Team Eight Paper

Software Strategy

CONTROL

Our robot was controlled by a Finite State Machine (FSM) that had four main states: Find Ball, Chase Ball, Find Goal, and Chase Goal. Both "Find" states used the same lower-level functions to implement the wandering algorithm, and both "Chase" states used the same helper functions to chase down an objective that was in sight.

The wander functions used a random walk to navigate the playing field. The random walk used its own FSM, the basic structure of which is as follows: the robot would move, forward or backward, for a random time or until it hit a wall. If it timed out, it would proceed to the turn phase. Otherwise, if it bumped, it would reverse direction for a short time before turning. The turn phase was similar: turn, left or right, for a random time or until hitting a wall. If it timed out, it would reverse the turn direction and go back to the move phase. If it bumped, it would reverse the turn direction before moving again.

This algorithm proved to be quite robust and resistant to getting stuck. The integrated use of the bump sensors allowed us to quickly determine, fairly reliably, whether we were stuck. And if our bump sensors failed, then we would time out and would eventually, due to the random nature of the exploration, get ourselves out of trouble. This ability was extremely important because of our bot's large size: our robot was big, had its center of rotation in the rear, and had square corners. All in all, these factors combined to make it very easy for our bot to get stuck, so we focused our efforts in the code on minimizing the chance of a fatal error.

As for the chase code, this was fairly simple: if the desired object is in sight and to the right, turn right. If it is in sight and to the left, turn left. If it is in sight and in the middle, drive straight ahead. If it is out of sight, give up and search again. If, while driving straight, a forward bump sensor is triggered, check which side the sensor was on. Then, back up, move a little to the opposite side, and chase the target again.

This behavior proved to be very robust and efficient in chasing down balls and mouse holes. If it could ever see a ball, it would move quickly towards it; the avoidance code that monitored the bump sensors allowed it to quickly maneuver around obstacles in its pursuit of the target. All in all, these functions were pretty successful, at least during practice. In our best run, we managed to collect eight balls into our storage bin, wandering from ball to ball, bouncing off walls and avoiding obstacles.

VISION

Our vision algorithm was well suited for our needs. As we wandered around the playing field, we continuously took images with the Logitech web cam. We processed these images and from them deduced the location of visible balls and field goals.

Image processing involved 2 steps. First we converted the images we took with the web cam from the RGB color space to the HSV color space. This allowed us to more easily distinguish the colors we were looking for. Next we searched for the blue strip on the top of the walls by scanning across down from the top in columns. Once a pixel was determined to be blue, we colored everything above that pixel white. This allows us to ignore everything that is not inside the playing field and prevents the rest of the vision algorithm from being confused with colors outside the playing field.

After we processed the images, we were ready to search for red balls and yellow field goals. We scanned across the image, keeping track of the width of each ball or field goal we came across. We then return the position of the object with the largest width. For red balls, we returned the angle from the center of the closest ball to the center of the entire image. 0 was the center of the image and the right half of the image was set to positive radians and the left set to negative radians. For field goals we returned the angle from the center of the image of both edges of field goal along with the slope of the top of the field goal. The slope was used in our docking code to determine which angle of approach we should take.

The biggest problem with the vision algorithm that we encountered was which thresholds to set for the blue strip, red balls, and yellow field goals. The lighting in the lab differed greatly from the lighting in room. Our vision worked extremely well in the lab during our tests. During the competition however, our thresholds were not calibrated correctly, leaving our robot to wander blind, either not being able to see objects of interest or seeing objects that weren&#8217;t actually present.

MECHANICAL STRATEGY

Our main mechanical strategy was ultimately to scoop up balls and store them somewhere until the robot reached a goal. Our method for collecting balls was to use a rotating combine which would come down from behind the ball and push it up and over a short ramp onto a sloped section where it would sit with several other balls. However, our final design was not what we had originally planned on fabricating.

Combine

The greatest time sink in our fabrication was undoubtedly the gear box we made. The main motivation behind the gearbox was that we needed a way to needed a way to rotate the combine by a motor which was 3 inches off axis. Since the two ball transfers were in line with the combine we realized we could not place a motor directly next to the combine. What we failed to realize was that a servo could be used in place of a motor and that it would also fit due to its smaller size. Consequently, we wasted four days making a fancy two-stage reduction gearbox. Upon completion of the gearbox we soon realized that it did not work because one of the cantilevered shafts deflected too much when it was loaded and would not rotate past a certain angle.

The original combine design we had also did not fully work. We started with a four-finned design with metal continuous from the center to the outer radius. This design had a tendency to trap the ball from above, which was unacceptable. We modified the combine by making a two-finned design where there was metal only at the outer radius and no central shaft. We also switched from aluminum fins to steel fins since the new thinner design required a greater material stiffness.

Storage and Release

Our storage and release mechanisms for the balls did not have any problems with them. The storage mechanism was essentially a bent sheet of aluminum, suspended on an angle from the main platform. The release mechanism was simply a small piece of aluminum attached to a servo, mounted to the main platform, which when rotated allowed the stored balls to roll out from the side of the robot.

Before settling on this idea, we had wanted to score five-pointers. To do this, we had come up with the idea of an "Archimedes Screw" to lift the balls from the carrying trough up to a ball hopper about 13 inches off of the floor. That way, we would be able to swing a gate out of the way using a servo and drop all of the balls into the bin.

The device was a tube with a flat bottom, a screw that rotated inside of it, and a motor to turn the screw. There were holes in the top and bottom so that that balls could fall into and out of the device.

The tube was made of plastic: white 3.5" PVC with lexan for the bottom and ABS for the motor mount. The screw was made of .2" solid copper grounding wire, which we found as scrap in the lab.

We bent the copper into the helical shape using a lathe and a 1.25" round aluminum bar. After drilling a hole in the bar to fit the wire, we chucked the bar up in a lathe and turned the machine on to the slowest possible speed, about 30 RPM. Holding the wire firmly, we were able to get the proper shape, although the whole process ended up being repeated three times, first because it was not big enough, then because the helix was right-handed and needed to be left-handed. It was disappointing to put in all this work, but then not end up using the part; our time could have been better spent making the other things work better.