Difference between DWA and TEB local planners

Christoph Rösmann

June 11th, 2018

The overall idea of both DWA and TEB is to predict/plan the motion of the robot along a given horizon while minimizing a given objective function and while adhering to kinodynamic constraints of the robot. After commanding only the first control action to the robot, the whole prediction/optimization is repeated. This principle is well known in control theory as receding horizon control or model predictive control. However, computing the optimal solution is computationally demanding and thus both approaches approximate the optimal solution in different ways and the actual optimization strategy differs.

Dynamic Window Approach (DWA)

The DWA performs a sample based optimization. It samples a control action in the feasible velocity space (usually a translational/angular velocity pair) and rolls out the trajectory for these particular sampled actions. Rollout means that the trajectories are simulated according to the specified horizon length based on the robots motion model. One important detail is: the control action is kept constant along the whole prediction horizon. Hence it cannot predict motion reversals etc. After rolling out predictions for all samples, the best candidate is selected based on a specified cost function and constraints (including distance to global path, smoothness, obstacle clearance, ...).

Consequently, DWA includes two simplifications in order to keep the computation times low while achieving a certain amount of control performance. I will not go into the details regarding stability and recursive feasibility arising from the inherent suboptimality, but the approach works pretty well for differential-drive and omnidirectional robots. A benefit is, that the cost function can be non-smooth and so it is well-suited for grid-based evaluations. E.g. trajectories can be rasterized into the costmap in order to evaluate the cost (considering lethal grid cells and inflation cost). Furthermore, the DWA does not get stuck in local minima based on its initialization (of course, the robot can still get stuck due to a limited horizon length and fewer degrees of freedom in terms of control actions). Since DWA assumes constant control actions along the prediction horizon, the control of car-like robots is rather limited. There are some basic extensions to restrict the velocity search space to car-like compatible actions. But motion reversals are still not subject to optimization and hence parking maneuvers in confined spaces are intractable. Note, of course, motion reversals can occur during closed-loop control, but they are not part of the open-loop solution / prediction.

In Summary:

- Suboptimal solutions without motion reversals (control actions are kept constant along the prediction horizon)
- Well-suited for diff-drive/omnidirectional robots, but not for car-like robots
- Supports non-smooth cost functions

Timed-Elastic-Band (TEB)

The TEB primarily tries to seek for the time-optimal solution, but you can also configure it for global reference path fidelity. The approach discretzies the trajectory along the prediction horizon in terms of time and applies a continuous numerical optimization scheme. Consequently, depending on the discretization resolution, the degrees of freedom along the prediction horizon can be very high and motion reversals are supported. Furthermore, the constrained optimization problem is transformed into an unconstrained optimization problem to gain shorter computation times. This also implies that constraints (e.g. obstacle avoidance, velocity bounds, ...) can not be guaranteed in any case, so I suggest to check emergency cases in the robot's base driver or a dedicated node. Furthermore, the optimizer only finds local solutions, e.g. if the trajectory is initialized on the left side of an obstacle it remains there. The teb_local_planner is able to optimize multiple trajectories in different topologies (e.g. left and right) at once in order to find the solution. Since the approach relies on continuous optimization, the cost function must be smooth. It cannot cope with grids and costmaps for function evaluation. Currently, every lethal obstacle cell is considered as point-shaped obstacle which limits the approach to small/mid-sized local costmap sizes (and a fairly coarse costmap resolutions). The planner also copes with polygon-shaped obstacles (see the costmap_converter package to convert the costmap to more primitive obstacles, but it is still experimental). However, given a certain amount of computational power and a mild problem size, the planner achieves a much better controller performance, resolves more scenarios and also supports car-like robot motions.

In a recent version of the teb_local_planner, support for dynamic obstacles has been introduced. The performance highly depends on the obstacle tracking and state estimation accuracy. However, the costmap-converter package tries to track dynamic obstacles from the local costmap (experimentally).

In Summary:

- The planned trajectories are closer to the actual optimal solution, but constraints are implemented as penalties only.
- Suited for all robot types
- Planning of multiple trajectory candidates in multiple topologies
- Dynamic obstacle support (experimental)
- Large computational burden

References

[1] D. Fox, W. Burgard, S. Thrun: The dynamic window approach to collision avoidance. In: IEEE Robotics & Automation Magazine. 1997, pp. 23–33.
[2] C. Rösmann, F. Hoffmann, T. Bertram: Integrated online trajectory planning and optimization in distinctive topologies. In: Robotics and Autonomous Systems, vol. 88, 2017, pp. 142–153.