## Mobile Robot Navigation and ABBY

One of the major tasks for a mobile robot is navigation through its environment. In order for the robot to get parts from inventory, it must first travel through the inventory shelves to the location of the parts. In an industrial application, the location may be retrieved from an inventory database, or it may be specified by a human operator, but the robot's task is the same. From its current location, the robot must plan a path to another location in its environment. The path must avoid obstacles, and it should be as direct and efficient as possible. The robot must then generate a trajectory to follow the path and travel to the goal location. The trajectory cannot violate the dynamic constraints of the robot. Given a trajectory, the robot must execute it by controlling the actuators as accurately as possible to adhere to the desired path and trajectory.

ABBY's differential drive system allows it to move forward and backward and rotate in place, but not move laterally. This makes navigation and control somewhat more difficult than for a holonomic drivebase such as the caster drive system on the Willow Garage PR2.

At the simplest and lowest level of the navigation task is speed control. Speed control on ABBY is implemented as a pair of PID controllers, one for each wheel. The PID controllers are implemented on the cRIO's FPGA for speed and robustness, with loop closure rates of 100 Hz. Each PID controller's setpoint is specified in meters/second and its output is an 8-bit signed integer. A simple geometric algorithm (equation ref), implemented on the cRIO's PowerPC processor, is used to convert twist-style commands (forward and rotational speed) into speed commands for each wheel. These signed integers represent the desired voltage to be output by the Sabertooth motor controller, with -127 being full reverse and 127 being full forward. Since the Sabertooth motor controller can vary its voltage output from -24 volts to 24 volts, the 7 bits of speed resolution in each direction correspond to a voltage output resolution of about 189mV. The PID controllers on ABBY were originally tuned for another similar robot based on the same drivetrain known as ALEN (cite ALEN paper). ALEN was significantly lighter than ABBY and also had a different weight distribution. As a result, it is likely that ABBY's PID controllers are not optimally tuned. This sub-optimal tuning makes it impossible for ABBY to execute low-speed commands because the controllers do not command a high enough voltage to the motors to overcome static friction in the drivetrain. As a result, ABBY's minimum achievable forward/reverse speed is SPEED m/s and minimum rotational speed is SPEED rad/sec. This compares unfavorably to the minimum speeds that Eric Perko was able to achieve on HARLIE, which were 0.1 m/second and 0.1 radians/second respectively.

The higher-level components of navigation are path and trajectory planning. Path planning is the task of determining a path from the robot's current location to a desired pose. Trajectory planning takes the path and determines a series of velocity commands to move the robot through the path without violating the acceleration and velocity constraints of the robot. On ABBY, these tasks are performed by a global and a local planner, respectively.

NavFn\cite{navfn} , the global planner node, operates on a grid-based global costmap populated by the *a priori* map and data from the LIDAR. Given a desired pose, NavFn finds a minimum-cost path using Djikstra's algorithm\cite{djikstra}. This path is defined as a series of intermediate "breadcrumbs," robot poses along the path. NavFn can successfully plan paths for ABBY in relatively open environments, but because it assumes a circular robot base, it will sometimes plan impossible paths in crowded environments.

DETAILS AND CHARACTERIZATION OF NAVFN

The local planner generates trajectories to follow the path produced by the global planner; it operates on a local costmap populated by data from the LIDAR. The robot performs local planning using a dynamic window approach,\cite{probabilistic-robotics} which forward-simulates translational and rotational velocities and evaluates the resulting trajectories for proximity to obstacles, proximity to the goal, and adherence to the global path. These scores and weighted and summed to determine the trajectory's score. The highest scoring velocity command is sent to the mobile base driver. On ABBY, dynamic window planning sometimes results in unintuitive behavior as the robot approaches the goal. Namely, the robot will sometimes rotate the wrong way, forcing it to turn all the way around to reach the proper heading.

DETAILS AND CHARACTERIZATION OF NAVFN

There are some alternatives to NavFn and the base local planner packages used on ABBY. The ROS navigation stack includes a global planner called Carrot Planner \cite{carrot\\_planner} which does not attempt to navigate around obstacles. Instead, it moves as close as possible to the goal along a straight line until it encounters an obstacle, then stops. This planner is only useful in very open environments where a straight-line path to the goal is likely to exist or obstacles are likely to move out of the way for the robot. As such, the planner is not suitable for most industrial environments, which are full of permanently fixed machines, assembly lines, and shelves. In his masters thesis,\cite{perko} Eric Perko of Case Western Reserve University addressed many of the problems with NavFn and base local planner and devised new algorithms for precision navigation of a mobile robot or wheelchair in an indoor environment. However, Perko's ROS implementations of his algorithms do not conform to the same API as existing ROS navigation nodes, nor do they provide the same functionality. Whereas the existing ROS navigation stack takes an arbitrary Pose (x, y, theta 2D coordinate) as a goal, Perko's path planner requires that all goals be predefined points in an a priori map. Furthermore, the path planner requires that path segments between the goals be predefined. In order to use this global planner in an industrial environment, every possible desired position in the inventory would have to be predefined, as well as a graph of paths between positions. This set-up task would be monumental in a large factory, so Perko's global planner was not used for this project. Perko's local planner uses a combination of local path linearization and a third-order steering algorithm to generate velocity commands. Unlike base local planner, which takes in arbitrary paths composed of a series of poses, Perko's local planner requires that paths be defined as a series of line segments and constant-curvature arcs. This prevents it from being interoperable with NavFn.