SBP-Guided MPC to Overcome Local Minima in Trajectory Planning

Emily Hannigan[1], Bing Song[1], Gagan Khandate[1], Ji Yin[1], Maximilian Haas Heger[1] and Matei Ciocarlie[1]

I. Introduction

Trajectory planning is often a difficult task for high-dimensional systems, especially those with non-linear dynamics. Two common methods for trajectory planning in non-linear systems are Model Predictive Control (MPC) and kinodynamic Sampling-Based Planning (SBP). In this paper, we focus on a variant of MPC called the iterative Linear Quadratic Gaussian (iLQG) algorithm[1]. iLQG has been shown to be fast and effective for generating trajectories to reach a goal. However, when optimizing for non-linear dynamics, it runs the risk of falling into local minima. Unlike iLQG, SBP methods such as Rapidly-exploring Random Trees (RRT) [2] can be robust to local minima because they explore by taking random actions instead of following a gradient. However, for similar reasons, SBP often produces inefficient trajectories.

We combine these two algorithms to take advantage of the specific strengths of each. We show that by using an RRT-produced trajectory as a warm start for iLQG, we can overcome local minima while still producing an efficient trajectory. On a specific system model (a robot snake), we show that the combination of these two algorithms allows the robot to reach a goal faster than the use of either algorithm alone in most cases.

In general terms, we define our problem as follows. Consider a robot model with state space $\boldsymbol{x}$ and action space $\boldsymbol{u}$. The dynamics (or forward) model f determines the behavior of the system in discrete time steps as $f(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{x}_{t+1}$. The goal is to determine an action sequence $U = \{\boldsymbol{u_0}...\boldsymbol{u}_T\}$ such that the final state of the robot $\boldsymbol{x}_T$ matches a desired configuration $\boldsymbol{x}_d$. We note that $\boldsymbol{x}_d$ might only specify desired values for a subset of $\boldsymbol{x}$, leaving the rest of the state variables unconstrained.

II. iLQG Trajectory Optimization

iLQG is an MPC algorithm that works by progressively refining an action sequence U in order to minimize a cumulative cost, defined for each pair of states and actions. Starting from an existing action sequence, iLQG uses the model of dynamics $f()$ to predict the state of the robot as it follows the resulting trajectory, as well as its cumulative cost. iLQG then attempts to optimize the cost w.r.t. to the actions by assuming locally linear dynamics around the current trajectory. These iterations repeat until convergence, at which point the robot takes the first or the first few actions from this improved action sequence, and the process restarts.

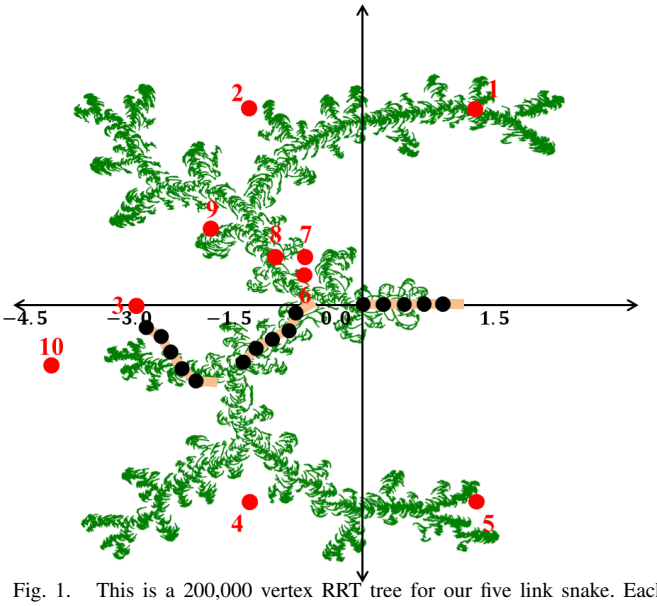

Fig. 1. This is a 200,000 vertex RRT tree for our five link snake. Each red numbered point represents a goal vertex which is used to generate ten trajectories used in the results section. The snake is shown at scale in three positions as it moves toward the goal

This methods is effective at generating action sequences for complex dynamical systems. However, due to the assumption of locally linear dynamics, it can get stuck in local optima. Furthermore, it requires an initial "guess" for an action sequence in order to begin the optimization process, and can be sensitive to the quality of this starting sequence.

III. Kinodynamic Sampling-based Planning

Kinodynamic SBP also aims to find action sequences that reach a certain goal. However, in contrast to iLQG, such methods work through random exploration of the state and action spaces. We focus here on kinodynamic RRT, a variant of SBP, as our method of choice. This method builds a tree of vertices, each storing a state x. The tree grows by using a local planner that is able to extend a tree vertex towards a randomly sampled point in state space; the local planner we use here consists of simply trying a set of random actions and choosing the best one. The action that corresponds to each branch of the tree is saved along with the newly introduced vertex, allowing us to trace an action sequence from any node in the tree back to the root.

SBP methods such as RRT are effective at avoiding local optima, thanks to the intrinsically stochastic nature of the exploration. While a model of the dynamics is needed when sampling actions to grow the tree, no gradients need to be

[1]Department of Mechanical Engineering, Columbia University, New York, NY 10027, USA.
{ejh2192, bing.song, gk2496, ji.yin, mkh2149, matei.ciocarlie}@columbia.edu

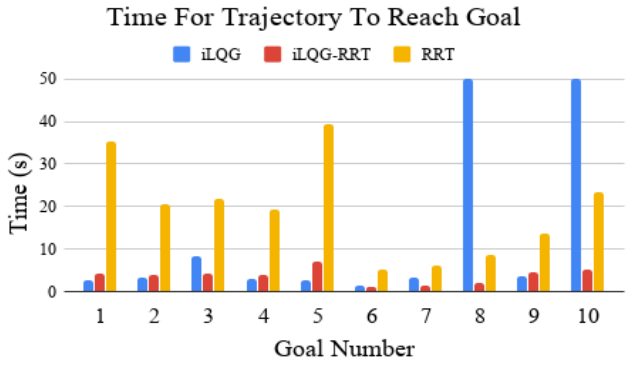

Fig. 2. Each goal number references a goal point in Figure 1. Each bar represents the time it took for a trajectory to reach that goal.

computed. However, these methods can be slow in high-dimensional state spaces with complex dynamics; furthermore, the trajectory produced is generally sub-optimal.

IV. Proposed Method

To improve the performance of iLQG, we use trajectories generated by RRT in two different ways. First, instead of optimizing over the entire distance to the final goal with iLQG, we split the computation into a series of waypoint goals generated by the longer RRT trajectory. This reduces the trajectory length that iLQG needs to improve over. Second, for each subtrajectory that iLQG is given, RRT also provides a "warm start" – an initial guess for iLQG to begin its optimization on. We show that this iLQG+RRT has a significant speedup in a range of scenarios. In our experiments, we show that for many trajectories, iLQG is either trapped or slowed down by local minima, and by using our iLQG+RRT method, we are able to completely avoid the local minima and get to the goal quick and effectively. To demonstrate this improvement, we compare trajectories generated by our method to the trajectories computed by iLQG alone (with an initial trajectory of all zero actions) and RRT alone.

V. Results

We demonstrate our method on a model of a five-link 2D simulated snake robot with four actuated joints. The state vector $\boldsymbol{x} \in \mathcal{R}^{14}$ consists of the angular positions and velocities of the joints, plus the cartesian position and velocity of the head. The action vector $\boldsymbol{a} \in \mathcal{R}^4$ contains the applied joint torques. The dynamic model $f()$ consists of an explicit time-stepping integration scheme that solves for joint accelerations at each time step then integrates to obtain the next state. To allow for forward locomotion, we use an anisotropic friction model for each link, with coefficients of friction of 0.1 in the forward direction and 0.9 in the tangential direction. Goals are specified as desired cartesian positions for the head of the snake.

For experiments involving RRT, we pre-compute a 200,000 node RRT tree spanning the vicinity of the start state. In the current version of the algorithm, tree geneartion takes 24 hrs to compute. However, we consider this to be pre-computation time, re-used for any subsequent goal. We note that the tree includes no cartesian obstacles; however, the dynamics constraints of our system are difficult enough to make progress in any cartesian direction a very difficult task even in the absence of obstacles.

We chose ten goals to compare trajectory planning algorithms. Figure 1 shows these goals superimposed on the RRT tree. For each goal, we compute a trajectory using three methods: iLQG, RRT, and iLQG+RRT. We plot the time for each trajectory to reach the goal as a bar graph in Figure 2.

We notice a trend in the performance of iLQG and the location of the goals. For goals 1, 2, 4, 5, and 9, iLQG+RRT and iLQG perform similarly. But for goals 3, 6, and 7, iLQG trajectores take a long time to begin moving, and for goals 8 and 10, iLQG is completely stuck. This is likely due to iLQG falling into local minima. For these cases, the time to goal for iLQG vs iLQG+RRT highlights the benefits of waypoints and "warm starts".

We hypothesize that the local minima that iLQG falls into in the forward direction may be a result of the discontinuities in our friction model. The discontinuity in our dry friction model is the boundary between static and kinematic friction which occurs right as the snake begins to move. It is in these cases where we see the benefit of using our method. iLQG is very sensitive to starting position, and the multiple warm starts it recieves using the RRT waypoints is likely helping it to avoid those minima. This study represents a work in progress, and there are a number of topics that require further attention:

- The RRT tree is expensive to compute, and, in the current form, specific to a given start state. We have also not tested our method with cartesian obstacles, which would require a separate RRT tree for each obstacle configuration, and significant cost engineering in iLQG to guarantee a collision-free path. We would like to develop an alternative formulation where the tree is broadly applicable to numerous start situations, in which case the resources invested in pre-computation deliver results later on at run time.
- It could be possible to overcome the local minima that sometimes thwart iLQG by using smoother dynamics models, instead of using a warm start. However, realistic frictional models are inherently non-smooth, and there may be other local minima that are not the result of discontinuities in dynamics.

Nevertheless, using RRT as a warm start for iLQG shows potential to be an important strategy for overcoming local minima. Having a precomputed RRT tree available for a robot to use in real time when planning trajectories may help speed up path planning, and in some cases prevent failure.

References

[1] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.