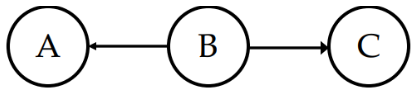
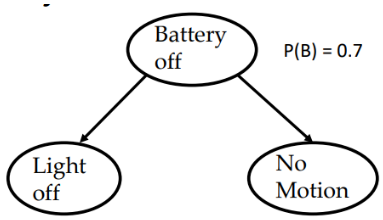
Slides 4

- Bayesian Networks
    - Dependencies between variables, if you know one, the probability of other ones may change



• Knowing about A will tell us something about C.
Knowing about C will tell us something about A.

- • If we know B propagation is blocked

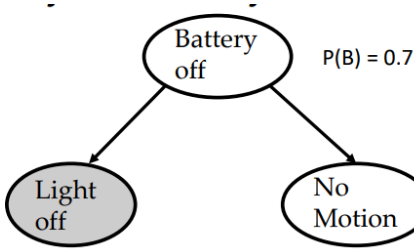


	P(L|B)
B	0.8
¬B	0.1

	P(M|B)
B	1.0
¬B	0.1

$$\begin{aligned} P(M) &= P(M|B)P(B) + P(M|\neg B)P(\neg B) \\ &= 1.0 * 0.7 + 0.1 * 0.3 \\ &= 0.7 + 0.03 \\ &= 0.73 \end{aligned}$$

-

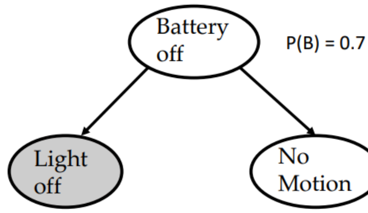


	P(L|B)
B	0.8
¬B	0.1

	P(M|B)
B	1.0
¬B	0.1

$$\begin{aligned} P(B|L) &= \frac{P(L|B)P(B)}{P(L)} \\ &= \frac{0.8 * 0.7}{0.59} \\ &= 0.95 \end{aligned}$$

-



	P(L|B)
B	0.8
¬B	0.1

	P(M|B)
B	1.0
¬B	0.1

$$\begin{aligned} P(M|L) &= P(M,B|L) + P(M,\neg B|L) \\ &= P(M|B,L)P(B|L) + P(M|\neg B,L)P(\neg B|L) \\ &= P(M|B)P(B|L) + P(M|\neg B)P(\neg B|L) \\ &= 1.0 * 0.95 + 0.1 * 0.05 \\ &= 0.95 + 0.005 \\ &= 0.96 \end{aligned}$$

- Markov Assumption: Only the previous state matters, nothing before it matters
- Stationary Dynamics: Rules of the environment do not change over time

Slides 5

$$b(x_t) = \eta P(z_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}, u_t) b(x_{t-1})$$

Algorithm **Bayes Filter**($b(x_{t-1}), u_t, z_t$)
For all x_t do:

$$\bar{b}(x_t) = \sum_{x_{t-1}} P(x_t|x_{t-1}, u_t) b(x_{t-1})$$

$$b(x_t) = \eta P(z_t|x_t) \bar{b}(x_t)$$

Endfor
Return $b(x_t)$

Prediction Step:

- `P(x_t | x_{t-1}, u_t)`: This term is the transition probability, representing the probability of transitioning from state `x_{t-1}` at time `t-1` to state `x_t` at time `t`, given the input `u_t`.
- `b(x_{t-1})`: This is the prior belief state, representing the probability distribution over the states at the previous time step.
- The summation over `x_{t-1}` indicates that the prediction step involves considering all possible previous states and their corresponding probabilities.

Update Step:

- `P(z_t | x_t)`: This is the likelihood function, representing the probability of observing the measurement `z_t` given the current state `x_t`.

- The multiplication with `b(x_t)` updates the belief state based on the new measurement.
- The normalization factor `η` ensures that the resulting belief state remains a probability distribution.

Sensor Model and Action Model Example:

Sensor Model:

$$
\begin{aligned}
P(z_t = sense - open | x_t = open) &= 0.6 \\
P(z_t = sense - closed | x_t = open) &= 0.4 \\
P(z_t = sense - open | x_t = closed) &= 0.2 \\
P(z_t = sense - closed | x_t = closed) &= 0.8
\end{aligned}
$$

Action Model:

$$
\begin{aligned}
P(x_t = open | x_{t-1} = open, u_t = push) &= 1.0 \\
P(x_t = closed | x_{t-1} = open, u_t = push) &= 0.0 \\
P(x_t = open | x_{t-1} = closed, u_t = push) &= 0.8 \\
P(x_t = closed | x_{t-1} = closed, u_t = push) &= 0.2
\end{aligned}
$$

$$
\begin{aligned}
P(x_t = open | x_{t-1} = open, u_t = do - nothing) &= 1.0 \\
P(x_t = closed | x_{t-1} = open, u_t = do - nothing) &= 0.0 \\
P(x_t = open | x_{t-1} = closed, u_t = do - nothing) &= 0.0 \\
P(x_t = closed | x_{t-1} = closed, u_t = do - nothing) &= 1.0
\end{aligned}
$$

- Example in slides of Bayes Filter
- Dynamics Model: Probabilty of Transitioning to another step

Slide 6

Propagation of mean and covariance

$$\mu_t = A\mu_{t-1} + Bu_t$$

$$\Sigma_t = A\Sigma_{t-1}A^T + R$$

Mean Propagation:

- `μ_t` represents the mean of the state distribution at time `t`.
- `A` is the state transition matrix, which determines how the state changes from one time step to the next.
- `μ_{t-1}` is the mean of the state distribution at the previous time step.
- `B` is the input matrix, which determines the effect of the input on the state.
- `u_t` is the input at time `t`.

Covariance Propagation:

- `Σ_t` represents the covariance matrix of the state distribution at time `t`.
- `Σ_{t-1}` is the covariance matrix of the state distribution at the previous time step.
- `A^T` is the transpose of the state transition matrix.
- `R` is the process noise covariance matrix, which represents the uncertainty in the system dynamics.

1: **Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2: $\bar{\mu}_t = A_t\, \mu_{t-1} + B_t\, u_t$
3: $\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\, C_t^T (C_t\, \bar{\Sigma}_t\, C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\, \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t\, C_t)\, \bar{\Sigma}_t$
7: return μ_t, Σ_t

K_t = Kalman Gain
Q_t = Measurement Noise (Covariance)

- Line 3: Kalman Gain: Determines how much weight is given to measurement
- Lines 4 and 5: Updates
- For Linear Dynamical System (Linear relationship between current state and previous state as well as input)
- For non linear systems, we use linear approximations (First taylor expansion)

Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$$
$$K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$$
return μ_t, Σ_t

Prediction Step:

- In the EKF, the state prediction `μ¯t` is calculated using the nonlinear state transition function `g(Ut, μt-1)`.
- The Jacobian matrix `Gt` is used to linearize the nonlinear function around the previous state estimate.

Update Step:

- In the EKF, the measurement prediction `h(μ¯t)` is calculated using the nonlinear measurement function `h(μ¯t)`.

- The Jacobian matrix `Ht` is used to linearize the nonlinear function around the predicted state.
- Example in Slides

Slide 8
Non Gaussian State Distributions

Algorithm Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

$$
\begin{aligned}
&\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset \\
&\text{for } m = 1 \text{ to } M \text{ do} \\
&\qquad \text{sample } x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]}) \\
&\qquad w_t^{[m]} = p(z_t \mid x_t^{[m]}) \\
&\qquad \bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle \\
&\text{endfor} \\
&\text{for } m = 1 \text{ to } M \text{ do} \\
&\qquad \text{draw } i \text{ with probability} \propto w_t^{[i]} \\
&\qquad \text{add } x_t^{[i]} \text{ to } \mathcal{X}_t \\
&\text{endfor} \\
&\text{return } \mathcal{X}_t
\end{aligned}
$$

- First for loop: Samples a new particle and updates the particle based on a weight calculation (from likelihood function representing probability of observing current measurement z given x)
- Second for loop: Particle is drawn with probability proportional to its weight (particles with higher weights have higher change of being selected)
- Resample to increase numberf of particles in regions of high probabilitiyes, remove particles with low weights

Slide 9

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

$$\hat{v} = v + \textbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$$
$$\hat{\omega} = \omega + \textbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$$
$$\hat{\gamma} = \textbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$$
$$x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}}\, sin(\theta + \hat{\omega}\Delta t)$$
$$y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}}\, cos(\theta + \hat{\omega}\Delta t)$$
$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$$
$$\text{return } x_t = (x', y', \theta')^T$$

Function Definition:

- Line 1: Defines the algorithm named `sample_motion_model_velocity`.
- Line 2: Takes two inputs: `Ut` (control input vector) and `Xt-1` (previous state vector).

Velocity Update:

- Line 3: Updates the estimated velocity `v^` by adding a random noise term to the previous velocity.

Angular Velocity Update:

- Line 4: Updates the estimated angular velocity `ω^` in a similar manner.

Yaw Rate Update:

- Line 5: Updates the estimated yaw rate `γ^` based on the velocity and angular velocity.

Position Update:

- Line 6: Updates the estimated position `(x', y')` using the estimated velocity, angular velocity, and orientation.

Orientation Update:

- Line 7: Updates the estimated orientation `θ'` based on the estimated angular velocity and yaw rate.

Slide 10