Sequential Trajectory Optimization for Path Planning Along Manifolds

Holly Dinkel, Sarah Mitchell, Emiko Soroka
Stanford University, AA 203 Spring 2020 Final Project

Problem

We consider the path planning problem for the end effector of a two-joint, two-link nonplanar robotic manipulator. The configuration space of the end effector (all attainable positions) is in the shape of a torus. The manipulator also operates in an environment with obstacles, which it must avoid as it moves from an initial state to a final one along the torus manifold.

The asymptotically optimal path planning method exploits knowledge of the geodesic for planning a path along the manifold from a starting position to a final position [4]. However, for an obstacle-cluttered configuration space, knowledge of the configuration space manifold becomes difficult to use as it fails to incorporate collision avoidance with obstacles.

A common method to perform in the presence of obstacles is random exploration to find a feasible path through various implementations of the Rapidly-Exploring Random Tree (RRT) algorithm, as it allows for this collision check [2, 6]. However, the feasible path will not be optimal. A common method to perform optimization while respecting collision constraints is Sequential Convex Programming. SCP is a direct method that can optimize a cost function with respect to many different constraints, although it does not guarantee convergence to the solution. For these reasons, modern algorithms have generalized the SCP method to provide theoretical convergence guarantees.

Robotic Manipulator Kinematics

With a task space trajectory $(\boldsymbol{x}, \dot{\boldsymbol{x}})$ known, a feasible joint space trajectory $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be found that produces the given trajectory. The differential kinematics equation has the form

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1}$$

where $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the Cartesian coordinates of the manipulator end effector in the world frame. We also define q_1 and q_2 as the joint angles, and use l_1 and l_2 represent link lengths. The transformation relating the position of the end effector to the world frame for a two-joint manipulator is given by (2).

$$\boldsymbol{x} = f(\boldsymbol{q}) = \begin{bmatrix} l_1 cos(q_1) + l_2 sin(q_2) cos(q_1) \\ l_1 sin(q_1) + l_2 sin(q_1) sin(q_2) \\ l_2 cos(q_2) \end{bmatrix} \tag{2}$$

The Jacobian matrix $\mathbf{J}$ relates the joint angular velocities to the positional velocities in Cartesian space. For the two-joint manipulator with joint angular velocity inputs $\dot{\mathbf{q}} = \begin{bmatrix} \dot{q_1} & \dot{q_2} \end{bmatrix}^T$ this Jacobian is given by

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} -l_1 sin(q_1) - l_2 sin(q_1) sin(q_2) & l_2 cos(q_1) cos(q_2) \\ l_1 cos(q_1) + l_2 sin(q_2) cos(q_1) & l_2 sin(q_1) cos(q_2) \\ 0 & -l_2 sin(q_2) \end{bmatrix}$$

The velocity of the end effector in the task space at the current state is determined by the error vector between the current end effector state, $\boldsymbol{x}_i$, in task space and the goal location, $\boldsymbol{x}_g$ in task space, as shown in Equation (3).

$$\dot{\boldsymbol{x}}_i = \boldsymbol{x}_g - \boldsymbol{x}_i \tag{3}$$

Because the Jacobian relating joint space to task space is not square, the inverse solution to (1) is found through the pseudoinverse, $\boldsymbol{J}^\dagger$. This pseudoinverse solution inverts the forward kinematics to find the control input in joint space, $\boldsymbol{q}_i$, at each time step, and is given in Equation (4).

$$\dot{\boldsymbol{q}}_i = \boldsymbol{J}_i^\dagger(\boldsymbol{q}_i)\dot{\boldsymbol{x}}_i \tag{4}$$

The input $\boldsymbol{q}_i$ is used to control the end effector to the next joint angle state, $\boldsymbol{q}_{i+1}$, after a time step δt as shown in Equation (5).

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \dot{\boldsymbol{q}}_i \delta t \tag{5}$$

Using the joint angle state in (5), the end effector position at the next time step, $\boldsymbol{x}_{i+1}$, can be found through the forward kinematics from (2). The implementation of this planning and control strategy is shown in Figure 1.

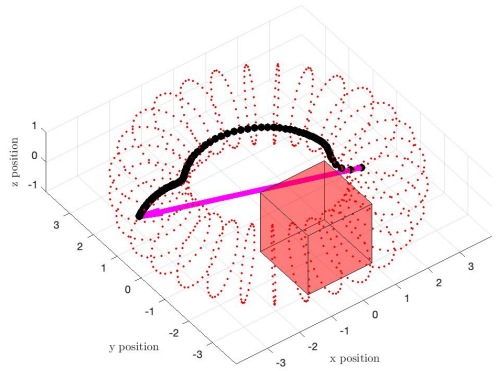

Figure 1: Obstructed configuration space manifold for a manipulator with link lengths $l_1 = 3$, $l_2 = 1$ and optimal manifold-constrained path from initial to goal position

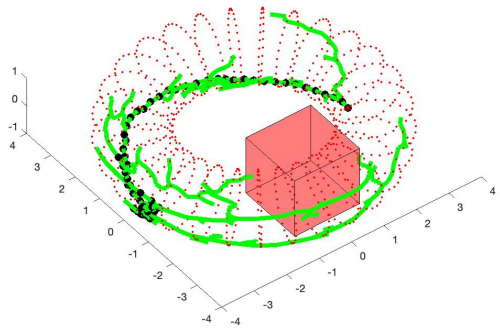

Figure 2: Obstructed configuration space manifold for a manipulator with link lengths $l_1 = 3$, $l_2 = 1$ and RRT with RRT-planned path to goal

Rapidly-Exploring Random Tree

The rapidly-exploring random tree algorithm was developed as a randomized path planning technique that computes a collision-free kinodynamic trajectory. The tree rapidly explores with bias towards a goal position, and tree expansion terminates when the desired end-effector position is reached [5]. The pseudo-code for the RRT planner is shown in Algorithm 1.

Algorithm 1 Pseudo-Code for the RRT Planner

function RRT($\boldsymbol{q}_{init}$,$\boldsymbol{x}_{init}$,$\boldsymbol{q}_{goal}$,$\boldsymbol{x}_{goal}$)

T.init($\boldsymbol{q}_{init}$,$\boldsymbol{x}_{init}$) – initialize the RRT
n = 1 – current size of RRT
for $k \leftarrow 1$ **to** K **do**
  $\boldsymbol{q}_{rand} \leftarrow$ RANDOM_STATE
  $\boldsymbol{x}_{rand} \leftarrow$ EE_TO_WORLD_FRAME($\boldsymbol{q}_{rand}$)
  $\boldsymbol{q}_{near}$, parent_idx $\leftarrow$ NEAREST_NEIGHBOR($\boldsymbol{x}_{rand}$, T)
  $\boldsymbol{x}_{near} \leftarrow$ EE_TO_WORLD_FRAME($\boldsymbol{q}_{near}$)
  if $\boldsymbol{x}_{near}$ ~*IN_OBSTACLE* **then**
    T.$\boldsymbol{q}$(n+1), T.$\boldsymbol{x}$(n+1) = STEER_TOWARDS($\boldsymbol{q}_{rand}$,$\boldsymbol{x}_{rand}$,$\boldsymbol{q}_{near}$,$\boldsymbol{x}_{near}$)
    P(n+1) = parent_idx
    n = n+1
    if *T.*$\boldsymbol{x}$*(end)* = $\boldsymbol{x}_{goal}$ **then**
      path.$\boldsymbol{q}$, path.$\boldsymbol{x}$ = BUILD_PATH(T,P)
    end
  end
end

The implementation of the RRT algorithm for path planning and control for the nonplanar manipulator with a cluttered configuration space is shown in Figure 2.

Generalizing SCP with GuSTO

Alternatively, many nonlinear trajectory optimization problems can be solved via sequential convex programming (SCP), which models a nonconvex problem using successive convex approximations. However, while a convex subproblem has a global optimum, a trajectory found via SCP may only be locally optimal and dependent on initial data provided to the solver [3]. SCP can also be computationally expensive, though leveraging similarities between each convex problem reduces this cost. Recent developments have improved the performance of SCP algorithms, enabling them to perform trajectory optimization in real time [2].

Guaranteed Sequential Trajectory Optimization (GuSTO) generalizes the concept of SCP algorithms to drift control-affine nonlinear dynamical systems. Bonalli et al. [2] show that this new algorithm offers advantages over many existing methods. Specifically, GuSTO provides hard enforcement of dynamics constraints and strong convergence guarantees, as well as allowing free final-time and goal-set constraints.

Acceleration The sequence of SCP problems is a converging subsequence towards a stationary point. This accelerates convergence by using the solution of the SCP problem at each iteration $i = 1, \ldots, T$ to initialize a shooting method that attempts to converge on the solution of the original nonlinear problem. If this solution is found, the algorithm can terminate.

Constraints Constraints are both passed to the solver and added as penalties on the objective function. This is advantageous because the constraints are inexact: the nonconvex

collision constraints must be convexified, while the nonlinear dynamics are linearized. GuSTO also implements checks for inaccurate linearization and trust region violations, which will cause a trajectory to be rejected.

GuSTO supports several types of constraints:

- Linear constraints on the state and control, e.g. a maximum and minimum value.
- Initial and final state constraints.
- Second-order-cone constraints on the state and control, useful for states implemented with quaternions which contain the constraint $\|q\| = 1$.
- Linearized dynamics constraints of the form: $x_{i+1} - A_i * (x_i - x_{i,prev}) - B_i * (u_i - u_{i,prev}) = 0$ for $i = 1, \ldots, T-1$
- Convexified collision constraints using the signed distance function, which returns a positive number if a point x is outside a set (representing an obstacle), and a negative number if x is inside the set.

We implemented a 2-link manipulator model for use with the GuSTO algorithm to minimize the integral cost shown below [2].

$$\begin{aligned} J(t_f, x, u) &= \int_0^{t_f} f^0(x(t), u(t))dt \\ &= \int_0^{t_f} \Big(\|u(t)\|_R^2 + u(t) \cdot f^0(x(t)) + g(x(t)) \Big) dt \end{aligned}$$

GuSTO Problem Formulation

In the SCP formulation, the manipulator dynamics are discretized with $i = 1, \ldots, N$ steps and step length $h = (t_f - t_i)/N$. The augmented state is defined as $\bar{\boldsymbol{x}} = [\boldsymbol{x}, \boldsymbol{q}]^T$, and the control is given by $\boldsymbol{u} = \dot{\boldsymbol{q}}$. The nonlinear dynamics used to implement SCP are shown in Equation (6) and the linearization of these dynamics is shown in Equation (7).

$$\bar{\boldsymbol{x}}_{i+1} = \bar{\boldsymbol{x}}_i + \boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{u}_i \delta t \tag{6}$$

$$\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_0, \boldsymbol{q}_0) + \boldsymbol{A}_i(\boldsymbol{x}_i - \boldsymbol{x}_0) + \boldsymbol{B}_i(\boldsymbol{u}_i - \boldsymbol{u}_0) \tag{7}$$

$$\text{where } \boldsymbol{A}_i = \begin{bmatrix} 0 & \boldsymbol{J}(\boldsymbol{q}_0)\delta t \\ 0 & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{B}_i = \begin{bmatrix} \boldsymbol{J}(\boldsymbol{q}_0) \\ \boldsymbol{I}\delta t \end{bmatrix} \tag{8}$$

Manifold Constraint The manipulator is implicitly constrained to the toroidal manifold by its dynamics: if the starting point is on the manifold, the equations of motion can only reach points on the manifold. Intuitively, linearizing the dynamics as is done in SCP removes this implicit constraint. However Bonalli et al. [1] show that this need not be a concern. Although individual SCP iterations may not respect the manifold constraint, if SCP converges, it must converge to a trajectory on the manifold.

Problem Formulation The dynamics are linearized as in Equation 7. A trust region constraint with box trust region $\mathcal{T}$ is added and the problem is formulated as follows:

$$\text{minimize} \quad \sum_{k=1}^{T} J(x_k, u_k, t_k) \tag{9}$$

$$\text{subject to} \quad x_{k+1} = A_k x_k + B_k u_k \tag{10}$$

$$x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 1, \ldots, T \tag{11}$$

$$(x_k, u_k) \in \mathcal{T},\ k = 1, \ldots, T \tag{12}$$

The results for 10 iterations of GuSTO with links $l_1 = 1$, $l_2 = 3$ and $T = 300$ discrete points are shown in Figures 3, 4, and 5.

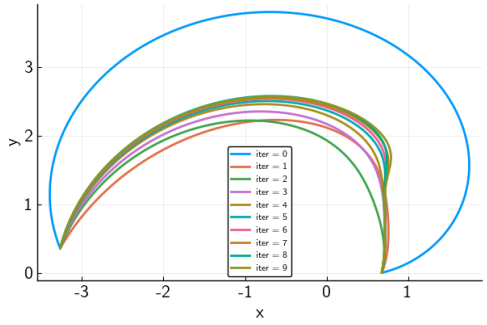

Figure 3: Configuration space manifold for a manipulator with link lengths $l_1 = 1$, $l_2 = 3$ and GuSTO path trajectory to goal, flattened to 2 dimensions

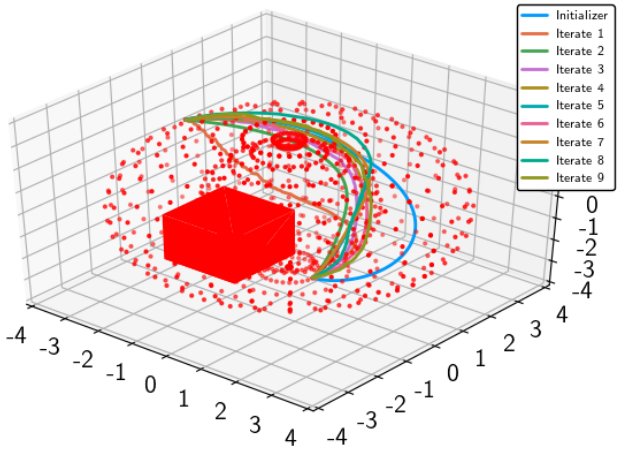

Figure 4: Trajectories for 10 iterations of GuSTO

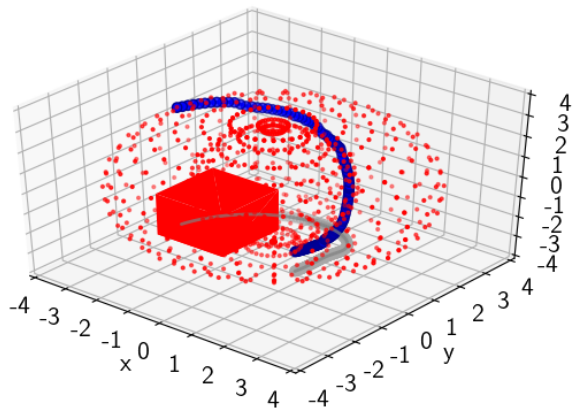

Figure 5: Final trajectory found by GuSTO

We initialized the algorithm with a path along the torus, shown in light blue in Figures 3 and 4. The main issues we encountered with the GuSTO implementation on the manipulator were discretization problems. Because the mapping between joint angles and Cartesian space is nonlinear, we had to use small time steps, requiring a large number of discrete points along the trajectory. The final trajectory in Figure 5 moved much closer to the center of the torus, where less control input is required to reach the final position. However, we still had trouble implementing the example with link lengths $l_1 = 3, l_2 = 1$ to match the torus space in the kinematic planning and RRT implementations.

Conclusion

In this paper we compared several approaches for controlling a 2-link nonplanar manipulator. RRT is a commonly used approach, which finds feasible (but not optimal) solutions. Sequential convex programming seeks an optimal solution, but the inaccuracy introduced by linearizing and convexifying constraints means that a globally optimal solution is not guaranteed. Recent developments such as the GuSTO algorithm seek to speed up the convergence of SCP and provide stronger guarantees on its performance, allowing SCP to be applied in real time.

Known Issues

- The manipulator is highly nonlinear, requiring a small discrete time step to prevent the linearization from being too coarse.
- The GuSTO implementation we used is still in development by the ASL lab and does not contain the shooting method implementation.
- For some initial points our implementation did not converge and multiple trajectories were rejected by the algorithm for trust region violations or poor linearization. The performance is dependent on the time step h because the nonlinear dynamics are already discretized: $x_{i+1} = x_i + h\dot{x}_i$.

Future work could focus on resolving these implementation issues. We could also model a torque-controlled manipulator, which contains both velocity and acceleration terms. This is preferable over the simpler kinematics model because it allows the control system to account for forces acting on the manipulator.

Code The RRT implementation and the manipulator model for GuSTO are both available at https://github.com/JeffersonAero/AA203project/

References

[1] Riccardo Bonalli, Andrew Bylard, Abhishek Cauligi, Thomas Lew, and Marco Pavone. Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex programming approach, 2019.

[2] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. Gusto: Guaranteed sequential trajectory optimization via sequential convex programming. 2019. URL http://arxiv.org/abs/1903.00155v1.

[3] John Duchi, Stephen Boyd, and Jacob Mattingley. Sequential convex programming. EE 364b University Lecture, 2020.

[4] Léonard Jaillet and Josep M. Porta. Asymptotically-optimal path planning on manifolds. *Robotics: Science and Systems*, VIII, 2012.

[5] Steven LaValle and James Kuffner. Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[6] Mike Vande Weghe, Dave Ferguson, and Siddhartha S. Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 477–482, 2007.