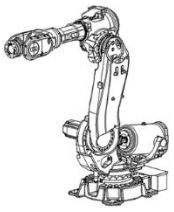
Fernando
AUAT CHEEIN



fernando.auat@usm.cl
http://profesores.elo.utfsm.cl/~fauat/

Tercera Guía: grupal

Entregue la guía en formato IEEE. El puntaje es el mismo para todos los ejercicios. Indique con detalle lo que fue realizado. Incluya análisis cuando estime conveniente. Esta guía será mostrada al profesor o al ayudante en reunión. Recuerde redactar su guía para pueda ser revisada. No incluya resultados sin explicaciones o análisis (causal de nota **0**). Recuerde que esta es una asignatura de postgrado. El código se da en Matlab, pero puede migrar a Python o Coppelia si lo estima pertinente.

Resultado de Aprendizaje: Analiza información y la procesa bajo la presencia de incertidumbre.

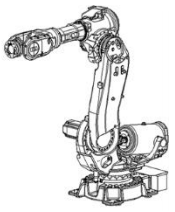
1) Sea el archivo "simulador.m" provisto en la asignatura. Diseñe un algoritmo tal que sea capaz de adquirir un mapa del entorno a medida que el robot se mueve. Para ello, considere que las características que observa el robot, son características puntuales. Al final de esta guía se incluye una *brief* de uso del simulador.

Suponga que el LiDAR del robot simulado, posee una matriz de covarianza fija, que no depende del tiempo ni del rango o ángulo de medición.

- Proponga una estrategia para determinar los puntos representativos de los postes (dicha estrategia no debe ser solamente el "promedio". Fusione la información disponible, aplique algoritmos de *clustering*, etc.).
- El programa debe entregar una matriz M con la ubicación de todas las características puntuales, y una matriz con la covarianza de cada característica y de M.
- Deje explicitadas todas las fórmulas y en su informe, entregue también el archivo *.m que haya desarrollado.

2) Ídem al ejercicio anterior, pero suponga que el robot si tiene error de odometría. Haga un análisis de consistencia de las características detectadas (suponga que sabe de antemano la posición real de cada característica, para así poder calcular el error).

3) Usando el archivo "simulador.m", proponga (analíticamente) e implemente (en software) un sistema de localización (usando mínimos cuadrados, máxima verosimilitud, máximo a posteriori o Filtro de Kalman), basado únicamente en las características que detecta el LiDAR del robot. No dispone de la información de la pose del robot, excepto la de la pose inicial, sin embargo, sabe exactamente dónde están las características en el mapa.



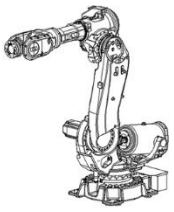
fernando.auat@usm.cl
http://profesores.elo.utfsm.cl/~fauat/

Incluya todo desarrollo analítico y de programación en su informe. Analice cómo evoluciona la incertidumbre asociada a la pose del robot. Realice un análisis de consistencia de la pose del robot (suponiendo que conoce perfectamente dónde está el robot en cada instante de tiempo).

4) Ídem al ejercicio anterior, pero no sabe dónde están las características.

Brief

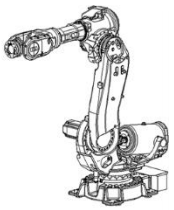
Fernando
AUAT CHEEIN



fernando.auat@usm.cl
http://profesores.elo.utfsm.cl/~fauat/

```
figure(1), hold on
clear all,
clc,
xlim([-5 10]);
ylim([-5 10]);
global maxRange;
maxRange = 10; %Distancia máxima del laser
% axis equal

pasoTiempo = 0.1; %Tiempo de Sampling de mi sistema


%Inicializo Robots

robot.x = 0;
robot.y = 0; %es el robot segun odometria
robot.tita = 0;

robot2 = robot; %es el robot real
%ambos robots empiezan en el mismo lugar

%Unico landmarks. Con tecla enter, finalizo ubicación. Landmarks son
%guaradadas en matriz M, que contiene las posiciones reales [x,y] de cada
%landmark.

M = ginput();
plot(M(:,1),M(:,2),'ob'); %grafico landmarks
[Caras,Vertices] = conversionLandmarks(M);

%Genero el camino que quiero que siga el robot, entre las landmarks. El
%camino solo va de izquierda a derecha

[X,Y] = ginput;
xmax = max(X);
xmin = min(X);

xx = xmin:0.01:xmax;

yy = interp1(X,Y,xx,'spline');
```

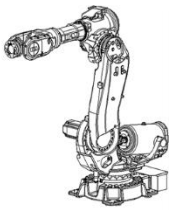
Fernando
AUAT CHEEIN



fernando.auat@usm.cl
http://profesores.elo.utfsm.cl/~fauat/

```
yy = interp1(X,Y,xx,'spline');

CaminoReferencia = [xx' yy'];
plot(xx',yy','-r');

%Inicializo Robots en el punto de partida del camino generado

robot.x = X(1);
robot.y = Y(1);
robot.tita = atan2(Y(2) - Y(1), X(2) - X(1)); %es la pose del robot según la odometría. Sin embargo, tener presente que el
%robot "cree" que tiene su pose sin error.

robot2 = robot; %es el robot real
%ambos robots empiezan en el mismo lugar

%Grafico ambos robots (odométrico y real)
H1 = plotRobot(robot);      % Robot odometrico
H2 = plotRobot2(robot2);    % Robot real
%Genero un bucle para controlar el camino del robot

for cont = 2:length(xx)-1
    delete(H1);
    delete(H2);
    %Sensado de Postes
    Laser = MedicionConLaserRapido(Vertices, Caras, robot2);
    H3 = plotLaser(Laser,robot2);
    BuscoPostes = [];

    BuscoPostes = ACA DEBEN PONER SU SOFTWARE DE BUSQUEDA DE TRONCOS

    if(~isempty(BuscoPostes))
        plot(BuscoPostes(:,1),BuscoPostes(:,2),'gx');

    end
```

Fernando
AUAT CHEEIN



fernando.auat@usm.cl
http://profesores.elo.utfsm.cl/~fauat/

```
    %referencias para el controlador (obviar esta parte) y obtención de las
    %señales de control
    Q = .01*eye(3,3);
    H = [cos(robot.tita) 0;sin(robot.tita) 0; 0 1];
    xref = xx(cont);
    yref = yy(cont);
    titaref = atan2(yy(cont+1) - robot.y,(xx(cont+1) - robot.x));
    deltax = (xref-robot.x)/pasoTiempo + 0.1*(robot.x - xx(cont-1))/pasoTiempo;
    deltay = (yref-robot.y)/pasoTiempo + 0.3*(robot.y - yy(cont-1))/pasoTiempo;
    sigma1 = (Q(1,1))^2;
    sigma2 = (Q(2,2))^2;
    sigma3 = (Q(3,3))^2;
    Control(2) = (titaref-robot.tita)/pasoTiempo;
    Control(1) = (2*deltax*sigma2^2*cos(robot.tita))/(sigma2^2*cos(2*robot.tita) - sigma1^2*cos(2*robot.tita) + sigma1^2 + sigma2^2) + (2*deltay*sigma1^2*sin(robot

    if Control(1) > 0.3
        Control(1) = 0.3;
    end
    if abs(Control(2)) > 1
        if (Control(2) <= 0)
            Control(2) = -1;
        else
            Control(2) = 1;
        end
    end
    V = Control(1);
    W = Control(2);
    %FIN SEÑALES DE CONTROL


    %ODOMETRIA "PURA"
    robot.x = robot.x + pasoTiempo*V*cos(robot.tita);
    robot.y = robot.y + pasoTiempo*V*sin(robot.tita);
    robot.tita = robot.tita + pasoTiempo*W;

    %ODOMETRIA "RUIDOSA"
    robot2.x = robot2.x + pasoTiempo*V*cos(robot2.tita)+0.001*rand(1,1);
    robot2.y = robot2.y + pasoTiempo*V*sin(robot2.tita)+0.001*rand(1,1);
    robot2.tita = robot2.tita + pasoTiempo*W+0.01*rand(1,1)*W;

    %PLOTEA
    H1=plotRobot(robot);
    H2=plotRobot2(robot2);
    pause(pasoTiempo);
    delete(H3);
end
```