Fast Tracking of Smooth Trajectories For a Constrained Differentially Flat System

Final Report

Martin Freeman (emmerich@stanford.edu)
Chandra Rajyam (rajyamch@stanford.edu)

I. Introduction

A core goal of motion planning and control of mobile robots is to determine a dynamically-feasible and collision-free path to the objective. Beyond this, specific use cases manifest their priorities in the design of trajectory optimization and control approaches. Time-sensitive use cases such as search and rescue operations can benefit by recognizing *minimum-time traversal* as a fundamental priority when designing such approaches.

We propose a two-part optimization and control approach that leverages a speed-optimized, convex optimization approach to smooth trajectory generation[1] and fast online tracking via model predictive control (MPC). MPC was chosen to address hard constraints on the feasible workspace, ensure constant coupling of the controller to the control problem itself, and achieve robustness to disturbances. In a non-deterministic environment with an optimal global path and speed profile generated by the Convex Elastic Smoothing (CES) algorithm, the fast MPC-based tracker thus acts as a robust local planner that quickly defaults to following the optimal path and speed profile but can re-plan feasible and loosely optimal trajectories on demand. The vehicle dynamics are described by the unicycle model.

II. Problem Statement

We begin by summarizing the problem statement from [1]. Let $\mathcal{W} \subset \mathbb{R}^2$ be the two-dimensional workspace for a car-like vehicle. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_m\}$, with $O_i \subset \mathcal{W}$ $i = 1, \ldots, m$, denote the set of obstacles. We presume that obstacles have polygonal shape. Let $\mathbf{q} \in \mathcal{W}$ represent the position of the vehicle (with unicycle dynamics), and $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ its velocity and acceleration. The vehicle is non-drifting and non-reversible, so the heading of the vehicle is the same as its instantaneous velocity vector. $\phi(\dot{\mathbf{q}})$ is the mapping from the vehicle's speed to its heading. The control input is $\mathbf{u} = \left[u^{\text{long}}, u^{\text{lat}}\right]$. With m as the mass, the dynamics of the vehicle are given by:

$$m\ddot{\mathbf{q}} = \begin{bmatrix} \cos\phi(\dot{\mathbf{q}}) & -\sin\phi(\dot{\mathbf{q}}) \\ \sin\phi(\dot{\mathbf{q}}) & \cos\phi(\dot{\mathbf{q}}) \end{bmatrix} \mathbf{u} \tag{1}$$

There is also friction circle constraint for $\mathbf{u}$. Let μ be the friction coefficient and g be the gravitational acceleration.

$$\|\mathbf{u}\| \leq \mu m g \tag{2}$$

The longitudinal force is bounded by:

$$u^{\text{long}} \leq \bar{U}^{\text{long}} \tag{3}$$

There is also a minimum turn radius (depending on specific vehicle parameters like wheelbase) associated for the vehicle:

$$u^{\text{lat}} \leq m\frac{\|\dot{\mathbf{q}}\|^2}{R_{\min}} \tag{4}$$

The goal is to apply the CES algorithm, taking into account the vehicle's model, obstacle set $\mathcal{O}$, and a discretized reference trajectory $\mathcal{P}$. The output will be a dynamically-feasible, collision-free, and smooth trajectory with an optimized speed profile. We then will use fast online tracking via model predictive control (MPC) to follow the trajectory.

III. Smooth, Speed-Optimized Trajectory Generation via CES

Broadly speaking, the CES outlined in [1] was implemented to generate a dynamically-feasible, smooth and speed-optimized trajectory. RRT was first be used to produce a rough, but feasible global reference path to be optimized. This trajectory was discretized into a set of waypoints $\mathcal{P} := \{P_0, P_1, \ldots, P_n\}$, where $P_i \in \mathcal{W} \backslash \mathcal{O}$ for $i = 1, \ldots, n$. Bubbles were placed sequentially along this reference trajectory, and the hull of the bubbles defines a tube-shaped collision-free set of positions $\mathcal{X}_c$ that the trajectory may be stretched through to while maintaining feasibility.

Two convex problems were then solved per iteration in an alternating procedure, one for optimizing the smoothness by stretching the trajectory given a speed profile, and one for optimizing the speed profile along a stretched trajectory. After a termination criterion is met, the optimal path $\mathcal{P}^*$ and speed profile $\mathcal{V}^*$ were passed to the closed-loop controller.

Bubble Generation

Each waypoint P_i has an associated bubble $\mathcal{B}_i$, where A_i and r_i denote the center and radius. The CES generated bubbles such that: (1) its radius has an upper bound of r_u, (2) if possible, a radius no smaller than r_l, and (3) its center is close to P_i. The bubbles were constructed such that the regions are collision-free and represent the feasible space in which new waypoints can be placed.

Elastic Stretching
The elastic stretching procedure views the initial trajectory as an elastic band, with n nodal points whose positions can be adjusted within the bubbles. Given this perspective, an artificial tensile force is defined between two points Q_k and Q_{k+1} as:

$$\mathbf{F}_k := Q_{k+1} - Q_k \tag{5}$$

The balancing force at point Q_k would be:

$$\mathbf{N}_k := \mathbf{F_{k-1}} - \mathbf{F_k} \tag{6}$$

We place new waypoints $Q_1, \dots, Q_n$ within the bubbles, such that we minimize the sum of the norms of the balance forces (subject to vehicle constraints). [1] explains what these constraints are in further detail, but the convex optimization problem presented is:

$$\begin{aligned}
&\min_{Q_3,\dots,Q_{n-2}} \ \textstyle\sum_{k=2}^{n-1} \|2Q_k - Q_{k-1} - Q_{k+1}\|^2 \\
&\text{subject to} \ \ Q_1 = P_1, Q_2 = P_1 + d \cdot \tfrac{\mathbf{v}_1}{\|\mathbf{v}_1\|} \\
&Q_n = P_n, Q_{n-1} = P_n - d \cdot \tfrac{\mathbf{v}_{n-1}}{\|\mathbf{v}_{n-1}\|} \\
&Q_k \in \mathcal{B}_k, \quad k = 3, \dots, n-1 \\
&\|2Q_k - Q_{k-1} - Q_{k+1}\| \leq \min\left\{\tfrac{d^2}{R_{\min}}, \alpha_k \left(\tfrac{d}{\mathbf{v}_k}\right)^2\right\} \\
&\text{for} \quad k = 2, \dots, n-1
\end{aligned}$$

$$\mathbf{a}^{\text{lat}} = \frac{\|\mathbf{v_k}\|^2}{R_k} \leq \sqrt{(\mu g)^2 - \left(\frac{u_k^{\text{long}}}{m}\right)^2} := \alpha_k \tag{7}$$

$$d := \frac{\sum_{k=1}^{n-1} \|P_{k+1} - P_k\|}{n-1}$$

Speed Optimization
We refer to [3] for speed optimization (particularly by calling MTSOS) given the sequence of waypoints from elastic stretching, the friction coefficient μ and the maximum traction force $\bar{U}^{\text{long}}$. The outputs are the corresponding velocity vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ and logitudinal control forces $\left\{u_1^{\text{long}}, \dots, u_n^{\text{long}}\right\}$.

IV. Fast Tracking with MPC

To execute the smooth, speed-optimized trajectory generated by the CES algorithm, two main implementations of online MPC are explored: 1) *Reference tracking*, where we penalize the trajectory based on the MSE generated by the distance difference to the reference trajectory, and 2) *Informed initialization*, where we feed forward the CES trajectory as initial guesses at the optimal solution.

A. Robot Dynamics

Instead of dealing with the unnecessary complexity of lower-level dynamics of a differential drive system, we make use of the much simpler unicycle model below

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} V\cos\phi \\ V\sin\phi \\ \omega \end{bmatrix} \tag{8}$$

A discrete-time approximation to the continuous-time dynamics is performed to generate the state transition function. The trapezoidal rule, an implicit second-order method, is used to define the state transition, a fairly common choice within the Runge-Kutta methods.

We also note that this is a non-linear gaussian process, and as such we will have zero-mean gaussian process noise considered in the state transition function,

$$\mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{w}[k], \mathbf{u}[k]) = \mathbf{x}[k] + \dot{\mathbf{x}}[k]dt$$

Where $\dot{\mathbf{x}}[k]$ is approximated in discrete-time via the trapezoidal rule and the Jacobians $\boldsymbol{J}_x$, $\boldsymbol{J}_u$ of the state transition functions are

$$\boldsymbol{J}_x = \begin{bmatrix} 0 & 0 & -vsin(\theta) \\ 0 & 0 & vcos(\theta) \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{J}_u = \begin{bmatrix} cos(\theta) & -vsin(\theta) \\ sin(\theta) & vcos(\theta) \\ 0 & 1 \end{bmatrix}$$

B. State Estimation

We apply a simple extended Kalman filter (EKF) to estimate the robot state during online control simulations. The EKF uses the state transition function described above and has a fully observable measurement model of the three states directly.

C. Reference tracking

Reference tracking can prove to be useful when the reference trajectory is trustworthy and there are no disturbances that may require rerouting to the goal on demand. As MPC makes use of the LQR tracking approach over the finite horizon (prediction/control window), reference tracking using MPC is susceptible to the same disadvantages. Namely, the LQR's linearization around the nominal non-linear trajectory can commonly lead to divergence if the deviation of the system is too high.

We invoke the standard reference tracking model and cost functions for the system. The cost function $J(\boldsymbol{x}(t))$ describes the MSE of the deviation from the reference trajectory and reasons in terms of control input deviation variables rather than reference control differences as the steady-state control input may not necessarily be zero. We describe the optimal cost over the finite horizon to be,

$$
\begin{aligned}
J^*_{0\,x[k]} = \min_{\delta\mathbf{u}_{k:N-1}} \mathbf{x}_T M \mathbf{x}'_T + \sum_k \|\mathbf{y}[k]-\mathbf{r}[k]\|^2_Q + \|\delta\mathbf{u}[k]\|^2_R \\
\text{subject to } \; \mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{w}[k], \mathbf{u}[k]) \;\; k=0,...,N-1 \\
\mathbf{y}[k] = I_3\mathbf{x}_k \;\; k=0,..,N-1 \\
\mathbf{x}[k] \in \mathcal{X} \;\; k=0,..,N-1 \\
\mathbf{u}[k] \in \mathcal{U} \;\; k=0,..,N-1 \\
\mathbf{x}_0 = \boldsymbol{x}[k]
\end{aligned}
$$

Where **r** denotes the reference trajectory passed by the CES algorithm, $\mathbf{x}_T$ is the terminal state for the horizon, M,Q,R are the penalties from deviating from the terminal state, reference state and zero respectively, $\mathcal{X}$ is the state space of positions not in collision with any obstacles and $\mathcal{U}$ denotes the set of allowable control inputs passed through $\mathbf{u}[k] = \delta\mathbf{u}^*_0 + \mathbf{u}[k-1]$.

D. Informed initialization

The goal of informed initialization is to obtain a solution similar to, or better than, the optimal CES trajectory in terms of traversal time. While reference tracking won't be able to beat the CES trajectory's traversal time on average, informed initialization can quickly lead to solutions that that shortcut vertices along the CES trajectory while obeying collision constraints. With a small enough prediction horizon, we essentially have a reference tracker that can improve the provided trajectory with a lower computational load and avoid the divergence risks that come with linearization around the reference trajectory. For this approach, we consider the simple cost function,

$$
\begin{aligned}
J^*_0 x[k] = \min_{\delta\mathbf{u}_{k:N-1}} \mathbf{x}_T M \mathbf{x}'_T \\
\text{subject to } \; \mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{w}[k], \mathbf{u}[k]) \;\; k=0,...,N-1 \\
\mathbf{x}[k] \in \mathcal{X} \;\; k=0,..,N-1 \\
\mathbf{u}[k] \in \mathcal{U} \;\; k=0,..,N-1 \\
\mathbf{x}_0 = \boldsymbol{x}[k]
\end{aligned}
$$

with the same term descriptions as above and the numerical solver is initialized with the corresponding state values of the CES trajectory.

V. Experiment

Given the optimal CES trajectory over the feasible domain defined by the CES' bubbles, the reference tracker and the informed initialization approaches both make use of the CES trajectory information to move from the start position to the goal position while obeying state and input constraints. The main parameter of interest will be the total trajectory time (i.e the time the controller has in between each point in the CES trajectory). The behavior of both approaches will be examined as the trajectory traversal time parameter is moved toward the "optimized" traversal time of the CES trajectory's speed profile. Also examined will be the online closed-loop control computation time when simulating the controller. The prediction/control horizon for both approaches is not varied.

An RRT-based planner generates a coarse traversal solution in an obstacle-filled space $\mathbb{R}^2$ from the start point to the goal point. This solution is rough, and is sampled to produce a trajectory of waypoints that are passed forward into the CES bubble generator, `GenerateBubble.m`.Bubbles are expanded out from each waypoint until they collide with an obstacle in the workspace or reach a maximum radius. These bubbles form the feasible domain for `CES.m` to stretch, smooth, and speed-optimize over. The smoothed, speed-optimized trajectory is generated only once over the course of the experiment, (as it is the master trajectory), and then it is passed to the MPC module for execution. The CES Trajectory has defined control limits and with its speed optimization, produces an optimal *traversal time* for it's execution.

If the reference tracker is able to follow the CES trajectory perfectly, we would expect it to also exhibit a similar traversal time. If control constraints are equal to the ones used in the generation of the CES trajectory, the traversal time of the reference tracker's overall trajectory should be lower bounded by the CES trajectory traversal time on average. The informed initialization approach utilizes the endpoint of the finite horizon as the corresponding point on the CES trajectory in its cost function. The CES trajectory states are then fed as initial guesses to the solution, allowing the controller to "default" to the CES trajectory and potentially find a more optimal solution as the traversal time is reduced below the CES trajectory traversal time.

VI. Results

Three iterations were used to generate the smooth CES trajectory taking the robot from $\mathbf{x}_0 = [0, 0, \pi/4]^T$ to $x_f = [10, 10, 0]^T$, see Figure 1. The trajectory consisted of $p = 24$ and the sampling time, T_s (and thus pT_s is the total traversal time) was reduced for the reference tracking (Figures 2, 3) and informed initialization (Figures 4,5) approaches until state collision constraints were violated or the trajectory diverged consistently. The resultant average fastest traversal time and controller runtime are tabulated below in Table 1. Additionally, while the 2-iteration CES trajectory is passed to the two MPC approaches, we also tabulate the algorithm's traversal time, and runtime with iteration.

Table 1: Traversal time and Runtime for RT, II approaches.

	Fastest Traversal (s)	**Runtime (s)**
Ref. Tracking	10.21	9.51
Informed Init.	8.06	3.66
CES Traj. (1 iter.)	15.36	1.55
CES Traj. (2 iter.)	**9.65**	**3.04**
CES Traj. (3 iter.)	9.19	4.63
CES Traj. (4 iter.)	9.03	5.98
CES Traj. (5 iter.)	8.98	7.61
CES Traj. (6 iter.)	8.97	9.01

*Trajectory passed to MPC approaches

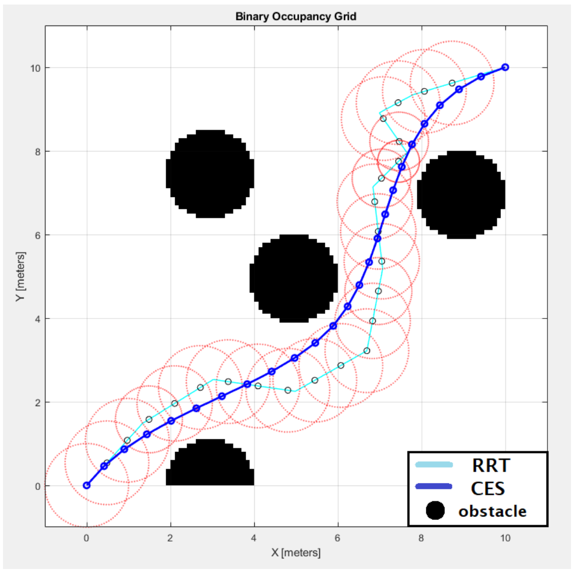

Figure 1: Smoothed and speed-optimized CES trajectory (2-iter)

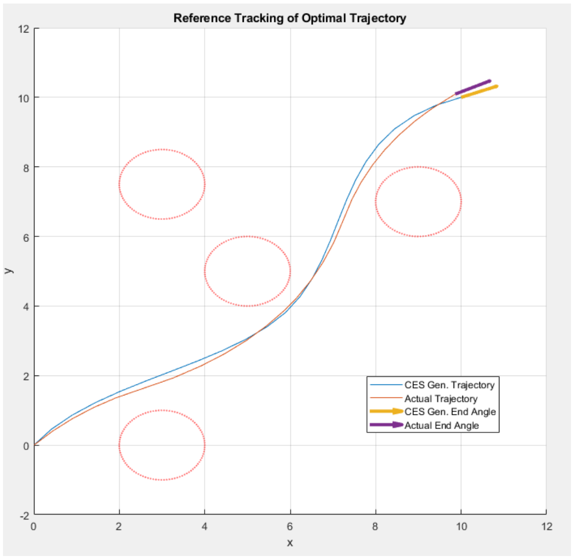

Figure 2: Fastest traversal trajectory for reference tracking

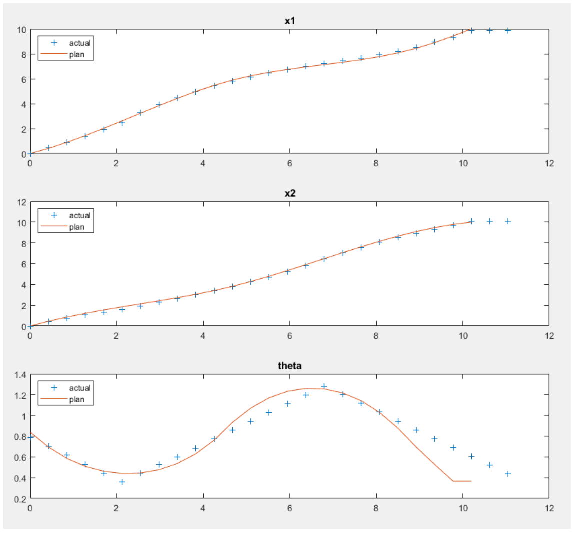

Figure 3: Reference tracking state profiles

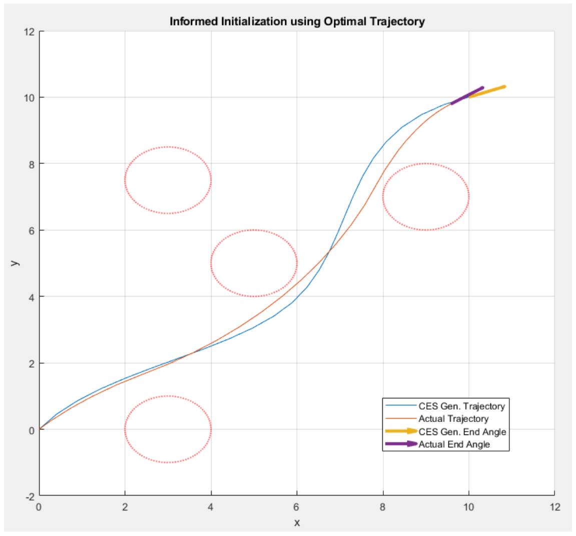

Figure 4: Fastest traversal trajectory for informed initialization

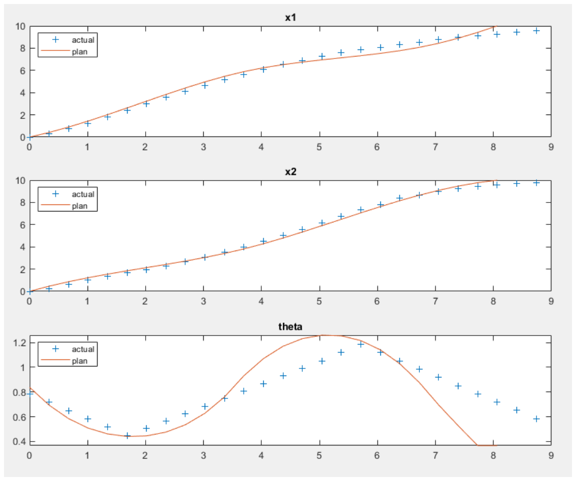

Figure 5: Informed initialization state profiles

VII. Discussion

The implementation of the CES algorithm worked as expected, as can be visually qualified from Figure 1. We can see that the CES algorithm pulls the RRT trajectory through the bubble set that defines the feasible domain toward a smoother, dynamically feasible and shorter pathlength solution. The CES algorithm was performed for up to 6 iterations and the 2-iteration solution was chosen produce a traversal time standard of 9.65s for the two other approaches due to the large diminishing returns in speedup over runtime.

The reference tracking MPC approach also appears to perform as expected, converging reasonably tight on the reference trajectory as the traversal time was reduced (Figure 2,3). Below the fastest traversal time of 10.21s, the trajectory commonly diverged or did not reach the final terminal state. Beyond this point, changing parameters such as the horizon window size and cost function penalties showed little improvement. Regardless, given the reference trajectory tracked quite well, we would expect that it's traversal time is roughly lower bounded by the standard. One reason for this is that the reference tracker will not necessarily err in the direction of reduced traversal time as overshooting and undershooting of the reference trajectory occurs. Thus, the traversal time of 10.21s agrees with the expectation that the traversal time should not be largely different from the standard. Additionally, we observe a large computational runtime of 9.51s, of which we may attribute to the numerical solver handling the a cost function more complex than the informed initialization approach.

The informed initialization MPC approach appears to perform very well compared to the reference tracking approach, and provides a substantial improvement in traversal time over the CES trajectory. The informed initialization allowed for consistent undercutting of the reference trajectory while still obeying state constraints. This resulted in a traversal time of 8.06s, much lower than even the 6-iteration converged CES traversal time. In addition to the speedup, the informed initialization approach exhibited a runtime of only 3.66s, or 40% of the reference tracking approach. One rationale for this is the simple cost function and the informed initialization jumpstarting convergence on an optimal solution.

While the informed initialization approach performs better than the reference tracker, we recognize that all of these approaches are performed over a fixed set of obstacles and initial and final conditions. From this analysis, we cannot necessarily generalize this performance over different condition sets, dynamics models or input constraints. Please see the code located in the github repository [4] if you wish to see further details on how the experiment was conducted.

VIII. References

[1] Z. Zhu, E. Schmerling, M. Pavone, ”A convex optimization approach to smooth trajectories for motion planning with car-like robots”, Proc. 54th IEEE Conf. Decis. Control, pp. 835-842, 2015.

[2] Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the driftless case. In: Proceedings of the IEEE Conference on Robotics and Automation (2015)

[3] T. Lipp and S. Boyd, “Minimum-time speed optimisation over a fixed path,”International Journal of Control, vol. 87, no. 6, pp. 1297–1311,2014.

[4] MATLAB Code Repository:
https://github.com/emmerich-martin/CES_MPC_Project/tree/master/CES_MPC_Project