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 reasoning to update our beliefs.
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 beliefs about it:

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

- New sensor data updates our belief:

  $\text{bel}(x_t) = p(z_t \mid x_t) \cdot \overline{\text{bel}}(x_t)$
Beliefs
are probability distributions

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

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

Figure from Thrun, Burgard, and Fox (2005) *Probabilistic Robotics*
Parametric Representations (2)

• Good points:
  - Compact representation: just a few numbers
    • For a Gaussian: mean $\mu$ and variance $\sigma^2$
  - Fast to compute
  - Nice mathematical properties

• 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 arbitrary distributions

• Drawbacks:
  – Requires more storage
  – Expensive to update
Particle Filters

- A particle filter is a non-parametric representation of a probability distribution based on sampling.
- Each particle is a sample.
- As the distribution shifts due to new information, we resample it.

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

- We can use a particle filter to represent the distribution of hypotheses about the robot's pose (location and orientation).

- Two types of updates: motion, and sensor readings.

- Self-motion information (odometry) $u_t$:
  - Noisy: describe the noise using a motion model.
  - Drag the particles along.

- Sensor information (landmarks) $z_t$:
  - Noisy: describe the noise using a sensor model.
  - Weight the particles based on their sensor predictions.
  - Resample based on the weighting in order to approximate the new distribution $p(z_t|x_t) \cdot p(x_t|x_{t-1}, u_t)$. 

Tekkotsu Particle Filter Demo
Initial World Map
Randomize the Particles
Sensor Input
Localize
Move 3 Meters to the East
Relocalize (Cheat)
Motion Model $p(x_t | x_{t-1}, u_t)$

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

- Try to model uncertainty in sensor data.
- Lots of work on rangefinder noise models.
- For visual landmarks:
  - Distance estimates might have variance proportional to the mean.
  - Bearing estimates might have variance inversely proportional to distance.
- Tekkotsu doesn't currently implement this.
Resampling

• Resampling generates a new set of particles.
• The alternative is to keep adjusting the weights on the existing set.
• 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.
• Problem: we want to sample $\text{bel}(x_t)$ but we have no representation for it. We have $\overline{\text{bel}}(x_t)$ and $p(z_t|x_t)$.
• Solution: importance sampling.
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 $\text{bel}(x_t)$
- Weighting comes from $p(z_t|x_t)$

Figure from Thrun, Burgard, and Fox (2005) *Probabilistic Robotics*
Tekkotsu's Particle Filter

• Generic particle filter: templated class.
  Shared/ParticleFilter.h

• For localization:
  Localization/ShapeBasedParticleFilter.h
  Localization/LocalizationParticle.h
  Localization/CreateMotionModel.h
  Localization/ShapeSensorModel.h
Demos

• PilotDemo allows you to experiment with the particle filter. Commands:
  - rand: randomize the particles
  - loc: localize
  - disp $n$: display $n$ particles

• Particle Filter Bingo (coming soon)
  - Trace the weighting of particles as sensor data comes in.