Robot report assignment 3

11610303 Huang Yu'an

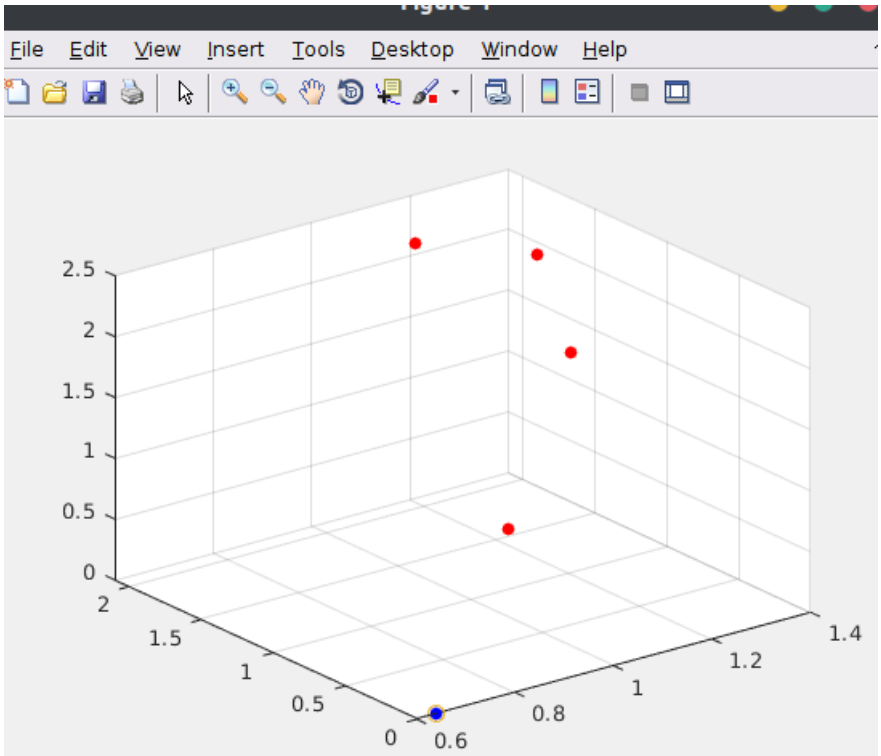
Problem 1:
If there are 100 lines in the grating, what is the smallest detectable change in motor-shaft angle?

360 degree / 100 = 3.6 degree

Problem 2:
Explain how to determine the rotation directions if the following encoders are used. List two concerns while choosing an encoder.

A and B 's output signal has phase differences. We can determined rotation direction by the analysis of it.
The event: "A is head of B and B is head of A" can represent different meaning. When choosing decoder, signal intensity and clock cycle should be think about.

Problem 3:
Simulate the process of localization with GPS signals. When sender-receiver clocks are either synchronized or not synchronized, how many satellites are needed to achieve 3D accurate positions, respectively? (HINT: use MATLAB fsolveto estimate the target location.



Blue point is the user position, red point is satellite position. Pink ring is the estimate position. The GPS problem is a equation solve problem, we can use fsolve to solve it. Those below are my code.

```
clear;
clc;

SatellitePosition=[
    1  1.75   0.73;
    1.21  -0.97  2.1;
    1.33 -1.81  1.43;
    1.40 -1.3  1.9
    ];
UserPosition=[0.64   0    0 ];   %用户真实位置（注意：定位程序并未用到此参数）

scatter3(SatellitePosition(:, 1), SatellitePosition(:, 3), SatellitePosition(:, 3), 'r', 'filled');
hold on;
scatter3(UserPosition(:, 1), UserPosition(:, 3), UserPosition(:, 3), 'b', 'filled');

[CalUserPosition, OK]=CalculateUserPosition2();
scatter3(CalUserPosition(1), CalUserPosition(2), CalUserPosition(3), 60);
```

```
function Pr=CalculatePseudoRange(SatellitePosition,UserPosition)
    c = 3e5;
    DeltaT = 1e-5;
    VisSatNum = 4;
    Pr=ones(1,VisSatNum); %求解用户接收机收到的伪距信息

        for k=1:VisSatNum
            Pr(k)=c*DeltaT;
            tempS = 0;
            for m=1:3
                 tempS = tempS + (UserPosition(m)-SatellitePosition(k, m))*(UserPosition(m)-SatellitePosi
tion(k, m));
            end
            Pr(k)=sqrt(tempS) + Pr(k);
        end

end

function P=myfun(pos)
    P = [0, 0, 0, 0];

    SatellitePosition=[
    1  1.75   0.73;
    1.21  -0.97  2.1;
    1.33 -1.81  1.43;
    1.40 -1.3  1.9];
    UserPosition=[0.64   0    0 ];   %用户真实位置（注意：定位程序并未用到此参数）
    c = 3e5;

    Prange =CalculatePseudoRange(SatellitePosition, UserPosition);
    len = length(Prange);
    for k=1:len
        P(k)=c*pos(4) - Prange(k);
        tempS = 0;
        for m=1:3
            tempS = tempS + (pos(m)-SatellitePosition(k, m))*(pos(m)-SatellitePosition(k, m));
        end
        P(k)=sqrt(tempS) + P(k);
    end
end

function [CalUserPosition, CalculateOK] = CalculateUserPosition2()
    CalculateOK = 1;
    CalUserPosition = fsolve(@myfun,[0,0,0,0]',optimset('Display','off'));
end
```

Problem 4:
Simulate the process of mapping of a room by using a moving range sensor which knows its location accurately (randomly walking, or moving along a circle). (Plot the geometry of a room and boxes first; select a motion trajectory of the robot; simulate the range sensor with a line (or a number of lines); compute the intersection points of range sensors and geometry of the room and boxes)

```
sim = ExampleHelperRobotSimulator('simpleMap');
setRobotPose(sim, [2 3 -pi/2]);

enableROSInterface(sim, true);

sim.LaserSensor.NumReadings = 50;

scanSub = rossubscriber('scan');

[velPub, velMsg] = rospublisher('/mobile_base/commands/velocity');

tftree = rostf;

pause(1);

path = [2, 3; 3.25 6.25; 2 11; 6 7; 11 11; 8 6; 10 5; 7 3; 11 1.5];
```

```
plot(path(:,1), path(:,2),'k--d');

controller = robotics.PurePursuit('Waypoints', path);
controller.DesiredLinearVelocity = 0.4;

controlRate = robotics.Rate(10);

goalRadius = 0.1;
robotCurrentLocation = path(1,:);
robotGoal = path(end,:);
distanceToGoal = norm(robotCurrentLocation - robotGoal);

map = robotics.OccupancyGrid(14,13,20);

figureHandle = figure('Name', 'Map');
axesHandle = axes('Parent', figureHandle);
mapHandle = show(map, 'Parent', axesHandle);
title(axesHandle, 'OccupancyGrid: Update 0');

updateCounter = 1;
while(distanceToGoal > goalRadius)
    scanMsg = receive(scanSub);
    pose = getTransform(tftree, 'map', 'robot_base', scanMsg.Header.Stamp, 'Timeout', 2);

    position = [pose.Transform.Translation.X, pose.Transform.Translation.Y];
    orientation = quat2eul([pose.Transform.Rotation.W, pose.Transform.Rotation.X, pose.Transform.Rotatio
n.Y, pose.Transform.Rotation.Z], 'ZYX');
    robotPose = [position, orientation(1)];

    scan = lidarScan(scanMsg);
    ranges = scan.Ranges;
    ranges(isnan(ranges)) = sim.LaserSensor.MaxRange;
    modScan = lidarScan(ranges, scan.Angles);

    insertRay(map, robotPose, modScan, sim.LaserSensor.MaxRange);

    [v, w] = controller(robotPose);
    velMsg.Linear.X = v;
    velMsg.Angular.Z = w;
    send(velPub, velMsg);

    if ~mod(updateCounter, 50)
        mapHandle.CData = occupancyMatrix(map);
        title(axesHandle, ['OccupancyGrid: Update ' num2str(updateCounter)]);
    end

    updateCounter = updateCounter + 1;
    distanceToGoal = norm(robotPose(1:2) - robotGoal);

    waitfor(controlRate);
end

show(map, 'Parent', axesHandle);
title(axesHandle, 'OccupancyGrid: Final Map');
```

The result is showing below.

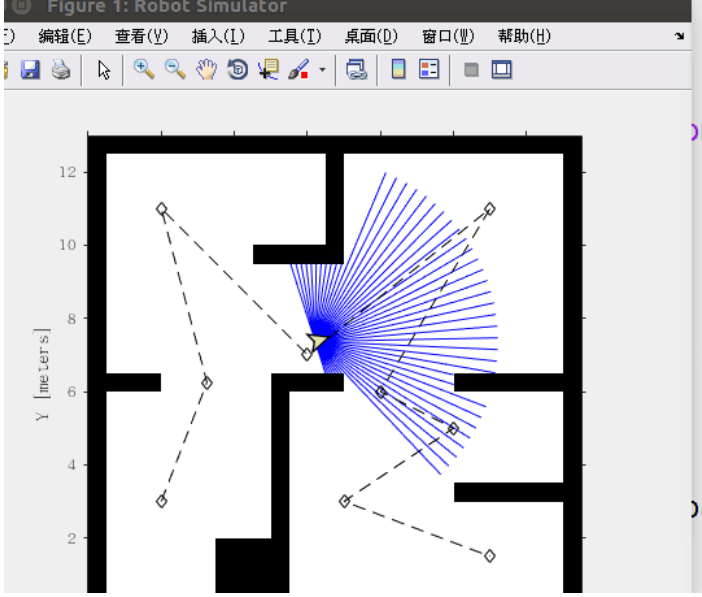
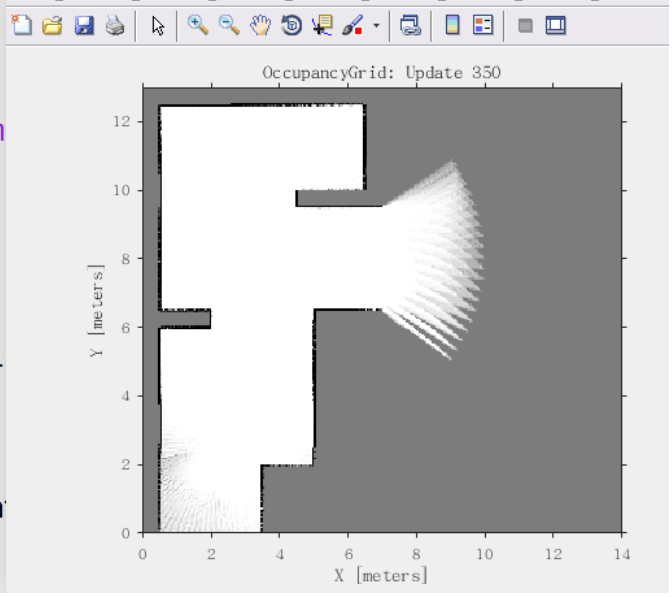


Figure 1: Robot Simulator
編輯(E)
查看(V)
插入(I)
工具(T)
桌面(D)
窗口(W)
幫助(H)
Y [meters]


OccupancyGrid: Update 350
Y [meters]
X [meters]