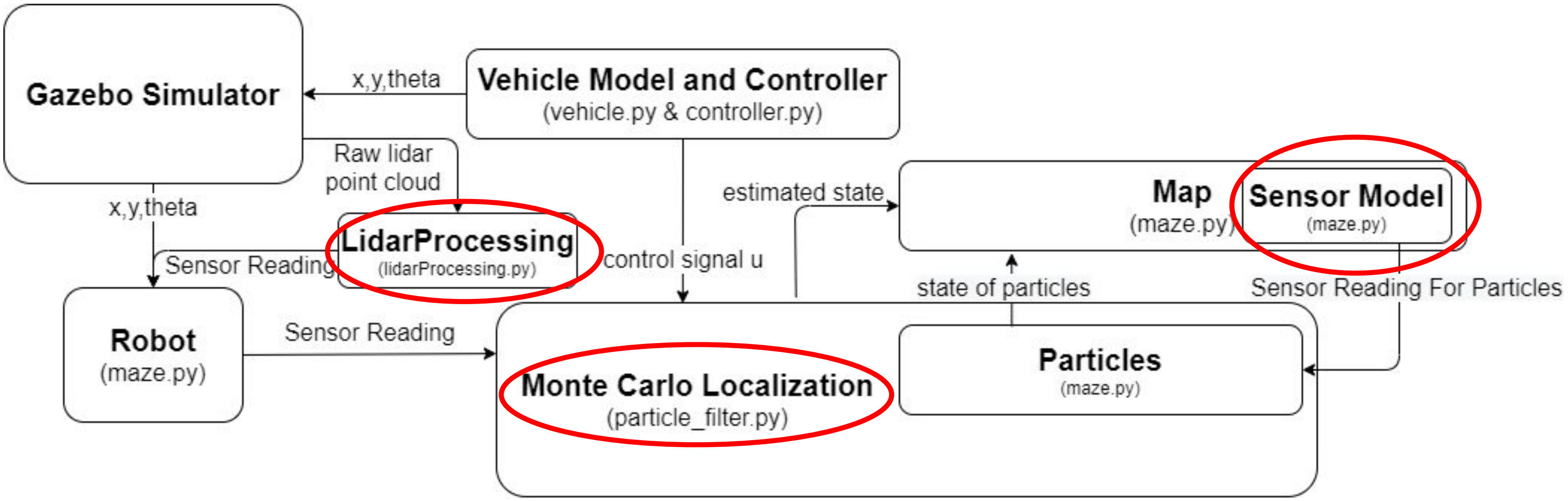
MP3: Filtering and Localization

Eric Liang

Overview

- Demo due 4/8/2021, Report due 4/9/2021
- 3 Written Questions, 4 Implementation Questions
- Written Questions:
  - Bayes Filter
  - Particle Filter
  - MP0 Revisited
- Implementation Questions
  - Number of Particles
  - Sensor Limit
  - Environment
  - Sensor Model

Module Architecture

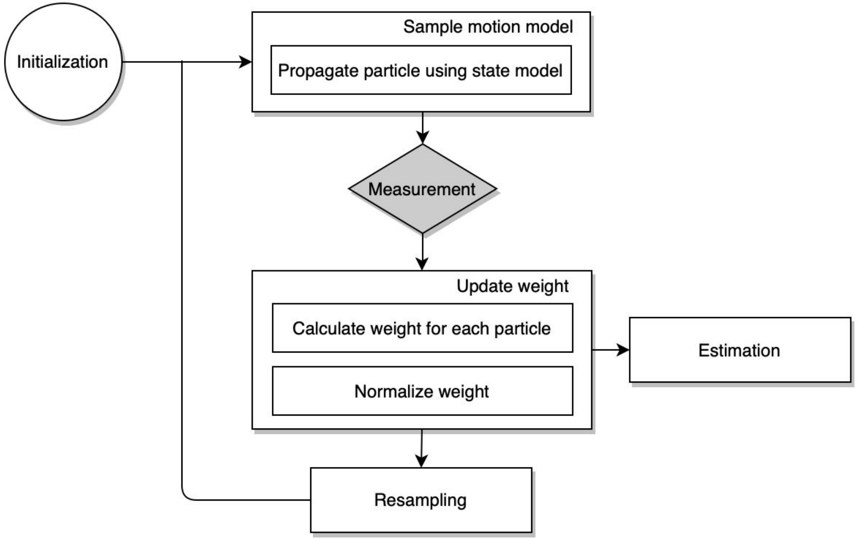


Particle Filter: Main Function

$X_t = x_t^{[1]}, x_t^{[2]}, \dots x_t^{[M]}$ particles

Algorithm MCL(X_{t-1}, u_t, z_t,m):
$\bar{X}_{t-1} = X_t = \emptyset$
for all m in [M] do:
$\quad x_t^{[m]} = \boldsymbol{sample\_motion\_model}(u_t\ x_{t-1}^{[m]})$
$\quad w_t^{[m]} = \boldsymbol{measurement\_model}(z_t, x_t^{[m],m})$
$\quad \bar{X}_t = \bar{X}_t + \langle\, x_t^{[m]}, w_t^{[m]} \rangle$
end for
for all m in [M] do:
$\quad$ draw i *with probability* $\propto w_t^{[i]}$
$\quad$ add $x_t^{[i]}$ *to* X_t
end for
return X_t

```
def runFilter
    while True:
        sampleMotionModel(p)
        reading = vehicle_read_sensor()
        updateWeight(p, reading)
        p = resampleParticle(p)
```



Sample Motion Model

- Imagine particles as multiple robots that have the same motion model as the actual robot
- Control: Linear and Angular Velocity
- State: Position and Heading
- From control to state: Integration

$$\dot{x} = vcos(\theta)$$

$$\dot{y} = vsin(\theta)$$

$$\dot{\theta} = \delta$$

Integration

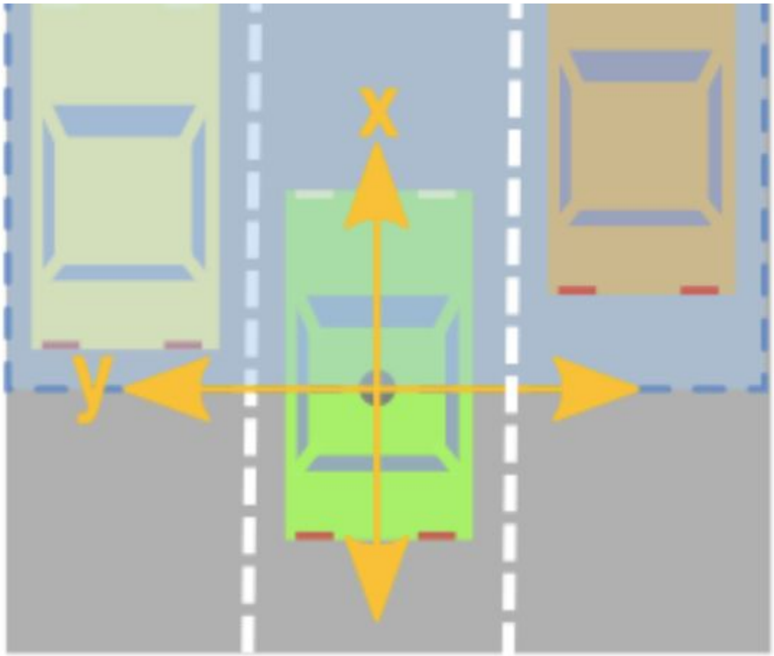
- Basic Idea: y += dy * Δt
  - Simple
  - Inaccurate
- SciPy ODE Integrator (scipy.integrate.ode)
  - Slightly Slower (Depends on the integrator)
  - Accurate
- How to use?
  - Set initial Value
  - Find f(t, y, ...(controls)) such that dy(t) = f(t, y, ...(controls))
  - Use a list of control signals to update integrator and integrate with respect to t
  - Why a list: Integrator may not be fast enough to synchronize with simulator

Integration Tricks

- All the particles move the same way.
  - Only Initial State is Different
  - ODE is expensive
  - Could you think of a way to apply ODE result on all particles?
- ODE Accuracy v.s. Frequency Trade Off
  - Inaccurate/simple integrator may outperform slower/accurate integrator because it can update and converge faster
  - Sweet Spot: Trial and Error

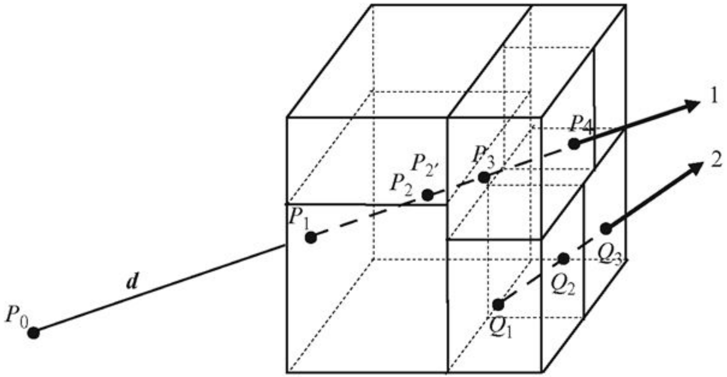
Sensor Model: Lidar

- Lidar: Coupled Distance and Heading Sensor
- Interpretation: 3D Point Cloud (X, Y, Z)
- Only Want 8 Directions
  - Provided: Front/Rear/Left/Right
  - TO-DO: Front-Right/Front-Left/Rear-Right/Rear-Left
- Conversion
  - Filter Points According to Criteria
  - Find Mean of Filtered Points



Sensor Model: Particles

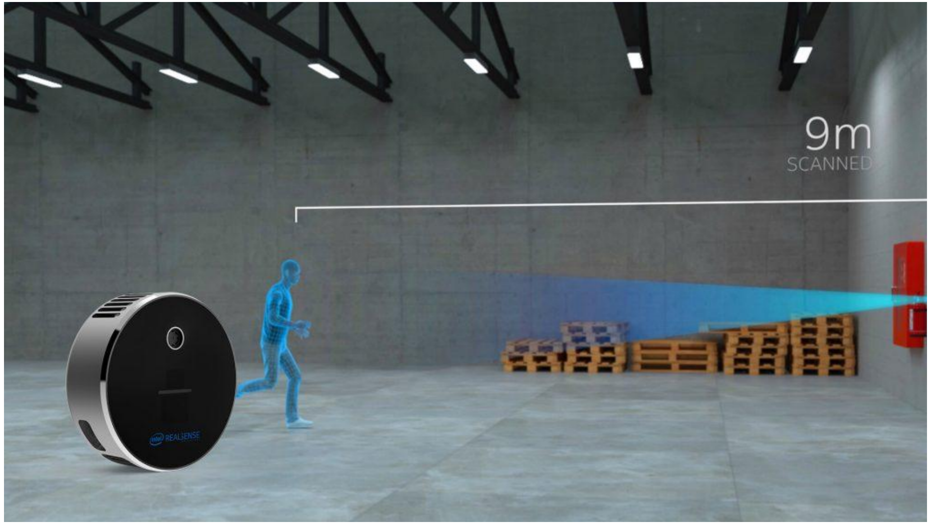
- How do we find out the distances in the 8 directions for particles?
    - Shooting rays and see if it hits walls in map
    - Record the distance
- Ray is defined by?
    - Initial Point (Car)
    - Orientation/Heading (?)
- Potential Problems
    - May miss if step is too large
    - Slow: Particle Position Dependent



http://what-when-how.com/advanced-methods-in-computer-graphics/collision-detection-advanced-methods-in-computer-graphics-part-6/

Sensing Limit

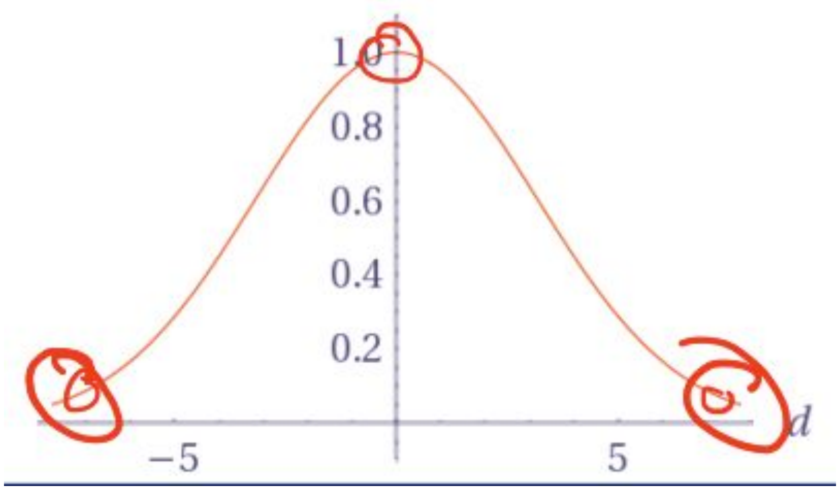
- Lidar and many other distance sensors have max range.
- In real life, your particle sensor model should reflect the behavior of actual sensor well enough to run the particle filter.
- Sensor limit as parameter
    - Estimation Accuracy
    - Converging Speed
    - Computation cost



https://www.intelrealsense.com/optimizing-the-lidar-camera-l515-range/

Update Weight

- Basic Idea: The Closer the Better
- Compare
  - Sensor Measurement (4 or 8)
  - Sensor Model (4 or 8)
- How? Gaussian Kernel (weight_gaussian_kernel)
  - Tune standard deviation
  - Or you can do something different
- Important Notice: Normalize to 1



Resampling Particles

- Update Belief by Updating Distribution of Particles
- Multinomial Resampling
  - Calculate Cumulative Sum of Weights (Again, normalize to 1 in the previous step)
    - NumPy cumsum
  - Randomly generate a number and determine which range in that cumulative weight array to which the number belongs
    - NumPy searchsorted/ Bisect bisect_left
  - Which index corresponds to that range? (Think about it)
  - Repeat Until Reach Desired Number of Particles
- There are many other resampling method: check lab manual

Other things to consider...

- What should you do when particles run inside walls or out of the maze?
- Does motion model perfectly matches simulator? What about noise?
- What if my particle filter converges and suddenly loses track? How should I recover?

Demo

- Students need to show their particle filter
  - Converges within reasonable number of iterations
  - Closely tracks the position of the vehicle
  - Can extend from 4 directions to 8 directions

Questions?

- This is a ~~much~~ harder MP compared to MP2
- Start early