The Cabrillo Robonauts

Cabrillo Robotics Swarmathon Tech Report

Cabrillo Community College

Santa Cruz California

April 2017

Reviewed By Michael Matera

Overall Strategy

Most of our team from the 2016 Swarmathon competition graduated from Cabrillo College or moved; of the 15 team members from spring 2016 only four of us were still enrolled at Cabrillo College during the 2016 fall semester.

To try to get a head start on the competition and the challenges ahead, we held weekly summer workshops aimed at building interest towards the Cabrillo Robotics Club which would help with recruitment and building up our team size. Officers were replaced and we held the overall strategy of dividing up the Swarmathon competition into several different small victories. We felt these would help make the competition seem less daunting, as well as boost moral as we went throughout the semester.

Victory 1) Application
Victory 2) Recruitment
Victory 3) Brainstorming / strategies / ideas to implement (times and days to work)
Victory 4) Swarmathon Workshops
Victory 5) Outreach
Victory 6) 5 page tech report (keep records as we go)
Victory 7) Finish Code
Victory 8) Finish any loose ends

Algorithms and methods

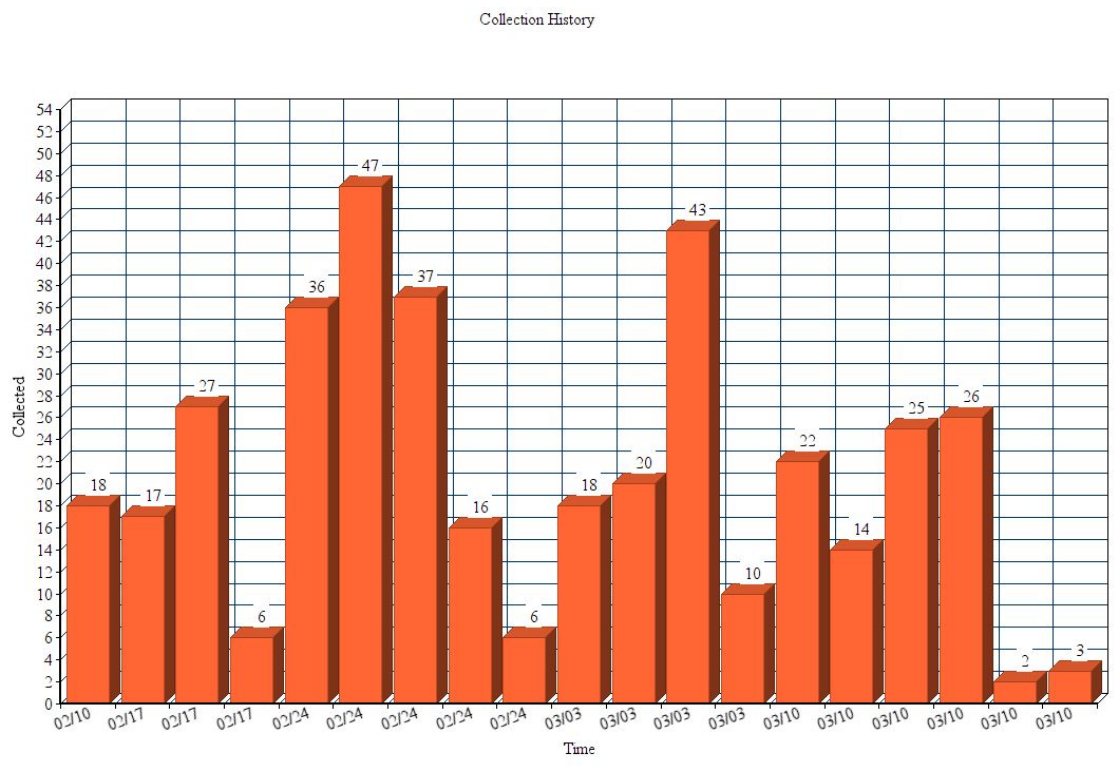
Avoiding Obstacles

The greatest change implemented was the rovers behavioral response to obstacles. Prior to the change, rovers would rotate 0.8 radians away from an obstacle detected through sonar, rotating just enough to avoid a collision. When the obstacle was a wall, however, this behavior caused the rovers to indefinitely follow the outer walls of the arena, a behavioral bug we have dubbed wall-following. To fix this bug, the rovers needed to be able to distinguish between obstacles that they should circumvent and those they should bounce away from.

Right now, the parameters of the competition give three legitimate possibilities for obstacles: walls, April tags, and other rovers. Now, if a rover has collected a target, then the rover will attempt to return to center. It will set absolute* (map reference frame) goals all of the time that it's not avoiding an obstacle. These absolute goals are supposed to land in the center of the arena. Therefore, any obstacle encountered along the way is either an April tag or a rover.Suppose, to the contrary, that the rover has not collected a target and it has detected an obstacle on the way to its current goal. The goal is not necessarily at the center because the rover is not trying to return a target to the center. The goal could be anywhere, and it will be a relative* goal. Therefore, the obstacle in this case could be an April tag, a rover, or a wall.

So, we have two cases for the rover to evaluate. First case, the obstacle is not a wall and the rover is attempting a return to center. Second case, the obstacle might be a wall and the rover's goal

could be anywhere. The second case is what generates the wall-following bug. It so happens that there's a constant left over from last year's code, called C_BOUNCE_CONST. Within obstacleHandler in mobility.cpp, the second case now uses C_BOUNCE_CONST to turn 1.8 radians (roughly 100 degrees), and in the first case (target collected and returning to center) the rover turns 0.6 radians.
As a result, the rovers will now "bounce" off of walls and go around the other objects, as they should. Along the way, we also experimented with different arrangements of the if-else tree that evaluates the obstacle handling.



Navigation:

As the rover's purpose is to collect tags and return them to a center home base, it was essential that the rovers would be able to properly navigate in order to return to the central drop point. This proved to be a challenge, as we discovered the limitations in the accuracy of GPS, as well as the drift in odometry caused by wheel slipping that made it so that we couldn't safely rely on odometry for navigation either. A large amount of our focus throughout the development of the rovers was working on an effective and reliable method of navigation.

The first thing that we noticed about navigation was that there were two EKFs, one that took odometry information, and one that received the output from the first EKF as well as satellite data as input. Because of the extreme drift with odometry, we decided it would be better to be computing information from one EKF at a time. Thus we implemented a Boolean flag, named useOdom, which was used to distinguish when odometry information was to be used in positioning, and when it was not. We then separated goal setting into two independent methods, robot relative goals, and world relative goals.

Our function to implement robot relative goal setting is called setRelativeGoal() and works with polar coordinates, taking a distance and an angle as arguments. We set useOdom to true, and rely solely on odometry readings for rover movement when setting a goal that was not a defined location in our map. Thus function computed the theta of the goal by adding the angle argument to the current angle of the rover, and computed the x and y coordinated of the goal using the polar distance and adding the computed x and y distances to the coordinates of the odometry based currentLocation. We then combed through the code, and everywhere a goalLocation was set, we determined if it was a relative goal or not, and if it was we called our new function and determined the polar angle and distance of the new goal.

Our second navigation function was used in world relative navigation, when we were setting an absolute and unchanging goal. The function setAbsoluteGoal() was defined, and in it useOdom was set to false so that we were relying solely on data from GPS. Despite the inaccuracy of GPS, it does not experience drift in the positioning as time goes on, so it was the most reliable for an absolute goal. The arguments for this function were the x and y coordinates of the desired goalLocation, and the function set the new goal to those coordinates and computed the angle to the goal. Everywhere a goal was set to the center in order to return a tag, we called the setAbsoluteGoal() function.

We experienced an increase in accuracy for center returning with these changes, particularly in the simulator, as the rovers retuned reliably right to the center location upon picking up a tag, but it was outside of the simulator that we were still unable to return directly to the nest. With testing we determined that GPS was only accurate to + or – 3 meters with the best of conditions. This area of uncertainty meant that our rovers were not able to consistently return to the center, but they were, however, able to come within close proximity to the nest. As we were unable to find a way of making GPS better, we instead implemented a search algorithm that the rover would follow when it had reached the location that it thought was the center, until it saw the tags that marked the border of the nest. The search algorithm would choose a randomly generated angle, would turn in that direction, and then would drive forward for a distance, before spinning to check its nearby surroundings, and then repeating the procedure. This created a spread out, spiral like shape, allowing it to search a large area for the center. If the rover was unable to find the center after one spiral, it would return to its computed GPS center, choose a new random angle, and search again.

The search algorithm is more time consuming then we would like, but it was the most efficient method for dealing with odometry readings we could not trust for absolute goal calculations and imprecise GPS.

Return To Food:

In order to increase the return of tags to the center, we implemented a return to food feature. Knowing that the tags can be stored in clusters, we realized that when a cluster is found, we could save time and increase tag collection by returning to this cluster rather than continuing the random search. When a block was found we stored the GPS location of the rovers current location into a variable named foodLocation. After dropping the block off in the center, an absolute goal was set to the recorded location of the previously found tag. Upon reaching this location the rover searches the area briefly for any nearby “food”, until it either finds some, or gives up and continues the random search. This significantly increased the number of tags returned, especially upon the frequent encounter with a cluster of blocks.

State Machine Addons:

We added a state called CIRCLE that made the rovers do circles every once in while during their random search wander. Since the rover’s camera screen is fairly tiny the circle state helped pick up blocks that the rover would normally miss on their way to their randomly picked goal.

Avoiding Blocks:

A big problem with the physical rovers is the fact that the blocks are a hazard to the rovers themselves. When we tested the physical rovers we noticed that the rovers would get stuck if it ran over blocks. We added a case to the Target Handler Function when the camera sees a tag. We checked to see if the rover was already holding a block, it sees a number of tags, it’s not already avoiding an obstacle, it’s not in the ROTATE state, and if the distance from the center is greater than or equal to one meter.

If all of those conditions are met then we check to see if any of tags the rovers see are the center home tags. If they aren’t a center home tag then we assume the tags are a block. Then we check to see if the tag the rover sees is not the tag it is already holding. If that is true then if the tag is on the right side of the rover it turns to the right and vice versa for the left side.

We know that this does not always fix the problem, if the rover does not see the block in it’s camera then it won’t be able to avoid the block.

Avoiding the Center:

We noticed that sometimes the rovers would run over the center on it's random searching. So to fix that we count how many center home tags are seen and if there is more on the left then it will turn almost 90 degrees to the right and vice versa if more center tags are on the right. This greatly improved the performance of the rover avoiding the center if it wasn't holding a block.

Coming home:

We noticed that when the rovers were coming home with a block that at certain angles the rover would get stuck and bulldozer over the center home. So along with the countRight and countLeft variables which we changed to ints, we added a variable sumCog. Which was the sum of all the distances relative to the rovers center of view. So we would know which blocks were closer to the rover. It helps if the rover enters the center close to parallel with the edges it should grab the edge and pull it towards the center. Although in the worst center enter the rover will still bulldozer the center edge.

Approaching goal:

To arrive at a goal, the rovers would set an initial angle to the goal and try to keep driving at that angle. If the angle was off from the original destination angle, it would stop and rotate before continuing its drive to the destination. The rover would, however, only realize they have reached the goal when the angle changed from zero to pi radians, causing it to overshoot the goal. To fix this behavior we implemented a change to tell the rover it has reached its goal when it was close enough (hence the distance to goal variable was small).

Physical Tests

The object of the Physical tests is to determine how well the current code works with the actual rovers.

Type of test done using the rovers:

- Measuring the distance before a rover will respond to a target. This was done by measuring the distance to the rovers to the food source. We recorded the distance from the rover to when it will react to cube. Initially, the data was inaccurate because of failure to completely reset the rover before attempting another test.
- Examine the behavior as it picks up a cube and attempts to return to home base.
- Examine the GPS at different locations to ascertain the reliability of using the GPS as a means of returning home.
- Do mock automate modes to see how many cubes were placed in the target home.

Type of problems found with physical simulations:

- Not being able to locate cubes or home base because rovers were moving too fast
- Dropping cubes outside home location
- Using odometer to find location has inherited problems.
- Using the distance based upon odometer is fairly accurate until a cube gets stuck under one of the wheels of the rover. After this event, the rover was not able to find the home base.
- The rover with the most cubes in the home base was the rover that avoided running over cubes.
- One rover who ran over a cube could not find home.
- With GPS the accuracy varied from couple of meters to several meters.
- Accuracy depended on if it is indoors or outdoors and if on the outside the maximum number of satellites was 8. Because GPS usually gets you within plus or minus several meters, it cannot be solely used to be able to return to your home base. Because GPS values hopped around it is not very reliable for finding home by itself.

Video Links

https://www.youtube.com/watch?v=EYQWp9rgb_c Cabrillo Robotics News Story (KSBW 8)
https://youtu.be/PyZ9UNyv7_A Field Test
https://youtu.be/YkCbELQ_000 Field Test

The Cabrillo Robonauts

Cabrillo Faculty

Michael Matera	Academic Advisor
Kelly Horner	Assistant Coordinator

Student Team Members

Christopher (Fritz) Billingsley	President / Project Manager
Allee Smallwood	Vice President / Programmer
Kiley Roberson	Technical Lead / Programmer
Benjamin Hung	Programming / Outreach
Andrew Thach	Programming / Testing
Jake Angobaldo	Programming / Virtual Testing
Kage Conner	Programming
Andrew Boyd	Programming
Takashi Tamasu	Field Testing Lead
Kenny Murray	Field Testing
Daniel DeLeon	Field Testing
Patrick Mojica	Field Testing
Morgan Knapp	Field Testing
Phoebe Zajac	Field Testing
Clayton Gjerstad	Field Testing
Chris Sweeney	Field Testing
Patrick Lewis	Field Testing
Devon Furtado	Field Testing
Gisella Pezzini	Field Testing
Garrett Mason	Outreach Lead / Programming
Zach Goulden	Outreach Cameraman
Zoë-Marlene Frei	Fundraising Lead / Programming
Gadi Rosen	Fundraising / Field Testing
Joseph Nguyen	Fundraising / Programming
Dan Rosen	Fundraising
Shawhan (Red) Shams	Fundraising
Arturo Cejudo	Virtual Testing Lead
Satya Zomer	Virtual Testing
Jaime Garcia	Virtual Testing
Tess Collinge	Virtual Testing

Kurt Degregorio	Rover Assembly Help
Rachel White	Rover Assembly Help
Xitlali Galmez	Rover Assembly Help
Nathan Wahler	Rover Assembly Help
Ewelina Biranowska	Rover Assembly Help
Nathan Meier	Rover Assembly Help
Jari Baker-Wills	Rover Assembly Help
Adelicia Johnson	Rover Assembly Help
Ariana Carruthers	Rover Assembly Help
Haley Powell	Rover Assembly Help
Peter Replogle	Rover Assembly Help
Alex Saldana	Rover Assembly Help
Colin Taylor	Rover Assembly Help
Anthony Mininni	Rover Assembly Help
Lucinda Belle Billingsley	Emotional Support Animal