Diseño de Librería en Matlab con Enfoque Didáctico para el Robot Móvil Diferencial Khepera IV.

Francesco José García Luna
Departamento de Industrial y Manufactura
Universidad Autónoma de Ciudad Juárez
Ciudad Juárez, Chihuahua
Email: francesco.garcia@uacj.mx

David Luviano Cruz
Departamento de Industrial y Manufactura
Universidad Autónoma de Ciudad Juárez
Ciudad Juárez, Chihuahua
Email: david.luviano@uacj.mx

Luis Ricardo Vidal Portilla
Departamento de Industrial y Manufactura
Universidad Autónoma de Ciudad Juárez
Ciudad Juárez, Chihuahua
Email: lvidal@uacj.mx

***Resumen*—En el presente articulo se muestran las especificaciones técnicas del robot Khepera en su cuarta iteración (Ver sección II). En la sección III se muestra el modelo odométrico utilizado en las funciones de control diseñadas del robot. Dichas funciones se muestran con detalle en la sección IV, en donde se especifica los argumentos de entrada de la función, así como la salida esperada. Por último en la sección V se muestran resultados experimentales de adquisición de datos de los sensores infrarrojos y ultrasónicos.**

I. Introducción

En el sector educativo de nivel licenciatura y superior, sobre todo en carreras de ingeniería afines a Mecatrónica es común encontrarse con distintos tipos de plataformas robóticas. Dichas plataformas son ampliamente utilizadas en prácticas de clases de nivel avanzado, en proyectos de titulación, o inclusive por los mismos docentes-investigadores para sus propios productos.

Las plataformas robóticas muy comúnmente cuentan con sus propias instrucciones para comunicación y programación, sin embargo, estas instrucciones pueden en ocasiones ser complejas o complicadas, o no estar en el lenguaje de programación deseado. Por esta razón, diversas personas, entendiendo el funcionamiento básico de redes de comunicación y la naturaleza de los sensores y actuadores involucrados, tienden a diseñar y programar sus propias subrutinas para comunicarse o controlar el robot en el lenguaje en el que más sean hábiles.

Tal es el caso de [1], quienes crearon un toolbox para Matlab llamado SpaceDyn, enfocado en soluciones numéricas para los campos de robótica e ingeniería espacial. Dicha librería contenía funciones que permiten el estudio y simulación de la cinemática y dinámica de sistemas multi-cuerpos articulados con base móvil con o sin gravedad.

Por otro lado [2], comentan que existen herramientas para el modelado y control de sistemas robóticos tipo serie y paralelos, pero no para sistemas híbridos, por lo que desarrollaron un toolbox para este tipo de sistemas. Otro ejemplo de equipos que se han enfocado en desarrollar herramientas para casos particulares de robots personalizados [3].

Se han diseñado herramientas para aplicaciones de competencia [4], y para robots industriales [5] y [6], las cuales simplificaban operaciones como configuración, control y administración de robots.

Entre otros tipos de aplicaciones, se han diseñado plataformas de experimentación completas con librerías propias para ciertos tipos de experimentos con robots móviles [7].

Existen otro tipo de librerías cuyo objetivo es disminuir el tiempo de la etapa de modelación cinemática y dinámica de los robots. Por lo que [8] diseñaron una herramienta que dados los parámetros de Denavit-Hartenberg y utilizado Euler-Lagrange son capaces de generar las ecuaciones requeridas para controlar un robot.

En particular, en Matlab, la librería más conocida para aplicaciones de robótica es la llamada Robotics Toolbox", desarrollada por Peter Corke [9]. Esta ha ido aumentando sus funciones a través de los años.

El objetivo de este articulo es presentar una alternativa para el control y la adquisición de datos de los sensores del robot móvil diferencial Khepera IV en un lenguaje de alto nivel.

El artículo está dividido en 5 secciones. En la sección I se plantea el contexto de las librerías con enfoque didáctico para plataformas robóticas. En la sección II se muestran las especificaciones del robot utilizado, así como algunas de sus características principales. El modelo odométrico para la estimación de la pose del robot en función del desplazamiento de cada llanta se muestra en la sección III. La contribución

ISBN: 978-607-97128-7-7

principal se muestra en la sección IV en donde se especifica el nombre de la función, sus argumentos de entrada, de salida y una breve descripción. Los resultados de los experimentos se muestran en la sección V y por último, las conclusiones en la sección VI.

II. Especificaciones

Desarrollado por K-Team, el Khepera IV (mostrado en la fig. 1) ha pasado por cuatro iteraciones (descritas en el cuadro I). Siendo la cuarta iteración la versión 3.0. Este robot es del tipo móvil diferencial con restricciones de no-holonomía. Cuenta con dos ruedas actuadas y una rueda caster (no actuada).

Figura 1. Khepera IV desarrollado por K-Team

Cuadro I
Historial de Revisiones [10]

Revisión	Fecha	Descripción
1.0	14/03/14	Creación Inicial
2.0	3/02/15	Adaptado para el Gumstix Overo FireSTORM-P
2.1	15/02/15	Micrófono mejorado
3.0	7/08/16	Adaptado para el Gumstix Overo FireSTORM-Y

Entre sus características se encuentran:

- Switch de encendido/apagado.
- LED de estatus.
- Conector MinUSB-B.
- Conector USB-A (500 mAh).
- Conector para cable de alimentación (9V, 1.5A, 0.65mm positivo al centro).
- Boton de reinicio.
- 12 sensores infrarrojos (4 por debajo y 8 en los costados).
- 5 sensores ultrasónicos.
- Cámara RGB.
- 3 LED RGB.
- Bluetooth y WiFi.
- IMU de 6 ejes (acelerómetro y giroscopio).

Se puede establecer la comunicación con el robot mediante tres diferentes protocolos, teniendo en cuenta las diferentes ventajas y desventajas del uso de cada uno:

- USB serial (115,200 Baudios, sin paridad y sin control de flujo).
- Bluetooth.
- WiFi.

II-A. Sensores

Los sensores infrarrojos colocados alrededor del robot, le permiten detectar obstáculos que se encuentren entre 2mm y 250mm, dependiendo de la calibración, las condiciones del ambiente y el color del obstáculo. También pueden ser utilizados para medir la luz ambiental. Estos sensores se encuentran con una separación de 45° entre ellos (mostrado en la fig 2).

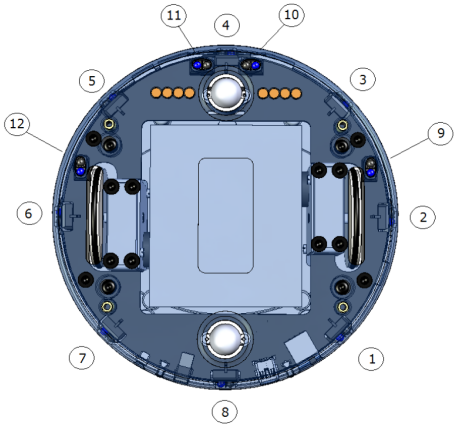

Figura 2. Localización de los sensores infrarrojos desde la vista inferior, del 1 al 8 se encuentran alrededor del robot y del 9 al 12 se encuentran por debajo.

Los sensores ultrasónicos están colocados, dos a cada lado del robot y uno al frente (mostrados en la fig. 3). Estos sensores le permiten al robot detectar obstáculos que se encuentren entre 25cm y 250cm. Al igual que los sensores infrarrojos, estos se encuentran con una separación de 45° entre ellos.

Cabe destacar que debido al principio de operación de los sensores ultrasónicos, no es posible detectar objetos a menos de 25cm. Por lo que es buena idea utilizar una técnica conocida como fusión sensorial para hacer redundante las mediciones con distintos tipos de sensores (en este caso, infrarrojos + ultrasónicos).

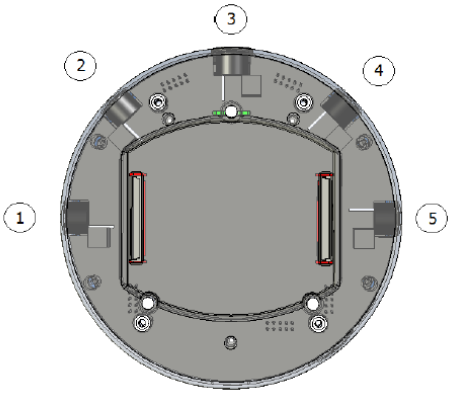

Figura 3. Localización de los sensores ultrasónicos desde la vista superior.

ISBN: 978-607-97128-7-7

Cuenta una cámara RGB al frente capaz de tomar fotos o videos con un tamaño máximo de 752x480 pixeles y un procesador ARM Cortex-A8 a 800 MHz integrado capaz de procesar las imágenes. Dicha cámara tiene una distancia focal de 2.1mm. Y su campo de visión es de 131° de forma horizontal y 101° de forma vertical.

La IMU de 6 ejes es una LSM330DLC que incluye un acelerómetro y un giroscopio. Se encuentra justo sobre el eje de giro en el baricentro del robot, sobre la tarjeta electrónica.

En cuanto al acelerómetro, el eje X positivo se encuentra hacia adelante, el eje Z positivo se encuentra hacia arriba y por convención dextrogiro, el eje Y positivo se encuentra hacia la izquierda (mostrada en la fig. 4). Este regresa información de 12 bits con un rango de ±2g. Esto significa que cuando mide 1g, regresará un valor de 16,384. El acelerómetro esta configurado para actualizar los datos cada 10Hz (100ms).

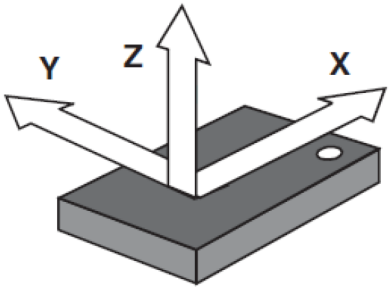

Figura 4. Localización del marco referencial del acelerómetro

Por otro lado, el giroscopio esta configurado a ±2000dps con una frecuencia de 95Hz. Se utiliza la ec. 1 para convertir la información cruda en grados/segundo. El eje X gira en el plano YZ, el eje Y gira en el plano XZ, y por último, el eje Z gira en el plano XY (mostrado en la fig. 5).

$$\frac{deg}{sec} = data_{raw} * 0{,}66 \tag{1}$$

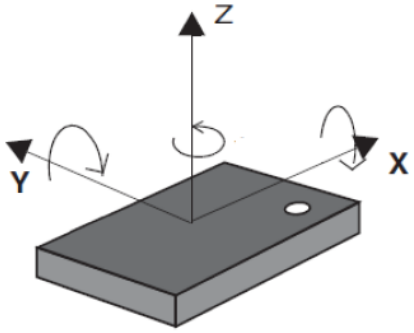

Figura 5. Localización de los ejes de giro detectables desde la vista superior

Además, cada rueda de 42mm de diámetro cuenta con encoder de 19,456 pulsos por vuelta. Y tiene una circunferencia de 131.94mm, lo que significa que genera 1 pulso cada 0.006782mm (6.7818μm).

III. Modelo odométrico de robots móviles diferenciales

En los robots diferenciales es normal utilizar odometría para el cálculo del posicionamiento, siempre considerando que no existe derrape. En este tipo de robots, el movimiento se consigue con dos ruedas actuadas y una o mas ruedas de apoyo (mostrado en la fig. 6).

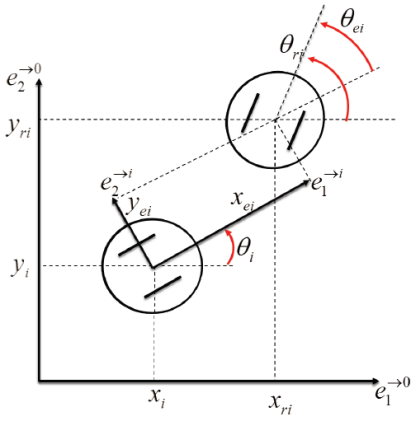

Figura 6. Esquema del modelo de un robot diferencial

El movimiento se genera considerando combinando la velocidad y dirección de cada una de las ruedas (mostrado en la fig. 7). Si se desea ir hacia adelante (considerando que los motores no están posicionados de forma encontrada) ambos motores deben de ir en la misma dirección; si se desea girar hacia algún lado, ambas llantas deben de tener diferente velocidad.

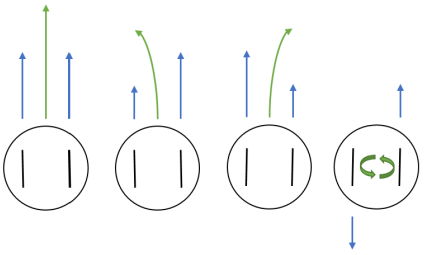

Figura 7. Movimientos posibles de un robot diferencial en función de la dirección y velocidad de cada rueda.

El desplazamiento del robot según [11] esta dado por la ec. 2:

$$D = \frac{D_1 + D_2}{2} \tag{2}$$

Donde D es el desplazamiento del centroide del robot, D_1 y D_2 es el desplazamiento individual de cada llanta.

Mientras que la orientación del robot esta descrita por la ecuación 3:

$$\theta = \frac{D_1 - D_2}{d} \tag{3}$$

Donde d es la distancia entre llantas.

Se puede determinar el desplazamiento lineal de la llanta si se considera el radio efectivo y el encoder (descrito en la ec. 4).

$$D_i = \frac{2\pi N_i}{C_i} R_{ei} \tag{4}$$

Donde:

- R_{ei}: radio efectivo de la llanta i (considerando compliancia).
- R_{ni}: radio nominal de la llanta i.
- N_i: Cantidad de pasos recorridos del encoder.
- C_i: Cantidad de pasos totales del encoder por revolución de la llanta.
- $R_{ei} = R_{ni}$ en el caso particular del Khepera IV.

Mientras que para la dirección la ec. 3 se puede expresar en función del encoder como:

$$\theta = \frac{2\pi R_e}{Cd}(N_1 - N_2) \tag{5}$$

Por último, utilizando la fig. 6 y las ecuaciones 4 y 5 se puede estimar la posición del robot como:

$$\begin{aligned} x &= Dcos(\theta) \\ y &= Dsin(\theta) \end{aligned} \tag{6}$$

IV. Contribución Principal

Considerando [10], los sensores del robot móvil diferencial y la naturaleza de los datos, se programó una serie de funciones en Matlab para controlar y obtener datos del robot. El toolbox se presenta como una alternativa para controlar al robot en modo esclavo.

Para lograr esto, primeramente hay que configurar el robot para que acepte comandos por puerto serial. Se recomienda utilizar un sistema operativo basado en UNIX para facilitar la conexión por bluetooth-serial. En el caso particular se utilizó el paquete Minicom para establecer la conexión bluetooth-serial con el robot.

El protocolo de comunicación del robot funciona del modo comando-respuesta. Todos los comandos son enviados en código ASCII. En caso de que se requiera hacer la conexión en otro sistema operativo, se tiene que cambiar la dirección serial a la correspondiente.

El toolbox actualmente cuenta en su versión 1.0 con 17 funciones:

1. data = khepera_ALValues(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Argumento(s) de salida:
     - arreglo de tamaño 12x1 (data)
   - Notas
     - Entrega el valor de la incidencia de rayos infrarrojos
     - $0 \rightarrow 1024$ (IR saturado - IR carente)
2. s = khepera_connect:
   - Argumento(s) de salida:
     - objeto serial (s)
   - Notas
     - Conecta el robot en modo esclavo en la dirección serial /dev/rfcommm0 a 115,200 Baudios
3. khepera_disconnect(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Notas
     - Detiene las ruedas del robot y se desconecta del puerto serial
4. data = khepera_encodersValues(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Argumento(s) de salida:
     - arreglo de tamaño 2x1 (data)
   - Notas
     - posición absoluta del encoder de cada llanta
5. data = khepera_getBatteryStatus(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Argumento(s) de salida:
     - celda con 7 parámetros de solo lectura (data)
   - Notas
     - Contiene información sobre el voltaje, corriente, corriente promedio, capacidad absoluta disponible (mAh), capacidad absoluta relativa (%), Temperatura y estatus del conector AC (conectado o no conectado)
6. khepera_getFirmware(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Notas
     - Da información en la ventana de comandos sobre la versión actual del firmware del robot
7. data = khepera_IRValues(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Argumento(s) de salida:
     - arreglo de tamaño 12x1 (data)
   - Notas
     - $0 \rightarrow 1024$ (Obstáculo alejado - Obstáculo cerca), lectura comenzando desde el sensor ubicado atrás a la izquierda y ordenado en sentido horario. Los úlitmos 4 valores corresponden a los sensores ubicados en la parte inferior de izquierda a derecha.
8. [raw_data, mean_data] = khepera_readAccelalues(s):
   - Argumento(s) de entrada:
     - objeto serial (s)

- Argumento(s) de salida:
  - arreglo de 10x3 (raw_data)
  - arreglo de 1x3 (mean_data)
- Notas
  - raw_data contiene 10 datos del historial de aceleración inercial sin procesar de los tres ejes cartesianos $[x, y, z]$
  - mean_data contiene el promedio de 10 mediciones de aceleración inercial de cada eje.

9. [raw_data, mean_data] = khepera_readGyroValues(s):
   - Argumento(s) de entrada:
     - objeto serial (s)
   - Argumento(s) de salida:
     - arreglo de 10x3 (raw_data)
     - arreglo de 1x3 (mean_data)
   - Notas
     - raw_data contiene 10 datos del historial del giroscopio sin procesar sobre los tres ejes cartesianos $[x, y, z]$
     - mean_data contiene el promedio de 10 mediciones del giroscopio sobre cada eje.
10. data = khepera_readSpeed(s):
    - Argumento(s) de entrada:
      - objeto serial (s)
    - Argumento(s) de salida:
      - arreglo de 2x1
    - Notas
      - Contiene información sobre la velocidad lineal (m/s) de cada rueda
11. khepera_resetEncoders(s):
    - Argumento(s) de entrada:
      - objeto serial (s)
    - Notas
      - Contiene información sobre la velocidad lineal (m/s) de cada rueda
12. khepera_RGB(s, led_1, led_2, led_3):
    - Argumento(s) de entrada:
      - objeto serial (s)
      - arreglo de 1x3 (led_i)
    - Notas
      - Los valores de saturación de cada canal están en el rango de 0-255
13. khepera_setGoal(s, motorTravel):
    - Argumento(s) de entrada:
      - objeto serial (s)
      - arreglo de 1x2 (motorTravel)
    - Notas
      - Ocasiona que el robot desplace la rueda(i) motorTrave(i) mm lineales
14. khepera_setSpeed(s, lspeed, rspeed):
    - Argumento(s) de entrada:
      - objeto serial (s)
      - escalar (lspeed y rspeed)
    - Notas
      - Ocasiona que la rueda(i) alcance la velocidad indicada a lazo cerrado utilizando el esquema de control PID interno
15. khepera_setSpeedOpens, ldir, lspeed, rdir, rspeed):
    - Argumento(s) de entrada:
      - objeto serial (s)
      - escalar (lspeed y rspeed)
      - booleano (ldir y rdir)
    - Notas
      - Manda una señal PWM (0-255) y dirección a cada rueda a lazo abierto
      - dir = verdadero: hacia adelante
      - dir = falso: hacia atrás
16. serial_out = khepera_talk(s, data):
    - Argumento(s) de entrada:
      - objeto serial (s)
      - cadena de caracteres (data)
    - Argumento(s) de salida:
      - cadena de caracteres (serial_out)
    - Notas
      - Es la base de todas las funciones, se envía en código ASCII el comando y se registra la respuesta
17. data = khepera_USValues(s):
    - Argumento(s) de entrada:
      - objeto serial (s)
    - Argumento(s) de salida:
      - arreglo de 5x1 (data)
    - Notas
      - Información acerca de los valores de los sensores ultrasónicos (en cm) de izquierda a derecha
      - un valor de 1000 significa que no se registra un obstáculo
      - un valor de 0 significa que existe un obstáculo a menos de 25cm
      - un valor entre 25 y 250 significa la distancia entre el sensor y el obstáculo en cm

V. Resultados

La librería se utiliza actualmente en distintas clases de la carrera de Ingeniería Mecatrónica en la Universidad Autónoma de Ciudad Juárez, por ejemplo:

- Diseño Mecatrónico.
- Robótica.
- Control I y II.
- Seminario de Tesis y Trabajo de Tesis.

Se diseñaron dos experimentos:

1. Estimación de la distancia de obstáculos mediante la adquisición de datos ultrasónicos.

2. Adquisición de datos de los sensores infrarrojos y ultrasónicos para realizar fusión sensorial.

Los resultados del experimento 1 mostrados en la fig. 8, en donde se puede observar la distancia a los obstáculos en centímetros desde el robot cada 0.001 segundos (mostrados en la ec. 7).

Se utilizó la función $data = khepera\_USValues(s)$, en donde:

- $data$: arreglo de tamaño 5x1 con información en cm de los sensores ultrasónicos.
- s: objeto serial que apunta a la dirección en la que está conectado el Khepera.

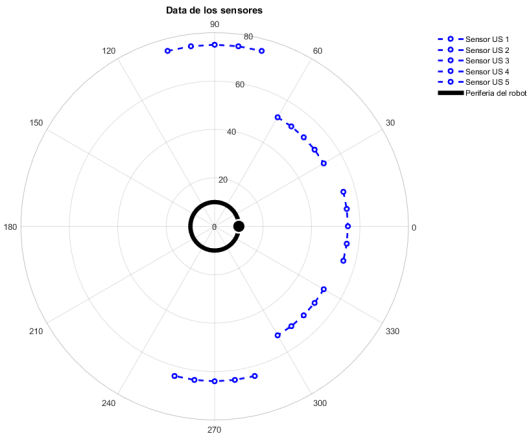

Figura 8. Experimento de adquisición de datos de los sensores ultrasónicos en cm.

$$\begin{bmatrix} \theta_1 & d_{u1} \\ \theta_2 & d_{u2} \\ \theta_3 & d_{u3} \\ \theta_4 & d_{u4} \\ \theta_5 & d_{u5} \end{bmatrix} = \begin{bmatrix} \pi/2 & 75 \\ \pi/4 & 52 \\ 0 & 55 \\ -\pi/4 & 52 \\ -\pi/2 & 64 \end{bmatrix} \tag{7}$$

Donde d_{ui} es la distancia en centímetros hacia un obstáculo desde el robot con un ángulo θ_i radianes en el plano XY.

En la fig. 9 se puede observar el resultado del experimento 2 de fusión sensorial en donde se obtiene en tiempo real información de los sensores infrarrojos al mismo tiempo que información de los sensores ultrasónicos.

Se utilizó la función $data = khepera\_IRValues(s)$, en donde:

- $data$: arreglo de tamaño 8x1 con información en cm de los sensores infrarrojos.
- s: objeto serial que apunta a la dirección en la que está conectado el Khepera.

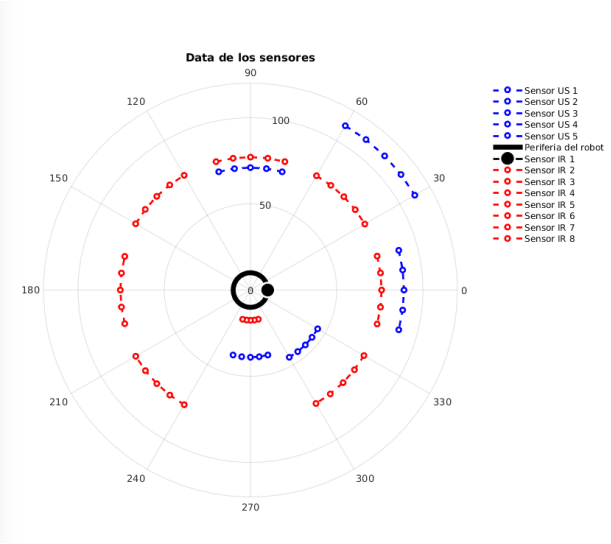

Figura 9. Experimento de fusión sensorial de datos de los sensores ultrasónicos e infrarrojos en cm.

Los dos experimentos fueron realizados en distintas situaciones, por lo que es de esperarse que ambas gráficas tengan información diferente.

VI. Conclusión

Debido a la naturaleza de los sensores infrarrojos, es necesario que los experimentos para la detección de obstáculos se realice bajo condiciones de luz artificial únicamente.

Sin embargo, la utilización de este toolbox permite al usuario enfocarse en los resultados de experimentación y no en la parte de programación del robot. Ya que esta última suele ser la de mayor duración.

En la versión actual del toolbox, puede ser utilizado para aplicaciones de SLAM, regulación punto a punto, seguimiento de trayectorias, detección de inclinación en el plano de navegación, detección de incidencia infrarroja, entre otras.

En siguientes versiones del toolbox, se planea incluir funciones para graficar los datos de los sensores y algoritmos de control y seguimiento de trayectorias a lazo cerrado de uno o múltiples agentes.

Reconocimiento

La librería puede adquirirse enviando un correo a la dirección de correo electrónico francesco.garcia@uacj.mx o david.luviano@uacj.mx.

Los autores extienden su agradecimiento a la Secretaría de Educación Pública de México por financiar este trabajo a través del Acuerdo de Investigación 511-6 / 17-7605.

Referencias

[1] K. Yoshida, “The spacedyn: A matlab toolbox for space and mobile robots,” 1999.
[2] H. Ozakyol, C. Karaman, and A. Bingül, “Robotics toolbox for kinematic analysis and design of hybrid multibody systems,” 2017.
[3] J. Nicho and B. Nowak, “A versatile toolbox for studying anthropomorphic robot hands,” 2009.

[4] J. Jang, C. K. Ahn, S. Han, and W. H. Kwon, “Rapid control prototyping for robot soccer system using simtool,” 2006.
[5] F. Chinello, S. Scheggi, F. Mordibi, and D. Prattichizzo, “Kct: A matlab toolbox for motion control of kuka robot manipulators,” 2010.
[6] E. Engels, “A versatile matlab toolbox for rapid-robot-prototyping of custom made industrial robots,” 2016.
[7] M. Kloetzer, S. Magdici, and A. Burlacu, “Experimental platform and matlab toolbox for planning mobile robots,” 2012.
[8] E. Dean-Leon, S. Nair, and A. Knoll, “User friendly matlab-toolbox for symbolic robot dynamic modeling used for control design,” 2012.
[9] P. Corke, *Robot Manipulator Capability in Matlab*. Mathworks, 2017.
[10] K-Team, *Khepera IV User Manual*. K-Team, 2016.
[11] M. A. G. Rodriguez. (2011, jan) Modelo odométrico diferencial de robots móviles. [Online]. Available: http://www.tamps.cinvestav.mx/~mgomez/