Revista Sinaloense de Ciencia, Tecnología y Humanidades

RSCTyH, VOL. I, No.01, Enero 2023

Desarrollo de librería para robótica humanoide en ambiente python

Artículo de Investigación

Humanoid Robotics Library development in python

Alan K. Abitia-González[1], Graciela Rodríguez Vega[2], Aldo N. Higuera Juarez[1], Dora A. Rodríguez-Vega[1*],

[1]Departamento de Ingeniería Mecatrónica, Universidad Politécnica de Sinaloa.
[2]Departamento de Ingeniería Industrial, Universidad de Sonora.
* Autor correspondiente: drodriguez@upsin.edu.mx

Resumen: El desarrollo de librerías en el área de robótica ha sido una tarea de interés debido a la cantidad de datos y operaciones matemáticas requeridas, a lo largo de los años se han desarrollado librerías en ambientes como C++, Java, MATLAB ® y más recientemente en Python. Los enfoques de desarrollo incluyen navegación autónoma, visión, simulación del comportamiento cinemático y/o dinámico, planificación de trayectorias entre otras. Se propone una librería en ambiente Python que proporciona herramientas para generar el modelo cinemático directo e inverso, generar trayectorias en espacio articular y cartesiano mediante interpolación lineal y finalmente simular las trayectorias en un ambiente gráfico tridimensional.

Palabras clave: *Python, Humanoide, Modelo Cinemático.*

Abstract: Development of libraries in the robotics area has been a task of interest due to the amount of data and mathematical operations required, over the years libraries have been developed in environments such as C++, Java, MATLAB® and more recently in Python. Approaches include autonomous navigation, computer vision, simulation of kinematic and/or dynamic behavior, trajectory planning among others. A library in Python environment is proposed, provides tools to generate direct and inverse kinematic model, trajectories by lineal interpolation in joint and cartesian space and finally simulate the trajectories in a three-dimensional graphic environment.

Keywords: *Python, Humanoid, Kinematics model*

Recibido:20/10/22; Aceptado: 12/12/2022; Publicado 25/01/2023

Introducción

A lo largo de los años se han desarrollado algunas librerías o paquetes para su uso en aplicaciones de modelado cinemático y dinámico en robótica desde enfoques de aplicación diversos. La librería Robotics Toolbox for MATLAB® [1], fue creada en el año 1991 y publicada en 1995, durante 25 años se ha actualizado a la par de las versiones de MATLAB®. El libro Robótica del autor John Craig [2] hace uso de esta librería en sus ejercicios. Aunque la librería es de código abierto, es necesario contar con MATLAB® para poder hacer uso de ella. En el ambiente Python las librerías PythonRobotics [3] y Klampt [4] se enfocan en navegación autónoma y planificación, iDynTree [5], Siconos [6] , PyDy [7] , DART [8], and PyBullet [9] proporcionan herramientas para evaluar el comportamiento dinámico; Pybullet y DART combinan C++ y Python para proporcionar ambientes gráficos de simulación dinámica. Pybotics [10] se concentra en las cadenas cinemáticas pero solo usa la notación de Denavit y Hartenberg modificada [11]. Python-robotics [12], [13] contiene algunas funciones de robótica. El análisis cinemático y dinámico de un robot humanoide implica el análisis de cuatro o más cadenas cinemáticas abiertas con un marco base móvil común para las cuatro cadenas. Por lo que se propone una librería en ambiente Python para el manejo del modelo cinemático del robot en conjunto, tal librería se enfoca en generar el modelo cinemático, directo e inverso, de robots humanoides, generar trayectorias y mostrar los resultados en un ambiente gráfico. Aunque la librería fue diseñada para trabajar con robots humanoides, por la forma en que se estructuró, es posible utilizarla en robots manipuladores de cadena cinemática abierta.

Materiales y Métodos

Se utilizó el ambiente Python debido a que es un lenguaje abierto y cuenta con recursos que brindan portabilidad, gráficos rápidos tridimensionales, operaciones numéricas y simbólicas rápidas, interfaces de usuario, entre otros. El código se implementó en Python 3.6, usa los paquetes de Numpy para el manejo de arreglos y PyYaml para la serialización de datos que corresponden a la configuración del robot. La figura 1 muestra la estructura general de la librería propuesta.

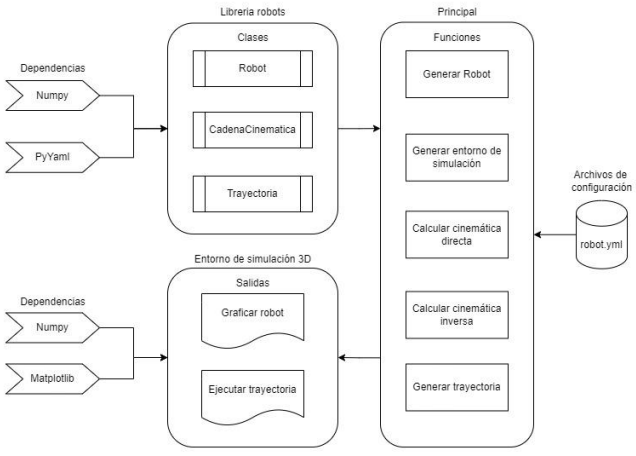

Figura 1. Estructura general de la Librería Propuesta.

La librería consta de tres clases (*robot*, *cadenaCinemática* y *Trayectoria*) y de una colección de funciones que permiten crear y manipular un robot, en la Tabla 1 se muestran las propiedades de cada una de las clases, mientras que en la Tabla 2 se presentan los métodos y en la Tabla 3 las funciones estáticas. En la clase robot se indica el nombre y tipo de robot, su posición y orientación en el espacio, mediante matriz homogénea del marco base (definida por la ecuación 1) así como la cantidad de cadenas cinemáticas abiertas que conforman al robot humanoide, se describe cada cadena cinemática mediante objetos de la clase *CadenaCinemática*.

Tabla 1. Propiedades de las clases de la Librería Propuesta

Propiedades		
Robot		
Nombre	Descripción	Tipo de dato
datosrobot	Almacena los datos completos del robot	Diccionario
nombre	Nombre del robot	String
tipo	Tipo de robot	String
base	Matriz de transformación de la base del robot	Numpy Array (Float)
numerocadenas	Cantidad de cadenas que conforman el robot	Int
cadenas	Lista de cadenas que conforman el robot	Numpy Array (CadenaCinematica)
CadenaCinematica		
datoscadena	Almacena los datos completos de la cadena	Diccionario
nombre	Nombre de la cadena	String
gdl	Cantidad de grados de libertad de la cadena	Lista (Int)
Apre	Matriz local sin considerar variables articulares	Numpy Array (Float)
A	Matriz local considerando variables articulares	Numpy Array (Float)
H	Matriz global respecto al origen	Numpy Array (Float)
posfin	Lista de posiciones finales para los puntos de interés del robot. Funciona como auxiliar para graficar el robot	Numpy Array (Float)
Aref	Matriz local de los marcos de referencia	Numpy Array (Float)
posmref	Lista de posiciones finales para los marcos de referencia del robot. Funciona como auxiliar para graficar el robot	
Trayectoria		
nombrecadena	Nombre de la cadena a la que pertenece la trayectoria	String
puntos	Puntos cartesianos de la trayectoria	Numpy Array (Float)
intervalosdetiempo	Intervalos de tiempo entre puntos de la trayectoria	Numpy Array (Float)
interpolacion	Tipo de interpolación de movimiento de la trayectoria	String
dt	Delta T considerado para generar la interpolación de movimiento	Float
q	Valores articulares de los puntos de la trayectoria	Numpy Array (Float)
dq	Trayectoria completa en intervalos de Delta T.	Numpy Array (Float)

Tabla 2. Métodos de la librería propuesta

Métodos		
Robot		
Nombre	Descripción	Datos de entrada
__init__	Método constructor. Crea un objeto de la clase Robot, e inicializa las propiedades iniciales.	datosrobot
crearcadenas	Extrae las cadenas de los datos del robot y las agrega a una lista para ser utilizadas	lista de cadenas
CadenaCinematica		
__init__	Método constructor. Crea un objeto de la clase CadenaCinematica, e inicializa las propiedades iniciales.	datoscadena
crearprelocales	Crea las matrices locales sin considerar variables articulares	paramsdh
calcularlocales	Crea las matrices locales considerando las variables articulares	angulos
cinematicadirecta	Calcula las matrices globales y obtiene las posiciones finales de los puntos de interés	angulos
obtenerposiciones	Extrae las posiciones finales de los puntos de interés desde su matriz global	
cinematicainversa	Calcula las variables articulares de la cadena a partir de una posición ingresada	x, y, z, codo
crearlocalesmref	Crea las matrices locales de los marcos de referencia	
obtenermref	Extrae las posiciones finales de los marcos de referencia de los puntos de interés	
Trayectoria		
__init__	Método constructor. Crea un objeto de la clase Trayectoria, e inicializa las propiedades iniciales.	cadena, puntos, intervalosdetiempo, interpolacion
crearvaloresarticulares	Extrae los puntos cartesianos de la trayectoria y los convierte en valores articulares	cadena
creartrayectoria	Crea la trayectoria a partir de los valores articulares, y los parámetros del tipo de trayectoria.	cadena

Tabla 3. Funciones Estáticas

Funciones		
Nombre	Descripción	Datos de entrada
extraerdatos	Extrae los datos del robot desde archivo YAML y los almacena en un diccionario	ubicacion
calcdh	Calcula una matriz local a partir de los parámetros DH	theta, d, a, alpha
cibi	Calcula las variables articulares del brazo izquierdo a partir de una posición ingresada	x, y, z, codo
cibd	Calcula las variables articulares del brazo derecho a partir de una posición ingresada	x, y, z, codo
interpolacionlineal	Crea la interpolación de movimiento entre dos puntos y dos intervalos de tiempo en intervalos de Delta T	qini, qfin, tini, tfin, dt

$$R^0_{base} = R_{z\psi}R_{y\phi}R_{x\theta} = \begin{bmatrix} c\phi c\psi & c\psi s\theta s\phi - c\theta c\psi & s\theta s\psi + c\theta c\psi s\psi & X \\ c\phi s\psi & c\theta c\psi + s\theta s\phi s\psi & c\theta s\phi s\psi - c\psi s\theta & Y \\ -s\phi & c\phi s\theta. & c\theta c\phi & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Por ejemplo, un robot humanoide tiene una cadena cinemática para cada pierna y para cada brazo, cuatro en total, y algunos robots cuenta con actuadores en el torso y/o cabeza incrementando el número de cadenas. Para el caso de estudio del Robot Humanoide Bioloid Premium™ mostrado en la figura 2, el robot cuenta con 18 grados de libertad distribuidos en cuatro cadenas cinemáticas mostradas en la Figura 3.

Figura 2. Robot Humanoide Bioloid Premium™

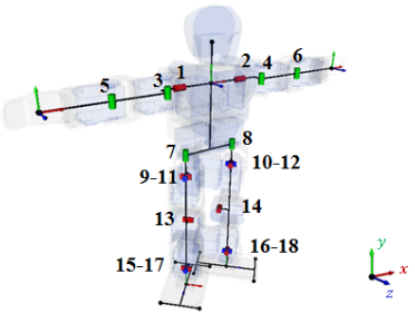

Figura 3. Cadenas Cinemáticas y articulaciones del robot Humanoide Bioloid Premium™

En la clase cadena Cinemática se indica el nombre asociado a la cadena, el número de grados de libertad, las matrices homogéneas locales y globales, así como las posiciones del final de cada eslabón. En cuanto a los métodos de la cadena cinemática se propone crear las matrices locales a partir de los parámetros Denavit Hartenberg con el método *calcularlocales*; dichos parámetros se obtienen mediante la función *extraedatos* a partir de un archivo descriptivo del robot en formato *YML* usando el paquete de Python PyYAML. Los parámetros Denavit Hartenberg son: *a, d,* α y θ para cada articulación, siguiendo la metodología DenaVit Hartenberg [14] se calculan las matrices locales mediante la ecuación 2, posteriormente se calculan las matrices globales con el método cinemática directa, de acuerdo a la ecuación 3, o bien se calculan las variables articulares a partir de posiciones deseadas mediante el método de cinemática inversa [15]; se presentan también los métodos para extraer la información de posición del final de cada eslabón de las matrices globales mediante *obtenerposiciones* los cuales corresponden a los primeros tres renglones de la cuarta columna de las matrices globales.

$$A_i = \begin{bmatrix} c\theta & -s\theta c\alpha & s\theta s\alpha & ac\theta \\ s\theta & c\theta c\alpha & -c\theta s\alpha & as\theta \\ 0 & s\alpha & c\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$H = T^0_n = A_1 \cdots A_n \quad (3)$$

En la clase *Trayectorias* las propiedades indican el nombre de la cadena para la trayectoria, los puntos en el espacio cartesiano y en el espacio articular para generar la trayectoria, el incremento de tiempo entre los puntos a generar, el tipo de interpolación a utilizar y el incremento de tiempo. El método *crearvalores* articulares extrae información de las posiciones articulares que se obtuvieron por el método de *cinemáticainversa* de la clase cadena cinemática a partir de puntos deseados en el espacio cartesiano, tales posiciones articulares y el tipo de interpolación seleccionado se usan por el método *creartrayectoria* para proporcionar un historial de posiciones.

Resultados y Discusión

Se analizó el caso del robot humanoide Bioloid Premium™ de 18 grados de libertad distribuidos en cuatro cadenas que se muestran en la figura 4, se calcularon las matrices locales y globales y se grafican las posiciones de cada articulación de las cuatro cadenas cinemáticas desde el marco base del robot. Se generaron trayectorias mediante interpolación lineal para el brazo derecho con posiciones deseadas en [0,100,50] y [0, -100, 50], mostrando gráficamente las trayectorias de cada posición articular en la figura 5.

Conclusiones

La librería se diseñó originalmente con el objetivo de proveer, al robot Bioloid Premium™, las herramientas necesarias para facilitar y simplificar el cálculo de su cinemática directa e inversa, la generación de trayectorias de movimiento y preparar los datos que serían enviados al entorno de simulación. No obstante, por su manera de procesar los datos, se consiguió que la librería puede ser utilizada para simular cualquier tipo de robot que cuente con cadenas cinemáticas abiertas en su

estructura. Como trabajo futuro se pretende implementar más algoritmos para la generación de trayectorias, incluir la simulación dinámica mediante PyBullet y las propiedades dinámicas del robot y finalmente agregar los archivos CAD del robot para que se observen en el gráfico del simulador.

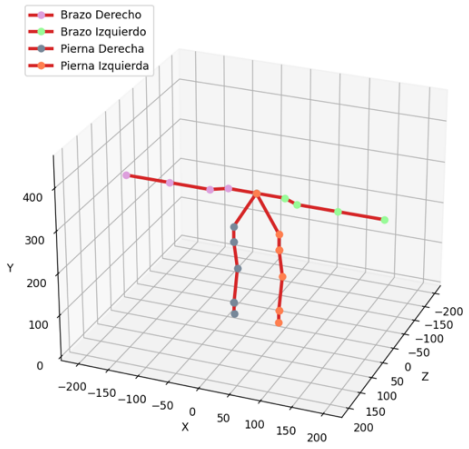

Figura 4. Gráfica del robot BIOLOID en su posición inicial

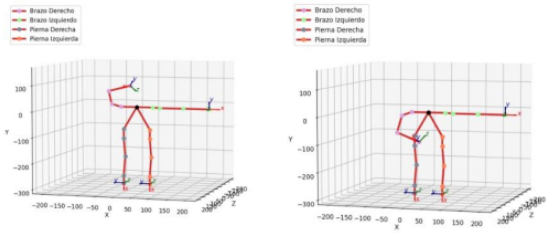

Figura 5. Gráfica de algunas posturas del robot durante la trayectoria calculada.

Conflicto de Intereses

Los autores expresan que no existen conflictos de interés al redactar el manuscrito.

Referencias

[1] P. I. Corke, «A robotics toolbox for MATLAB», *IEEE Robot. Autom. Mag.*, vol. 3, n.º 1, pp. 24-32, mar. 1996, doi: 10.1109/100.486658.

[2] J. J. Craig, *Robótica*, 3. ed. Mexico: Pearson educacion, 2006.

[3] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, y A. Paques, «PythonRobotics: a Python code collection of robotics algorithms», 2018.

[4] K. Hauser, «klamp't. (Kris' Locomotion and Manipulation Planning Toolbox)». 2022. [En línea]. Disponible en: https://github.com/krishauser/Klampt

[5] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, y D. Pucci, «iCub Whole-Body Control through Force Regulation on Rigid Non-Coplanar Contacts», *Front. Robot. AI*, vol. 2, mar. 2015, doi: 10.3389/frobt.2015.00006.

[6] V. Acary, O. Huber, y M. Shpakovych, «Simulation framework for nonsmooth dynamical systems». 2022. [En línea]. Disponible en: https://github.com/siconos/siconos

[7] G. Gede, D. L. Peterson, A. S. Nanjangud, J. K. Moore, y M. Hubbard, «Constrained Multibody Dynamics With Python: From Symbolic Equation Generation to Publication», en *Volume 7B: 9th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, Portland, Oregon, USA, ago. 2013, p. V07BT10A051. doi: 10.1115/DETC2013-13470.

[8] J. Lee *et al.*, «DART: Dynamic Animation and Robotics Toolkit», *J. Open Source Softw.*, vol. 3, n.º 22, p. 500, feb. 2018, doi: 10.21105/joss.00500.

[9] E. Coumans y Y. Bai, «PyBullet, a Python module for physics simulation for games, robotics and machine learning». 2022. [En línea]. Disponible en: https://pybullet.org/

[10] N. Nadeau, «Pybotics: Python Toolbox for Robotics», *J. Open Source Softw.*, vol. 4, n.º 41, p. 1738, sep. 2019, doi: 10.21105/joss.01738.

[11] M. Granja, N. Chang, V. Granja, M. Duque, y F. Llulluna, «Comparison between Standard and Modified Denavit-Hartenberg Methods in Robotics Modelling», presentado en The 2nd World Congress on Mechanical, Chemical, and Material Engineering, jul. 2016. doi: 10.11159/icmie16.118.

[12] R. Fraanje, T. Koreneef, A. Le Mair, y S. de Jong, «Python in robotics and mechatronics education», en *2016 11th France-Japan & 9th Europe-Asia Congress on Mechatronics (MECATRONICS) /17th International Conference on Research and Education in Mechatronics (REM)*, Compiègne, France, jun. 2016, pp. 014-019. doi: 10.1109/MECATRONICS.2016.7547108.

[13] R. Fraanje, «Robot kinematics with python-visual visualization». 2022. [En línea]. Disponible en: https://github.com/prfraanje/python-robotics

[14] M. W. Spong, S. Hutchinson, y M. Vidyasagar, *Robot Modeling and Control*, 1st ed. Wiley, 2005.

[15] J. V. Nunez, A. Briseno, D. A. Rodriguez, J. M. Ibarra, y V. M. Rodriguez, «Explicit Analytic Solution for Inverse Kinematics of Bioloid Humanoid Robot», en *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, Fortaleza, Brazil, oct. 2012, pp. 33-38. doi: 10.1109/SBR-LARS.2012.62.