Team Twelve Paper

Coming into Maslab with very limited programming experience, we are quite satisfied with what we accomplished over the course of these four weeks in January. It was very exciting to attend lecture and to learn how to incorporate some of the theories that we have learned in previous classes. This class incorporated many studies at MIT which would not have mixed otherwise. The process of building a robot requires knowledge of software, hardware, and mechanical design, which are not currently taught exclusively within one specific major at MIT. Because each member of our team majors in a different area -- Environmental Engineering, Mechanical Engineering, and Electrical Engineering and Computer Science -- we each learned a lot about the whole process of robotics as well as each others' areas of specialty.

One thing Maslab has taught us is that expectations and goals should be difficult, yet realistically attainable within limited time projects such as this. Like all teams, we came in with many wild ideas about what our robot would look like, and what it should do. Our initial idea during the first week was to build a robot resembling a bull that will go after red balls. We even planned on playing music from Carmen in the background whenever the robot was going after the ball.

Overall Strategy

Our initial strategy was simple; we wanted to score as many points as possible within the three minute round. However, as we tried to implement this strategy over the month of January, we slowly realized that some aspects of this was much more difficult than previously anticipated and thus, our strategy involved into a much more simple, but only slightly less effective idea.

Phase One:

From examining the various scoring options and point values associated with each possibility, our initial strategy capitalized on the points advantage from scoring field goals. Because scoring a field goal offered 5 points in comparison to 3 points from scoring through the hole, we decided that our main focus would be scoring through a field goal.

This initial plan involved three tasks of varying difficulty. First, the robot had to be able to find and capture balls. Second, these balls had to somehow be transported upwards towards a holding bin. And finally, third, the robot had to be able to find goals and release the captured balls into the field goals.

Phase Two:

Once it became apparent that the initial strategy would be unattainable given the amount of time left and our novice experience in programming, the strategy evolved into a simpler gathering and bulldozing

mechanism. This merely involved two tasks: finding balls and capturing them, and finding goals and releasing balls into the mousehole.

Mechanical Design and Sensors

Following our phase two strategy, the mechanical design became a simple bulldozer-type robot that would catch balls into a holding area, and drive them into a mousehole.

Chassis

We wanted a simple yet effective design, something which would not demand much precision on the software side. However, it needed to be able to work well, allow for slight errors in robot trajectory, and still score effectively. For this we chose a dual layer chassis with a rounded front, and two recessed areas in the back to prevent the wheels from protruding outside of the robot body.

In order to keep our robot within the size limit, and to maximize available space, we built two layers. The spacy, rounded top layer held all important components of the robot, including the computer, the Orc Pad controller, the Orc Board controller, the battery, the camera, and three infrared distance sensors. The lower layer of the robot was comprised of a large hole -- the ball hold -- in the main area of the robot, a gate at the front, and some space near the back on which the recessed wheels were mounted.

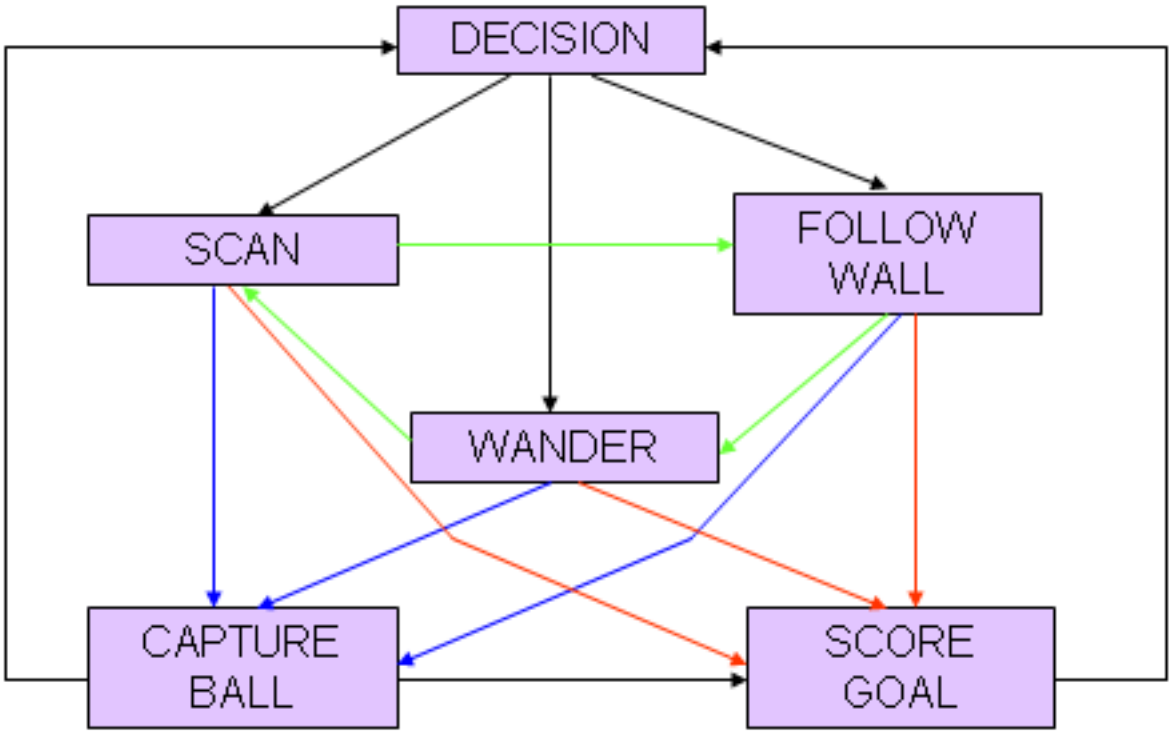
Sensors

Our robot used a total of four sensors and one camera to provide input to our software. One short-range infrared sensor was mounted on the front while one long-range infrared sensor was mounted 45 degrees each on the left and right sides. The camera was located at the front of the robot. Finally, a simple bump sensor was mounted on the back of the robot.

Software Design

The software structure is a Finite State Machine that is implemented through a large switch statement based on sensor input and the current state. We decided to have 6 basic states for the robot to pass through: decision, scanning, follow wall, wandering, capture ball, and score goal. Each state would either accomplish its task, time out, or find something (a ball or a goal), in order to exit from that state.

When the program is launched, our robot starts in the decision state. It will move forward a tiny bit, take input from its sensors to establish its surroundings, and then make a choice of either scanning if the surroundings are clear, following a wall if there's a narrow corridor, or wandering if there's only one wall.



Because the algorithms for finding balls and finding goals are so similar, we also decided to incorporate two modes into our finite state machine. The default mode is Ballmode, while the other is Goalmode. On the model, the green arrows represent changes of state from our timing out mechanisms, the red arrows account for the recognition that the robot has found a ball, and the blue arrows depict changes of state that result from the recognition that the robot has found a goal.

Once the robot has captured 4 balls, or if there are fewer than 45 seconds left in the round and the robot has at least one ball, the robot will switch from Ballmode into Goalmode, go back to the decision state, and start the goal searching process. We wrote our code in such a way that scanning, wandering, and follow wall would either look for red balls or look for yellow goals depending on the current mode of the robot. That way, a lot of code is saved and reused.

Overall Performance

Unfortunately, our robot did not manage to score any points in the final competition. It detected a ball, went towards the ball to capture it, but veered off to the side and missed. The code worked as we expected for a minute and a half, but then experienced emergent behavior when it started to slowly rotate non-stop. This behavior continued until the end of the round.

However despite the unexpected performance at final competition, our robot was able to pass Checkpoint Two, as well as capture balls and score through a mousehole countless times as we were testing. So we are convinced of its ability to function as programmed.

Conclusions/Suggestions for future teams

Our team made a significant amount of progress during the month -- we went from almost no knowledge of the topics, and particularly, little programming experience, to a robot that was able to locate and capture red balls and put them into the goal, at least some of the time.

In retrospect, we realized that we could have done better to prepare beforehand. Knowing that none of us was familiar with Java programming, reading the tutorials and learning a little Java would have made the first week much easier.

It is also apparent that group discussions are vital to additional progress. Not only is it important to keep everyone up to date on the latest software changes and method implementations, it also gives us a great chance to brainstorm new ideas and to make sure that we keep track of what has been done and what still needs to be done.

Nevertheless, our team is very proud of what we have accomplished within the four weeks of IAP. Maslab is a great class that has challenged us and allowed us to put to use the theory we have already learned in our majors. Without challenge, there would be no progress.
