ECE 484: Principles of Safe Autonomy (Fall 2025)
Lecture 11
State Estimation, Filtering and Localization

Professor: Huan Zhang

https://publish.illinois.edu/safe-autonomy/

https://huan-zhang.com

huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025;

Reference: Probabilistic Robotics by Sebastian Thrun, Wolfram Burgard, and Dieter Fox

Some slides are from the book's website

Announcements

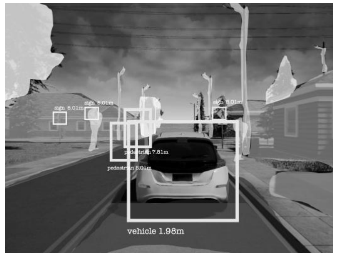
- All project teams have been formed
  - https://docs.google.com/spreadsheets/d/1fj2NtL1jLd-B9Y9_oAPuqF_AZW2nrhsQY5lfml86wQU/edit?gid=0#gid=0
  - Connect with your teammates
  - Check CampusWire if you want to switch team
- GEM, F1-Tenth, Drone Safety Training required (Check CampusWire)

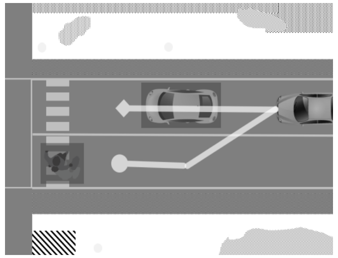
Announcements

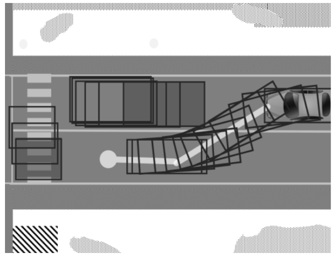
- Project pitch presentation next week! (**15%** of your project grades!)
    - Check out previous semesters' projects:
    - https://www.youtube.com/playlist?list=PLcA4s4DKSOF1Kzp0_OqOlNAGWoft2G7z6
    - https://www.youtube.com/watch?v=J0_EZeZfXWk
    - You must upload slides to Gradescope by 11:59pm on Monday, October 13th. We will only display the uploaded version during Pitch.
    - Given the time limit, we will enforce a STRICT 5-min presentation + 1-min Q&A
- Your presentation will be graded – check campuswire for grading rubrics & hints
- Check CampusWire for presentation schedule

GEM platform

Autonomy pipeline

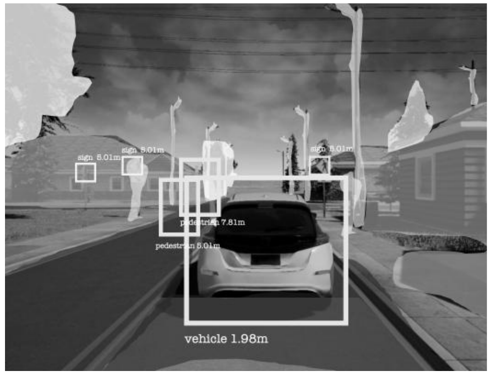






Sensing	Perception	Decisions and planning	Control
Physics-based models of camera, LIDAR, RADAR, GPS, etc.	Programs for object detection, lane tracking, scene understanding, etc.	Programs and multi-agent models of pedestrians, cars, etc.	Dynamical models of engine, powertrain, steering, tires, etc.

Can you name a few challenges in the Perception pipeline?



Perception

Programs for object detection, lane tracking, scene understanding, etc.

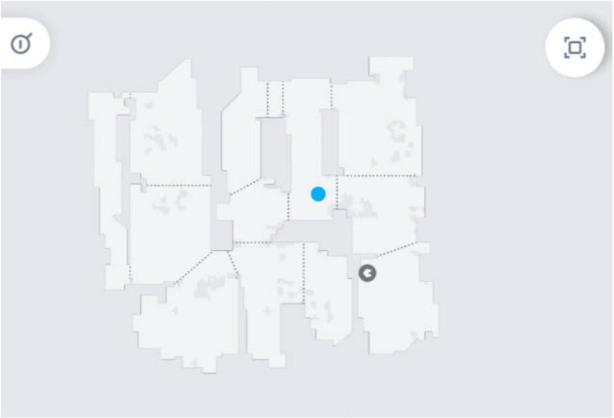
Outline of state estimation module

- Introduction: **Localization** problem, taxonomy
- Review of probability: **conditional probability** and **Bayes**’ Rule
- Probabilistic models: **motion** and **measurements**

Next lectures:

- Discrete Bayes Filter
- Histogram filter and grid localization
- Particle filter

Roomba mapping



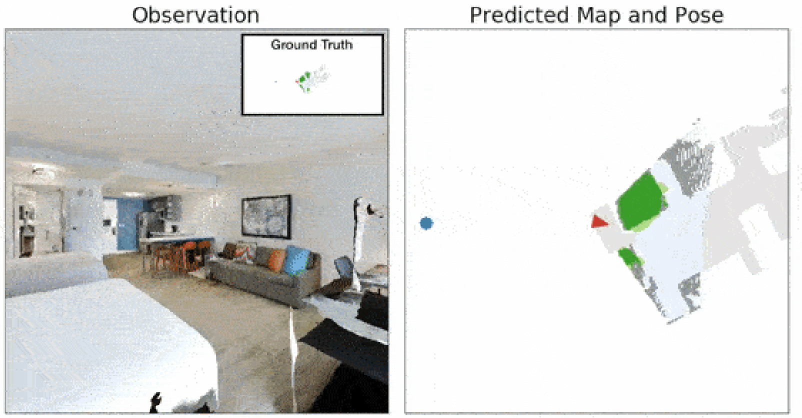


Image credit: Devendra Singh Chaplot

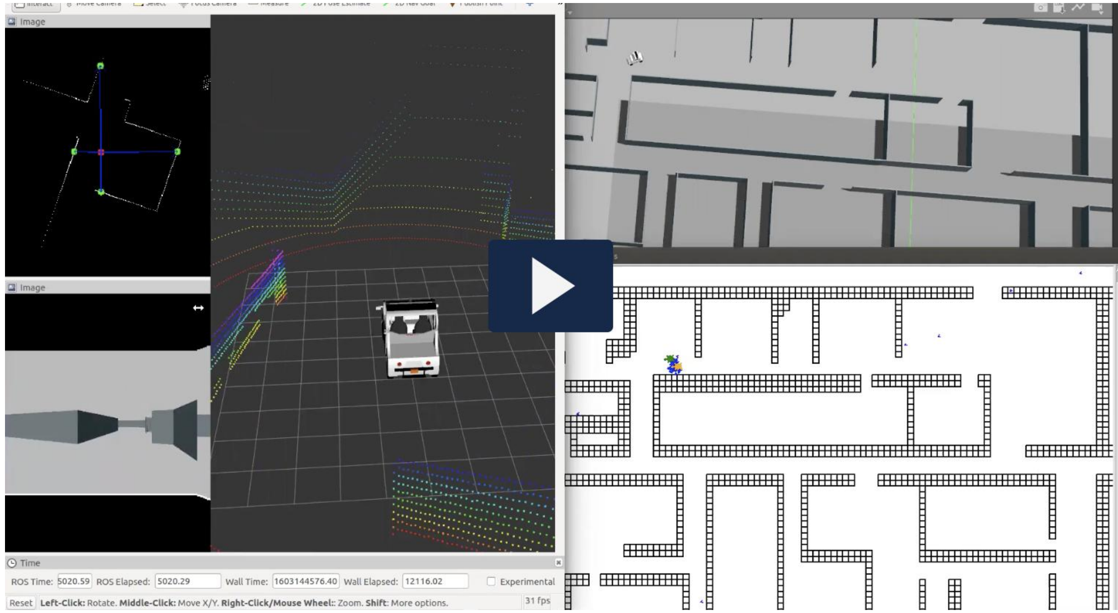
iRobot Roomba uses SLAM algorithm to create maps for cleaning areas

SLAM: Simultaneous **Localization** and **Mapping**

Also in underground, underwater, and space robots, in GPS-denied environments

State estimation and localization problem (MP3)

- For closed loop control, the controller needs to know the current state (position, attitude, pose)
  - $x_{t+1} = f(x_t, u_t);\ u_t = g(x_t)$
- Typically, the state x_t is not available directly. We have some other observables $z_t = h(x_t)$ that are available.
- Example observables: images, lidar scans, GPS, IMU
- We have to compute a **state estimate** $\hat{\boldsymbol{x}}_t$ from observations z_t so that $\hat{x}_t \approx x_t$
- Then we can use $u_t = g(\hat{x}_t)$
- **Localization** is a special case of the state estimation problem where we have to determine the **pose** of the robot relative to the *given map* of the environment



Setup: State evolution and measurement models

Familiar Deterministic model:

System evolution: $x_{t+1} = f(x_t, u_t)$

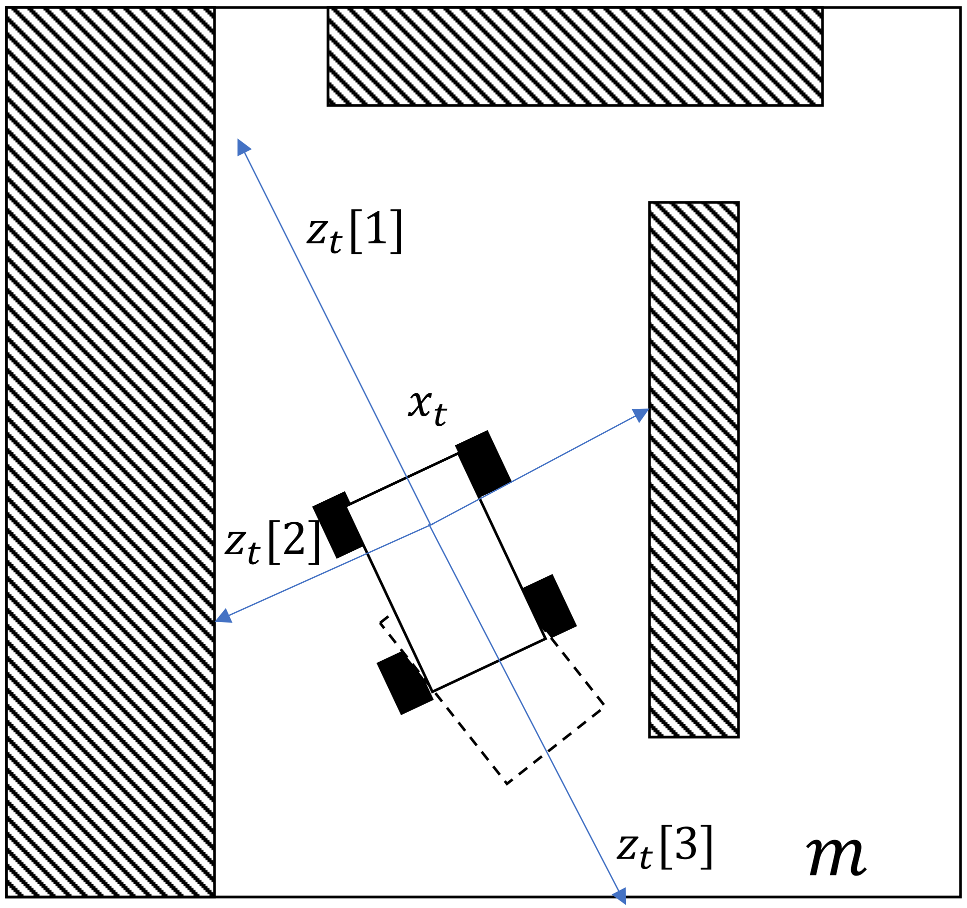
- x_t: unknown state of the system at time t
- u_t: known control input at time t, $u_t = g(\hat{x}_t)$
- f: known dynamic function, possibly stochastic

Measurement or observation: $z_t = h(x_t, m)$

- z_t: known measurement of state x_t at time t
- m: unknown underlying map
- h: known measurement function

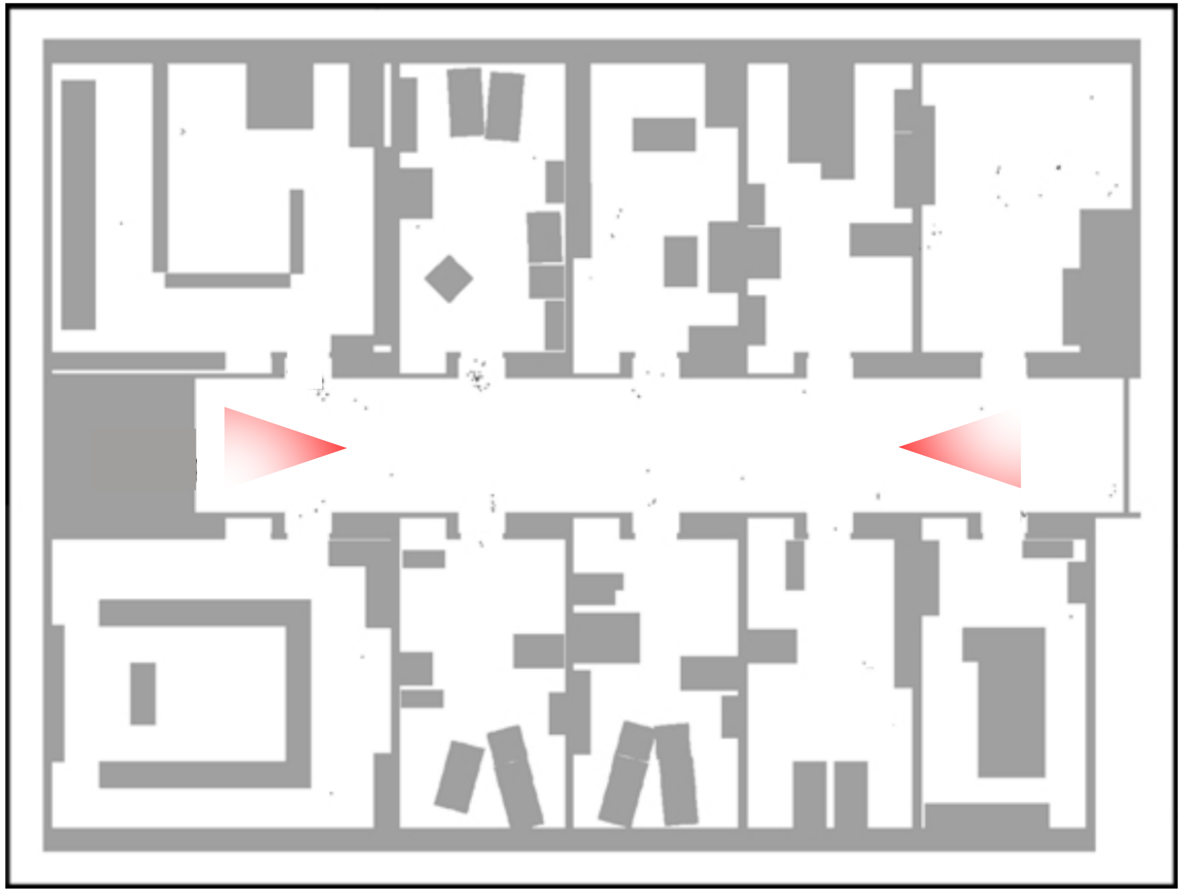
Problem: Given the sequence of measurements $z_1, z_2, \dots z_{t-1}$ and control inputs $u_1, u_2, \dots u_{t-1}$

We will use probabilistic models going forward



This is not exactly the measurement model of MP3

Ambiguity in global localization arising from locally symmetric environment

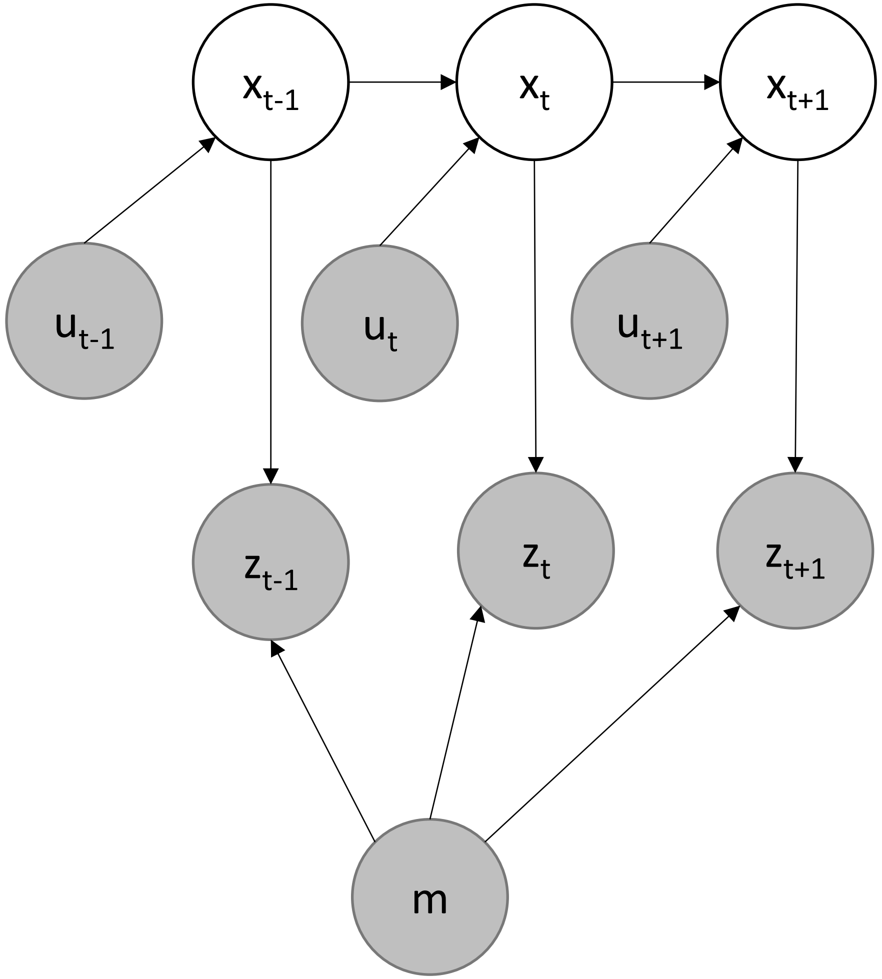


Localization as coordinate transformation

Shaded known:
map (m), control inputs (u), measurements(z).
White nodes to be determined (x)

maps (m) are described in global coordinates.
Localization = establish *coord transf.* between m and robot's local coordinates

Transformation used for objects of interest (obstacles, pedestrians) for decision, planning and control



Localization taxonomy

Global vs Local

- Local: assumes initial pose is known, has to only account for the uncertainty coming from robot motion (*position tracking problem)*
- **Global**: initial pose unknown; harder and subsumes position tracking
- Kidnapped robot problem: during operation the robot can get teleported to a new unknown location (models failures)

Static vs Dynamic Environments

Single vs Multi-robot localization

Passive vs Active Approaches

- **Passive**: localization module only observes and is controlled by other means; motion not designed to help localization (Filtering problem)
- Active: controls robot to improve localization

Discrete time model: Automaton with inputs/outputs

We will describe the systems state, inputs, and outputs as a sequence

- System evolution: $x_{t+1} = f(x_t, u_t)$
  - x_t: state of the system at time t
  - u_t: control input at time t
- Measurement: $z_t = g(x_t, m)$
  - z_t: measurement of state x_t at time t
  - m: unknown underlying map

Instead of nondeterministic automata or set-valued functions (like we used in the first part of this course), now we will model uncertainty in f and g with probability distributions

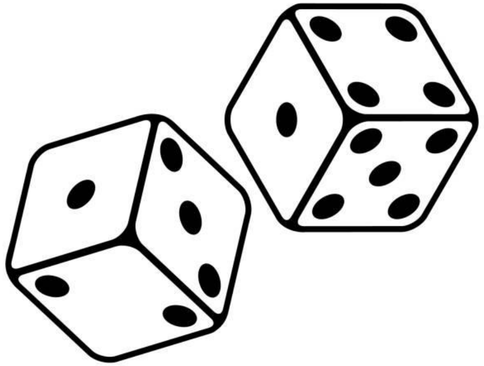
Setup, notations

- $x_{t_1:t_2} = x_{t_1}, x_{t_1+1}, x_{t_1+2}, \ldots, x_{t_2}$ sequence of states t_1 to t_2
- Robot takes one measurement at a time
  - $z_{t_1:t_2} = z_{t_1}, \ldots, z_{t_2}$ sequence of all measurements (observations) from t_1 to t_2
- Control also exercised at discrete steps
  - $u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$ sequence control inputs

Review of conditional probabilities

I rolled two fair six-sided dice. Define two random variables:

X1=value rolled on die 1 X2=value rolled on die 2



Can you calculate the probability of the following events?

P(X1=2)

P(X1+X2<=4)

P(X1=2 | X1+X2<=4) conditional probability

Conditional probabilities and Bayes Rule

A **random variable** is a function $X: \Omega \rightarrow \mathbb{R}^n$ that assigns numerical values to the outcomes of a random experiment. Ω is the sample space.

Random variable X takes values $x_1, x_2 \in \mathbb{R}^n$

Example: Result of a dice roll (X) and $x_i = 1, \ldots, 6$

$P(X = x)$ is written as $P(x)$

$P(X = x, Y = y)$ is written as $P(x, y)$

Conditional probability: $P(X = x \,|Y = y) = P(x|y) = \frac{P(x,y)}{P(y)}$ provided $P(y) > 0$

$P(x, y) = P(x|y)P(y)$

= $P(y|x)P(x)$

Substituting in the definition of Conditional Prob. we get **Bayes Rule**

$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$, provided $P(y) > 0$

Using measurements to update state estimates

$P(x|z) = \frac{P(Z|X)P(x)}{P(z)}$, provided $P(z) > 0$ *(*)*

X *: Robot position,* Z *: measurement,*

$P(x)$: *Prior distribution/belief (before measurement)*

$P(x|z)$: *Posterior distribution (after measurement)*

$P(z|x)$*: Measurement model / inverse conditional / generative model*

$P(z)$: does not depend on x; normalization constant

Example: Light Sensor Robot

- Problem Setup:
- A robot has a light sensor that detects if a light is 'on' or 'off'
- The sensor is noisy and cannot be fully trusted
- Goal: Estimate the true state of the light using noisy measurements

Sensor Model (Measurement Model)

- Sensor characteristics:
- **If light is ON: sensor reads 'on' 90% of the time**
  - p(z = 'on' | x = 'on') = 0.9
- **If light is OFF: sensor reads 'on' 40% of the time**
  - p(z = 'on' | x = 'off') = 0.4

Step 1: Initial Belief (Prior)

- Before sensing, the robot's initial belief:
- $p(x_t = \text{'on'}) = 0.5$
- $p(x_t = \text{'off'}) = 0.5$
- *The robot is completely uncertain about the light's state*

Step 2: Sensor Reading

- The sensor now reads:
- **z_t = 'on'**
- *Question: What should the robot now believe?*

- *Mathematically: we need $P(x = \text{'on'} \mid z_t = \text{'on'})$*

Step 2: Sensor Reading

- We can now apply the Bayes Rule

- $P(x_t = \text{'on'} \mid z_t = \text{'on'}) = [\ P(z_t = \text{'on'} \mid x_t = \text{'on'}) \cdot P(x_t = \text{'on'})\ ]\ /\ P(z_t = \text{'on'})$

Posterior *Measurement* *Prior/Belief* *Normalizing constant*

Step 3: Calculate normalization constant

- Calculate normalization constant $p(z_t = \text{'on'})$:

$$p(z = \text{'on'}) = p(z = \text{'on'} \mid x = \text{'on'}) \cdot p(x = \text{'on'}) + p(z = \text{'on'} \mid x = \text{'off'}) \cdot p(x = \text{'off'})$$

$= (0.9)(0.5) + (0.4)(0.5)$

$= 0.45 + 0.20$

$= 0.65$

Step 4: Calculate Posterior

- Update belief using Bayes' Law:

$p(x = \text{'on'} \mid z = \text{'on'}) = [p(z = \text{'on'} \mid x = \text{'on'}) \cdot p(x = \text{'on'})] / p(z = \text{'on'})$

$= (0.9 \times 0.5) / 0.65$

$= 0.45 / 0.65$

≈ 0.692 or 69.2%

Result & Interpretation

- Before sensing: 50% confident light is on
- After sensing 'on': 69.2% confident light is on
- The robot updated its belief by combining:
  - Prior knowledge (initial 50% belief)
  - Sensor measurement (noisy reading 'on')
  - Sensor reliability (90% accurate when on)

Evolution: probabilistic Markov Chain models

A probability distribution $\pi \in P(Q)$ over a finite set of states Q can be represented by a vector $\pi \in \mathbb{R}^{|Q|}$ where $\sum \pi_i = 1$

Recall deterministic discrete transitions for automata $D: Q \to Q$

Probabilistic discrete transitions give a probability distribution $D: Q \to \boldsymbol{P}(Q)$ according to which the next state is chosen, i.e., $D(q)$ is a particular probability distribution over Q

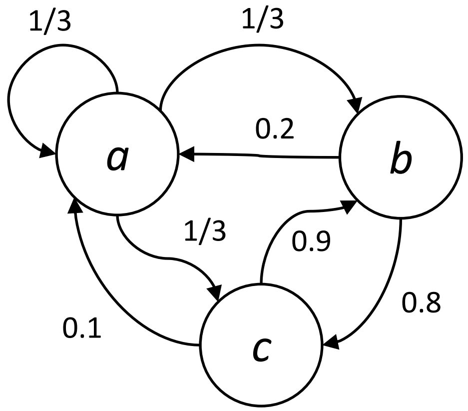
For the example on the right $p_D(X_{t+1} = b \mid X_t = a) = \frac{1}{3}$, i.e., $D(a) = [a{:}\frac{1}{3} \quad b{:}\frac{1}{3} \quad c{:}\frac{1}{3}]$ $D(b) = [a{:}\frac{1}{5} \quad b{:}0 \quad c{:}\frac{4}{5}]$

Such a state machine model is called a **Markov chain**

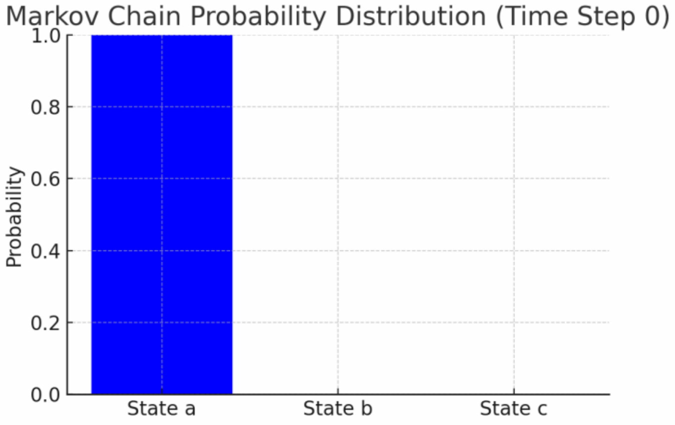
A probabilistic transition $\boldsymbol{D}$ can be represented by a matrix $\boldsymbol{D} \in \mathbb{R}^{|Q| \times |Q|}$ where D_{ij} gives the probability of state i to transition to j

The evolution of the probability π over states can be represented as

$\pi_{t+1} = \boldsymbol{D}\pi_t$ starting with an initial distribution $\pi_0 \in \boldsymbol{P}(Q)$

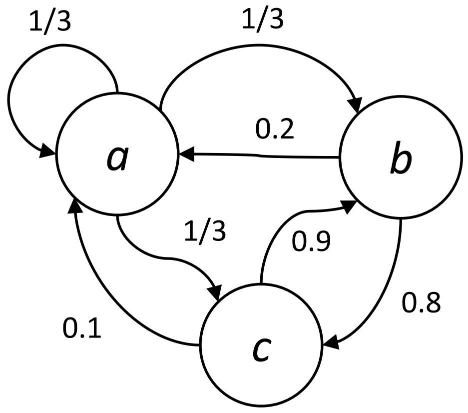


$$\boldsymbol{D} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{5} & 0 & \frac{4}{5} \\ \frac{1}{10} & \frac{9}{10} & 0 \end{bmatrix}$$



Evolution: probabilistic MDP models

More generally, transitions depend on input in which case the transition function $D: Q \times U \rightarrow P(Q)$ also depends on the control action U

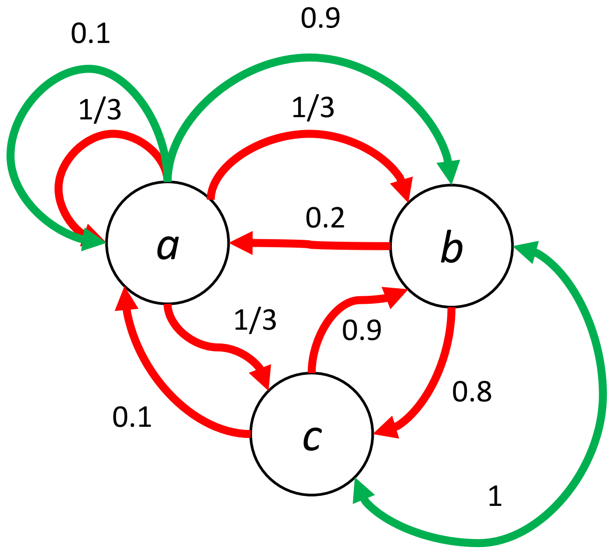


For the example below $p_D(X_{t+1} = b \,|X_t = a, U_t = red) = \frac{1}{3}$

Such a state machine model with inputs is called a **Markov Decision Process (MDP)**

The probabilistic transitions $\boldsymbol{D}$ can be represented by a collection of matrices $\boldsymbol{D}: U \rightarrow \mathbb{R}^{|Q|\times|Q|}$ where $D_{ij}(u)$ gives the probability of state i to transition to j under action u

$p_D(x'|x,u)$ if transition probabilities are time invariant



Evolution and measurement: probabilistic models

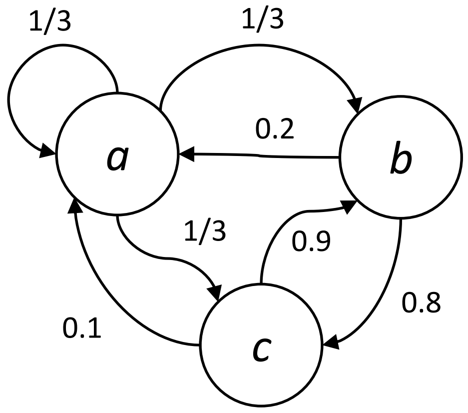
Even more generally, transitions depend on outputs and history

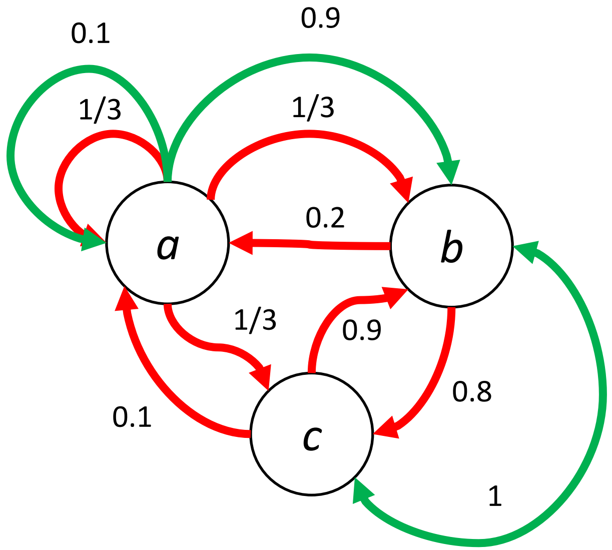
$p_D(X_t = x_t \,|X_0 = x_0, \dots X_{t-1} = x_{t-1,} Z_1 = z_1, \dots Z_{t-1} = z_{t-1,}, U_1 = u_1, \dots U_t = u_{t,})$ describes state evolution model

$p_D(x_t \,|x_{0:t-1}, z_{1:t-1}, u_{1:t})$ describes motion/state evolution model

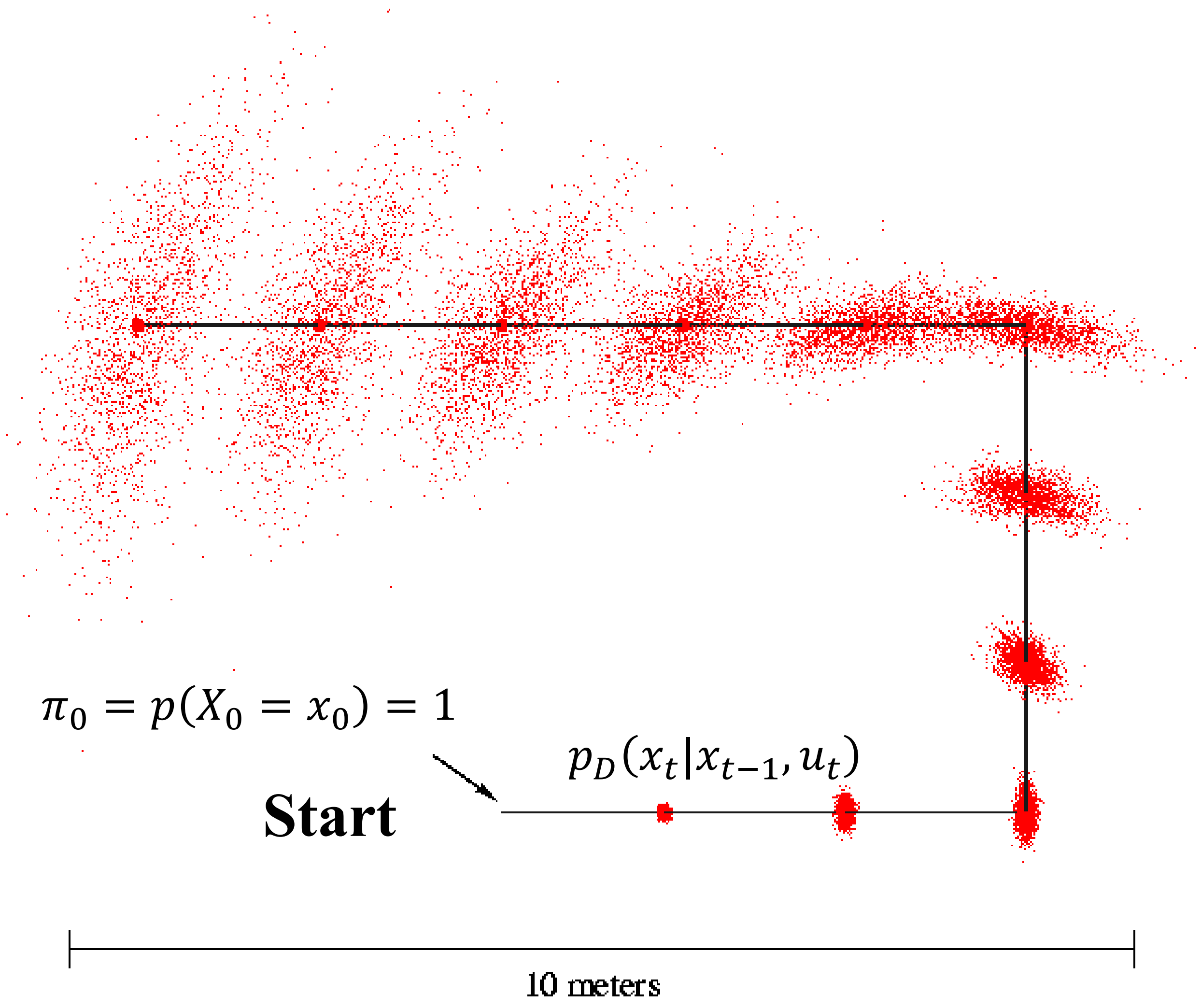
If state is complete, sufficient summary of the history then:

- $p_D(x_t \,|x_{0:t-1}, z_{0:t-1}, u_{0:t-1}) = p_D(x_t \,|x_{t-1}, u_t)$ transition prob.
- $p_D(x'|x, u)$ if transition probabilities are time invariant





Example Motion Model without measurements



The state transition probabilities are defined by
$x_{t+1} = f(x_t, u_t) + \omega_t$

where $\omega_t \sim N(0,1)$

Probabilistic measurements

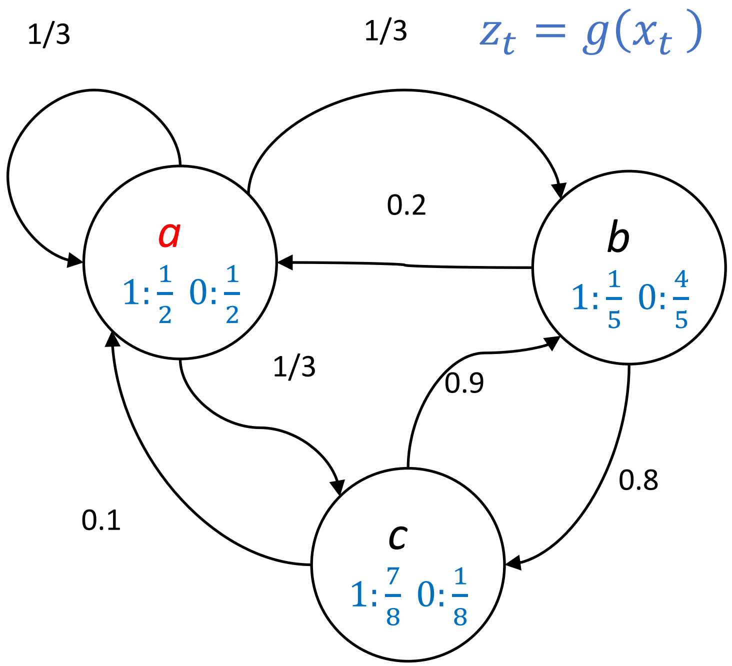
A **measurement model** gives the *output* or observation probability for a given state, e.g.:

$$p_M(z_t = 1 | x_t = a) = \frac{1}{2}$$

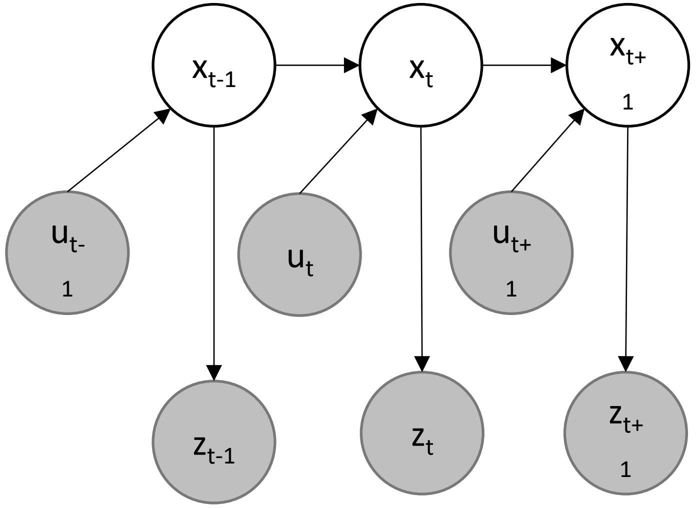
Generally, measurements can depend on history

$$p_M(z_t | x_{0:t}, z_{1:t-1}, u_{0:t-1})$$

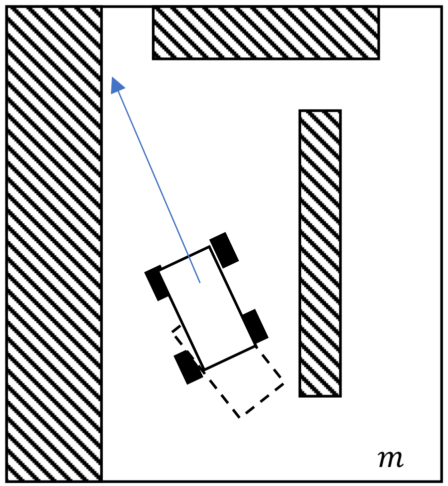
- If state is complete $p_M(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)$
- $p_M(z_t | x_t)$: measurement probability
- $p_M(z | x)$: time invariant measurement probability



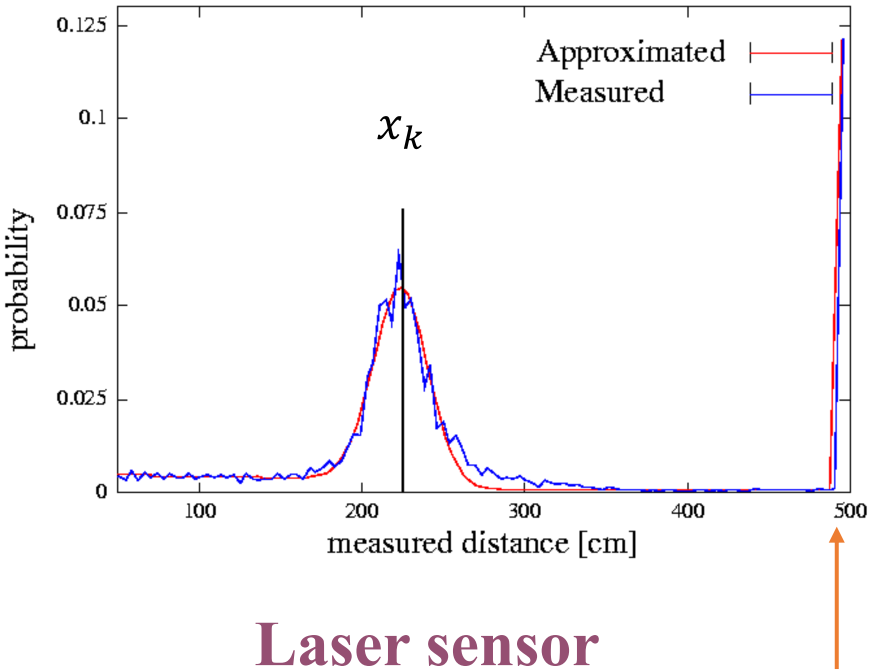
State a produces output 1 and 0 each with probability 0.5



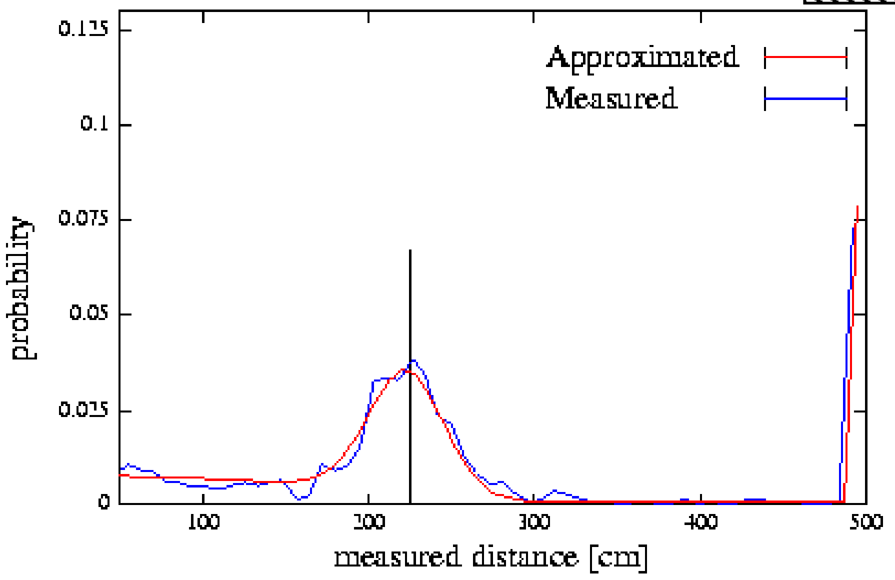
Example Proximity Sensor Measurement Models



$p_M(z_t \mid X_t = x_k)$



Laser sensor



Sonar sensor

Summary so far: Evolution and measurement

$p_D(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$ describes motion/state evolution model

If state is complete, sufficient summary of the history then

- $p_D(x_t \mid x_{0:t-1}, z_{0:t-1}, u_{0:t-1}) = p_D(x_t \mid x_{t-1}, u_t)$ **motion** model
- $p_D(x' \mid x, u)$ if transition probabilities are time invariant

$p_M(z_t \mid x_{0:t}, z_{1:t-1}, u_{0:t-1})$ describes measurement

If state is complete

- $p_M(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$ **measurement** model
- $p_M(z \mid x)$: time invariant measurement probability