Team Three Paper

Team Three: Order of Random Cauchy

- **Daniel Kane** (class of 2007)
- **Anders Kaseorg** (class of 2008)
- **Andrew Spann** (class of 2007)
- **Yoyo Zhou** (class of 2007)

We’re pretty sure we violated every single principle of traditional good project management in existence. We formed a team of all math majors/double majors, we all lived in the same floor of the same dorm, we were all either freshmen or sophomores, we took longer than the recommended time to complete our robot’s mechanical construction, we scored 0 points on the Mock Contest held three days before the real one, and we made many major revisions to our software in the last 48 hours before the robot impounding deadline as opposed to debugging existing code. Yet when the contest day came, our robot, Archimedes Pi, somehow scored the most points.

Contents

    - IR sensors

- Image Processing
    - Color voting
    - Positions in 3-space

- More Math (You Want to Come Here)
    - Driving in arcs
    - Docking with field goals
    - Kalman odometry
    - Mapping (or lack thereof)
    - Force fields

- Nifty Telemetry

- Robot Strengths
    - Steady-state lifting device
    - Fast, effective image processing algorithm
    - Good docking procedure
    - Traveled in arcs

- Robot Weaknesses and Suggested Solutions
    - Heavy robot = slow
    - Known modes of failure
    - We never even saw half of the field
    - Didn’t use exploration round
    - Takes balls slightly over 20 seconds to climb screws
    - Definitely not made by mechanical engineers
    - There’s so much that could have gone wrong

- Results

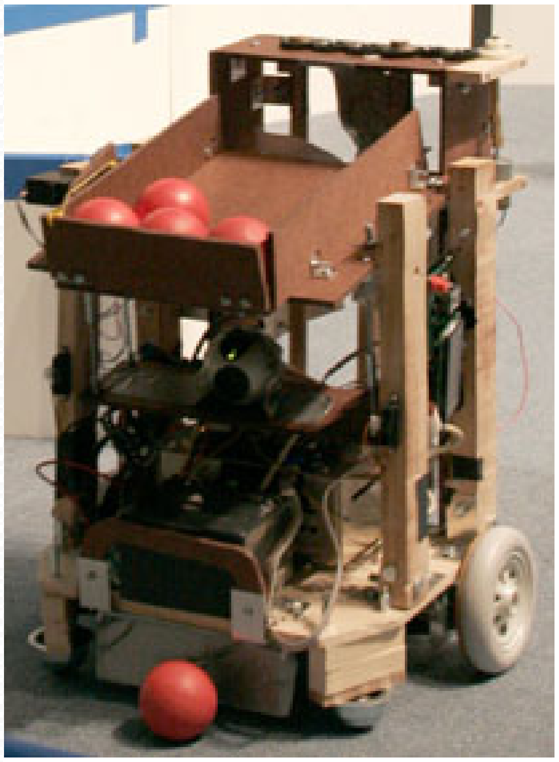
- Lessons Learned

- For future MASLab participants wanting to learn more

Planning and Strategery

Early on, we decided we wanted to score field goals—a reasonable decision, as it would yield the most points and, with a lifting device, ought to be just as easy as going through the mousehole, but the decision that would alter our destiny the most drastically. We were also influenced by our lack of mechanical experience to try to design something that would not be too mechanically complicated; gripping arms and the like were right out. We were also ambitious enough to want to map the playing field in the exploration round, though we ran out of time to implement it.

Mechanics and Sensors

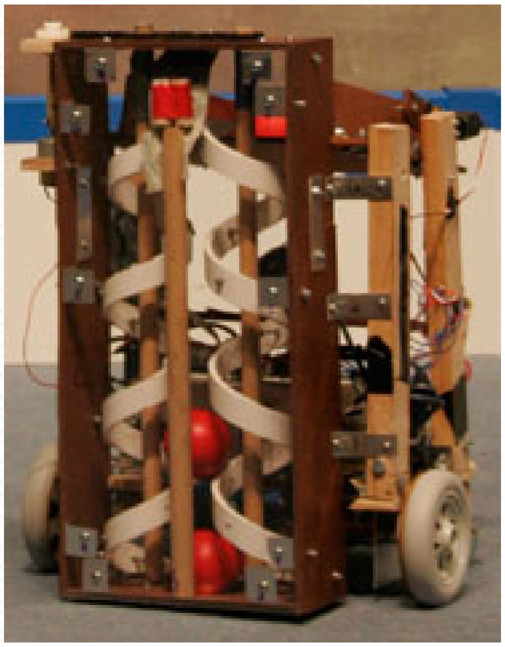


Ball storage

A rectangular platform, supported by four pillars at its corners, somewhat more than 12 inches above the ground, is tilted so that balls roll toward its front. It has a capacity of about 10 to 12 balls, assuming random packing, and does not funnel down at its opening because we feared that balls might become jammed against each other.

Archimedes would be proud

A lot of teams independently thought of the Archimedes' screw to lift balls since it's a compact mechanism and it can be run continuously without giving attention to individual balls. Only one other team besides us actually implemented an Archimedes' screw since it is hard to make the actual screw and there are difficulties loading and unloading balls. We made our loading mechanism robust by building two screws, which are mounted at the rear of the robot. We constructed the screws using the amazing power of low-tech tools: since helices are geodesics on a cylinder, we used a string to measure around a PVC pipe and traced the pipe so we could cut it with a simple hacksaw. We succeeded in cutting two screws of opposite chirality, each about 15 inches long.



Screw casing and gears

The Archimedes Screws are connected to two 18" shafts by a series of bolts. The bolts actually wound up being troublesome to us because if they extend too far through the shaft they can lead to jamming; altogether, too much time was spent on readjusting these bolts. The bottom of each shaft has a small hole cut into it and a screw sticks up from the bottom of the casing, sliding into this hole. The bottom of the casing rests 1/4"-3/8" above the floor. The center of the bottom casing is hollowed out and a metal strip lies there which gives the ball its initial lift. A pole is attached to the back center of the casing, and as the ball travels up it is always adjacent to the pole. Gears are on the top of the casing and a gear train leads to the side where a high-torque motor is attached with an inverted universal joint. The casing is mounted on the back of the wooden base, behind the wheels. It would have looked cooler in all of the pictures if we had made our ball storage area and all sides of the screw casing out of polycarbonate, but everything was already cut well and working fine and time was an issue so we only made the casing back see-through and stuck with our wooden prototypes for the rest. Special thanks to Ross Hatton for giving us good mechanical engineering input on how to make the gears. We were a bunch of mathematicians who hadn’t realised that gears involved attention when we made our design.

One-way bottom gate

The bottom “gate” consists of duct tape covering PVC foam held up by reinforced wire attached to joints that pivot one way only. The underside of our robot guides balls to the rear screw assembly. The gate rests between two wheels that are giant ball bearings (the robot has rear-wheel drive).

Servo-operated top gate

The top gate is hinged at the bottom and has a servo mounted to a piece of wood that rotates pulling a piece of wire reinforced with hollow copper cylinder to open and close the gate. The hinged front wall swings open. We actually stop just in front of the field goal and swing the hinge open rather than running all the way up against the mouse hole wall so that we have a larger margin of error allowed for our goal alignment (even though our field goal aligning algorithm gave us remarkably good results as long as the field goal was visible).

IR sensors

We have an IR sensor mounted 9 inches above the ground on a front pillar on each side of the bot, facing forward, to provide information about the nearness of walls; it is at that height to avoid being confused by the mouseholes.

Image processing

Our camera was intentionally placed at the height of blue tick marks, about 9 inches above the ground, so that we could ignore anything above the horizon. The disadvantages were having to tilt the camera down to see anything useful, which we compensated for (see **More Math**), and even then we were unable to see anything between the front of the bot and about 1 foot futher away.

Color voting

We took pictures with our camera in the appropriate playing fields (using the mock contests to get pictures in the lighting of room) of various features and separated the colors in each picture into one of 7 designations: white (walls), blue (wall tops and tick marks), yellow (goals), red (balls), black (barcodes), green (barcodes), and floor-grey. Using random distortions to run this through a “voting” program yielded a 16×16×16 lookup table, mapping the most significant bits of any color to one of these 7 colors. While this training method was somewhat time-consuming, it yielded fairly good and very fast results as long as the lighting conditions weren’t too different from that of the sampled pictures; the lab and room had similar enough lighting for our purposes.

Positions in 3-space

We also used some trigonometry to map pixel coordinates in camera images to coordinates in the reference frame of the bot, with x axis horizontal, y axis depth, and z axis vertical, with the bot at the origin facing the positive y axis. Eventually, using odometry and the knowledge of our current position and orientation contained therein, we turned these into absolute positions, with the origin at the start position.

More Math (You Want to Come Here)

Driving in arcs

Early on, we decided that if at all possible, we should be efficient in driving the bot. Better than attempting to turn precisely and drive exactly straight was driving in a continuous arc, using feedback from the camera to improve our course as we approached. The arc driving algorithm is used to both chomp balls and score goals.

Docking with field goals

When we see a goal we go for it if (0) we think we have at least 6 balls, (1) we think we have at least 4 balls and there are 105 or fewer seconds remaining, (2) we think we have at least 1 ball and there are 75 or fewer seconds remaining, or (3) there are 30 or fewer seconds remaining. To dock we find a point 27 inches from the front of the goal, drive in an arc to it, turn toward the goal, and then drive in an arc toward the goal, using feedback from the IR sensors when it gets close enough. Our front gate is 7 inches wide, and the goal posts are 10 inches wide. We've programmed the robot so that it stops with enough clearance that the front of the front gate stops just before the goal posts rather than sticking out between them (see picture in Results section). This way we allow for error in the docking procedure in position and angle. The hinges, as an unexpected benefit, also wind up funneling balls to the center.

We didn't finish docking until Wednesday at 9pm of the last week (robots were called for the impounding deadline started Thursday at 5, which was 6:30 when they came to us) and during the day Wednesday we were a bit uneasy that we wouldn't have time to debug all of our existing code because we were writing new procedures, but getting good docking was absolutely essential in the contest. You can't win with just possession points, so we figured that even if we had a less than 50% chance of it working that it was the right thing to go for. It turned out that our docking code worked beautifully so we were happy.

We set our time cutoffs somewhat arbitrarily, but we're glad that we did it this way. A couple other teams missed their chance to dock early because they wanted to wait until closer to the end and dock once (which theoretically *is* better) and got single digit scores when they really had nice robots. Given the facts that (1) the need for the mechanical construction of complex lifting devices this year slowed all the teams down considerably and (2) there was a three minute time limit this year, not a four minute limit, docking when you had just a few balls and sealing your position instead of being greedy actually made a lot of sense this year even though I would have been reluctant to do so at first.

Kalman odometry

We kept track of our bot's location using an Extended Kalman Filter on out data from the wheel encoders and the gyro. We assumed that all of the error from the encoders came from the encoders being at some fractional number of ticks of which we know only the integer part (though we later increased this assumed error after discovering that our encoders were highly biased towards even values). We kept track of the variables `x` and `y` for position, `th` for orientation, two variables `el` and `er` for the fractional parts on our left and right encoders, and two variables `go` and `gr` which tracked the calibration error of the gyro.

Mapping (or lack thereof)

We realized during the mock contests that the shadows cast by the lights on the playing field make it impossible for our camera to tell the walls apart from the floor, significantly dampening our interest in trying to map the playing field. Given that we nearly ran out of time as it is, maybe that's a good thing. But hopefully, next year's floor will have a more distinguishable color.

We make no effort to use the mapping round or to record the long-term positions of objects. We did use our odometry and knowledge of positions in 3-space to keep track of the absolute position of our bot, as well as the ball or goal we're tracking.

Force fields

We used our odometry to come up with a wandering program that attempts to explore new areas without using direct wall following. We imagined our robot dropping invisible "force fields" at locations where it has been and at locations where it detected walls. We programmed the robot to travel in the arc that causes the direction that the front of the robot travels in to be in the same direction as the gradient of the total force field at that point, albeit giving precedence to avoiding walls. We didn't have enough time to fully understand the ideas needed to make this method work optimally, but noticed a few things. For example, we needed to implement a forward bias to prevent a problem that caused our robot to get stuck in a loop of travelling back and forth. We also needed to make the strength of the force fields decay exponentially with time to prevent excessive buildup and to compensate for inaccurate odometry. We programmed all of the force field stuff in the last 24 hours before the Thursday night impounding deadline. We wish we could have tweaked the parameters a bit more since there are situations where we know that the robot doesn't do as well, but the robot did its job on contest day.

Nifty Telemetry

Our robot spits out text messages and channel names that are all titles of puzzles, taken from the 2005 MIT Mystery Hunt. We have a vector graphics rendering of our local map which shows the robot's orientation, field of view, IR sensor ranges, spotted goals or balls, and force fields. We also mark many of these items on our video channel when they're in view.

Robot Strengths

Steady-state lifting device

- Don’t need to give individual balls attention
- Just run over balls and forget about them
- Dual Archimedes’ screws mean easy loading and no jams (but took effort to make it this clean)

Fast, effective image processing algorithm

- No need for RGB to HSV conversion, so fast
- Good use of multiple threads
- Can detect positions in 3-space for mapping purposes

Good docking procedure

- Early dock (had a little over 60 seconds left when we started it) wound up helping us a lot, but might not be a good idea under all possible future years’ circumstances
- Accurate docking that wouldn’t miss the bin or get caught on the goal posts
- Waiting just the right time for the most recently picked up ball to climb our screws helped, although if we thought we’d have enough time for 2 docks we might not have done this.
- Local mapping with gyro and wheel encoder Kalman-filtered odometry did wonders for accuracy

Traveled in arcs

- Let us grab balls accurately
- Let us grab balls quickly
- Force field wandering performed better than random “walk until you see a wall then turn” in our tests

---

Robot Weaknesses and Suggested Solutions

Heavy robot = slow

- Could have used fewer bolts/smaller bolts, cut down on metal L-brackets, and used lighter materials (cardboard?) for nonstructural components
- Keeping the battery charged so motors have full current partially counteracts this
- Carrying this thing to lab everyday was not fun

Known modes of failure

- Wheel can get caught on cylindrical wall extensions
- Tight corners have mixed results (could be fixed by tweaking force field parameters more)
- No bump sensors (they're Nintendo Power and Reset buttons, they're so cool!), but our dual IRs were working fine

We never even saw half of the field

- Although we did we got all the balls on the half that we did see. This is also partially due to the fact that docking came when it did
- But our force field algorithm could have definitely been improved

Didn't use exploration round

- The rest of the system was time-consuming to construct
- IAP was too short, and our efforts not solely focused on MASLab

Takes balls slightly over 20 seconds to climb screws

- Can probably adjust gear ratio to get faster rotation
- Maybe can get away with using normal motor (higher RPM) than high-torque one

Definitely not made by mechanical engineers

- Not designed to be industrializable/scalable to larger sizes
- Wouldn't count on this thing being able to run for hundreds of hours before breaking down

- Some parts of construction required physical labor (i.e. sawing away at PVC pipe by hand for a few hours)
- But our mathematician's approach did give us ideas for the design of the screws and its bearings that were good ideas

There's so much that could have gone wrong

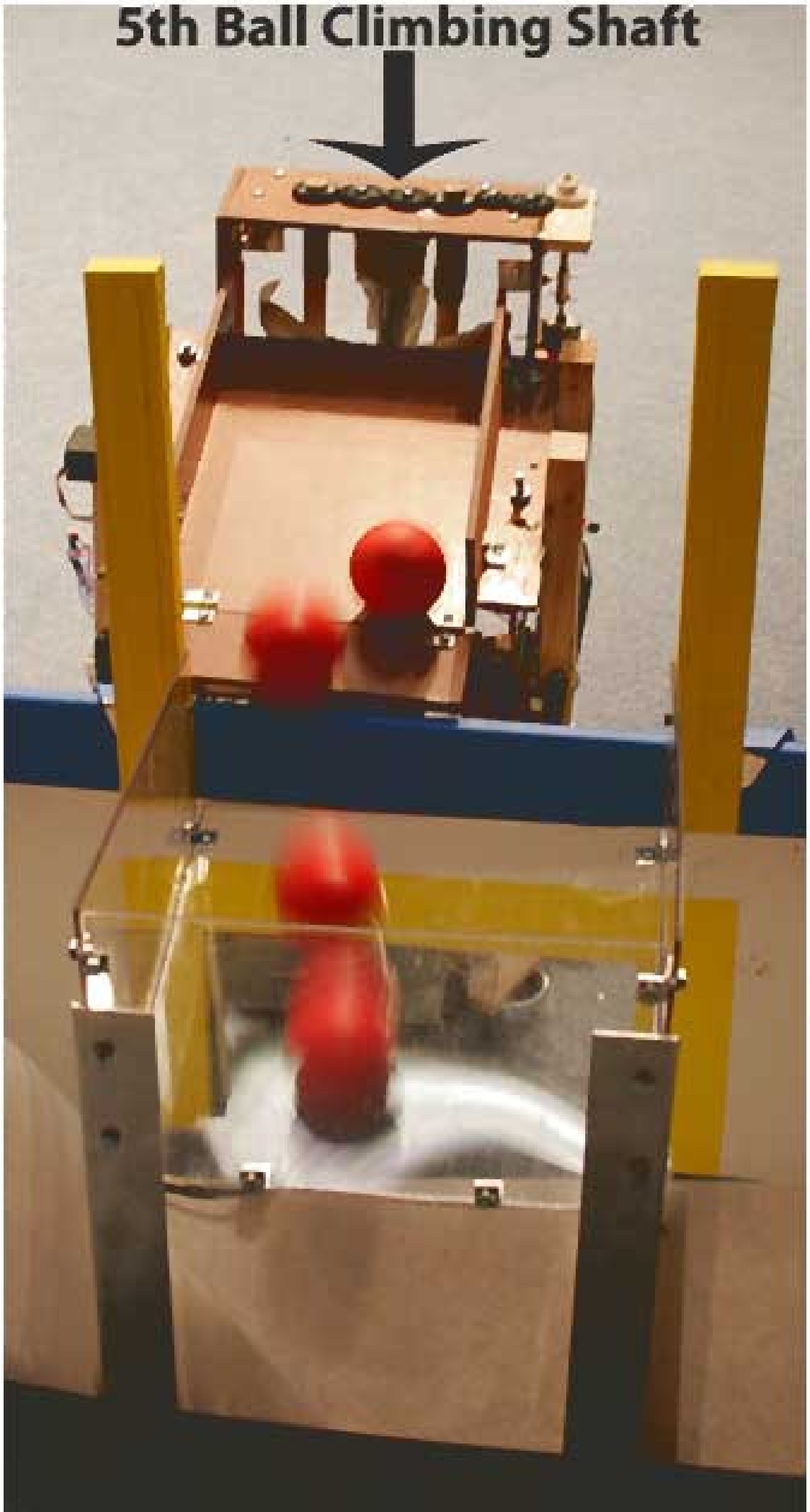
- Drastic changes in code so close to the impounding deadline are dangerous, although we'd calculated the risk and it seemed viable
- Massive Orc Pad failure just an hour before the competition (turned out to be USBMOD's fault, see below)
- Not working on the weekends is not recommended, but Mystery Hunt is too fun

Shhhh! This paper you're reading is twice the maximum word length that it's supposed to be. Don't tell anyone.

Results

On contest day, our USBMOD had several pins break off after we tried to hotglue it into place. After some onerous minutes of last-minute debugging, the USBMOD was replaced and all was well.

Our robot did not use its exploration round. In the scoring round, it wandered through only about the half of the playing field in which it started, picking up balls as it went along. It was fortunate that as it picked up the 5th ball, a goal was directly in its view. The field goal docking mechanism worked successfully, as did the waiting until 25 seconds after the last ball was picked up; the last ball escaped the shaft and rolled into the goal just before the gate closed. Then, less than 30 seconds remained in the round; the bot had already picked up all the nearby balls and scored no more points. This total of 25 points wound up being the high score. MASLab was a lot of fun. We really didn't expect to win.


5th Ball Climbing Shaft

Lessons Learned

1. If it looks like you're not going to have enough time to implement your grandiose plans, you won't. Orc Pad Tetris (complete with music!) never saw the light of day. Oh, yeah, and there were more serious things were this lesson applied too.

2. Hofstadter's Law: It always takes longer than you think, even when you take into account Hofstadter's Law. Especially if you have 0 mechanical engineers on the team.

3. Test early, test often.

4. Balls are much better in the zone where they score 5 points than in possession.

5. Having 1.5 mechanical people and 2.5 software people worked for us. (We actually wound up having to funnel a bit more manpower than that into mechanical due to our lack of mechanical engineering knowledge.

6. By Murphy's Law, things will break at the last minute, so do as few things as possible at the last minute. But for the next to last minute, do a lot:
   - The mechanical construction wasn't truly complete until Monday of the last week (we were still recalibrating the screws down to the last day to get them working perfectly).
   - On Tuesday of the final week we scored 0 points at the mock competition, although this was partly due to a few really silly bugs.
   - Our docking code wasn't working until 9pm Wednesday when the robot had to be submitted Thursday night around 5–6, but all day Wednesday we kept trying to code it anyway since field goals were just worth too many points not to do.
   - Our force field wandering program was done entirely after our docking code and we were changing parameters down to the last few minutes (if we'd actually run out of time we knew we could go back to our more random-walk type primitive code).

For future MASLab participants wanting to learn more

.

- **TeamEleven** won the engineering award and had a robot that really had the potential to score big. They had a fast robot that could wander around the field and pick up balls with a steady-state conveyor belt method. Their programming was solid. They also gave their robot an amusing voice library and were fun to watch.

- **TeamThirteen** overall had a solid execution (the second highest score total) and had a robot that was really fun to watch. They get better odometry results by taking apart an optical mouse and using that as a super wheel encoder. During the contest, they had the most exciting ending to their 3 minute round because they managed to just barely make it to the goal and deposit their balls with only one or two seconds to spare.

- Many other teams had good ideas that are worth reading about (too many to mention here)

- Lastly, we're real people with real email addresses. You could email us all or obtain our individual email addresses by the usual MIT ways. Just don't expect a fast reply if you email us during Mystery Hunt…