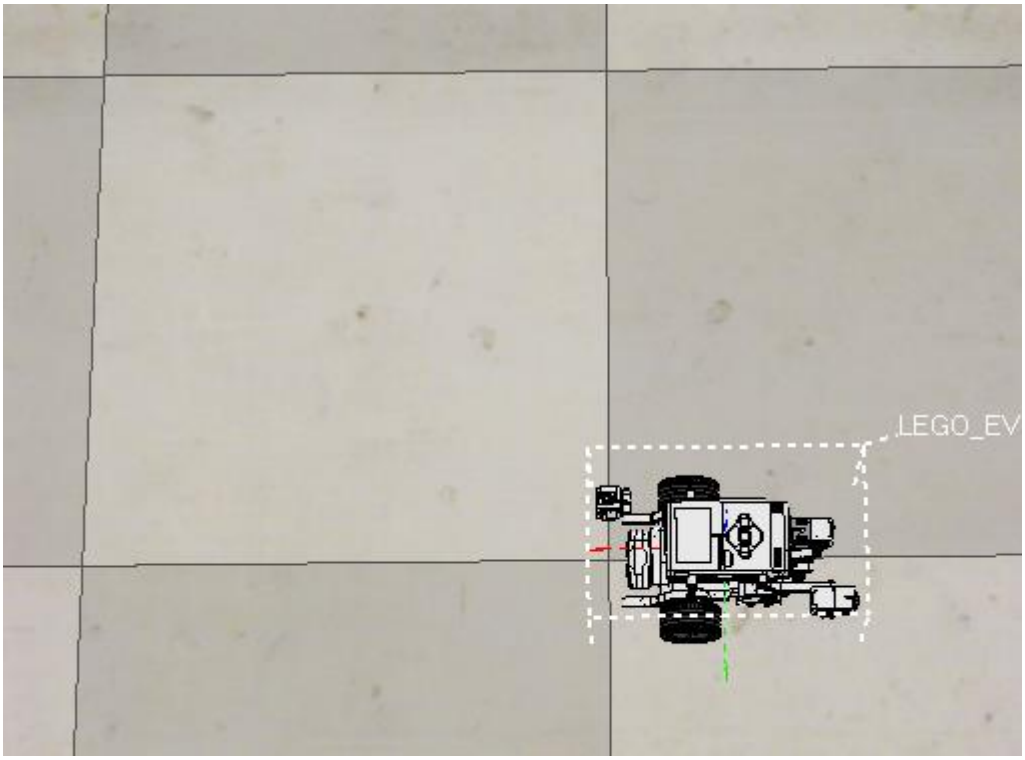


Informe de Práctica 5

Programación de Robots

David Ávila Jiménez || Pedro Antonio Aguilar Lima

Contenido

Resumen

En este informe se recoge toda la información relacionada con la Práctica 5 de programación de robots móviles. El objetivo de la práctica es aprender a realizar la localización del robot Lego usando odometría.

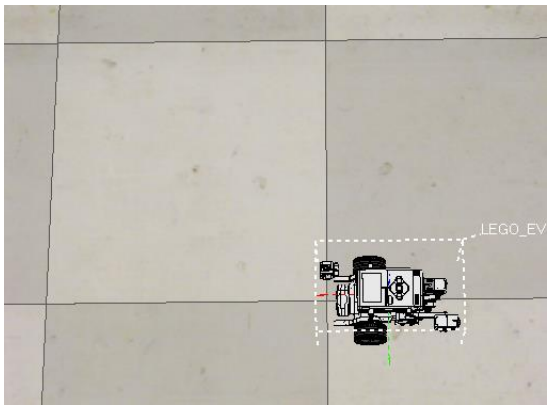
La principal tarea desarrollada en esta práctica es el control de la localización del robot en diferentes escenarios. Para esto, se realiza un reconocimiento de la superficie mediante la lectura de distintos sensores recolectando toda la información necesaria para monitorizar de forma off-line la localización. Además, se mostrarán los datos recabados de forma inmediata (online).

Introducción

En este apartado comenzaremos hablando un poco del escenario usado en las prácticas y de los requisitos que se deben tener en cuenta para su realización. También se comentará el esquema general que se debería seguir.

Escenario

Para esta práctica se volverá a utilizar el escenario usado en las prácticas 2 y 3, que se nos proporcionaba en el campus virtual de la asignatura. En este caso, para esta práctica se trabajará en la zona abierta, donde el robot se moverá para ir recolectando los datos requeridos en la práctica.



Restricciones

En la práctica se nos han impuesto una serie de restricciones o requisitos que se deben cumplir y que son los que se relatan a continuación:

- Para recorrer los tramos rectos, los motores A y C deben ser arrancados a una potencia de 50 durante un cierto tiempo.
- Los giros se realizarán apagando el motor C durante un periodo de tiempo mientras el motor A se acciona a máxima potencia.

Código de la práctica

Primer apartado

En el primer apartado siguiendo el esquema general que se nos ha dado hemos realizado el control de movimiento y la recogida de datos del robot. Para el control del movimiento en el bucle principal se planifica como lo hace, para ello se ha dado un determinado tiempo para que el robot haga cada giro del cuadrado una vez hecho el recorrido recto. Para averiguar esto ha tenido que ser a prueba y error.

```
manejador=[];
cantidad=[];
CreateFile('muestras_p5.txt', 10000, manejador);

while( (CurrentTick()-t_ini) <= tiempo)
     OnFwd(OUT_AC,50);


     WriteLnString(manejador, sprintf('%u\t%u\t%u', (CurrentTick()-t_ini) ...
         , MotorRotationCount(OUT_A), MotorRotationCount(OUT_C)),cantidad);
     while( ((CurrentTick()-t_ini) >= 2000 && (CurrentTick()-t_ini) <= 2650))
         OnFwd(OUT_C,0);
         WriteLnString(manejador, sprintf('%u\t%u\t%u', (CurrentTick()-t_ini) ...
         , MotorRotationCount(OUT_A), MotorRotationCount(OUT_C)),cantidad);
     end
     while( ((CurrentTick()-t_ini) >= 4300 && (CurrentTick()-t_ini) <= 5000))
         OnFwd(OUT_C,0);
        WriteLnString(manejador, sprintf('%u\t%u\t%u', (CurrentTick()-t_ini) ...
         , MotorRotationCount(OUT_A), MotorRotationCount(OUT_C)),cantidad);
     end
     while( ((CurrentTick()-t_ini) >= 6600 && (CurrentTick()-t_ini) <= 7300))
         OnFwd(OUT_C,0);
         WriteLnString(manejador, sprintf('%u\t%u\t%u', (CurrentTick()-t_ini) ...
         , MotorRotationCount(OUT_A), MotorRotationCount(OUT_C)),cantidad);
     end
     while( ((CurrentTick()-t_ini) >= 9000 && (CurrentTick()-t_ini) <= 9700))
         OnFwd(OUT_C,0);
         WriteLnString(manejador, sprintf('%u\t%u\t%u', (CurrentTick()-t_ini) ...
         , MotorRotationCount(OUT_A), MotorRotationCount(OUT_C)),cantidad);
     end
end
Off(OUT_AC);
CloseFile(manejador);
```

Así dentro del bucle, tras crear el fichero que guardará nuestros datos, le establecemos un tiempo que estará en marcha el bucle, que será en torno a 10-12 segundos, y encada iteración del bucle principal se dedicará guardar el tiempo y lo marcado por los sensores del motor izquierdo y derecho. Además, se establecerán una serie de bucles dentro para cada giro, donde también se guardarán datos del robot y de tiempo que así puedan ser analizados más tarde.

Localización offline

En el segundo apartado, tras haber realizado el primero correctamente y guardado los datos, nos pide que realicemos la estimación offline, es decir que utilicemos los datos del apartado 1 y calculemos sus posiciones con las siguientes fórmulas:

$$\Delta x = \frac{\Delta\theta_l \cdot R + \Delta\theta_r \cdot R}{2} \cdot cos(\theta_0)$$
$$\Delta y = \frac{\Delta\theta_l \cdot R + \Delta\theta_r \cdot R}{2} \cdot sin(\theta_0)$$
$$\Delta\theta = \frac{\Delta\theta_r \cdot R - \Delta\theta_l \cdot R}{D}$$

Para ellos se debe utilizar la función que debe tener la siguiente cabecera:

```
function [x1,y1,theta1]=odometry(x0,y0,theta0,t0,rotl0,rotr0,t1,rotl1,rotr1)
```

Tras saber esto, el script de la función quedaría de la siguiente forma:

```
function [x1,y1,theta1] = odometry(x0,y0,theta0,t0, rotl0,rotr0,t1,rotl1,rotr1)
    D = 0.12;
    R = 0.028;
    vT = t1 - t0;

    if vT == 0 %Se devuelve lo mismo en caso de que no varie el tiempo
         x1 = x0;
         y1 = y0;
         theta1 = theta0;
    else %En caso de variar el tiempo se calcula
        %Convertimos angulos de grados a radianes
        radRotr = deg2rad(rotr1) - deg2rad(rotr0);
        radRotl = deg2rad(rotl1) - deg2rad(rotl0);

        % valor Theta con variacion
        theta1 = double(theta0) + (((radRotr*R) - (radRotl*R))./D);
        % valor X con variacion
        x1 = x0 + (((radRotr.*R + radRotl.*R)./2).*cos(theta0));
        % valor Y con variacion
        y1 = y0 + (((radRotr.*R + radRotl.*R)./2).*sin(theta0));
    end


end
```

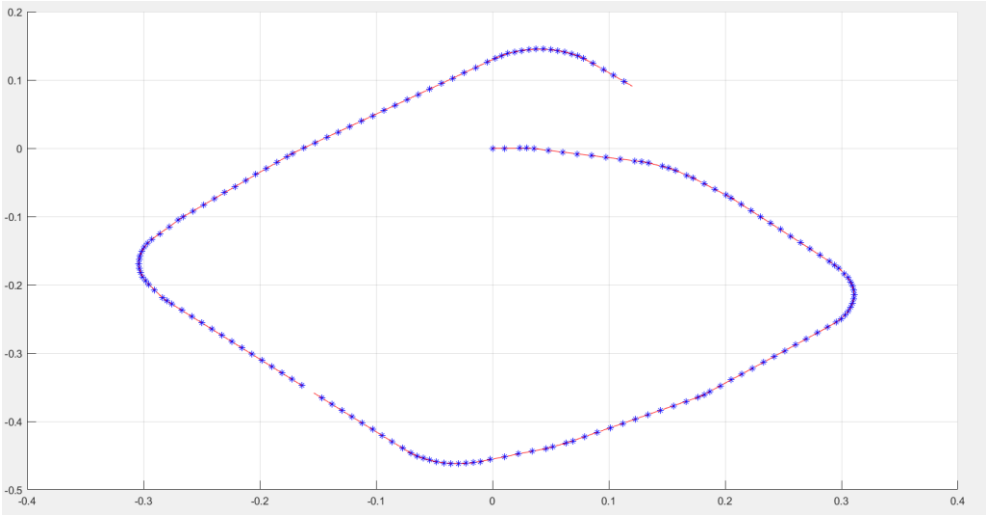
En caso de que la variación de tiempo sea 0, se devolverá la misma posición pues quiere indicar que el robot no se ha movido de su lugar. Si la variación de tiempo es diferente a 0, entonces se realizará el calculo de las posiciones y orientación mediante las fórmulas dadas, siendo devueltas. Para el cálculo se a convertido el giro de las ruedas a radianes con la función deg2rad y que así esté en el sistema internacional, el tiempo no es necesario, pues no afecta a la fórmula. Las constantes D y R han sido establecidas como 0.12 y 0.028 respectivamente.

Tras esto en este apartado se nos pide que mostremos en una gráfica de color azul y rojo las posiciones (x,y) del robot y las orientaciones del mismo.
Para ello hemos realizado el siguiente script que llama a la función odometry.

```
A = importdata('muestras_p5.txt');
[filas,columnas] = size(A);
grid;
hold on;

x0 = 0;
y0 = 0;
theta0 = 0;
t0 = 0;
rotl0 = 0;
rotr0 = 0;

for i = 1:filas
    [x1, y1, theta1] = odometry(x0, y0,theta0, t0, rotl0, rotr0, A(i,1), ...
        A(i,2), A(i,3));

    plot (x1, y1, '*b')
    % Valor de orientacion en x
        ox = x1 + 0.01*cos(theta1);
         % Valor de orientacion en y
        oy = y1 + 0.01*sin(theta1);
        % Al poner r, la gráfica se dibuja en rojo
        plot([x1 ox], [y1 oy], '-r')
        % Igualamos valores de las variables
        x0 = x1;
        y0 = y1;
        theta0 = theta1;
        t0 = A(i,1);
        rotr0 = A(i,3);
        rotl0 = A(i,2);
end

hold off;
```

En él, se comienza importandos los datos del primer apartado y se establece a 0 las diferentes variables usadas en el apartado dos. Así para cada fila leída de datos se llama a la función odometry que nos devuelve los valores x1, y1, theta1, calculados. Después de esto se muestra en la gráfica de color azul los puntos de las posiciones y para las orientaciones se realiza un segmento que comprende entre dos puntos, entre x1 y resultado de calcular donde se encontraría un punto a 1 mm del que hemos calculado al principio.

El resultado sería el siguiente:



Se ha de comentar que en la fórmula se establece r como 0.01 en vez de 0.001, puesto que en el resultado la línea roja no se aprecia y por tanto es necesario ampliar mucho hasta apreciar la diferencia.