Is your robot motion jerky?

Generalizing Trajectory Retiming to Quadratic Objective Functions

Gerry Chen, Frank Dellaert, and Seth Hutchinson
Georgia Institute of Technology

Georgia Tech Institute for Robotics and Intelligent Machines

ICRA2024 YOKOHAMA | JAPAN

Introduction

Trajectory Retiming:
compute a feasible speed profile to execute a path

Example Applications:
Decoupled approach to Motion Planning

Path Planning	→	Trajectory Retiming
Compute a collision-free, kinematically feasible path		*Compute dynamically feasible timestamps*

Predefined path (e.g. painting, machining)

Problem Formulation

Path: $\boldsymbol{q}(s): [0,1] \to \mathbb{R}^n$
Parameterization: $s(t): [0,T] \to [0,1]$

$$s^*(t) = \underset{s(t)}{\arg\min} \quad C(s) \quad \longleftarrow \text{objectives}$$

subject to

$$\boldsymbol{A}(s)\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T\boldsymbol{B}(s)\dot{\boldsymbol{q}} + \boldsymbol{f}(s) \in \mathscr{C}(s),$$
$$\boldsymbol{A}^v(s)\dot{\boldsymbol{q}} + \boldsymbol{f}^v(s) \in \mathscr{C}^v(s)$$

Dynamics & state/control limits

Related Works

Time-Optimal Path Parameterization (TOPP)
Minimize trajectory duration ($C(s) \coloneqq T$)

The Problem
Bang-bang solution saturates control limits
- ➛ No margin for closed-loop controller
- ➛ Cannot handle secondary objectives

Approach

Instead of maximizing speed, let's minimize quadratic objectives!

$$s^*(t) = \underset{s(t)}{\arg\min} \quad \|SpeedObjective\|^2 + \|ControlEffort\|^2 + \ldots$$
$$\text{subject to} \quad Dynamics$$
$$Control/State\ Limits$$

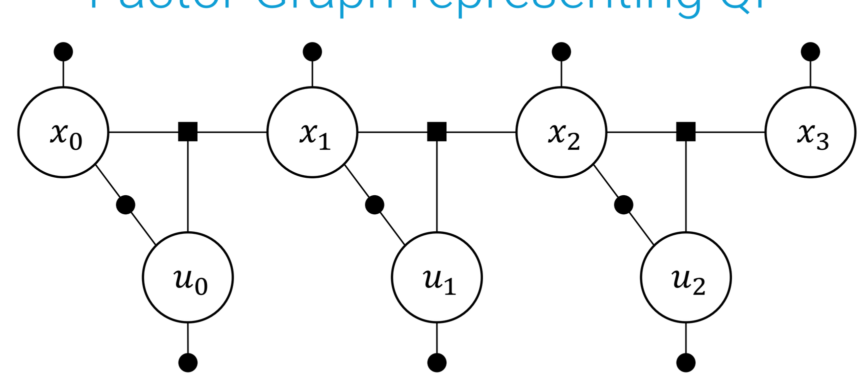
With a general quadratic objective function, we can balance multiple objectives such as max speed (min time), match target speed, max control margin, min control effort, etc.

We call this "QOPP": Quadratic Objective Path Parameterization

Sparse QP

$$\boldsymbol{x}^*, \boldsymbol{u}^* = \underset{u_0,\ldots,x_N}{\arg\min} \sum_{k=0}^{N} Q_k x_k^2 + R_k x_k u_k + N_k u_k^2$$
subject to
$$\boldsymbol{a}_k u_k + \boldsymbol{b}_k x_k + \boldsymbol{c}_k \in \mathscr{C}_k, \quad k = 0,\ldots,N,$$
$$x_{k+1} - x_k - 2u_k\Delta_s = 0, \quad k = 0,\ldots,N-1,$$
$$x_k > 0, \quad k = 0,\ldots,N$$

Factor Graph representing QP



Animated elimination procedure

How to solve QOPP

1. Transcribe into sparse QP using standard TOPP parameterization & discretization
2. Solve sparse QP using factor graph variable elimination[2]

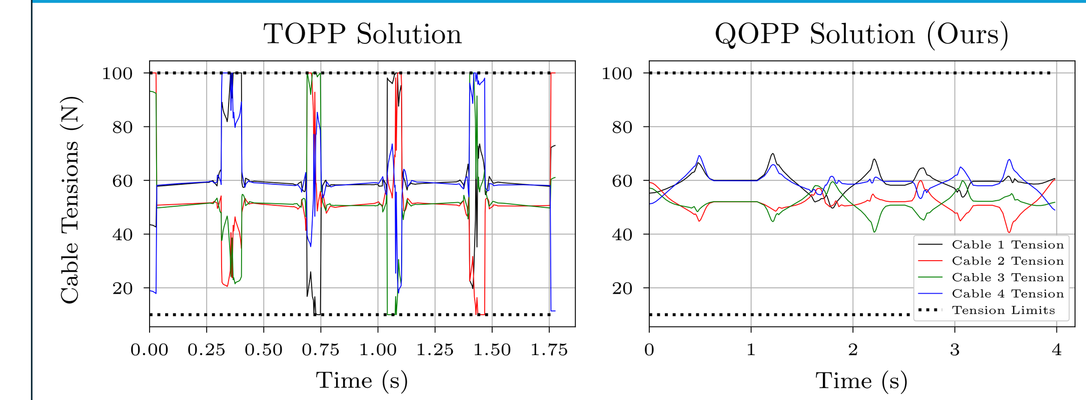
The factor graph depicts the sparsity pattern of the problem.

Performing elimination: During elimination, we only ever need to do 2 types of operations:
- Eliminate $\boldsymbol{u}_k$: Re-write equality constraint & substitute
- Eliminate $\boldsymbol{x}_k$: Solve 2-var parametric piecewise QP

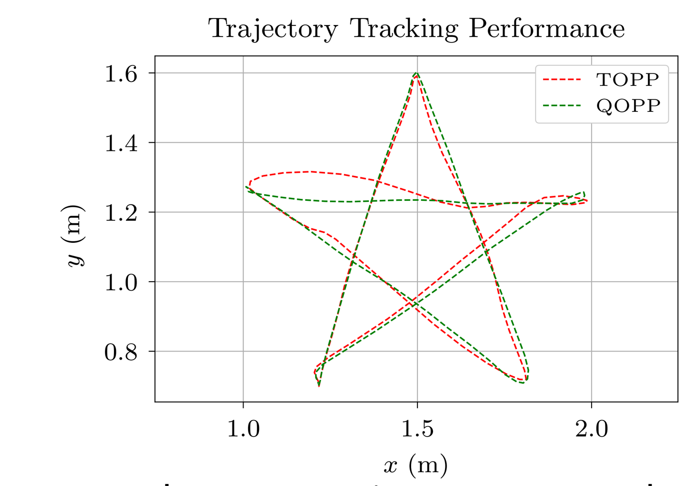
Thanks to the special structure we leverage with factor graphs,

we can solve QOPP in $\mathcal{O}(n)$ time!

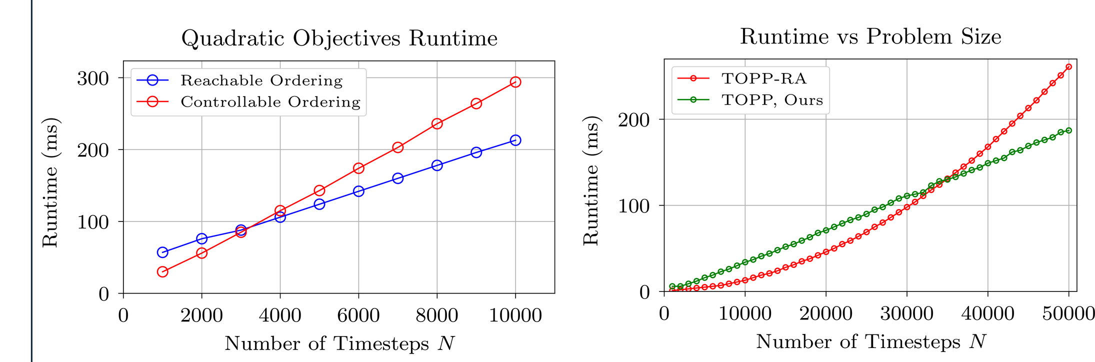
Results



TOPP constantly hits control limits. Meanwhile, QOPP allows us to trade-off speed for control margin.



As a result, executing on a real robot, TOPP has poor tracking performance (QOPP is still good).



QOPP is as-fast or faster than TOPP (C++)
Runtime is $\mathcal{O}(n)$ w.r.t. trajectory length

Conclusions

By balancing multiple objectives & constraints, in practice QOPP achieves better tracking performance and can be solved as-fast or faster than TOPP.

Select References

[TOPP-RA]: H. Pham and Q.-C. Pham, "A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis," TRO, 2018.
[2]: S. Yang, G. Chen, Y. Zhang, H. Choset, and F. Dellaert, "Equality Constrained Linear Optimal Control With Factor Graphs," ICRA, 2021.