,

Monte Carlo Tree Search with Reinforcement Learning for Motion Planning

Philippe Weingertner[1] Minnie Ho[2] Andrey Timofeev[3]
Sébastien Aubert[1] Guillermo Pita-Gil[1]

[1]Renault Software Labs

[2]Zoox Corporation

[3]Experis Switzerland

ITSC, September 2020

Table of Contents

Table of Contents

Motion Planning

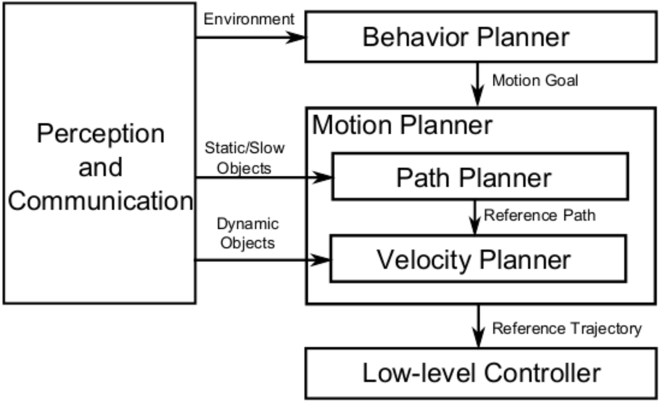

Figure 1: Decision Making and Planning

Two-Steps Path-Velocity Decomposition

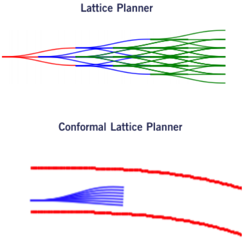

Figure 2: Path Planner

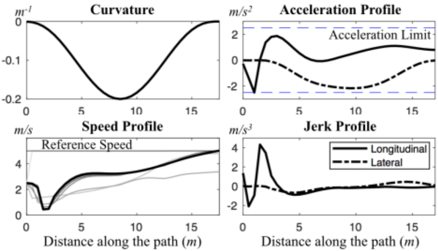

Figure 3: Velocity Planner

Avoiding multiple dynamic obstacles

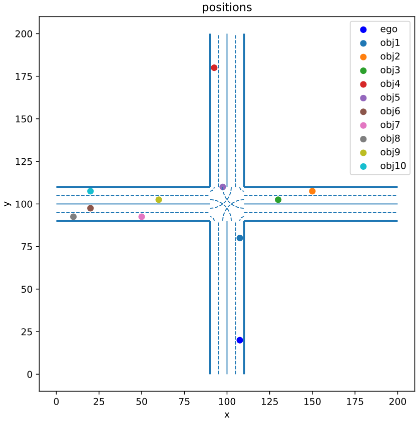

Figure 4: Multilanes Crossing

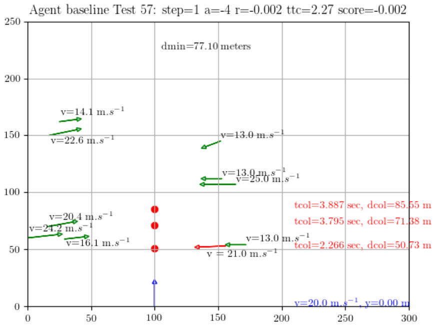

Figure 5: Benchmark Setup

Motion Planning Challenges

Dynamic Obstacles Avoidance

- Combinatorial explosion:
  - Usually 1 single dynamic obstacle is considered
  - We will consider up to 10 dynamic obstacles
- Non convex collision avoidance problem:
  - Proceed OR yield the way
  - OR constraints typically handled via Mixed Integer Programming

Real Time constraints

- Plan in less than 40 ms
- Plan a velocity profile for multiple (> 6) candidate paths

Table of Contents

Complexity Reduction via S-T projection

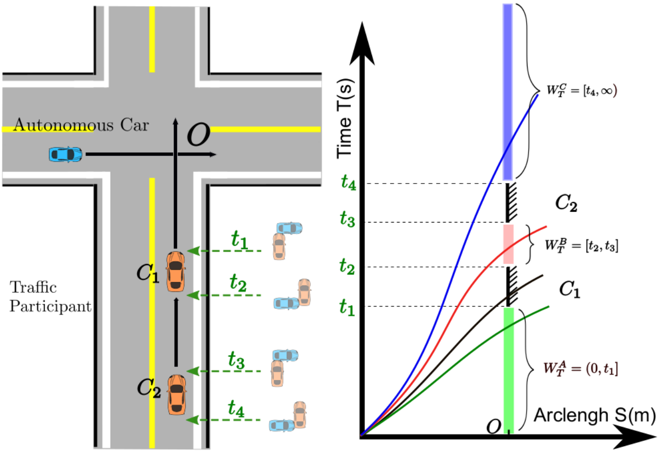

Figure 6: S-T projection

Uncertainty Handling in S-T graphs

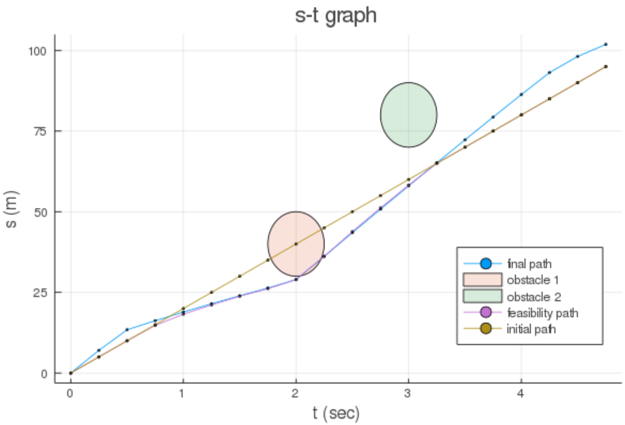

Figure 7: S-T graph

Search Model: for A*, MCTS and DQN algorithms

- States: $\mathcal{S} = \left\{ \frac{s}{s_{max}}, \frac{\dot{s}}{\dot{s}_{max}}, t \right\}$
- Desired Terminal State: $\left(\frac{s_f}{s_{max}}, \frac{\dot{s}_f}{\dot{s}_{max}} \right)$
- Undesired Terminal States:

$$\mathcal{S} \cap \mathcal{C} = \left\{ \left(\frac{s_i}{s_{max}}, t_i \right) \right\}_{i=1..n}$$

- Actions: $\mathcal{A} = [-4, -2, -1, 0, 1, 2]$ ms^{-2}
- Transitions:

$$\begin{bmatrix} s \\ \dot{s} \end{bmatrix}_{t+1} = A_d \begin{bmatrix} s \\ \dot{s} \end{bmatrix}_t + B_d \begin{bmatrix} \ddot{s} \end{bmatrix}_t \text{ with } A_d = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B_d = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

- Reward or cost model:

$$R(s, a) = -0.001 - 1 \times \mathbb{1}\left[\text{distance(ego, obj)} \leq \ d_{\text{col}}\right] - 0.002 \times \mathbb{1}\left[a \leq -4\right]$$

MPC Model: for MPC algorithms

$$\min_{u_0,\dots,u_{T-1}} (x_T - x_{\text{ref}})^\intercal \, Q_T \, (x_T - x_{\text{ref}}) + \sum_{k=0}^{T-1} (x_k - x_{\text{ref}})^\intercal \, Q \, (x_k - x_{\text{ref}}) + u_k^\intercal \, R \, u_k$$

$$\text{subject to} \quad \begin{cases} x_{k,\min} \leq x_k \leq x_{k,\max} \\ u_{k,\min} \leq u_k \leq u_{k,\max} \\ x_{k+1} = A_d x_k + B_d u_k \\ x_0 = x_{\text{init}} \\ \forall \left(t_{\text{col}}, s_{\text{col}}\right)_{i \in [1,10]} \qquad x_{t_{\text{col}}^{(i)}}[1] < s_{\text{col}}^{(i)} - \Delta_{\text{safety}} \text{ or } x_{t_{\text{col}}^{(i)}}[1] > s_{\text{col}}^{(i)} + \Delta_{\text{safety}} \end{cases}$$

$$\begin{bmatrix} s \\ \dot{s} \end{bmatrix}_{k+1} = A_d \begin{bmatrix} s \\ \dot{s} \end{bmatrix}_k + B_d \begin{bmatrix} \ddot{s} \end{bmatrix}_k \text{ with } A_d = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B_d = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

Table of Contents

Algorithms Benchmark

- Baselines: rules based
  - v1: focus on 1 single dynamic obstacle (with smallest TTC)
  - v2: emergency braking, same as v1 with maximum braking
  - v3: constant speed, no braking
- Oracles: exhaustive tree search based
  - A*
  - DP: Dynamic Programming
- MCTS: sampling based tree search
- DDQN: reinforcement learning based
- MCTS-NNET: MCTS with a Neural Network heuristic (paper)
- MPC: Model Predictive Control based
  - default: with dynamic obstacles avoidance pre-convexification
  - latest: with Mixed Integer Programming and Elastic Model (new)

Table of Contents

Average Tests Results

Table 1: Results with multiple crossing-points tests

Mean over 89 Tests	Success	Runtime	Hardbrakes	Steps	Collision Speed
Baseline v1	43%	40 μs	0	43	13.04 $m.s^{-1}$
Baseline v2	72%	40 μs	10.57	53	7.38 $m.s^{-1}$
MCTS	85%	3 ms	4.43	42	14.77 $m.s^{-1}$
DDQN	84%	300 μs	5.15	62	17.13 $m.s^{-1}$
MPC	82%	18 ms	6.22	47	8.72 $m.s^{-1}$
MCTS-NNET v1	98%	4 ms	4.79	62	19.04 $m.s^{-1}$
MCTS-NNET v2	96%	4 ms	4.89	48	18.40 $m.s^{-1}$
Oracle (A* search)	100%	6 sec	1.52	40	-

Distribution of Tests Results

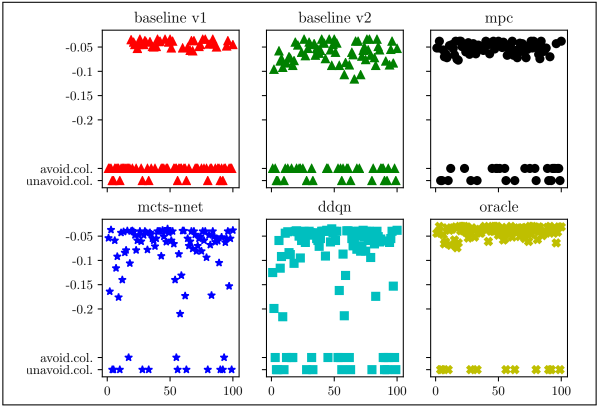

Figure 8: Scores (higher is better) for Multiple Crossing-Points Tests

Runtime vs Accuracy

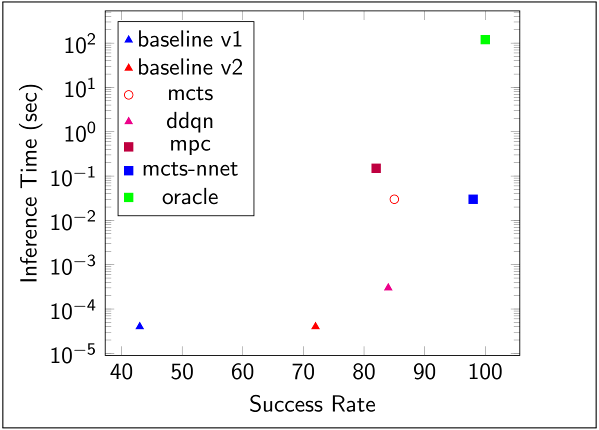

Figure 9: Runtime vs Accuracy (Multiple Crossing-Points Tests)

Very Latest MPC Improvements (not in the paper)

Mean	Success	Runtime	HardBrakes	Steps	Collision Speed
Baseline v3 (Constant Speed)	22%	$< 1\mu s$	0.0	40.0	20.0 $m.s^{-1}$
MPC_MIP	100%	22 ms	0.19	53.8	-
MPC_SIMPLEX	100%	6 ms	4.72	43.48	-
Oracle (A*)	100%	6 sec	0.48	33.1	-

Table 2: With Elastic Model and Big-M reformulation of disjunctive constraints

$$\min_{x} \quad \text{Quadratic}(x) + y$$
$$\text{s.t.} \;\; a^T x \leq b + \text{y (safety distance constraint)}$$
$$\text{elastic slack variable: } y \in \mathbb{R}$$

mpc-mip

Table of Contents

Conclusion

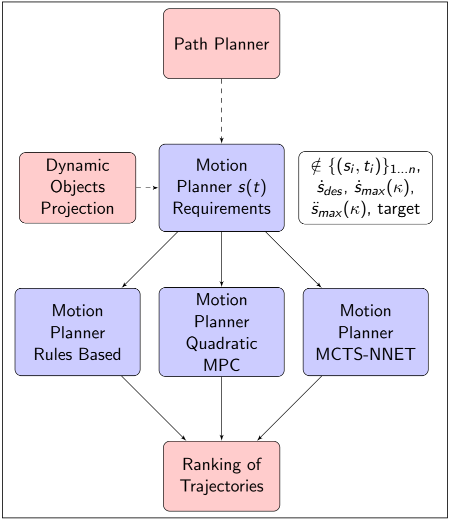

Figure 10: Motion Planner

Key Ideas and Improvements

MCTS improvements (paper)

- Faster search with a Neural Network heuristic
- Deep Learning (DL) to learn a faster version of an accurate model
- We do not depend on collected data for training
- DL to reduce computational load, not to make final decision
- Increase MCTS exploration when NNET heuristic confidence is low

MPC improvements for Collision Avoidance in Real-Time (new)

- Mixed Integer Programming to handle disjunctive constraints
- Use an Elastic Collision Avoidance Model
- Linearization to enable optimization with a faster Simplex algorithm

References I

- C. Liu, W. Zhan, and M. Tomizuka, *Speed profile planning in dynamic environments via temporal optimization*, 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 154–159.
- M. McNaughton, C. Urmson, J. M. Dolan, and J. Lee, *Motion planning for autonomous driving with a conformal spatiotemporal lattice*, 2011 IEEE International Conference on Robotics and Automation, May 2011, pp. 4889–4895.
- B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, *A survey of motion planning and control techniques for self-driving urban vehicles*, IEEE Transactions on Intelligent Vehicles 1 (2016), no. 1, 33–55.

References II

X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde, *Motion planning for urban autonomous driving using bézier curves and mpc*, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 826–833.

Tommy Tram, Ivo Batkovic, Mohammad Ali, and Jonas Sjöberg, *Learning when to drive in intersections by combining reinforcement learning and model predictive control*, 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (2019), 3263–3268.

Yu Zhang, Huiyan Chen, Steven L Waslander, Tian Yang, Sheng Zhang, Guangming Xiong, and Kai Liu, *Toward a more complete, flexible, and safer speed planning for autonomous driving via convex optimization*, Sensors (Basel, Switzerland) 18 (2018), no. 7.