Supplementary Materials

IEEE Transactions on Industrial Electronics,

RAMPAGE: Towards Whole-body, Real-Time and Agile Motion Planning in Unknown Cluttered Environments for Mobile Manipulators

Yuqiang Yang, Fei Meng, Zehui Meng, Chenguang Yang, *Fellow, IEEE*

Comparative Results of Different Optimal Control Solvers

In this section, various off-the-shelf solvers are used to solve the **same optimal control** problem for mobile manipulators. Solution time and trajectories are compared to show our solver's advantages. It's worth noting that our AL-DDP solver is proposed to solve the **integrated planning and control (IPC)** problem instead of a pure optimal control problem. The IPC solver needs to retrieve information (specific costs and gradients) from the map and optimize for **a collision-free trajectory and an optimal feedforward-feedback controller** to track it. On the contrary, the optimal control problem only considers **accurate tracking for a given trajectory**, which means that all costs and gradients are **independent** of the environment. Existing solvers, such as Casadi and Acados, mostly **don't** provide the API for environment-dependent costs and gradients, which makes them only suitable for the optimal control solver.

For the commonly used solvers of optimal control, we need a front-end tool for problem formulation and a back-end optimization solver to do the numerical iteration. Casadi [1] and Acados [2] are two front-end tools for problem formulation and algorithmic differentiation. **The former formulates the optimal control as a general high-dimension non-linear optimization problem using the direct method, while the latter will utilize the structure of optimal control and formulate a relatively low-dimensional non-linear optimization problem.** After the problem formulation, sequential quadratic programming (SQP) or generally non-linear programming (NLP) methods will be used to solve the optimization. OSQP [3] and HPIPM [5] are two popular and efficient QP solvers for a SQP sub-problem. IPOPT[4] is an efficient NLP solver based on the interior point method. The above tools [1-5] are all open-source and thus selected as the baselines of our benchmark.

Casadi + OSQP, Casadi + IPOPT, Acados + OSQP, Acados + HPIPM and our AL-DDP method are used to solve a stabilization problem, where the mobile manipulator is to reach a goal state with minimum tracking errors and control efforts under the control limits. The optimal control problem is:

$$\begin{aligned} &\underset{\boldsymbol{x}(t),\boldsymbol{u}(t)}{\arg\min} \;\; \boldsymbol{x}_T\boldsymbol{Q}\boldsymbol{x} + \int_0^T (\boldsymbol{x}-\boldsymbol{x_g})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(\boldsymbol{x}-\boldsymbol{x_g}) + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}\; dt \\ &\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \\ &\boldsymbol{u}_{\min} < \boldsymbol{u} < \boldsymbol{u}_{\max} \end{aligned} \tag{1}$$

where, $\boldsymbol{x} = [x_b, y_b, \theta_b, v_b, w_b, \boldsymbol{q}_m^{\mathrm{T}}, \dot{\boldsymbol{q}}_m^{\mathrm{T}}]^{\mathrm{T}} \in \mathrm{R}^{17}$ is the state vector of mobile manipulator. $\boldsymbol{u} = [a_v, a_w, \ddot{\boldsymbol{q}}_m^{\mathrm{T}}]^{\mathrm{T}} \in \mathrm{R}^8$ is the control vector. The system function $f(x)$ is:

$$\begin{aligned} \dot{x}_b &= v_b\cos(\theta_b) \\ \dot{y}_b &= v_b\sin(\theta_b) \\ \dot{\theta}_b &= w_b \\ \dot{v}_b &= a_v \\ \dot{w}_b &= a_w \end{aligned} \tag{2}$$

The parameters are set as:

$$\begin{aligned} \boldsymbol{Q} &= \mathrm{diag}(10, 10, 1, 1, 1, 1, 1, 1, 1, 0.1, , 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1) \\ \boldsymbol{R} &= \mathrm{diag}(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1) \\ |a_v| &< 3\; m/s^2 \\ |a_w| &< 3\; rad/s^2 \\ |a_j| &< 2\; rad/s^2 \end{aligned} \tag{3}$$

All python codes for the benchmark are open-source at https://github.com/yuqiang-yang/TIE-Supplementary-video. We first choose different time horizons T to test the solution time of different solvers, the results of which are shown in Fig.R1.

We can draw some conclusions from Fig.R1:

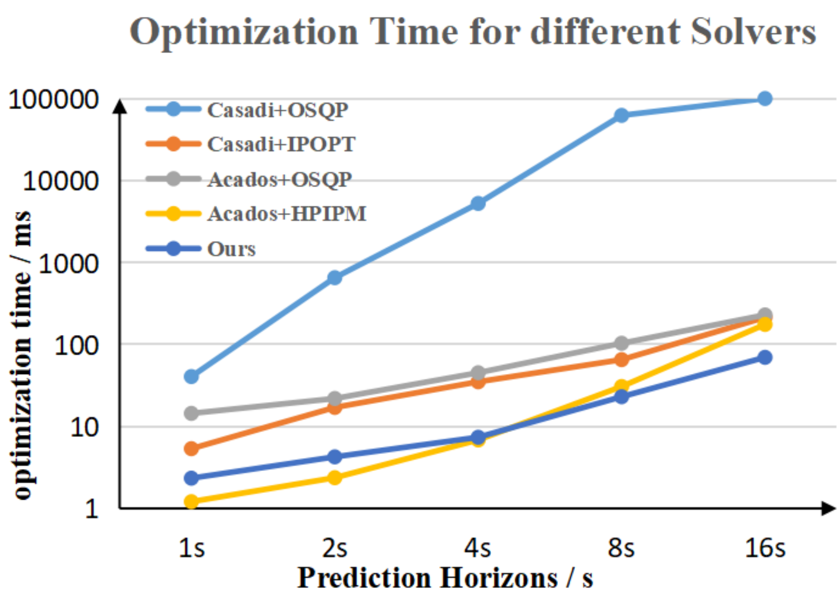

Figure R1: The optimization times w.r.t the time horizons T. In general, our AL-DDP solver and the Acados+HPIPM solver achieves the best performance.

- Utilizing the **special structure of optimal control** can significantly accelerate the solution, especially when the time horizons become larger. Acados and our method decompose the large optimization problem into a series of small ones, which makes the solution time generally smaller than the Casadi.

- Solving the optimal control problem with pure direct method and SQP solver is several orders of magnitude slower than our method. Besides, the experimental time complexity is close to the theoretical $O(N^3 m^3)$, where N is the step size and m is the state dimension.

- An excellent back-end NLP solver, such as IPOPT, can also solve the optimal control well. Even though the special structure of optimal control is **not explicitly** considered in the problem formulation, the NLP solver will implicitly utilize the sparse nature to accelerate the solution.

- Compared to Acados+HPIPM, our AL-DDP method is more flexible for planning because we can design custom environmental costs and gradients for collision avoidance or other tasks. Besides, an additional feedforward-feedback controller can be gained for free through our method.

Then we set the parameter as following to compare the solution result (Fig.R2) of different solvers:

$$
\begin{aligned}
\boldsymbol{x}_0 &= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^{\mathrm{T}} \\
\boldsymbol{x}_0 &= [1.5,1.5,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0]^{\mathrm{T}} \\
T &= 8s
\end{aligned}
\tag{4}
$$

The results show that different solvers can converge to similar minima, given the same initial state. All these methods can handle the control constraints on u and reach the target states after the optimizations converge.

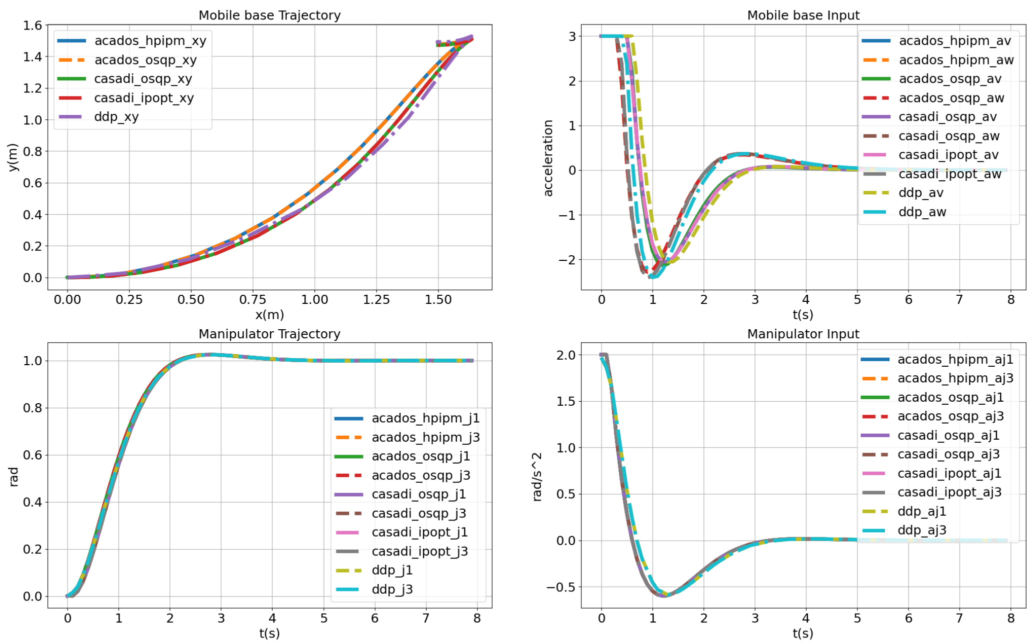

Figure R2: Optimized trajectories and controls of different solvers. (a) The optimized trajectory for the mobile base. (b) The optimal control u solved by different solvers. (c) The optimal state x for the manipulator. (d) The optimal input u for the manipulator.

Reference

[1] Andersson, Joel AE, et al. "CasADi: a software framework for nonlinear optimization and optimal control." Mathematical Programming Computation 11 (2019): 1-36.

[2] Verschueren, Robin, et al. "acados—a modular open-source framework for fast embedded optimal control." Mathematical Programming Computation 14.1 (2022): 147-183.

[3] Stellato, Bartolomeo, et al. "OSQP: An operator splitting solver for quadratic programs." Mathematical Programming Computation 12.4 (2020): 637-672.

[4] Wächter, Andreas, and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." Mathematical programming 106 (2006): 25-57.

[5] Frison, Gianluca, and Moritz Diehl. "HPIPM: a high-performance quadratic programming framework for model predictive control." IFAC-PapersOnLine 53.2 (2020): 6563-6569.