Filtering Methods for Localization

Part 1 : Bayes Filter

University of Technology Sydney

Spring 2018

Probability Backgrounds

- Product Rule (Factorization)

$$P(X, Z) = P(X) \cdot P(Z|X)$$

- Sum Rule (Marginalization)

$$P(X) = \sum_i P(X, Y = y_i) = \sum_i P(X|Y = y_i)P(Y = y_i)$$

- Bayes Rule

$$P(X|Z) = \frac{P(X, Z)}{P(Z)} = \frac{P(X)P(Z|X)}{P(Z)}$$

Bayes Rule | Normalization trick

- Sometimes, the Bayes rule is given by

$$P(X|Z) = \frac{P(X,Z)}{P(Z)} = \frac{P(X)P(Z|X)}{P(Z)} = \frac{1}{\eta} \cdot P(X)P(Z|X)$$

- This trick is useful in implementation.

Bayes Rule | Normalization trick

Normalization Trick

$$P(X|Z) = \frac{P(X,Z)}{P(Z)} = \frac{P(X)P(Z|X)}{P(Z)} = \frac{1}{\eta} \cdot P(X)P(Z|X)$$

- Let's see why by a discrete example.

$$\begin{aligned} &P(X = x_i|Z) \\ =&\frac{P(X = x_i)P(Z|X = x_i)}{P(Z)} \end{aligned}$$

Bayes Rule | Normalization trick

Normalization Trick

$$P(X|Z) = \frac{P(X,Z)}{P(Z)} = \frac{P(X)P(Z|X)}{P(Z)} = \frac{1}{\eta} \cdot P(X)P(Z|X)$$

- Let's see why by a discrete example.

$$\begin{aligned} & P(X = x_i|Z) \\ = & \frac{P(X = x_i)P(Z|X = x_i)}{P(Z)} \\ = & \frac{P(X = x_i)P(Z|X = x_i)}{\sum_i P(X = x_i)P(Z|X = x_i)} \end{aligned}$$

Bayes Rule | Normalization trick

Normalization Trick

$$P(X|Z) = \frac{P(X,Z)}{P(Z)} = \frac{P(X)P(Z|X)}{P(Z)} = \frac{1}{\eta} \cdot P(X)P(Z|X)$$

- Let's see why by a discrete example.

$$\begin{aligned} & P(X = x_i|Z) \\ =& \frac{P(X = x_i)P(Z|X = x_i)}{P(Z)} \\ =& \frac{P(X = x_i)P(Z|X = x_i)}{\sum_i P(X = x_i)P(Z|X = x_i)} \end{aligned}$$

$$\bar{P}(X = x_i|Z) = P(X = x_i)P(Z|X = x_i)$$

$$\eta = \sum_i \bar{P}(X = x_i|Z)$$

$$P(X = x_i|Z) = \frac{1}{\eta} \cdot \bar{P}(X = x_i|Z)$$

- Both equations are doing the same thing!

Bayes Filter

- Prediction:

$$P(X_t|u_t) = \int P(X_t|X_{t-1}, u_t) \cdot P(X_{t-1}|z_{t-1}, u_{t-1}) \cdot dX_{t-1}$$

- Update:

$$P(X_t|z_t, u_t) = \frac{1}{\eta} \cdot P(z_t|X_t) \cdot P(X_t|u_t)$$

Bayes Filter

Bayes Filter

- Prediction:

$$P(X_t|u_t) = \int P(X_t|X_{t-1}, u_t) \cdot P(X_{t-1}|z_{t-1}, u_{t-1}) \cdot dX_{t-1}$$

- Update:

$$P(X_t|z_t, u_t) = \frac{1}{\eta} \cdot P(z_t|X_t) \cdot P(X_t|u_t)$$

Motion model

$$P(X_t|X_{t-1}, u_t)$$

Observation model

$$P(z_t|X_t)$$

Filtering Methods for Localization

Part 2 : Particle Filter

University of Technology Sydney

Spring 2018

Motivation

Bayes Filter - 3 pieces

- Prediction by motion model

$$\mathbf{P}(\mathbf{X_t}|\mathbf{u_t}) = \int \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}}, \mathbf{u_t}) \cdot \mathbf{P}(\mathbf{X_{t-1}}|\mathbf{z_{t-1}}, \mathbf{u_{t-1}}) \cdot dX_{t-1}$$

- Update by observation model

$$\mathbf{P}(\mathbf{X_t}|\mathbf{z_t}, \mathbf{u_t}) = \frac{1}{\eta} \cdot \mathbf{P}(\mathbf{z_t}|\mathbf{X_t}) \cdot \mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$$

Particle Filter

- **Target distribution:** approximated by samples/particles
- **Motion model:**
- **Observation model:**

Particles

- A partice is an extension of sample with 2 domains
  - **state:** An instantiation of the random variable it represents, i.e., a sample of the random variable.
  - **weight:** A real number belongs to $[0, 1]$ representing the importance/impact of the sample.

Particles

- A partice is an extension of sample with 2 domains
  - **state:** An instantiation of the random variable it represents, i.e., a sample of the random variable.
  - **weight:** A real number belongs to $[0, 1]$ representing the importance/impact of the sample.
- Some examples of particles

$$
\begin{aligned}
< (x_1, y_1),\ w_1 > &:< (1.2, 1.7),\ 0.050 > \\
< (x_2, y_2),\ w_2 > &:< (3.9, 5.8),\ 0.025 > \\
< (x_3, y_3),\ w_3 > &:< (7.1, 3.7),\ 0.800 > \\
< (x_4, y_4),\ w_4 > &:< (1.3, 9.3),\ 0.025 > \\
< (x_5, y_5),\ w_5 > &:< (2.5, 0.1),\ 0.100 >
\end{aligned}
$$

Here, (x_i, y_i) is randomly sampled from $X = (x, y)$. The weights satisfy $w_1 + w_2 + w_3 + w_4 + w_5 = 1$.

Particle Filter

Bayes Filter - 3 pieces

- Prediction by motion model

$$\mathbf{P}(\mathbf{X_t}|\mathbf{u_t}) = \int \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}}, \mathbf{u_t}) \cdot \mathbf{P}(\mathbf{X_{t-1}}|\mathbf{z_{t-1}}, \mathbf{u_{t-1}}) \cdot dX_{t-1}$$

- Update by observation model

$$\mathbf{P}(\mathbf{X_t}|\mathbf{z_t}, \mathbf{u_t}) = \frac{1}{\eta} \cdot \mathbf{P}(\mathbf{z_t}|\mathbf{X_t}) \cdot \mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$$

Particle Filter

- Approximate state with particles
- Prediction: sample motion model
- Update: update weight with Bayes rule

Prediction by motion model

Prediction

$$\mathbf{P}(\mathbf{X_t}|\mathbf{u_t}) = \int \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}}, \mathbf{u_t}) \cdot \mathbf{P}(\mathbf{X_{t-1}}|\mathbf{z_{t-1}}, \mathbf{u_{t-1}}) \cdot dX_{t-1}$$

- Robot state $\mathbf{P}(\mathbf{X_{t-1}}|\mathbf{z_{t-1}}, \mathbf{u_{t-1}})$ is represented by particles

$$< x_{t-1}^{[1]}, w_{t-1}^{[1]} >, < x_{t-1}^{[2]}, w_{t-1}^{[2]} >, \cdots, < x_{t-1}^{[N]}, w_{t-1}^{[N]} >$$

- The sample for state $\mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$ is obtained by sampling

$$x_t^{[i]} \leftarrow \mathbf{P}(\mathbf{X_t}|\mathbf{x_{t-1}^{[i]}}, \mathbf{u_t}) \quad i = 1, 2, \ldots, N$$

- The set of particles obtained for $\mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$ are

$$< x_t^{[1]}, w_{t-1}^{[1]} >, < x_t^{[2]}, w_{t-1}^{[2]} >, \cdots, < x_t^{[N]}, w_{t-1}^{[N]} >$$

Update by observation model

Update

$$\mathbf{P}(\mathbf{X_t}|\mathbf{z_t}, \mathbf{u_t}) = \frac{1}{\eta} \cdot \mathbf{P}(\mathbf{z_t}|\mathbf{X_t}) \cdot \mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$$

- The set of particles obtained for $\mathbf{P}(\mathbf{X_t}|\mathbf{u_t})$ are

$$< x_t^{[1]}, w_{t-1}^{[1]} >, < x_t^{[2]}, w_{t-1}^{[2]} >, \cdots, < x_t^{[N]}, w_{t-1}^{[N]} >$$

- The set of particles for $\mathbf{P}(\mathbf{X_t}|\mathbf{z_t}, \mathbf{u_t})$ are given by

$$< x_t^{[1]}, w_t^{[1]} >, < x_t^{[2]}, w_t^{[2]} >, \cdots, < x_t^{[N]}, w_t^{[N]} >$$

where new weights $w_t^{[i]}$ are computed by Bayes rule

$$\mathbf{w_t^{[i]}} = \frac{\mathbf{P}(\mathbf{z_t}|\mathbf{x_t^{[i]}}) \cdot w_{t-1}^{[i]}}{\sum_i \mathbf{P}(\mathbf{z_t}|\mathbf{x_t^{[i]}}) \cdot w_{t-1}^{[i]}} \qquad i = 1, 2, \ldots, N$$

Assignment 1 - part2 : particle filter

- **Particles**
  - Each particles is described by

$$< (x,\ y,\ \theta),\ \ weight >$$

    where (x, y, θ) is a sample from the robot state (X, Y, Θ), and *weight* is its weight.
  - In the template code, a particle is a structure given by

```
struct Particle{
    double x;        // x coordinate
    double y;        // y coordinate
    double o;        // orientation, yaw angle
double weight;  // weight
};
```

Assignment 1 - part2 : particle filter

- **Motion model** $\mathbf{P(X_{t+1}|x_t^{[i]}, u_t)}$
  - Differential drive robot

$$
\begin{aligned}
x_{t+1} &= x_t + (d + w_d)\cos(\theta_t) \\
y_{t+1} &= y_t + (d + w_d)\sin(\theta_t) \\
\theta_{t+1} &= \theta_t + (\Delta\theta + w_\theta)
\end{aligned}
$$

  - w_d is the distance noise and w_θ is orientation noise.

$$w_d \sim N(0, \delta_d^2), \quad w_\theta \sim N(0, \delta_\theta^2)$$

  - **Sampling**: Given samples of state (x_t, y_t, θ_t), and control data $(d, \Delta\theta)$, sample Gaussian distribution w_d, w_θ to obtain samples for state $(x_{t+1}, y_{t+1}, \theta_{t+1})$.
  - Weight domain remains unchanged.

Assignment 1 - part2 : particle filter

- **Observation model** $\mathbf{P(z_t|x_t^{[i]})}$
  - Observation model

$$P(z_i|x_i) = P(z_i|\hat{z}_i) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\{-\frac{(\hat{z}_i - z_i)^2}{2\sigma_z^2}\} \quad (1)$$

  - where z_i is the real measurement, $\hat{z}_i$ is the *predicted* measurement based on the location of the robot and our map.
  - Gaussian measurement noise: σ_z^2 is the variance.
  - **Bayes Update**: For each particle $< (x_i,\ y_i,\ \theta_i),\ \ weight_i >$, update its weight by

$$weight_i \leftarrow \frac{weight_i * P(z_i|x_i)}{\sum_i weight_i * P(z_i|x_i)}$$

  - State domain remains unchanged.

- **Initialization**
  - The initial guess for Bayes filter
    - **uniform distribution**
  - The initial guess for particle filter
    - **uniformly distributed samples**
  - Why uniform distribution/samples?

Assignment 1 - part2 : particle filter

- **Calculate estimate**
  - State of the particle with the largest weight
  - Weighted average of all particles' states

$$Estimate = \sum_i state_i * weight_i$$

where $state_i$ and $weight_i$ are state and weight of particle i respectively.
  - How to average an angle? What's the average of 0 and 2π?

$$\bar{\theta} = \operatorname{atan2}\left(\frac{\sum_{i=1}^{n} \sin(\theta_i)}{n}, \frac{\sum_{i=1}^{n} \cos(\theta_i)}{n}\right)$$

Template Code — Particle filter

```
/** main process function for localization */
void PFLocalization::process() {
    ... ...

        if ( (robot_odom_[0] != 0.0 || robot_odom_[1] != 0.0) ) {
        ROS_INFO("Robot␣odom:␣distance␣=␣%f␣and␣orientation␣=␣%f", robot_odom_[0
            motion( robot_odom_[0], robot_odom_[1] );

            //! add update code block here
            //! Notice:
            //!     1. what is the likelihood
            //!     2. how to update previous belief
            //!     3. how to do normalization of updated belief







            calc_estimate();
            if ( step_count_ == 5 ) {
                resampling();
                step_count_ = 0;
            }
            step_count_ ++;

        }

    ... ...
}
```

Template Code — Initialization

```
/** particle filter init */
void PFLocalization::init_particles() {
    particles_.resize( n_particles_ );
    //! add particles initialistion code here
    //! Notice:
    //!    1. x, y range
    //!    2. orientation range
    //!    3. weight







}
```

Template Code — Observation

```
/** sense function */
double PFLocalization::sense(double sigma, double x, double y, double theta) {
double error = 1.0;
... ...

    for ( int i = 0; i < (int)scan_data_.size(); ++ i ) {
    ... ...

        //! raydist is the scan data given particle's pose
        //! scan_data_[i] is the actually sensor data
        raydist = sqrt((obspt.x-stpt.x)/100.0*(obspt.x-stpt.x)/100.0+
                       (obspt.y-stpt.y)/100.0*(obspt.y-stpt.y)/100.0);
        //! compute the likelihood of each beam
        //! put your code here





    }
    //! return the final error
    //! Notice:
    //!     1. how to combine the likelihood of each beam together

    return error;
}
```

Template Code — Motion

```
/** motion function */
void PFLocalization::motion(double dist, double ori) {
    // check dist negativity
    if ( dist < 0 ) {
        cout << "ERROR: distance < 0\n";
        exit(-1);
    }

    for ( int i = 0; i < (int)particles_.size(); ++i ) {
        //! add noise to motion
        //! put your code here
        //! Notice:
        //!     1. sampling distribution
        //!     2. range of updated orientation






    }
}
```

Template Code — Calculate estimate

```
/** calculate estimate robot pose */
void PFLocalization::calc_estimate() {
    // reset estimated pose
    esti_pos_[0] = 0.0;
    esti_pos_[1] = 0.0;
    esti_pos_[2] = 0.0;

    //! compute estimate pose
    //! put your code here
    //! Notice:
    //!      1. average, how?
    //!      2. again, range of x, y, orientation




    //! uncomment the following line if you want to see the estimated pose
    //     cout << "Estimated pose: " << esti_pos_[0] << ", " << esti_pos_[1] <<
}
```

Template Code — Variables

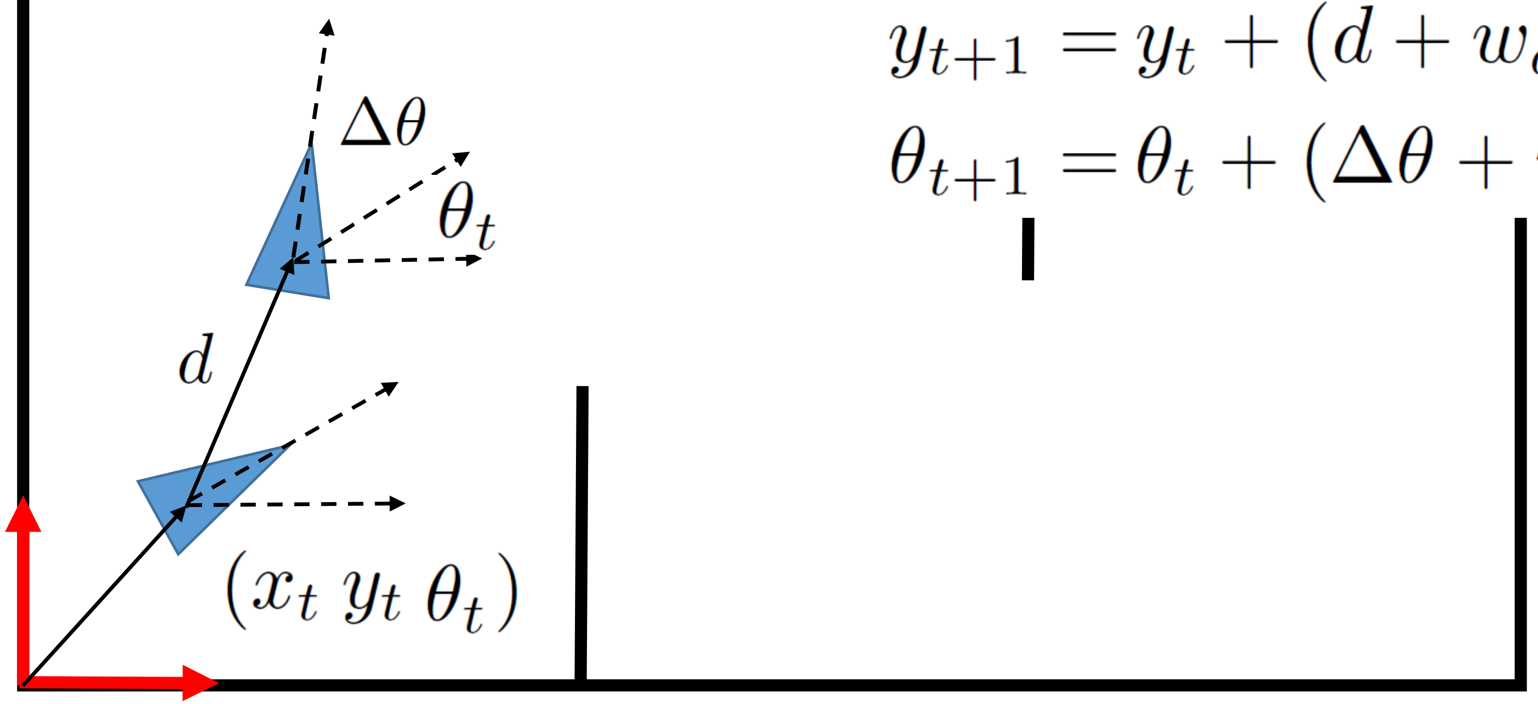
```
vector<double> scan_data_;  // laser scan data array, beam number is 5
double dist_noise_;         // sigma value of gaussian distribution of distance 
double ori_noise_;          // sigma value of gaussian distribution of orientatio

// particles
int n_particles_;           // number of particles, set to be 1000
vector<Particle> particles_;// array of particles

double esti_pos_[3];        // estimated pose of the robot using particles
```

How to Compile and Run — Particle Filter

- Download zip package, extract the file into `/home/vmuser/catkin_ws/src/`
- Execute the following command sequence to compile
  - `cd ~/catkin_ws`
  - `catkin_make`
  - `source devel/setup.bash`
- Run the code using launch file
  - `source devel/setup.bash`
  - `chmod +x src/pf_localization/src/scripts/map_frame.py`
  - `roslaunch pf_localization pf_localization.launch`

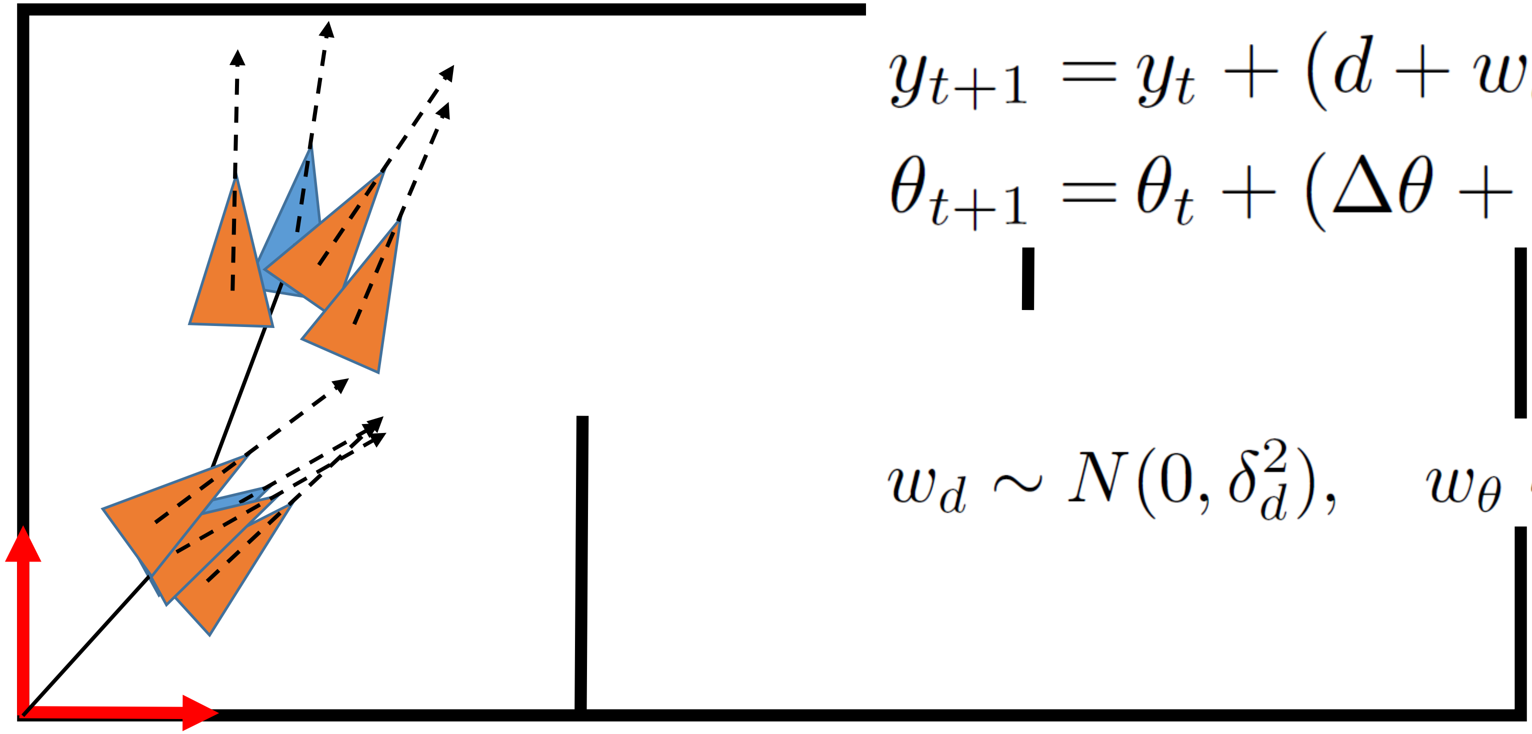
Motion Model

$$x_{t+1} = x_t + (d + w_d)\cos(\theta_t)$$
$$y_{t+1} = y_t + (d + w_d)\sin(\theta_t)$$
$$\theta_{t+1} = \theta_t + (\Delta\theta + w_\theta)$$



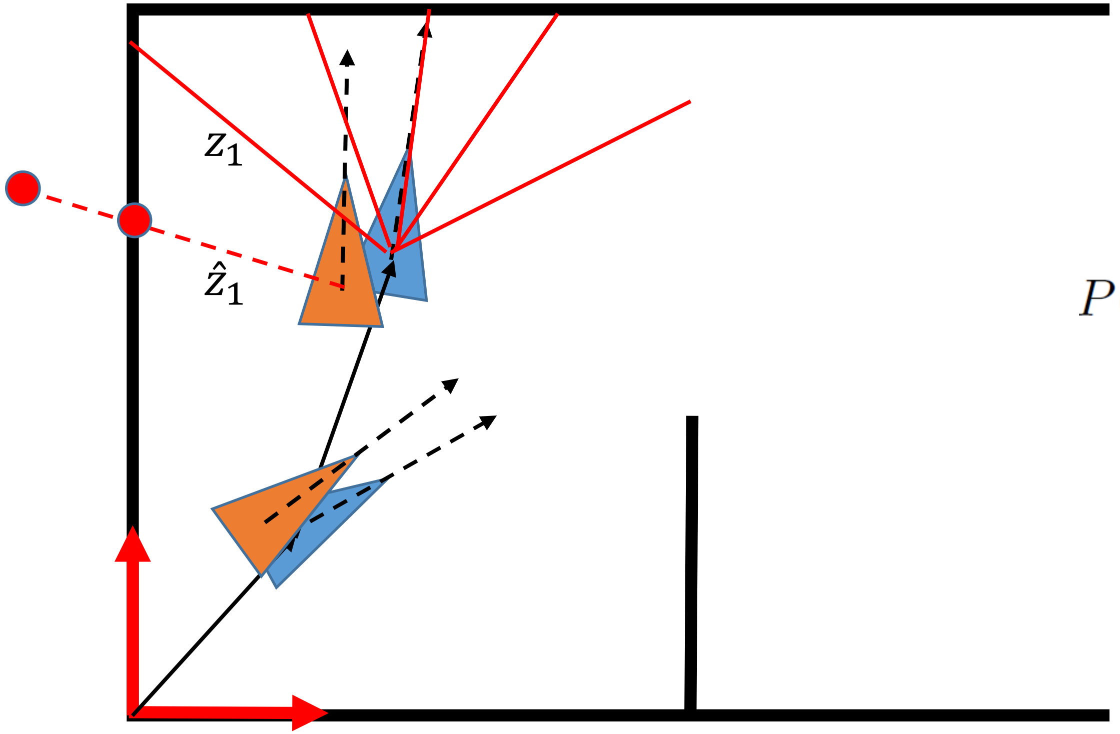
Motion Model in Particle Filter

$$x_{t+1} = x_t + (d + w_d)\cos(\theta_t)$$
$$y_{t+1} = y_t + (d + w_d)\sin(\theta_t)$$
$$\theta_{t+1} = \theta_t + (\Delta\theta + w_\theta)$$



$$w_d \sim N(0, \delta_d^2), \quad w_\theta \sim N(0, \delta_\theta^2)$$

Observation Model in Particle Filter



$$z = \{z_1, z_2, \ldots z_M\}$$

$$\hat{z_i} = \{\hat{z}_1, \hat{z}_2, \ldots \hat{z}_M\}_i$$

$$P(z_i|x_i) = P(z_i|\hat{z}_i) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\{-\frac{(\hat{z}_i - z_i)^2}{2\sigma_z^2}\}$$

$$weight_i \leftarrow \frac{weight_i * P(z_i|x_i)}{\sum_i weight_i * P(z_i|x_i)}$$