Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Implementación de robótica inteligente (Gpo 501)

Evaluación 11.1 (Trayectorias en lazo abierto)

Alumno
José Diego Tomé Guardado A01733345

Fecha de entrega
Viernes 17 de Mayo de 2024

INSTRUCCIONES: Implementar el código requerido para generar las siguientes trayectorias a partir del tiempo y de las velocidades angulares y lineales en un plano 2D, según corresponda. La altura de cada letra debe ser de **3m.**, el ancho puede ser ajustable **a criterio propio** y la separación entre cada letra debe ser de **0.5m.**

RUTA A SEGUIR:
José Diego Tomé G: **JOSE_DIEGO**

IMPLEMENTACIÓN EN MATLAB:

```
%% Simulation parameters
sampleTime = 0.05;                % Sample time [s]
tVec = 0:sampleTime:185;          % Time array
```

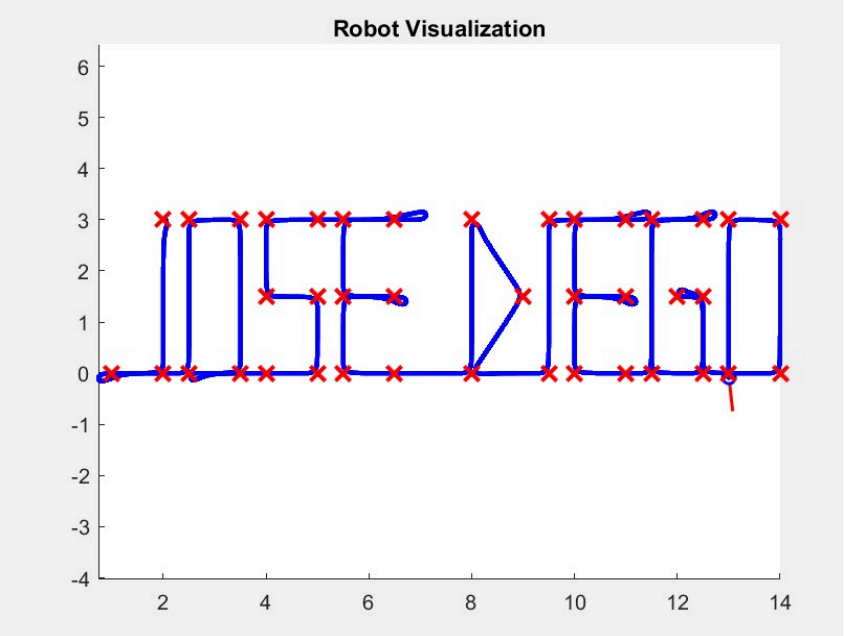
- **Definición de la trayectoria para las letras:**

```
waypoints = [2,3; 2,0; 1,0; 2.5,0;
             2.5,3; 3.5,3; 3.5,0; 2.5,0;
             4,0; 5,0; 5,1.5; 4,1.5; 4,3; 5,3;
             5.5,3; 6.5,3; 5.5,3; 5.5,1.5; 6.5,1.5; 5.5,1.5; 5.5,0; 6.5,0
             8,0; 8,3; 9,1.5; 8,0;
             9.5,0; 9.5,3; 10,3; 11,3; 10,3; 10,1.5; 11,1.5; 10,1.5; 10,0;
11,0;
             11.5,0; 11.5,3; 12.5,3; 11.5,3; 11.5,0; 12.5,0; 12.5,1.5;
12,1.5; 12.5,1.5;
             12.5,0; 13,0; 14,0; 14,3; 13,3; 13,0]
```

- **Variables usadas:**

```
controller.LookaheadDistance = 0.25; %muy corto no vera el waypoint
controller.DesiredLinearVelocity = 0.4;
controller.MaxAngularVelocity = 10;
```

EJECUCIÓN:



ELECCIÓN DE ESTRATEGIA: LANDMARKS

Decidi elegir esta estrategia y código que utilizamos en la actividad de landmarks debido a que se me hizo de una forma más rápida poder definir la trayectoria con los waypoints, además de que se controla más fácilmente los valores que necesitamos cambiar como las velocidades y el lookaheadDistance es mucho más sencillo poder controlarlos y poder ver como cambian uno a uno dependiendo de los valores implementados. Al tener este método puede ser un poco impreciso en el dibujo de las letras debido a que en las ocasiones donde hay que volver al mismo waypoint puede ocasiones que se generen giros abiertos que afectan el dibujo de las letras, pero al realizar esta implementación de landmarks es mucho más sencillo y rápido que otras estrategias y se genera un ajuste considerablemente muy correcto.

PREGUNTAS:

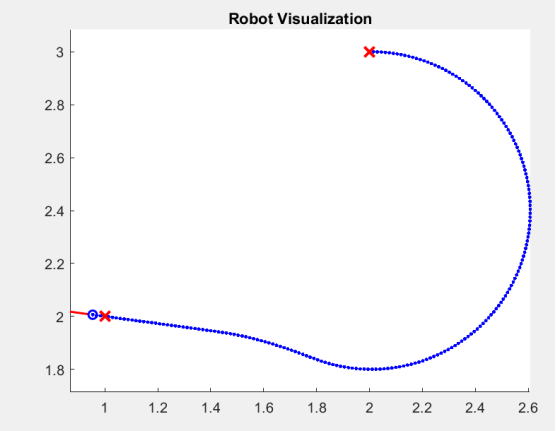
a) ¿Cuál fué el o los parámetros que se modifican para obtener una trayectoria recta? ¿Por qué?

Primeramente es importante poder tener la definición correcta de los waypoints para poder lograr correctamente la trayectoria, deben estar alineados a una línea recta donde posicionando los puntos se asegura que el vehículo siga una trayectaria en línea recta entre los puntos, de otra forma el parámetro de LookaheadDistance debe ser lo suficiente pequeño para poder tener una mejor precisión en la dirección sin causar las oscilaciones o curvas como un valor pequeño podria proponerse como el qué implementamos en nuestro código de 0.25.

Despues continuando con las velocidades primeramente en la lineal necesitamos tener un movimiento suave debido a qué no tenemos qué generar cambios abruptos qué puedan desviar la trayectoria por lo que al ser un valor pequeño también puede realizar un mejor ajuste para seguir las líneas rectas. De otra forma para la MaxAngularVelocity necesitamos que sea lo suficientemente alta para poder corregir esas pequeñas desviaciones de tal forma que el giro sea cerrado para poder ajustar mejor a las líneas rectas pero no debe ser tan alta tampoco para no generar giros indeterminados.

b) ¿Cuál fué el o los parámetros que se modifican para obtener una trayectoria curva? ¿Por qué?

Para las trayectorias curvas podemos tener una definiciòn de los waypoints diferentes debido a que deberian estar dispuestos para generar una curva, continuando con el LookaheadDistance necesitamos de tener un valor que este moderado para que el vehículo pueda observar más adelante la trayectoria como un valor de 0.3 donde permite ver los siguientes waypoints sin tener una desviación. Con la velocidad Linear podemos tener una velocidad aún más alta si se requiere que el seguimiento

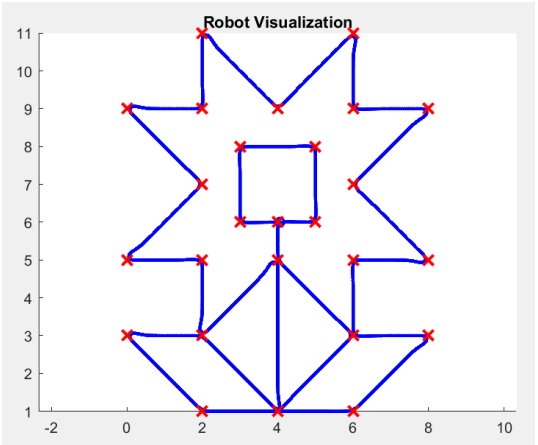


de la curva sea de mejor manera para poder observar como llega más rápido a la curva con un valor también podria ser de 0.3 para probar ahorita, lo qué es necesariamente importante si es que queremos generar una trayectoria curva es la MaxAngularVelocity donde necesitamos que sea un valor mucho más bajo a lo que tenemos en las trayectorias lineales como un valor de 0.5 para que pueda tener un giro más abierto donde probando se puede generar así la trayectoria punto a punto pero de manera curva:

```
controller.LookaheadDistance = 0.3; %muy corto no vera el waypoint
controller.DesiredLinearVelocity = 0.3;
controller.MaxAngularVelocity = 0.5;
```

c) ¿Cuál fué el o los parámetros que se modifican para obtener un giro? ¿Por qué?

También cambiamos los mismos parámetros como se explicó arriba solo que ahora depende si necesitamos un giro más cerrado o un giro más abierto, por lo qué en el caso de si se requiere un giro más cerrado para el trazado de trayectorias más lineales y que se tenga un mejor ajuste a la figura podemos decir que necesitamos de un LookaheadDistance pequeño para que no pueda ver los otros waypoints y se pierda y con el valor pequeño lograremos qué haga el giro de mejor forma, con la combinación de un valor pequeño en la velocidad lineal para poder generar un giro muchisimo más cerrado y una angular mucho mayor que la lineal para hacer un giro más cerrado. Tenemos como ejemplo estos parámetros que fueron puestos a prueba con la figura de una flor donde podemos observar qué se logró un buen desempeño en cuanto a cada uno de los giros tanto de petalos, las hojas y el centro de la flor.




```
controller.LookaheadDistance = 0.25;
%muy corto no vera el waypoint
controller.DesiredLinearVelocity =
0.6;
controller.MaxAngularVelocity = 10;
```

d) ¿Qué papel desempeña el vector del tiempo en la generación de la trayectoria?

El vector de tiempo tVec es fundamental para la simulación debido a que este mismo es el qué define los instantes de tiempo en los cuales se va a ir actualizando la posición del vehículo a lo largo de toda la trayectoria de la simulación, tenemos el sampleTime que

determina la frecuencia de actualización de la posición para asegurar que las actualizaciones sean frecuentes y precisas ponemos un valor bajo de 0.05 segundos.

El vector que tenemos que va de 0 a 185 segundos es lo que define la duración total de la simulación lo que afecta al tiempo que el vehículo tiene para seguir todos los waypoints, donde podemos decir que al tener un recorrido más largon con más waypoints tenemos que aumentar este tiempo, no es lo mismo dibujar un cuadrado que una flor o algo parecido.

e) ¿Cuáles fueron los parámetros que se ajustaron para obtener las dimensiones de las trayectorias deseadas?

Como ya lo dijimos anteriormente principalmente en este método se necesitan de estas tres principales parámetros para poder ajustar todo el recorrido y generar la trayectoria deseada:

```
controller.LookaheadDistance = 0.25; %muy corto no vera el waypoint
controller.DesiredLinearVelocity = 0.4;
controller.MaxAngularVelocity = 10;
```

Generando así que tenga un valor pequeño para LookaheadDistance para poder hacer que el robot vea solo el waypoint qué debe de seguir ya que al tener los waypoints más cercanos unos a los otros podria que en ocasiones teniendo un valor más alto alcance a ver otros y los giros sean demasiado imprecisos. Con la combinación de este valor pequeño y el valor de una velocidad lineal pequeña podemos hacer que los giros sean más precisos dando tiempo para que el robot pueda recorrer tranquilamente el giro y vaya hacia el siguiente waypoint. Finalmente tenemos el valor para la velocidad angular qué pusimos uno mucho más alto de 10 debido a que necesitamos que los giros sean cerrados para que tenga un mejor ajuste sobre el recorrido para cada una de las letras.

Finalmente algo que también puede mejor la precisión de nuestro código es el declarar la posición inicial del robot, donde necesitamos declarar desde qué punto comenzará sobre todo el theta debido a que si el waypoint como en este caso estaba abajo del primero podemos hacer que la orientación del robot comience abajo y no pierda de vista ni haga un giro brusco al comienzo.

```
initPose = [2;3;2*pi];              % Initial pose (x y theta)
pose = zeros(3,numel(tVec));     % Pose matrix
pose(:,1) = initPose;
```