Monte Carlo Localization

Andrew Larger
Zihao Lu
Trevor Plassman
Matt Schmucki
Alicia Zwiebel

Agenda

- Problem Statement
- Approach
- Course
- A* Navigation/MCL
- Issues/Future Work
- Demonstration/Questions

Problem Statement

- Most robots rely on GPS
- GPS is only so accurate
- Have robots localize themselves
- Navigate based on self localization

Approach

- Use a Lego Mindstorm
- Use A* Navigation
- Use MCL
- Combine all three together

Course

- 140 cm x 63 cm
- Movable walls

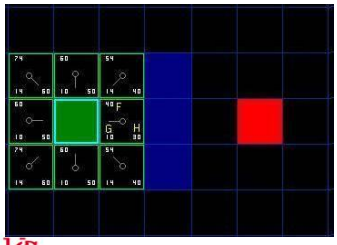
A* Navigation

- First documented in 1968
- Expanded on a 1959 formula by introducing heuristics

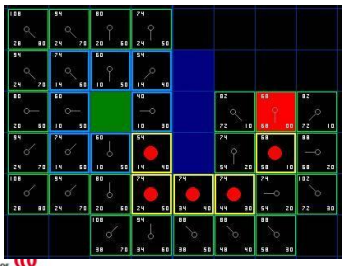
‘Cost’

- Each node has a ‘cost’
- G – Exact Cost
  - Exact distance from the starting point
- H – Heuristic Cost
  - Estimated distance from end point
- F – Total Cost
  - F=G+H

‘Cost’ Example Cont.



‘Cost’ Example Cont.



Implementing

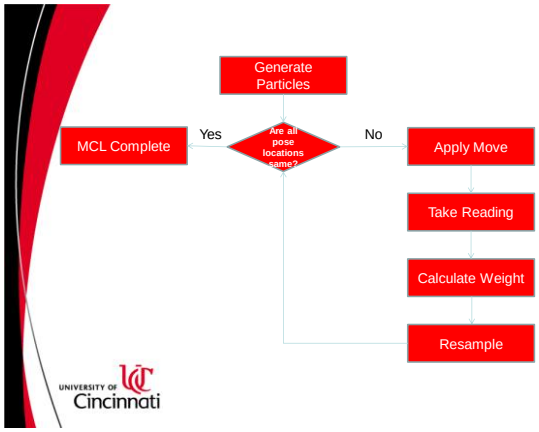
- Add the starting node to open list
- Find lowest F in open list (called current)
  - Move it to the closed list
- Investigate current’s 8 neighbors
  - If a wall or on closed list; ignore
  - If not on open list, add to open; calculate F
  - If on open list, compare G; if lower, change parent

Implementing Cont.

- Repeat
- When current node is the same as end node; stop
- Trace path of parents
- A* Demonstration

MCL

- Particle filter applied to robotic localization
- Random initial guesses are populated
- Guesses selected based on probability of being correct location



Generate Particles

- Create a matrix of particles
- Check if location is a wall (obstruction).
- If not place 4 particles
  - cardinal directions.
- Get cardinal distances.

University of Cincinnati

Generate Particles

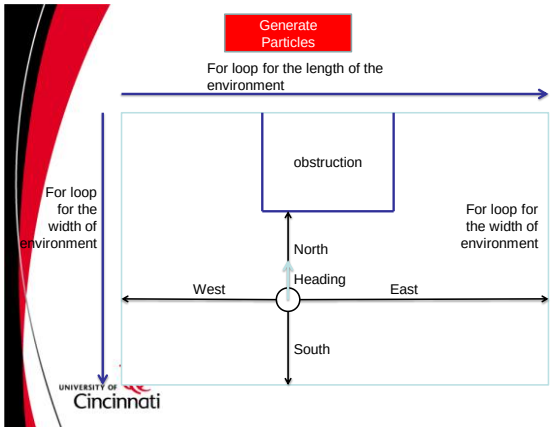
Code Logic:
For length of environment

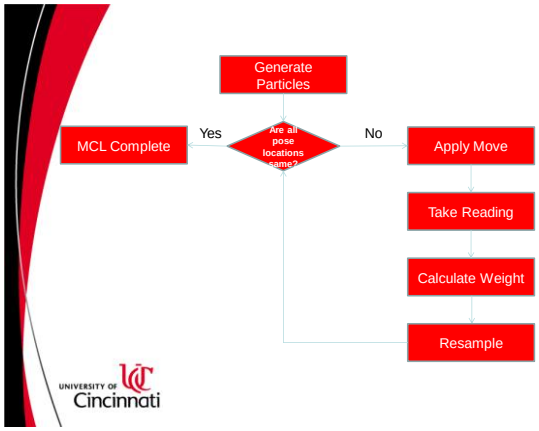
For width of environment

If location is not an obstruction

create pose (x, y, 0°, North, South, East, West)
create pose (x, y, 90°, North, South, East, West)
create pose (x, y, 180°, North, South, East, West)
create pose (x, y, 270°, North, South, East, West)

University of Cincinnati





Same Particles

- Check zero particle against all other particles.
- If particles are the same increase a counter variable.
- If the counter variable is above the convergence criteria MCL has finished.

University of Cincinnati

Same Particles

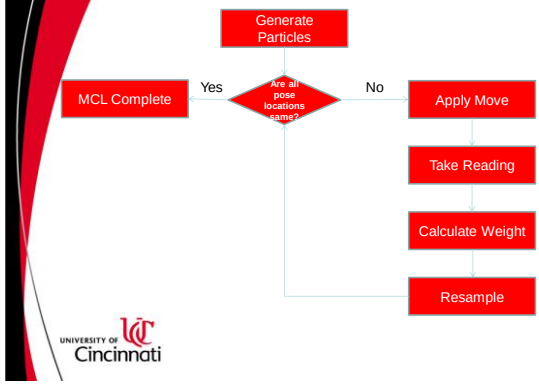
Code Logic:
For length of pose set

If Particle(0) == Particle(i)

Increase counter

If counter is greater than convergence percentage x total pose set size

MCL have finished successfully



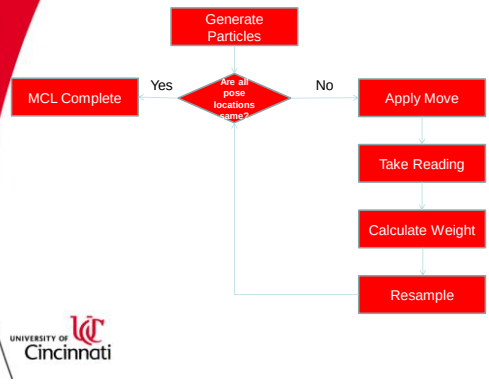
Apply Move

- Select a move (forward, backward, left, right).
- Send the move to the robot.
- Apply move to each particle in the pose set.
- Update each pose (location, heading, and distances to nearest obstruction).

Apply Move

Code Logic:
Select move (1-4 w/ 1: Forward, 2: Backward, 3: Left, 4: Right)
Send move to robot
For length of pose set

Update x as a function of move and current heading
Update y as a function of move and current heading
Update heading as a function of move
Update North as a function of move and current heading
Update South as a function of move and current heading
Update East as a function of move and current heading
Update West as a function of move and current heading



Take Reading

- Command robot to take sonar reading.
- Set a while loop to ensure distance <255.
- Send reading back to PC to be used in weight calculation.

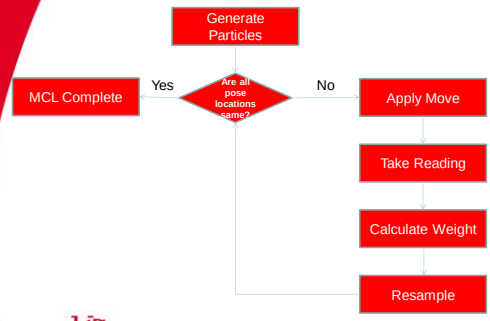
Take Reading

Code Logic:
Send reading command to robot

While the reading >255

Continue to take a reading until reading is <255

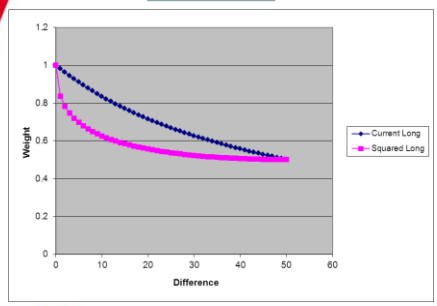
Send reading back to PC for use in weight calculation



Calculate Weight

- Check the current heading
- Select the pose distance to nearest obstruction
  - $\text{diff} = \sqrt{\text{abs}(\text{robotReading}^2 - \text{particleReading}^2)}$
- Normalize this number such that it > 0 and < 1
  - $\text{weight} = 1/\ (1 + (\frac{\text{diff}}{\text{robotReading}}))$

Calculate Weight



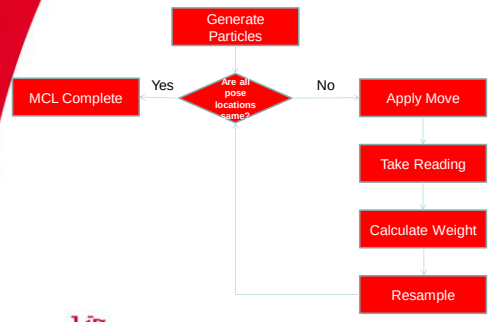
Calculate Weight

Code Logic:
For length of pose set

Select Distance to nearest obstruction as a function of heading

Calculate difference between pose distance and robot distance

Normalize difference into a weigh such that 0>weight>1.



Resample

- Select a random number between 0.5 and 1.
- Copy pose set
- Clear pose set.
- Check weight of particles in copy set against random number.
- If weight > random number
  - Copy particle
- If weight < random number
  - Move on (eliminate particle)

Resample

Code Logic:
Select a random number 0.5>random number>1

Create old pose set = current pose set

Clear current pose set

While number of particles in current pose set < number of particles in old pose set

For length of old pose set

If particle weight > random number and distances to obstruction >= 0

Create copy of particle in current pose set
Increase counter of current pose set by 1

Issues

- Inaccurate compass
- MCL not converging
- Pixel-Centimeter errors

Future Work

- More accurate sonar
- Check location
- Symmetry solver
- Smarter measurement algorithm

Demonstration

- Questions?