Team Six Paper

Introduction

The MASLAB competition this year was a stimulating challenge which our team decided to take on as soon as we heard we had been admitted into the class. As in previous years, the competition consisted of collecting red wooden balls and placing them in goals. Additionally, this year teams were also allowed to place balls over goals for a significant increase in points. Our team immediately set out to make a robot that not only efficiently captured red balls but also would be capable of raising them to sufficient heights to make field goals. We faced many challenges, including software issues, mechanical obstacles, and especially the minimal time frame in which we had to complete our robot. In the end, however, we were able to construct a sound robot which fulfilled all of our expectations and performed solidly in the final competition.

Mechanical Design

Design Considerations:

The task to pick up a round object is not as trivial as it might sound, and our design evolved over the duration of the competition to maximize efficiency. Not only did we build 3 prototypes, we also drew upon our past experience in the Lego Robotics class 6.270 to decide our design for picking up balls. We initially decided that there are 3 primary methods for collecting a ball with a robot, including using the robot itself to drive over and chorale balls, using a claw-like mechanism to pick up balls, and using rollers to feed balls into a holding mechanism.

We ruled out using the robot to chorale balls because we wished to lift the ball and score a field goal. In order to lift the ball, we would need to have the ball in the robots possession, not simply stuck underneath it. Although using a claw to pickup, and potentially lift the ball to field goal height is possible, we ruled it out because our experience in 6.270 proved that this design has major technical difficulties. This left only the roller mechanism for our design, which appeared to have several mechanical advantages over the other two methods, including simplicity, large tolerances for ball positioning relative to the robot, and the ability to funnel balls to a lifting mechanism. Implementation

Ball Retrieval

Once we had decided on implementing the roller design to pick up balls with our robot, we designed a simple lifting mechanism which coupled with our rollers and lifted balls to our goal height. We originally had implemented rollers side by side; however we decided

to change our design to a single horizontal roller due to the increased “sweet spot” for collecting balls and the simpler drive train to power the single roller. We also decided to place the roller lower and on hinges rather then fixing it at the ball height, allowing the roller to maintain contact with the ball for an increased period of time. By fixing the roller on hinges, it allowed the robot to collect balls without needing to drive as closely to them and also to shoot them more effectively into our lifting mechanism.

Ball Lifting

We decided to use a motor to mechanically lift the ball up 13 inches which is high enough to roll balls over the goals for field goal points. The roller forced the ball into a V shaped funnel which then placed the ball at the bottom of the lifting mechanism. The lift consisted of two gears, one at ground height and one 13 inches higher, with a bike chain connecting the two. A small high speed motor powered the chain, spinning it around within a custom built metal box. One of the pins holding the chain together was replaced with a 3 inch long pin which would grab the ball from underneath and drive the ball up through the metal box onto the top of the robot.

Ball Storage

When the ball had been lifted by our chain drive, we originally planned to have a servo dump the ball over the goal; however we ran out of sensor points and did not have enough time to make a mechanical gate. Because of our lack of time, we instead built a storage box on the side of the robot, which would hold up to 20 balls in an effort to maximize ball possession points. Balls lifted would run through the top tray if it was empty, and fall through a hole into the storage box, until eventually the box filled and the platform above would then also be filled.

Movement

All of the above mechanical design relates to the ball retrieval mechanism, but a brief mention of the drive train and structural frame upon which this was mounted is necessary. We drove the robot with two motors mounted in the center and allowed efficient turning about the center by pulsing motors forward or reverse. The motors were attached to the bottom of a three tier structure. The bottom tier stored the batter and motors, the second tier stored the computer, and the third tier was the ball storage level.

Software

Considerations:

In any autonomous system, stability and robustness must be paramount; if a situation

arises for which no software is written, the robot will likely fail. There are no chances to communicate with the system to allow it to deal with the new problem, so all such provisions must be considered ahead of time.

> Secondly, we wanted software that was conducive to being worked on by both team members at different times. Importable or obscure code would make it practically impossible to write as a team without undoing each other's work.
>
> Lastly, it was important that the software be versatile since the mechanical design of the robot, as well as its strategy, were undergoing constant revision. Code tied too closely to a particular physical piece or action would be rendered useless every time the robot was changed. Conversely, by creating enough low-level functions such that all higher-level behaviors could be built completely without references to robot-specifics, behaviors would be simple to create as well as to modify.

Implementation:

To create stable code, some care was taken to handle exceptions thrown by methods. Additionally, all variables were initialized to safe values. Thus, problems would not cascade through our software; if a method failed and could not return an expected variable, the code would continue to operate based on our initialized values. While this situation would still probably cause the robot to behave poorly for a time, the system as a whole would continue to operate. Additionally, all processes were broken into basic threads, such that if a single thread failed, the others would persist. Ideally, a thread could be dedicated to monitoring other threads for such crashes, and restarting them as necessary. In the interests of time, this functionality was never fully implemented, but it would have been relatively simple to do so. Threading did allow us, however, to forgo adding timeouts to behaviors to ensure robust software design. Because one thread's job was to monitor the robot for signs that it was stuck (an unchanging gyroscope, high current draw from motors, etc.), we were able to ensure that even if other systems failed and left the robot in an unexpected state where it was unable to move, our software would free see this and take appropriate action.

In the interest of portable code, we tried to take advantage of the object-oriented features of Java. Wherever possible, we made methods private and avoided global variables. Also, we commented our code prolifically. Additionally, we ensured standard formatting of our code by sticking to Eclipse's built-in function, which made our code more readable.

Object-oriented programming also served to make our code versatile. We created a separate class for the robot itself, which included everything necessary to determine its

state and access its functions. In this way, whenever the physical design was modified, fields could be added or removed from the robot's class and all methods acting on the robot were automatically given the right information. Moreover, changes to low-level methods would not adversely affect higher-level behaviors, so long as methods still did what they were originally designed to do.

Our image processing scheme is a good example of versatile software. Rather than assigning pixels individual colors representing the most likely choice, we represented pixels by "color distributions." The visible spectrum was divided into 12 colors and for every pixel, a finite probability was assigned to each color using the HSV space. Pixels were dealt with in this state for as long as possible—only when a final color determination had to be made was the candidate with the highest probability selected as the pixels "true color." This approach gave us many useful features. Methods could add or subtract from a pixel's color distribution to reflect any factors beyond of the captured image's color. For instance, if a group of pixels were part of a horizontal line, the constraints of the competition made this group of pixels more likely to be blue, regardless of their appearance to the webcam.

Analysis:

Overall, our software performed extremely well. Everything was fast and consistent enough to function without adjusting our strategy, and no systems failed permanently during the competition (the robot did get stuck for 10 or 15 seconds for unknown reasons, but it automatically freed itself). In particular, our image processing code behaved effectively. While other teams struggled with lighting changes and calibrations, we never had to adjust any code to achieve flawless ball-recognition.

Although at first we adhered strongly to the considerations above, a lack of time forced us to make some concessions as the competition approached. For instance, our final code contains an entire class of publicly-visible variables that act as globals throughout the code.

Conclusion

Although we began Maslab worried about our two person team, with neither a computer scientist nor a mechanical engineer, we finished very satisfied with our robot and its software. Because we recognized that fabrication and code would not be our strengths, we worked hard to find an inherently robust design; unlike a claw, our roller could consistently grab any balls along the entire width of the robot, even if they were not seen. Had we had time to include a way to dump balls at a goal, we believe we could have been a serious threat to the winners. Of course, if my grandmother had wheels, she would be a trolley-car.

John Aquadro and Elvio Sadun

Team Bob Hagopian