Multi-segment Path Specifications and Desired State Generation

Wyatt Newman
February, 2015

To apply a steering algorithm, one needs to compare a desired state (e.g. x, y and heading) to a measured state, and from the disparity derive an appropriate response (speed/spin commands to perform corrections). The measured state can use the "odometry" topic, which is updated frequently and smoothly (avoiding discontinuous jumps). The desired state needs to be generated in real time from a path plan.

A path specification does not contain timing—only shape. A path might be generated by considering a balance between safety (keeping an assured distance from collisions) and efficiency (leading to a goal in an acceptably short path length). A path may be represented, e.g., as a rendering in Rviz.

A "trajectory" is an augmented path. It prescribes the time-history of a desired journey along a path, including Pose (position and orientation) as a function of time and Twist (velocity and angular velocity) as a function of time. A node can be responsible for generating the trajectory by frequent (and timely) publication of the desired "state" (pose and twist) such that sequentially published desired states lead a robot along a specified path while respecting dynamic constraints (including constraints on velocities and accelerations). Preferably, publication of desired states would specify poses with respect to the odometry frame, and these publications would be updated frequently (comparable to the update rate of odom). With these qualities, steering calculations would not suffer jitter or jumps due to low sampling rates or discontinuous commands.

The job of the desired-state generator is to provide a viable stream of desired states, suitable for use by a steering algorithm.

Path specification: ROS has extensive support for navigation, known as the "nav stack" (see: http://wiki.ros.org/navigation). In the nav stack, a planned path is conventionally communicated as a message of type nav_msgs/Path.msg, which contains a vector of (stamped) poses. These poses are of type geometry_msgs/Pose, which is in turn comprised of a point and a quaternion. In short, a ROS path is equivalent to an ordered list of frames.

This style of path specification may not be suitable for immediate execution. Additional refinement may be required to achieving a desirable trajectory. Nonetheless, a path of thus specified can be followed by a robot.

A ROS path description is equivalent to a polyline in 2-D: a sequence of line segments connected at listed vertices (origins of successive frames) in a plane. In such a description, vertices of the polyline correspond to discontinuities of heading. Since a real robot has inertia, instantaneous change in heading is impossible. If the robot attempted to maintain constant speed through a vertex, it would require an instantaneous change in momentum, which would require infinite force and infinite torque.

To follow a polyline precisely, the robot must start each line segment from rest, then come to a full stop at each vertex. At a vertex, the robot must spin in place, ramping the angular velocity up from zero and back down to zero. After a spin-in-place—which results in the robot aligned with the next line segment—the robot can begin to ramp up its translational velocity to travel along this line segment. In some instances, this can be a reasonable strategy. With reference to the 2nd floor of Glennan, starting in front of the elevator and pointing towards the lab door, the first path segment may proceed forward, come to a full halt approximately 2m in front of the lab door, spin in place by

-90 deg, then proceed down the long hall.

A faster trajectory would avoid coming to a complete halt. Instead, a line segment may blend into a circular arc, and hence to another line segment. This would allow continuous motion by rounding the corners of a polyline. This desirable extension is deferred for the present while we consider polyline paths.

Desired state generation for a line segment: A polyline is comprised of line segments, specified in terms of a start vertex and an end vertex. Equivalently, the line segment can be described in terms of properties: starting vertex (a reference point); tangent angle (heading from start to end vertex); and a length (distance from start to end vertices). Optionally, a segment may have associated speed limits (max and min speed) and acceleration limits (max and min accelerations). (Alternatively, one might have uniform speed and acceleration limits applicable over the entire path).

Since a polyline has each line segment start from rest and end at rest, one can compute a velocity profile (triangular or trapezoidal) for the path segment that conforms to the dynamic constraints. Within a timed loop, the desired-state generation node can perform the following:

- Initialize time t=0; initialize distance_to_go = segment_length; initialize speed command =0; initialize spin command=0; compute **t**, the unit tangent vector for the current segment
- while (distance_to_go>0)
  - update velocity command (e.g., from trapezoidal profile)
  - update the desired (x,y) position as: $\boldsymbol{p}(t+\Delta t)=\boldsymbol{p}(t)+\boldsymbol{t}\, v\, \Delta t$
  - publish the desired state, including position, heading, speed and angular velocity (spin)
  - sleep for time Δt

The above algorithm will output updated desired states (equivalently, poses and twists) every Δt seconds, with speeds and accelerations that are feasible, terminating with the robot braking to rest at the end vertex.

An important extension of the above is that the trajectory generator should be aware of disruptions, such as an E-stop, reflexive halt to prevent collision, or a software halt command. For any such disruption, the desired-state generator would need to re-initialize its loop, starting from the current pose of the robot (treating this as the new “start” vertex).

Desired state generation for spin-in-place: to be continued

Desired state generation for circular arcs: to be continued

Augmenting polyline paths with blended, circular arcs: to be continued