Cinemática de robots

Transformaciones lineales aplicadas a robots articulados

Badenes, Tomás
tomasbadenes@alu.ing.unlp.edu.ar

Fernández, Santiago Adriel
TBD

Majoros, Lorenzo
lorenzomajoros@alu.ing.unlp.edu.ar

Seery, Juan Martín
juan.seery@alu.ing.unlp.edu.ar

Facultad de Ingeniería, Universidad Nacional de La Plata

1. Introducción

La cinemática de robots es la ciencia que estudia el movimiento de cadenas articuladas con múltiples grados de libertad. Se asocia con el estudio de cuerpos rígidos y hace fuerte uso de la geometría. Las técnicas descritas por esta disciplina se aplican en todo tipo de ámbitos, desde animaciones por computadora hasta la biomecánica. Excelentes ejemplos son los brazos utilizados en líneas de ensamblaje o los brazos robot utilizados para operar en el exterior de la estacion espacial internacional.

Aquí se tratará la solo parte más superficial del área: obtener ecuaciones que relacionan un punto en el espacio con, en este caso, rotaciones de motores (los llamados parámetros del robot). Esto es esencial, ya que nos permitiría saber cuánto hay que rotar los motores para que el robot llegue a un punto, a dónde está apuntando en un momento dado, cuál es el alcance del mismo, entre un sinfín de otros ejemplos.

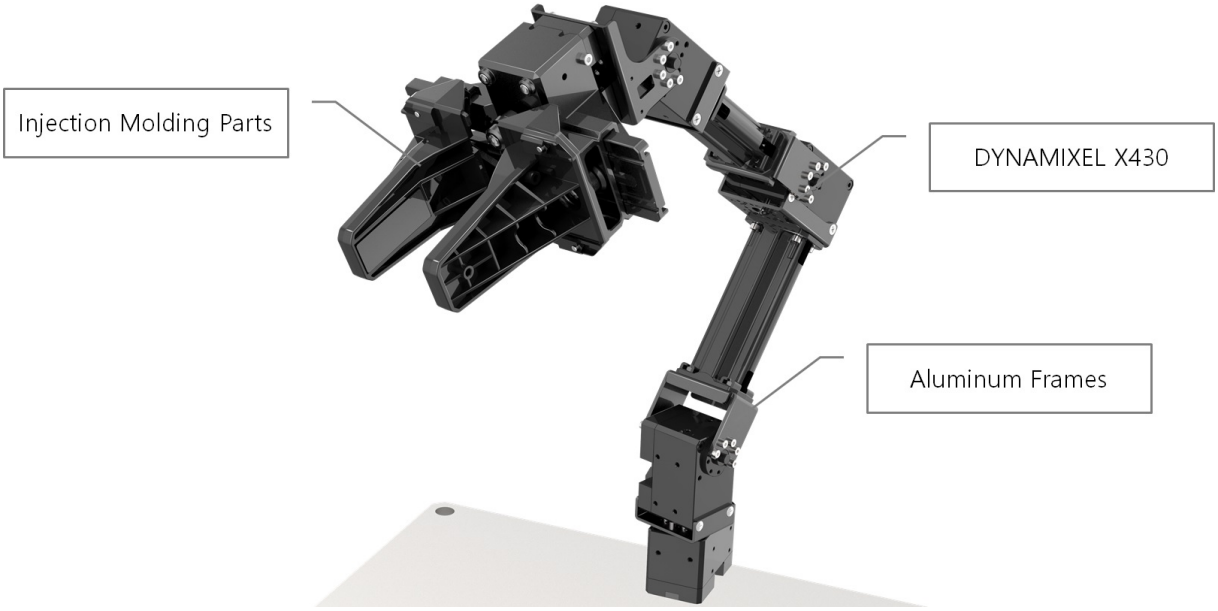

Figura 1: Brazo OpenMANIPULATOR-X

Se verá cómo modelar y controlar a partir de estas operaciones el robot **OpenMANIPULATOR-X**[1], un brazo robótico de 4 grados de libertad creado por la empresa ROBOTIS[2] para el proyecto TurtleBot 3[3] en conjunto a Open Robotics[4]. Se eligió este robot porque es de plataforma, hardware y software abierto, y provee un entorno de simulación. Además, está pensado como proyecto educativo y viene acompañado de extensa documentación, lo cual resulta ideal para el proyecto.

2. Forward kinematics – ¿a dónde estoy apuntando?

Un brazo robótico no es más que un conjunto de articulaciones unidas con enlaces. Para empezar, nos gustaría poder mover el robot. Físicamente, se necesita pasarle electrónicamente a cada uno de los rotores cuántos grados se quiere que roten y los mismos rotarán. ¿Cómo se podría calcular el punto en el que terminará el extremo del brazo dados los ángulos de los motores?

Para ejemplificar, se proponen tres puntos articulados $\{0\}, \{1\}, \{2\}$ unidos por enlaces L_1, L_2. L_i, numéricamente, es la distancia entre $\{i-1\}$ e $\{i\}$. $\{0\}$ se encuentra fija al suelo, y $\{0\}$ y $\{1\}$ pueden rotar sobre un plano. En este ejemplo (y en el robot utilizado) solamente se tienen articulaciones que rotan, lo cual simplifica bastante la teoría necesaria. Estas articulaciones también podrían expandirse, moverse de manera esférica o incluso helicoidal, que por suerte no es el caso.

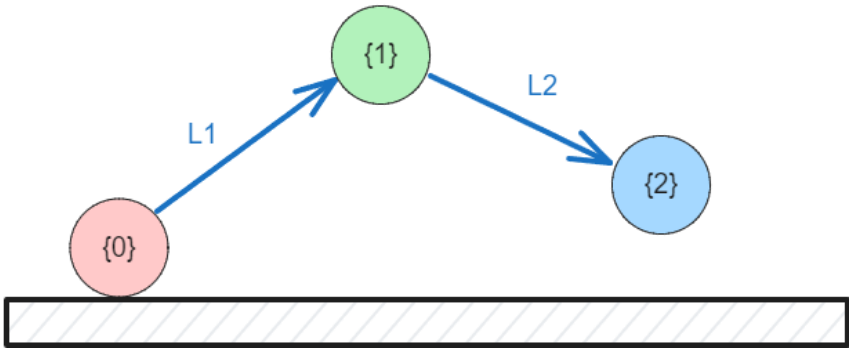

Figura 2: Puntos articulados en el plano XY.

Ahora, a cada articulación $\{i\}$ se le asignará su propio sistema de coordenadas dado por la base arbitraria $B_i = \left\{\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_i\right\}$. Cuando $\{i\}$ rote, también lo harán sus ejes y, por consiguiente, se moverán y trasladarán todos los ejes de las articulaciones $\{j\} : j > i$.

[1] https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview
[2] https://en.robotis.com
[3] https://www.turtlebot.com/turtlebot3/
[4] https://www.openrobotics.org

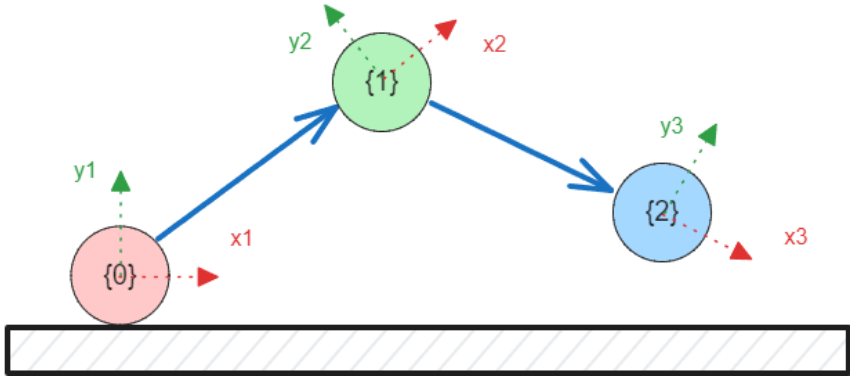

Figura 3: Los puntos articulados en el plano con sus respectivos ejes asociados.

Así, la estrategia será plantear transformaciones que nos permitan pasar de uno de estos sistemas (o espacios) a otro. Así, cuando se rote uno de los espacios, podremos ver dónde terminan los espacios siguientes.

Se plantean las transformaciones T_{i-1}^{i}, que nos permiten pasar de la «base» B_i a la «base» B_{i-1}. Pero esto es imposible. Estas transformaciones necesitan que se mueva el $\mathbf{0}$ de un espacio al siguiente, lo cual no es una transformación lineal.

2.1. Coordenadas homogéneas

Para solucionar esto, se utilizan las llamadas *coordenadas homogéneas* (Cox, Little, and O'Shea 2015, cap. 8), que hacen uso de vectores de una dimensión más y los proyectan sobre el subespacio deseado. Al igual que las transformaciones lineales tradicionales nos permiten escalar e inclinar los ejes, y además nos permite transladarlos (entre otras opciones).

Este tópico es muy extenso y admite muchos tipos de proyecciones. En nustro caso, se utilizarán matrices de este tipo:

$$H = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{d} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

$\boldsymbol{R} \in \mathbb{R}^{n\times n}$ es análoga a una matriz transformación sobre el subespacio proyectado: representa escalación, rotación, inclinación, etc. $\boldsymbol{d} \in \mathbb{R}^n$ es un vector que representa un desplazamiento del origen del subespacio. Ejemplos prácticos en $\mathbb{R}^2$ son:

$$\text{Rotación}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{Translación}(x) = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

que nos permiten rotar los ejes y trasladarlos en dirección el eje x respectivamente. Finalmente, para obtener los vectores de los subespacios transformados, se aplica

$$\mathbf{v}' = H\mathbf{v}$$
$$\begin{pmatrix} {x_1}' \\ {x_2}' \\ \vdots \\ {x_n}' \\ 1 \end{pmatrix} = H \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

quedando el vector deseado con una componente extra que será siempre 1. Cabe mencionar nuevamente que esta componente simpre queda 1 en este caso particular, y no para toda transformación en coordenadas homogéneas.

2.2. Parámetros de Denavit–Hartenberg

Ahora que tenemos un método para movernos del sistema de referencia de una articulación a la anterior con T_{i-1}^i, siendo esta traslaciones y rotaciones en el coordenadas homogéneas, la matriz transformación que nos «lleva» o nos «cambia de base» de la última unión a la base será:

$$T_0^n = T_0^1 \; T_1^2 \; ... \; T_{n-1}^n$$

¿Por qué se querría esta matriz? Porque $T_0^n \cdot (0, ..., 0, 1)^T$ será igual al origen de la punta de nuestro brazo (llamado *end-effector*) pero en las coordenadas de la base del robot (es decir, de nuestro espacio canónico). Nos permite saber dónde está posicionado el robot.

Se llama *forward kinematics* a este campo de estudio (Lunia et al. 2023, cap. 2). Hay varios métodos para obtener las matrices T_{i-1}^i, siendo el más popular en $\mathbb{R}^3$ el de Denavit–Hartenberg, conocido como *parámetros de Denavit–Hartenberg*. Consta de representar cada una de las matrices T_{i-1}^i en base a cuatro parámetros: una rotación en torno al eje z (θ), una traslación sobre el eje z (d), una traslación sobre el eje x (r) y una rotación en torno al eje x (α):

$$\begin{aligned} T_{i-1}^i &= T_{rz}(\theta_i) T_z(d_i) T_x(r_i) T_{rx}(\alpha_i) \\ &= \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & r_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & r_i \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

La ventaja de este método es que los parámetros son fáciles de obtener[5] y el método sirve para la gran mayoría de casos.

Ahora, estas matrices T_{i-1}^i son estáticas. Se necesita que varíen según ángulos arbritarios. Las articulaciones del robot *OpenManipulator X*, como se planteó anteriormente, solamente rotan. Se puede plantear que todos nuestros motores rotan en torno a sus respectivos ejes z y que sus matrices sean

$$T_{i-1}^i(q_i) = T_{rz}(q_i + \theta_i) T_z(d_i) T_x(r_i) T_{rx}(\alpha_i)$$

[5]Una visualización del método de obtención de estos parámetros se puede encontrar en youtu.be/rA9tm0gTln8.

siendo q_i los parámetros del robot y θ_i las condiciones iniciales.

2.3. Modelando el *OpenMANIPULATOR-X*

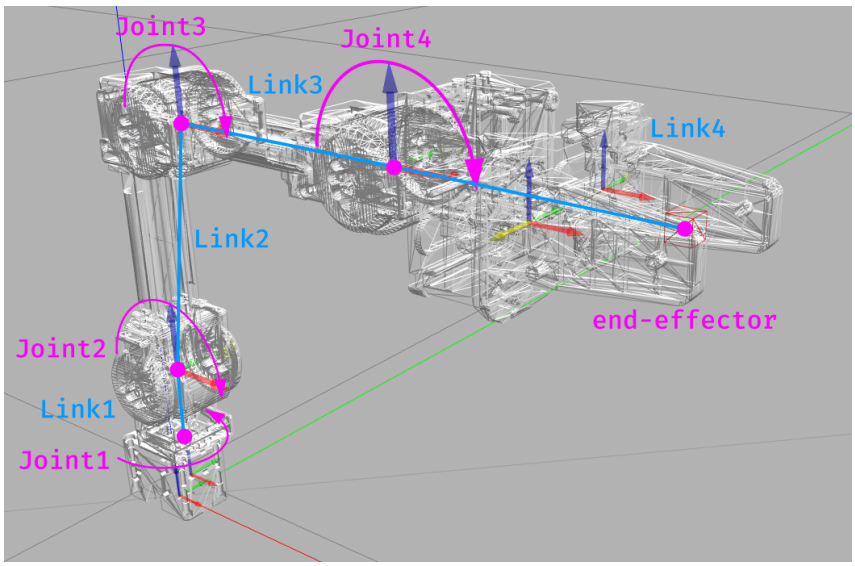

Figura 4: Representación digital del robot con anotaciones.

El robot consta de cuatro rotores. Estos se han modelado de tal forma que sus parámetros de Denavit–Hartenberg son los siguientes:

Enlace	d_i	θ_i	r_i	α_i
1	L_1	q_1	0	$\frac{\pi}{2}$
2	0	$q_2 + \frac{\pi}{2}$	L_2	0
3	0	$q_3 - \frac{\pi}{2}$	L_3	0
4	0	q_4	L_4	0

Tabla 1: Parámetros de Denavit–Hartenberg.

siendo L_i la longitud del segmento i.

Así, cada una de las matrices intermedias quedan

$$T_0^1(q_1) = T_{rz}(q_1)T_z(L_1)T_{rx}\left(\frac{\pi}{2}\right) = \begin{pmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_1^2(q_2) = T_{rz}\left(q_2 + \frac{\pi}{2}\right)T_x(L_2) = \begin{pmatrix} -\sin(q_2) & -\cos(q_2) & 0 & -L_2\sin(q_2) \\ \cos(q_2) & -\sin(q_2) & 0 & L_2\cos(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_2^3(q_3) = T_{rz}\left(q_3 - \frac{\pi}{2}\right) T_x(L_3) = \begin{pmatrix} \sin(q_3) & \cos(q_3) & 0 & L_3\sin(q_3) \\ -\cos(q_3) & \sin(q_3) & 0 & -L_3\cos(q_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_3^4(q_4) = T_{rz}(q_4) T_x(L_4) = \begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 & L_4\cos(q_4) \\ \sin(q_4) & \cos(q_4) & 0 & L_4\sin(q_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Luego de todo esto, finalmente se puede obtener la posición del *end-effector* con la función:

$$P(q_1, q_2, q_3, q_4) = \left(T_0^1(q_1)\ T_1^2(q_2)\ T_2^3(q_3)\ T_3^4(q_4)\right)\ (0, 0, 0, 1)^T$$

3. Inverse kinematics – Obteniendo los parámetros

Es muy interesante aprender sobre *forward kinematics*, pero realmente no tiene demasiadas aplicaciones. En el espacio de la robótica resulta esencial poder realizar el procedimiento inverso: averiguar los ángulos necesarios para que el brazo se ubique en una posición deseada. Obtener esta función inversa se llama *inverse kinematics* (Lunia et al. 2023, cap. 3).

Al buscar la función inversa, no siempre se podrá obtener en forma analítica, ya que hay configuraciones de robots que permiten varios (o hasta infinitos) conjuntos de ángulos para alcanzar un punto. En ese caso, se suele el método de aproximación de Newton–Raphson para obtener una solución numérica.

En el caso del *OpenManipulator X*, se logró obtener una solución analítica.

3.1. Obteniendo la solución

Este problema se puede dividir en dos problemas más sencillos. El ángulo de la primera articulación q_1, la que está en la base, solo depende las coordenadas x e y. Particularmente,

$$q_1 = \arctan\left(\frac{y}{x}\right)$$

Para el resto de ángulos q_2, q_3 y q_3, se puede aprovechar el hecho de que los tres se mueven sobre un mismo plano perpendicular al plano xy. Así, podemos hacer un cambio de variables:

$$x' = \sqrt{x^2 + y^2}$$
$$y' = z$$

Así podemos obtener la matriz de tranformación T_1^n a partir de los parámetros de Denavit–Hartenberg presentados en la tabla 1, (obviando T_0^1) pero modificada para que funcione con vectores de $\mathbb{R}^2$ en la base dada por los ejes x' e y'.

$$\omega = q_2 + q_3 + q_4$$

$$T_1^{n\prime}(q_2, q_3, q_4) = \begin{pmatrix} \cos(\omega) & -\sin(\omega) & L_2\cos(q_2) + L_3\cos(q_2 + q_3) + L_4\cos(\omega) \\ \sin(\omega) & \cos(\omega) & L_2\sin(q_2) + L_3\sin(q_2 + q_3) + L_4\sin(\omega) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{v} = T_1^{n\prime}(q_2, q_3, q_4) \cdot (x', y', 1)^T$$

ω representa el ángulo de ataque del robot. Así, nos queda un sistema de tres incógnitas:

$$
\begin{aligned}
\omega &= q_2 + q_3 + q_4 \\
x' &= L_2 \cos(q_2) + L_3 \cos(q_2 + q_3) + L_4 \cos(\omega) \\
y' &= L_2 \sin(q_2) + L_3 \sin(q_2 + q_3) + L_4 \sin(\omega)
\end{aligned}
$$

Resolviendo[6], los ángulos finales quedan

$$
\begin{aligned}
x'' &= x' - L_4 \cos(\omega) \\
y'' &= y' - L_4 \sin(\omega) \\
q_2 &= \arctan\left(\frac{y''}{x''}\right) \pm \arccos\left(\frac{(x'')^2 + (y'')^2 + (L_2)^2 - (L_3)^2}{\sqrt{(x'')^2 + (y'')^2}}\right) \\
q_3 &= \arctan\left(\frac{y'' - L_2 \sin(q_2)}{x'' - L_2 \cos(q_2)}\right) - q_2 \\
q_4 &= \omega - q_2 - q_3
\end{aligned}
$$

Esto no queda de forma matricial porque las soluciones no siempre son únicas y cuando no lo son suelen ser finitas, principalmente porque hay funciones trigonométricas de por medio.

Así, para aplicar los contenidos tratados, planteamos controlar una simulación del robot OpenMANIPULATOR-X a partir del desarrollo obtenido anteriormente.

4. Simulando el robot

El proyecto práctico realizado aplicando la teoría anterior se basa en ROS[7] (*Robot Operating System*), un set de herramientas ampliamente utilizado en el campo de la robótica para el desarrollo de software. Se creó un entorno basado en Docker para facilitar la portabilidad del proyecto en el que se incluye la simulación en Gazebo[8] provista por Robotis y el programa que se desarrolló.

Este programa es un CLI (*command-line interface*) permite el ingreso de coordenadas cartesianas que son transformadas a parámetros y enviadas al robot. Esta CLI puede ser utilizada tanto para controlar el robot simulado como un robot real. La función de ingreso de posición recibe cuatro datos, la posición en los ejes x, y y z y el ángulo del *end-effector* al momento de llegar a la posición destino.

El proyecto desarrollado incluyendo el entorno de simulación se encuentra disponible en github.com/b-Tomas/robot-kinematics de manera abierta.

Bibliografía

Cox, D. A., Little, J., & O'Shea, D. (2015). Ideals, varieties, and algorithms (4th ed.). *Undergraduate Texts in Mathematics*. https://doi.org/10.1007/978-3-319-16721-3

Lunia, A., Stevens, A., Holt, C., Morgan, R., Norris, J., Poyyamozhi, S., & Zhong, Y. (2023, March 21). *Modeling, motion planning, and control of manipulators and mobile robots.* Clemson University Open Textbooks. https://opentextbooks.clemson.edu/wangrobotics

[6] Una explicación paso a paso puede verse en youtu.be/1-FJhmey7vk.

[7] https://www.ros.org

[8] https://gazebosim.org/home