Team Fourteen Paper

Overview

The experience in Maslab of my team was excellent, occupying and stimulating; the project in general greatly exceeded all our collective expectations, both in terms of commitment and reward. Inevitably, we made many mistakes and concurrently acquired enormous amounts of information on everything from Java to managing team dynamics and from mechanical structure to computational objectives and implementation. We ended up spending approximately 40 – 50 hours per week per person on our task, and spending additional time would have been beneficial. Our team had minimal experience both in the mechanical implementation of the rigor of a robotics course and with computational requirements of robotics in particular. This course is certainly feasible for groups and individuals of no experience, but in such instances, we strongly recommend a certain ingenuity and willingness to explore alternative possibilities and paradigm shifts.

Overall Strategy

We reasoned that the winning strategy would implement field goals, since their score yield (5 per ball) was greater than traditional goals (3), and we hoped to win rather than merely excel. To that end, we fashioned our robot around the requirements of detecting, capturing, lifting, and scoring balls, within the constraints of fitting our robot into the storage box and within the various time restrictions and material limitations.

Mechanical Design

Structurally, our bot was divided into three horizontal tiers; their precise configuration emerged as a consequence of our design needs. In general, our chassis began with a rectangular platform twenty centimeters wide and thirty centimeters long with two motors mounted on the back corners and two free wheels mounted on the front corners.

We divided the mechanical problem into three parts: collecting the balls, lifting them, and depositing them in the goal. To collect the balls, we found inspiration from a Wisconsin girl in our dorm and mounted a ‘bailer’ on the front of our robot akin to the mechanism used to collect wheat and form it into bales; the mechanism consisted of a servo rotating a horizontal bar about twenty centimeters wide approximately five centimeters from the axis. We installed a ramp slightly behind the bailer, which lifted the balls up approximately four centimeters onto the bottom level of the bot. We had to spend considerable time calibrating the spacing and inclination of the ramp with respect to the

bailer so as to achieve a path that a ball would always follow regardless of approach and initial velocity, but we succeeded; surprisingly, the angle of the ramp was very steep, but in general, any ball in the path of the bailer would be collected. The bottom tier was inclined slightly toward the front, so after the balls were collected by the bailer and ramp, they would roll to the back of the bot. The second mechanism had to lift the balls from the bottom tier to the top tier, which was at its peak approximately thirty-five centimeters above the ground; we resolved to use an arm powered by a high-torque motor mounted directly on the robot superstructure. Positioning the arm was difficult; it had to be sufficiently compact to fit in the box, long enough to raise the ball to the appropriate height and positioned correctly to avoid crashing into any other parts of the robot – clearance was a factor. The 'ball end' of the arm had a short metal cylinder we referred to as the 'cup;' the ball would firmly stay in place due to minor shaping adjustments we implemented. The arm only rotated through 180 degrees; it would hit 'bumpers' positioned at the desired limits of motion. The motor was slightly off-center with respect to the back of the bot along the horizontal axis. The third mechanism required the fewest moving parts; the balls were at this point thirty-five centimeters off the ground and had to pass over a thirty-centimeter tall wall; we had a gentle, constant incline that took the shape of a wedged siphon at the back of the bot which gave way to a slender, single pathway comprised of a number of dowels. A servo powered barricade restrained the balls in the siphon until such a point as we desired them to move. The final ramp extended over the bailer at the front of the robot, so the robot did not have to drive all the way up to the wall for the balls to have clearance. Given the three mechanisms, then, the chassis and remaining hardware structure of the bot became little more than a mounting platform for the ball transporting devices, the computational hardware and the sensors. We constructed it using three vertical columns between the bottom tier and the top tier; a middle level was installed to mount the Eden, battery and Orcpad, leaving the bottom level completely free for the balls to roll to the back of the bot, where they would be lifted by the arm. Because of small but annoying misalignments in the construction of the bot, different parts of the bot were slightly crooked, but in general the construction was sound. Wood was our primary material, with metal sufficing for parts which were required to be especially sturdy, especially precise, or especially light.

Sensors

Our primary navigational sensors were a pair of short-range infrared sensors mounted on two structural columns near the front of the bot. The distance along the length of the robot from the front to the infrared sensors was approximately ten centimeters, which we believed was sufficient to avoid the non-linear behavior of the sensors as the range approaches zero. The sensors were mounted high on the columns – sufficiently high to avoid being inadvertently triggered by either our spinning bailer or by gaps at the goal. The camera was mounted between the IR sensors with slight declination, and with an elevation of about twelve centimeters with respect to the ground. This caused some

problems; as the robot closed on a ball, the ball would drop out of visual range. This made modifications in our algorithms necessary. Also critical for navigation in the original configuration were two encoders mounted on our drive wheels and the gyroscope supplied. Additional motor output came from the servo on the bailer, the servo governing the barricade on the release tier, and the high-torque motor on the back of the bot.

Software Design

Because we had two primary programmers, we relegated the division of code into modules based on either sensory input, processing and interpretation or motor and hardware response. We considered threading, but decided that the vast bulk of the processing would be in a single thread, with minor secondary threads to monitor such small tasks as the clock, rotating the back arm, and Our primary sensor input besides the infrared sensor navigation was the camera; processing was executed in HSV format; we implemented an HSV function which was more efficient than the provided procedure. Rather than completely converting every pixel in the image, it sampled regions in a search for red pixels (or other colors, depending upon our needs); we incorporated blue-line filtering to negate the wall. While the input code was initially more complicated to digest and accurately plot the positions of objects on the field, once odometry was abandoned, our image needs vastly simplified. Our initial intent was to construct a powerful map based on the gyroscope, odometry, and landmark recognition, loosely founded on Kallman odometry. We devised a method to drive straight by providing feedback to the motors' power settings based on their odometry rates; turning hinged upon our gyroscope, which was fantastically temperamental. Retrospectively, calibrating the gyroscope with a regression function would have been fantastic, but we did not believe the massive gyroscope drift was entirely deterministic. To proceed once our bot had constructed the detailed map during the exploration round, we devised a series of algorithms to plot waypoints for the robot to follow in an effort to maximize our search efficiency and to optimize our red ball and goal finding processes. A more sophisticated wander algorithm was implemented to avoid the shortcomings of wall-following and of random wandering; the algorithm divided regions into cells based on the positioning of the walls, and then thoroughly evaluated each cell to determine whether a ball was present by turning slightly more than the angle required to dodge walls and turning 360 degrees at calculated intervals. Unfortunately, during the testing of this powerful series of algorithms, seemingly overwhelming problems emerged; the behavior was nothing like designed. After nearly a day of debugging and little progress, on the Wednesday before competition, the algorithms were abandoned in favor of simpler structures which we believed were more reliable, largely due to their simplicity. Unfortunately, because of time constraints and hardware malfunctions with the encoders, the odometry was completely abandoned. Our basic structure was a looped finite state machine; a wander algorithm would run, taking pictures intermittently until a red ball was located. At that point, a getRedBall function would retrieve the ball, returning to the wander algorithm. After two minutes had

elapsed, red ball operations would terminate and the bot would seek the goal again using the wander algorithm. This was a vastly simplified structure to our original, more ambitious configuration. The new wander algorithm was vastly simpler, merely encountering and turning away from walls, with a periodic 360 degree turn; it lacked the higher order functions that were possible with accurate odometry and was effectively a random algorithm based on the positioning of the walls it encountered. Also, we discovered shortly before the competition that one of our two infrared sensors was critically miscalibrated, such that its behavior no longer linearly varied with distance. It is likely this error contributed to the failure of our original algorithms; there were no available replacement sensors and time quickly ran out before we were able to counter this problem. Unfortunately, we still had not determined a viable solution to the gyroscope problem (a problem we had hoped odometry would have been able to assist), so our navigation had critical errors going into the final competition and turning was erratic.

Overall Performance

While our theoretical design was well-thought and extremely robust, our execution encountered often insurmountable problems. Our hardware design malfunctioned because our the gears of our high-torque motor stripped away, making any hopes of a complete 180 degree rotation impossible. Because we failed to solve that problem in the last few days, as well as the gyroscope and infrared sensor bugs, our robot had very little chance of actually scoring during the final round. Moreover, in an effort to get the simple algorithms up and running for the competition, the time stream was neglected while attention was diverted to the wander algorithms, so our robot would not universally stop after three minutes. As it turns out, the only way our robot would have come to a stop would have been had it successfully located a goal in its field of view between t = 170s and 180s. That did not happen during the competition, and we were penalized. The pragmatic success of our robot was severely lacking, but we generated many theoretical and interesting approaches to the obstacles which faced us. Obviously the general failure of our bot to complete its objectives was extremely disheartening, but the practical experience which we obtained in addition to mentally attacking the problem was stimulating and highly rewarding.

Suggestions for future teams:

- When you design an algorithm or mechanical solution, it doesn’t have to be the optimal or perfect solution, it merely has to work. Once you have a working implementation, you can improve it enormously.
- Make sure you have something working very early on; if you attempt the impossible and find yourself incapable of completing your task before the deadline, your substitute will lack many improvements and refinements you solved very early on.
- Test ridiculously frequently and refine everything as many times as you can reasonably afford.

- Delusions of grandeur and computing excellence have no place in the pragmatic world; a solution to the problem must be implemented before idealism can hope to survive.
- Organize early on and outline clear objectives for everyone on the team. Address conceptual and design problems together, then implement them solitarily.
- Try to work on designs long before you need them; start building the chassis as early as you have clear objectives and designs, e.g.
- Invest as much time as you can humanly afford from the beginning.
- Always have a backup plan; Murphy was right.
- When in doubt, use a regression equation.
