Taller 8: Control para el seguimiento de trayectorias

Introducción a la Robótica Movil

May 2, 2018

Se recomienda repasar las clases teóricas para la resolución de este trabajo práctico.

1 Introdución

El objetivo de este Taller es poner en práctica conceptos de cinemática y control vistos en la segunda parte de la materia. Se desarrollará un nodos de **ROS** capaz de hacer el seguimiento de trayectorias utilizando técnicas de control a **lazo cerrado**.
De la página de la materia pueden descargarse los **paquetes Catkin** necesarios para completar el Taller.

1.1 Esqueleto y configuración del paquete provisto

El paquete provisto por la materia se llama **lazo_cerrado** y posee un único nodo denominado **KinematicPositionController**. Este nodo esta **incompleto** y deberán completarlo para la resolución de los ejercicios del Taller. Deberán trabajar sobre el archivo:
lazo_cerrado/src/KinematicPositionController.cpp

Este nodo tiene que permitir 3 modos de operación distintos dependiendo del tipo de selección de la pose objetivo (**goal**). Modos de operación:

1. **FIXED_GOAL:** Se envian comandos de velocidad para alcanzar una pose final objetivo **pre definida**.
2. **TIME_BASED:** En base a una trayectoria con requerimientos temporales definidos. Se define en cada momento la pose objetivo como aquella "donde debería encontrarse actualmente el robot".
3. **PURSUIT_BASED:** En base a una trayectoria con requerimientos de pose (**x , y,** θ). Se designa un punto vía "cercano" que permita realizar el seguimiento de la trayectoria en base a **un algoritmo de selección que deberán completar ustedes**.

Es posible configurar el modo de operación a ejecutar modificando los siguientes parámetros en el archivo **/lazo_cerrado/src/launch/lazo_cerrado.launch**:

```
<param name="goal_selection" type="str" value="FIXED_GOAL"/>

<param name="fixed_goal_x" type="double" value="3"/>
```

```
<param name="fixed_goal_y" type="double" value="0"/>
<param name="fixed_goal_a" type="double" value="-1.57"/>
```

Los valores permitidos para el parámetro **"goal_selection"** son: **FIXED_GOAL**, **TIME_BASED** y **PURSUIT_BASED**.

Los parámetros **fixed_goal_x**, **fixed_goal_y**, **fixed_goal_a** solo son válidos al configurar el modo de operación **FIXED_GOAL** y representan una pose objetivo requerida **en referencia a la pose inicial que comienza el robot**.

NOTA: $-1.57 \simeq -\frac{\pi}{2}$. Deben configurar el ángulo requerido en valores absolutos expresados en radianes.

NOTA2: El nodo publica un mensaje de tipo **geometry_msgs/Pose** en el tópico **'/goal_pose'** para poder **visualizar la pose objetivo en RViz**.

1.2 Nodo de registro

Se provee además de un nodo de registro el cual escribe en texto plano las posiciones y orientaciones del robot en cada momento de tiempo.

Se registra la información de odometría, la pose real **ground-truth** publicada por el **V-Rep** y la pose objetivo seleccionada. Los archivos son generados en el **home**, en la carpeta **oculta** (revelar con Ctrl+H) **/.ros/**.

El nombre de los archivos corresponde a:

1. ***timestamp*_poses.log** las poses publicadas por la odometría.
2. ***timestamp*_ground-truth.log** las poses reales del robot en la simulación.
3. ***timestamp*_goals.log** las poses objetivo seleccionadas en cada momento.

El formato de los archivos **.log** es:

```
t x y theta
```

NOTA: t corresponde al tiempo en que fue publicada la pose en segundos.

Ejercicio 1: Converger a una pose objetivo

Configurar al nodo de control en modo **FIXED_GOAL** y definir una pose objetivo la cual se desee alcanzar (por defecto el archivo .launch viene configurado con una interesante). Implementar **el algoritmo de control a lazo cerrado visto en clase** completando el método:

```
bool control(const ros::Time& t, double& v, double& w)
```

Para inicializar la simulación en el **V-Rep** deben utilizar la misma escena provista en el taller de lazo abierto:
/lazo_abierto/launch/lazo_abierto.launch
Y luego lanzar los nodos utilizando el comando:
roslaunch lazo_cerrado lazo_cerrado.launch

Como resultado del comportamiento, el robot pioneer deberá **converger** a la pose objetivo de manera de respetas tanto la **posición x,y** y la **orientación** θ definidas.

NOTA: Recordar hacer la conversión necesaria a las variables de manera de **establecer el marco de referencia del objetivo como el marco inercial** (revisar las cuentas vistas en clase).
NOTA2: Leer detenidamente los comentarios del código para guiarse en la forma que debe ser completado.

a) ¿Que sucede cuando el robot se acerca al objetivo?

b) Probar distintos valores de K_ρ, K_α y K_β ¿Como modifican el comportamiento estos parámetros? ¿Que afectan cada uno de ellos?

c) **(Opcional)** Graficar la trayectoria realizada por el robot.

Ejercicio 2: Seguimiento de una trayectoria con requerimiento temporal

Configurar al nodo de control en modo **TIME_BASED** y Definir una trayectoria con un requerimiento temporal **"exigente"** en el archivo .launch (por defecto ya viene una configurada).

Utilizar los parámetros K_ρ, K_α y K_β **más estables** que se encontraron en el ejercicio anterior.

a) ¿El método de control es capaz de seguir la trayectoria a la perfección? Si no lo logra, ¿por que creen que sucede esto?

b) Buscar los mejores parámetros K_ρ, K_α y K_β que permitan seguir lo mejor posible la trayectoria.

c) **(Opcional)** Graficar y comparar la trayectoria del robot con respecto a la trayectoria definida por las poses objetivos seleccionadas.

Ejercicio 3: Algoritmo de seguimiento de persecución

Dado que cumplir requerimientos temporales resulta muy restrictivo, vamos a plantear un algoritmo de seguimiento que permita seguir una trayectoria de puntos vía (waypoints) definidos por poses (**x, y,** θ). Para esto seleccionaremos puntos vía cercanos pero que mantengan una determinada distancia del de la posición actual del robot. Ver figura 1

Lo que se desea lograr es **completar** la trayectoria de "manera segura". Deberán implementar un algoritmo de selección de pose objetivo completando el método:

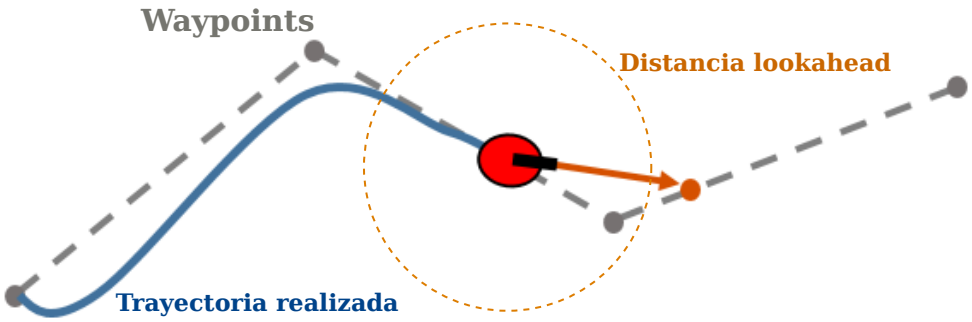

Figure 1: Circuito de simulación del robot

```
bool getPursuitBasedGoal(const Time& t, double& x,
                         double& y, double& a)
```

Este método debe revisar la trayectoria requerida y determinar una **"pose objetivo adecuada"**. Para esto pueden suponer que la trayectoria posee una gran cantidad de waypoints definidos y curvas **"suficientemente suaves"** con angulos de giro abiertos.
Para probar el algoritmo de seguimiento deberán configurar el nodo de control utilizando el modo de operación **PURSUIT_BASED**.

NOTA: Se recomienda encontrar el waypoint de la trayectoria más cercano al robot (**en términos de x,y**) y luego buscar el **primer** waypoint que se encuentre a una **distancia predefinida de lookahead en x,y**.

NOTA2: Leer con atención los comentarios del código para lograr el cometido.

a) Explicar detalladamente el método propuesto. ¿Que ventajas tiene en comparación a la selección basada en la restricción temporal?

b) Utilizar los mismos parámetros K_ρ, K_α y K_β hallados en el ejercicio anterior. Y realizar el seguimiento de la misma trayectoria seleccionada. Comparar los resultados obtenidos.

c) **(Opcional)** Graficar y comparar la trayectoria del robot y la trayectoria definida por las poses objetivos seleccionadas.