RLSS: Real-time Multi-Robot Trajectory Replanning using Linear Spatial Separations

Baskın Şenbaşlar[1], Wolfgang Höng[1,2], Nora Ayanian[1]

Introduction

RLSS computes trajectories in real-time. It
1. explicitly considers dynamic limits of robots,
2. does not rely on a central computer,
3. handles agent failures and compensates poorly performing controllers,
4. does not rely on communication for safety,
5. requires perfect sensing of positions of other robots and obstacles,
6. avoids deadlocks (only experimentally),
7. safety using hard constraints and reports back if it can’t ensure safety,
8. and works in the presence of obstacles.

Approach

Goal Selection

Select a goal on the desired trajectory to plan towards: Given a desired
planning horizon, compute the actual planning horizon such that the desired
trajectory is collision-free at the actual time horizon.

Discrete Search

Plan a discrete path to the selected goal position: We use A* search to
compute a discrete path from the current position to the goal position. In A*
search, forward moves and rotations have costs of 1. There are also actions to
to enter the grid and leave the grid. The desired number of segments is given as a
hyperparameter and the segments are split if there are fewer segments than
desired. (Segments with blue end points in the image are computed by A*
search, since the desired number of segments is 6, red end points are
introduced by segment splitting.) If there are more segments than desired,
segments from the end are omitted. We also compute segment durations in
discrete search.

Trajectory Optimization

Compute a smooth trajectory over discrete segments: We fit a piecewise Bezier curve on discrete
segments using one piece per segment.

Enforcing Robot-to-Robot Collision Avoidance
To enforce robot to robot collision avoidance we
compute SVM hyperplanes among robot collision shapes. We constrain the first piece to stay inside the
SVM half-space. We enforce that the duration of the first piece is more than the planning period.

Enforcing Robot-to-Obstacle Collision Avoidance: We compute the SVM hyperplanes between each
discrete segment and obstacle. We constrain trajectory pieces to stay in the computed SVM half-spaces.
We also add constraints for continuity between trajectory pieces and between planning iterations.

Validity Check

Check if the resulting trajectory is valid: We impose derivative magnitude limits on the trajectory in the validity check. If the
magnitude of any derivative of any desired degree is more than the maximum amount, the trajectory is invalid. To compute the
maximum derivative magnitude along the trajectory, we execute linear search with small increments over the duration of the
trajectory. If the resulting trajectory is invalid, we increase the piece durations and re-run the optimization.