Team Fifteen Paper

Overall Strategy

Team fifteen took a cue from last year's winners and decided that -whatever comes up, our robot needs to be constantly moving. In order to do this we implemented fast vision and control algorithms and a threading structure that allowed for simultaneous processing. We chose a simple behavioral approach that can accept goals from steady state machine layouts. This allowed the possibility to use mapping if it was deemed applicable, as it was after more planning.

The mapping round offered a chance to make a very accurate map of the arena (assuming everything was calibrated correctly and all error was attempted to be accounted for). Using this map is was then possible to set waypoints in the contest arena. These waypoints were goals for the robot. It would attempt to reach these waypoints, then search for balls in a wall following/object avoidance mode. We decided that our bot would not have any real "scan for balls" behavior because it could be implemented to can as the bot was naturally moving and/or avoiding walls. By setting these waypoints, it is possible to guarantee that the bot cover the entire contest arena during the scoring phase of competition, and thus not miss any balls.

As far as scoring was concerned, team fifteen chose to place balls through the "field goal," or over the wall over the goal. This method of scoring landed our team the most number of points per ball, and it also separated the collection and scoring areas of the robot. This strategy combined with an accurate map would wreak devastation on other teams.

Mechanical Design and Sensors

A very distinct characteristic of our robot is its laser-cut chassis. As soon as the team agreed on a general strategy and plan of attack, we began to design our ideal robot using CAD software. Over the course of a few days, the robot was completely designed in Rhino 3d. Rhino is an inexpensive CAD package similar to Solidworks. Several design cycles were completed until the final model emerged.

Having a design on a computer screen is a far cry from having a working robot sitting in front of you. We virtually disassembled the Rhino model into flat plates that could be cut with the laser. Then, the silhouettes of these shapes were projected onto a plane and exported in a standard .dwg format for processing. We placed beautiful white acrylic plastic on the laser-cutter's bed and clicked print. A few hours later a real robot was sitting on the ground, ready to be programmed. The design cycle we chose not only allowed us to have a completely built robot only a few days into the contest, but also gave us the ability to focus more on software while other teams were troubled with mechanical issues.

The robot is completely round, with no part of it extending beyond the circular shape. This ensures that the robot can spin in place and not worry about bits of robot getting caught up on walls and goal posts. The bumper design was based on the bumper from the Roomba Robotic Vacuum. iRobot spent years developing bumper geometry with a wide range of motion and maximum impact absorption. By replicating that design, we were able to use the benefits of years of commercial research. Needless to say our robot never got stuck on a wall.

The ball lifting mechanism is quite simple. It consists of a rotating drum and a vertical conveyor belt powered by a small DC gear motor. The drum is made of foam pipe insulation and is positioned so it will have positive interference with the balls so they are 'sucked' into the robot. The vertical conveyor belt is made of carpet liner and rides on wooden dowel rods covered in foam pipe insulation. This creates a compressible surface that is ideal for lifting wooden balls.

The balls are stored on the top of the robot until it's time to score. A solenoid holds a spring loaded gate arm down while the robot drives around. When the robot is ready to score, power is simply cut off from the solenoid, releasing the spring loaded gate and allowing the balls to roll down the slanted top and into the scoring bin.

Software Design

The software for our robot was modeled after standard robot sensor input software architecture. A constant thread runs in the background pulling updating data from the Orc board at 50 Hz while time stamping each data input so that upper level software threads know not only what data was read, but when it was read as well.

From here, the software architecture has two other threads running under the main behavior. A thread controlling dead reckoning navigation updated the robot’s position at 5 Hz. Each loop through, the thread would read the raw gyro data, then using calculated quadratic regressions, adjust for gyro drift to within 2 degrees a minute. Odometry data was then integrated with gyro data, and from the two, an extremely accuracy X, Y position could be derived. The robot, in running a course over three minutes, would only lose about 6 inches in accuracy.

The other loop running behind the main behavior control was the sensor thread. The sensor thread controlled all low level threshold triggers. It directly controlled private data Booleans such as LeftBumper and RightBumper, and converted analog values from the IR sensors to distances in feet. From the data calculations, higher level behaviors could access its senses without having to worry about overburden the orcboard. The low level sensor thread also includes some low level behaviors such as obstacle avoidance. If bumped from the left side, the robot would involuntarily stop, then backup three inches, and right a few degrees. Similarly, the robot turns left when bumped right.

Behavior-wise, the robot was simple, it would map the course by wall following, then, in the second round, it would follow walls by tracing its map, and stray to balls that it sees. The robot's mechanical features allowed ball following to be quick and imprecise; the robot needed only to roll over the ball and didn't need to stop and calibrate based upon the ball's location. In ball chasing mode, the wheel speeds are proportional to the offset of the ball from the camera center. Overall, the philosophy behind the robot behavior was keep it simple, keep it working.

Overall performance

On a whole our robot performed well. Although we did not score a very large number of points, three compared to the winning team's 25, most of the components of "The White Castle" functioned as intended. Those problems we did have were simple mechanical issues that could have been worked out given a bit more time.

One of the parts of our robot which worked extremely well was our mapping program. During the exploratory phase, our robot wall followed very closely, making an accurate map of the playing field. It completed this map within a short period of time, about one minute, meaning that even in a plying field with a greater perimeter, our robot would have finished mapping well within the three minute exploratory period. Although we didn't have enough time to make real use of this map, other than to show a pretty picture at the competition, with a bit more work, such an accurate map could be very useful. In this case, the credit goes to Peter Lai, the mastermind behind the map. Another aspect of our robot which worked out very well was the laser cut Plexiglas body. The precision cut pieces ensured a robot which was sturdy and well proportioned, eliminating most of the human error which could have occurred if pieces were hand cut. This also created a very sleek look which, combined with the golden crown adorning the top of our robot, won us the Best Dressed award.

The ball collection mechanism was one area that we should have worked on more. During the first run of our robot, we forgot to turn on the roll bar that pushes balls into the robot, leading a ball to become lodged underneath our robot, preventing it from moving. This also meant that none of the balls collected were lifted to the upper level. Still, this was less of a problem with or robots design and more with our own carelessness. Thus we were given a second chance, during which our robot successfully collected balls, pushed them back to the elevator, and raised them to the upper level. However, as the balls passed though the friction drive, the base of the drive was gradually pushed downward, causing it to catch in the carpet and bring our robot to a disappointing halt.

One more crazy awesome feature of our robot was our ball following code. During the practice round, the software guided the robot in such a way that it collected nearly all the balls in the arena, a performance that put it above the other bots at that time. Had our robot not gotten stuck during the actual competition, we can only assume that it would have performed equally well.

Conclusions / Suggestions

Without threading the robot will be to slow to successfully complete the tasks needed within the competition time limit. Threading is absolutely necessary. Plan your threads with thought though, and remember that even though something is threaded, it is still sharing CPU power with the other threads in the program. A CPU intensive thread will slow down the rest of your threads.

There is certain logic in the scoring / software of the robot. If you can score very well -- that is, create an accurate map and actually implement this map for scoring purposes -- you may not need to score the most number of points, as this requires some sort of lifting mechanism to get the balls to the top of the robot. Time is tight with the competition, and a good map takes time to get, so scoring three points with every ball may be better than 5 with only 3 or 4.

There should be a checkpoint where the bots must score a three or five points. This will force teams to get their goal code working earlier, which tended to hurt many teams, as the code did not tend to function very well. By forcing the teams to find the goal, at least the bots will be able to score and make the competition more fun to watch.