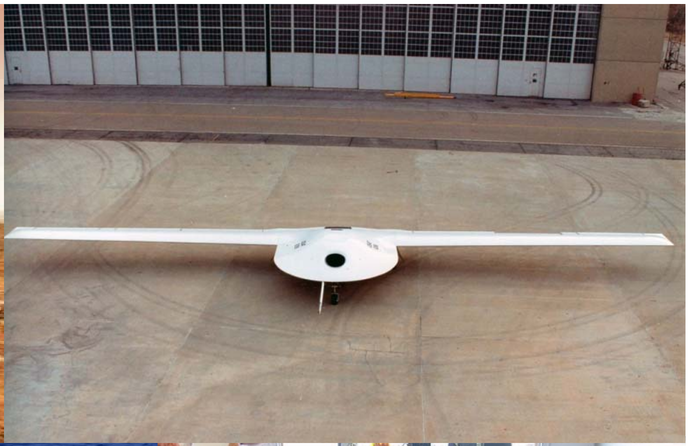
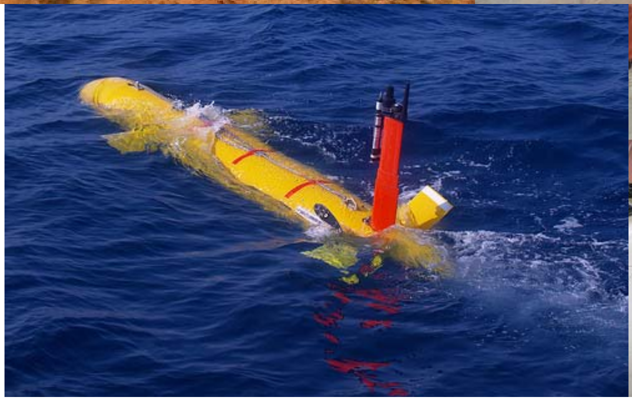




ME 597: Autonomous Mobile Robotics Review Lecture

Prof. Steven Waslander

EXAM FORMAT

- Length
  - 3 questions similar to example problems
  - Total worktime: hopefully <16 hours
    - Depends mostly on programming skill
  - Total available 24 hours

- Schedule
  - Exam posted to Learn at 9:00 AM on Saturday, December 13th, 2014
  - Submit one PDF file and one code zip folder electronically into exam dropbox by Sunday December 14th, 2014 at 9:00 AM
- Questions: Email me directly at stevenw@uwaterloo.ca. Note I'll be asleep 12-6AM

Likely Exam Topics

- Probability basics
- Coordinate systems
- Motion models
- Linear Quadratic Regulator and Tracking
- Nonlinear vehicle steering control
- Measurement and inverse measurement models
- Kalman Filter
- Extended Kalman Filter
- Particle Filter
- EKF Localization
- Particle Localization
- Occupancy grid mapping
- EKF SLAM
- FastSLAM 1.0
- RANSAC
- Scan Registration
- Potential fields
- Trajectory Rollout
- Wavefront
- Dijkstra, A* algorithm
- Visibility graphs
- Probabilistic Roadmaps
- Rapidly Expanding Random Trees
- Nonlinear programming*

- Occupancy Grid SLAM*
- GraphSLAM*

* Possible, not likely.

Unlikely Exam Topics

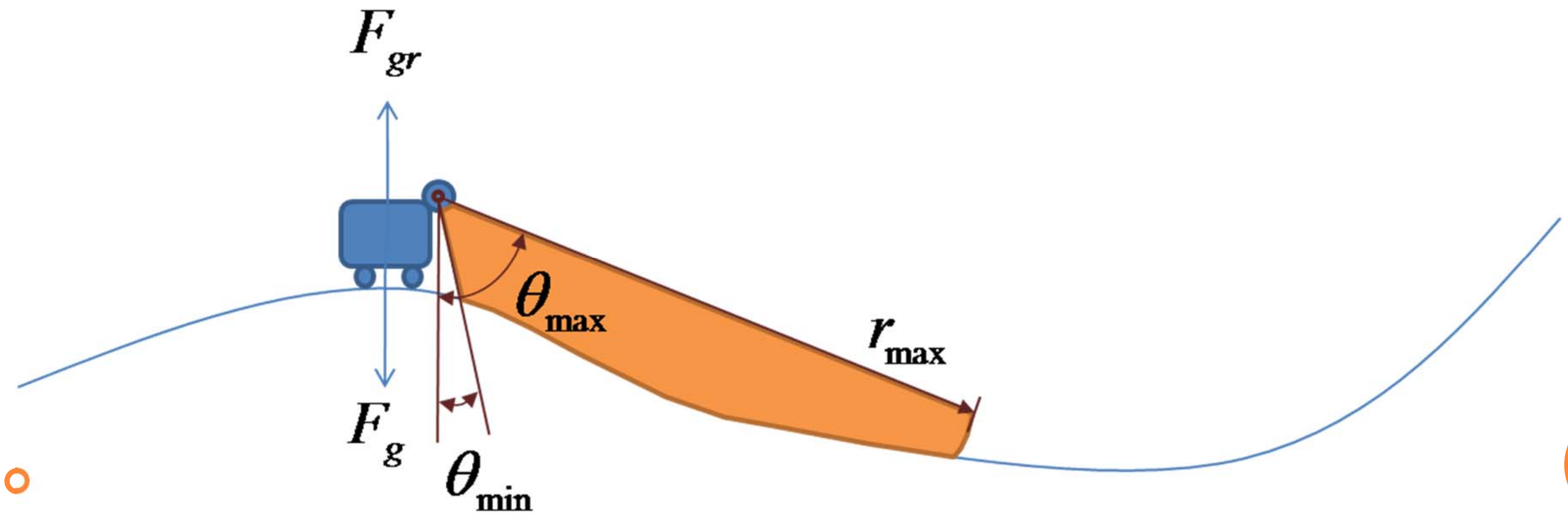
- Optimization Theory
- Optimization Algorithms
- Dynamic programming
- Aerial, legged or tracked vehicles
- Feedback linearization, Backstepping, Sliding mode control
- Contact sensors
- Vision
- Unscented Kalman Filters
- Bug algorithms
- Generalized Voronoi Decomposition
- Linear Programming
- Mixed Integer Linear Program

Final Exam

- No communication with other human beings during exam
- No communication with other human beings during exam
- No communication with other human beings during exam
- No communication with other human beings during exam
- No communication with other human beings during exam
- No communication with other human beings during exam

Review Questions

- 3. A wheeled robot of neglible length is moving along undulating terrain in a straight line, as defined below. The current vehicle position is known exactly thanks to a very accurate GPS sensor, its orientation due to a 2-axis accelerometer measuring the ground reaction force, F_{gr}. The robot is controlled to maintain a constant horizontal velocity of 0.5 m/s regardless of the slope of the terrain. Equipped with a 5 Hz laser scanner pointing forward and down along the direction of motion, your job is to create a map of the terrain elevation as the robot moves.
-



Review Questions

- a) [5] Define the states of the vehicle and the terrain and define the described motion in order to simulate it. Define a two axis occupancy grid (x,z) with 0.1m grid spacing.

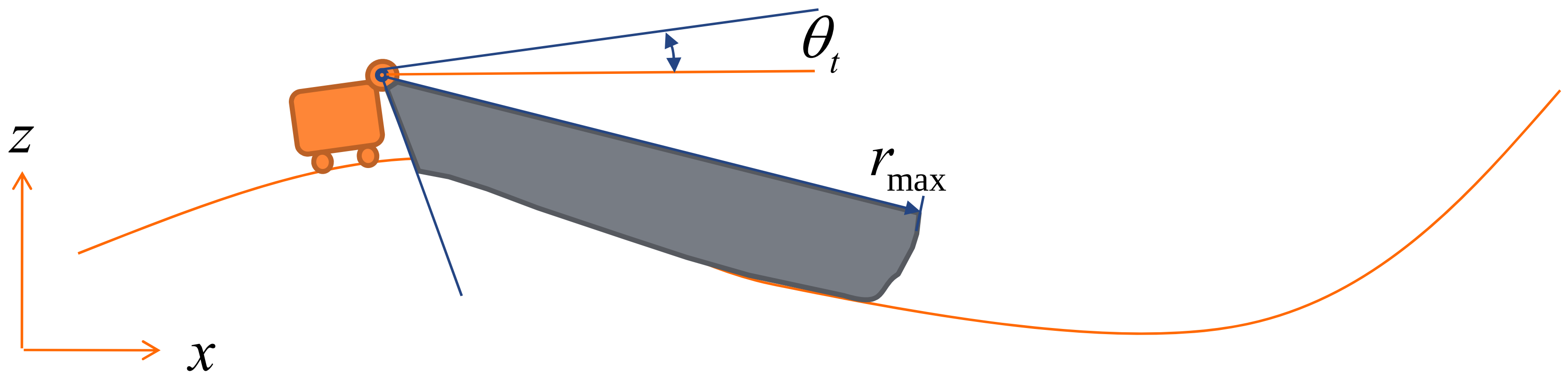
REVIEW QUESTIONS

- a) [5] Define the states of the vehicle and the terrain and define the described motion in order to simulate it. Define a two axis occupancy grid (x,z) with 0.1m grid spacing.

- Note: Use clear, consistent notation, work out details on paper before coding.
  - Don't try and see (it will take much longer to debug than to get it right the first time).

Review Questions

- Forces don't matter, this is a pure mapping problem
- Redefine Θ relative to forward axis



- 5 Hz update rate $dt = 0.2$
- Grid [0,10] X [0,10] with 0.1 m cell size is 100 X 100
- Assume X moves one grid to the right each timestep and takes new measurements.

Review Questions

- Simulation of robot motion can proceed without a real motion model, since its state is assumed to be known exactly from GPS and tilt sensor.
  - z is 1 meter above ground.
  - Θ is calculated based on local slope approximation (or derivative).

$$x_t = x_{t-1} + 0.5dt = x_{t-1} + 0.1$$

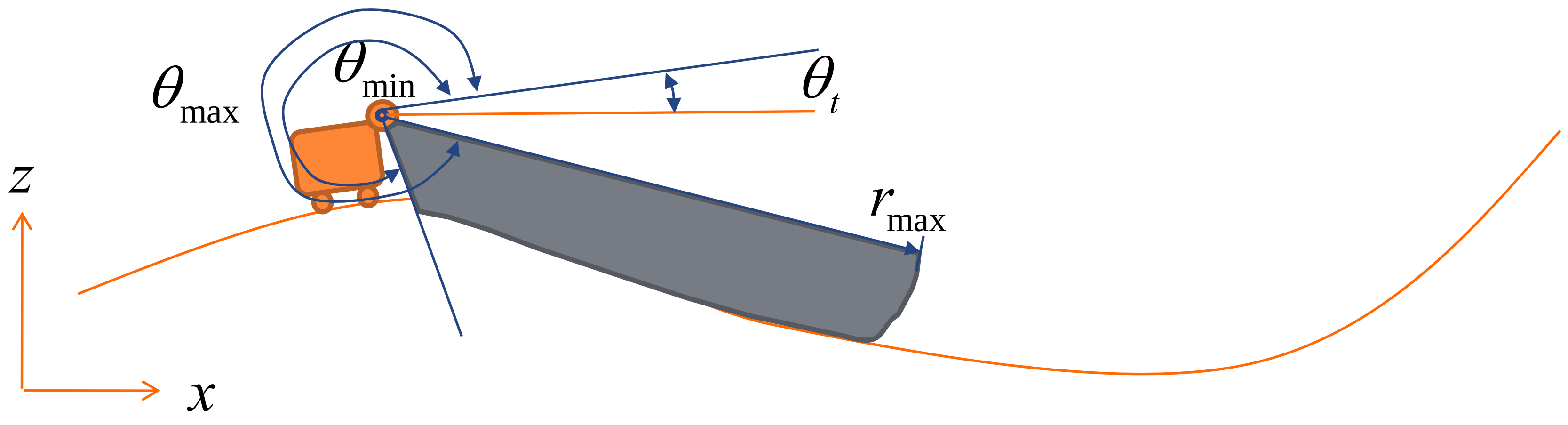
$$z_t = 10(\sin(x_t + 1) + \sin(x_t / 2 + 0.3) + \sin(x_t / 3 + 0.5)) + 1$$

$$\theta_t = \tan^{-1}\left(\frac{z_t - z_{t-1}}{x_t - x_{t-1}}\right)$$

Review Questions

- b) [10] Create an inverse measurement model that receives 16 measurements evenly spaced between Θ_{min} = 60° and Θ_{max} = 15° down from the forward horizontal with r_{max} = 5 m and returns the probability of a grid being occupied by the ground. The height of the scanner is 1m above the ground. For the probability of ground, use 0.7, and the probability of not ground, use 0.3. Is it possible to assume the ground is solid beyond a valid range measurement, why or why not?

Review Questions



- Measurement angles, with spacing of 3 degrees ((60-15)/15), 16 measurements total.

$$\theta^s = \left[3\pi/2 + \theta_{\min} \quad \cdots \quad 3\pi/2 + \theta_{\max}\right]$$

$$\alpha = 0.1, \quad \beta = 3^\circ$$ Or Bresenham

Review Questions

- Inverse Measurement Model
  - Get range and bearing to each cell

$$r^i = \sqrt{\left(m_x^i - x_{1,t}\right)^2 + \left(m_y^i - x_{2,t}\right)^2}$$

$$\phi^i = \tan^{-1}\left(\frac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}}\right) - x_{3,t}$$

  - Find relevant range measurement for that cell
    - Closest bearing of a measurement

$$k = \arg\min\left(\left|\phi^i - \phi^j\right|\right)$$

Review Questions

- What's different?
  - Only if max range is returned do we have no info beyond a range measurement
    - if $r^i = r^s_{\max}$ or $|\phi^i - \phi^s_k| > \beta/2$

      $$p(m^i \mid y_t) = 0.5$$

    - else if $r^s_k < r^s_{\max}$ and $(r^i - r^s_k) >= \alpha/2$

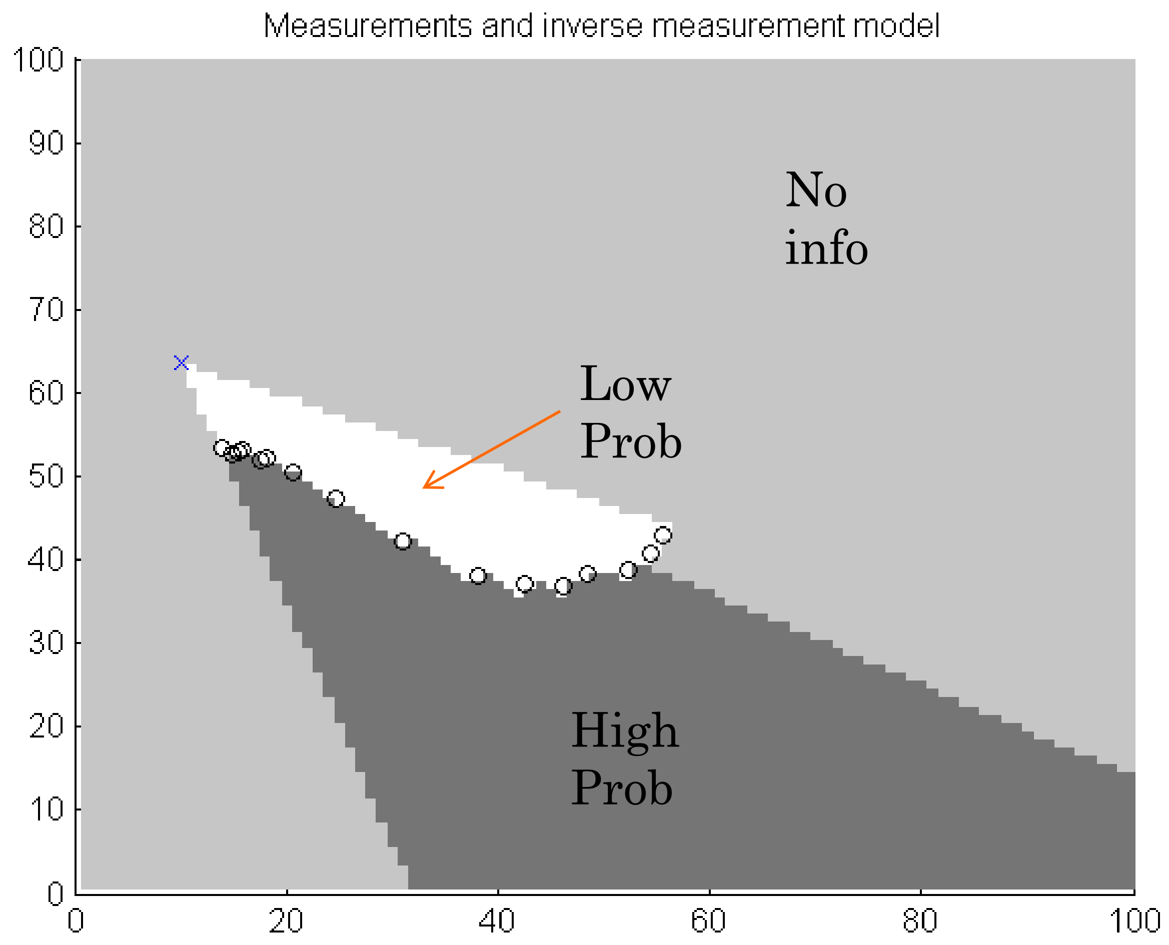
      $$p(m^i \mid y_t) = 0.7$$

    - else if $r^i < r^s_k$

      $$p(m^i \mid y_t) = 0.3$$

REVIEW QUESTION

- Example Inverse Measurement Model



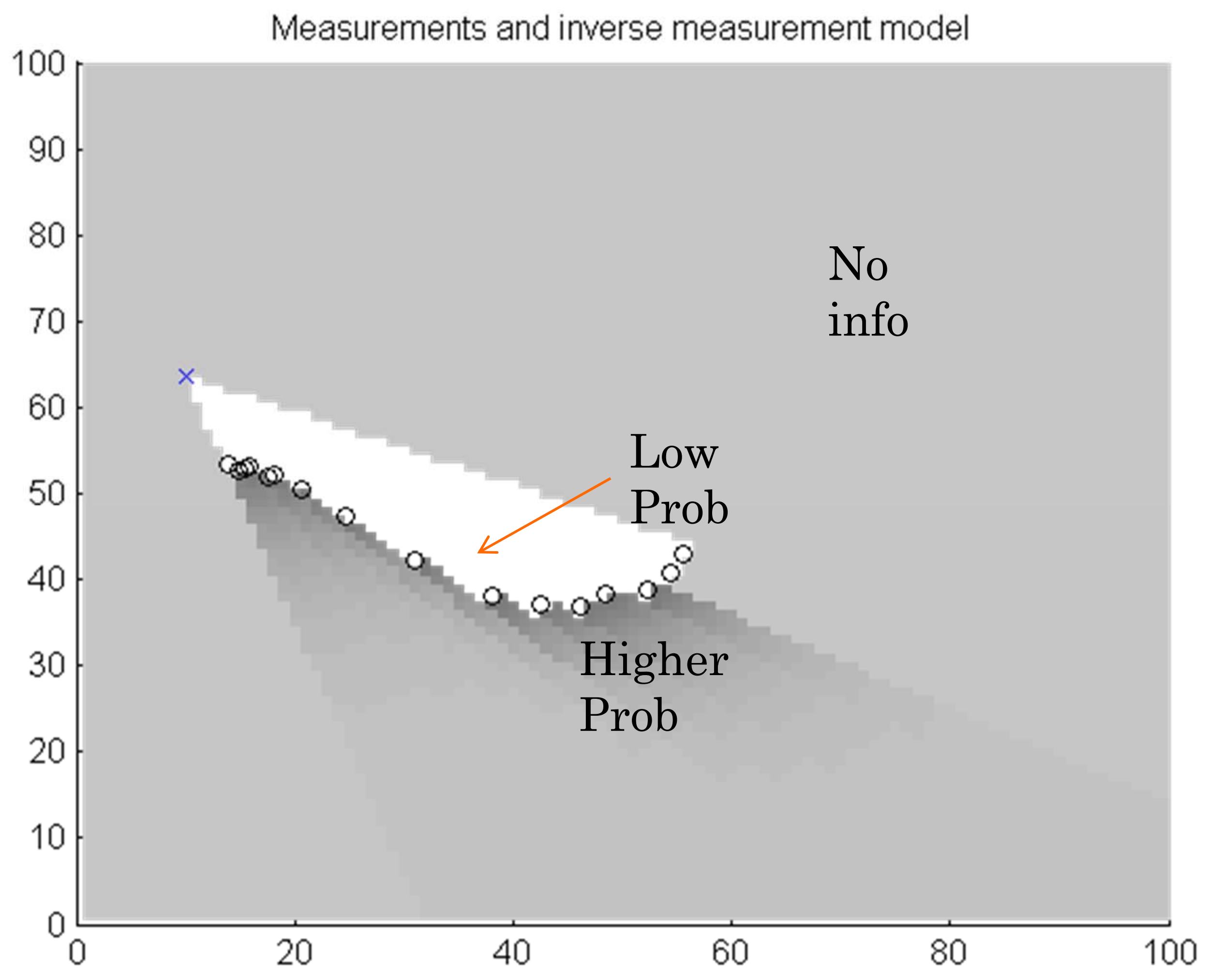
REVIEW QUESTIONS

- Is it possible to assume solid ground beyond valid range measurements?
  - Not absolutely, but it is a reasonable assumption if we leave some uncertainty in that region
  - As defined above, we have assumed that ground is equally probable beyond a valid range measurement as at the valid range measurement.
  - We could also adapt this to have diminishing certainty with increasing distance beyond measurement range

$$p(m^i \mid y_t) = 0.7 - 0.2(1 - e^{-(r^i - r_k^s)})$$

Review Question

- Example Inverse Measurement Model



Review Questions

- Equally valid to use Bresenham version of sensor model
  - Adapt update regions along line, and proceed to end of map
    - if $r_k^s < r^i$

$$p(m^i \mid y_t) = 0.3$$

    - if $r^i < r_{\max}^s$ and $r_k^s > r^i$

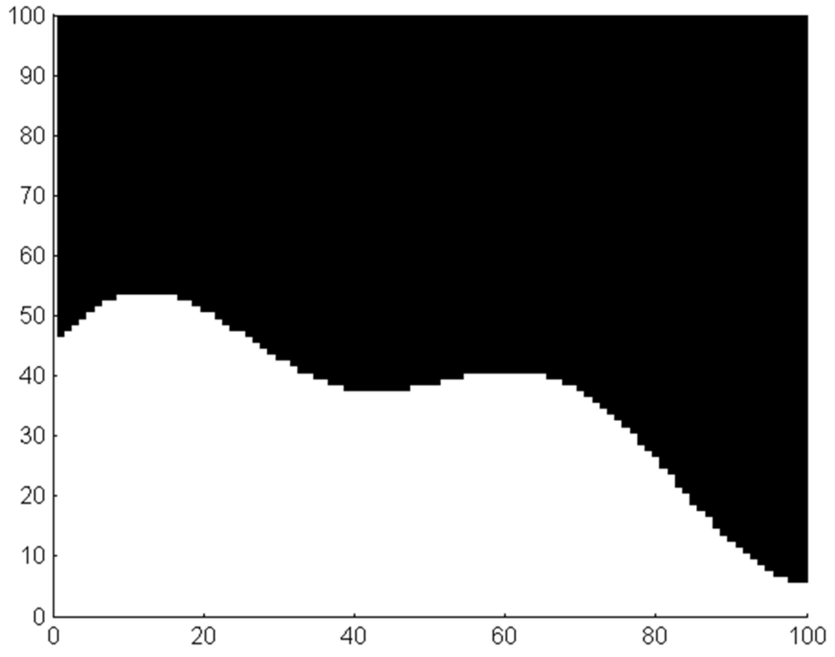
$$p(m^i \mid y_t) = 0.7 \quad \text{or} \quad p(m^i \mid y_t) = 0.7 - 0.2(1 - e^{-(r^i - r_k^s)})$$

REVIEW QUESTIONS

- c) [10] Develop a simulation with the following terrain profile, where *h(x)* is the height of the ground at position *x* in [0,10].

$$h(x) = 10(\sin(x+1) + \sin(x/2+.3) + \sin(x/3+.5))$$

Simulate for a 20 second journey at a constant horizontal velocity of 0.5 m/s, with the scanner returning measurement sets at 5 Hz.



Review Questions

- So now we use our full mapping simulation to generate the results. Code is available, but changes were needed to
  - Getranges(), to account for 0.1m cell spacing
  - Inversescanner(), to account for changes in scanner model when measuring the ground
  - Mapping.m, to change vehicle motion and sensor height, environment map, and x, z axis configuration.

Review Questions

Modify getranges( ) to take cell size alpha (could also convert to Bresenham

```
…
for i=1:length(meas_phi) for j=1:round(rmax/alpha)
       % Determine the x,z range to the cell
       xi = x+alpha*j*cos(th+meas_phi(i));
       zi = z+alpha*j*sin(th+meas_phi(i));
       % Determine cell coordinates
       ix = round(xi/alpha);
       iz = round(zi/alpha);

       % If not in the map, set measurement there and stop going further
       % If not in the map, set measurement invalid and stop going further
       if (ix<=1||ix>=M||iz<=1||iz>=N)
           meas_r(i) = rmax; % alpha*j;
           break;
       % If in the map but hitting an obstacle, set measurement range and
       % stop going further
       elseif (map(ix,iz))
           meas_r(i) = alpha*j;
           break;
       end
    end
```

Review Questions

```
Change inversescanner() per model
for i = 1:M
    for j = 1:N
        % Find range and bearing to the current cell
        r = sqrt((i*alpha-x)^2+(j*alpha-z)^2);
        phi = mod(atan2(j*alpha-z,i*alpha-x)-theta,2*pi);

        % Find the applicable range measurement
        [meas_cur,k] = min(abs(phi-meas_phi));
        phi_s(i,j) = phi;

        % If behind out of range measurement, or outside of field
        % of view, no new information is available
        if ((meas_r(k) == rmax) && (r - rmax >= -alpha/2)) || (abs(phi-
meas_phi(k))>beta/2)
            m(i,j) = 0.5;
        % If the range measurement was before this cell, likely to be an object
        elseif ((r - meas_r(k) >= -alpha/2))
            m(i,j) = 0.7 - 0.2*(1-exp(-(r-meas_r(k))));
        % If the cell is in front of the range measurement, likely to be empty
        else
            m(i,j) = 0.3;
        end
end
```

Review Questions

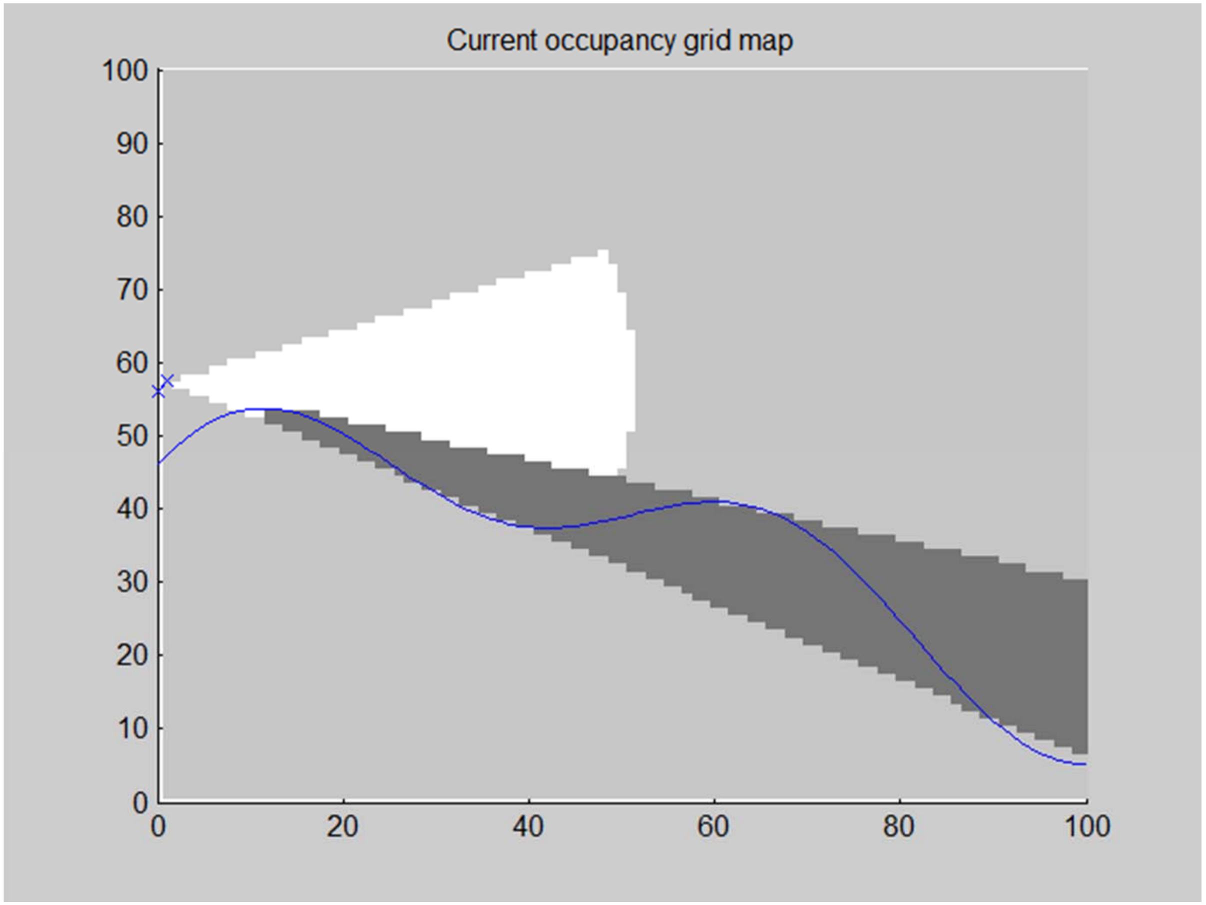
```
% Robot motion defined by terrain
 x(:,t) = [xmap(t); zmap(t)+1; thmap(t)];

 % Generate a measurement data set
 meas_r = getranges(map,x(:,t),meas_phi,rmax,alpha);

 %% Map update
 % Get inverse measurement model
 invmod = inversescanner(N,N,x(1,t),x(2,t),x(3,t),meas_phi,meas_r,
                           rmax,alpha,beta);
```

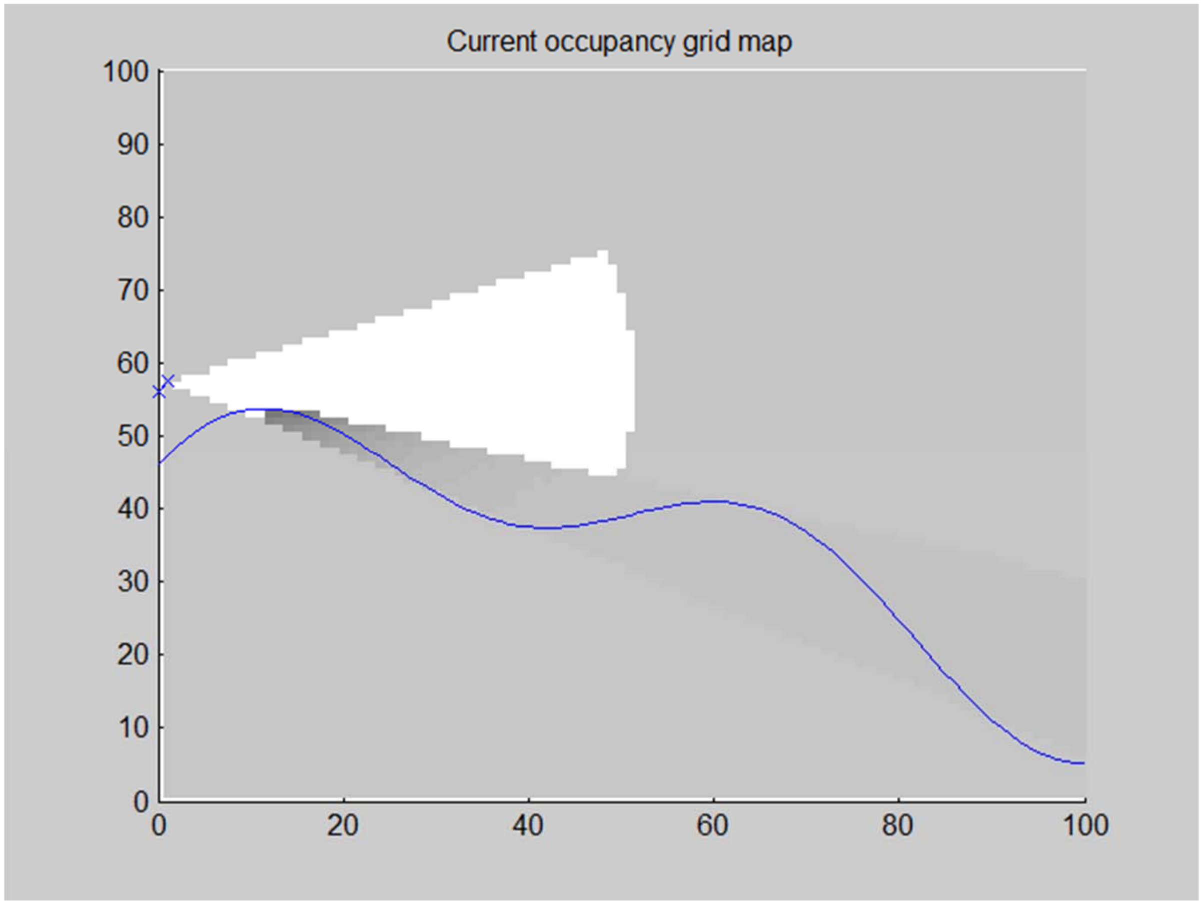
Review Questions

- Didn’t ask for very specific graphs, will do so on the final.



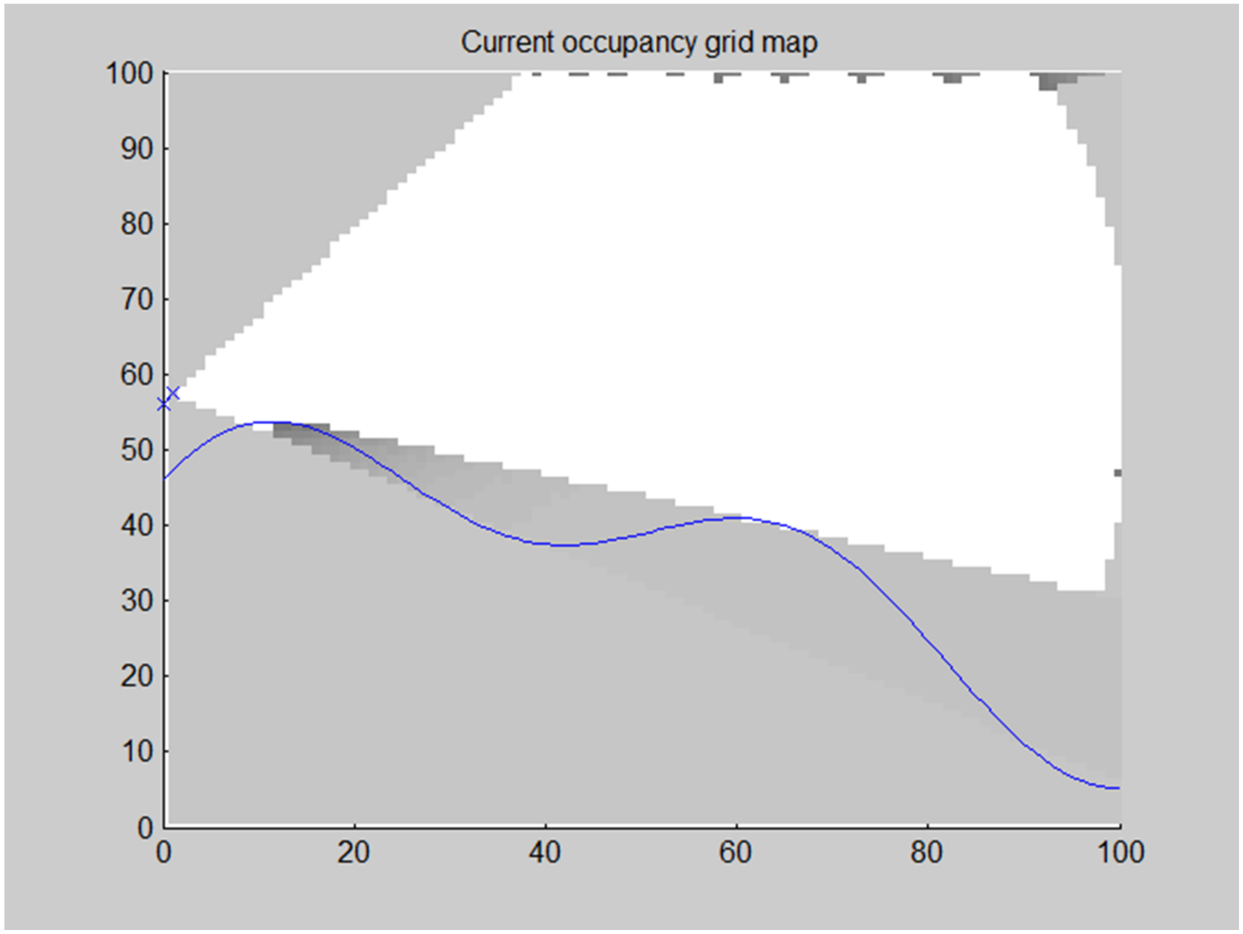
Review Questions

- And with the decay in probability beyond range measurement.



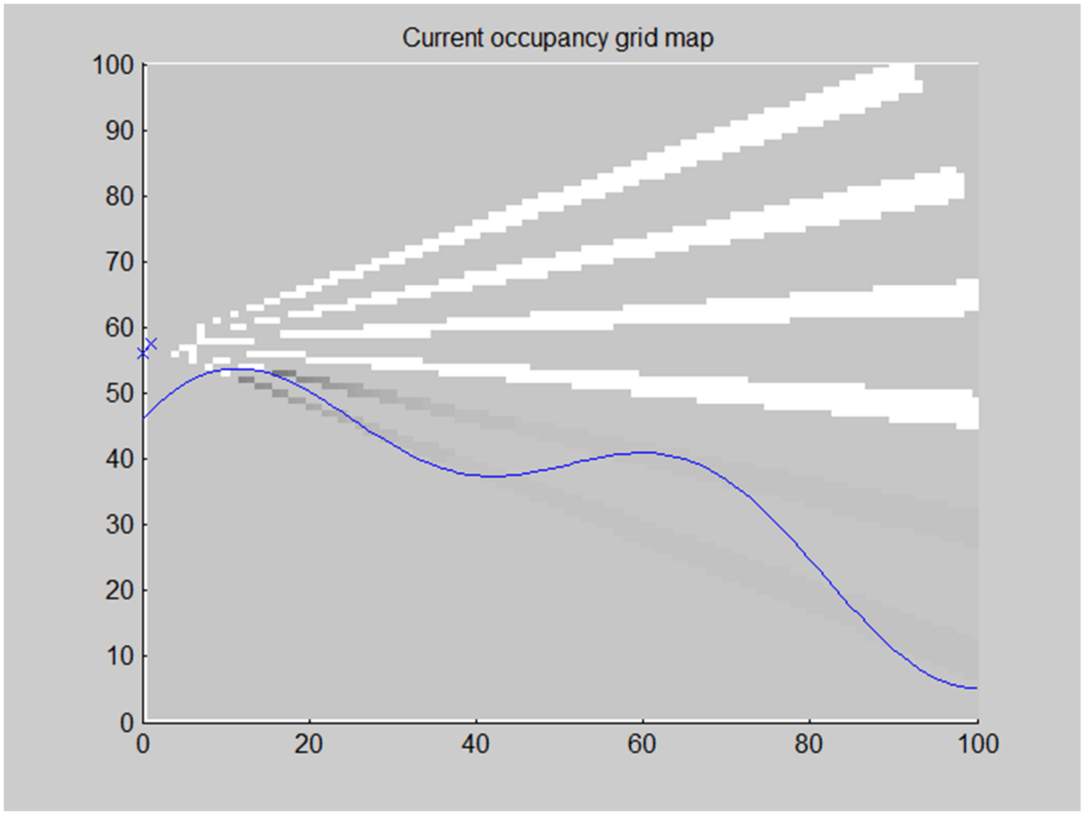
Review Questions

- Finally, this whole exercise indicates some clear benefits to a different laser configuration



Review Questions

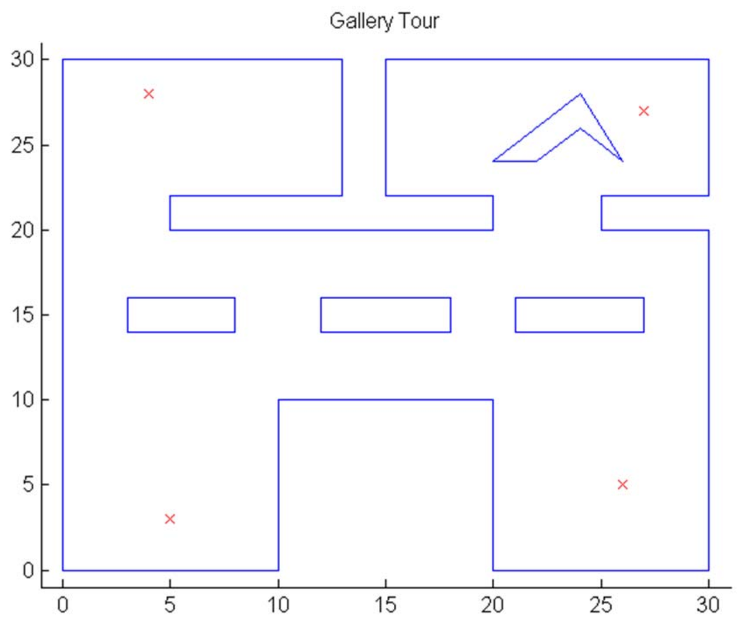
- Can also easily test the assumption that laser is an ever expanding beam
  - Reduce to 5 measurements, but maintain 3 degree cone

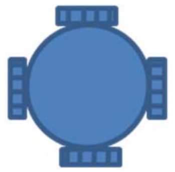


Review Questions

- In this problem, we will develop a reactive motion planner for 2D motion of ground vehicle for navigation in a gallery. The robot is required to lead a tour through the gallery and arrive at the various points of interest, while avoiding sculptures, chairs, walls etc. en route. The environment is most easily represented as an occupancy grid, which is defined by the file ”gallery.m”, available on ACE and is depicted in Figure 2. The robot is the four swedish wheel variety as depicted in Figure 1, and so can be described with linear dynamics (note: speakers are located on all sides of the robot guide, so that heading is irrelevant and can be maintained at a constant value).

- a) Define a motion model for the robot with four 90 swedish wheels arranged as in Figure 1.





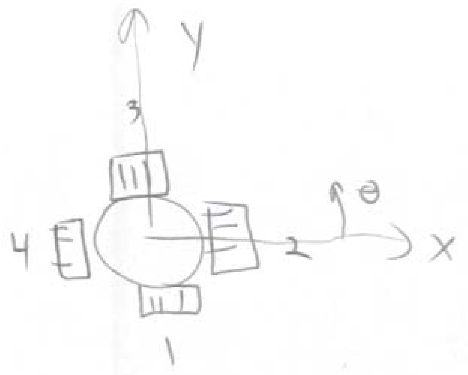
Review Questions

- Again, clearly define variables, coordinates and model

Motion model

Inputs $u_1 = V_1$ & V_3

$U_2 = V_2$ & V_4



so both wheels are commanded at same speed, can be considered and axel command.

assume motor torque is instantaneous from voltage command results in accel control.

$$\begin{bmatrix} x \\ V_x \\ y \\ V_y \end{bmatrix}_t = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ V_x \\ y \\ V_y \end{bmatrix}_{t-1} + \begin{bmatrix} 0 & 0 \\ dt & 0 \\ 0 & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t$$

Review Questions

- b) Develop a potential field planner by defining attractive potentials for each of the targets locations on the tour, and repulsive potentials for all of the obstacles in the environment. Should all the attractive potentials be active at every point in time? How would you handle the sequential nature of the tour? How would you translate the steepest descent direction of the potential into control commands for the motion model defined in part a)? What happens if the robot gets stuck in a local minimum?

- Note: Solution outlined below, be sure to write out equations used for exam.

Review Questions

- The map in this question is harder than the ones presented in class, with nonconvex obstacles and lots of difficult passages
  - Likely a lot of local minima

- The active potentials should only be active one at a time
  - Add a logic layer to handle the sequence of waypoints to visit.
    - While not at final point
      - If within 1m of current point
        - Set attractive potential to next point
      - End
    - End

Review Questions

- The repulsive potentials will be as in class, with some exceptions
  - Since the boundary is complex, it can be converted to a set of polygon obstacles
  - Each interior obstacle can be converted into discrete 1X1m obstacle chunks to facilitate shortest distance calculations on irregular boundaries.
- To extract a motion command from the potential field, just take gradient components in x and y as desired accelerations

Review Questions

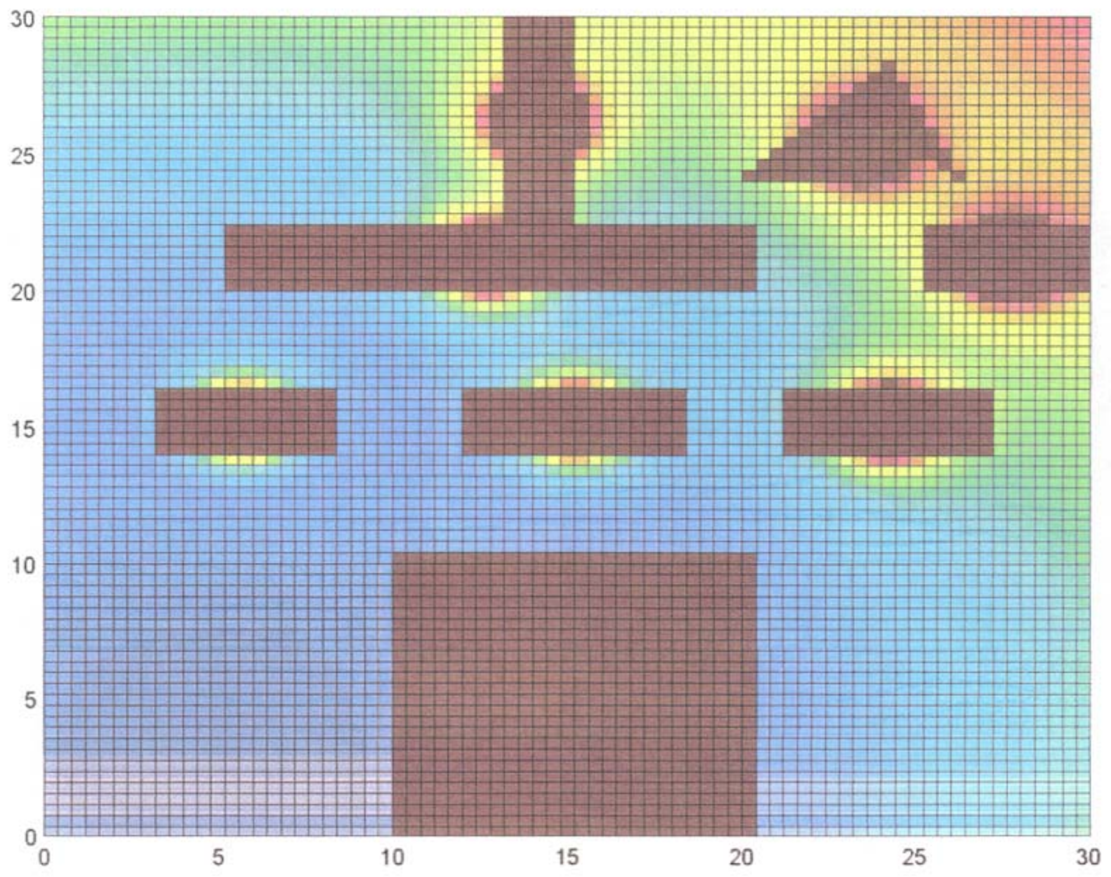
- c) Simulate a trajectory through the environment based on the potential field method. Be sure to pause for 5 seconds at each target location. Is the vehicle able to exactly achieve all of the target locations? Present a plot of the vehicle trajectory through the environment, and of the potential field that results for the first target location and obstacles.

Review Questions

- Take potential field code
  - Modify attractive potential selection
  - Discretize obstacles
  - Modify collision checks for discrete square obstacles

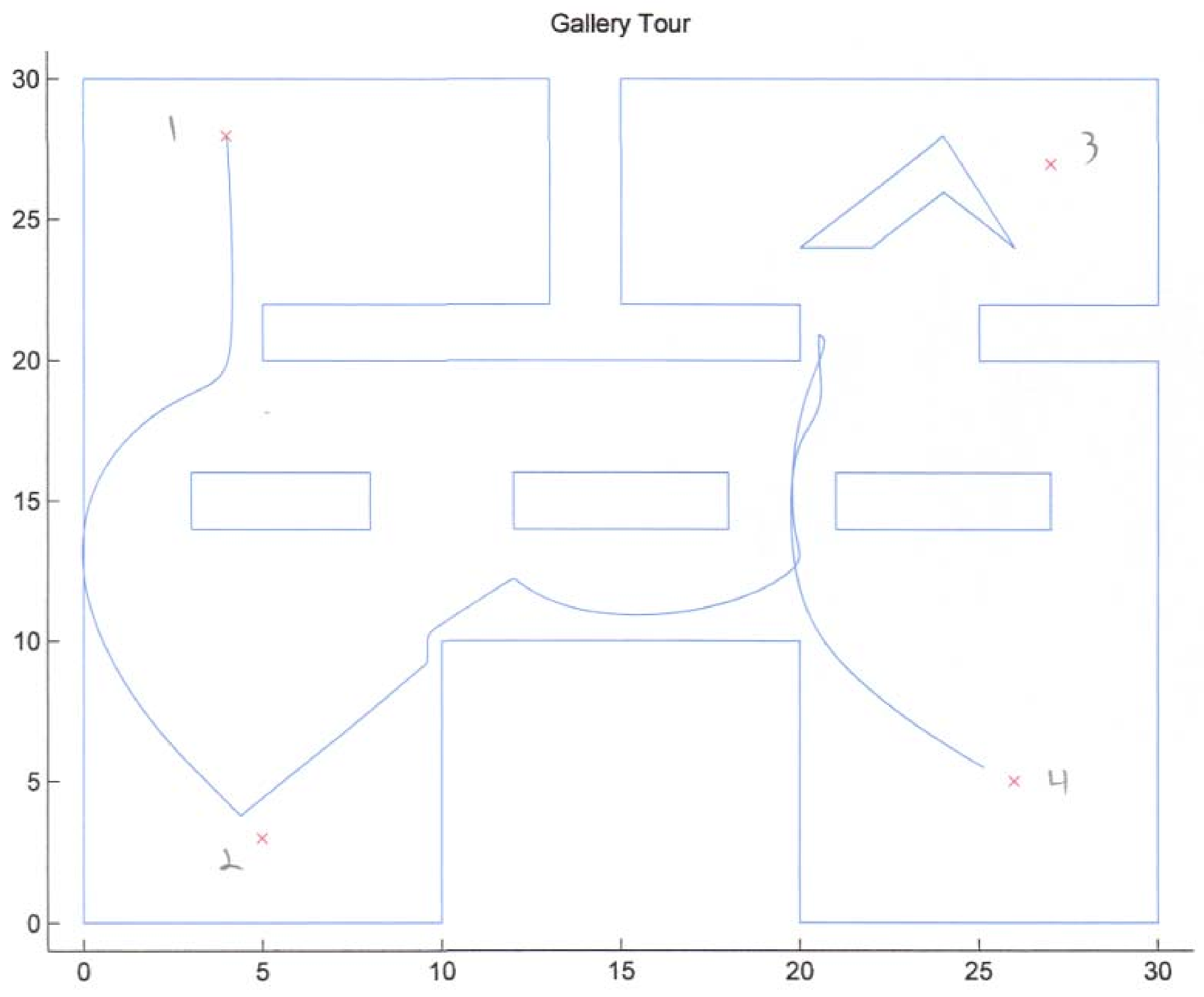
REVIEW QUESTIONS

- Potential Field for first leg of journey to waypoint 2.



Review Questions

- Full trajectory – with limitations (added a timeout)



Review Problems

- Repeat the above problem, but use the wavefront algorithm instead of the potential field method.
- a) Define the algorithm used to create a wavefront emanating from the current target location.
- b) Given a wavefront based potential field, define the algorithm used to select a trajectory to the current target location.
- c) Implement the algorithm for the gallery and target locations defined in the file ”gallery.m” available on LEARN. Present the trajectory followed for the entire tour of the gallery, and the wavefront potential generated for the first target location.

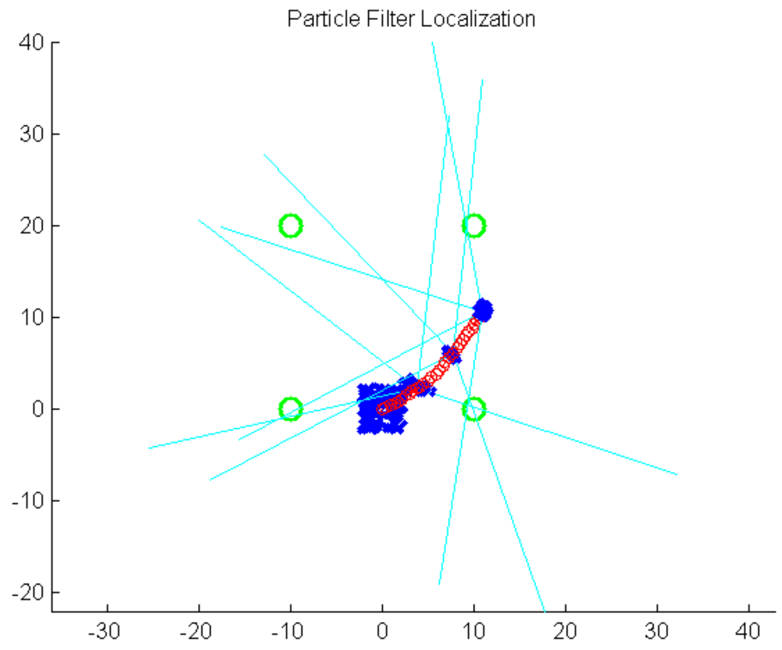
Review Questions

- Solution
  - Take wavefront code and apply in place of potential field

Review Questions

Particle Filters:

A surveillance aircraft is travelling at a fixed altitude and airspeed, but is affected by unknown constant winds. There are four visual landmarks the aircraft can use to localize its position, and it receives only bearing information from each of the four landmarks located at {(−10, 0), (10, 0), (10, 20), (−10, 20)}. The aircraft motion can be modelled identically to a two wheeled robot, with the exception of sideslip due to wind.



Review Questions

Let the state vector be

$$x_t = [X \quad Y \quad \theta \quad v_{x,w} \quad v_{y,w}]^T$$

where X, Y are the horizontal position of the aircraft, θ is its heading and $v_{x,w}$, $v_{y,w}$ are the horizontal components of the wind velocity.

The kinematic model of motion is,

$$x_t = \begin{bmatrix} x_{1,t-1} + (v\cos(x_{3,t-1}) + x_{4,t-1})dt \\ x_{2,t-1} + (v\sin(x_{3,t-1}) + x_{5,t-1})dt \\ x_{3,t-1} + \omega dt \\ x_{4,t-1} \\ x_{5,t-1} \end{bmatrix}$$

Review Questions

- a) [5] Define a measurement model for bearings only measurement of all four landmarks.

$$y_t = \begin{bmatrix} \tan^{-1}\left(\frac{\delta y_1}{\delta x_1}\right) \\ \tan^{-1}\left(\frac{\delta y_2}{\delta x_2}\right) \\ \tan^{-1}\left(\frac{\delta y_3}{\delta x_3}\right) \\ \tan^{-1}\left(\frac{\delta y_4}{\delta x_4}\right) \end{bmatrix} + \delta_t$$

$$\delta x_i = x_{1,t} - x_i$$
$$\delta y_i = x_{2,t} - y_i$$

$$\delta_t \sim N(0, R)$$

No mention of altitude above ground, assume measurements are projected onto x,y plane to reveal a 2D bearing only. If not stated, ask me, or state your assumption.

Review Questions

- b) [5] Define motion disturbances and measurement noise for both models.

$$Q = \begin{bmatrix} \sigma_X^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_Y^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{V_X}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{V_Y}^2 \end{bmatrix} \qquad R = \begin{bmatrix} \sigma_b^2 & 0 & 0 & 0 \\ 0 & \sigma_b^2 & 0 & 0 \\ 0 & 0 & \sigma_b^2 & 0 \\ 0 & 0 & 0 & \sigma_b^2 \end{bmatrix}$$

REVIEW QUESTIONS

- c) [15] Develop a particle filter for aircraft localization and wind speed estimation. Present all necessary variables and equations, and define each step in the algorithm.

- No changes, direct implementation of existing particle filter.

Review Questions

- d) [15] Implement the particle filter with in simulation with dt = 0.1 for an aircraft starting at the origin at altitude 25 m, with zero heading, with constant inputs of v = 5m/s, ω = 0.3rad/s, and with a constant windspeed of v_w= (−0.5, 2). Generate the following plots
  - i) the true state, the measurements and the particle set (only at 1 second intervals for the particle set)
  - ii) the estimate of the wind velocity over time.

Review Questions

- Code differences required
  - New motion model with five states
    - Initialization of simulation
    - Initialization of particles
    - Disturbance covariance
    - Motion update with wind
    - Particle prediction update
  - New measurement model with only bearing
    - Initialization of measurements
    - Noise covariance
    - Measurement simulation
    - Weighting – normpdf
  - Clean up plotting to generate required output

Review Questions

- Recall the Particle Filter Algorithm (simplified)

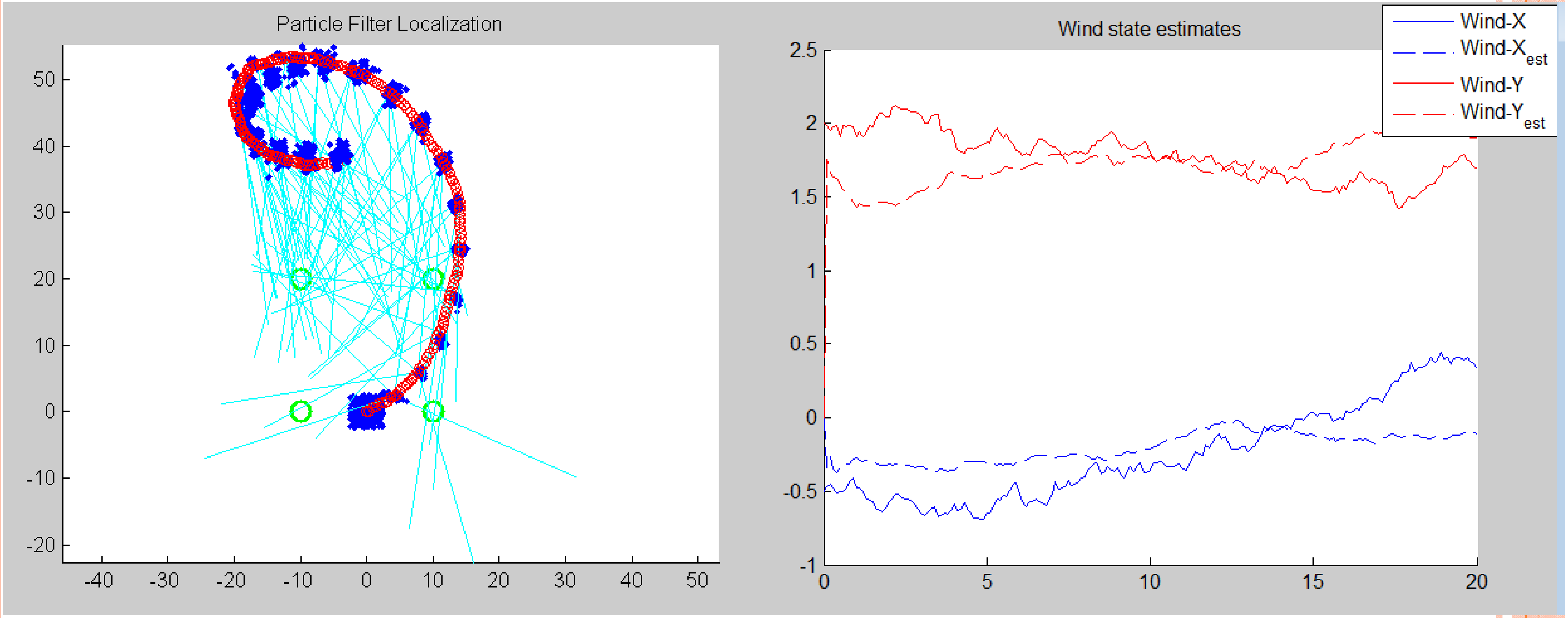
```
1. For each particle in $S_{t-1}$
    1. Propagate sample forward using motion model (sampling)
           $x_t^{[i]} \sim p(x_t \mid x_{t-1}^{[i]}, u_t)$
    2. Calculate weight                                   (importance)
           $w_t^{[i]} = p(y_t \mid x_t^{[i]})$
    3. Store in interim particle set
           $S'_t = S'_t + \{s_t^{[i]}\}$
2. For j = 1 to D
    1. Draw index $i$ with probability $\propto w_t^{[i]}$           (resampling)
        1. Add to final particle set
               $S_t = S_t + \{s_t^{[i]}\}$
```

Review Questions

- Particle filter code

```
%% Particle filter estimation
   for dd=1:D % for each particle
       e = RE*sqrt(Re)*randn(n,1); % Generate motion disturbance
       Xp(:,dd) = [X(1,dd)+(u(1,t)*cos(X(3,dd))+X(4,dd))*dt;
                   X(2,dd)+(u(1,t)*sin(X(3,dd))+X(5,dd))*dt;
                   X(3,dd)+u(2,t)*dt;
                   X(4,dd);
                   X(5,dd)] + e;  % propagate dynamics
       % Calculate expected measurements and weights
       hXp = [atan2(map(:,2)-Xp(2,dd),map(:,1)-Xp(1,dd))-Xp(3,dd)] + d;
       w(dd) = mvnpdf(y(:,t),hXp,Qm);
   end
   W = cumsum(w); % Switch to cumulative distribution
   for dd=1:D % Perform importance sampling
       seed = max(W)*rand(1);  % Draw sample uniformly
       X(:,dd) = Xp(:,find(W>seed,1)); % Keep corresponding particle
   end
   muParticle = mean(X'); % Calculate mean if needed
   SParticle = var(X');
```

Review Questions



Review Questions

- e) [5] How many particles did you choose to use in your particle filter implementation and why? What are the advantages and drawbacks of using more or less particles?
- I used 300 particles, 100 caused deprivation issues that crash the code when all weights are zero, and 1000 took too long.

Review Questions

- Nonlinear programming

Collision avoidance for aircraft using nonlinear programming. In this problem, your task will be to formulate an optimal path planner that resolves conflicts between multiple aircraft trajectories in the 2D plane using a receding horizon nonlinear program. We will assume that all vehicle position information is accurate and is known centrally, avoiding the need to perform vehicle state estimation or to coordinate inter-vehicle communication. The system of interest will involve *n* vehicles with trajectories defined in discrete time steps of 10 seconds, as follows,

$$x^{d,i} = \begin{bmatrix} x_{1,0}^{d,i} & x_{2,0}^{d,i} \\ x_{1,1}^{d,i} & x_{2,1}^{d,i} \\ \vdots & \vdots \\ x_{1,T}^{d,i} & x_{1,T}^{d,i} \end{bmatrix}$$

where the superscript *d,i* denotes the desired trajectory for vehicle *i*, and the subscript 1,*t* denotes the state variable *1* at time *t*. The trajectory is define for *T* timesteps into the future, and can be extended indefinitely in a straight line at constant speed if necessary.

Review Questions

The receding horizon to be used will be denoted . The vehicles must maintain a minimum separation of km. The aircraft have minimum and maximum speed requirements of $v \in [v_{\min}, v_{\max}]$ and maximum turn rates of $|\omega| \le \omega_{\max}$

a) Define a motion model for the aircraft, using the standard 2D two-wheeled robot model, and include additive Gaussian disturbances for simulation purposes only, with covariance values of

$$\Sigma_{xx} = \Sigma_{yy} = 0.1,\ \Sigma_{\theta\theta} = 0.01$$

Standard two wheel model, from motion model slides

$$\begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = g(x_{t-1}, u_t, \varepsilon_t) = \begin{bmatrix} x_{1,t-1} + u_{1,t} \cos x_{3,t-1} dt \\ x_{2,t-1} + u_{1,t} \sin x_{3,t-1} dt \\ x_{3,t-1} + u_{2,t} dt \end{bmatrix} + \varepsilon_t$$

$$R = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

Review Questions

b) Define all of the equality and inequality constraints that result from the motion model (ignoring disturbances), the collision avoidance constraints and input bounds for a receding horizon length .

$$\begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = g(x_{t-1}, u_t, \varepsilon_t) = \begin{bmatrix} x_{1,t-1} + u_{1,t} \cos x_{3,t-1} dt \\ x_{2,t-1} + u_{1,t} \sin x_{3,t-1} dt \\ x_{3,t-1} + u_{2,t} dt \end{bmatrix} + \varepsilon_t$$

Motion constraints, nonlinear equality

Review Questions

Expressing the constraints in our usual notation (at each timestep, for each vehicle or vehicle pair

$$\left|u_{t,2}^{i}\right| \le \omega_{\max}$$

$$u_{t,1}^{i} \le v_{\max}$$

$$u_{t,1}^{i} \ge v_{\min}$$

for all i,t

Input bounds

$$\left|x_{t}^{i}\right| \le x_{\max}$$

for all i,t

$$x_{0}^{i} = const.$$

for all i

State bounds
Linear equality

$$\left(x_{t,1:2}^{i} - x_{t,1:2}^{j}\right)^{T}\left(x_{t,1:2}^{i} - x_{t,1:2}^{j}\right) \ge d_{s}^{2}$$

for all i,j, t

Collision Avoidance constraints
Nonlinear inequality

Review Questions

c) Define a cost function that penalizes quadratic deviation from the desired trajectory at each timestep. Define a separate cost that penalizes the turn rate quadratically and combine the two linearly with weighting factors for the first term and for the second.

$$f = \beta f_x(x) + (1-\beta) f_u(u)$$

$$f_x = \sum_{t=1}^{T}\sum_{i=1}^{M}\left(x_{t,1}^{i,d} - x_{t,1}^{i}\right)^2 + \left(x_{t,2}^{i,d} - x_{t,2}^{i}\right)^2$$

$$f_u = \sum_{t=1}^{T}\sum_{i=1}^{M}\left(u_{t,2}^{i}\right)^2$$

Review Questions

d) Formulate the full nonlinear program for a single time interval, then define a receding horizon algorithm that solves this program at each time step. Explain how the nonlinear program is modified at each successive time step.

$$
\begin{aligned}
\min_{x,u} \quad & f(x,u) \\
s.t. \quad & \left|u_{t,2}^{i}\right| \le \omega_{\max} \\
& u_{t,1}^{i} \le v_{\max} \\
& u_{t,1}^{i} \ge v_{\min} \\
& \left|x_{t}^{i}\right| \le x_{\max} \\
& x_{0}^{i} = const. \\
& \left(x_{t,1:2}^{i} - x_{t,1:2}^{j}\right)^{T}\left(x_{t,1:2}^{i} - x_{t,1:2}^{j}\right) \ge d_{s}^{2} \\
& \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = g(x_{t-1}, u_t, \varepsilon_t) = \begin{bmatrix} x_{1,t-1} + u_{1,t}\cos x_{3,t-1}dt \\ x_{2,t-1} + u_{1,t}\sin x_{3,t-1}dt \\ x_{3,t-1} + u_{2,t}dt \end{bmatrix} + \varepsilon_t
\end{aligned}
$$

Review Questions

e) Implement the single stage nonlinear program for the following problem parameters.

$$\begin{aligned} n &= 3 \\ T_r &= 3 \\ dt &= 10 \text{ s} \\ v &\in [50,150] \text{ m/s} \\ \omega_{\max} &= 0.03 \text{ rad/s} \\ d_s &= 2 \text{ km} \end{aligned}$$

$$x^{d,1} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 4 & 0 \\ 5 & 0 \\ 6 & 0 \end{bmatrix} \text{km} \quad x^{d,2} = \begin{bmatrix} 3 & 3 \\ 3 & 2 \\ 3 & 1 \\ 3 & 0 \\ 3 & 0 \\ 3 & -1 \\ 3 & -2 \end{bmatrix} \text{km} \quad x^{d,3} = \begin{bmatrix} -1 & -3 \\ 0 & -2.5 \\ 1 & -2 \\ 2 & -1.5 \\ 3 & -1 \\ 4 & -0.5 \\ 5 & 0 \end{bmatrix} \text{km}$$

The trajectories are a result of the vehicles flying at 100 m/s with constant heading, traveling 1 km every time step. Provide a plot of the desired and optimal trajectories for both vehicles.

Review Questions

- To code up part e), the following steps are needed.
  - Grab a copy of the receding horizon control code, along with the cost and constraint functions from the Nonlinear Programming example. Run it to make sure it works.
  - In the main nonlinear programming file, rewrite the desired trajectories, and add a variable for the number of vehicles.
  - Remove anything to do with obstacles. There are none in this problem.
  - Redefine the lower and upper bounds, and make sure to define a consistent optimization vector (x = [x1 u1, x2 u2,…], where each variable runs through each state or input, then through each timestep, then through each vehicle).
  - Modify the initial feasible solution
  - Change the way the dynamics are propagated after the optimization is complete
  - Rename the results properly for plotting.
  - In the cost function file, redefine the cost function per the required form in part c). Make sure the necessary global variables are declared here and in the main file.
  - In the constraint function file, define the dynamics for all vehicles
  - Also in the constraint function file, add in the collision avoidance constraints between each pair of vehicles

- You should now be able to run your code for the collision avoidance problem. Some debugging of constraints and costs and lower and upper bounds may be required. Check each one as you go along.

Review Questions

- This is the hard part of this question, putting all of the constraints in the right form. There are many matrices to create, and they must all be done over all time steps. Use variable bounds where you can, and the rest you must define as equality and inequality constraints. Note that nonlinear constraints must all be coded in the constraints function.

Need problem in the form

```
X = fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)
```

Full optimization vector for M vehicles and T timesteps

$$x = \begin{bmatrix} x_1^1 & u_1^1 & \cdots & x_1^M & u_1^M & \cdots & x_T^1 & u_T^1 & \cdots & x_T^M & u_T^M \end{bmatrix}^T$$

Review Questions

```
% Linear Inequality Constraints (none)
A = [];
B = [];
% Linear Equality Constraints (the initial positions of the vehicles)
Aeq = zeros(nx*nv,NT);
for i = 1:nv
    % Row index increments by nx for each new vehicle,
    % Column index increments n to select correct values of optimization
    % vector
    Aeq((i-1)*nx+1:i*nx, (i-1)*n +1:(i-1)*n+nx) = eye(nx);
    % Rows of Beq match rows of Aeq
    Beq((i-1)*nx+1:i*nx) = p0(:,i);
end

% State and input bounds, (if no bounds are given, still a good idea to
% include them but set them outside the problem scope).
LB = -500*ones(NT,1);  % Lower bounds on all variables
LB(4:n:end,1) = 0.050; % Lower bound on velocity (in km/s)
LB(5:n:end,1) = -0.03; % Lower bound on turn rate (in rad/s)

UB = 500*ones(NT,1);  % Upper bounds on all variables
UB(4:n:end,1) = 0.150; % Upper bound on velocity (in km/s)
UB(5:n:end,1) = 0.03; % Upper bound on turn rate (in rad/s)
```

Review Questions

```
% Dynamics, nonlinear equality constraints, nv vehicles times nx states
% times Tr periods (from 0 to 1, 1 to 2,... Tr to Tr+1)

Ceq = zeros(nv*nx*Tr,1);
k = 1; % Increment constraint evaluations
for t = 1:Tr
    for i = 1:nv
        xprev = x(N*(t-1)+n*(i-1)+1:N*(t-1)+n*i); % Previous state and inputs
        xcur = x(N*(t)+n*(i-1)+1:N*(t)+n*i); % Current state and inputs
        % Dynamic constraints on the three states
        Ceq(k:k+2) = xcur(1:3)-xprev(1:3)-dt*[cos(xprev(3))*xprev(4);
                                              sin(xprev(3))*xprev(4);
                                              xprev(5)];
        k = k+3;
    end
end
```

Review Problems

```
% Collision Avoidance inequality constraints
% There are nv(nv-1)/2 pairs of vehicles and Tr time steps to
% evaluate (do not include the initial positions, as they are
% already fixed).

C = zeros(nv*(nv-1)/2*Tr,1);
k = 1; % Increment constraint evaluations
for t = 2:Tr+1 % For each time step after initial
    for i = 1:nv % For each vehicle i
        for j = i+1:nv % For each other vehicle j greater than i
            xcuri = x(N*(t-1)+n*(i-1)+1:N*(t-1)+n*(i-1)+2); % Position i
            xcurj = x(N*(t-1)+n*(j-1)+1:N*(t-1)+n*(j-1)+2); % Position j
            C(k) = ds - norm(xcuri-xcurj); % min distance constraint
            k = k +1;
        end
    end
end
```

Review Questions

```
%% Cost Function
f = 0;
for t = 2:Tr+1 % For each timestep after the initial
    for i = 1:nv % For each vehicle

        % current vehicle state and inputs
        xcur = x(N*(t-1)+n*(i-1)+1:N*(t-1)+n*(i-1)+n);

        % Position error cost, quadratic distance from desired position
        f = f + beta*((xd(1,t-1,i) - xcur(1))^2+ (xd(2,t-1,i)- xcur(2))^2);

        % Turn rate control input cost, quadratic sum of turn rate inputs
        f = f + (1-beta)*xcur(5)^2;
    end
end
```

Review Questions

- Running the optimization

```
% Solve nonlinear program
    options = optimset('algorithm', 'interior-point','maxfunevals',50000);

% Limit how long your optimization can run for
    tic; % used to time the optimization, starts the timer
% The actual optimization call should not change if everything is
% defined correctly above
    [X,FVAL,EXITFLAG,OUTPUT,LAMBDA] = fmincon(@(x), cost(x), x0, A, B,
Aeq, Beq, LB, UB, @(x) constraints(x), options);
    toc; % returns the elapsed time
```

Review Questions

f) Demonstrate the full receding horizon algorithm with 3 vehicles, for a receding horizon $T_r = 5$ time steps, for the first 5 time steps of a full simulation of T = 15. Select any configuration of straight line trajectories you wish. Provide the desired trajectories and the optimal solution at each timestep for all vehicles.

- This involves modifying the code in two ways, and hopefully you coded the first changes in such a way that these modifications are easy and correct.
  - First, increase the number of timestep to more than 1.
  - Next, make sure that all of the matrices and nonlinear cost and constraint equations are being defined properly. If you are happy with all the changes, run the expanded problem for one time step and make sure the solutions make sense.
  - Then update the initial conditions and extend the feasible solution and create the receding horizon control we are interested in
- If you coded everything correctly in the previous example, this should now run the code for multiple time step, correctly updating the initial position of the vehicles after each optimization. Check that the solutions you get make sense and generate the necessary trajectories.

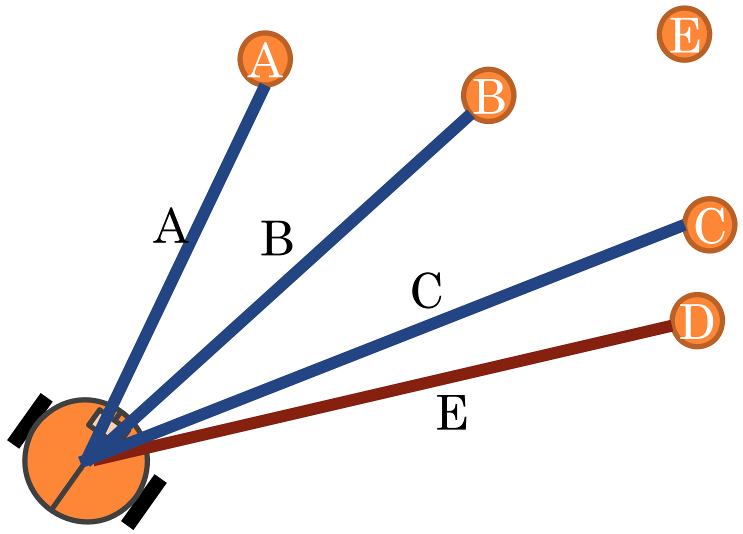
Review Questions

g) What are the run times for a single stage optimization with 2 vehicles and T = 3,4,5? Be sure to have the collision avoidance constraint be active during the calculation window. How many optimization variables are in each of the problems? What is the theoretical growth in computational complexity for nonlinear programs and how do your simulation results compare? Plot or tabulate the results.

T	Computation Time (5 timesteps)
3	0.56
4	1.10
5	1.67

Review Questions

- Random Sample Consensus (RANSAC)
  - Given a set of measurements, some of which are wrong, find the good ones
  - Can be done by looking at agreement between sets of measurements

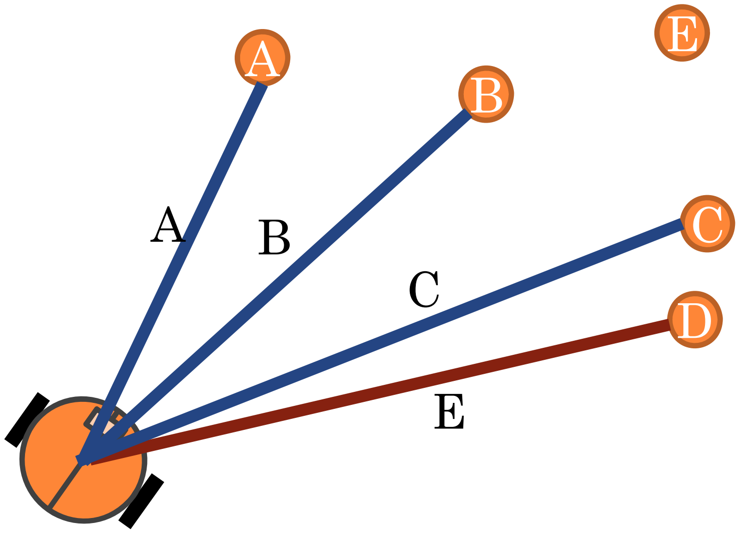


One bad correspondence

Localization:
  A,B,C,D known
  Robot pose unknown
  Bearing and range to features
  D mistakenly assigned to E

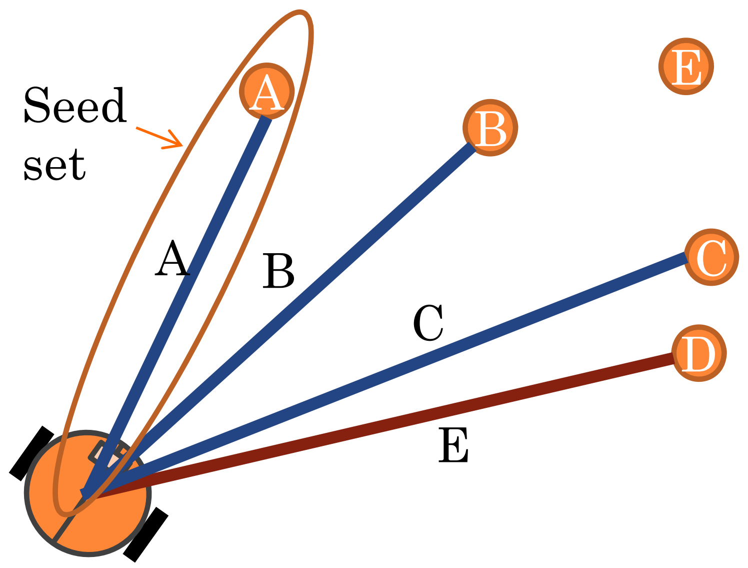
Review Questions

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



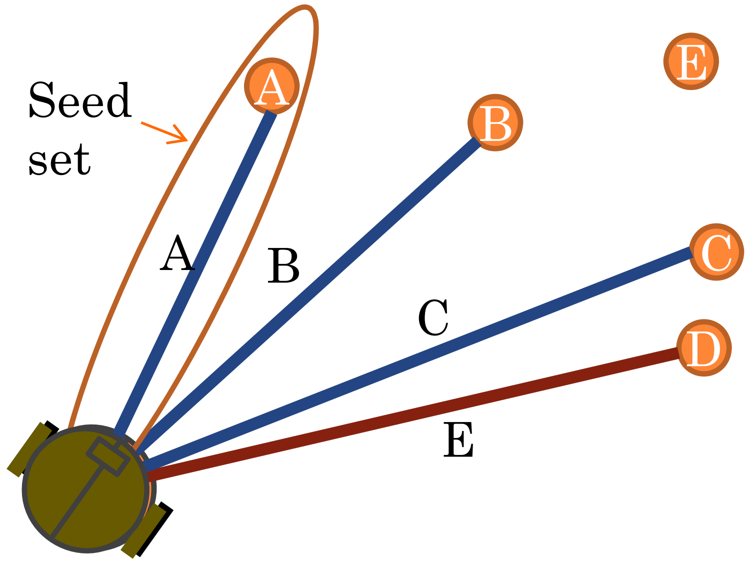
Review Questions

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



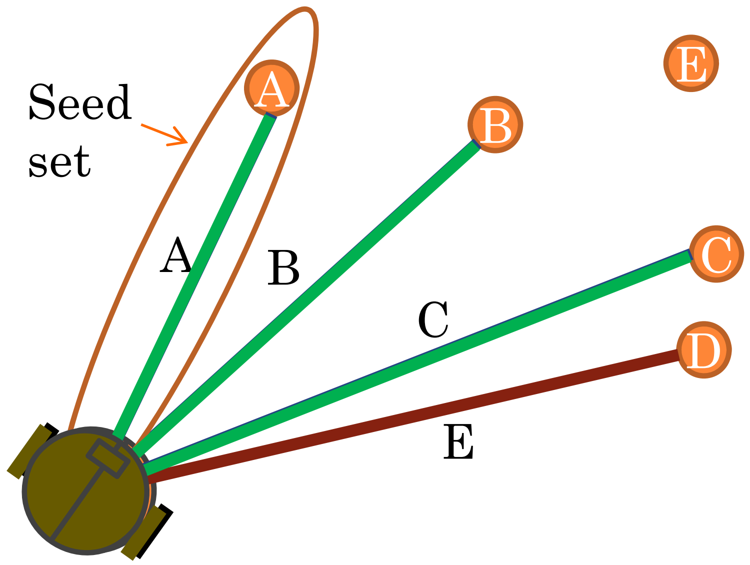
Review Questions

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



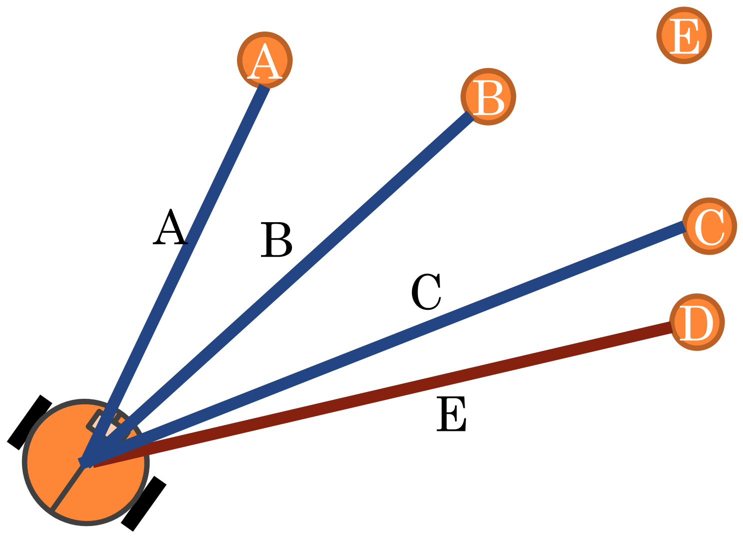
REVIEW QUESTIONS

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



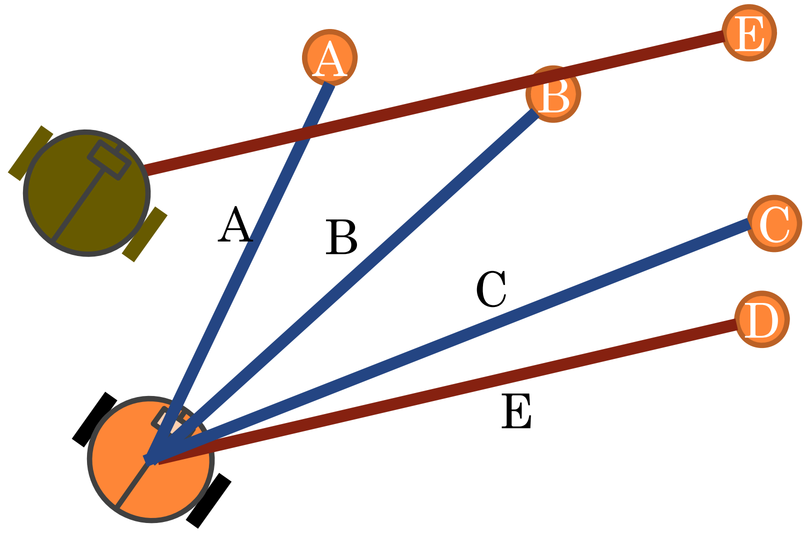
Review Questions

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



Review Questions

- Random Sample Consensus (RANSAC)
  - RANSAC picks a subset of measurements: seed
  - Then solves for robot pose
  - Then finds inlier set: those measurements that agree with pose
  - Repeat and keep largest inlier set to compute solution



Review Questions

- RANSAC Summary
  - While not out of time
    - Pick a small subset of measurement correspondences

$$y^k \subset y_t$$

    - Perform temporary measurement update with this subset

$$\mu^k = EKF(\bar{\mu}_t, y^k)$$

    - Find all features that agree with current estimate to within a fixed threshold (identify inlier set)

$$I^k = \left\{ y^k \middle| \left\| y^k - h\left(\mu^k\right) \right\| < \varepsilon \right\}$$

  - Select largest inlier set, reject all outliers

$$I^* = \max_k \left| I^k \right|$$

  - Recompute solution using the inlier set

$$\mu_t = EKF(\bar{\mu}_t, I^*)$$

Review Problems

- RANSAC Example – EKF SLAM
  - EKF SLAM with more feature
    - On average about 8 per timestep
  - Two of the measurements are selected and their indices are swapped
    - Measurements to feature i attributed to feature j
  - RANSAC uses EKF SLAM measurement update with only a single feature measurement
    - Generate new state hypothesis
    - Create inlier set with fixed threshold on innovation
    - Select largest inlier set as best one
    - Properly update EKF using best inlier set

Review Problems

```
% Confuse a random pair of current measurements
s1 = 0; s2= 0; % Indices to confused features
if(1)
    ind = find(flist);
    n_m = length(ind);
    % Remove new features to make task more manageable
    ranlist = find(flist & ~newfeature);
    ind = ranlist;
    n_m = length(ind);
    % If more than 3 features measured, swap 2 of them
    if (n_m > 3)
        s = datasample(ind,2,'Replace',false);
        s1 = s(1);
        s2 = s(2);
        tmp = y(2*(s1-1)+1:2*s1,t);
        y(2*(s1-1)+1:2*s1,t) = y(2*(s2-1)+1:2*s2,t);
        y(2*(s2-1)+1:2*s2,t) = tmp;
    end
end
```

Review Problems

```
function mu_out = EKF_meas(mu,S,y,Q)

i=1;
dx = mu(3+2*(i-1)+1)-mu(1);
dy = mu(3+2*i)-mu(2);
rp = sqrt((dx)^2+(dy)^2);

Fi = zeros(5,length(mu));
Fi(1:3,1:3) = eye(3);
Fi(4:5,3+2*(i-1)+1:3+2*i) = eye(2);
Ht = [ -dx/rp -dy/rp 0 dx/rp dy/rp;
    dy/rp^2 -dx/rp^2 -1 -dy/rp^2 dx/rp^2]*Fi;

I = y(2*(i-1)+1:2*i)-[rp;(atan2(dy,dx) - mu(3))];
I(2) = mod(I(2)+pi,2*pi)-pi;

% Measurement update
K = S*Ht'*inv(Ht*S*Ht'+Q);
mu_out = mu + K*I;
```

Review Problems

```
% Measurement RANSAC
if (length(ranlist)>3)
    NR = 10;
    ran_eps = 0.2;
    best = [];
    for k = 1:NR
        inliers = [];
        curS = datasample(ranlist,1);  % select seed set
        curY =  y(2*(curS-1)+1:2*curS,t) % get measurements
        curmu = EKF_meas(mu,S,curY,Qi); % update EKF
        for kk = 1:length(ranlist) % Calculate innovation and select inliers
            i = ranlist(kk);
            dx = curmu(3+2*(i-1)+1)-curmu(1);
            dy = curmu(3+2*i)-curmu(2);
            rp = sqrt((dx)^2+(dy)^2);
            I = y(2*(i-1)+1:2*i,t)-[rp; (atan2(dy,dx) - mu(3))];  % Innovation
            if (norm(I) < ran_eps)  % if below threshold
                inliers = [inliers, i];
            end
        end
        if (length(inliers)>length(best)) % Keep best inlier set
            best = inliers;
        end
    end
    flist = zeros(M,1);  % Update list of measurements to only include inliers
    flist(best) = 1;
 end
```

REVIEW QUESTIONS

- Linear Quadratic Regulators
  - Lecture Slides
  - Code

Review Questions

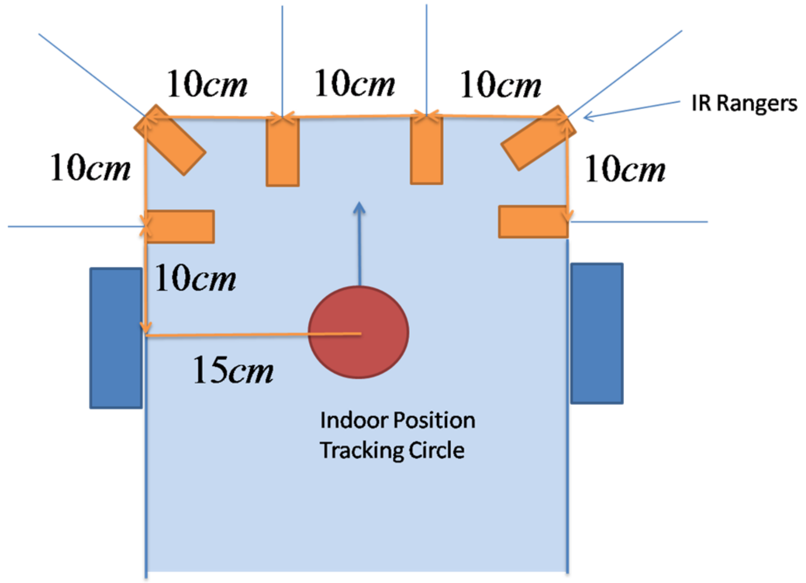
- Scan Registration
  - Lecture Slides
  - Code

Review

- FastSLAM Review
  - Lecture Slides
  - Code

Review Questions

- **The poor man's laser scanner.** In order to avoid the high cost of a full laser scanner, a group of students working on their fourth year design project would like to build an autonomous robot that relies on 6 IR rangers for mapping of an indoor environment. The robot is aided by an indoor positioning system which provides *x, y, Θ* at 1 Hz. They decide to use the lab robots for their project. The 6 IR rangers are positioned on the front of the robot, with two pointing forward, spaced apart by 10 cm, two at the corners pointing at ± 45° and two at ± 90° as depicted below:


10cm
10cm
10cm
IR Rangers
10cm
10cm
10cm
15cm
Indoor Position Tracking Circle