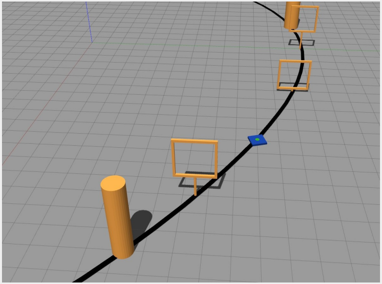
Autonomous Mobile Robotics

Robot Localization

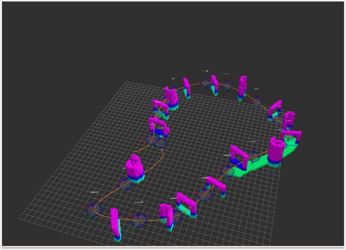
Geesara Kulathunga

March 13, 2023



Robot Localization

- A Taxonomy of Localization Problems
- Markov localization
  - Environment Sensing
  - Motion in the Environment
  - Localization in the Environment
- EKF localization with known correspondence
- Particle filter localization with known correspondence



A Taxonomy of Localization Problems

- Local Versus Global

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)
- Static Versus Dynamic Environments

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)
- Static Versus Dynamic Environments
  - In static environment, robot's pose is only the variable quantity

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)
- Static Versus Dynamic Environments
  - In static environment, robot's pose is only the variable quantity
  - Dynamics environment, whole configuration can be changed over the time

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)
- Static Versus Dynamic Environments
  - In static environment, robot's pose is only the variable quantity
  - Dynamics environment, whole configuration can be changed over the time
- Passive Versus Active Approaches

A Taxonomy of Localization Problems

- Local Versus Global
  - Position tracking where initial position is known (local tracking)
  - Robot position is unknown, initially has to assume that pose of robot is uniform in the most of the cases (global)
  - Kidnapped robot problem; anytime robot can be moved to different location without prior knowledge (global)
- Static Versus Dynamic Environments
  - In static environment, robot’s pose is only the variable quantity
  - Dynamics environment, whole configuration can be changed over the time
- Passive Versus Active Approaches
  - In passive, robot is controlled through some other means, robot motion is not aiming at facilitating localization

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$)**:**

for all x_t do

$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$

$bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$

endfor

return $bel(x_t)$

- **Markov localization** is derived from the algorithm **Bayes filter**

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$)**:**

for all x_t *do*

$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$

$bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$

endfor

return $bel(x_t)$

- **Markov localization** is derived from the algorithm **Bayes filter**
- However, it requires information about the **map** to **estimate the measurement model** $p(z_t|x_t, m)$

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):

for all x_t *do*

$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\; bel(x_{t-1})\; dx$

$bel(x_t) = \eta\; p(z_t \mid x_t, m)\; \overline{bel}(x_t)$

endfor

return $bel(x_t)$

- **Markov localization** is derived from the algorithm **Bayes filter**
- However, it requires information about the **map** to **estimate the measurement model** $p(z_t|x_t, m)$
- Markov localization addresses the **global localization**, the position tracking, and the kidnapped robot problem in **static environment**

Markov localization

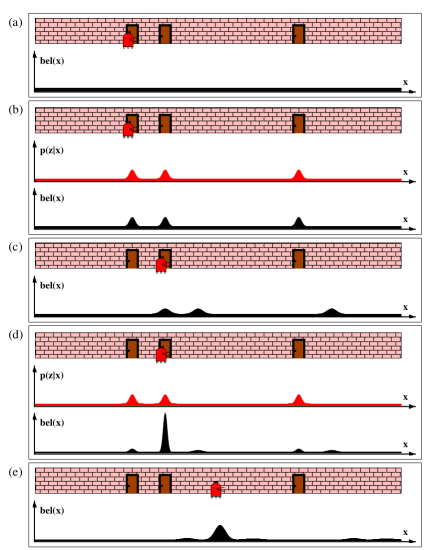
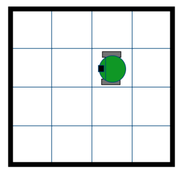


Illustration of the Markov localization algorithm, Thrun, Sebastian. "Probabilistic robotics." Communications of the ACM 45.3 (2002): 52-57.

Grid-based localization

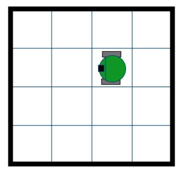


.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

robot initial belief

- The map is discretized into 16 cells, each of which has an area of $1m^2$

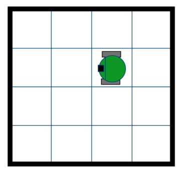
Grid-based localization



.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

robot initial belief

- The map is discretized into 16 cells, each of which has an area of $1m^2$
- Consider the initial belief of the robot position is given

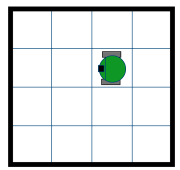


.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

robot initial belief

- The map is discretized into 16 cells, each of which has an area of $1m^2$
- Consider the initial belief of the robot position is given
- If **the control command** to the robot is given by $\delta x, \delta y$ = -1.0 cells, 0.0 cells, what is the probability that robot be in the position (2,3)

Grid-based localization



.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

robot initial belief

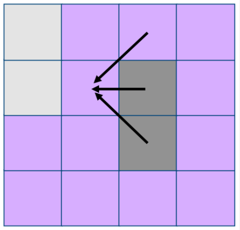
- The map is discretized into 16 cells, each of which has an area of 1m^2
- Consider the initial belief of the robot position is given
- If **the control command** to the robot is given by $\delta x, \delta y$ = -1.0 cells, 0.0 cells, what is the probability that robot be in the position (2,3)
- The following outcomes are possible when **the control command** is being applied

.00	.00	.00
.00	.00	1.0
.00	.00	.00

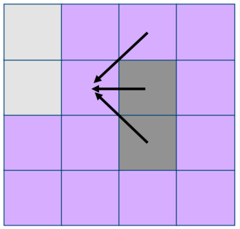
$\xrightarrow{(\Delta x, \Delta y)}$

.00	.20	.00
.00	.50	.10
.00	.20	.00

- How many possible ways to get to (2,3)?



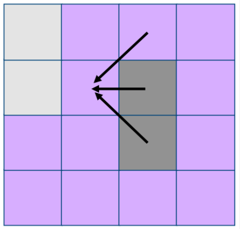
- How many possible ways to get to (2,3)?



  - Prediction step

$$p(x_k|z_{1:k-1}, u_{1:k-1}) = \Sigma_{x_{k-1}\in X}\ p(x_k|x_{k-1}, u_{k-1})p(x_{k-1}|z_{1:k-1}, u_{0:k-1}) \quad (1)$$

- How many possible ways to get to (2,3)?



  - Prediction step

$$p(x_k|z_{1:k-1}, u_{1:k-1}) = \Sigma_{x_{k-1} \in X}\ p(x_k|x_{k-1}, u_{k-1}) p(x_{k-1}|z_{1:k-1}, u_{0:k-1}) \tag{1}$$

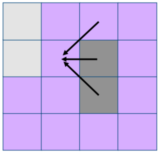
  - Correction step

$$p(x_k|z_{1:k}, u_{0:k-1}) = \frac{p(z_k|x_k) p(x_k|z_{1:k-1}, u_{0:k-1})}{p(z_k|z_{1:k-1}, u_{0:k-1})} \tag{2}$$

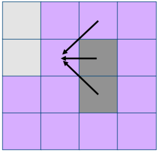
, where

$$p(z_k|z_{1:k-1}, u_{0:k-1}) = \Sigma_{x_k \in X}\ p(z_k|x_k) p(x_k|z_{1:k-1}, u_{0:k-1})$$

- How many possible ways to get to (2,3)?



- How many possible ways to get to (2,3)?

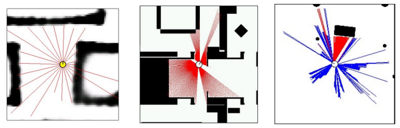


  - Prediction step

$$
\begin{aligned}
p(x_{i,t}|u_t) &= \Sigma_{j=1}^{n} p\left(x_{i,t}|x_{j,t-1}, u_t\right) p\left(x_{j,t-1}\right) \\
&= p\left(x_{i,t} = (2,3)|x_{j,t-1} = (3,3), u_t = (-1,0)\right) p\left(x_{j,t-1} = (3,3)\right) \\
&+ p\left(x_{i,t} = (2,3)|x_{j,t-1} = (2,3), u_t = (-1,0)\right) p\left(x_{j,t-1} = (2,3)\right) \\
&+ p\left(x_{i,t} = (2,3)|x_{j,t-1} = (3,2), u_t = (-1,0)\right) p\left(x_{j,t-1} = (3,2)\right) \\
&+ p\left(x_{i,t} = (2,3)|x_{j,t-1} = (3,4), u_t = (-1,0)\right) p\left(x_{j,t-1} = (3,4)\right) \\
&= 0.5 \cdot 0.18 + 0.1 \cdot 0.05 + 0.18 \cdot 0.2 + 0.05 \cdot 0.2 \\
&= 0.141
\end{aligned}
$$

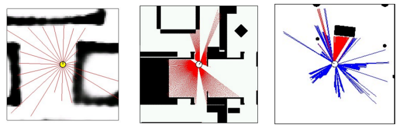
Correction step

- How can we estimate the $p(z_t|x_{i,t})$?



Correction step

- How can we estimate the $p(z_t|x_{i,t})$?

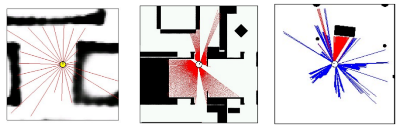


- If each sensor reading consists of N measurements, i.e., $z = \{z_1, ..., z_n\}$, assuming each such **measurement is independent** given the robot pose,

$$p(z_t|x_{i,t}) = \Pi_{j=1}^{n} p(z_j|x_{i,t}, m)$$

Correction step

- How can we estimate the $p(z_t|x_{i,t})$?



- If each sensor reading consists of N measurements, i.e., $z = \{z_1, ..., z_n\}$, assuming each such **measurement is independent** given the robot pose,

$$p(z_t|x_{i,t}) = \Pi_{j=1}^{n} p(z_j|x_{i,t}, m)$$

- Such measurements can be caused by known obstacles, dynamic obstacles, reflections, etc.

- Correction step

$$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t})p(x_{i,t}|u_t)}{p(z_t)}$$

- Correction step

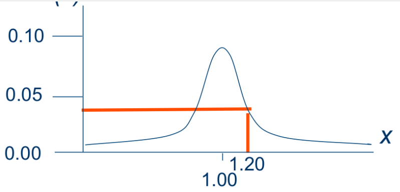
$$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t})p(x_{i,t}|u_t)}{p(z_t)}$$

- $p(z_t|x_{i,t})$ getting measurement z_t from state $x_{i,t}$

- Correction step

$$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t})p(x_{i,t}|u_t)}{p(z_t)}$$

- $p(z_t|x_{i,t})$ getting measurement z_t from state $x_{i,t}$
- Let z_t be 1.2m and range sensor has the following distribution



- $p(z_t)$ probability of the sensor measurement z_t. Calculated so that the sum over all states $x_{i,t}$ equals 1

$$1 = \Sigma_{i=1}^{n} p(x_{i,t}|z_t = 1.2)$$

$$1 = \frac{\Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t}) p(x_{i,t}|u_{i,t})}{p(z_t = 1.2)}$$

$$p(z_t = 1.2) = \Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t}) p(x_{i,t}|u_{i,t})$$

- $p(z_t)$ probability of the sensor measurement z_t. Calculated so that the sum over all states $x_{i,t}$ equals 1

$$1 = \Sigma_{i=1}^{n} p(x_{i,t}|z_t = 1.2)$$

$$1 = \frac{\Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t})p(x_{i,t}|u_{i,t})}{p(z_t = 1.2)}$$

$$p(z_t = 1.2) = \Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t})p(x_{i,t}|u_{i,t})$$

- $$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t})p(x_{i,t}|u_t)}{p(z_t)}$$

$$= \frac{p\Big(z_t = 1.2|x_{i,t} = (2,3)\Big)p(x_{i,t}|u_t)}{p(z_t = 1.2)} = \frac{0.04 \cdot 0.141}{p(z_t = 1.2)}$$

- $p(z_t)$ probability of the sensor measurement z_t. Calculated so that the sum over all states $x_{i,t}$ equals 1

$$1 = \Sigma_{i=1}^{n} p(x_{i,t}|z_t = 1.2)$$
$$1 = \frac{\Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t}) p(x_{i,t}|u_{i,t})}{p(z_t = 1.2)}$$
$$p(z_t = 1.2) = \Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t}) p(x_{i,t}|u_{i,t})$$

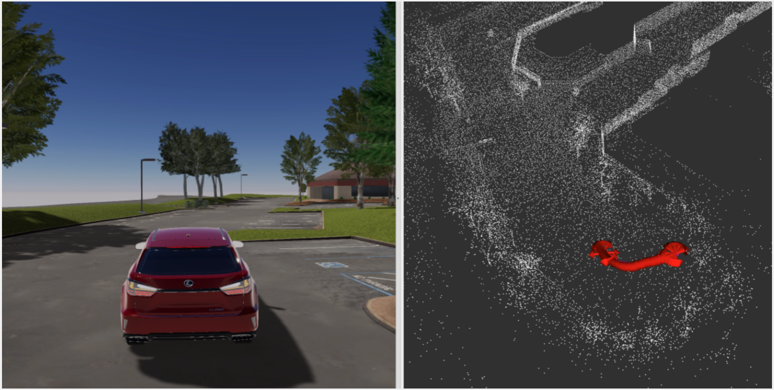
- $$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t}) p(x_{i,t}|u_t)}{p(z_t)}$$

$$= \frac{p\Big(z_t = 1.2|x_{i,t} = (2,3)\Big) p(x_{i,t}|u_t)}{p(z_t = 1.2)} = \frac{0.04 \cdot 0.141}{p(z_t = 1.2)}$$

- Can we calculate this?

$$p(z_t = 1.2) = \Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t}) p(x_{i,t}|u_{i,t})$$

EKF Localization



https://autowarefoundation.gitlab.io/autoware.auto/AutowareAuto/ekf-localization-howto.html

- Specific case of **Markov localization**

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:
  - robot pose at time $k-1$ with μ_{t-1}, Σ_{t-1}

EKF Localization

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:
    - robot pose at time $k-1$ with μ_{t-1}, Σ_{t-1}
    - Control input u_{t-1}

- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:
  - robot pose at time $k-1$ with μ_{t-1}, Σ_{t-1}
  - Control input u_{t-1}
  - Map and a set of features $z_t = \{z_t^1, z_t^2, ...\}$ measured at time k and those are corresponded to variables $c_t = \{c_t^1, c_t^2, ...\}$

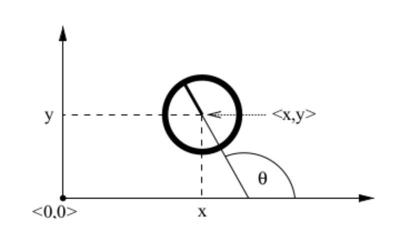
- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:
  - robot pose at time $k-1$ with μ_{t-1}, Σ_{t-1}
  - Control input u_{t-1}
  - Map and a set of features $z_t = \{z_t^1, z_t^2, ...\}$ measured at time k and those are corresponded to variables $c_t = \{c_t^1, c_t^2, ...\}$
- Output is a new, revised estimation: μ_t and Σ_t

KF	EKF
	$\boxed{\Phi_k = \left.\frac{\partial f(\mathbf{x}_k, t)}{\partial \mathbf{x}}\right\vert_{\mathbf{x}_k}}$
$\hat{\mathbf{x}}_k^- = \Phi_k \mathbf{x}_k$	$\boxed{\hat{\mathbf{x}}_k^- = f(\mathbf{x}_k, \mathbf{t})}$
$\mathbf{P}_k^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k$	$\mathbf{P}_k^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k$
	$\boxed{\mathbf{H} = \left.\frac{\partial h(\hat{\mathbf{x}}_k^-)}{\partial \hat{\mathbf{x}}}\right\vert_{\hat{\mathbf{x}}_k^-}}$
$\mathbf{y} = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-$	$\mathbf{y} = \mathbf{z}_k - \boxed{h(\hat{x}_k^-)}$
$\mathbf{K_k} = \mathbf{P_k^-} \mathbf{H_k}^T (\mathbf{H_k} \mathbf{P_k^-} \mathbf{H_k}^T + \mathbf{R}_k)^{-1}$	$\mathbf{K_k} = \mathbf{P_k^-} \mathbf{H_k}^T (\mathbf{H_k} \mathbf{P_k^-} \mathbf{H_k}^T + \mathbf{R}_k)^{-1}$
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K_k} \mathbf{y}$	$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K_k} \mathbf{y}$
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K_k} \mathbf{H_k}) \mathbf{P_k^-}$	$\mathbf{P}_k = (\mathbf{I} - \mathbf{K_k} \mathbf{H_k}) \mathbf{P_k^-}$

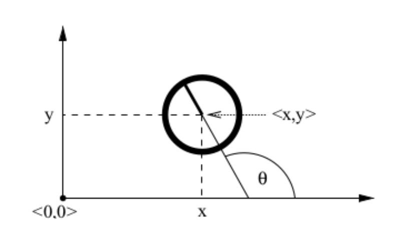
- Motion models comprise the state transition probability $p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$ (prediction step of the Bayes filter)

Probabilistic Motion Model

- Motion models comprise the state transition probability $p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1})$ (prediction step of the Bayes filter)
- Robot pose $[x \quad y \quad \theta]^\top$, shown in a global coordinate system



- Motion models comprise the state transition probability $p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$ (prediction step of the Bayes filter)
- Robot pose $[x \quad y \quad \theta]^\top$, shown in a global coordinate system



- Probabilistic kinematic model, or motion model (velocity motion model or odometry motion model), describes the posterior distribution over kinematic states that a robot assumes when executing the motion command $\mathbf{u}_t$ at $\mathbf{x}_t$

- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$

- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$
- Differential drives, Ackerman drives, and synchro-drives can be controlled in this way

Velocity Motion Model (Noise-free)

- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$
- Differential drives, Ackerman drives, and synchro-drives can be controlled in this way
- Let $\mathbf{x}_{t-1} = [x_{t-1} \quad y_{t-1} \quad \theta_{t-1}]^\top, \mathbf{x}_t = [x_t \quad y_t \quad \theta_t]^\top$ be pose and time $t-1$ and successor pose, respectively, after applying applying control u_{t-1} for δt duration

Velocity Motion Model (Noise-free)

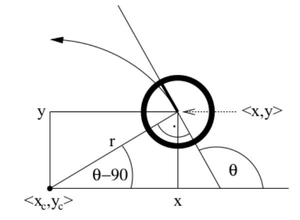
- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$
- Differential drives, Ackerman drives, and synchro-drives can be controlled in this way
- Let $\mathbf{x}_{t-1} = [x_{t-1} \quad y_{t-1} \quad \theta_{t-1}]^\top, \mathbf{x}_t = [x_t \quad y_t \quad \theta_t]^\top$ be pose and time $t-1$ and successor pose, respectively, after applying applying control u_{t-1} for δt duration
- If **both velocities are kept at a fixed value** for the entire time interval, [t-1, t], **robot moves on a circle with radius** $r = |\frac{v}{u}|$

Velocity Motion Model (Noise-free)

- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$
- Differential drives, Ackerman drives, and synchro-drives can be controlled in this way
- Let $\mathbf{x}_{t-1} = [x_{t-1} \quad y_{t-1} \quad \theta_{t-1}]^\top, \mathbf{x}_t = [x_t \quad y_t \quad \theta_t]^\top$ be pose and time $t-1$ and successor pose, respectively, after applying applying control u_{t-1} for δt duration
- If **both velocities are kept at a fixed value** for the entire time interval, [t-1, t], **robot moves on a circle with radius** $r = |\frac{v}{u}|$
- For linear motion, r becomes infinite

- A robot can be control through linear and angular velocities $\mathbf{u}_t = [v_t \quad \omega_t]^\top$
- Differential drives, Ackerman drives, and synchro-drives can be controlled in this way
- Let $\mathbf{x}_{t-1} = [x_{t-1} \quad y_{t-1} \quad \theta_{t-1}]^\top, \mathbf{x}_t = [x_t \quad y_t \quad \theta_t]^\top$ be pose and time $t-1$ and successor pose, respectively, after applying applying control u_{t-1} for δt duration
- If **both velocities are kept at a fixed value** for the entire time interval, [t-1, t], **robot moves on a circle with radius** $r = |\frac{v}{u}|$
- For linear motion, r becomes infinite
- After δt units of time, the noise-free robot has progressed $v\delta t$ along the circle, which caused its heading direction to turn by $\omega\delta t$

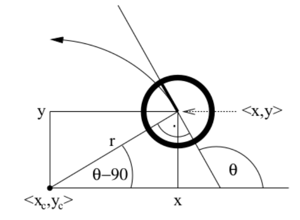
Velocity Motion Model (Noise-free)



- The center of the circle is at, assuming v and ω, denoted linear and angular velocities relative to $< x, y >$,

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x - \frac{v}{\omega} sin(\theta) \\ y + \frac{v}{\omega} cos(\theta) \end{bmatrix}$$

Velocity Motion Model (Noise-free)



- The center of the circle is at, assuming v and ω, denoted linear and angular velocities relative to $< x, y >$,

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x - \frac{v}{\omega} sin(\theta) \\ y + \frac{v}{\omega} cos(\theta) \end{bmatrix}$$

- After δt time, ideal robot will be at $\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} & y_{t+1} & \theta_{t+1} \end{bmatrix}$

$$= \begin{bmatrix} x_c + \frac{v}{\omega} sin(\theta_t + \omega\delta t) \\ y_c - \frac{v}{\omega} cos(\theta_t + \omega\delta t) \\ \theta_t + \omega\delta t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega} sin(\theta_t) + \frac{v}{\omega} sin(\theta_t + \omega\delta t) \\ \frac{v}{\omega} cos(\theta_t) - \frac{v}{\omega} cos(\theta_t + \omega\delta t) \\ \omega\delta t \end{bmatrix}$$

- In reality, robot motion is subject to noise, to model such noise, which is formed a zero-centered random variable with finite variance, we can do the following approach

$$\begin{pmatrix} \hat{v}_t \\ \hat{\omega}_t \end{pmatrix} = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v_t^2 + \alpha_2 \omega_t^2} \\ \varepsilon_{\alpha_3 v_t^2 + \alpha_4 \omega_t^2} \end{pmatrix} = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + N(0, M_t) \qquad (3)$$

where $M_t = \begin{pmatrix} \varepsilon_{\alpha_1 v_t^2 + \alpha_2 \omega_t^2} & 0 \\ 0 & \varepsilon_{\alpha_3 v_t^2 + \alpha_4 \omega_t^2} \end{pmatrix}$

- Real motion model

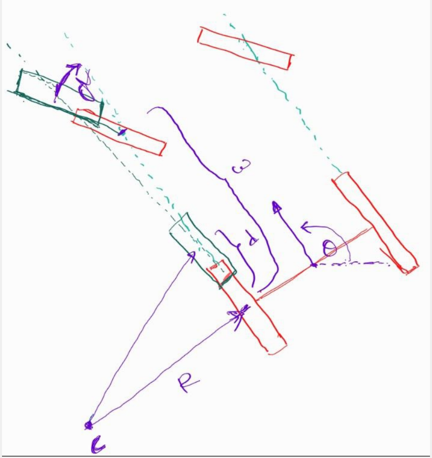
$$\underbrace{\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}_t}{\hat{\omega}_t} sin(\theta) + \frac{\hat{v}_t}{\hat{\omega}_t} sin(\theta + \hat{\omega}_t \delta t) \\ \frac{\hat{v}_t}{\hat{\omega}_t} cos(\theta) - \frac{\hat{v}_t}{\hat{\omega}_t} cos(\theta + \hat{\omega}_t \delta t) \\ \hat{\omega}_t \delta t + \hat{\gamma} \delta t \end{pmatrix}}_{f(u_t, \mathbf{x}_t)} \quad (4)$$

, where $\hat{\gamma} \sim \varepsilon_{\alpha_5 v_t^2 + \alpha_6 \omega_t^2}$

- Approximated motion model, i.e., replacing true motion $\hat{v}_t$ and $\hat{\omega}_t$ by executed control (v_t, ω_t)

$$\underbrace{\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \delta t) \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \delta t) \\ \omega_t \delta t \end{pmatrix}}_{f(u_t, \mathbf{x}_t)} + N(0, Q_t) \quad (5)$$

Consider the following robot model



The front tire is pointing in direction α relative to the wheelbase. Over a short time period the car moves forward and the rear wheel ends up further ahead and slightly turned inward, as depicted with the green dotted tire. Over such a short time frame we can approximate this as a turn around a C .

Prove that

- $\beta = \frac{d}{w} tan(\alpha)$
- $R = \frac{d}{\beta}$, where $d = \delta t v$, if robot robot move with v forward velocity for δt time
- The position of the C is given by
  $Cx = x - Rsin(\theta), \quad Cy = y + Rcos(\theta)$
- If robot move forward for time δt, the new pose is given by
  $$\begin{bmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{bmatrix} = \begin{bmatrix} x - Rsin(\theta) + Rsin(\theta + \beta) \\ y + Rcos(\theta) - Rcos(\theta + \beta) \\ \theta + \beta \end{bmatrix}$$

Remark: $sin(-\theta) = -sin(\theta), cos(-\theta) = cos(\theta)$

How can we define the state variables, control inputs, and the system model?

How can we define the measurement model?

EKF Localization with known correspondence

State variables, control inputs, and the system model

- state variables: $\mathbf{x} = [x\ , y\ , \theta]$
- control inputs: $\mathbf{u} = [v,\ \alpha]$
- $\bar{\mathbf{x}} = \mathbf{x} + f(\mathbf{x}, \mathbf{u}) + N(0, Q)$, where Q is the white noise

Measurement model

If the installed sensor gives a noisy bearing and range to multiple known landmarks, bearing and range can be estimated in the following way, e.g., let p_x, p_y be a landmark location,

$$r = \sqrt{(p_x - x)^2 + (p_y - y)^2}, \quad \phi = arctan(\frac{p_y - y}{p_x - x}) - \theta \qquad (6)$$
$$\mathbf{z} = h(\mathbf{x}, P) + N(0, R),$$

R is the white noise

- Approximated motion model, i.e., replacing true motion $\hat{v}_t$ and $\hat{\omega}_t$ by executed control (v_t, ω_t)

$$\underbrace{\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \delta t) \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \delta t) \\ \omega_t \delta t \end{pmatrix}}_{f(u_t, \mathbf{x}_t)} + N(0, Q_t) \tag{7}$$

- Let μ_{t-1}, Σ_{t-1} be the previous optimal state ($\hat{\mathbf{x}}_{t-1}^{-}$) as a Gaussian distribution

- Taylor expansion is used to linearize the function $f(u_t, \mathbf{x}_{t-1})$

$$f(u_t, \mathbf{x}_{t-1}) \approx f(u_t, \hat{\mathbf{x}}^-_{t-1}) + \Phi_t(\mathbf{x}_{t-1} - \hat{\mathbf{x}}^-_{t-1}) \tag{8}$$

$$\Phi_t = \frac{\partial f(u_t, \hat{\mathbf{x}}^-_{t-1})}{\partial \hat{\mathbf{x}}^-_{t-1}} = \begin{pmatrix} \frac{\partial x'}{\partial \hat{\mathbf{x}}^-_{t-1,x}} & \frac{\partial x'}{\partial \hat{\mathbf{x}}^-_{t-1,y}} & \frac{\partial x'}{\partial \hat{\mathbf{x}}^-_{t-1,\theta}} \\ \frac{\partial y'}{\partial \hat{\mathbf{x}}^-_{t-1,x}} & \frac{\partial y'}{\partial \hat{\mathbf{x}}^-_{t-1,y}} & \frac{\partial y'}{\partial \hat{\mathbf{x}}^-_{t-1,\theta}} \\ \frac{\partial \theta'}{\partial \hat{\mathbf{x}}^-_{t-1,x}} & \frac{\partial \theta'}{\partial \hat{\mathbf{x}}^-_{t-1,y}} & \frac{\partial \theta'}{\partial \hat{\mathbf{x}}^-_{t-1,\theta}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & \frac{v_t}{\omega_t}(-cos(\hat{\mathbf{x}}^-_{t-1,\theta}) + cos(\hat{\mathbf{x}}^-_{t-1,\theta} + \omega_t \delta t)) \\ 0 & 1 & \frac{v_t}{\omega_t}(-sin(\hat{\mathbf{x}}^-_{t-1,\theta}) + sin(\hat{\mathbf{x}}^-_{t-1,\theta} + \omega_t \delta t)) \\ 0 & 0 & 1 \end{pmatrix}$$

where $\hat{\mathbf{x}}^-_{t-1} = \hat{\mathbf{x}}^-_{t-1,x}, \hat{\mathbf{x}}^-_{t-1,y}, \hat{\mathbf{x}}^-_{t-1,\theta}$ denotes the mean estimate factored into its individual three values

EKF Localization with known correspondence

- Motion model with respect to control

$$V_t = \frac{\partial f(u_t, \hat{\mathbf{x}}_{t-1}^{-})}{\partial u_t} = \begin{pmatrix} \frac{\partial x'}{\partial v_t} & \frac{\partial x'}{\partial \omega_t} \\ \frac{\partial y'}{\partial v_t} & \frac{\partial y'}{\partial \omega_t} \\ \frac{\partial \theta'}{\partial v_t} & \frac{\partial \theta'}{\partial \omega_t} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{-sin(\theta)+sin(\theta+\omega_t\delta t)}{\omega_t} & \frac{v_t(sin(\theta)-sin(\theta+\omega_t\delta t))}{\omega_t^2} + \frac{v_t(cos(\theta+\omega_t\delta t)\delta t}{\omega_t} \\ \frac{cos(\theta)-cos(\theta+\omega_t\delta t)}{\omega_t} & -\frac{v_t(cos(\theta)-cos(\theta+\omega_t\delta t))}{\omega_t^2} + \frac{v_t(sin(\theta+\omega_t\delta t)\delta t}{\omega_t} \\ 0 & \delta t \end{pmatrix} \tag{9}$$

Let's calculate using sympy
`https://colab.research.google.com/drive/1Zd3ymJ0Cq83X_G1eTQXpJxPFiLBtS8Bh?usp=sharing`

EKF Localization with known correspondence

- Correction step: sensor reading

$$\underbrace{\begin{bmatrix} r_t^i \\ \theta_t^i \end{bmatrix}}_{z_t^i} = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ atan2(m_{j,y} - y, m_{j,x} - x) - \theta \end{pmatrix}}_{h(x_{t,j,m})} + N(0, R) \tag{10}$$

  , where $m_{j,x}, m_{j,y}$ denotes the coordinates of jth landmark detection at time t, $R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$, and $\mathbf{x}_{t,x}^- = x, \mathbf{x}_{t,y}^- = y$

- The Taylor approximation of the measurement model
  $h(x_t, j, m) \approx h(\hat{\mathbf{x}}_t^-, j, m) + H_t^i(x_t - \hat{\mathbf{x}}_t^-)$

$$
\begin{aligned}
H_t^i = \frac{\partial h(\hat{\mathbf{x}}_t^-, j, m)}{\partial \hat{\mathbf{x}}_t^-} &= \begin{pmatrix} \frac{\partial r_t^i}{\partial \hat{\mathbf{x}}_{t,x}^-} & \frac{\partial r_t^i}{\partial \hat{\mathbf{x}}_{t,y}^-} & \frac{\partial r_t^i}{\partial \hat{\mathbf{x}}_{t,\theta}^-} \\ \frac{\partial \Phi_t^i}{\partial \hat{\mathbf{x}}_{t,x}^-} & \frac{\partial \Phi_t^i}{\partial \hat{\mathbf{x}}_{t,y}^-} & \frac{\partial \Phi_t^i}{\partial \hat{\mathbf{x}}_{t,\theta}^-} \end{pmatrix} \\
&= \begin{pmatrix} -\frac{m_{j,x} - \hat{\mathbf{x}}_{t,x}^-}{\sqrt{q}} & -\frac{m_{j,y} - \hat{\mathbf{x}}_{t,y}^-}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \hat{\mathbf{x}}_{t,y}^-}{q} & -\frac{m_{j,x} - \hat{\mathbf{x}}_{t,x}^-}{q} & -1 \end{pmatrix}
\end{aligned} \tag{11}
$$

where $q = (m_{j,x} - \hat{\mathbf{x}}_{t,x}^-)^2 + (m_{j,y} - \hat{\mathbf{x}}_{t,y}^-)^2$

We can formulate the location problem with EKF with known correspondence

EKF

$$\boxed{\Phi_t = \left.\frac{\partial f(\mathbf{x}_t, t)}{\partial \mathbf{x}}\right|_{\mathbf{x}_t}}$$

$$\boxed{\hat{\mathbf{x}}_k^- = f(\mathbf{x}_t, \mathbf{t})}$$

$$\mathbf{P}_t^- = \Phi_t \mathbf{P}_t \Phi_t^\mathsf{T} + \mathbf{Q}_t$$

$$\boxed{\mathbf{H} = \left.\frac{\partial h(\hat{\mathbf{x}}_t^-)}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}_t^-}}$$

$$\mathbf{y} = \mathbf{z}_t - \boxed{h(\hat{x}_t^-)}$$

$$\mathbf{K_t} = \mathbf{P_t^-} \mathbf{H_t}^\mathsf{T} (\mathbf{H_t} \mathbf{P_t^-} \mathbf{H_t}^\mathsf{T} + \mathbf{R}_t)^{-1}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_t^- + \mathbf{K_t} \mathbf{y}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K_t} \mathbf{H_t}) \mathbf{P_t^-}$$

PF Localization with known correspondence

- Monte Carlo localization (MCL), also known as particle filter localization
- Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled. Ultimately, the particles should converge towards the actual position of the robot

PF Localization with known correspondence

- Initialize a set of N particles x_k^i random or some prior distribution $p(x_0)$
- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}_{k-1|k-1}^i$, to which add random noise
  - The obtained predicted a set of particles $\hat{x}_{k|k-1}^i$
- Correction
  - Estimate the measurement for each particles $\hat{x}_{k|k-1}^i$
  - Evaluate the particle importance: difference between obtained measurement z_k and estimated particle measurements $\hat{z}_k^i$, i.e., $innov_k^i = z_k - \bar{z}_k^i$ (innovation or measurement residual)
  - Importance sampling: way to select importance particles $w_k^i = det(2\pi R)^{-1/2} e^{1/2 (innov_k^i)^\top R^{-1} (innov_k^i)}$
  - Estimate $\hat{x}_{k|k}^i$ as the average value of the all the particles

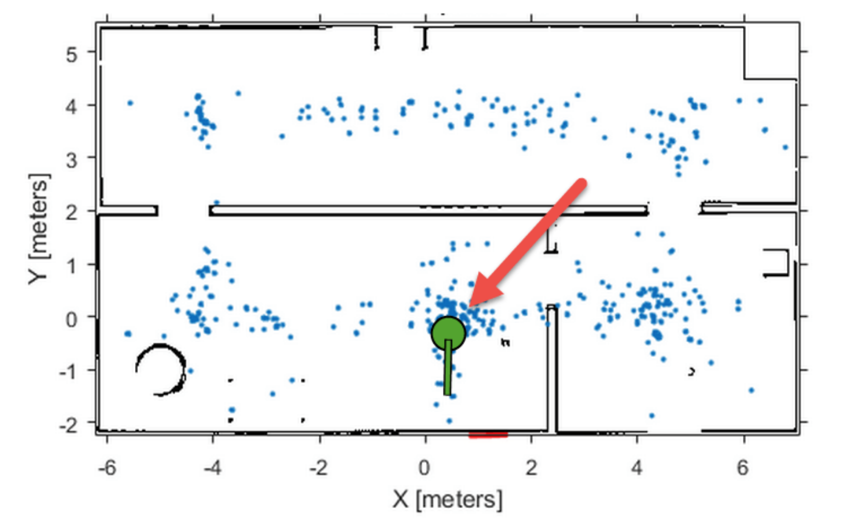
- Sampling motion model

Algorithm 1: Sample motion model velocity

Input: $\mathbf{u}_k, \mathbf{x}_k$
Result: $\mathbf{x}_{k+1} = (x', y', \theta')^\top$
$v = N(\mathbf{u}_k^v, \alpha_v)$
$\omega = N(\mathbf{u}_k^\omega, \alpha_\omega)$
$x' = \mathbf{x}_k^x + \delta_k \cdot v \cdot cos(x_k^\theta)$
$y' = \mathbf{x}_k^y + \delta_k \cdot v \cdot sin(x_k^\theta)$
$\theta' = \mathbf{x}_k^\theta + \delta_k \cdot \omega$

Algorithm requires as input the current robot pose $\mathbf{x}_k$, and the desired control $\mathbf{u}_k$ that is expressed as normal distributions separately for velocity and angular velocity with α_v, α_ω variances, respectively.

PF Localization with known correspondence



https://nl.mathworks.com/help/nav/ug/monte-carlo-localization-algorithm.html

- Measurement update

Algorithm 2: Measurement update

Input: $f_k^i, \mathbf{x}_k, m$

Result: w

$\hat{r} = \sqrt{(m_{j,x} - \mathbf{x}_k^x)^2 + (m_{j,y} - \mathbf{x}_k^y)^2}$

$\hat{\Phi} = atan2(m_{j,y} - \mathbf{x}_k^y, m_{j,x} - \mathbf{x}_k^x)$

$w = prob(r_k^i - \hat{r}, \sigma_r) \cdot prob(\Phi_k^i - \hat{\Phi}, \sigma_\Phi)$

Algorithm requires as input an observed feature $f_k^i = (r_k^i, \Phi_k^i)$, current robot pose $\mathbf{x}_k$, and the map m

- Monte Carlo Localization based on particle filter

Algorithm 3: Monte Carlo Localization based on particle filter

Input: $\mathbf{x}_{k-1}, \mathbf{u}_k, m, z_k$
Result: $\mathbf{x}_k = (x', y', \theta')^\top$
for *m=1 to to M do* **do**
  for *p=1 to to N do* **do**
    $\mathbf{x}_k^{[p]} = sample\_motion\_model(\mathbf{u}_k, \mathbf{x}_{k-1}^{[m]})$
    $\mathbf{w}_k^{[p]} = measurement\_model(z_t, \mathbf{x}_k^{[m]}, m)$
  end
  $importance\_sampling(\mathbf{w}_k)$ Resample according to importance weights
  $\mathbf{w}_k = \frac{\mathbf{w}_k}{\Sigma_{p=0}^{N}\mathbf{w}_k^{[p]}}$
  $\mathbf{x}_k = \frac{1}{N}\Sigma_{p=0}^{N}\mathbf{x}_k^{[p]}$
end

Algorithm requires as input an observed feature z_k, robot pose $\mathbf{x}_{k-1}$, the map m. The pth particle is denoted by $\mathbf{x}_k^{[p]}$ and corresponding weight, denoted $\mathbf{w}_k^{[p]}$

References

- Clark Chris.
  Markovlocalization, September 2022.
  [Online; posted 2-September-2022].

- Gregor Klancar, Andrej Zdesar, Saso Blazic, and Igor Skrjanc.
  Wheeled mobile robotics: from fundamentals towards autonomous systems.
  Butterworth-Heinemann, 2017.

- Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza.
  Introduction to autonomous mobile robots.
  MIT press, 2011.

- Sebastian Thrun.
  Probabilistic robotics.
  Communications of the ACM, 45(3):52–57, 2002.