FINAL PROJECT – Due June 17
EBS 289K: Topics in Agricultural Robotics and Automation

A tree and plant nursery has a field block with *K* rows of trees. Tree rows are aligned in a North-South direction, are 20 m long, and are spaced 3 m apart from each other. The southern mid-point of the west row is at 20m, 20m. Trees are of various ages (trunk diameters). Some trees may be missing from the row because they were removed for sale, or due to disease or other damage. You need to program a mobile robot equipped with a GPS, a digital compass, and a 2D laser scanner to traverse the field block and *count the number of trees in each row*. You also need to *estimate the diameter of the trunk* of each tree, so that management can decide which trees are ready for the market.

Tasks:

1) Estimate the odometry error covariance matrix (2x2), and the GPS and compass error covariance matrix (3x3). You can use the true states to do so.
2) Implement an EKF filter for robot state estimation using the odometry model described in class for prediction, and the GPS-compass sensor. Develop also a filter for the laser.
3) Traverse the orchard block (access the bitmap only through your sensors).
4) Detect and localize trees; estimate their diameters.
5) Write a file with your output (see format below).
6) Write a script to produce bitmaps from output file.
7) Write a script to calculate error histograms of tree positions and diameters between a test bitmap (ground truth synthetic environment) and the output of task 5 (perceived environment). Extract descriptive statistics from histograms and report them: mean, standard deviation, RMS, min, max, 95th percentile.

INPUTS (see Appendix)

1) A Matlab function robot_odo.p that models a mobile robot as a bicycle model with steering and speed first order closed-loop dynamics. The function integrates the model for time DT with integration interval dT (global variables), and returns the new state (vector [x y θ γ v]) and a noisy estimate of the odometry, i.e., a vector $[\delta_d+v_d\ \delta_\theta+v_\theta]$.
2) A Matlab function LaserScannerNoisy.p provided by the instructor. This function will return range measurements that are noisy; they contain white gaussian noise and spike-type noise (due to laser reflections). When a reading is out of range, *Inf* is returned.
3) A Matlab function GPS_CompassNoisy.p provided by the instructor. This function receives as arguments the true pose of your robot and returns a noisy pose, thus emulating real sensors. Also, the GPS and compass provide

fresh measurements every 1 second. However, the control system operates at a rate of DT=100 ms. So, readings are available once every 10 control intervals.

4) A Matlab function that generates a 2D grid/bitmap with trees (ground truth). Your robot is not allowed to access this grid directly, but ONLY through its lidar sensor. The grid is provided for testing, and you are encouraged to test your code with your own test bitmaps.

OUTPUT

After the robot finishes its travel, the output should be a file that looks like this: Row number followed by index of each tree detected, tree coordinates (x, y) and diameter (d). E.g.:

1
1, x1, y1, d1
2, x2, y2, d2
3, x3, y3, d3
N, x_N, y_N, d_N

2
1, x1, y1, d1
2, x2, y2, d2
3, x3, y3, d3
M, x_M, y_M, d_M

K
1, x1, y1, d1
2, x2, y2, d2
3, x3, y3, d3
M, x_M, y_M, d_M

ROBOT

You have an Ackermann steered vehicle with wheelbase L=3 m, width=2 m, and maximum steering angle 55^o. The steering lag is 0.1 s and the velocity lag is 0.2 s. The robot is positioned initially at 0,0 facing 90^o North. You can set your own robot speed.

The 2D lidar has 20 m maximum range, 180^o scanning angle and 0.125^o angular resolution. You must use the true pose ONLY for vehicle motion and sensor simulation. All your control, sensing and processing algorithms must rely ONLY on noisy data from the odometry, lidar and GPS-compass sensors.

EVALUATION

To evaluate your code, I will use it with my own nursery grids. **Beware**: Tall grass or twigs near the ground may manifest themselves as speckles (pixels) of noise in the grid that I will use! Simulate this to test your algorithm's robustness.

APPENDIX

```
function [q_true_next, odo] = robot_odo(q_true, u, umin, umax,Qmin,
Qmax, L, tau_gamma, tau_v)
% model of a vehicle with closed loop steering and velocity control
% the combined effect of steering/vehicle inertia and control is
% a first order system for steering (tau_phi) and velocity (tau_v)
% state is
% q(1) -> x
% q(2) -> y
% q(3) -> theta (orientation in world frame)
% q(4) -> gamma (steering angle)
% q(5) -> linear velocity

% inputs are:
% u(1) -> desired steering angle
% u(2) -> desired linear velocity
%
% the model returns the next state q1 and noisy odometry, i.e.,
% distance traveled in DT in odo(1) and angle change in DT in odo(2).

function p = laserScannerNoisy(angleSpan, angleStep, rangeMax, Tl, map,
Xmax, Ymax)
% function assumes a laser scanner with a pose in world coordinates
% defined by Tl (from q_true). It shoots rays from
% -angleSpan/2 to +angleSpan/2 with step angleStep.
% Given a map (occupancy grid with obstacles) with origin at (0,0) and
% upper corner (Xmax, Ymax), the result is ray angles (p(:,1)) and
% noisy ranges (p(:,2)).

function [ x_n, y_n, theta_n ] = GPS_CompassNoisy( x, y, theta )
% function accepts as arguments the true pose of the robot (from
q_true) and returns a noisy measurement of the pose. Angle is in
radians.
```