CS 7638 Artificial Intelligence for Robotics Lecture Notes

Jie Wu, Jack

Fall 2022

1 Localization Overview (Histogram Filters)

1.1 Localization

How can we know where we are within 10cm accuracy? This is a problem of **localization**. Note that much of the content in this lesson assumes you have a good map first. Without it, the techniques here either won't work or won't work very well. There is also another version of localization called SLAM, or simultaneous localization and mapping, that does not need a good map prior to beginning.

1.2 Total Probability

The maximum confusion about location is a uniform probability distribution of position. But usually there are bumps associated with landmarks in the probability distribution of position to distinguish them from the environment (e.g. doors vs. wall). This is our **posterior belief function**, where the word "posterior" means after a measurement and the bumps together express our best belief of current position. Motion would lead to shift of the belief function in the direction of motion. This process is called *convolution*. If after another measurement the posterior belief function has only one bump, then we said the robot has localized itself and we have localization.

1.3 Inaccurate Robot Motion

It is common case that robots move with uncertainty, sometimes fall short and sometimes overshoot. This is the primary reason why localization is hard.

1.4 Limit Distribution Quiz

What is the limit of the probability distribution of the robot's position after it has run for infinitely many steps? It is the uniform distribution. The reason is that due to the inexact motion of the robot, every time it moves we lose the information about its position. Hence while measurement reduces robot position uncertainty, motion increases it.

1.5 Sense and Move

Localization is nothing but iteration of sense and move. An initial belief is tossed into the loop and then undergoes cycles of sense and move. At the sense stage we gain information about robot position through measurement against landmarks, whereas at the move stage we lose information about robot position due to its inexact motion.

1.6 Localization Summary

Localization maintains a probability distribution of robot position which represents our belief of robot position. The measurement update or sense function takes our initial belief and multiply the probability distribution with the measurement outcome followed by normalization. The convolution or move function adds the probability of reaching current position from all possible prior positions.

1.7 Bayes Rule

For the sense stage, let X be grid cell and Z measurement. Bayes rule reads

$$p(X_i|Z) = \frac{p(Z|X_i)p(X_i)}{p(Z)} = \frac{p(Z|X_i)p(X_i)}{\sum_i p(Z|X_i)p(X_i)}$$

1.8 Theorem of Total Probability

For the move stage, let X_i^t be current position and X_j^{t-1} prior position. Theorem of total probability reads

$$P(X_i^t) = \sum_j P(X_i|X_j)P(X_j^{t-1})$$

The operation of a weighted sum over other variables is often called **convolution**.

2 Problem Set 1

2.1 Localization

In a grid-based localization, the memory required scale exponentially in the number of state variables. Because if each state variable takes 20 possible values and there are n state variables, then there are 20^n possible combinations of state variable values and thus 20^n possible states. This exponential scaling law is the biggest disadvantage of grid-base localization or histogram filter.

3 Kalman Filters

3.1 Tracking Intro

Kalman filter is a popular technique for estimating the state of a system. If the state of a system is represented by its position, then Kalman filter performs exactly localization. Different from the grid-based or Monte Carlo localization discussed above, Kalman filter estimates continuous states instead of discrete states, and gives us a unimodal distribution instead of a multimodal distribution. Both of them are applicable to robot localization and tracking other vehicles. Later in the class we will learn particle filter which addresses the same problem but are continuous and multimodal.

3.2 Gaussian Intro

In Kalman filter the probability distribution of a robot's position is given by a Gaussian distribution, which is characterized by mean μ and variance σ^2

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

The product of two Gaussians is still a Gaussian.

3.3 Measurement and Motion

Identical to localization, Kalman filter iterates two different things—measurement updates and motion updates, the latter is usually called *prediction* instead.

3.4 Predicting the Peak

The peak of the product of two Gaussians is higher than either of the two, while the covariance of the product is smaller than either of the two. This is because by combining the information content from the two Gaussians we gain information thus reduce the uncertainty.

3.5 Parameter Update

Suppose we multiply two Gaussians as in Bayes rule—a prior belief with mean μ and variance σ^2 and a measurement probability with mean ν and variance ρ^2. The new mean is the sum of the old means weighted by the other's variance, and the new variance is half of the harmonic mean of the old variances

$$\lambda = \frac{\rho^2\mu + \sigma^2\nu}{\rho^2 + \sigma^2} \qquad \xi^2 = \frac{1}{\dfrac{1}{\sigma^2} + \dfrac{1}{\rho^2}}$$

The mean formula suggests that the new mean would be closer to the Gaussian with smaller variance (⇒ the more certain one), while the variance formula suggests that the new variance will be smaller than either of the old ones.

3.6 Gaussian Motion

Since inexact motion increases uncertainty, the resulted Gaussian would have a larger variance. Mathematically suppose the mean and variance of our prior belief of position is μ and σ^2 respectively and the inexact motion can be described by a Gaussian with mean ν and variance ρ^2, the mean and variance of the new belief or probability distribution of position is just the sum of the two old ones

$$\lambda = \mu + \nu \qquad \xi^2 = \sigma^2 + \rho^2$$

3.7 Kalman Filter Code

Suppose the initial belief of position has a mean that is very different from the initial measurement outcome and a very small variance (wrong but stubborn belief). As time goes by the update function will gradually drag the belief towards measurement outcome.

3.8 Kalman Filter Land

The probability distribution of a D-dimensional Gaussian is characterized by a mean vector μ and a covariance matrix Σ

$$f(x) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

When we have a 2D Gaussian we can plot its contour lines in a Cartesian coordinate. If the contour is a tilted elongated ellipse then the uncertainty of x and y is correlated, which means that if I get information about x then it suggests the most probable y value. In Kalman filter land the two dimensions are position x and velocity $\dot{x}$.

3.9 Kalman Filter Prediction

Initially we are certain about the position of the robot but not its velocity, thus the contour lines of our belief is an ellipse elongated in the $\dot{x}$ or y direction. Interestingly there is a correlation between position and velocity based on Newton's first law. Take the initial position as the origin i.e. $x = 0$. Suppose our belief of velocity is v. Then we should predict that the robot position will be at $x = v$ after one time step and our belief of velocity remains unchanged at v. This is the consequence of Newton's first law, which states that every object would remain at rest or in uniform motion in a straight line unless compelled to change its state by the action of an external force. As a consequence the contour lines of the Gaussian representing the motion should be an ellipse elongated along the $y = x$ diagonal line. Suppose after one time step we measure the robot is positioned at $x = 2$ but still without any information of its velocity. This measurement outcome can again be represented by an ellipse elongated in the y direction. But multiplying this measurement with our prediction of robot position based on Newton's first law, we obtain a 2D Gaussian localized at $(x, y) = (2, 2)$, which represents a good estimate of both position and velocity. Multiplying this updated belief with the 2D Gaussian predicting future motion based on Newton's first law, we get an even more localized 2D Gaussian further away along the $y = x$ diagonal line. The miracle here is that although we can observe only one state variable, we are able to measure that variable so as to infer the other unobserved state variable, which is based on Newton's first law $x \leftarrow x + \dot{x}\Delta t$.

3.10 More Kalman Filters

The variables of a Kalman filter are often called states because they reflect the states of a system. The states are separated into two subsets—the observables and the hidden. The key mechanism that empowers Kalman filters is that subsequent observations of the observables give us information about the hidden.

3.11 Kalman Filter Design

To design a Kalman filter we need (1) a state transition function F (2) a measurement function H. In our 1D motion example above they are given by (where we take $\Delta t = 1$ for Newton's first law)

$$\begin{pmatrix} x \\ \dot{x} \end{pmatrix} \longleftarrow \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{F} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad Z \longleftarrow \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{H} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

Let x be the estimate of the observables, P the uncertainty covariance of our belief of the observables, Z the measurement of the observables, and U the motion vector. With a Kalman filter characterized by state transition matrix F and measurement matrix H, the prediction is given by

$$\begin{aligned} x &= Fx + U \\ P &= F \cdot P \cdot F^T \end{aligned}$$

and the measurement update is given by (where $y = Z - H \cdot x$ is the discrepancy from the measurement or error and R is the measurement noise)

$$\begin{aligned} S &= H \cdot P \cdot H^T + R \\ K &= P \cdot H^T \cdot S^{-1} \\ x &= x + K \cdot y \\ P &= (I - K \cdot H) \cdot P \end{aligned}$$

where K is called the *Kalman gain.*

4 Problem Set 2

4.1 Heavytail Gaussian

The asymptotic behavior of Gaussian is that as x goes to infinity it will approach zero

$$\lim_{|x|\to\infty} f(x) = \lim_{|x|\to\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] = 0$$

5 Office Hour Week 2

Bayes rule allows you to incorporate sensor measurements one after another.

6 Particle Filters

6.1 Efficiency

When it comes to scaling in the number of dimensions of the state space, histogram filter is exponential in the amount of memory required whereas Kalman filter is quadratic (covariance matrix).

6.2 Exact or Approximate

Histogram filter is approximate because the world is continuous rather than discrete. Kalman filter is also approximate because it is exact only for linear systems whereas the world is nonlinear.

6.3 Particle Filters

The state space of particle filter is continuous and its belief can be multimodal. It is also an approximation of the real world. In terms of efficiency there is no consensus though. The required storage scales exponentially with the number of dimensions in certain cases, e.g. when the number of dimensions is more than four. But in other cases, mostly in tracking domain, they tend to scale much better. Actually the key advantage of particle filter is none of these but the ease of programming.
The essence of particle filter is to let particles guess where the robot might be moving, and have them survive according to the survival of the fittest rule such that particles that are more consistent with the measurements are more likely to survive. As a result places of high probability will collect more particles, and therefore be more representative of the robot's posterior belief.

6.4 Importance Weight

In particle filters each particle represents a guess of robot position, and the measurement of its position is a predicted measurement associated with this guess. The mismatch of the predicted measurement and the actual measurement gives rise to an **importance weight** for this particular guess. We then sample with replacement these particles according to their importance weights (probability proportional to importance weight). Those that are selected survive whereas those that are not demise. This step is called *resampling* and it completes one iteration of particle filtering.

6.5 Resampling Wheel

The pseudocode is

```
index = uniform random$[1..N]$
$\beta = 0$
for $i = 1..N$
    $\beta \leftarrow \beta +$ uniform random$[0..2 \cdot w_{\max}]$
    while $w[\text{index}] < \beta$
        $\beta \leftarrow \beta - w[\text{index}]$
        index $\leftarrow (\text{index} + 1)\,\%N$
select $p[\text{index}]$
```

6.6 Filters

Putting importance weight and resampling into the context of filters

$$\text{measurement update:} \quad P(X|Z) \underbrace{\propto}_{\text{resampling}} \underbrace{P(Z|X)}_{\text{importance weights}} \times \underbrace{P(X)}_{\text{particles}}$$

$$\text{motion update:} \quad P(X'|X) = \sum \underbrace{P(X'|X)}_{\text{set noise \& move}} \times \underbrace{P(X)}_{\text{particles}}$$

7 Kinematic Bicycle Model

7.1 Representation

We will use a 2-wheel vehicle to approximate the 4-wheel vehicle to make the calculation easier. This model is known as the **bicycle model**.

7.2 Control

The bicycle model has two controls

steering angle: the angle the front wheel makes relative to the orientation of the vehicle (positive if to the left)

forward movement: the distance from the rear axle at the start of the movement to the rear axle at the end of the movement, the rear axle and thus the rear wheel will always be parallel to the orientation of the vehicle

7.3 Robot Pose

The robot pose consists of three variables—the (x, y) location of the robot and the orientation angle θ.

7.4 Simple Movement

Simple movement involves no rotation of front wheel i.e. no change in the orientation angle $\alpha = 0$. In this case the new location is given by

$$x \leftarrow x + d\cos\theta$$
$$y \leftarrow y + d\sin\theta$$

7.5 Compound Movement

In a compound movement involving both steering angle rotation and forward movement, there will be two tracks left by the wheels, which are in fact concentric. The inner circle is left by the rear wheel, which is our concern.

7.6 Problem to Solve

We will solve (1) radius R (2) center point (3) turning angle (4) offset from center, which are depicted in Figure 1 below.

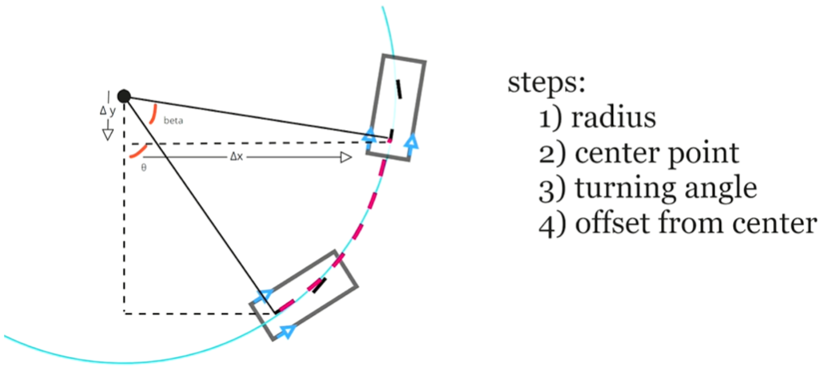

Figure 1: The problem to solve for bicycle model

7.7 Circle Center

As the steering angle is increased, the center of the circle becomes closer to the robot.

7.8 Radius

As depicted in Figure 2 below, the radius and the robot form a right angle. According to trigonometry the radius is thus given by (where α is the steering angle)

$$\text{radius} = \frac{L}{\tan \alpha}$$

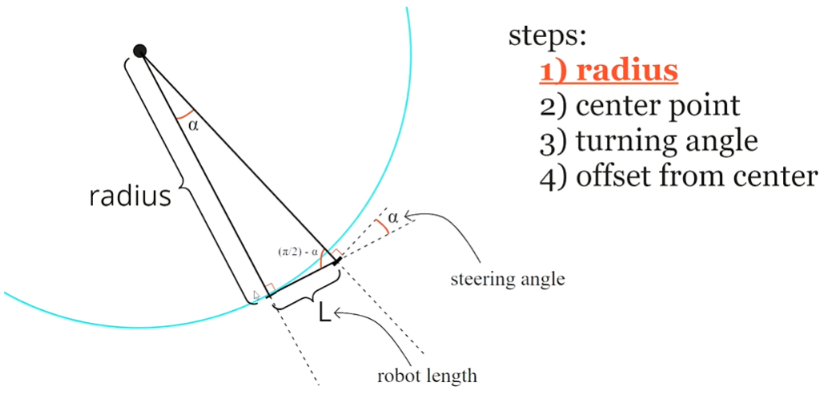

Figure 2: The trigonometry for solving the radius

7.9 Center Point

To find the coordinates of the center point, we again use the right angle formed by the radius and the robot as shown in Figure 3 below. The relevant trigonometry is (where θ is the orientation angle)

$$\begin{aligned}\text{x\_dist} &= \text{radius} \times \sin(\theta)\\ \text{y\_dist} &= \text{radius} \times \cos(\theta)\\ \text{center\_x} &= \text{x} - \text{x\_dist}\\ \text{center\_y} &= \text{y} + \text{y\_dist}\end{aligned}$$

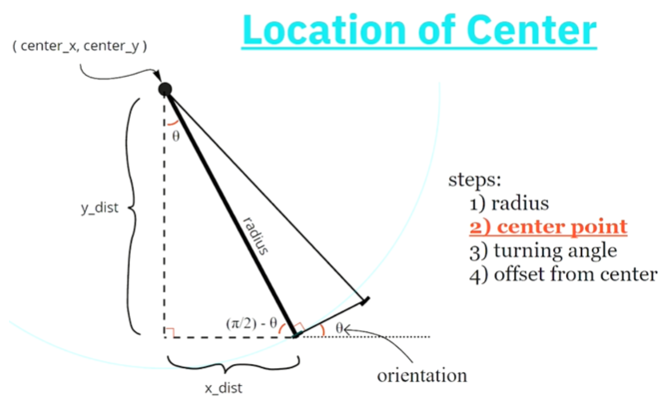

Figure 3: The trigonometry for solving the center point position

7.10 Turning Angle

To find the turning angle, we just need to apply the formula of arc length as depicted in Figure 4 below.

$$\beta = \frac{\text{forward distance}}{\text{radius}}$$

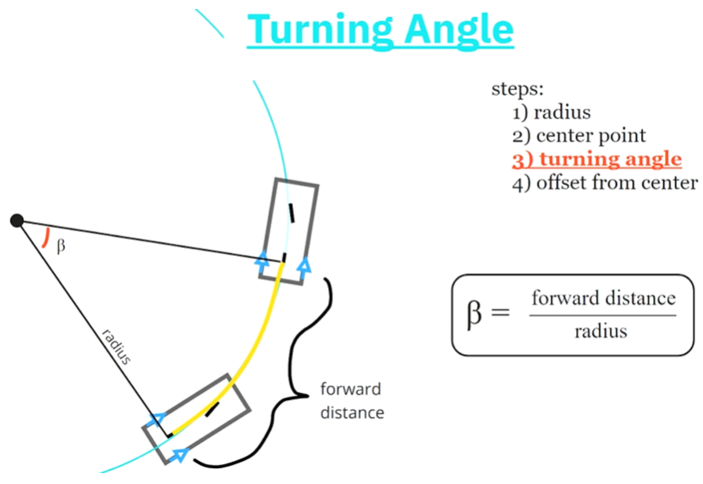

Figure 4: The arc for solving the turning angle

7.11 New Orientation

The new orientation angle can be read directly from the geometry as depicted in Figure 5 below.

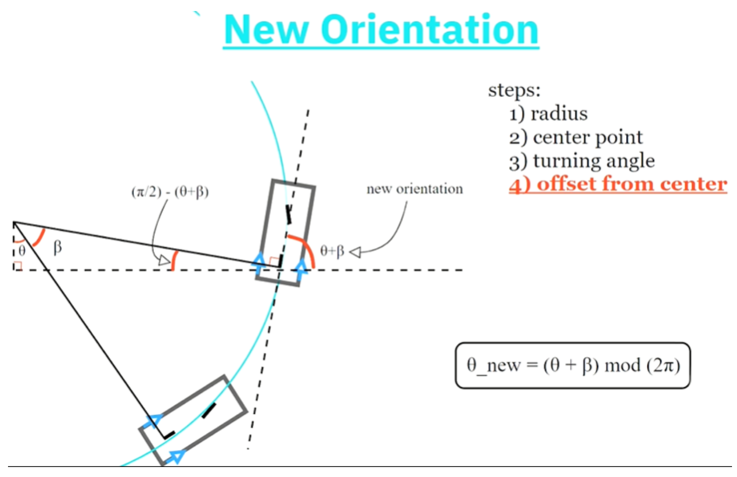

Figure 5: The geometry for reading the new orientation angle

7.12 Offset from Center

To find the offset from the center point of the robot's new position, we just need to reverse the formula for the center point using the new orientation angle $\theta + \beta$ as shown in Figure 6 below.

$$\begin{aligned} \text{x\_dist\_new} &= \text{radius} \times \sin(\theta + \beta) \\ \text{y\_dist\_new} &= \text{radius} \times \cos(\theta + \beta) \\ \text{x\_new} &= \text{center\_x} + \text{x\_dist\_new} \\ \text{y\_new} &= \text{center\_y} - \text{y\_dist\_new} \end{aligned}$$

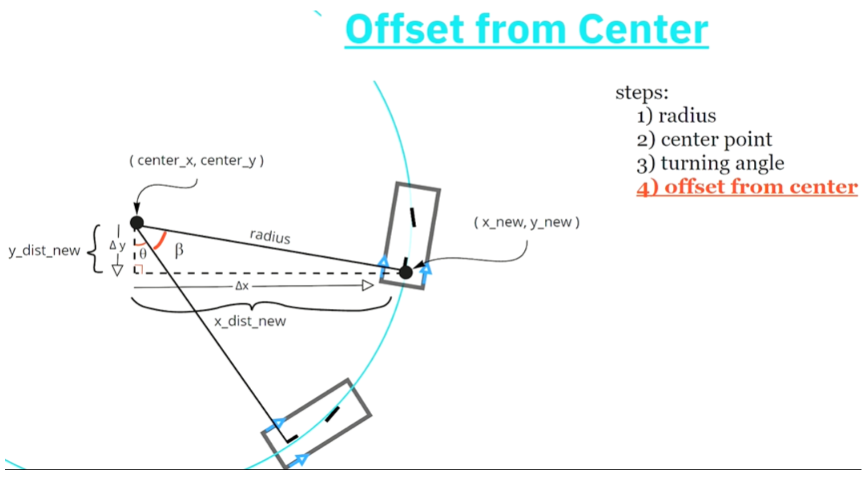

Figure 6: The trigonometry for solving the offset from the center point of the robot's new position

7.13 Formulas

Now we consolidate all the formulas for the bicycle model in Figure 7 below.

8 Problem Set 3

8.1 Single Particle

A particle filter with only one particle will

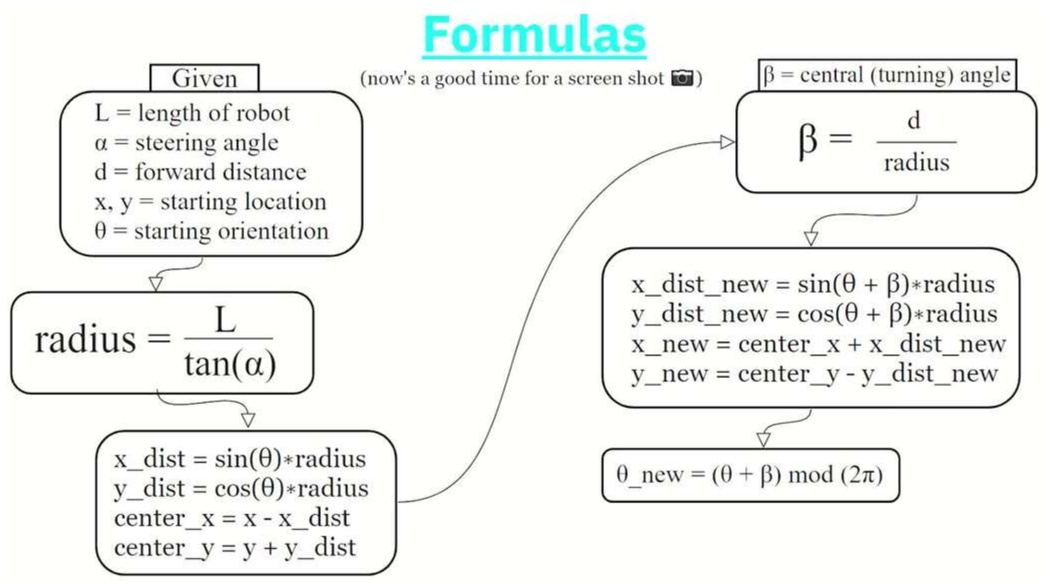

Figure 7: All the formulas for the bicycle model

- ignore robot measurement, because the measurement sets the importance weight in the resampling process but with only one particle it will be resampled regardless of the weight
- likely fail, because it ignores robot measurement

9 Office Hour Week 3

Which filter to choose in what situation?

particle filter: the easiest to implement, but its complexity scales exponentially with the number of dimensions

Kalman filter: the only filter that doesn't scale exponentially with the number of dimensions, but it is unimodal thus it can't support multiple hypotheses

histogram filter: applicable in situations similar to particle filter, but is more systematic than particle filter (in particle filter if you lose track of the correct hypothesis you might never regain it later), however its accuracy is limited by the resolution of the grid

The recommendation is that

- if you have a multimodal distribution, use particle filter if you can
- if the state space is continuous with a unimodal distribution, use Kalman filter

Switching filters on the fly is not recommended, because that leads to moments of increased uncertainty. The most common way of combining filters is the Rao-Blackwellized partilce filter. Rao and Blackwell found out that in a particle domain, sometimes if we nail certain dimensions with particles, everything else conditional on the particle becomes Gaussian or unimodal. Then we can exploit the efficiency of Kalman filter that is now attached to individual particles, which enables the particle filter to estimate in spaces with hundreds of dimensions.
If a particle moves into an invalid space, you simply kill that particle.
Dynamically setting the number of particles is a good idea under certain circumstances, e.g. when all the particles are centered in one location we don't need that many particles. The way to set the total

number of particles is to look at the total non-normalized importance weights. If all the importance weights are large, then the particles are doing a good job tracking thus we don't need that many particles. In contrast if all the importance weights are small, then the particles are doing a bad job tracking thus we need more particles. A good heuristic is to particle sample until the non-normalized importance weights reach a certain threshold upon which stop sampling.

10 Search

10.1 Motion Planning

A motion planning problem is formulated as follows: given a map, a starting location, a goal location, and a cost function (e.g. time spent), find a minimum cost path from the starting location to the goal location.

10.2 First Search Program

The idea is to keep a list of nodes that we would like to explore from (remember to check mark those nodes that have already been visited) after confirming that it is not the goal. At each iteration the list is replaced by the nodes that our footage has expanded to. At the same time we maintain the number of steps that it takes to reach the nodes in question. When the list contains more than one node, we choose to expand from the one with the smallest number of steps (a tie breaker may be needed if the number of steps are the same).

10.3 A-Star

A-star search algorithm utilizes a heuristic function which assigns to each cell of the grid an optimistic guess of the distance from that cell to the goal, optimistic in the sense that it ignores obstacles. The heuristic function helps to decide which is probably the better direction to search for in case there is more than one choice. If the heuristic function is zero throughout, then A-star search algorithm is identical to the expansion search algorithm above.
To implement A-star search algorithm, we simply need to replace the number of steps to reach the node of interest with the number of steps plus the value of the heuristic function at that cell, and then choose to expand from the node with the smallest sum.

10.4 Dynamic Programming

An alternative method of motion planning is dynamic programming. It is formulated as given a map and a goal what is the best path from anywhere. Thus dynamic programming gives you a motion plan for every position, which is called a **policy** that maps a grid cell to an action (⇒ plan vs. policy).

10.5 Computing Value

A **value function** associates to each grid cell the length of the shortest path to the goal. Mathematically it is recursively defined as

$$f(x,y) = \min_{x',y'} f(x',y') + 1 \qquad \text{with } f(\text{goal}) = 0$$

With a value function, the optimal control action is obtained by minimizing the value.

10.6 Left Turn Policy

With the additional parameter orientation, robot position is now defined by a 3-tuple (x, y, orientation). As a result the policy of dynamic programming should be 3D instead of 2D, with the extra dimension devoted to orientation.

11 Problem Set 4

A heuristic function is *admissible* if its value is less than or equal to the distance to goal. As long as this inequality holds, even if the value is absurdly less than the actual distance it is still admissible.
If a heuristic is not admissible, then A-star search algorithm may find a suboptimal path. Note that the algorithm would not fail to find a path if there exists one, albeit it may not be the optimal one.

12 Office Hour Week 4

A-star search algorithm cannot deal with branching outcomes where you flip a coin. It cannot deal with information gathering either, which pertains to actions just for the sake of reducing uncertainty. Dynamic programming can achieve these, but it is computationally very inefficient and does not scale well with the number of dimensions.
In A-star you can certainly designate multiple states to be goals, so does dynamic programming.
What Google self-driving car has been doing is pre-caching a lot of the principal subplans, so that the remaining planning problem of finding the shortest path becomes mostly a table lookup.
Two heuristics are used for Google self-driving car

2D planning heuristic: the motivation is that 2D planning is much faster than 3D planning

obstacle-free heuristic: 3D planning without obstacles can be done once and for all

The general principle of heuristic function selection is that, as long as you have a good way of planning quickly by lessening constraint, that tends to be a good heuristic for A-star.

13 PID Control

13.1 Robot Motion

We want to generate smooth paths i.e. we want to minimize the maximum steering angle change at all steps.

13.2 Smoothing Algorithm

First initialize $y_i = x_i$, then minimize all $(y_i - x_i)^2$ and $(y_i - y_{i-1})^2$ and $(y_i - y_{i+1})^2$ through weights α and β i.e.

$$\min_{y_i} \left[\frac{\alpha}{2}(y_i - x_i)^2 + \frac{\beta}{2}(y_i - y_{i-1})^2 + \frac{\beta}{2}(y_i - y_{i+1})^2 \right]$$

13.3 Path Smoothing

We use gradient descent to minimize the three terms

$$F(y_i) = \frac{\alpha}{2}(y_i - x_i)^2 + \frac{\beta}{2}(y_i - y_{i-1})^2 + \frac{\beta}{2}(y_i - y_{i+1})^2$$

$$y_i \leftarrow y_i + \alpha(x_i - y_i) + \beta(y_{i+1} + y_{i-1} - 2y_i) = y_i - \nabla F(y_i)$$

$\Rightarrow$ to see why it is plus sign in front of α and β consider what if $y_i > x_i$ and $y_i > (y_{i-1} + y_{i+1})/2$

13.4 Proportional Control

Steering in proportion to crosstrack error

$$\alpha = -\tau \text{CTE}$$

is a proportional or P controller. The problem with this P controller is that it overshoots no matter how small the constant τ is. Because when the robot hits the reference trajectory, its wheels are oriented away from the trajectory if $\tau \neq 0$. The overshoot may be very small but it never converges. The robot will be in a marginally stable state.

13.5 Oscillation Quiz

Increasing τ increases the frequency of oscillation.

13.6 PD Controller

To avoid overshoots we can implement a PD controller, in which steering is proportion to not only crosstrack error but also the rate of change in crosstrack error

$$\alpha = -\tau_P \text{CTE} - \tau_D \frac{d}{dt}\text{CTE}$$

13.7 Systematic Bias

Even though the bias was in steering, it manifests itself as increase in crosstrack error through a shift in normal trajectory.

13.8 PID Implementation

To correct systematic bias we can implement a PID controller, in which steering is proportion to not only crosstrack error and the rate of change in crosstrack error but also the cumulative crosstrack error over time

$$\alpha = -\tau_P \text{CTE} - \tau_D \frac{d}{dt}\text{CTE} - \tau_I \int dt \text{CTE}$$

14 Problem Set 5

14.1 Missing Parameters

Initial move away from reference trajectory suggests the absence of correction in proportion to crosstrack error i.e. $\tau_p \approx 0$.
Growing oscillations suggests too small a differential term.

14.2 Constrained Smoothing

Previously we add $(x_{i-1} - x_i) - (x_i - x_{i+1})$ vector discrepancy penalty. The two vectors compared do not cross the fixed corner point if x_i is next to a corner. Thus we need a vector comparison that crosses the corner. If the point lies before the corner then it is $(x_{i+1} - x_i) - (x_{i+2} - x_{i+1})$ with x_{i+1} being the corner; if the point lies after the corner then it is $(x_{i-1} - x_i) - (x_{i-2} - x_{i-1})$ with x_{i-1} being the corner.

15 Office Hour Week 5

Twiddle is good for only one dimension, meaning it would fail if more than one correlated parameters need to be tuned simultaneously, and twiddle is susceptible to local minima because it is essentially a hill-climbing technique.

16 SLAM

16.1 Localization Quiz

	multi-modal	exponential
Kalman filter	no	no
histogram filter	yes	yes
particle filter	yes	yes

16.2 Planning Quiz

	continuous	optimal	universal	local
breadth first	no	yes	no	no
A-star	no	yes	no	no
dynamic programming	no	yes	yes	no
smoothing	yes	no	no	yes

where universal means once the solution is found it can be applied to arbitrary state, while local means it is only a local solution.

16.3 PID Quiz

	avoid overshoot	minimize error	compensate drift
P	no	yes	no
I	no	no	yes
D	yes	no	no

16.4 SLAM

SLAM = simultaneous localization and mapping.

16.5 Is Localization Necessary Solution

In nearly all cases of mapping we need to address the uncertainty in robot motion which will grow over time if without localization. Otherwise environment mapping will look bad.

16.6 Graph SLAM Quiz

Suppose we have one initial position and n motions and m measurements of landmarks, then the total number of constraints = 1 (initial location constraint) + n (relative motion constraints) + m (relative measurement constraints) $= 1 + n + m$.

16.7 Implementing Constraints Quiz

The matrix equation that enforces the relative motion constraint $x_1 - x_0 = +5$ is

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} -5 \\ 5 \end{pmatrix}$$

The matrix equation that enforces the relative motion constraints $x_1 - x_0 = +5$, $x_2 - x_1 = -4$ is

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -5 \\ 9 \\ -4 \end{pmatrix}$$

16.8 Adding Landmarks Quiz

The matrix equation that enforces the relative motion constraints $x_1 - x_0 = +5$, $x_2 - x_1 = -4$ and the relative measurement constraint $L_0 - x_1 = 9$ is

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ L_0 \end{pmatrix} = \begin{pmatrix} -5 \\ 0 \\ -4 \\ 9 \end{pmatrix}$$

16.9 Graph SLAM Quiz

Graph SLAM is all about local constraints and require additions instead of multiplications.

16.10 Matrix Modification Quiz

The matrix equation that enforces the relative motion constraints $x_1 - x_0 = +5$, $x_2 - x_1 = -4$ and the relative measurement constraint $L_0 - x_1 = 9$ and the initial location constraint $x_0 = 6$ is

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ L_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -4 \\ 9 \end{pmatrix}$$

16.11 Introducing Noise Quiz

Reduction in the last measurement (due to noise etc.) would not affect the initial location because it is an absolute constraint. But it does bring the last robot pose and the landmark closer to each other so as to compensate for the reduction. The preceding motion position is brought closer to the landmark too due to the relative motion constraint.

16.12 Confidence Measurement Quiz

The Ω and ξ matrix equation enforcing the various constraints above is really maximum likelihood in disguise. For example the Gaussian distribution of x_1 given a $+5$ motion measurement from x_0 is

$$f(x_1|x_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x_1 - x_0 - 5)^2}{\sigma^2}\right]$$

To maximize this probability density function we should let

$$\frac{x_1}{\sigma} - \frac{x_0}{\sigma} = \frac{5}{\sigma}$$

However σ introduces a notion of confidence that can be incorporated into the Ω and ξ matrix equation. If you have high confidence on the $x_1 - x_0 = 5$ constraint. You can represent it in the matrix equation as $100x_1 - 100x_0 = 500$ to make the relevant entries dominate over the others in the matrices.
Note that we can also write down the probability density function for x_2 given a -4 motion measurement from x_1 as

$$f(x_2|x_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x_2 - x_1 + 4)^2}{\sigma^2}\right]$$

The overall probability density $f(x_1, x_2|x_0)$ is the convolution of $f(x_2|x_1)$ and $f(x_1|x_0)$.

17 Problem Set 6

One weakness of SLAM is that, as we move along and map a world by seeing landmarks, the Ω matrix grows linearly with the length of the path even when the environment is fixed thus the map is of fixed size. To remedy this shortcoming online SLAM was introduced which contains only the most recent position in the path. Online SLAM is carried out in the following steps

1. for each new movement $r_t \to r_{t+1}$ expand the Ω and ξ matrices by 2 dimensions for the x- and y-coordinates of the new position r_{t+1}

2. apply the regular motion update to the expanded Ω and ξ

3. cut out the submatrices Ω' and ξ' involving only the new position r_{t+1} and the landmarks from the expanded Ω and ξ

4. denote the remaining three submatrices in the expanded Ω matrix by A, B, and A^T (from top right to bottom left), denote the remaining one submatrix in the expanded ξ matrix by C, the updated Ω and ξ matrices are given by

$$\Omega = \Omega' - A^T B^{-1} A \qquad\qquad \xi = \xi' - A^T B^{-1} C$$

   which can be proved to be equivalent to integrating out r_t from the joint Gaussian probability distribution of the two positions and measurements

18 Office Hour Week 6

Just as we add uncertainty to robot position in our system of equations, we can do exactly the same for landmarks to deal with dynamic environment. We can even introduce velocity of landmarks into graph SLAM.
The Ω matrix is the inverse covariance matrix, inverting it we get the covariance for all robot positions and landmarks.