15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 5:

Particle Filters and Localization

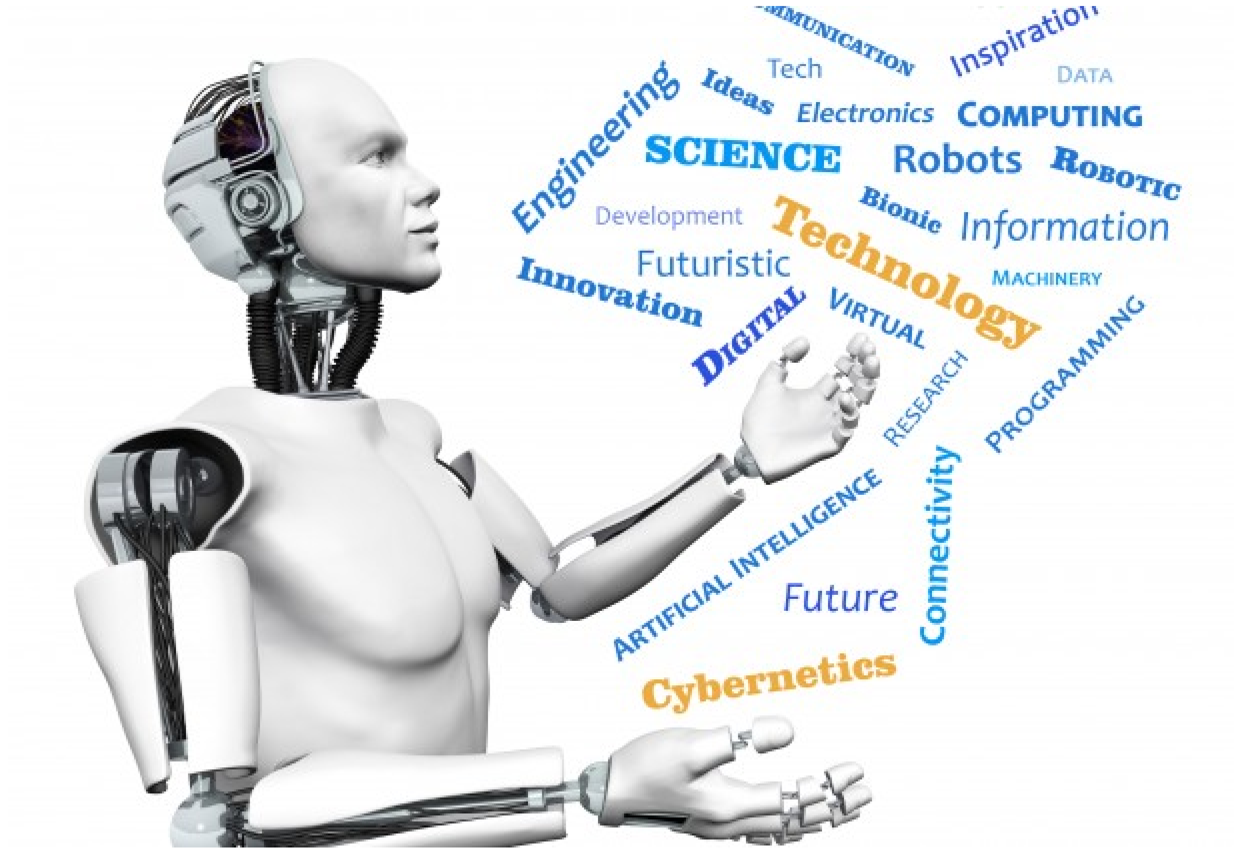


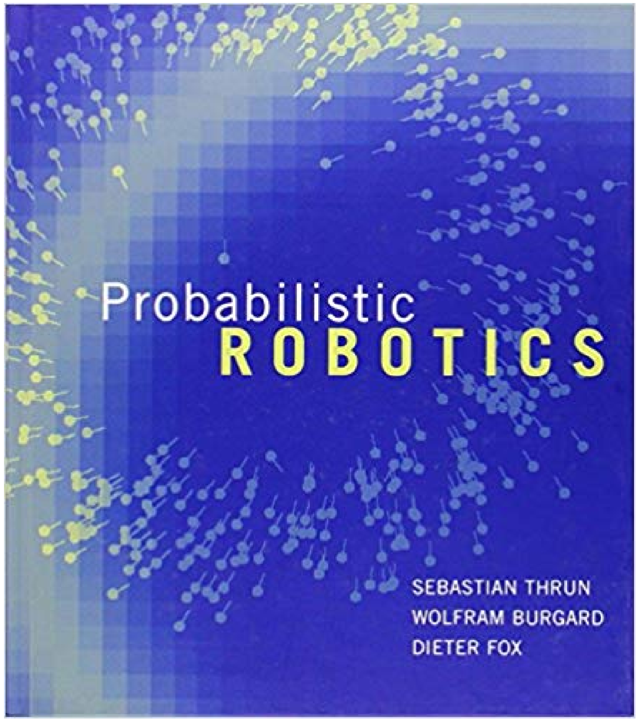
Image from http://www.futuristgerd.com/2015/09/10

Outline

- Probabilistic Robotics
- Belief States
- Parametric and non-parametric representations
- Motion model
- Sensor model
- Evaluation and resampling
- Demos

Probabilistic Robotics

- The world is uncertain:
  - Sensors are noisy and inaccurate.
  - Actuators are unreliable.
  - Other actors can affect the world.

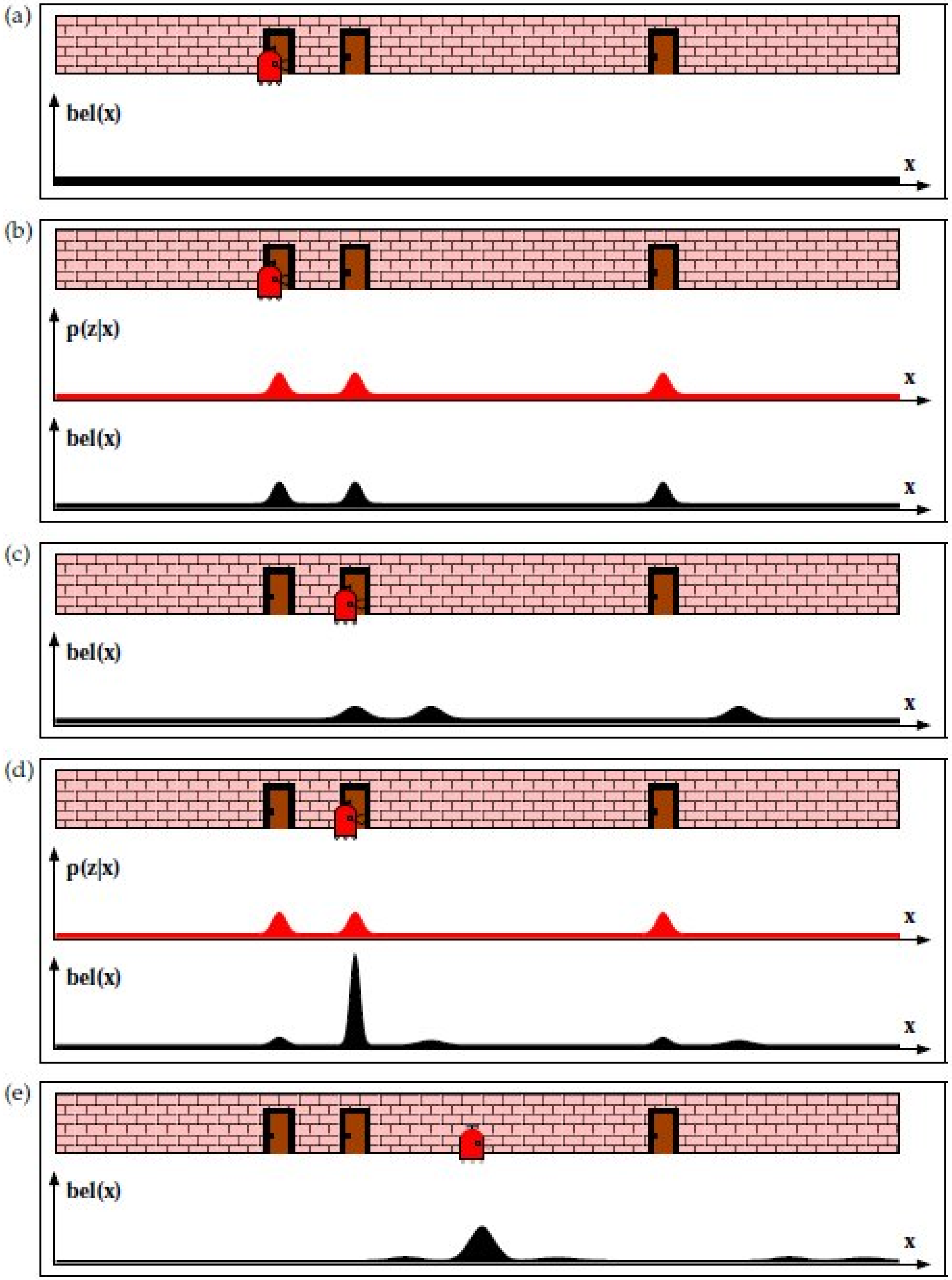


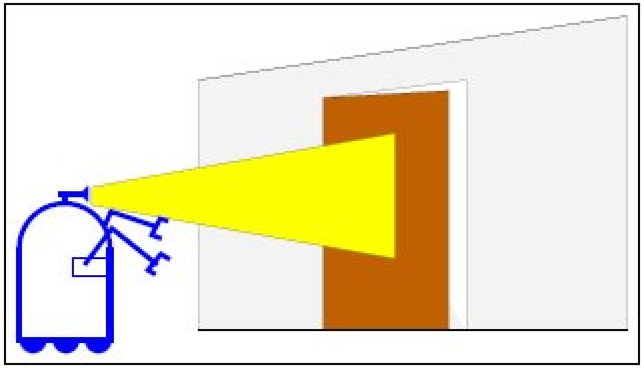
- Embrace the uncertainty!

- How?
  - Explicitly *model* our uncertainty about sensors and actions.
  - Replace discrete states with beliefs: *probability distributions* over states.
  - Use Bayesian filtering to update our beliefs.

Beliefs

are probability distributions





Figures from Thrun, Burgard, and Fox (2005)
Probabilistic Robotics

Some Notation

- x_t = state at time *t*
- *u_t = control signal at time t*
- *z_t = sensor input at time* t
- We don't know x_t with certainty; we have *a priori* (before measurement) beliefs about it:

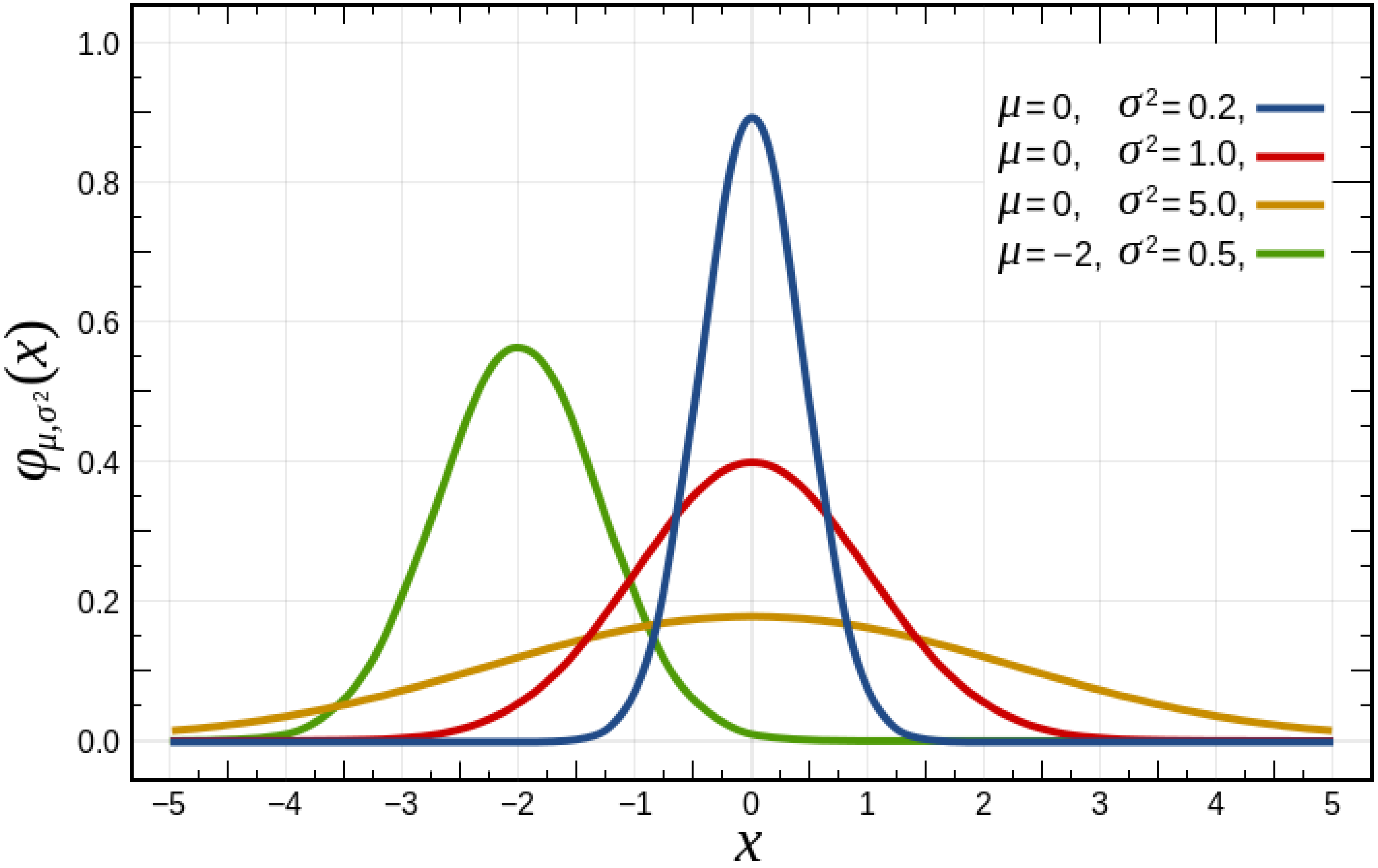
$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t})$$

- New sensor data z_t updates our belief:

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

Parametric Representations (1)

- Represent a probability distribution using an analytic function described by a small number of parameters.
- Most common example: Gaussian



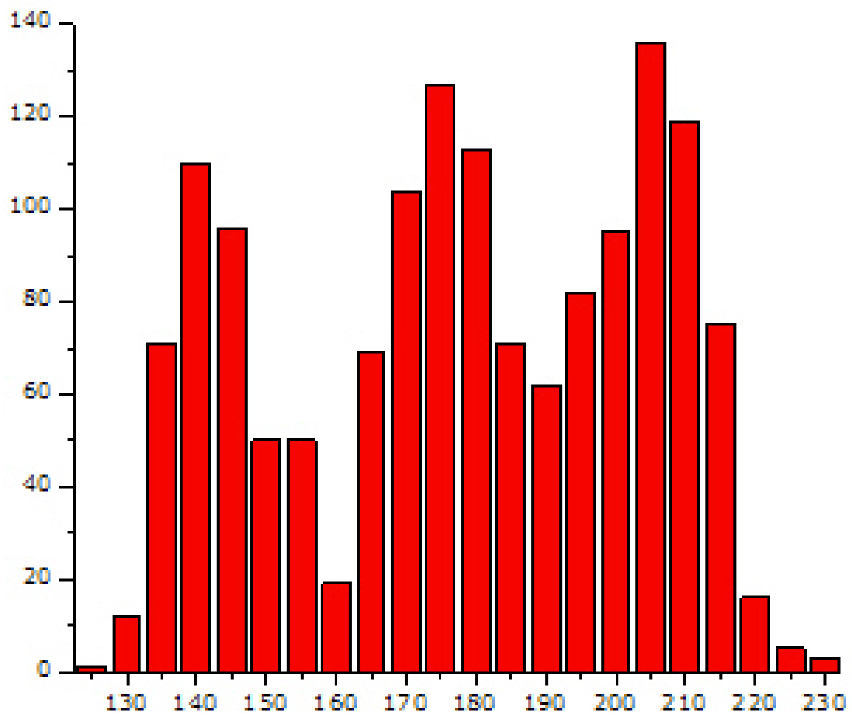
Parametric Representations (2)

- Good points:
  - Compact representation: just a few numbers
    - For a Gaussian: mean μ and variance σ^2
  - Fast to compute
  - Nice mathematical properties
  - **Easy to sample from**

- Drawbacks:
  - May not match the data very well
  - Can give bad results if the fit is poor

Nonparametric Representations

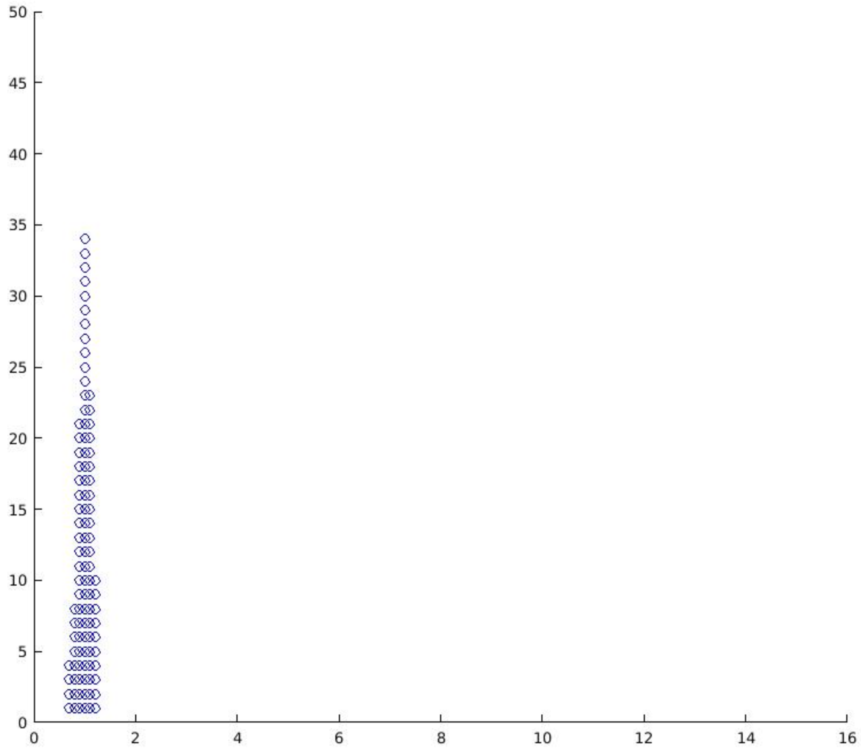
- No preconceived formula for the distribution.
- Instead, maintain a representation of the actual distribution, via *sampling*.
- Example: histogram
- Good points:
  - Can represent completely arbitrary distributions
- Drawbacks:
  - Requires more storage
  - Expensive to update



Where Is The Robot?

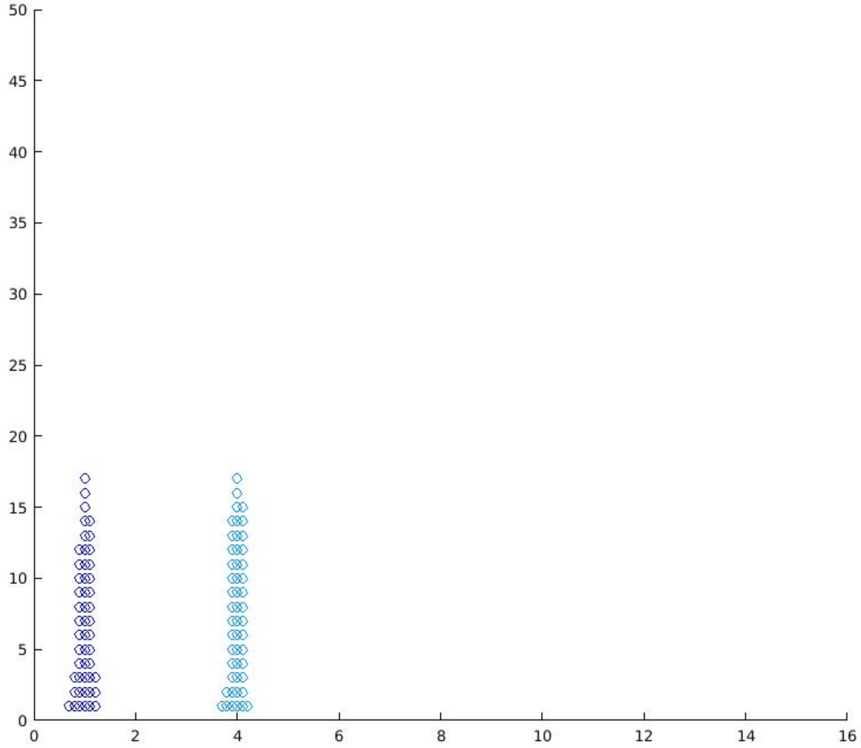
- Parametric: the robot is at x=1 with $\sigma^2 = 0.2$

- Non-parametric: 100 samples indicating robot position.



Where Is The Robot?

- Parametric: fail (or put robot at the mean: x=2.5)
- Non-parametric: 100 samples.



Particle Filters

- A particle filter is an efficient non-parametric representation of a distribution.

- Each particle represents a sample drawn from the distribution.

- As the distribution changes, we update the particles.

- Three kinds of updating:
    - Change the *value* the particle encodes (motion model).
    - Change the *weight* assigned to the particle (sensor model).
    - *Resample* the distribution, getting a fresh set of particles with initially equal weights.

Bayesian Filter, part 1

- Our belief about the robot's position at time *t*-1 is a probability distribution p(x_{t-1}), which we represent as a set of *samples*.

- At time *t* the robot moves, following some control signal u_t, producing a new distribution p(x_t).

- A *motion model* defines how our new prediction $\overline{\text{bel}}(x_t)$ arises from applying u_t.

$$\overline{bel}(x_t) = \int p(x_t | x_{t-1}, u_t) \cdot bel(x_{t-1}) \; dx_{t-1}$$

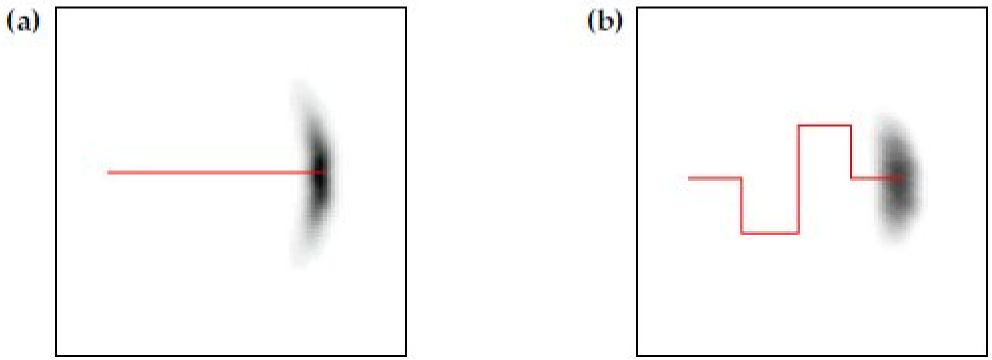
Why Are We Integrating?

$$\overline{bel}(x_t) = \int \underbrace{\underbrace{p(x_t|x_{t-1},u_t)}_{\substack{\text{Probability of}\\ \text{arriving at } x_t \text{ given}\\ \text{that we were}\\ \text{previously at } x_{t-1}\\ \text{and got control}\\ \text{signal } u_t.}} \cdot \underbrace{bel(x_{t-1})}_{\substack{\text{Belief that we}\\ \text{were previously}\\ \text{at location } x_{t-1}}} \; \underbrace{d\,x_{t-1}}_{\substack{\text{All}\\ \text{possible}\\ \text{previous}\\ \text{locations}\\ x_{t-1}}}}_{\text{Integrated over all possible starting locations } x_{t-1}.}$$

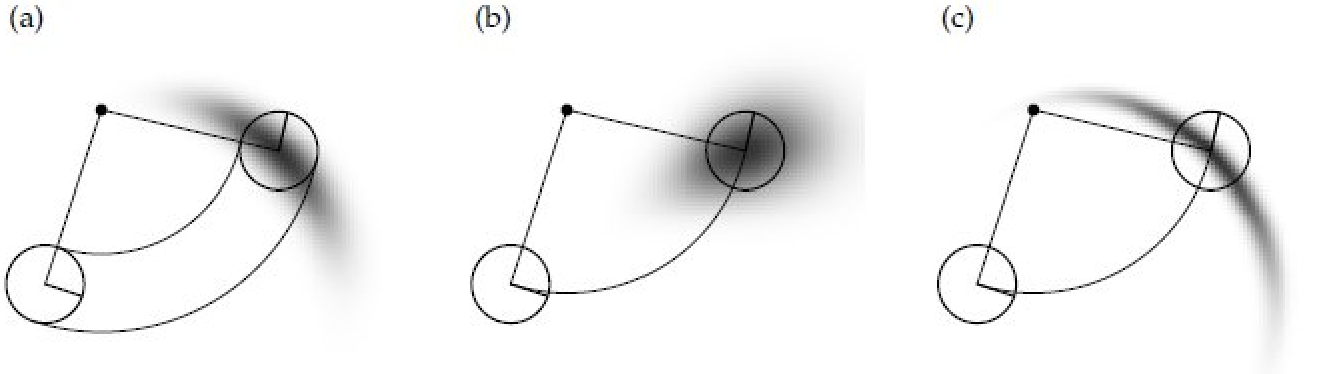
Motion Models

- Motion models express the noisiness of motion u_t.
- Typically use a simple parametric distribution.
  - Easy to sample.
- We represented the distribution $p(x_{t-1})$ as a set of *a posteriori* samples $bel(x_{t-1})$. Motion gives us $\overline{bel}(x_t)$.
- How do we sample $\overline{bel}(x_t)$?
- Solution: for each sample in $bel(x_{t-1})$, draw a value from the motion model's distribution and add it to the sample value.

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



Figures from Thrun, Burgard, and Fox (2005) *Probabilistic Robotics*

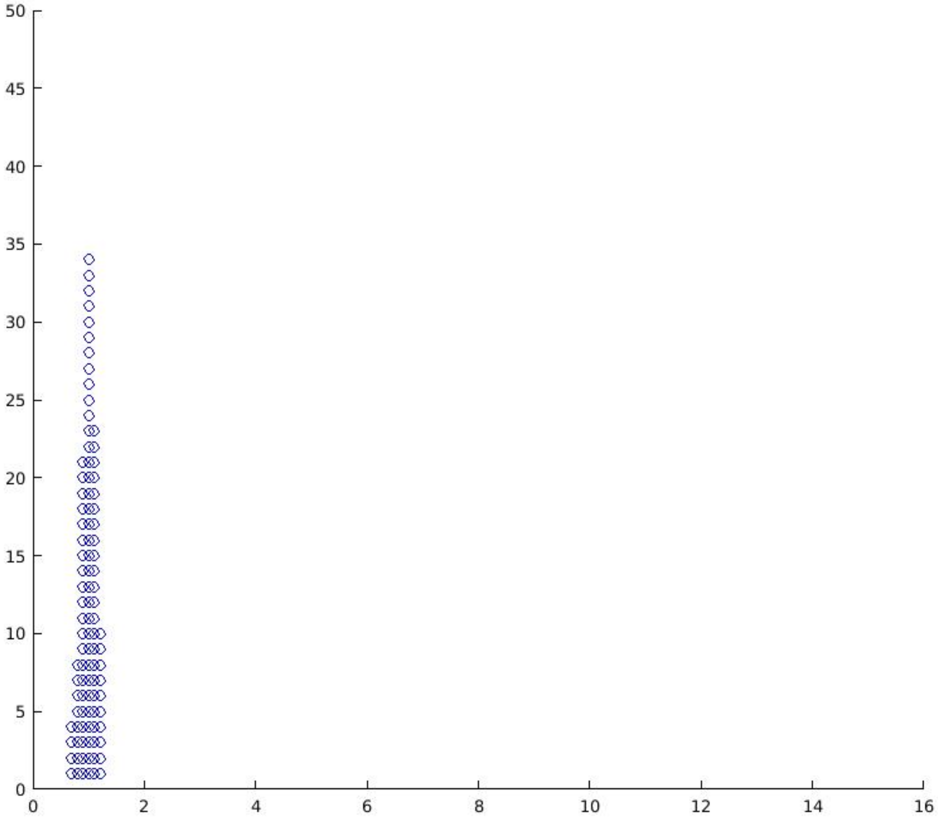


Moderate Noise Values

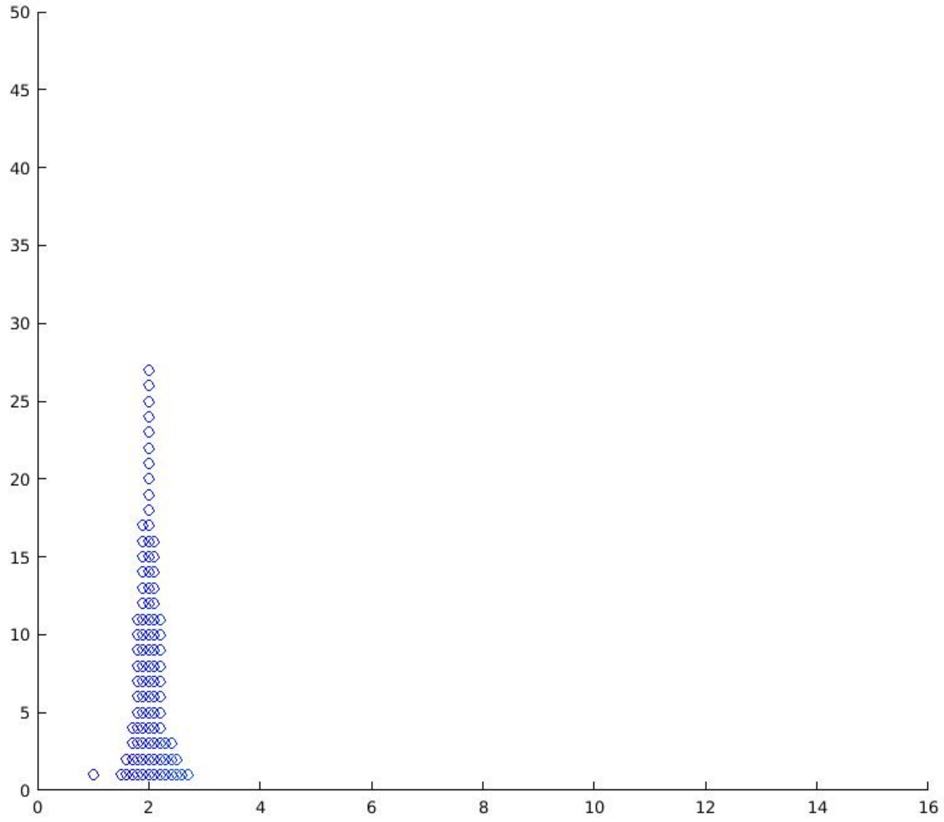
High Translational Uncertainty

High Rotational Unvertainty

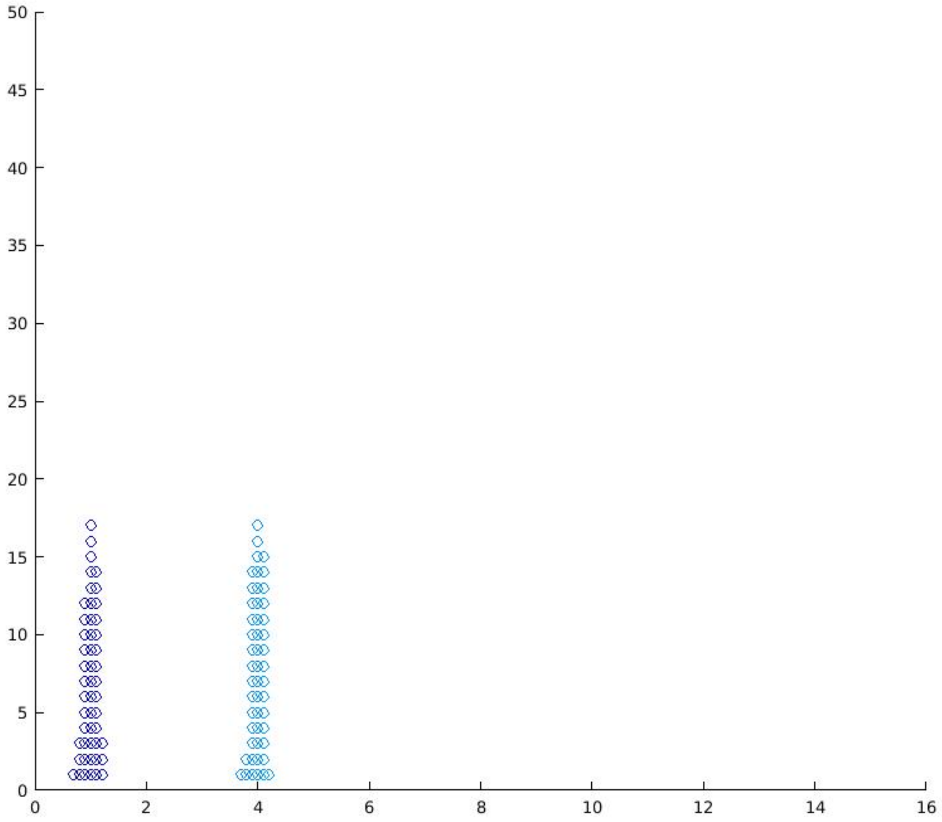
Robot at t=0: $bel(x_0)$



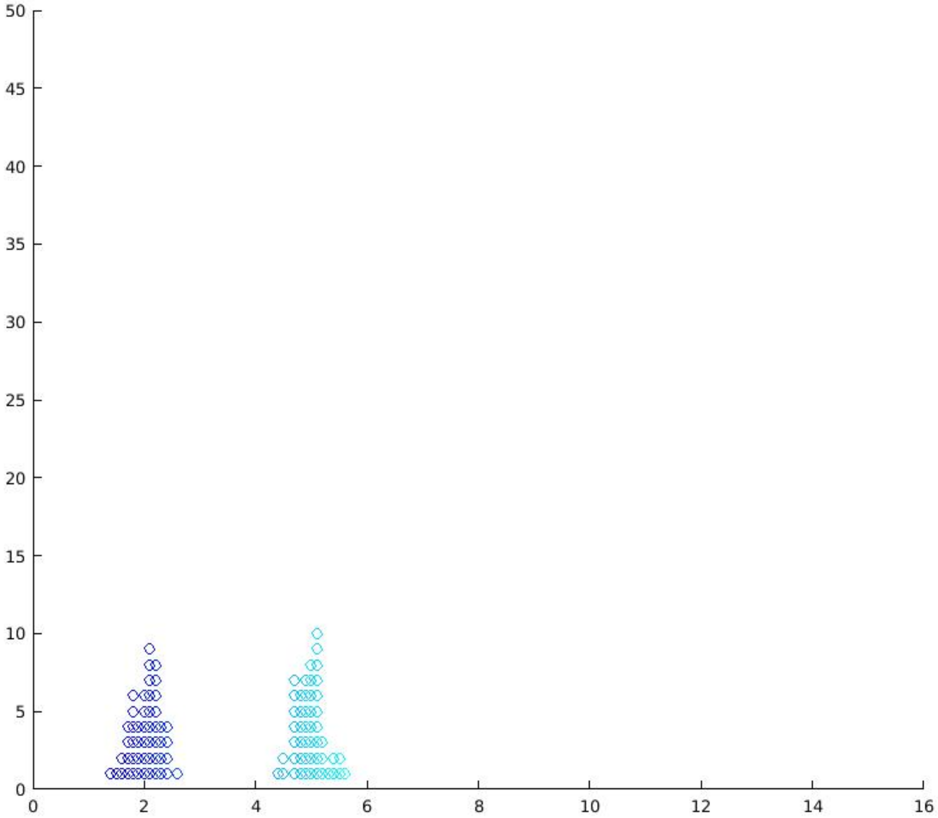
Prediction at t=1: $\overline{\text{bel}}(x_1)$



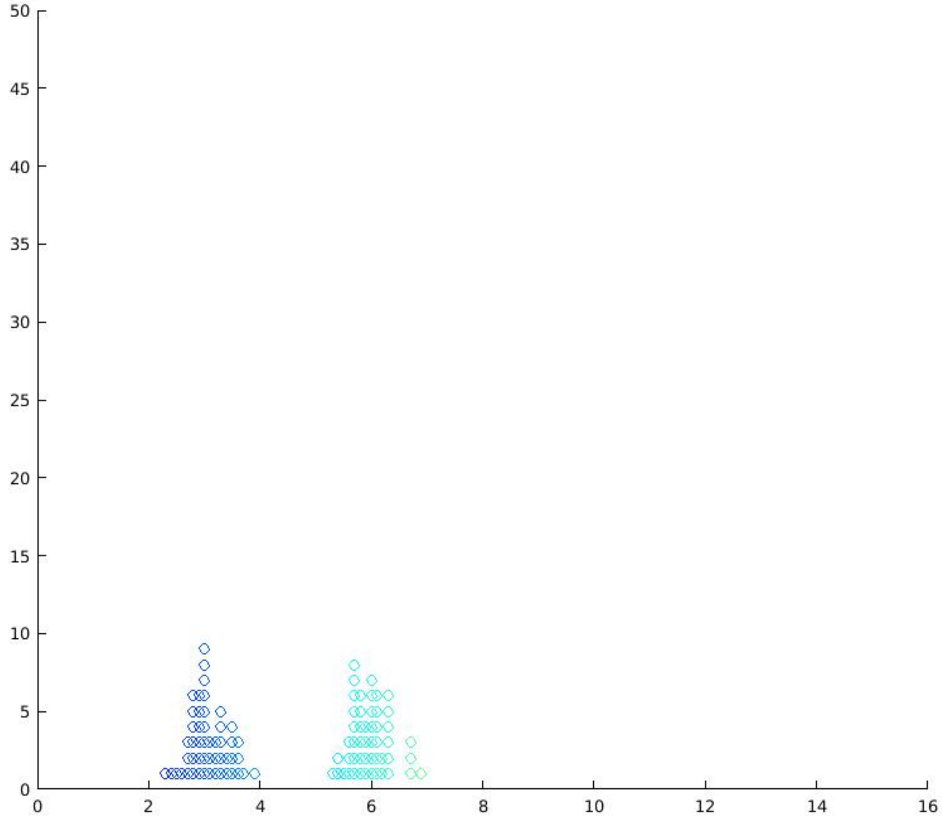
Robot at t=0: $bel(x_0)$



Prediction at t=1: $\overline{\text{bel}}(x_1)$



Prediction at t=2: $\overline{\text{bel}}(x_2)$



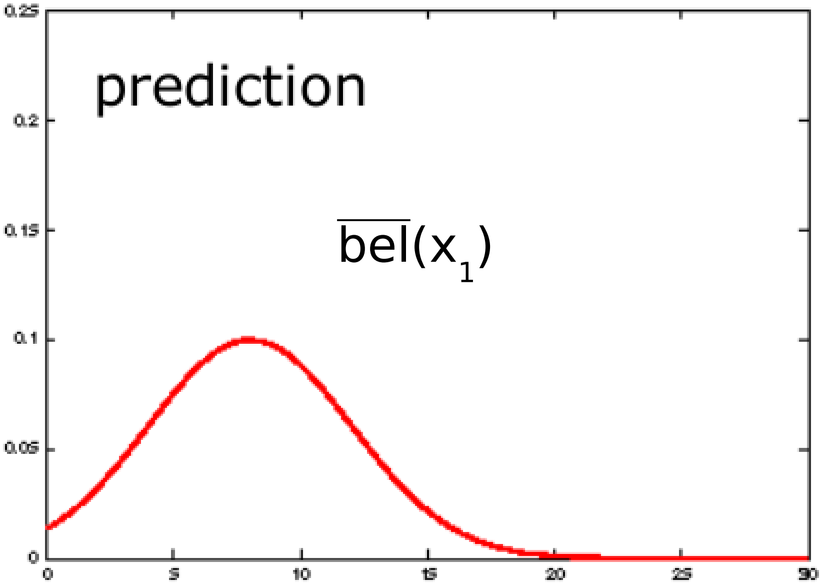
Correcting Our Prediction

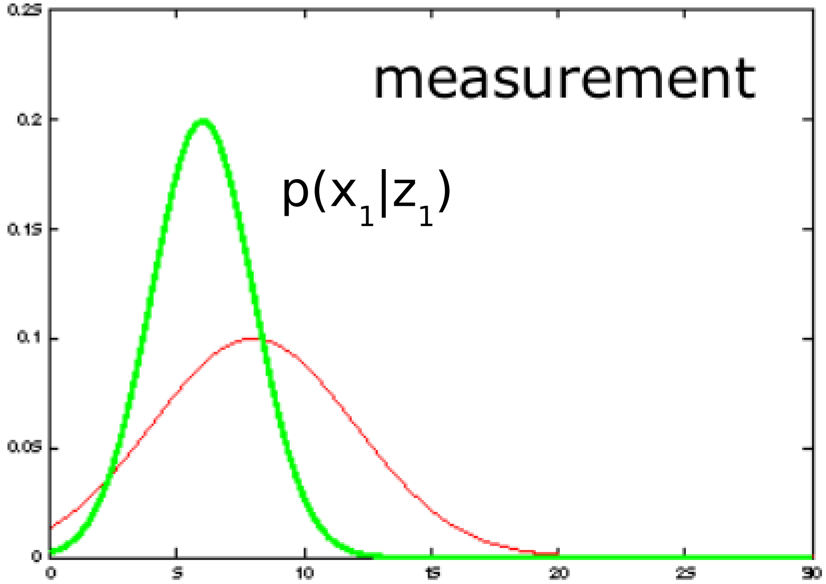
- To mitigate the noisiness of our motion model, we use sensor readings z_t to correct our belief distribution.
- Our sensors give us a probability distribution $p(x_t|z_t)$.
- Can't our sensors just tell us where we are?
- **NO!**
  - They're noisy.
  - An individual reading may not be that informative because the world can be ambiguous (e.g., doors look alike).
  - Need to combine information.

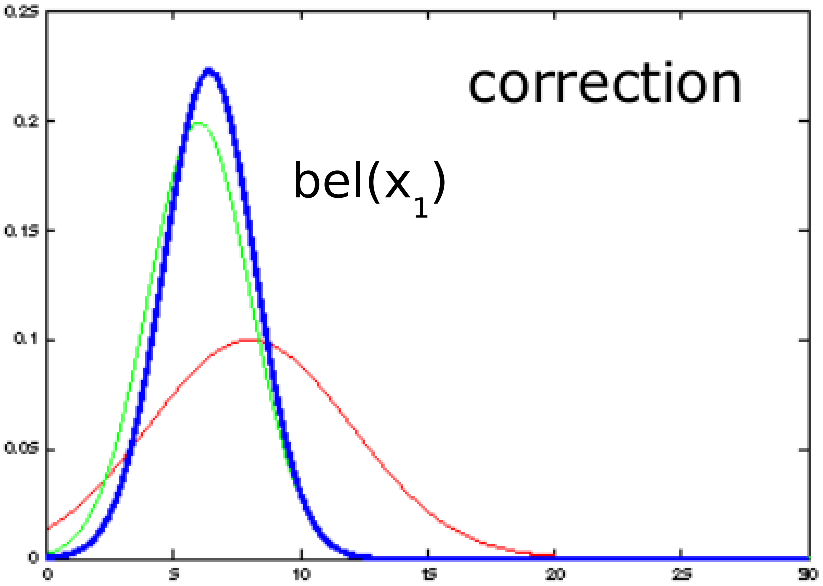
Sensor Model

- We should try to model uncertainty in our sensor data.
- Lots of work on sonar and laser rangefinder noise models (e.g., effects of reflections, viewing angle, etc.)
- For visual landmarks:
  - Effects of camera resolution.
  - Distance estimates might have variance proportional to the mean.
  - Bearing estimates might have variance inversely proportional to distance.

If distributions are gaussians, we can combine them using a **Kalman filter**. Weighting is inversely proportional to variance.



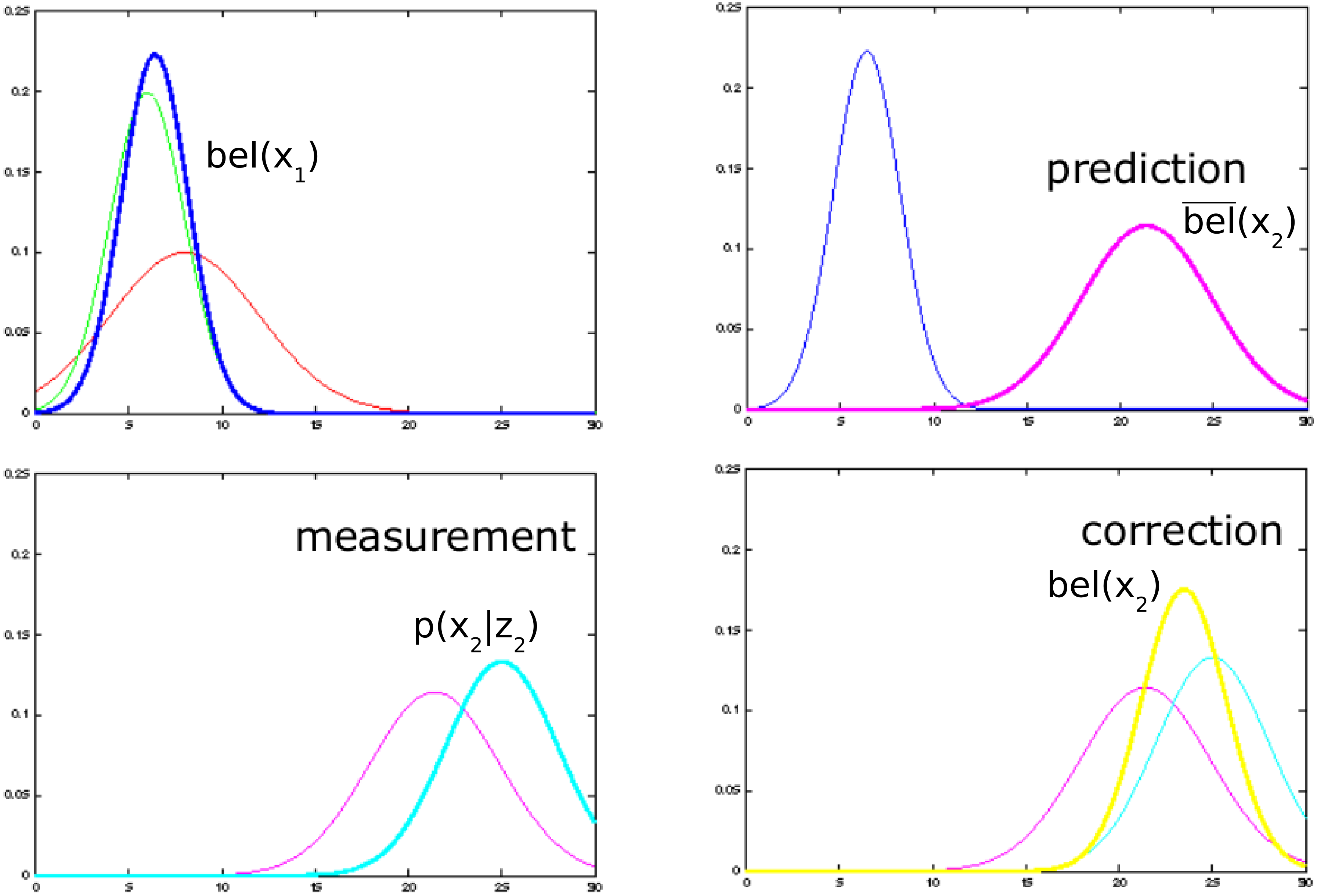




It's a weighted mean!

Slide modified from Burgard et al., “Introduction to Mobile Robotics”, 2014, lecture 9: “Bayes Filter – Kalman Filter”.

Second iteration: prior belief → prediction → measurement → correction.



Slide modified from Burgard et al., “Introduction to Mobile Robotics”, 2014, lecture 9: “Bayes Filter – Kalman Filter”.

Bayesian Filter, part 2

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) \cdot bel(x_{t-1})\ dx_{t-1}$$

Sensor reading z_t gives distribution $p(x_t|z_t)$.

Corrected: $bel(x_t) = \eta\, p(z_t|x_t) \cdot \overline{bel}(x_t)$

η is a normalization constant.

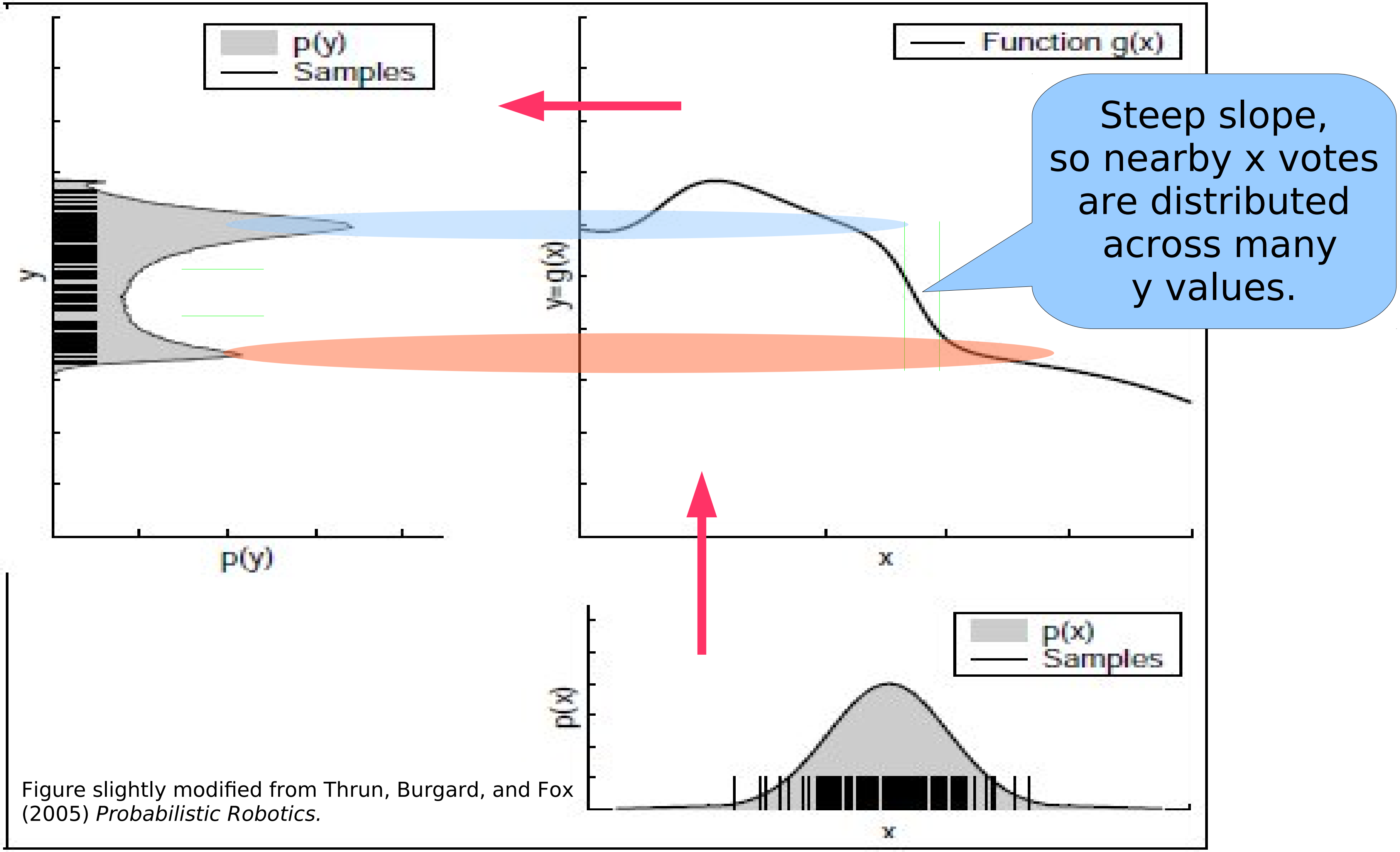
Corrected Sampling Representation

- Distribution $\overline{\text{bel}}(x_t)$ is “corrected” by weight $p(z_t|x_t)$ to give $\text{bel}(x_t)$.
- The *weighted* particles are a sampling representation of the new distribution $p(x_t)$.
- The robot can move around and we can move the particles and update their weights.
- But is this a <u>good</u> representation?
- Particles whose weights become low aren't representing useful hypotheses. Eventually the representation falls apart because we're sampling the wrong regions.

Resampling

- Things break down when too many particles are representing the wrong regions of bel(x_t), so their weights are low.
- We can fix this by resampling bel(x_t), giving a fresh set of particles distributed correctly.
- But we have no formula for bel(x_t), and no direct representation of it.
- So how do we sample from it? *Importance sampling.*

Sampling y=g(x) From An Arbitrary Distribution x



Steep slope, so nearby x votes are distributed across many y values.

Figure slightly modified from Thrun, Burgard, and Fox (2005) *Probabilistic Robotics.*

Importance Sampling

- Want to sample from f.
- Can only sample from g.
- Weight each sample by f(x) / g(x).
- The weighted samples approximate f.
- g is $\overline{bel}(x_t)$
- Weighting comes from $p(z_t|x_t)$
- Draw from the weighted sample.

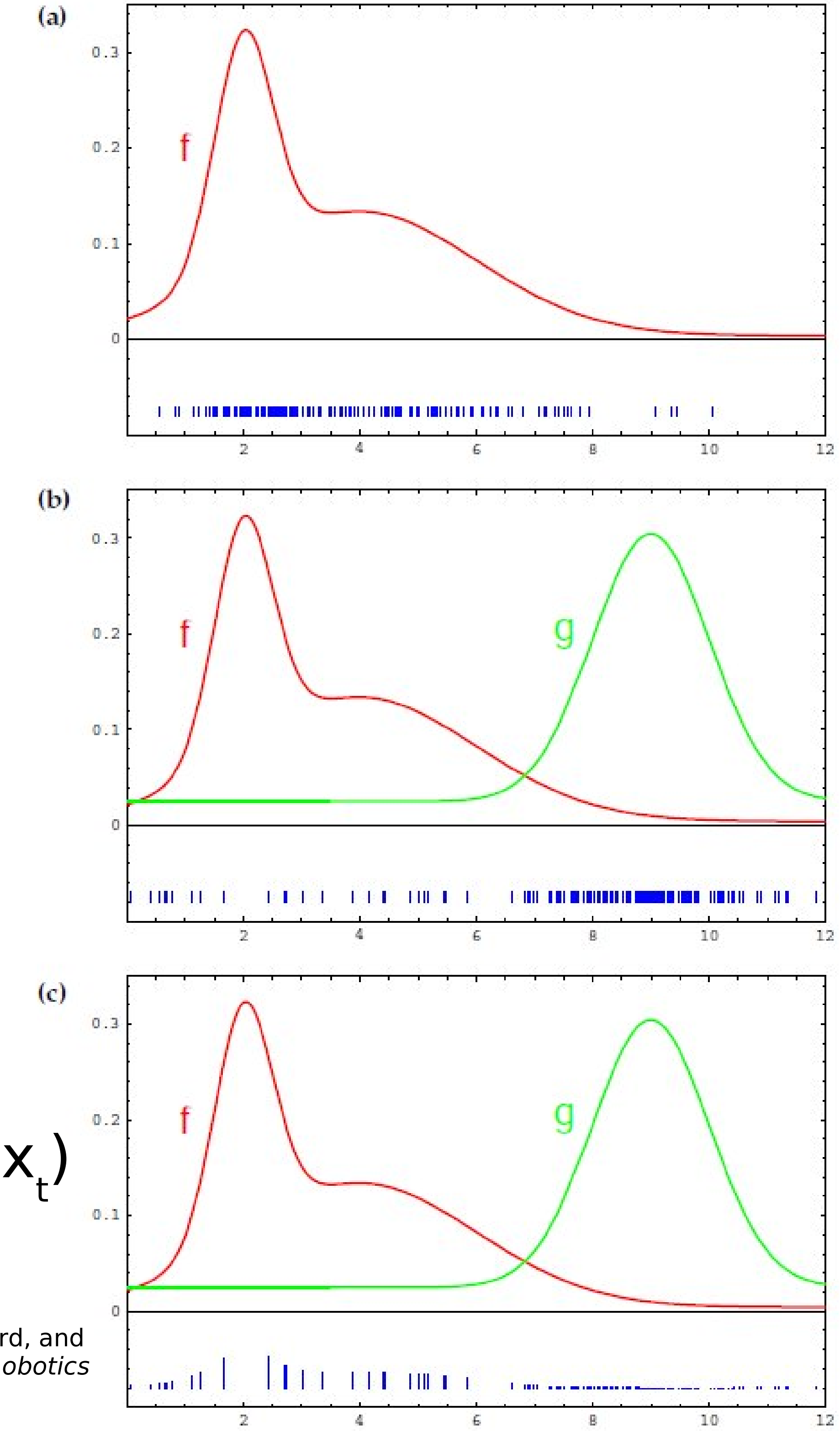


Figure from Thrun, Burgard, and Fox (2005) *Probabilistic Robotics*

Resampling

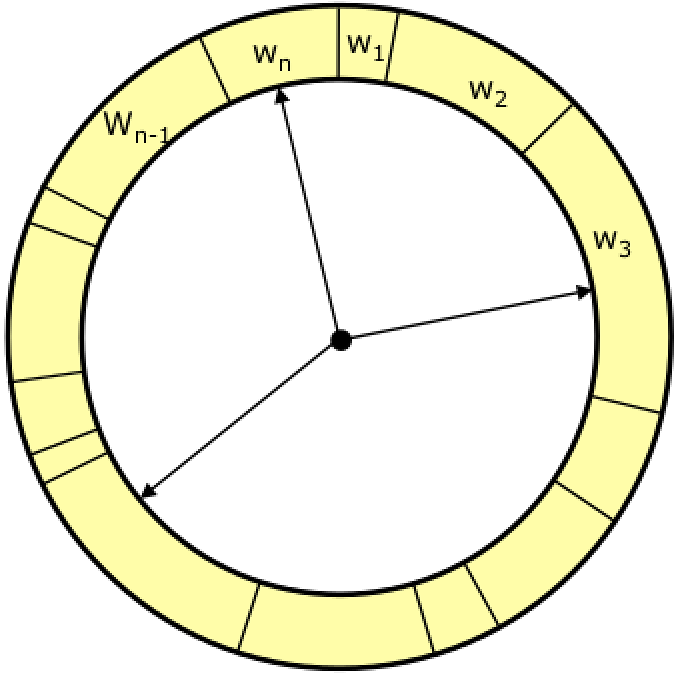
- We don't need to resample on every time step t.
- We can accumulate sensor data for several time steps, so our weights are more accurate. We can use the weights to estimate the robot's location (if unimodal).

$$\hat{x}(t) = \sum_i w_t^{(i)} \cdot x_t^{(i)}$$

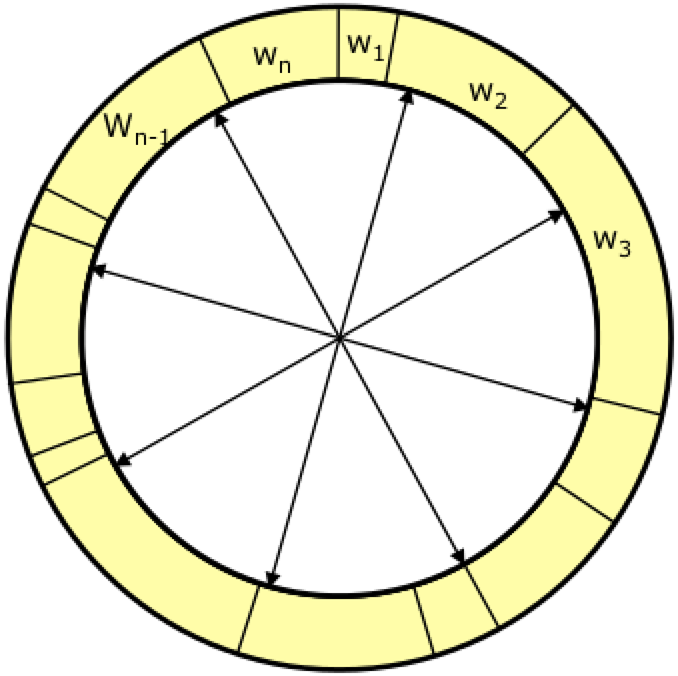
- When to resample?
  - If the variance on the weights is high, then many particles are representing non-useful portions of the space.
  - Resampling redistributes the particles so they are concentrated where the probability density is highest.

How To Resample

- Stochastic universal sampling is a trick for drawing samples from a weighted distribution as fairly as possible (low variance).



3 samples



8 samples

Image from Burgard et al., “Introduction to Mobile Robotics”, 2014, lecture 12: “Bayes Filter – Particle Filter and Monte Carlo Localization”.

Weighting in a Corridor

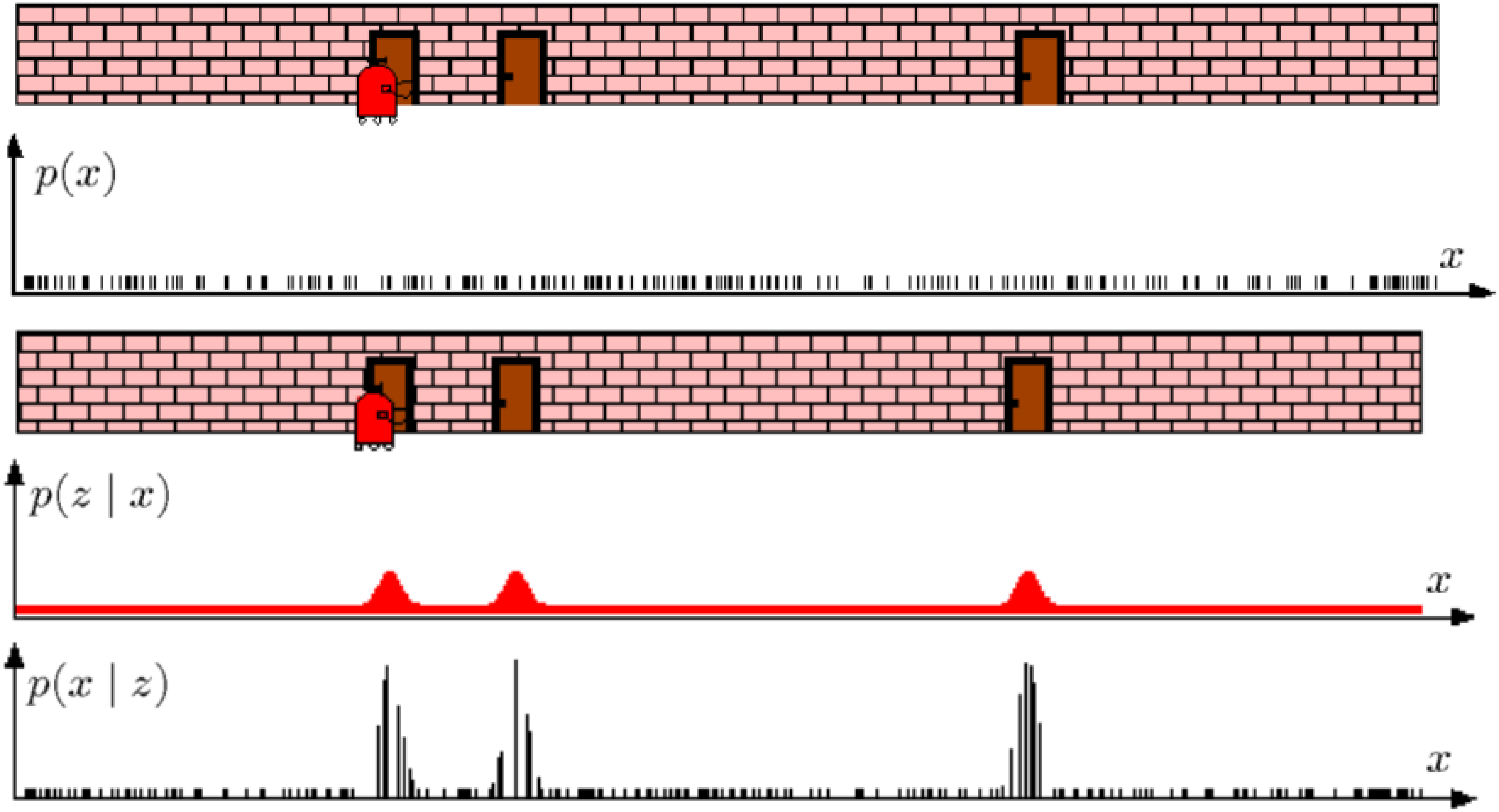


Image from Burgard et al., “Introduction to Mobile Robotics”, 2014,
lecture 12: “Bayes Filter – Particle Filter and Monte Carlo Localization”.

Resampling and Motion

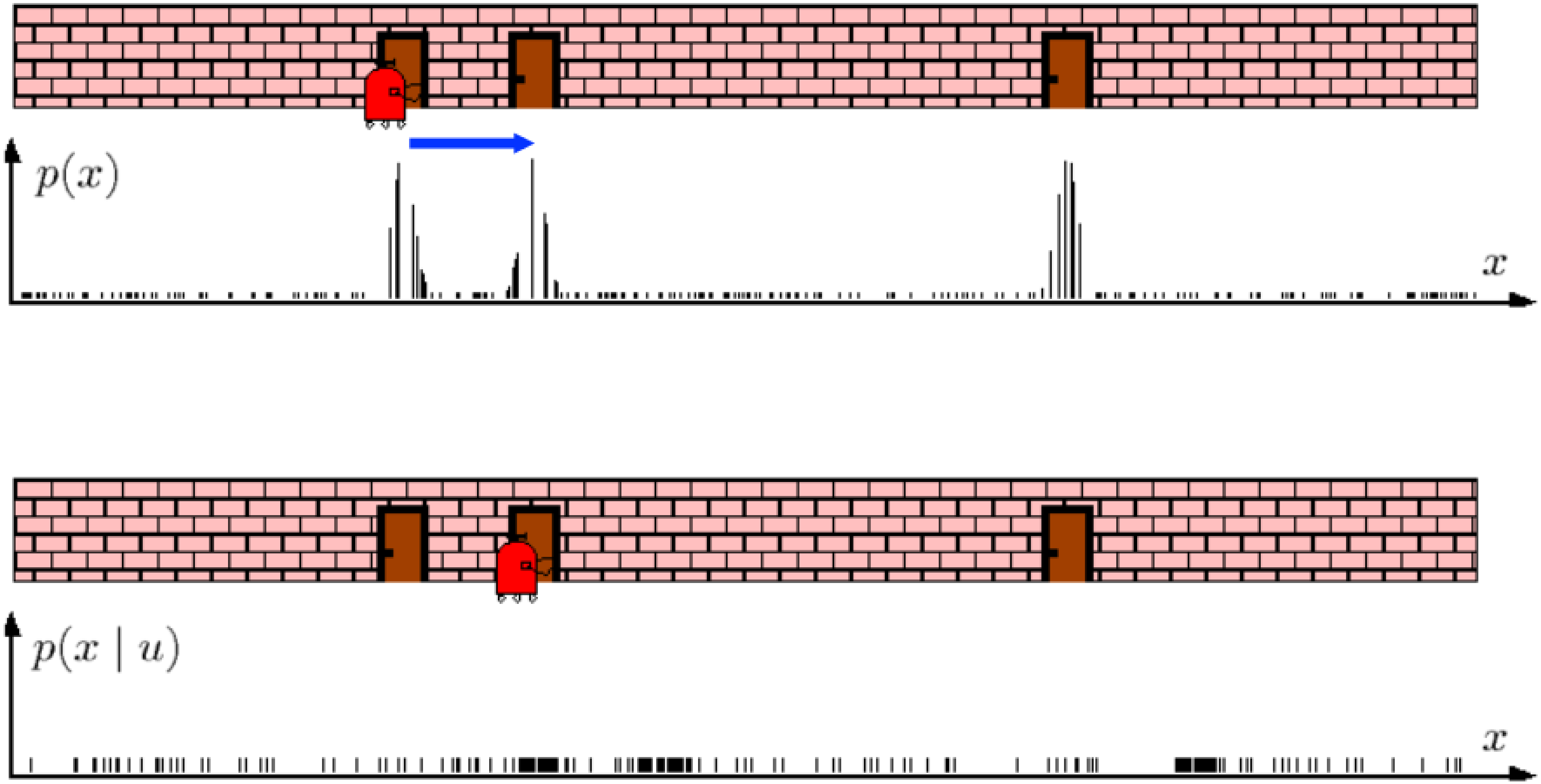


Image from Burgard et al., “Introduction to Mobile Robotics”, 2014,
lecture 12: “Bayes Filter – Particle Filter and Monte Carlo Localization”.

Sensing and Weighting

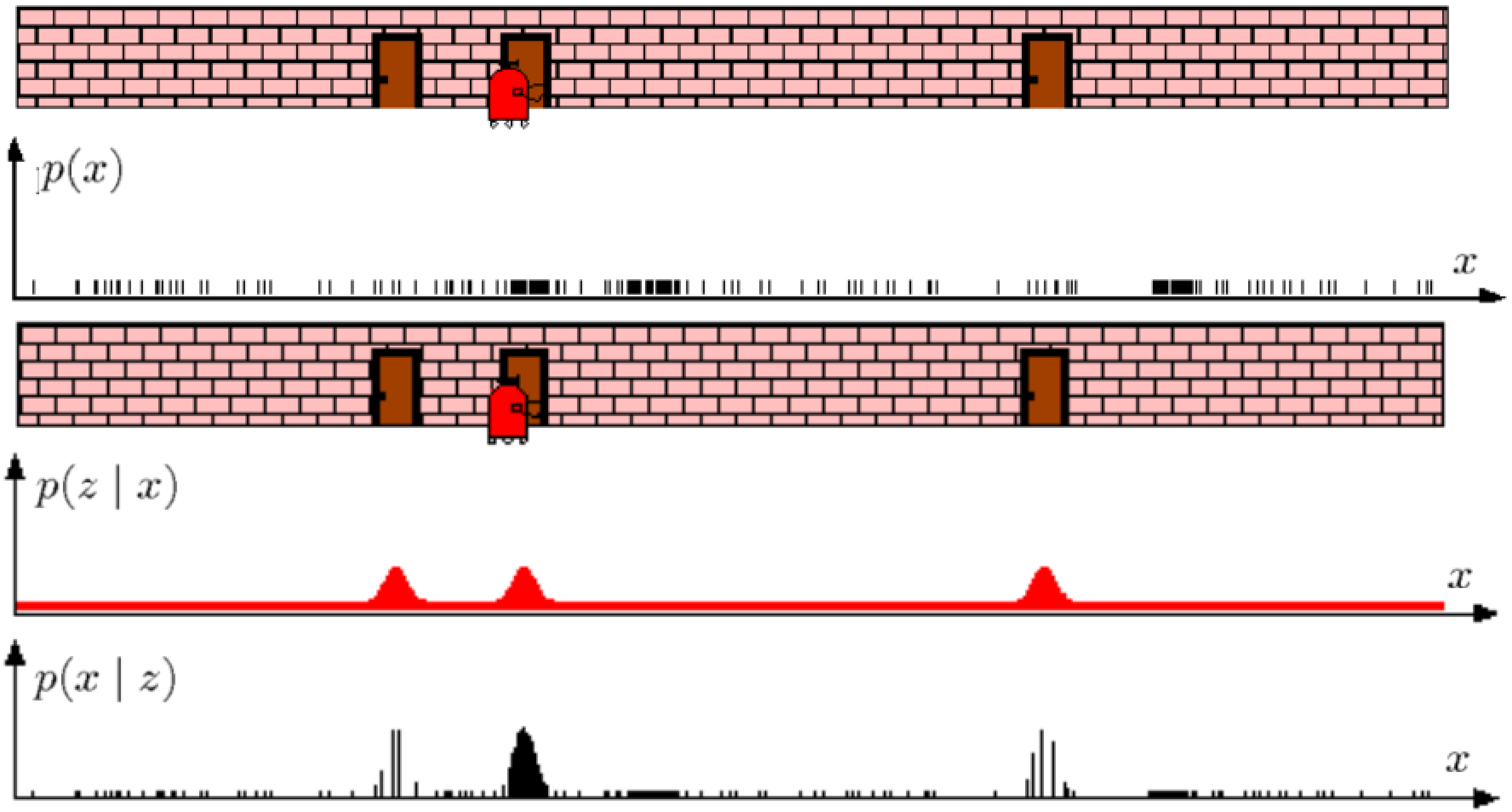


Image from Burgard et al., “Introduction to Mobile Robotics”, 2014,
lecture 12: “Bayes Filter – Particle Filter and Monte Carlo Localization”.

Resampling and Motion

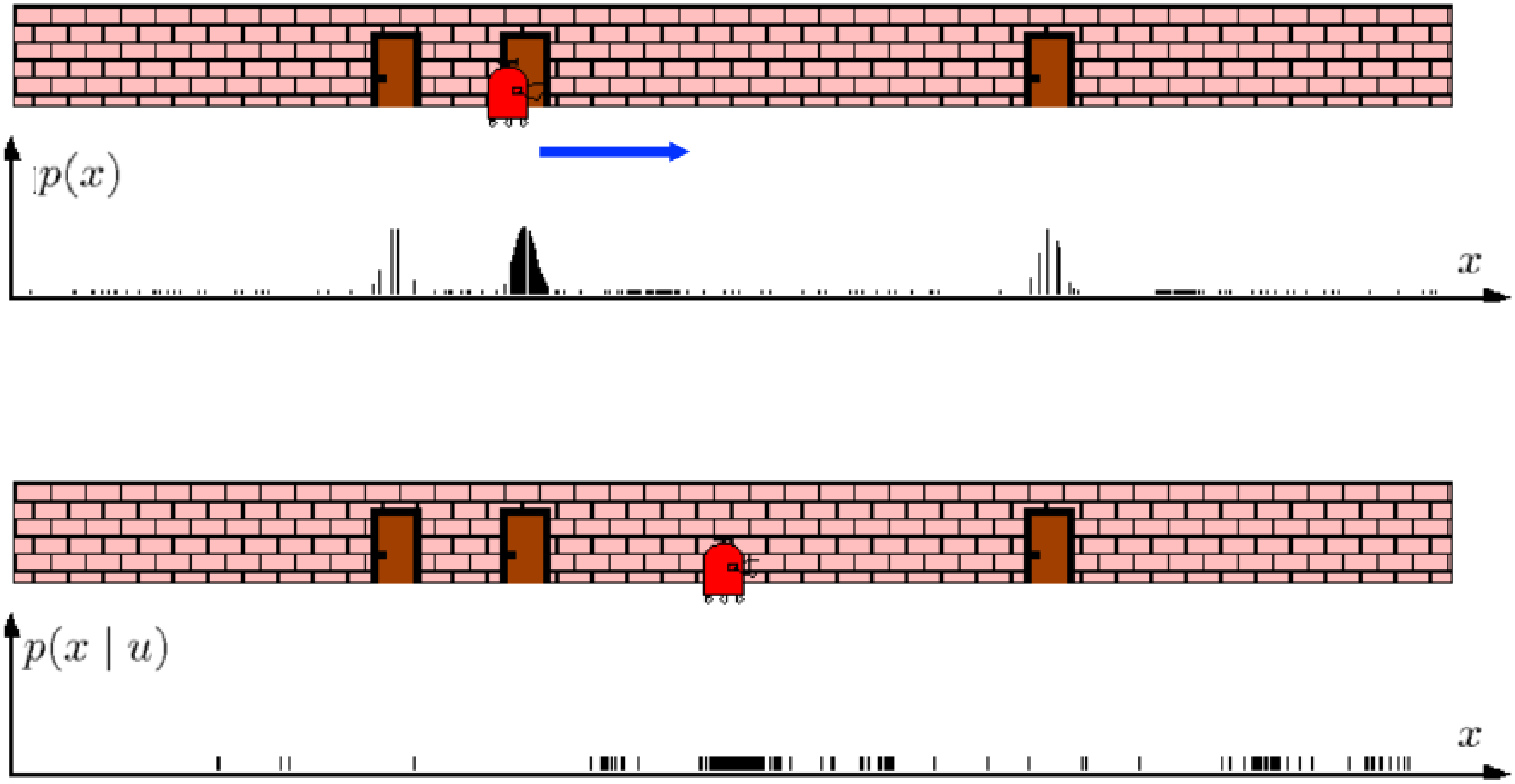


Image from Burgard et al., “Introduction to Mobile Robotics”, 2014,
lecture 12: “Bayes Filter – Particle Filter and Monte Carlo Localization”.

Summary

- Particle filters are the preferred method for robot localization in the real world.
- Robot pose typically encoded as (x,y,θ).
- A map is needed to define how sensor values indicate locations. But what if we don't have a map?
- SLAM: Simultaneous Localization and Mapping.
- Particles can be used to represent hypotheses about the map as well as about the robot's location.