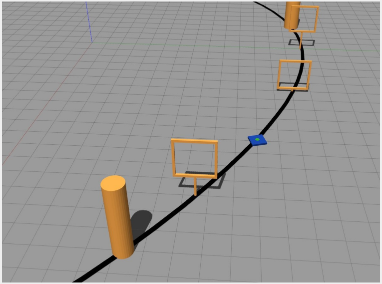
Autonomous Mobile Robotics

Robot Localization

Geesara Kulathunga

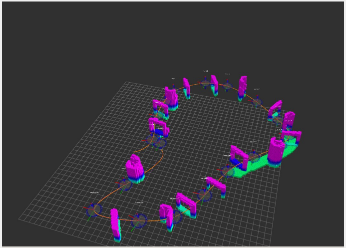
March 7, 2023



Robot Localization

Contents

- 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

- 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)

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)

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

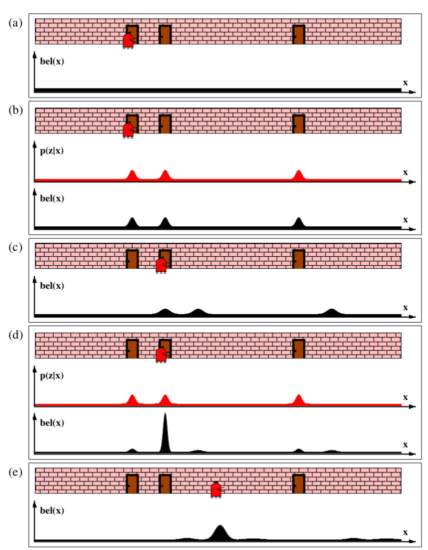
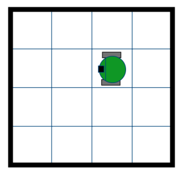


Illustration of the Markov localization algorithm, Thrun, Sebastian. "Probabilistic robotics." Communications of the ACM 45.3 (2002): 52-57.

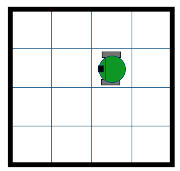


.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

robot initial belief

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

Grid-based localization

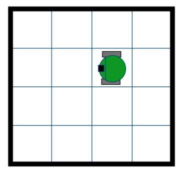


.02	.05	.05	.05
.02	.05	.18	.05
.05	.05	.18	.05
.05	.05	.05	.05

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

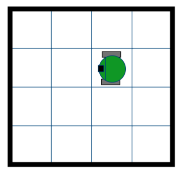
Grid-based localization



.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)



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

robot initial belief

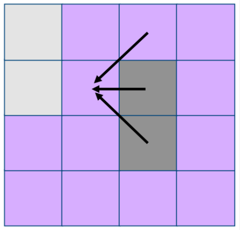
- The map is discretized into 16 cells, each of which has an area of 1m^2
- Consider the initial belief of the robot position is given
- If **the control command** to the robot is given by $\delta x, \delta y$ = -1.0 cells, 0.0 cells, what is the probability that robot be in the position (2,3)
- The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

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

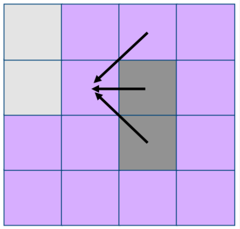
$\xrightarrow{(\Delta x, \Delta y)}$

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

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



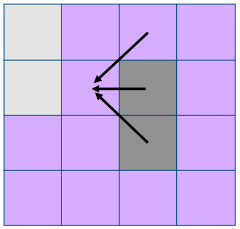
- How many possible ways to get to (2,3)?



  - 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

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

  - Correction step

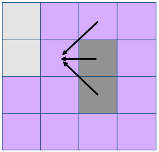
$$p(x_k|z_{1:k}, u_{0:k-1}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1}, u_{0:k-1})}{p(z_k|z_{1:k-1}, u_{0:k-1})} \quad (2)$$

, where

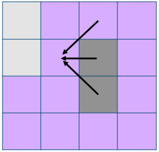
$$p(z_k|z_{1:k-1}, u_{0:k-1}) = \Sigma_{x_k\in X}\ p(z_k|x_k)p(x_k|z_{1:k-1}, u_{0:k-1})$$

Grid-based localization

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



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

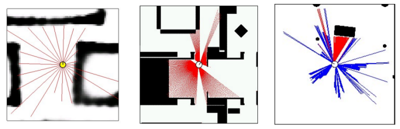


  - Prediction step

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

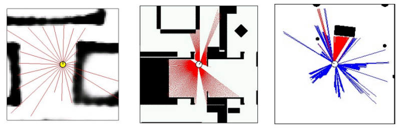
Correction step

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



Correction step

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

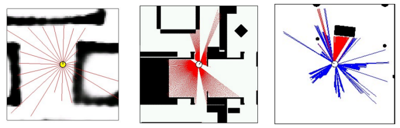


- If each sensor reading consists of N measurements, i.e., $z = \{z_1, ..., z_n\}$, assuming each such **measurement is independent** given the robot pose,

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

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

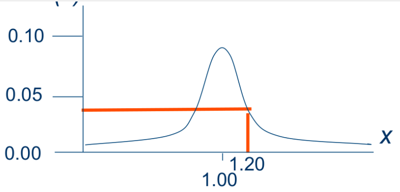
$$p(x_{i,t}|z_t) = \frac{p(z_t|x_{i,t})p(x_{i,t}|u_t)}{p(z_t)}$$

- $p(z_t|x_{i,t})$ 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

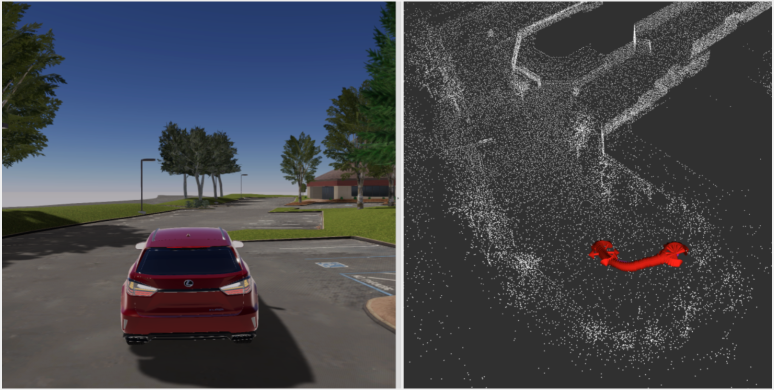
$$1 = \Sigma_{i=1}^{n} p(x_{i,t}|z_t = 1.2)$$
$$1 = \frac{\Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t})p(x_{i,t}|u_{i,t})}{p(z_t = 1.2)}$$
$$p(z_t = 1.2) = \Sigma_{i=1}^{n} p(z_t = 1.2|x_{i,t})p(x_{i,t}|u_{i,t})$$

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

- 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

EKF Localization

- Specific case of **Markov localization**

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

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

- 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:

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}

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}

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}
  - Map and a set of features $z_t = \{z_t^1, z_t^2, ...\}$ measured at time k and those are corresponded to variables $c_t = \{c_t^1, c_t^2, ...\}$

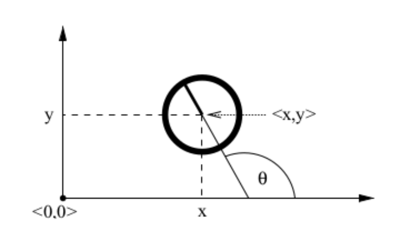
- Specific case of **Markov localization**
- Represents beliefs $bel(x_t)$ by **their first and second moment**, i.e., the mean μ_t and the covariance Σ_t
- **Map** is represented by **a collection of features** and those are known
- Initially, it requires following information:
  - robot pose at time $k-1$ 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
	$\boxed{\Phi_k = \left.\frac{\partial f(\mathbf{x}_k, t)}{\partial \mathbf{x}}\right\|_{\mathbf{x}_k}}$
$\hat{\mathbf{x}}_k^- = \Phi_k \mathbf{x}_k$	$\boxed{\hat{\mathbf{x}}_k^- = f(\mathbf{x}_k, \mathbf{t})}$
$\mathbf{P}_k^- = \Phi_k \mathbf{P}_k \Phi_k^\mathsf{T} + \mathbf{Q}_k$	$\mathbf{P}_k^- = \Phi_k \mathbf{P}_k \Phi_k^\mathsf{T} + \mathbf{Q}_k$
	$\boxed{\mathbf{H} = \left.\frac{\partial h(\hat{\mathbf{x}}_k^-)}{\partial \hat{\mathbf{x}}}\right\|_{\hat{\mathbf{x}}_k^-}}$
$\mathbf{y} = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-$	$\mathbf{y} = \mathbf{z}_k - \boxed{h(\hat{x}_k^-)}$
$\mathbf{K_k} = \mathbf{P_k^-} \mathbf{H_k}^\mathsf{T} (\mathbf{H_k} \mathbf{P_k^-} \mathbf{H_k}^\mathsf{T} + \mathbf{R}_k)^{-1}$	$\mathbf{K_k} = \mathbf{P_k^-} \mathbf{H_k}^\mathsf{T} (\mathbf{H_k} \mathbf{P_k^-} \mathbf{H_k}^\mathsf{T} + \mathbf{R}_k)^{-1}$
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K_k} \mathbf{y}$	$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K_k} \mathbf{y}$
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K_k} \mathbf{H_k}) \mathbf{P_k^-}$	$\mathbf{P}_k = (\mathbf{I} - \mathbf{K_k} \mathbf{H_k}) \mathbf{P_k^-}$

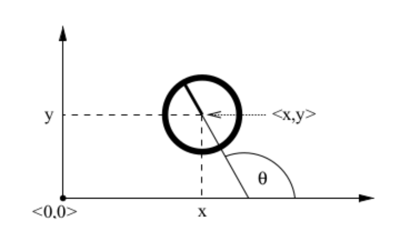
Probabilistic Motion Model

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

- 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 $\mathbf{x}_t$

Velocity Motion Model (Noise-free)

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

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

- 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** $r = |\frac{v}{u}|$

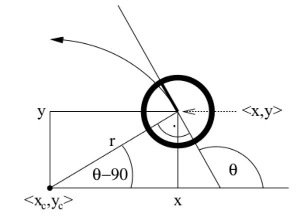
Velocity Motion Model (Noise-free)

- 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

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
- 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 $\omega\delta t$

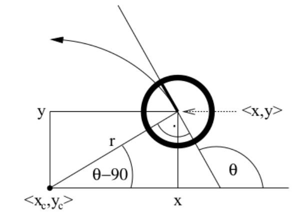
Velocity Motion Model (Noise-free)



- 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 $\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} & y_{t+1} & \theta_{t+1} \end{bmatrix}$

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