Why build a map?

• Time!
  – Playing field is big, robot is slow
  – Driving around perimeter takes a minute!
  – Scoring takes time… often ~20 seconds to “line up” to a mouse hole.

• Exploration round gives advantage to robots that can map
Attack Plan

- **Motivation:** why build a map?
- Terminology, basic concepts
- Mapping approaches
  - Metrical
    - State Estimation
    - Occupancy Grids
  - Topological
- Data Association
- Hints and Tips
What is a feature?

- An object/structure in the environment that we will represent in our map
- Something that we can observe multiple times, from different locations
What is an Observation?

- Where do we get observations from?
  - Camera
    - Range/bearing to ticks and landmarks
  - Corners detected from camera, range finders
- For now, let’s assume we get these observations plus some noise estimate.
Data Association

The problem of recognizing that an object you see now is the same one you saw before

- Hard for simple features (points, lines)
- Easy for “high-fidelity” features (barcodes, bunker hill monuments)

With perfect data association, most mapping problems become “easy”
Attack Plan

- Motivation: why build a map?
- Terminology, basic concepts
- Mapping approaches
  - **Metrical**
    - *State Estimation*
    - Occupancy Grids
  - Topological
- Data Association
- Hints and Tips
Metrical Maps

- Try to estimate actual locations of features and robot
  - “The robot is at (5,3) and feature 1 is at (2,2)”
    - Both “occupancy grid” and discrete feature approaches.

- Relatively hard to build
- Much more complete representation of the world
Metrical Maps

- **State Estimation**
  - Estimate discrete quantities: “If we fit a line to the wall, what are its parameters y=mx+b?”
  - Often use probabilistic machinery, Kalman filters

- **Occupancy Grid**
  - Discretize the world. “I don’t know what a wall is, but grids 43, 44, and 45 are impassable.”
Bayesian Estimation

- Represent unknowns with probability densities
  - Often, we assume the densities are Gaussian
    \[ P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{\sigma^2}} \]
  - Or we represent arbitrary densities with particles
    - We won’t cover this today
Bayesian Data Fusion

• Example: Estimating where Jill is standing:
  
  – Alice says: $x=2$
    • We think $\sigma^2 = 2$; she wears thick glasses
  
  – Bob says: $x=0$
    • We think $\sigma^2 = 1$; he’s pretty reliable

• How do we combine these measurements?
Simple Kalman Filter

• Answer: algebra (and a little calculus)!
  – Compute mean by finding maxima of the log probability of the product $P_A^*P_B^*$.
  – Variance is messy; consider case when $P_A=P_B=N(0,1)$

• Try deriving these equations at home!

\[
\frac{1}{\sigma^2} = \frac{1}{\sigma_A^2} + \frac{1}{\sigma_B^2}
\]

\[
\mu = \frac{\mu_A \sigma_B^2 + \mu_B \sigma_A^2}{\sigma_A^2 + \sigma_B^2}
\]
Kalman Filter Example

- We now think Jill is at:
  - $x = 0.66$
  - $\sigma^2 = 0.66$
Kalman Filter: Properties

- You incorporate sensor observations one at a time.
- Each successive observation is the same amount of work (in terms of CPU).
- Yet, the final estimate is the global optimal solution.

The Kalman Filter is an optimal, recursive estimator.
Kalman Filter: Properties

- Observations *always* reduce the uncertainty.
Kalman Filter

- Now Jill steps forward one step
- We think one of Jill’s steps is about 1 meter, \( \sigma^2 = 0.5 \)
- We estimate her position:
  \[
  x = x_{\text{before}} + x_{\text{change}}
  \]
  \[
  \sigma^2 = \sigma_{\text{before}}^2 + \sigma_{\text{change}}^2
  \]
- Uncertainty \textit{increases}
State Vector

- We’re going to estimate robot location and orientation \((x_r, x_y, x_t)\), and feature locations \((f_{nx}, f_{ny})\).

\[
x = [ x_r \ x_y \ x_t \ f_{1x} \ f_{1y} \ f_{2x} \ f_{2y} \ \ldots \ f_{nx} \ f_{ny} ]^T
\]

- We *could* try to estimate each of these variables independently
  - But they’re correlated!
State Correlation/Covariance

- We observe features relative to the robot’s current position
  - Therefore, feature location estimates covary (or correlate) with robot pose.

- Why do we care?
  - We need to track covariance so we can correctly propagate new information:
  - Re-observing one feature gives us information about robot position, and therefore also all other features.
Correlation/Covariance

- In multidimensional Gaussian problems, equal-probability contours are ellipsoids.

- Shoe size doesn’t affect grades:
  \[ P(\text{grade}, \text{shoesize}) = P(\text{grade})P(\text{shoesize}) \]

- Studying helps grades:
  \[ P(\text{grade}, \text{studytime}) \neq P(\text{grade})P(\text{studytime}) \]
  - We must consider \( P(x,y) \) jointly, respecting the correlation!
  - If I tell you the grade, you learn something about study time.
We use a Kalman filter to estimate the whole state vector \textit{jointly}.

\[ x = [ x_r \ x_y \ x_t \ f_{1x} \ f_{1y} \ f_{2x} \ f_{2y} \ \ldots \ f_{nx} \ f_{ny} ]^T \]

State vector has \( N \) elements.

We don’t have a scalar variance \( \sigma^2 \), we have \( N \times N \) covariance matrix \( \Sigma \).

- Element \((i,j)\) tells us how the uncertainties in feature \( i \) and \( j \) are related.
Kalman Filters and Multi-Gaussians

- Kalman equations tell us how to incorporate observations
  - Propagating effects due to correlation
- Kalman equations tell us how to add new uncertainty due to robot moving
  - We choose a Gaussian noise model for this too.
System Equations (EKF)

- Consider range/bearing measurements, differentially driven robot
- Let $x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$  \( u = \) control inputs, \( w = \) noise
- Let $z_k = h(x_k, v_k)$  \( v = \) noise

\[
\begin{align*}
  f &= \begin{pmatrix}
    x' = x + (u_d + w_d) \cos(\theta + w_\theta) \\
    y' = y + (u_d + w_d) \sin(\theta + w_\theta) \\
    \theta' = \theta + u_\theta + w_\theta
  \end{pmatrix} \\
  h &= \begin{pmatrix}
    z_d = [(x_f - x_r)^2 + (y_f - y_r)^2]^{1/2} + v_d \\
    z_\theta = \arctan 2(y_f - y_r, x_f - x_r) - x_\theta + v_\theta
  \end{pmatrix}
\end{align*}
\]
EKF Update Equations

- **Time update:**
  - $x' = f(x, u, 0)$
  - $P = APA^T + WQWT$

- **Observation**
  - $K = PH^T(HPH^T + VRV^T)^{-1}$
  - $x' = x + K(z - h(x, 0))$
  - $P = (I - KH)P$

- $P$ is your covariance matrix

- They look scary, but once you compute your Jacobians, it just works!
  - $A = df/dx \quad W = df/dw \quad H = dh/dx \quad V = dh/dv$
  - Staff can help… (It’s easy except for the atan!)
## EKF Jacobians

\[ f = \begin{pmatrix}
  x' &= x + (u_d + w_d) \cos(\theta + w_\theta) \\
  y' &= y + (u_d + w_d) \sin(\theta + w_\theta) \\
  \theta' &= \theta + u_\theta + w_\theta \\
  x_1' &= x_1 \\
  y_1' &= y_1
\end{pmatrix} \]

\[ d = \left[ (x_f - x_r)^2 + (y_f - y_r)^2 \right]^{1/2} \]

\[ d_x = x_f - x_r \]

\[ d_y = y_f - y_r \]

\[
A = \begin{bmatrix}
  1 & 0 & -u_d \sin(\theta) & 0 & 0 \\
  0 & 1 & u_d \cos(\theta) & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
W = \begin{bmatrix}
  \cos(\theta) & -u_d \sin(\theta) \\
  \sin(\theta) & u_d \cos(\theta) \\
  0 & 1 \\
  0 & 0 \\
  0 & 0
\end{bmatrix}
\]

\[
Q = \begin{bmatrix}
  \sigma_{w_d}^2 & 0 \\
  0 & \sigma_{w_\theta}^2
\end{bmatrix}
\]
EKF Jacobians

\[
\begin{align*}
    h &= \left\{ z_d = \left[ (x_f - x_r)^2 + (y_f - y_r)^2 \right]^{1/2} + \nu_d \right. \\
    &\quad \left. z_\theta = \arctan 2(y_f - y_r, x_f - x_r) - x_\theta + \nu_\theta \right\} \\

    \lambda &= 1/(1 + (d_y/d_x)^2) \\
    H &= \begin{bmatrix}
        -d_x/d & -d_y/d & 0 & d_x/d & d_y/d \\
        \lambda d_y/d_x^2 & -\lambda/d_x & -1 & -\lambda d_y/d_x^2 & \lambda/d_x \\
    \end{bmatrix} \\
    V_{R}V^{T} &= \begin{bmatrix}
        \sigma_{\nu_{d}}^2 & 0 \\
        0 & \sigma_{\nu_{\theta}}^2 \\
    \end{bmatrix}
\end{align*}
\]
Kalman Filter: Properties

- In the limit, features become highly correlated
  - Because observing one feature gives information about other features
- Kalman filter computes the posterior pose, but **not** the posterior trajectory.
  - If you want to know the path that the robot traveled, you have to make an extra “backwards” pass.
Kalman Filter: a movie
Kalman Filters’ Nemesis

- With $N$ features, update time is $O(N^2)$!
- For Maslab, $N$ is small. Who cares?
- In the “real world”, $N$ can be $10^6$.

- Current research: lower-cost mapping methods
Non-Bayesian Map Building
Attack Plan

• Motivation: why build a map?
• Terminology, basic concepts
• Mapping approaches
  – Metrical
    • State Estimation
    • Occupancy Grids
  – Topological
• Data Association
• Hints and Tips
Occupancy Grids

• Another way of mapping:
• Divide the world into a grid
  – Each grid records whether there’s something there or not
  – Use current robot position estimate to fill in squares according to sensor observations
Occupancy Grids

- Easy to generate, hard to maintain accuracy
  - Basically impossible to “undo” mistakes
- Occupancy grid resolution limited by the robot’s position uncertainty
  - Keep dead-reckoning error as small as possible
  - When too much error has accumulated, save the map and start over. Use older maps for reference?
Attack Plan

• Motivation: why build a map?
• Terminology, basic concepts
• Mapping approaches
  – Metrical
    • State Estimation
    • Occupancy Grids
  – Topological
• Data Association
• Hints and Tips
Topological Maps

- Try to estimate how locations are related
- “There’s an easy (straight) path between feature 1 and 2”
- Easy to build, easy to plan paths
- Only a partial representation of the world
  - Resulting paths are suboptimal
Topological Maps

- *Much* easier than this metrical map stuff.

- Don’t even try to keep track of *where* features are. Only worry about *connectivity*. 
Note that the way we draw (where we draw the nodes) does not contain information.
Topological Map-Building Algorithm

- Until exploration round ends:
  - Explore until we find a previously unseen barcode
  - Travel to the barcode
  - Perform a 360 degree scan, noting the barcodes, balls, and goals which are visible.
  - Build a tree
    - Nodes = barcode features
    - Edges connect features which are “adjacent”
    - Edge weight is distance
Topological Maps: Planning

- Graph is easy to do process!
- If we’re lost, go to nearest landmark.
  - Nodes form a “highway”
- Can find “nearest” goal, find areas of high ball density
  - A* Search
Attack Plan

• Motivation: why build a map?
• Terminology, basic concepts
• Mapping approaches
  – Metrical
    • State Estimation
    • Occupancy Grids
  – Topological
• Data Association
• Hints and Tips
Data Association

- If we can’t tell when we’re reobserving a feature, we don’t learn anything!
  - We need to observe the same feature twice to generate a constraint
Data Association: Bar Codes

- Trivial!
- The Bar Codes have unique IDs; read the ID.
Data Association: Tick Marks

- The blue tick marks can be used as features too.
  - You only need to reobserve the same feature twice to benefit!
  - If you can track them over short intervals, you can use them to improve your dead-reckoning.
Data Association: Tick Marks

- Ideal situation:
  - Lots of tick marks, randomly arranged
  - Good position estimates on all tick marks

- Then we search for a *rigid-body-transformation* that best aligns the points.
Data Association: Tick Marks

- Find a rotation that aligns the most tick marks…
  - Gives you data association for matched ticks
  - Gives you rigid body transform for the robot!

Rotation+Translation
Attack Plan

• Motivation: why build a map?
• Terminology, basic concepts
• Mapping approaches
  – Metrical
    • State Estimation
    • Occupancy Grids
  – Topological
• Data Association
• **Hints and Tips**
Using the exploration round

- Contest day:
  1. During exploration round, build a map.
  2. Write map to a file.
  3. During scoring round, reload the map.
  4. Score lots of points.

- Use two separate applications for explore/score rounds.

- Saving state to a file will ease testing:
  - You can test your scoring code without having to re-explore
  - You can hand-tweak the state file to create new test conditions or troubleshoot.
Debugging map-building algorithms

- You can’t debug what you can’t see.

- Produce a visualization of the map!
  - Metrical map: easy to draw
  - Topological map: draw the graph (using graphviz/dot?)
  - Display the graph via BotClient

- Write movement/sensor observations to a file to test mapping independently (and off-line)
Course Announcements

- Gyros:
  - Forgot to mention that your first gyro costs ZERO sensor points.
  - Gyro mounting issues: axis of rotation

- Lab checkoffs
  - Only a couple checkoffs yesterday
Today’s Lab Activities

- No structured activities today
- Work towards tomorrow’s check-off:
  1. Robot placed in playfield
  2. Find and approach a red ball.
  3. Stop.
- Keep it simple!
  - Random walks are fine!
  - Status messages must be displayed on OrcPad or BotClient