Jornadas de Automática

Proyecto “SimRobots”: Colaboración entre “SimScape Multibody” y “Robotic System Toolbox”

Herreros López, Alberto.[a,*]

[a] *GIR Tecnologías avanzadas de la producción, Universidad de Valladolid, Paseo Prado de la Magdalena 3-5, 47011 Valladolid*

To cite this article: Herreros López, A. 2024. “SimRobots” Project: Collaboration between “SimScape Multibody” and “Robotic System Toolbox”. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10770

Resumen

El proyecto “SimRobots” descrito en este artículo representa la segunda versión del mismo. La principal diferencia con la primera versión radica en el uso de todos los elementos de la librería de Simulink “SimScape MultiBody” y, sobre todo, de la librería de Matlab “Robotic System Toolbox”. El objetivo del proyecto es crear una herramienta que ayude a los alumnos a modelar y programar los movimientos de una estación robótica. Aunque la librería “SimScape MultiBody” es una herramienta de simulación potente, carece de control de trayectorias. Por otro lado, la librería “Robotic System Toolbox” ofrece numerosas herramientas de control de trayectorias, pero la simulación en Matlab no es completamente realista. Este proyecto busca combinar ambas librerías para crear un simulador de robots comparable a los softwares de marcas comerciales, al tiempo que permite a los alumnos comprender las matemáticas subyacentes en la simulación. Hasta ahora, el proyecto ha sido utilizado durante cuatro años con resultados positivos.

Palabras clave: Tecnología Robótica, Robot, Robot Manipuladores, Ingeniería de control

“SimRobots” Project: Collaboration between “SimScape Multibody” and “Robotic System Toolbox”

Abstract

The “SimRobots” project described in this article represents the second version of it. The main difference with the first version lies in the use of all the elements of the Simulink library “SimScape MultiBody” and, above all, the Matlab library “Robotic System Toolbox”. The objective of the project is to create a tool that helps students to model and program the movements of a robotic station. Although the “SimScape MultiBody” library is a powerful simulation tool, it lacks trajectory control. On the other hand, the “Robotic System Toolbox” library offers numerous trajectory control tools, but the simulation in Matlab is not completely realistic. This project seeks to combine both libraries to create a robot simulator comparable to commercially available software, while allowing students to understand the mathematics underlying the simulation. So far, the project has been used for four years with positive results.

Keywords: Robotic Technology, Robots, Robotic Manipulators, Automatic Control.

*Autor para correspondencia: autor1@ceautomatica.es

1. Introducción

La robótica industrial se ha convertido en un área de docencia emergente debido a su demanda industrial y social (Barrientos, A 2007). Sin embargo, la educación en robótica no es sencilla ya que la mayoría de los robots industriales tienen su propio software. Estos softwares han sido realizados con un objetivo industrial y no proporcionan información sobre lo que matemáticamente están realizando. Un ejemplo claro es que es casi imposible encontrar dos marcas de robot que usen las mismas unidades para definir los puntos de una trayectoria.

Los primeros trabajos realizados por nuestro equipo sobre la simulación de robot industriales usando Matlab y Simulink fueron desarrollados en (Mato, A 2014). Hace unos años se propuso la primera versión del proyecto "SimRobots" basado en la librería de Simulink "Simscape Multibody", para la simulación y control de robot industriales (Arévalo S. 2021). Dicha herramienta usaba como software de control de trayectorias la librería de Corke (Corke, P. 2016). La idea de este proyecto tiene su origen en el software ARTE (Gil A. 2015). La diferencia básica entre ambos proyectos es que ARTE simula los robots en Matlab, mientras que nuestro proyecto lo realiza en "Simscape Multibody", lo que mejora la simulación. En (Gil A. 2024) una nueva versión de ARTE, pyARTE, usa una librería de python para control y simulación. La nueva versión de "SimRobot" usa la librería "Robotic System Toolbox" de Matlab en vez de la librería de Corke para el control de trayectorias del robot. El objetivo es aprovechar los avances que esta librería ha introducido. Por ejemplo, la cinemática inversa puede ser hallada con cuatro algoritmos diferentes.

La estación robótica estará formada por dos tipos principales de elementos, los elementos estáticos (objetos de trabajo, herramientas y cargas) y los elementos con articulaciones, es decir, los robots. Los elementos estáticos están diseñados por un icono de Simulink con máscara donde se define las propiedades dinámicas del elemento y su fichero visual correspondiente, en formato **.stl*. La información de estos elementos está contenida en un objeto que permite definir el elemento de forma sencilla e intercambiar la información de un icono a otro. Esta es la herramienta básica para tomar y dejar cargas.

Los robots pueden ser definidos manualmente o importados de programas CAD. Los robots pueden ser importados en formato DH (Denavit-Hartenberg), (Pérez, M 2014) o en formato URDF (Unified Robot Description Format) de ROS (Robotic Operative System) (MathWorks, 2024h). La librería "Simscape Multibody" permite importar robots en formato URDF y el manejo de los mismo es mucho más sencillo que en ROS. El robot creado es incluido en un icono con máscara. De esta forma, el modelo Simulink es sencillo de entender con iconos robots, herramientas, cargas y objetos de trabajo.

Las entradas de las articulaciones "Simscape Multibody" puede ser dinámicas o cinemáticas. Las dinámicas, a partir de par motor o fuerza, obtienen el ángulo o la posición. Las cinemáticas, a partir de posición o ángulo, obtienen la fuerza o par motor necesario. Con ello se puede diseñar estaciones cinemáticas y dinámicas.

Si queremos definir una trayectoria del TCP (*Tool Center Point*) en el espacio cartesiano, es preciso calcular la cinemática inversa (Pérez, M 2014) con una herramienta externa a "Simscape Multibody". Este proyecto usa la librería "Robotic System Toolbox". Esta librería también puede importar robots en formato URDF y lo más importante, importar los elementos de una estación definida en "Simscape Multibody". De forma paralela, se va a poder trabajar con el fichero Simulink de la estación en "Simscape Multibody", y el objeto *RigidBody* de la librería "Robotic System Toolbox".

Esta es la clave del proyecto "SimRobots", calcular trayectorias con la librería "Robotic System Toolbox" para luego simularlas con "Simscape Multibody". Para que este objetivo sea más sencillo, se ha diseñado un objeto que combina la información de ambos sistemas, calcula trayectorias con "Robotic System Toolbox" y las simula con "Simscape Multibody". Este objeto tiene métodos para definir movimientos en cinemática directa, fijando la posición final articular, y movimientos en cinemática inversa, fijando la posición cartesiana del TCP respecto a una referencia.

La ejecución de la estación robótica activa un programa simulador "Mechanics Explorers" (MathWorks, 2024a) que logra una definición similar a los softwares de los robots comerciales. La velocidad de simulación puede ser modificada. Se simula el movimiento del robot a la vez que se dibuja un rastreador con las posiciones de la trayectoria del TCP del robot.

2. Modelado de una estación robótica en "SimScape Multibody"

La estación robótica se va a diseñar en "SimScape Multibody" usando principalmente dos tipos de iconos, los iconos estáticos y los iconos de robots.

2.1. Definición de ejes cartesianos

Los ejes cartesianos para este proyecto van a ser definidos como matrices homogéneas (4x4) o en el formato vector de posición-rotación $\left[[x,y,z]_{\mathrm{m}},\left[r_z,r_y,r_x\right]_{\mathrm{rad}}\right]$. Sin embargo, se pueden usar cualquiera de los objetos *SO3* de "Robotic System Toolbox" y convertirlos a este formato con la función *Prod()*, que devuelve ambos formatos. La misma función vale para multiplicar dos ejes definidos en diferentes formatos, incluidos *Pieza()* y *Cin()*, véase los siguientes apartados.

Las trayectorias cartesianas estarán definidas por una matriz de vectores posición-rotación.

2.2. Elementos estáticos de la estación

Se definen todos los elementos estáticos en un icono con máscara. La entrada es un objeto de la clase *Pieza()* que tiene todas la información del elemento. El interior del icono contiene una parte visual y otra de inercia. La función inicio del icono modifica el mismo con respecto al objeto de entrada. Los elementos estáticos que van a estar unidos al robot, herramienta y/o carga, disponen de una articulación fija a

mayores que va a servir para fijar el TCP de la herramienta y/o carga, ver la figura 1.

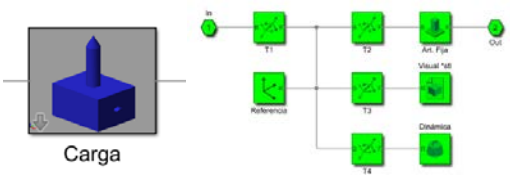

Figura 1: Máscara e interior de un elemento estático

La figura 2 muestra un conjunto de elementos estáticos. Hay iconos con una entrada e iconos con una entrada y salida. Los hilos del modelo trasmiten la posición relativa de los iconos. El objeto asociado al icono define la base y salida del mismo con respecto al anterior. Por ejemplo, la posición de la semi-esfera (objeto *pExt1*) se define con respecto al eje base total, mientras que la torre (objeto *pTab1*) se define respecto de la posición de salida del tablero (objeto *TabAje*).

Hay iconos nulos (objeto *pExt4*) que van a servir de comodín para intercambiar información entre iconos. El método *PegarEn()* consigue mover la información de un icono, por ejemplo *pTab1* (Torre) a otro icono, *pExt4* (nulo). Visualmente no vemos la diferencia en la simulación, pero en el esquema de Simulink la torre ha cambiado de icono. Si se mueve el tablero, se moverá sin la torre, que ahora pertenece a la base. Está va a ser la estrategia para que un robot tome una carga de un icono y la devuelva a otro icono, véase apartado *3.2*.

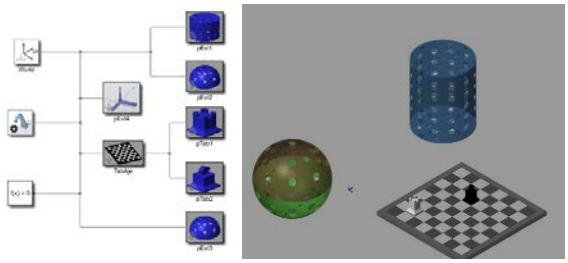

Figura 2: Conjunto de elementos estáticos

2.3. Elementos con articulaciones, robots

Los elementos con articulaciones, robots, se define a base de iconos de rotación, prismáticos o fijos, e iconos de brazo, similares a los elementos estáticos, véase figura 3.

La definición de su estructura puede ser manual o bien importando la misma del formato DH o URDF del robot. Se ha diseñado un icono estándar para los robots definidos en formato DH. En el interior de la máscara se definen la matriz de valores DH $[\theta, d, a, \alpha]$ del robot y los límites de las articulaciones (Pérez, M 2014). Los robots modelados en la librería ARTE podrían ser importados con este formato.

Los robots importados del formato URDF (MathWorks, 2024a) generan una estructura de articulaciones y brazos acorde con la figura 3.

En ambos casos, es preciso definir los actuadores de las articulaciones e introducir en los iconos visuales la ubicación exacta del fichero **.stl* de cada brazo. Cada articulación debe tener un actuador que puede ser dinámico o estático. Si el actuador es dinámico, la entrada es el par motor o la fuerza. Si el actuador es cinemático, la entrada es el ángulo y derivadas o posición y derivadas.

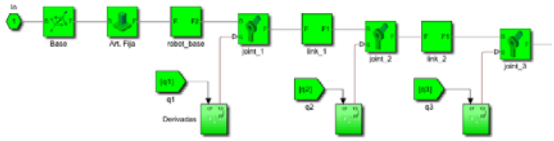

Figura 3: Articulaciones y brazos de un robot cinemático

El robot se introduce en un icono con mascara cuyas variables de entrada son la posición de la base y el valor inicial de sus articulaciones. La figura 4 muestra un robot con máscara y dos iconos estáticos, la herramienta y la carga, y su simulación.

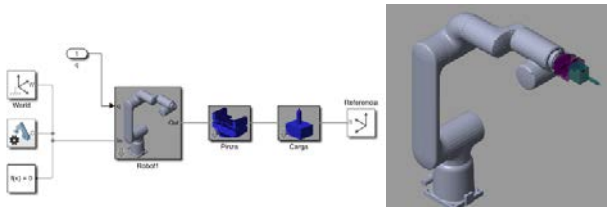

Figura 4: Icono robot, con herramienta y carga

3. Importar la estación robótica "Robotic System Toolbox" desde "SimScape MultiBody"

La simulación de la estación definida en "SimScape Multibody" solo puede simular la cinemática directa. Si se desea mover el TCP del robot a través de trayectorias cartesianas, necesitamos algoritmos de cinemática inversa (Pérez, M 2014). El cálculo de la cinemática inversa se va a realizar con el uso de la librería "Robotic System Toolbox". Esta librería puede importar el fichero Simulink de la "SimScape MultiBody" a un objeto *RigidBody* con un árbol de articulaciones equivalente. Los cálculos de trayectorias y cinemática inversa van a ser realizados en el objeto *RigidBody* y la simulación en Simulink.

Para facilitar esta tarea se ha definido un objeto *Cin()* que posee información de los dos sistemas equivalentes. Los métodos de este objeto proporcionan trayectorias que serán calculadas con el objeto *RigidBody* y simuladas en "SimScape Multibody". Esta es la clave de este proyecto.

3.1. Algoritmos para realizar la cinemática directa e inversa

Una de las razones para usar en esta versión "Robotic System Toolbox" es poder usar algoritmos más complejos para el cálculo de trayectorias. El objeto *Cin()* dispone de los siguientes algoritmos:

- **Algoritmo de optimización local con pesos**: Se usa el objeto *robotics.InverseKinematics* para obtener la optimización en función de pesos en los objetivos de posición y rotación.
- **Algoritmo de optimización generalizado**: Se emplea el objeto *generalizedInverseKinematics* para obtener la optimización con restricciones. Estas restricciones pueden ser muy diversas. Por ejemplo, espacio cartesiano donde se puede mover el robot, límites en las articulaciones del robot, y más.
- **Algoritmo de optimización analítica**: Se emplea el objeto *analyticalInverseKinematics* para definir un fichero que resuelva de forma analítica la cinemática

inversa. No todos los robots pueden resolver su cinemática inversa por este método. Solo si las tres últimas articulaciones son esféricas es posible formular esta función (Pérez, M 2014).

- **Algoritmo de optimización incremental**: Basado en el jacobiano, se puede formular un algoritmo incremental para obtener la cinemática inversa. Los incrementos de ángulos deben formularse por la fórmula de rotación de Rodrigues, que evita discontinuidades en el incremento angular (Pérez, M 2014).
- **Algoritmos de manipuladores**: Se emplea el objeto *manipulatorRRT* para obtener la interpolación entre ejes articulares salvando restricciones.

3.2. *Métodos principales de la clase Cin()*

El objeto *Cin()*, usando los algoritmos definidos en el apartado anterior, dispone de los siguiente métodos de movimiento.

- ***MoveAbsJ()***: Movimiento por ejes articulares de una posición a otra.
- ***MoveJ()***: Movimiento del TCP del robot a una posición dada con respecto a una referencia, usando la iteración del objeto *SO3*.
- ***MoveL()***: Movimiento del TCP del robot a una posición con respecto a una referencia, en línea recta.
- ***MoveC()***: Movimiento del TCP del robot a dos posiciones dadas con respecto a una referencia, en un arco de semicircunferencia.

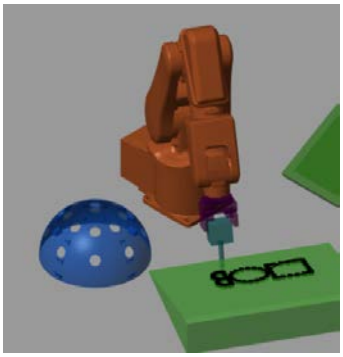

Figura 5: Trayectorias lineal y circular

A partir de las trayectorias definidas por los métodos anteriores, el objeto *Cin()* simula en Simulink dicha trayectoria. La velocidad de la simulación puede ser definida. Las diversas trayectorias que se van trazando con el robot pueden ser grabadas de forma que luego se puedan repetir. "SimScape MultiBody" dispone de un icono que puede dibujar trayectorias en el espacio, y éste es usado como rastreador de las trayectorias del TCP del robot. El resultado final puede ser el mostrado en la figura 5. El robot se mueve en línea recta para crear el cuadrado, con dos movimientos circulares para generar la circunferencia y en línea recta y circular para crea la letra B. Todos los movimientos del robot se han realizado respecto del objeto de trabajo mesa inclinada.

3.2. *Tomar y dejar carga*

Las principales acciones de un robot son su movimiento y su capacidad para tomar y dejar cargas. Este segundo apartado se obtiene con el método *PegarEn()* definido en el objeto *Pieza()*, véase apartado 2.2. Se muestra a modo de ejemplo el código de la figura 6 para tomar una torre del tablero:

```
% Ir a posición inicial
rob.MoveAbsJ([0,0,0,0,90,0]*pi/180);
% Punto por encima de la torre girado
punto= [[0,0,50]*1e-3,[0,0,180]*pi/180];
% ir al punto, referencia pTab1(torre)
rob.MoveJ(punto, Prod(TabAje, pTab1));
% Pegar en Carga (rob), pTab1 (TabAje)
PegarEn(Carga, pTab1, rob, TabAje);
% Volver con la pieza pegada
rob.MoveAbsJ([0,0,0,0,90,0]*pi/180);
```

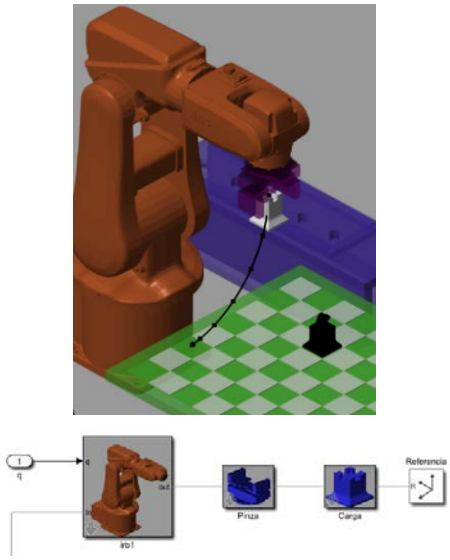

Figura 6: Pegar en *Carga*, *pTab* (torre) y mover

3.3. *App de movimiento*

El proyecto "SimRobots" incorpora varias App realizadas en "Appdesigner Toolbox" (MathWorks, 2024g). La más útil puede ser la mostrada en la figura 7. Se puede usar para cualquier robot formulado con el objeto *Cin()*. El movimiento se realiza sobre una referencia dada, incluida el TCP del robot. También se puede mover por ejes articulares de forma absoluta para mover el robot a una posición de inicio.

Figura 7: App para el movimiento de un robot

3.4. *Robot con controladores reales y robots colaborativos*

Los robots diseñados hasta ahora tienen articulaciones con entradas cinemáticas, rotación o posición y sus derivadas. El par motor o fuerza es calculado internamente por la

articulación. La articulación puede tener entrada dinámica, par motor o fuerza, y la articulación calcula la rotación o posición correspondiente. Un robot dinámico tiene que ser realimentado externamente para poder alcanzar una posición dada. Esta realimentación se puede realizar en Simulink a partir de las señales medidas por sensores, véase figura 8.

El robot realimentado puede usar el objeto *Cin()* para definir sus movimientos, y se comprobará el efecto de los controladores y la realimentación. Cuando un robot toma una carga con mucho peso se pueden ver y medir las oscilaciones de los brazos del mismo.

También es posible modelar robots colaborativos con fuerzas externas. Cuando el par motor de las articulaciones no concuerde con el esperado, dinámica inversa (Pérez, M 2014), se realimenta la diferencia al sistema para que el robot se mueva en dirección contraria a la fuerza externa.

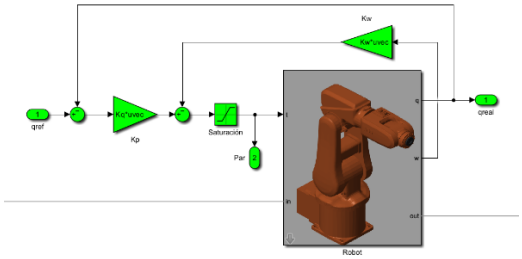

Figura 8: Robot dinámico realimentado con un PD

3.5. *Estaciones Multi-Robots*

Uno de los principales problemas de la conversión entre "SimScape MultiBody" y los objetos *RigidBody* de la "Robotic Sytems Toolbox" son las estaciones multi-robots, véase figura 9. Si se importa la estación completa, el objeto *RigidBody* tiene 12 grados de libertad y sus movimientos son muy lentos y no tienen sentido físico. Normalmente, cada robot tiene su propio controlador. Por ello, la nueva versión del proyecto "SimRobots" propone compilar los robots, herramienta y carga de forma individual.

- Cada icono de robot con su herramienta y carga convencional se definen en un subsistema de Simulink.
- Dichos subsistemas son importados a un objeto *RigidBody*, que se guarda en un fichero **.mat*. En ese fichero se tiene el objeto *RigidBody*, la base del robot, la herramienta y carga (objetos *Pieza()*). El objeto *Cin()* puede modificar estos elementos, por ejemplo, la base del robot, con el método *Hbody()*.
- Se creará un objeto *Cin()* por robot, con la información de sus respectivos ficheros **.mat* y el fichero "SimScape Multibody" de la estación. Cada objeto (robot) sabe las entradas de Simulink que le pertenecen.
- La simulación de cada robot normalmente será secuencial, aunque también se pueden simular simultáneamente.

Las ventajas obtenidas son muy grandes. Se puede cambiar de robot de forma inmediata usando iconos de referencia a subsistemas y se pueden formular varios robots independientes en la misma estación. Por otro lado, el tiempo de computación es mucho menor.

En la figura 9, se definen dos objetos *Cin()*, uno para cada robot. La base de los robots se mueve de su posición inicial. Se obtiene la posición del TCP del primer robot, y se mueve el segundo robot a esa posición girada 180º en el eje x.

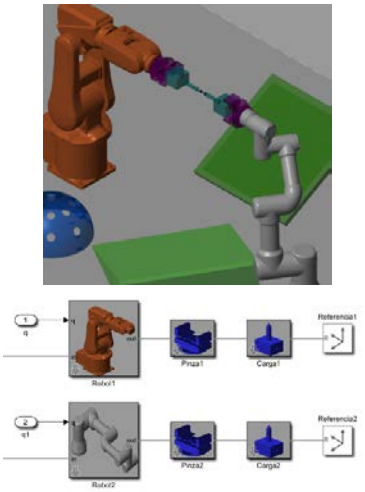

Figura 9: Estación Multi-Robots

3.5. *Humanoides*

Un humanoide puede ser considerado como un multi-robots con brazos y piernas. La librería "SimScape MultiBody" tiene iconos que simulan la fuerza de choque entre dos elementos. En la figura 10 se muestra un humanoide en equilibrio sobre sus pies que hacen fuerza en el suelo. En (Sánchez C., 2022) se programó los movimientos del humanoide para dar pasos, aunque mantener el equilibrio es complejo.

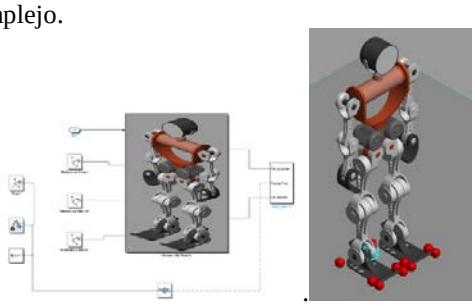

Figura 10: Humanoide sujeto por los pies (bolas) al suelo

3.5. *Robótica Médica*

Otro campo en el que se pueden desarrollar aplicaciones es en la robótica médica. Los diferentes componentes del robot Da Vinci están modelos en formato URDF (Fontanelli 2017). En la figura 11 se ha modelado el brazo PSM del robot y se puede simular su movimiento de rotación y traslación de su TCP. El mayor problema de este modelo es lo pequeño que son las herramientas en relación al robot. Es preciso usar dos cámaras para poder ver la simulación, una general y otra apuntando a la herramienta.

De la misma forma, muchos exoesqueletos han sido modelados en URDF y sus movimientos podría ser simulados con este proyecto.

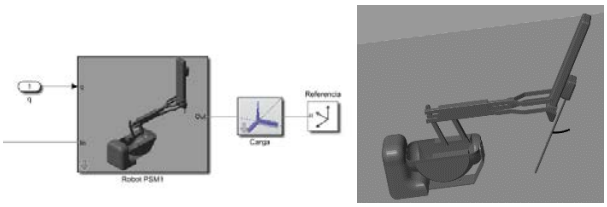

Figura 11: Simulación del PSM del robot DaVinci

4. Otras posibles librerías de Matlab que se pueden usar en el proyecto

La librería básica de Matlab en el proyecto es "Robotic Systems Toolbox". Sin embargo, se han usado otras librerías de Matlab para realizar diferentes tareas.

4.1. Comunicaciones

Muchos de los robots industriales disponen de comunicaciones TCP/IP, OPC, ROS y más (MathWorks, 2024d). En (Beneyto, A. 2023) se diseñó una App para mover un robot Niryo, programado en ROS, de forma simultánea a su simulación en Simulink.

4.2. Visión

El simulador de "SimScape Multibody", "Mechanics Explorers", puede grabar la simulación desde distintos puntos de referencia. Se podría incorporar a los robots cámaras en su TCP para poder grabar imágenes que luego fueran tratadas por las librerías de visión de Matlab (MathWorks, 2024e). Tras identificar elementos, el robot podría tomarlos.

5. Estructura de ficheros del proyecto "SimRobots"

El proyecto "SimRobots" cuelga de una carpeta con un fichero "SimRobots.prj" para definir los *path* del proyecto. Los directorios de que se componen son básicamente:

- "05_Teoría": Ficheros para definir la base teórica del proyecto, tanto respecto a "SimScape MutliBody" como "Robotic Systems Toolbox".
- "10_Cinemática": Ficheros para simular plantas cinemáticas con uno o varios robots.
- "15_Dinámica": Ficheros para simular plantas dinámicas con robots colaborativos.
- Otros directorios para definir diferentes estaciones, por ejemplo, médicas, visión, …
- "Mfile": Directorio donde están definidos los objetos básicos de proyecto.
- "Robots": Directorio donde están definidos los elementos de los robots
- "Biblioteca": Directorio donde se definen los elementos estáticos de las estaciones.
- "RobMat": Directorio donde están definidos los ficheros **.mat* de los robots (objetos *RigidBody* y más).
- "RobSubSys": Directorio donde están definidos los ficheros subsistemas de los robots.
- "Documentación": Directorio con los artículos, póster y manuales del proyecto.

Se está trabajando en una versión "final" para colgarla en una plataforma pública. El problema es que, en cada curso, se modifica el proyecto con la experiencia del mismo.

6. Conclusiones

El proyecto "SimRobots" trata de obtener un software donde colaboren las librerías "SimScape MultiBody" y "Robotic Systems Toolbox". El objetivo es obtener una herramienta fácil de programar y que dé al alumno una visión general sobre la programación robótica. Esta es la segunda versión del proyecto. En los cuatro años desde la primera, esta herramienta ha sido usada en parte de las asignaturas de diferentes grados de Ingeniería Industrial con bastante éxito. En cursos de no muchas horas (20 horas de media), los alumnos han sido capaces de desarrollar nuevas estaciones robóticas y moverlas.

Las prácticas que se proponen al alumno son muy libres. Por ejemplo, crea una estación con varios robots y objetos de trabajo que realicen diferentes tareas, dibujar tu nombre en tabla, jugar al ajedrez, y más. No puede haber dos estaciones iguales. El grado de satisfacción de los alumnos ha sido alto, ya que programar un robot es algo que gusta a la mayoría de los alumnos, sean del grado que sean.

Referencias

(Arévalo S. 2021). Arévalo Fernández, S., Herreros López, A. Aplicaciones de Matlab y Simulink para estaciones robóticas. En XLII Jornadas de Automática: libro de actas. Castelló, 1-3 de septiembre de 2021 (pp.625-631). http://hdl.handle.net/2183/28347

(Barrientos, A 2007). Barrientos, A., Peñin L. F., Balaguer C. y Santoja R. A., Fundamentos de Robótica (2º edición), MacGraw-Hill, 2007.

(Beneyto, A. 2023). Beneyto, A., Herreros, A. 2023. Control de un robot Niryo desde Matlab y Simulink. XLIV Jornadas de Automática, 523-527. https://doi.org/10.17979/spudc.9788497498609.523

(Corke, P. 2016). Corke, P. Robotics, Vision and Control. Fundamental Algorithms in MATLAB, DOI: 10.1007/978-3-319-54413-7

(Fontanelli 2017). Fontanelli, G. A., Ficuciello, F.,Villani L. and Siciliano B., "Modelling and identification of the da Vinci Research Kit robotic arms," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 2017, pp. 1464-1469, doi: 10.1109/IROS.2017.8205948.

(Gil A. 2015). Arturo Gil, Oscar Reinoso, Jose Maria Marin, Luis Paya and Javier Ruiz. Development and deployment of a new robotics toolbox for education. Computer Applications in Engineering Education (May, 2015) - Vol. 23(3), pp. 443-454 Ed. Wiley ISSN:1099-0542 1061-3773

(Gil A. 2024) Arturo Gil. Python toolbox focussed on robotic manipulators pyARTE (https://github.com/4rtur1t0/pyARTE) accessed: 2024-05-30,

(MathWorks, 2024a). Simscape™ Multibody™ User's Guide

(MathWorks, 2024b) Robotics System Toolbox™ User Guide

(MathWorks, 2024d) OPC Toolbox™ User's Guide

(MathWorks, 2024e) Computer Vision Toolbox™ User's Guide

(MathWorks, 2024f) Simscape™ User's Guide

(MathWorks, 2024g) MATLAB® App Building

(MathWorks, 2024h). ROS Toolbox User's Guide

(Mato, A 2014). Mato, A., Simulación, control cinemático y dinámico de robots comerciales usando la herramienta de Matlab, "Robotic Toolbox", TFG Univ. Valladolid, 2014.

(Pérez, M 2014) Pérez, M., Valdemar, E., Zaldívar D. Fundamentos de robótica y mecatrónica con Matlab y Simulink, Editorial RA-MA ISBN: 978-84-9964-451-6.

(Sánchez C. 2022). Sánchez-Girón Coca, C., Simulación Dinámica de Robots usando Simscape (Simulink). Aplicación en el Movimiento de Robots Humanoides y Cuadrúpedos, TFG-Univ. Valladolid, 2022.