map = helperCreateBinaryOccupancyMap;

figure

show(map)

title('Warehouse Floor Plan With Obstacles and AGV')

pose = [5 40 0];

helperPlotRobot(gca, pose);

lidar = rangeSensor;

lidar.HorizontalAngle = [-pi/2 pi/2];

lidar.Range = [0 5];

load waypoints.mat

traj = waypointsMap;

Vehiclepose = traj(350, :);

[ranges, angles] = lidar(Vehiclepose, map);

scan = lidarScan(ranges, angles);

plot(scan)

title('Ego View')

helperPlotRobot(gca, [0 0 Vehiclepose(3)]);

display = helperVisualizer;

hRobot = plotBinaryMap(display, map, pose);

figure

detAxes = gca;

title(detAxes,'Define Detection Area')

axis(detAxes, [-2 10 -2 4])

xlabel(detAxes, 'X')

ylabel(detAxes, 'Y')

axis(detAxes, 'equal')

grid(detAxes, 'minor')

t = linspace(-pi/2, pi/2, 30)';

colors = [1 1 1; 1 1 0; 1 0.5 0; 1 0 0];

radius = [5 2 1];

detAreaHandles = repmat(images.roi.Polygon, [3 1]);

pos = [cos(t) sin(t)] \* radius(1);

pos = [0 -2; pos(14:17, :); 0 2];

detAreaHandles(1) = drawpolygon(...

'Parent', detAxes, ...

'InteractionsAllowed', 'reshape', ...

'Position', pos, ...

'StripeColor', 'black', ...

'Color', colors(2, :));

pos = [cos(t) sin(t)] \* radius(2);

pos = [0 -1.5; pos(12:19, :); 0 1.5];

detAreaHandles(2) = drawpolygon(...

'Parent', detAxes, ...

'InteractionsAllowed', 'reshape', ...

'Position', pos, ...

'StripeColor', 'black', ...

'Color', colors(3, :));

pos = [cos(t) sin(t)] \* radius(3);

pos = [0 -1; pos(10:21, :); 0 1];

detAreaHandles(3) = drawpolygon(...

'Parent', detAxes, ...

'InteractionsAllowed', 'reshape', ...

'Position', pos, ...

'StripeColor', 'black', ...

'Color', colors(4, :));

axesDet = gca;

[detArea,bbox] = helperSaveDetectionArea(axesDet, detAreaHandles);

ax3 = getDetectionAreaAxes(display);

h = imagesc(ax3, [bbox(1) (bbox(1) + bbox(3))], ...

-[bbox(2) (bbox(2) + bbox(4))], ...

detArea);

colormap(ax3, colors);

plotObstacleDisplay(display)

%Red — Collision is imminent

%Orange — High chance of collision

%Yellow — Apply caution measures

for ij = 27:size(traj, 1)

currentPose = traj(ij, :);

[ranges, angles] = lidar(currentPose, map);

scan = lidarScan(ranges, angles);

cart = scan.Cartesian;

cart(:, 3) = 0;

pc = pointCloud(cart);

minDistance = 0.9;

[labels, numClusters] = pcsegdist(pc, minDistance);

updateMapDisplay(display, hRobot, currentPose);

plotLidarScan(display, scan, currentPose(3));

if exist('sc', 'var')

delete(sc)

clear sc

end

nearxy = zeros(numClusters, 2);

maxlevel = -inf;

for i = 1:numClusters

c = find(labels == i);

xy = pc.Location(c, 1:2);

a = [xy(:, 1) xy(:, 2)] - repmat(bbox([1 2]), [size(xy, 1) 1]);

b = repmat(bbox([3 4]), [size(xy, 1) 1]);

xy\_org = a./b;

idx = floor(xy\_org.\*repmat([size(detArea, 2) size(detArea, 1)],[size(xy\_org, 1), 1]));

validIdx = 1 <= idx(:, 1) & 1 <= idx(:, 2) & ...

idx(:, 1) <= size(detArea, 2) & idx(:, 2) <= size(detArea, 1);

cols = idx(validIdx, 1);

rows = idx(validIdx, 2);

levels = double(detArea(sub2ind(size(detArea), rows, cols)));

if ~isempty(levels)

level = max(levels);

maxlevel = max(maxlevel, level);

xyInds = find(validIdx);

xyInds = xyInds(levels == level);

nearxy(i, :) = helperNearObstacles(xy(xyInds, :));

else

nearxy(i, :) = helperNearObstacles(xy);

end

end

switch maxlevel

case 3

circleDisplay(display, colors(4, :))

case 2

circleDisplay(display, colors(3, :))

case 1

circleDisplay(display, colors(2, :))

otherwise

circleDisplay(display, [])

end

for i = 1:numClusters

sc(i, :) = displayObstacles(display, nearxy(i, :));

end

updateDisplay(display)

pause(0.01)

end

function map = helperCreateBinaryOccupancyMap()

map = binaryOccupancyMap(100, 80, 1);

occ = zeros(80, 100);

occ(1, :) = 1;

occ(end, :) = 1;

occ([1:30, 51:80], 1) = 1;

occ([1:30, 51:80], end) = 1;

occ(40, 20:80) = 1;

occ(28:52, [20:21 32:33 44:45 56:57 68:69 80:81]) = 1;

occ(1:12, [20:21 32:33 44:45 56:57 68:69 80:81]) = 1;

occ(end-12:end, [20:21 32:33 44:45 56:57 68:69 80:81]) = 1;

setOccupancy(map, occ);

helperAddObstacle(map, 5, 5, [10, 30]);

helperAddObstacle(map, 5, 5, [20, 17]);

helperAddObstacle(map, 5, 5, [40, 17]);

end

function helperAddObstacle(map, obstacleWidth, obstacleHeight, obstacleLocation)

values = ones(obstacleHeight, obstacleWidth);

setOccupancy(map, obstacleLocation, values)

end