, they can be used to plan from scratch: one initializes with a trajectory that contains collisions and perhaps violates constraints, and one hopes that the optimization converges to a high-quality trajectory satisfying constraints. Using an optimization algorithm to plan from scratch is an especially attractive option in problems with many degrees of freedom (DOF), since the computation time scales favorably with the number of DOF.

We'll consider two types of motion planning problems, both of which can be solved with our method. Let N be the number of degrees of freedom of the robot (including its pose, if it has a moveable base), and let M be the number of actuated degrees of freedom. For example, in a walking humanoid, the joints are actuated but the six-DOF pose is not, so $N = M + 6$.

1. **Kinematic planning**: the optimization problem has $T \times N$ variables—the joint angles at each timestep.

$$\min_{\boldsymbol{\theta}1:T} \sum_t \|\dot{\boldsymbol{\theta}}_t\|^2 + \text{ other costs} \tag{1}$$

subject to

no collisions

kinematic constraints, e.g. end-effector pose

2. **Dynamic planning**: the optimization problem has $T \times (N + M)$ variables—the joint angles $\boldsymbol{\theta}_t$ and torques (or generalized forces) $\boldsymbol{\tau}_t$ at each timestep. We also include the dynamics constraint at each timestep

$$M(\boldsymbol{\theta}_t)\ddot{\boldsymbol{\theta}}_t + C(\boldsymbol{\theta}_t, \dot{\boldsymbol{\theta}}_t) = \boldsymbol{\tau}_t \tag{2}$$

Two of the key ingredients in trajectory optimization for motion planning are (1) the numerical optimization method, and (2) the method of checking for collisions and penalizing them. For numerical optimization, we use sequential convex optimization, with ℓ_1 penalties for equality and inequality constraints. This approach involves solving a series of convex optimization problems that approximate the cost and constraints of the true problem, which is non-convex. For collisions, we compute signed distances using convex-convex collision detection,

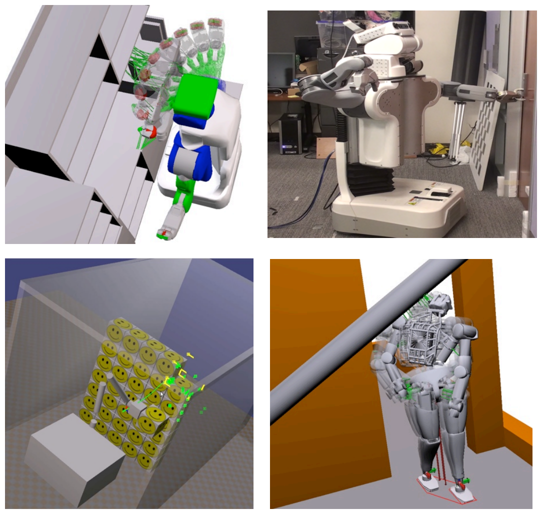

Figure 1: Several problem settings were we have used our algorithm for motion planning. Top left: planning an arm trajectory for the PR2 in simulation, in a benchmark problem. Top right: PR2 opening a door with a full-body motion. Bottom left: industrial robot picking boxes, obeying an orientation constraint on the end effector. Bottom right: humanoid robot model (DRC/Atlas) ducking underneath an obstacle while obeying stability constraints.

and we ensure the continuous-time safety of a trajectory by considering the swept-out volume. These two aspects of our approach are complementary, since our collision checking method yields a polyhedral approximation of the free part of configuration space, which can be directly incorporated into the convex optimization problem that is solved at each iteration of the optimization.

The first advantage of our approach is speed. Our implementation solves typical arm planning problems in around $100-200$ ms and solves problems involving many more degrees of freedom in under a second. This is largely enabled by our novel formulation of the the collision penalty, which guarantees safety in continuous time by considering swept-out volumes. This cost formulation has little overhead in collision checking and allows us to use a sparsely sampled trajectory. The second advantage of our approach is its reliability—it solves a surprisingly large fraction of planning problems. In our experiments, our algorithm solved a larger fraction of problems than any of the sampling-based planners, which were given a ten second time limit. The third advantage of our approach regards path quality: once the trajectory is free of collisions, our approach will treat collision avoidance as a hard constraint (i.e., keep a certain safe distance from obstacles.) Our algorithm will converge to a locally optimal solution subject to this constraint, without

[*]UC Berkeley, EECS Department

compromising the other objective criteria. The fourth advantage of our approach is flexibility: new constraints and cost terms can easily be added to the problem since the underlying numerical optimization method is numerically robust, and it can deal with initializations that are deeply infeasible.

We performed a quantitative comparison between our algorithm and several open-source implementations of motion planning algorithms, including sampling based planners from OMPL, as well as an implementation of CHOMP. Overall, our algorithm was not only faster than the alternatives, but it solved a larger fraction of the problems. All planners were given a ten second time limit and run with default parameters.

Our source code is available as a BSD-licensed open source package and can be found at `https://github.com/joschu/trajopt`. (See README for documentation link.)