Universidad de Ingeniería y Tecnología

FUNDAMENTOS DE ROBÓTICA

Prof. Oscar E. Ramos Ponce

Guía de Laboratorio 4:

Control Cinemático mediante Cinemática Diferencial

Lima - Perú

2017 - 1

Laboratorio 4: Control Cinemático mediante Cinemática Diferencial

1. Objetivos

- Implementar la cinemática diferencial para un robot manipulador
- Controlar el movimiento de un manipulador usando cinemática diferencial
- Controlar la orientación del efector final de un robot manipulador

2. Marco Teórico

La cinemática diferencial relaciona la velocidad articular ($\dot{\boldsymbol{q}}$) con la velocidad del efector final, la cual puede estar expresada como un *twist* (velocidad lineal $\boldsymbol{v}$ y velocidad angular $\boldsymbol{\omega}$) o como la derivada temporal de la posición y orientación. Una vez que se dispone de una velocidad articular, ésta puede ser integrada para obtener una nueva configuración articular $\boldsymbol{q}$.

2.1. Control Cinemático de Posición

Considérese que se desea controlar la posición $\boldsymbol{x} = [x \quad y \quad z]^T$ del efector final para que alcance la posición deseada $\boldsymbol{x}_d = [x_d \quad y_d \quad z_d]^T$. Se define un error $\boldsymbol{e}$ tal que

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_d. \tag{1}$$

Considerando una referencia constante, la derivada temporal de la posición deseada se anula y la derivada temporal del error es igual a la derivada temporal de la posición actual. Así, se tiene la siguiente relación de primer orden a nivel de velocidades:

$$\dot{\boldsymbol{e}} = J\dot{\boldsymbol{q}} \tag{2}$$

donde $J = \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}$ representa el Jacobiano analítico que relaciona la derivada de la posición del efector final con la velocidad articular. A nivel cinemático se invierte la relación anterior como

$$\dot{\boldsymbol{q}} = J^{\#}\dot{\boldsymbol{e}} \tag{3}$$

donde $J^{\#}$ representa la pseudo-inversa del Jacobiano. Se suele usar la pseudo-inversa de Moore Penrose, pero cuando el sistema está cerca de singularidades se puede usar la pseudoinversa amortiguada.

Para controlar la posición a nivel cinemático de primer orden se requiere escoger $\dot{\boldsymbol{e}}$ de tal modo que el error converja a cero. Esto se consigue utilizando la siguiente *ley de control* cinemática:

$$\dot{\boldsymbol{e}}^* = -k\boldsymbol{e} \tag{4}$$

donde k es un escalar que se conoce como ganancia cinemática. Así, el control cinemático obtiene la velocidad articular usando (3), con una referencia deseada dada por (4) y definiendo el error como en (1). Luego, esta velocidad es integrada usando, típicamente, una integración de Euler:

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} + \Delta t\ \dot{\boldsymbol{q}}_k \tag{5}$$

donde Δt es el período de control que tiene el robot. Esta integración es usada frecuentemente debido a que el período de control es típicamente pequeño para robots reales (1 ms a 10 ms normalmente) y por tanto el error de integración es despreciable.

Pregunta. Encontrar la solución a la ecuación diferencial dada en (4) asumiendo que $\boldsymbol{e}$ es unidimensional. A partir de la solución obtenida, ¿qué efecto tiene esta ley de control a medida que el tiempo se incrementa?

2.2. Control Cinemático de Orientación

El control de orientación se da de una manera semejante al control de posición, pero la gran diferencia consiste en la definición del error. En el caso de la orientación no se puede definir una simple resta entre matrices de rotación. Es posible definir resta en ángulos de Euler o roll, pitch, yaw pero ello lleva a problemas cerca de las singularidades de la configuración. Algo más usual es definir el error a partir del eje/ángulo de la matriz $R^T R_d$, donde R es la matriz de rotación actual, y R_d es la deseada.

Alternativamente se puede definir el error de orientación en términos de cuaterniones. Considérese que $Q_d = (w_d, \boldsymbol{\epsilon}_d)$ es el cuaternión deseado para el efector final, y $Q = (w, \boldsymbol{\epsilon})$ es el cuaternión actual del efector final (donde w es la parte escalar y $\boldsymbol{\epsilon}$ la parte vectorial). El error entre ambos cuaterniones queda descrito por $Q_e = Q_d Q^{-1}$. Cuando ambos cuaterniones son iguales, y por tanto el error es nulo, se tiene la identidad $Q_e = [1\ 0\ 0\ 0]^T$. Luego, el error se define con relación a esta identidad deseable. El cuaternión de error Q_e, usando la definición de producto de cuaterniones, se escribe como

$$\begin{aligned} Q_e &= (w_d, \boldsymbol{\epsilon}_d)(w, -\boldsymbol{\epsilon}) \\ &= (w_d w + \boldsymbol{\epsilon}_d^T \boldsymbol{\epsilon}, -w_d \boldsymbol{\epsilon} + w\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon}_d \times \boldsymbol{\epsilon}) \\ &= (w_e, \boldsymbol{\epsilon}_e). \end{aligned}$$

Se define el error de orientación e_o con respecto a la identidad del cuaternión como

$$\boldsymbol{e}_o = \begin{bmatrix} w_e - 1 \\ \boldsymbol{\epsilon}_e \end{bmatrix}. \tag{6}$$

Para controlar tanto la posición como la orientación, se define el error de posición y orientación como

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}_d \\ \boldsymbol{e}_o \end{bmatrix} \tag{7}$$

Luego, se utiliza el mismo procedimiento que para el control de posición; es decir, se calcula la velocidad articular con (3) usando como referencia (4) y el error dado por (7).

2.3. Jacobiano Analítico

El Jacobiano analítico, también denominado Jacobiano de la tarea (*task Jacobian*) se utiliza en todos los casos mencionados en las dos secciones anteriores. Una vez que se tiene la cinemática directa de un robot cualquiera, este Jacobiano puede ser calculado mediante métodos numéricos.

Para el caso considerado aquí, la posición está dada en coordenadas Cartesianas $\boldsymbol{x} = [x \;\; y \;\; z]^T$ y la orientación está dada por un cuaternión $Q = [w \;\; \epsilon_x \;\; \epsilon_y \;\; \epsilon_z]^T$. El Jacobiano analítico con respecto a las derivadas temporales de estos elementos, para un robot con n grados de libertad está dado por

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \\ \frac{\partial w}{\partial q_1} & \frac{\partial w}{\partial q_2} & \cdots & \frac{\partial w}{\partial q_n} \\ \frac{\partial \epsilon_x}{\partial q_1} & \frac{\partial \epsilon_x}{\partial q_2} & \cdots & \frac{\partial \epsilon_x}{\partial q_n} \\ \frac{\partial \epsilon_y}{\partial q_1} & \frac{\partial \epsilon_y}{\partial q_2} & \cdots & \frac{\partial \epsilon_y}{\partial q_n} \\ \frac{\partial \epsilon_z}{\partial q_1} & \frac{\partial \epsilon_z}{\partial q_2} & \cdots & \frac{\partial \epsilon_z}{\partial q_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{x}}{\partial q_1} & \frac{\partial \boldsymbol{x}}{\partial q_2} & \cdots & \frac{\partial \boldsymbol{x}}{\partial q_n} \\ \frac{\partial Q}{\partial q_1} & \frac{\partial Q}{\partial q_2} & \cdots & \frac{\partial Q}{\partial q_n} \end{bmatrix} \tag{8}$$

donde cada elemento puede calcularse mediante diferencias finitas. Por ejemplo, la primera columna puede calcularse como:

$$\frac{\partial \boldsymbol{x}}{\partial q_1} \approx \frac{\boldsymbol{x}(\boldsymbol{q} + \delta q_1) - \boldsymbol{x}(\boldsymbol{q})}{\delta q_1} \qquad \text{y} \qquad \frac{\partial Q}{\partial q_1} \approx \frac{Q(\boldsymbol{q} + \delta q_1) - Q(\boldsymbol{q})}{\delta q_1} \tag{9}$$

donde $\boldsymbol{x}(\boldsymbol{q})$ es la posición en función de las variables articulares, $Q(\boldsymbol{q})$ es el cuaternión en función de las variables articulares, y δq_1 es un pequeño incremento en la articulación 1. Notar que $\boldsymbol{x}(\boldsymbol{q})$ puede obtenerse a partir de la cuarta columna de la matriz de transformación homogénea que relaciona al efector final con respecto a la base. De igual modo, el cuaternión se puede obtener convirtiendo la parte rotacional de la matriz de transformación homogénea a cuaternión.

3. Implementación de Cinemática Diferencial

Descargar el archivo `lab4.zip`, descomprimirlo y copiar la carpeta `lab4` dentro de la carpeta `src` que se encuentra en el espacio de trabajo `lab_ws`. La carpeta `lab4` es ya un paquete y posee algunos archivos que serán usados y/o modificados en los pasos posteriores.

Nota: en el caso que no se tenga los paquetes del robot KUKA, estos pueden ser descargados del repositorio de `github` de la misma manera que se hizo en el laboratorio 1. Los paquetes necesarios son `iiwa_control`, `iiwa_description`, `iiwa_gazebo`. Estos paquetes deben estar dentro de la carpeta `lab_ws/src/kuka`.

3.1. Control de Posición del Efector Final

El modelo de robot que se utilizará es el del robot LBR iiwa 7 de KUKA. Los parámetros de Denavit-Hartenberg de este robot están dados en la siguiente tabla, donde $L_1 = 0{,}34$ m, $L_2 = 0{,}4$ m, $L_3 = 0{,}4$ m, $L_4 = 0{,}126$ m.

i	d	θ	a	α
1	L_1	$\pi + q_1$	0	$\pi/2$
2	0	$\pi + q_2$	0	$\pi/2$
3	L_2	q_3	0	$\pi/2$
4	0	$\pi + q_4$	0	$\pi/2$
5	L_3	q_5	0	$\pi/2$
6	0	$\pi + q_5$	0	$\pi/2$
7	L_4	q_6	0	0

Tareas.

1. Calcular la matriz de transformación homogénea que relaciona al efector final con la base del robot (cinemática directa) completando la función llamada `iiwa_fkine` del archivo `functions.py`. Utilizar la función `dh` ya implementada. De igual manera, calcular la matriz de transformación homogénea que relaciona el codo del robot (eslabón 4) con la base del robot.

2. Para probar la función anterior, ejecutar el archivo `display_iiwa.launch` usando roslaunch y luego, en otro terminal, ejecutar el archivo Python `test_iiwa_fkine` con `rosrun`. El marcador azul del codo se puede apreciar si se reduce el factor `alpha` del modelo del robot a, por ejemplo, 0.5 en RViz.

3. En el archivo llamado `iiwa_diffkine_ef` implementar control cinemático de posición usando cinemática diferencial tal como se indica en la sección 2.1 (notar que el Jacobiano de posición ya ha sido implementado como en el laboratorio anterior `iiwa_jacobian_pos`). Usar la pseudoinversa de Moore-Penrose y una ganancia de $k = 0{,}5$. Verificar si se alcanza la posición deseada.

4. La posición actual, deseada y la configuración articular se almacenan en archivos llamados `xactual.dat`, `xdeseado.dat` y `q.dat`, respectivamente, en la carpeta temporal `/tmp`. Usando estos datos, para la parte anterior, graficar la posición deseada y la posición actual en función del tiempo. Luego, graficar la posición deseada y la posición actual (trayectoria) en el espacio Cartesiano (x, y, z).

5. Usar valores de ganancia $k = 0{,}1, 1, 10, 100$ y graficar el resultado en función del tiempo así como las trayectorias Cartesianas obtenidas. Comentar sobre los resultados obtenidos.

6. Realizar una prueba (`if`) para determinar si el Jacobiano es singular en alguna configuración. En este caso la matriz J no es cuadrada, ¿cómo se puede verificar dicha condición? ¿En qué configuración (de las usadas) el robot está en una configuración singular?

7. Cuando J es singular, la pseudoinversa de Moore-Penrose da origen a valores de control altos. Por este motivo, cuando J es singular (a partir de la condición determinada en el punto anterior) calcular la pseudo-inversa amortiguada con un valor constante de $= 0{,}01$.

8. Para las mismas ganancias k usadas anteriormente graficar el resultado en función del tiempo así como las trayectorias Cartesianas obtenidas. ¿Cuál es la diferencia con respecto al resultado sin la verificación de singularidades?

9. Realizar una trayectoria circular cartesiana para que robot pueda seguir. Graficar el resultado temporal y Cartesiano.

3.2. Control de Posición del Codo

En esta sección se busca implementar el control cinemático de posición pero en lugar de usar el efector final, se usará el codo del robot que se encuentra al final de la articulación número 4.

Tareas.

1. Calcular el Jacobiano para el codo en el archivo `functions.py` con una función que se denominará `iiwa_jacobian_pos_elbow`.

2. Crear un archivo similar al de la sección anterior y modificarlo para controlar la posición del codo. Utilizar como configuración articular inicial al vector $\boldsymbol{q} = (0, -0{,}7, 0, -1{,}5, 0, -0{,}8, 0)$ y hacer que el codo siga la trayectoria circular dada por $x = rcos(t)$, $y = rcos(t)$ y $z = 0{,}623$, con $r = 0{,}28$.

3. Usar 3 valores diferentes de k y graficar los resultados.

3.3. Control de Posición y Orientación del Efector Final

Hasta ahora solamente se ha realizado el control de la posición del robot. Ahora se desea implementar el control de la orientación del mismo. El procedimiento será similar al seguido para la implementación del control de posición pero se utilizará cuaterniones y el error definido para los cuaterniones.

Tareas.

1. Implementar el Jacobiano de posición y orientación en el archivo `functions.py`, el cual se denominará `iiwa_jacobian_full`. ¿Cuál es el tamaño de este Jacobiano?

2. Completar el archivo `iiwa_diffkine_ef2` para implementar la el control cinemático de posición y orientación usando cuaterniones. Utilizar la descripción teórica de la sección 2.2. Esta implementación debe usar la pseudoinversa de Moore-Penrose solamente cuando el sistema no se encuentra en singularidad cinemática. De lo contrario se usará la pseudoinversa amortiguada.

3. Para 3 valores diferentes de k, graficar los resultados en función del tiempo y en el espacio Cartesiano.

4. Robot Fetch

Para verificar que el control cinemático de posición y orientación es un enfoque genérico, implementar dicho control en el robot fetch. Para esto adaptar el programa de control del robot fetch desarrollado en el laboratorio 2.

5. Conclusiones

Elaborar algunas conclusiones sobre el laboratorio desarrollado e indicarlas en el informe del laboratorio.