5

Robot Motion

5.1> Introduction

⇒ This chapter focus on motion model.

⇒ These models comprise the state transition probability $p(x_t | u_t, x_{t-1})$.

(Plays important role in the Prediction Step of the Bayes filter)

⇒ Probabilistic robotics generalizes Kinematic equations to the fact that the outcome of a Control is uncertain, due to Control noise or unmodeled exogenous effects.

⇒ In theory, the goal of a proper probabilistic model may appear to accurately model the specific type of uncertainity that exist in robot actuation and perception.

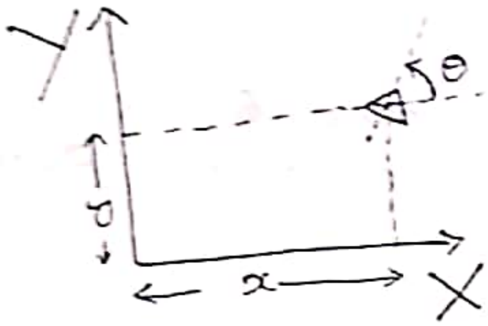
→ In practice, the exact shape of the model often seems to be less important than the fact that some provisions for the uncertain outcome are provided in the first place.

5.2> Preliminaries

5.2.1> Kinematic configuration

⇒ It is calculus describing the effect of control actions on the configuration of a robot.

⇒ Mobile robot operating in planar environments, kinematic state or pose, is summarized by three variables.



⇒ So, pose of the robot is described by the following vector:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

⇒ The orientation of a robot is often called bearing or heading direction.

⇒ Pose without orientation will be called location.

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

⇒ Both the pose and the locations of objects in the environment may constitute the Kinematic state x_t of the robot-environment system.

5.2.2 ⟩ Probabilistic Kinematics

$$p(x_t \mid u_t, x_{t-1})$$

⇒ x_t and x_{t-1} are robot pose at time t and $t-1$ respectively.

⇒ u_t is motion command. (for duration of $(t-1) \rightarrow t$)

⇒ This chapter provides in detail two specific probabilistic motion models $p(x_t \mid u_t, x_{t-1})$, both for mobile robots operating in the plane.

⇒ First model

↳ Assumes that the motion data u_t specifies the Velocity commands given to the robot's motors.

⇒ Second model

↳ Assumes that one is provided with Odometry information.

(distance travelled, angle turned)

⇒ In practice, Odometry models tend to be more accurate than velocity models, for the simple reasons that most commercial robots donot execute velocity commands with the level of accuracy that can be oblained by measuring the revolution of the robot's wheels.

⇒ Odometry models are usually applied for estimation, whereas velocity models are used for probabilistic motion planning.

5.3> Velocity motion model

⇒ The velocity motion model assumes that we can control a robot through two velocities:
- → Rotational velocity
- → Translational velocity

⇒ Translational velocity at time t ⇒ V_t {forward motion positive}

⇒ Rotational velocity at time t ⇒ ω_t {Counter-clockwise rotation positive}

So $\boxed{u_t = \begin{pmatrix} V_t \\ \omega_t \end{pmatrix}}$

5.3.1) Closed Form Calculation

$x_{t-1} = (x, y, \theta)^T$ {Initial pose}

Input: $u_t = (v, \omega)^T$ {Control Command}

$x_t = (x', y' \theta')^T$ {hypothesized successor pose}

Output: $p(x_t | u_t, x_{t-1})$

{Probability of being at x_t after executing control u_t beginning in state x_{t-1} assuming the control is carried out for a fixed duration Δt.}

$\alpha_1 - \alpha_6$ are robot specific motion error parameters.

⇒ The algorithm first calculates the controls of an error-free robot. ($\hat{v}$ and $\hat{\omega}$)

⇒ The function prob(a, b) models the motion errors.

{Probability of parameter a under a zero-centered random variable with variance b.}

Algorithm motion_model_velocit (x_t, u_t, x_{t-1}):

$$\mu = \frac{1}{2} \frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}$$

$$x^* = \frac{x+x'}{2} + \mu(y-y')$$

$$y^* = \frac{y+y'}{2} + \mu(x'-x)$$

$$r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$$

$$\Delta\theta = \operatorname{atan2}(y'-y^*, x'-x^*) - \operatorname{atan2}(y-y^*, x-x^*)$$

$$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$$

$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$

$$\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$$

return $\text{prob}(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|)$
$\cdot\, \text{prob}(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|)$
$\cdot\, \text{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

⇒ Two possible implementation of Prob() function.

① Error varies with normal distribution

Prob(a, b):

return $\frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{a^2}{b}}$

② Error varies with triangular distribution

Prob(a, b):

if $|a| > \sqrt{6b}$

return 0

else

return $\frac{\sqrt{6b} - |a|}{6b}$

5.3.2> Sampling Algorithm

⇒ In sampling, one is given u_t and x_{t-1} and seeks the generate a random x_t drawn according the motion model $p(x_t \mid u_t, x_{t-1})$

⇒ line 2 through 4 "perturb" the commanded control parameters by noise, drawn from the error parameters of the kinematic motion model

Algorithm sample_motion_model_velocity (u_t, x_{t-1}):

$\hat{v} = v + \text{sample}(\alpha_1 |v| + \alpha_2 |\omega|)$

$\hat{\omega} = \omega + \text{sample}(\alpha_3 |v| + \alpha_4 |\omega|)$

$\hat{\gamma} = \text{sample}(\alpha_5 |v| + \alpha_6 |\omega|)$

$x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega} \Delta t)$

$y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega} \Delta t)$

$\theta' = \theta + \hat{\omega} \Delta t + \hat{\gamma} \Delta t$

return $x_t = (x', y', \theta')^T$

$\Rightarrow$ **Sampling algorithm for normal distribution** (b):

Sample(b):

return $\frac{b}{6} \sum_{i=1}^{12} \text{rand}(-1, 1)$

$\Rightarrow$ **Sampling algorithm for triangular distribution** (b):

Sample(b):

return $b \cdot \text{rand}(-1, 1) \cdot \text{rand}(-1, 1)$

5.4> Odometry motion model

⇒ Alternatively one might want to use the odometry measurements as the basis for calculating the robot's motion over time.

Odometry
↳ Commonly obtained by integrating wheel encoder information.

5.4.1> Closed Form Calculation

⇒ Technically, odometry are sensor measurements, not control.

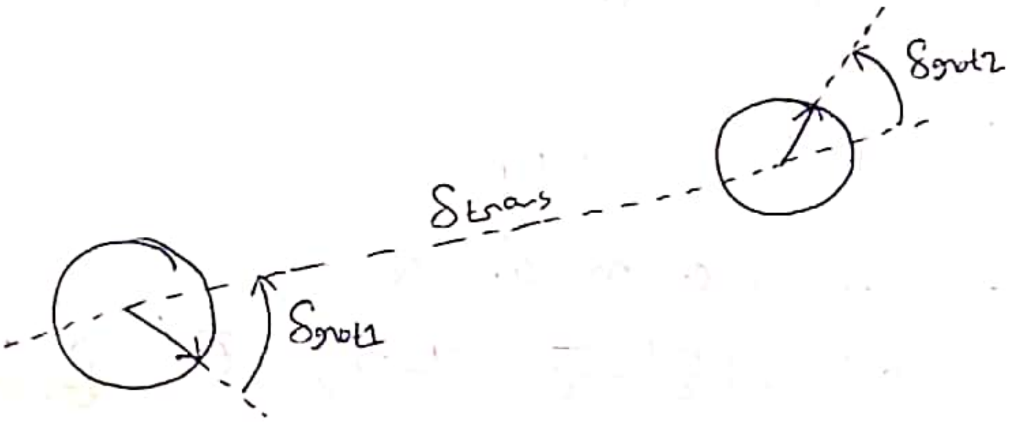
⇒ To model odometry as measurements, the resulting Bayes filter would have to include the actual velocity as state variables – which increases the dimension of the stat space.

↳ To keep state space small, it is therefore common to simply consider the odometry as if it was a control signal.

↳ The resulting model is as the core of many of today's best probabilistic robot systems.

⇒ The odometry model uses the relative information of the robot's internal odometry.

$$u_t = \begin{pmatrix} \bar{x}_{t-1} \\ \bar{x}_t \end{pmatrix} \left\{ \text{bar indicate that these are odometry measurement} \right\}$$



⇒ To extract relative odometry, u_t is transformed into a sequence of three steps:

① Rotation (δ_{rot1})
↓
② translation (δ_{trans})
↓
③ Rotation (δ_{rot2})

⇒ For each pair of position $(\bar{s}, \bar{s}')$ has a unique parameter vector $(\delta_{rot1}\ \delta_{trans}\ \delta_{rot2})^T$.

⇒ Our motion model assumes that these three parameters are corrupted by independent noise.

Input: x_{t-1} {Initial pose}

$u_t = (\bar{x}_{t-1} \;\; \bar{x}_t)^T$ {Pair of pose}

x_t {hypothesized final pose}

Output: $p(x_t \mid u_t, x_{t-1})$

Algorithm motion model odometry (x_t, u_t, x_{t-1}):

$\delta_{rot1} = atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$

$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

from Odometry

$\hat{\delta}_{rot1} = atan2(y' - y, x' - x) - \theta$

$\hat{\delta}_{trans} = \sqrt{(x - x')^2 + (y - y')^2}$

$\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$

from given Values

$P_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1 \hat{\delta}_{rot1} + \alpha_2 \hat{\delta}_{trans})$

$P_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3 \hat{\delta}_{trans} + \alpha_4 (\hat{\delta}_{rot1} + \hat{\delta}_{rot2}))$

$P_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1 \hat{\delta}_{rot2} + \alpha_2 \hat{\delta}_{trans})$

return $P_1 \cdot P_2 \cdot P_3$

⇒ Here the implementer must observe that all angular difference must lie in $[-\pi, \pi]$.

⇒ The variables α_1 through α_4 are robot-specific parameters that specify the noise in robot motion.

5.4.1) Sampling Algorithm

⇒ If particle filters are used for localization, we would also like to have an algorithm for sampling from $p(x_t \mid u_t, x_{t-1})$

Input: x_{t-1} (initial pose)

u_t (odometry reading)

Output: x_t (distributed according to $p(x_t \mid u_t, x_{t-1})$)

Algorithm sample motion odometry (u_t, x_{t-1}):

$$\delta_{rot1} = \operatorname{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

$$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$

$$\hat{\delta}_{rot1} = \delta_{rot1} - \text{sample}(\alpha_1 \delta_{rot1} + \alpha_2 \delta_{trans})$$

$$\hat{\delta}_{trans} = \delta_{trans} - \text{sample}(\alpha_3 \delta_{trans} + \alpha_4 (\delta_{rot1} + \delta_{rot2}))$$

$$\hat{\delta}_{rot2} = \delta_{rot2} - \text{Sample}(\alpha_1 \delta_{rot2} + \alpha_2 \, trans)$$

$$x' = x + \hat{\delta}_{trans} \cos(\theta + \delta_{rot1})$$

$$y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$$

$$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$$

return $x_t = (x', y', \theta')^T$

5.5> <u>Motion and Map</u>

⇒ In above motion model describes robot motion ~~ext~~ in absence of any knowledge about the nature of environment.

↳ In many cases, we are also given a map m, which may contain information pertaining to the places that a robot may or may not be able to navigate.

{Example: Occupancy Map}

⇒ This consideration calls for a motion model that takes the map m into account.

$$p(x_t | u_t, x_{t-1}, m)$$

⇒ The motion model $p(x_t | u_t, x_{t-1}, m)$ should give better results than the map-free motion model $p(x_t | u_t, x_{t-1})$.

↳ We will refer to $p(x_t | u_t, x_{t-1}, m)$ as <u>map-based motion model</u>.

⇒ Unfortunately, computing this motion model in closed form is difficult.

⇒ Luckily, there exists an efficient approximation for the map-based motion model, which works well if the distance between x_{t-1} and x_t is small.

↳ (i.e. Smaller than half of robot dimension)

$$p(x_t \mid u_t, x_{t-1}, m) = \eta \, p(x_t \mid u_t, x_{t-1}) \, p(x_t \mid m)$$

→ $p(x_t \mid m) = 0$ {if robot collides with an occupied grid cell in the map}

Const {Otherwise}

<u>Algorithm motion model with map</u> (x_t, u_t, x_{t-1}, m):

  return $p(x_t \mid u_t, x_{t-1}) \cdot p(x_t \mid m)$

<u>Algorithm sample motion model with map</u> (u_t, x_{t-1}, m):

  do

    x_t = Sample-motion-model (u_t, x_{t-1})

    $\pi = p(x_t \mid m)$

  until $\pi > 0$

  return $\langle x_t, \pi \rangle$