

**Uncertainty**

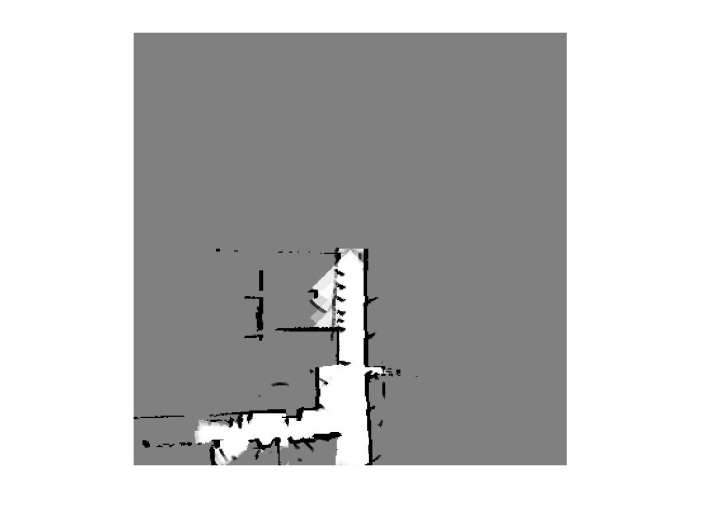
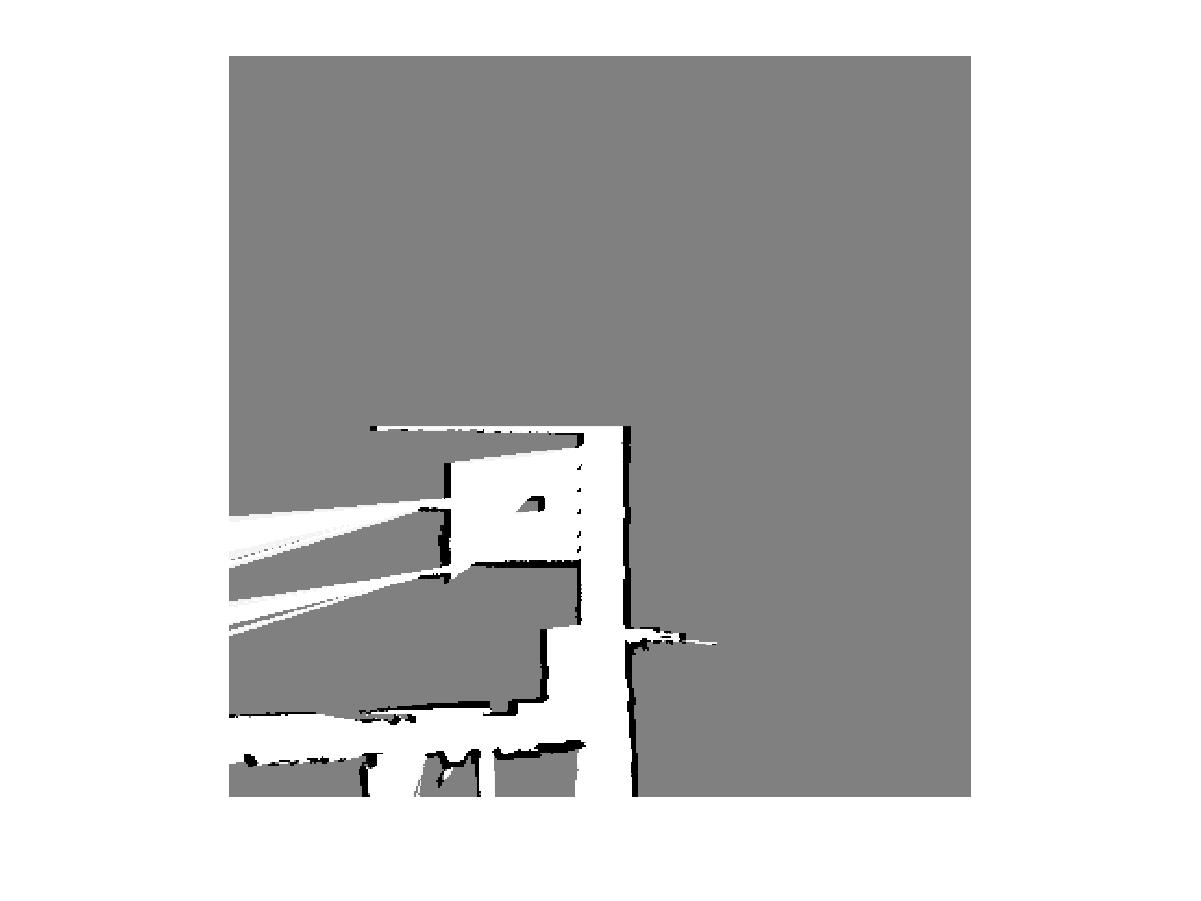
Sonar sensor is more uncertain, or precisely the precision is limited to 15 degree of the cone. Laser sensor however, have higher precision (180/361=0.5degree). Yet, the laser have feature that laser beams may penetrate transparent stuffs, e.g. glass windows. Therefore, if the map is for obstacle avoidance purpose, this feature is not a good thing because the robot won’t detect the glasses and bump into it. Also, reflective objects, like mirrors, may confuse the robot that there are additional spaces behind the mirror. On the other hand, sonar generally has the ability to detect physical obstacle as long as the incident angle is almost orthogonal with the surface and the surface is not too reflective and absorptive. So the two sensors have complementary features.

In the inverse sensor model, because of higher uncertainty of sonar sensor, so the *l*occ for sonar is set to log(0.7/0.3) and log(0.95/0.05) for laser.

**Fusion**

To utilize the good feature from two sensors, that is to fuse the information. I fuse the map using max operation. That is we intend to believe the space is occupied once a sensor detect that. Therefore, the map is more useful for navigation purpose.

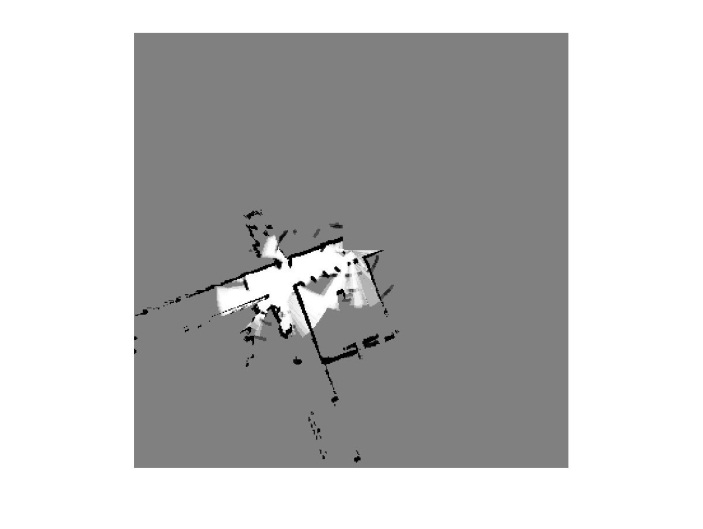
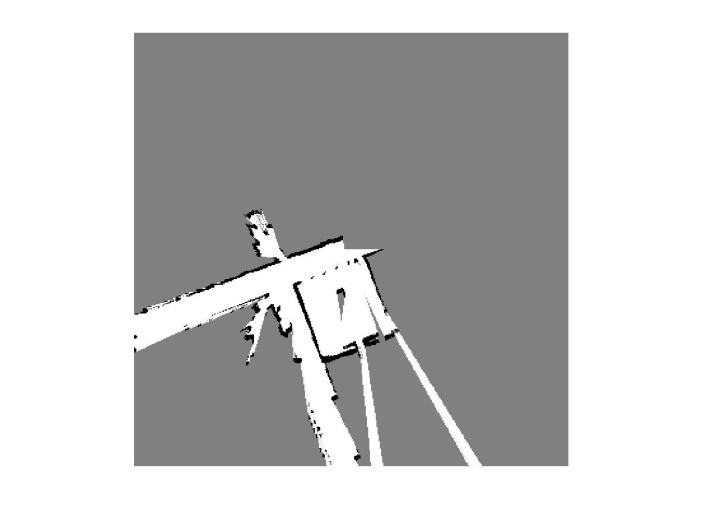
**Test1**



(a) Pure laser (b) Pure sonar (c) Fusion

Consider the part circled in red, laser penetrates the transparent wall – probably are glass windows – and map the surroundings outside. However, sonar receiver capture the signal bounced back from the transparent wall but there are still some signals that penetrate the wall. Beside the lower precision of sonar cones, from the previous homework, when the sonar beam is not parallel to the wall, the received value is smaller than the groundtruth in range. I guess this phenomenon account for several obstacles near the wall but jagged into the clear path.

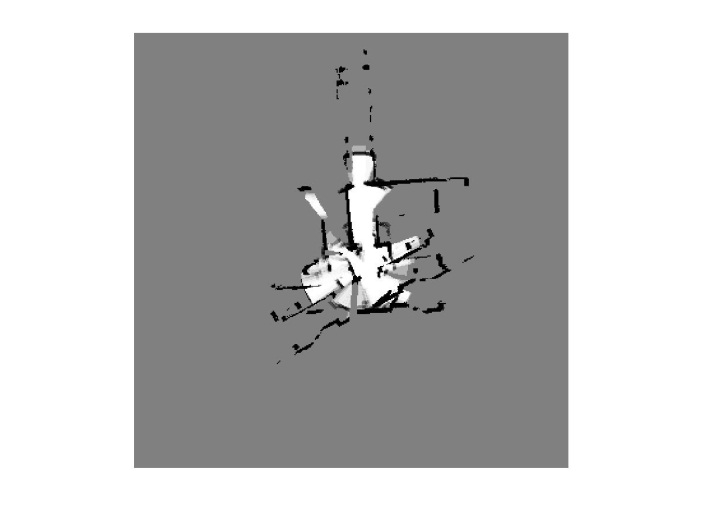
**Test2**



(a) Pure laser (b) Pure sonar (c) Fusion

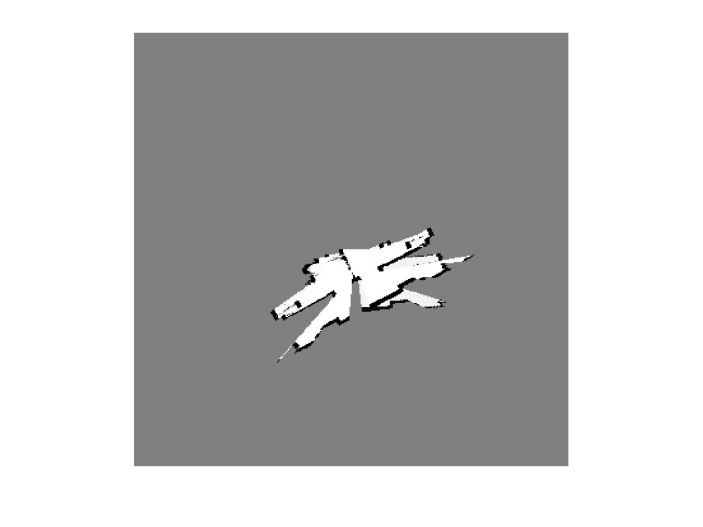
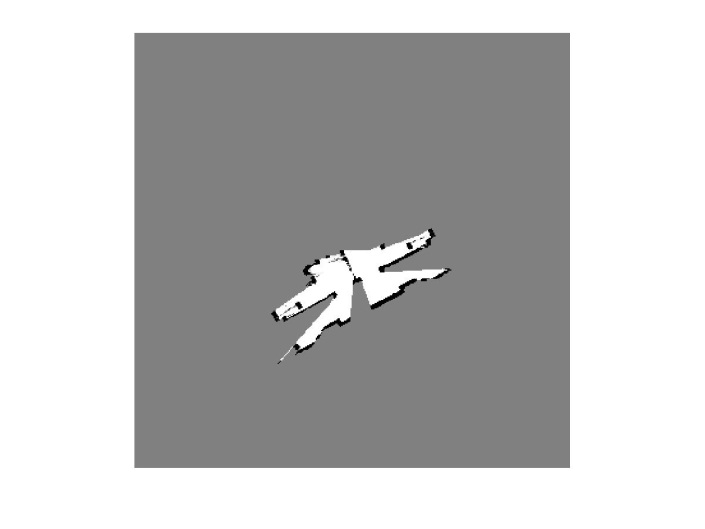
Except the findings in test1, we can observe the part circled in red in test2. The corner in the laser data is very precise, but for sonar sensor, the sonar beam has a high chance that the beam can’t successfully bounce back. Also, consider the effective range. Laser can more efficiently generate maps for much farther away environment. Because in this case the robot didn’t move around as far as test1, therefore, the unoccupied region is limited to Sonar in the fusion map.

**Test3**



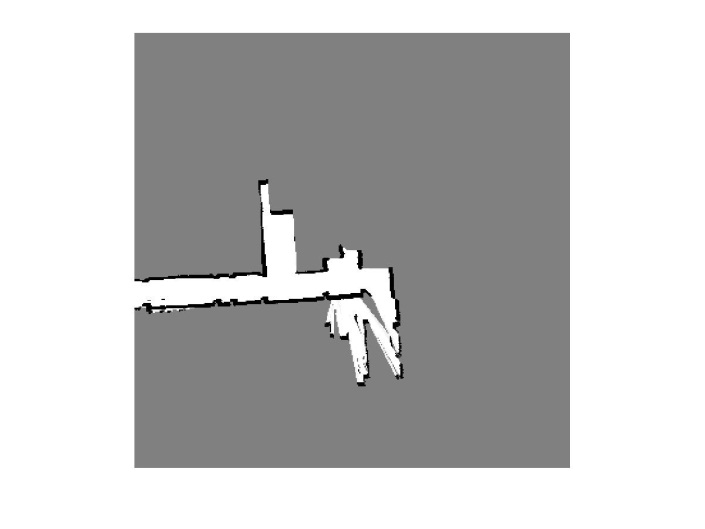
(a) Pure laser (b) Pure sonar (c) Fusion

Regarding Test3, at the beginning, the mapping process seems to generate a symmetric environment along the redline. Below is a better illustration (d) that shows the symmetric shape when timestamp = 135. In addition, the robot seems to slip at timestamp 207 (e) because the shape registered after that rotates in a specific angle.



(d) (e)

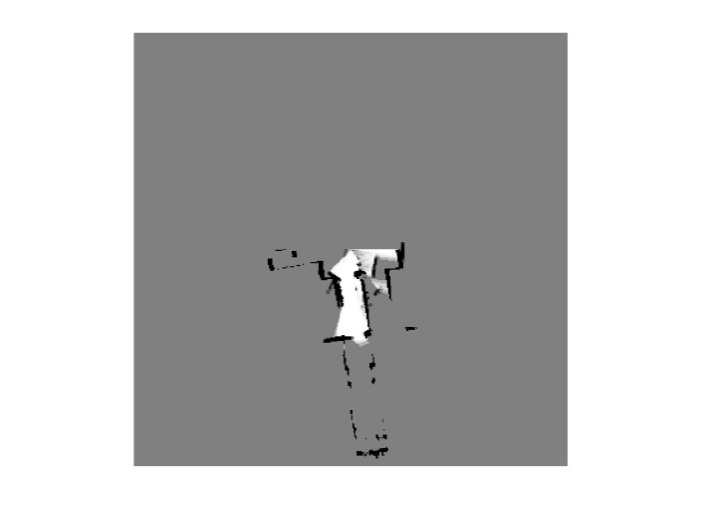
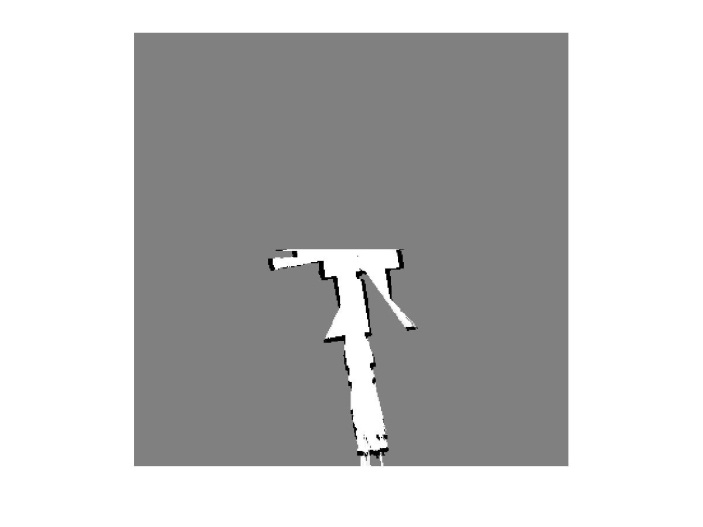
**Test4**



(a) Pure laser (b) Pure sonar (c) Fusion

As in Test1, the part in the red circle probably contains laser-penetrable boundary, from which the sonar beams are able to receive responses. Also, the jagged wall in sonar case is mainly caused by the large cone size of a sonar sensor.

**Test5**



(a) Pure laser (b) Pure sonar (c) Fusion

Consider the portion circled in red, the sonar seems to detect an obstacle that is not detected by laser range finder. Probably, this is result of different heights at which the laser and sonar was mounted. Therefore, the fused map has the benefit of detecting obstacles at different height.

In sum, laser and sonar have different features in map building. Laser is generally superior in building high precision map, but the Achilles heels are the reflective and penetrable objects. Sonar has lower precision due to the coarse cone size, but it has a high chance to complement the shortcomings using laser scanner. Finally the fused map combines the information from both sensors. The fusing method favors tend to believe the space is occupied if one of the sensors detect so such that the robot won’t bump into obstacles that can only be detected from either one.

Implementation

function hw7

hw7\_run('test1');

hw7\_run('test2');

hw7\_run('test3');

hw7\_run('test4');

hw7\_run('test5');

end

function hw7\_run(testcase)

global x\_res y\_res t\_res x\_bound y\_bound x\_axis y\_axis x\_len y\_len t\_axis Px t\_lbound t\_ubound

x\_res = 100;

y\_res = 100;

%t\_res = 2\*pi/360;

x\_bound = 20000;

y\_bound = 20000;

%t\_lbound = 0;

%t\_ubound = 2\*pi;

% system initial

x\_axis = -x\_bound:x\_res:x\_bound;

y\_axis = -y\_bound:y\_res:y\_bound;

%t\_axis = t\_lbound:t\_res:t\_ubound;

x\_len = size(x\_axis,2);

y\_len = size(y\_axis,2);

%t\_len = size(t\_axis,2);

Fmap = ones(x\_len,y\_len); %% fused likelihood map

Smap = zeros(x\_len,y\_len); %% sonar likelihood map

Lmap = zeros(x\_len,y\_len); %% laser likelihood map

L = readL(strcat(testcase, '\laser.txt'));

O = readO(strcat(testcase, '\od.txt'));

S = readS(strcat(testcase, '\sonar.txt'));

for t = 1:2:size(L,1)

if t>1

if (norm(O(t,1:3) - O(t-1,1:3)) < 10)

continue;

end

end

% L(t,1:361)

% S(t,1:8)

% O(t,1:3)

t

for x = x\_axis

for y = y\_axis

if norm([x,y] - O(t,1:2)) < 80000

% do laser map

Lmap(indx(x),indy(y)) = Lmap(indx(x),indy(y)) + inverse\_sensor\_model\_laser([x,y], O(t,1:3), L(t,1:361)) - 0;

end

if norm([x,y] - O(t,1:2)) < 5000

% do laser map

Smap(indx(x),indy(y)) = Smap(indx(x),indy(y)) + inverse\_sensor\_model\_sonar([x,y], O(t,1:3), S(t,1:8)) - 0;

end

end

end

h1 = figure;

h2 = figure;

h3 = figure;

figure(h1)

imshow(1./(1+exp(Lmap)));

figure(h2)

imshow(1./(1+exp(Smap)));

figure(h3)

imshow(min( 1./(1+exp(Smap)), 1./(1+exp(Lmap)) ));

%pause(0.5);

print (h1,'-djpeg',strcat(testcase,'MapL',sprintf('%03d',t),'.jpg'));

print (h2,'-djpeg',strcat(testcase,'MapS',sprintf('%03d',t),'.jpg'));

print (h3,'-djpeg',strcat(testcase,'MapF',sprintf('%03d',t),'.jpg'));

close(h1);

close(h2);

close(h3);

end

end

function index = ind(x,y)

global x\_res y\_res t\_res x\_bound y\_bound x\_axis y\_axis x\_len y\_len t\_axis Px t\_lbound t\_ubound

%size(Px)

%floor((x+x\_bound)/x\_res)+1

%floor((y+y\_bound)/y\_res)+1

%floor((t-t\_lbound)/t\_res)+1

%size(Px)

index = sub2ind(size(Px), floor((x+x\_bound)/x\_res)+1, floor((y+y\_bound)/y\_res)+1);

end

function index = indx(x)

global x\_bound x\_res

index = floor((x+x\_bound)/x\_res)+1;

end

function index = indy(y)

global y\_bound y\_res

index = floor((y+y\_bound)/y\_res)+1;

end