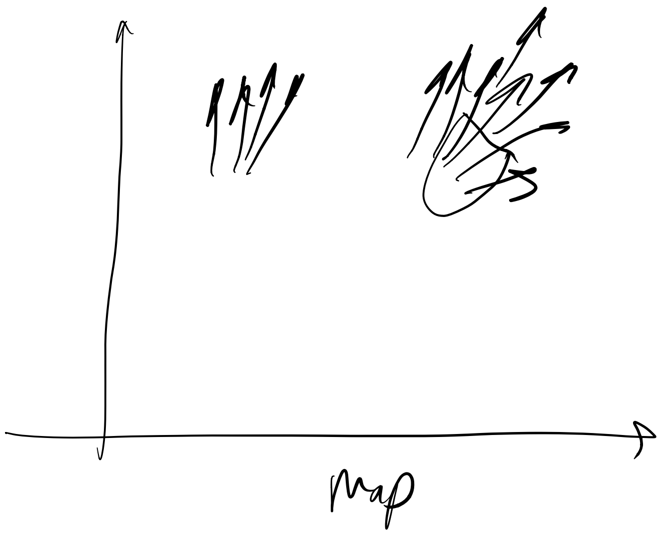
Particle Filter

State is x_t , sensors are z_t, motor cmd is u_t

$$p(x_t \mid u_1 \ldots u_t, z_1 \ldots z_t)$$

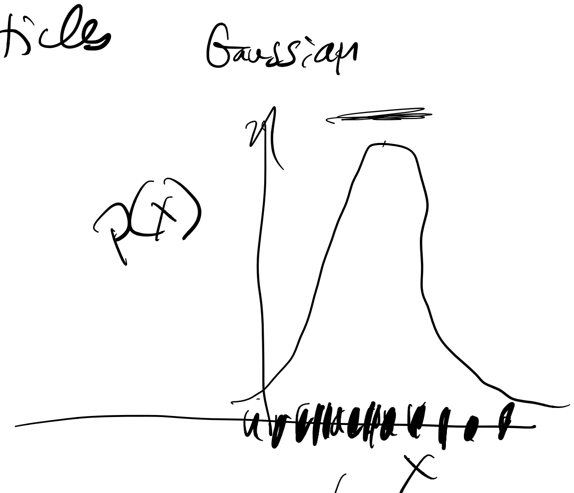
approximate using samples

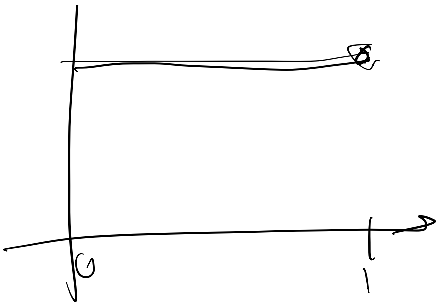


Step 1: Create initial particles

For i 1..m $\quad x_i^{(0)} \sim p(x_0)$

$\sim$ means sample





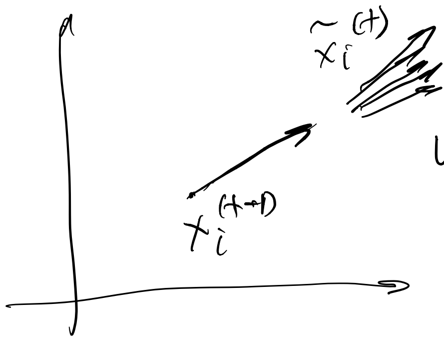
rand() ✓

randn()

Step 2: apply motion model

For i from 1 to m:

$$\tilde{X}_i^{(t)} \sim p(X_t \mid X_{t-1} = X_i^{(t-1)}, u_t)$$

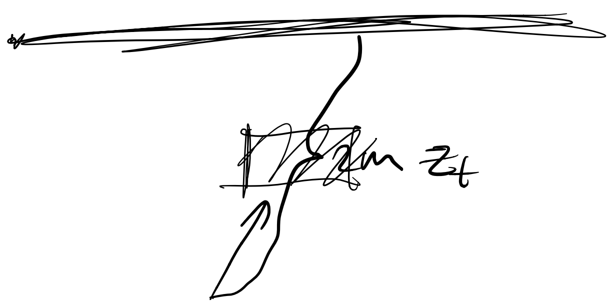


u_t = drive 1m forward

Step 3 apply sensor model laser scan

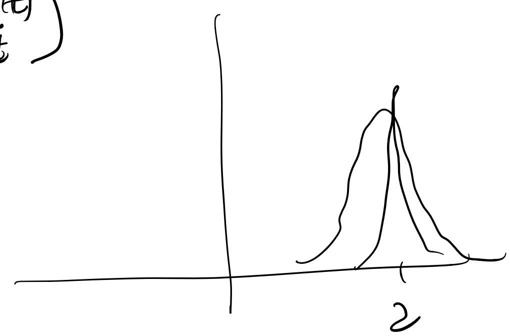
$$w_i^{(t)} = \frac{p(z_t \mid X_t = \tilde{X}_i^{(t)})}{\sum_{j=1}^{M} p(z_t \mid X_t = \tilde{X}_j^{(t)})}$$

weights sum to 1



$\tilde{x}_i^{(t)}$

$p(z_t \mid x_t = \tilde{x}_i^{(t)})$

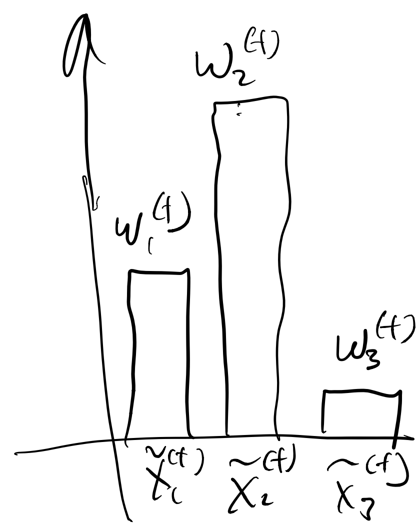


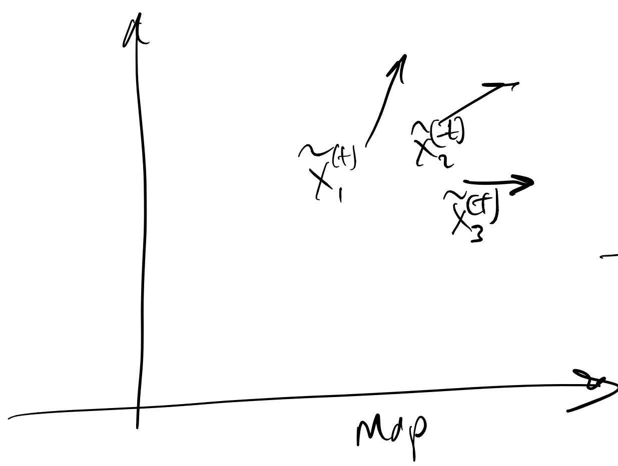
Step 4 resampling

For i from 1 to m

$$x_i^{(t)} \sim p(x_i^{(t)} = \tilde{x}_j^{(t)}) = w_j^{(t)}$$

Step 5 go to step 2





Particle Filter and Coordinate Frames

Motion Estimates

(1) Odometry (/odom)

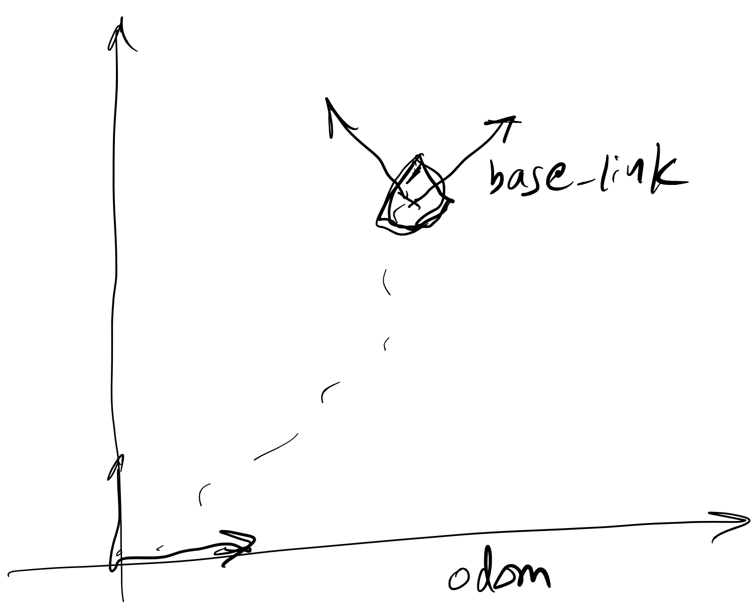
derived from sensors wheel encoders

pros: very fast, always available
cons: subject to drift

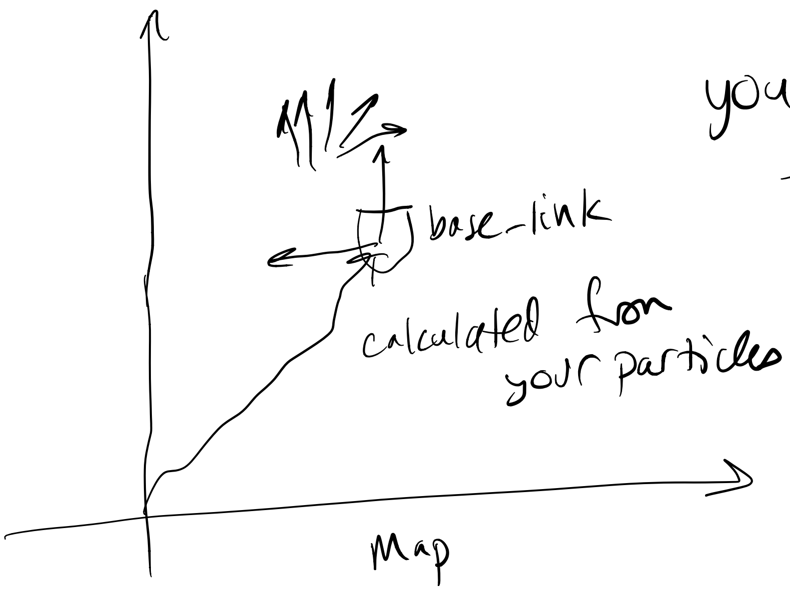
(2) Landmarks (/scan)

Estimating absolute position of robot relation to fixed landmarks

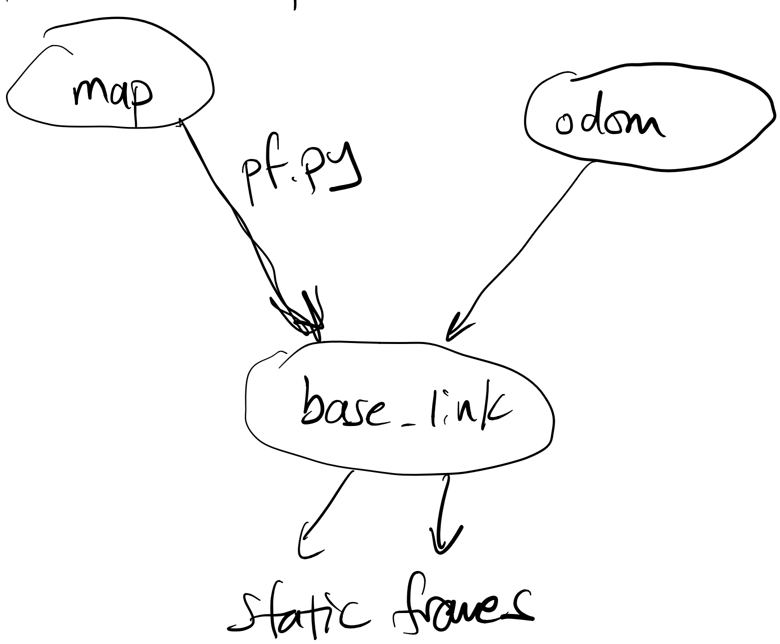
cons: slow, not always available
pros: correct motion estimation errors

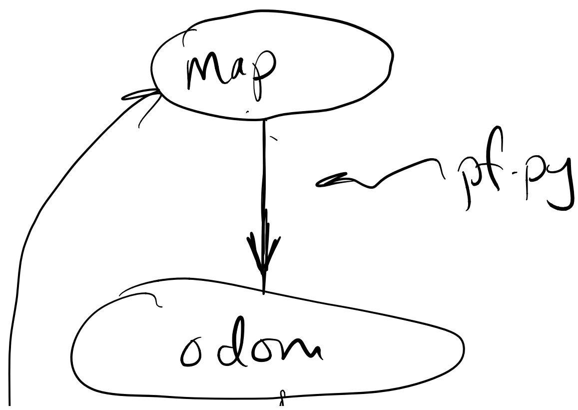


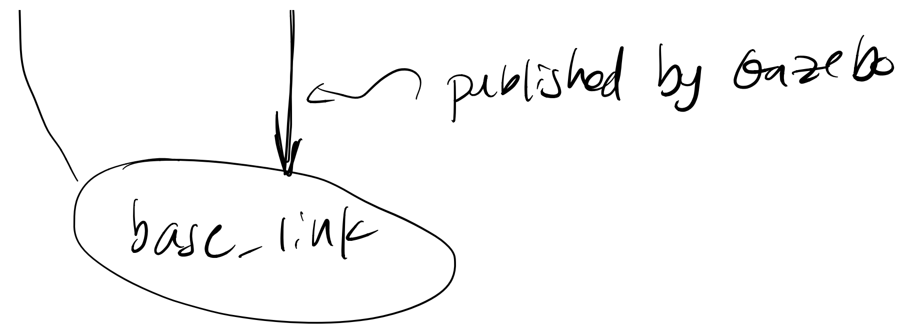
already implemented!

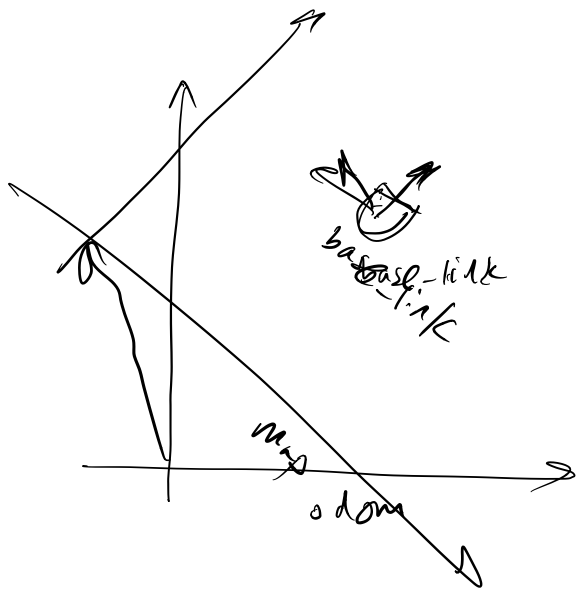


you will implement
this






published by Gazebo
base_link


base_link
map
odom