



Taller 1 - Robótica y Control Servo-Visual

Taller Modelo Cinemático

Alejandro Ojeda Olarte - aojedao@unal.edu.co
Camilo Ernesto Campo Pacheco - cecampop@unal.edu.co
Juan Camilo Torres Mestra - juctorresme@unal.edu.co

Contenido

1	Componente Teórico	2
1.1	Descripción Cinemática del Robot Kobuki	2
1.2	Datos Recolectados	3
1.3	Construcción del modelo cinemático Kobuki V3	3
1.4	Construcción del modelo cinemático Robotino	4
2	Componente Práctico	5
2.1	Construcción del modelo básico	5
	Referencias	

Índice de figuras

1	Kobuki fabricado por iClebo [1]	3
2	Modelo CAD del diseño propio Kobuki V3	3
3	Robotino en su parte exterior. [2]	4
4	Robotino en su parte interior. [3]	4
5	Visualización en Gazebo.	5
6	Visualización del robot en Rviz.	6
7	Ventanas de comandos previos.	6
8	Control del robot al aparecer en Gazebo.	6
9	Visualización de ROS Graph.	7
10	Control del robot por teclado.	7
11	Simulación completa.	7



1. Componente Teórico

1.1. Descripción Cinemática del Robot Kobuki

El robot Kobuki fabricado por iClebo es una base móvil para investigación. A continuación algunas características del robot tomadas de su página web [1].

▪ Functional Specification:

- Maximum translational velocity: 70 cm/s
- Maximum rotational velocity: 180 deg/s (± 110 deg/s gyro performance will degrade)
- Payload: 5 kg (hard floor), 4 kg (carpet)
- Cliff: will not drive off a cliff with a depth greater than 5cm
- Threshold Climbing: climbs thresholds of 12 mm or lower
- Rug Climbing: climbs rugs of 12 mm or lower
- Expected Operating Time: 3/7 hours (small/large battery)
- Expected Charging Time: 1.5/2.6 hours (small/large battery)
- Docking: within a 2mx5m area in front of the docking station

▪ Hardware Specification:

- PC Connection: USB or via RX/TX pins on the parallel port
- Motor Overload Detection: disables power on detecting high current ($\geq 3A$)
- Odometry: 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm

- Gyro: factory calibrated, 1 axis (110 deg/s)

- Bumpers: left, center, right

- Cliff sensors: left, center, right

- Wheel drop sensor: left, right

- Power connectors: 5V/1A, 12V/1.5A, 12V/5A

- Expansion pins: 3.3V/1A, 5V/1A, 4 x analog in, 4 x digital in, 4 x digital out

- Audio : several programmable beep sequences

- Programmable LED: 2 x two-coloured LED

- State LED: 1 x two coloured LED [Green - high, Orange - low, Green & Blinking - charging]

- Buttons: 3 x touch buttons

- Battery: Lithium-Ion, 14.8V, 2200 mAh (4S1P - small), 4400 mAh (4S2P - large)

- Firmware upgradeable: via usb

- Sensor Data Rate: 50Hz

- Recharging Adapter: Input: 100-240V AC, 50/60Hz, 1.5A max; Output: 19V DC, 3.16A

- Netbook recharging connector (only enabled when robot is recharging): 19V/2.1A DC

- Docking IR Receiver: left, centre, right

- Diameter : 351.5mm / Height : 124.8mm / Weight : 2.35kg (4S1P - small)

▪ Software Specification:

- C++ drivers for linux and windows

- ROS node

- Gazebo Simulation



Figura 1: Kobuki fabricado por iClebo [1]

1.2. Datos Recolectados

Para la recolección de los diferentes parámetros se busco información recopilada entre los datos del proveedor, datos usados por otros casos de estudio del robot y estimaciones propias.

- Para la primer rueda:

$$\alpha_1 = \frac{\pi}{2}$$

$$\beta_1 = 0$$

$$l = 177mm$$

$$r = 35mm$$

- Para la segunda rueda:

$$\alpha_2 = \frac{-\pi}{2}$$

$$\beta_2 = \pi$$

$$l = 177mm$$

$$r = 35mm$$

1.3. Construcción del modelo cíne-mático Kobuki V3

Los anteriores datos funcionaron como modelo de referencia para la creación de un diseño propio tanto en ROS, en sus formatos SDF, URDF y Xacro; como su archivo CAD. Sus características se ven a continuación:

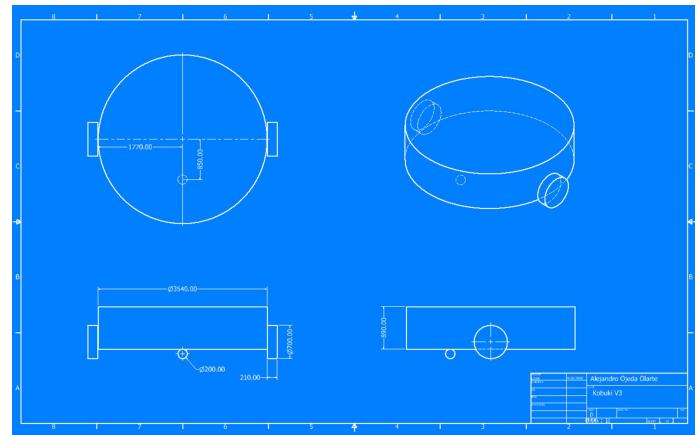


Figura 2: Modelo CAD del diseño propio Kobuki V3

El robot posee dos llantas estándar fijas motorizadas y una rueda esférica que funciona de soporte y garantiza estabilidad estática.

$$N_s = 0$$

$$N_f = 2$$

Primero se propone la matriz :

$$J_{1f} = \begin{bmatrix} \sin(\alpha_1 + \beta_1) & -\cos(\alpha_1 + \beta_1) & -l\cos(\beta_1) \\ \sin(\alpha_2 + \beta_2) & -\cos(\alpha_2 + \beta_2) & -l\cos(\beta_2) \end{bmatrix}$$

Que reemplazando se convierte en:

$$J_{1f} = \begin{bmatrix} 1 & 0 & -177 \\ 1 & 0 & 177 \end{bmatrix}$$



Así como la matriz :

$$C_{1f} = \begin{bmatrix} \cos(\alpha + \beta) & \sin(\alpha + \beta) & l\sin(\beta) \end{bmatrix}$$

Que reemplazando se convierte en la matriz:

$$C_{1f} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

Mientras tanto la matriz de radios es:

$$J_2 = \begin{bmatrix} 35 & 0 \\ 0 & 35 \end{bmatrix}$$

Gracias a esto podemos deducir que:

$$\delta_m = 2$$

$$\delta_s = 0$$

$$\delta_M = 2$$

Preguntas:

- Si el parámetro l del robot se modifica siendo $l_1 = l_2 / 2$ lo que sucede es que tendrá un movimiento circular, puesto que la velocidad angular depende del parámetro l .
- Para que el robot tenga un movimiento lineal puro, $v = 1, w = 0$, es necesario que la velocidad $\phi_1 = \phi_2$ puesto que el radio es igual para las dos llantas.
- Para que el robot tenga un movimiento rotacional puro, $v = 0, w = 1$, es necesario que la velocidad $\varphi_1 = -\varphi_2$ puesto que los demás términos son iguales.

1.4. Construcción del modelo cinemático Robotino

Robotino es un robot popular en las competencias de RoboCup. Fabricado por Festo como plataforma de desarrollo robótico, podemos observar su configuración exterior a continuación.



Figura 3: Robotino en su parte exterior. [2]

A continuación se puede observar que el robot Robotino se moviliza usando tres Ruedas Suecas (Omnidireccionales) de dos partes.

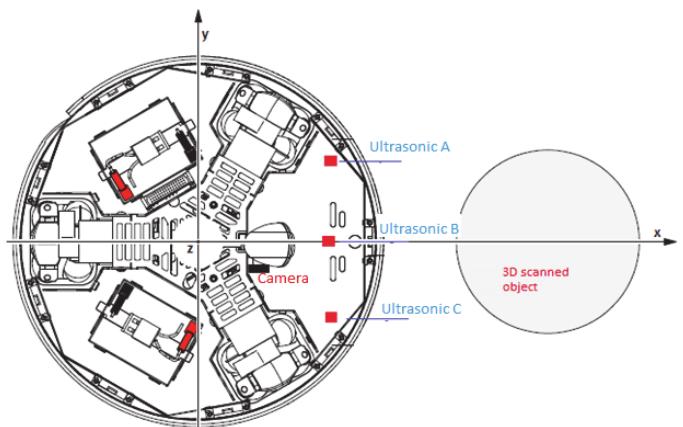


Figura 4: Robotino en su parte interior. [3]

Previo a presentar la matriz J_1 es importante resaltar que los ángulos α y β dependerán de donde se oriente el robot, si una rueda es perpendicular al



eje X,Y. En este caso se asume como en la imagen mostrada anteriormente.

$$\alpha_1 = \frac{\pi}{3}$$

$$\alpha_2 = \pi$$

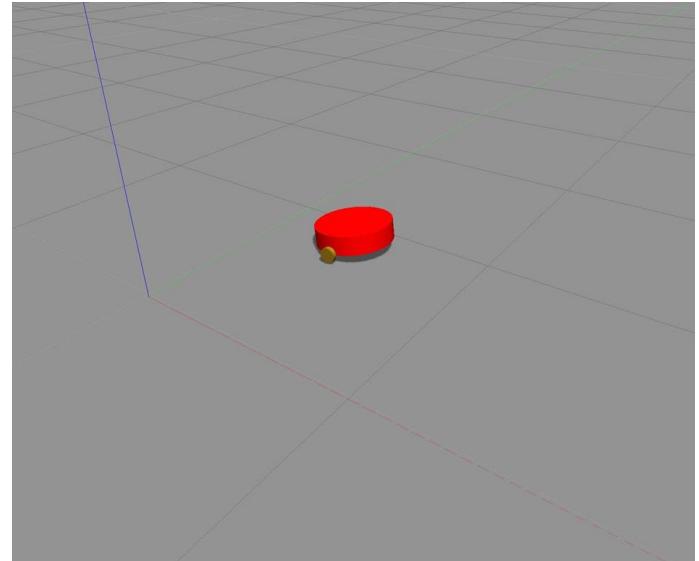
$$\alpha_3 = \frac{-\pi}{3}$$

$$\beta_1 = 0$$

$$\beta_2 = 0$$

$$\beta_3 = 0$$

$$r = 40mm \quad l = 225,3mm$$



$$J_{1f} = \begin{bmatrix} \sin(\frac{\pi}{3}) & -\cos(\frac{\pi}{3}) & -225,3 \\ 0 & -\cos(\pi) & -225,3 \\ \sin(\frac{-\pi}{3}) & -\cos(\frac{-\pi}{3}) & -225,3 \end{bmatrix}$$

2. Componente Práctico

Para la construcción del robot se pasaron por diferentes etapas en el modelo utilizado.

2.1. Construcción del modelo básico

En esta fase se desarrollaron los modelos de 4 maneras diferentes con 3 versiones diferentes.

La primera corresponde al diseño en el editor de modelos propio de gazebo, el cuál genera un modelo Simulation Description Format (SDF), construido con la herramienta visual nativa, que es similar a los CAD con enfoque en diseño artístico.

Figura 5: Visualización en Gazebo.

Posteriormente se encontró que este diseño se guarda en la base de datos del modelo de gazebo más no hace parte de un paquete propio de ROS, así que se optó por generar un modelo SDF en una carpeta destinada al paquete de ROS. Finalizada su construcción se encontró que los modelos SDF no tienen compatibilidad nativa con ROS en general, y en especial no son compatibles con el paquete ros-control el cuál es necesario para controlarlo desde comandos de ROS.

Debido a lo mencionado anteriormente se construyó un modelo en el formato Universal Robot Description Format (URDF) que tuvo como segunda versión la inclusión de parámetros de inercia, que fueron calculados usando las geometrías básicas de los componentes, despreciando la inercia de la rueda esférica y sin asignar inercia en las ruedas.

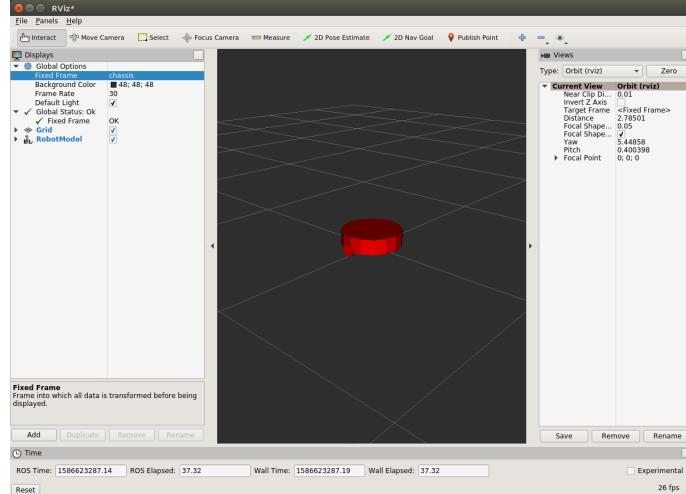


Figura 6: Visualización del robot en Rviz.

Estos dos formatos anteriores comparten la creación de links como cuerpos rígidos de las partes del robot, estos cuentan con una característica visual y una de colisión que corresponde al área que ocupa de manera aproximada. De igual manera comparten los joints, estas junturas describen la cinemática y dinámica de la juntura, así como limitaciones. Sin embargo la inclusión de nuevos parámetros los diferencia. Para esta segunda versión se incluyó la transmisión como componente del archivo URDF, esta transmisión describe la relación entre un actuador y la juntura. Esto es necesario para tener un control del robot por medio del paquete ros-control.

Para la ejecución en el simulador gazebo se realizo u archivo spawn.launch el cuál inserta el modelo en una ventana existente, en estos casos se usó el mundo empty_world. Previo a la adición de la transmisión el robot se podía mover aplicándole fuerza en la barra de herramientas izquierda del simulador, sin embargo posterior a la adición ya no se usa este método.

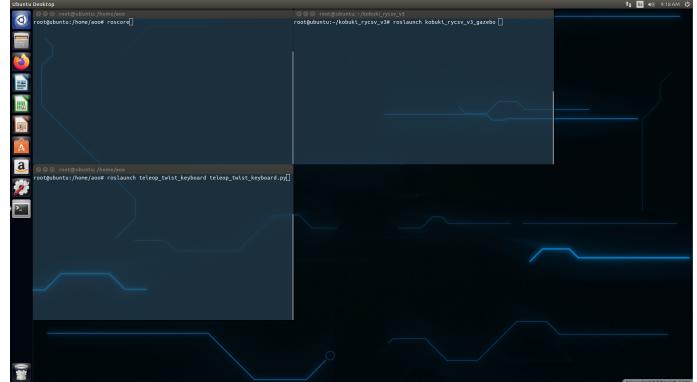


Figura 7: Ventanas de comandos previos.

El control del robot se utiliza por medio del plugin differential_drive_controller el cual ya interactúa con Gazebo y utiliza los la transformación para un robot de dos ruedas. Esto permite que el robot conecta sus actuadores al topic /cmd_vel por donde enviaremos comandos de velocidad en que eje queremos que se dirija, y el controlador ya transforma las respectivas funciones.

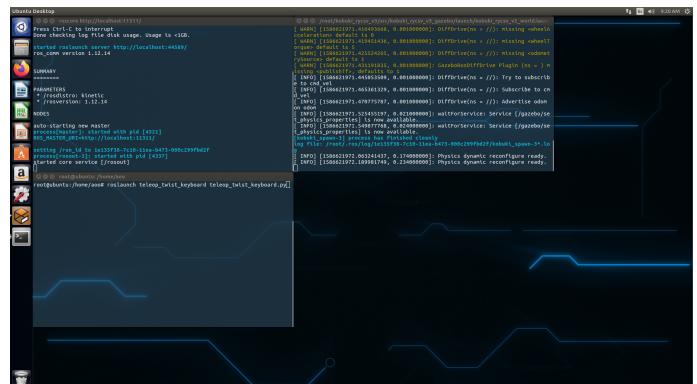


Figura 8: Control del robot al aparecer en Gazebo.

A continuación podemos observar como se comunican los nodos con el tópico.

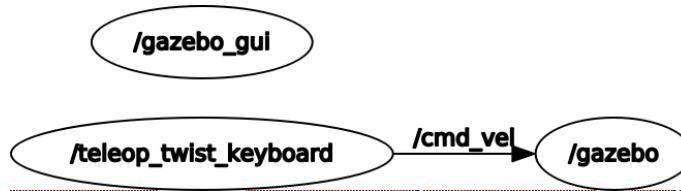


Figura 9: Visualización de ROS Graph.

Para la versión final, el Kobuki V3, se adicionaron las inercias que faltaban, materiales de Gazebo y se implementó toda la descripción del robot en un archivo Xacro, un XML Macro que permite separar las secciones como lo es Gazebo, la transmisión y el resto del URDF. Esta versión también contó con el uso del nodo teleop_twist_keyboard el cual es un nodo que recolecta información de la misma manera que el controlador de teclado de Turtlebot_sim pero este le permite publicarlo en el tópico con el mismo nombre en el que está recibiendo el robot. Su namespace es Kobuki_rycsv_V3.

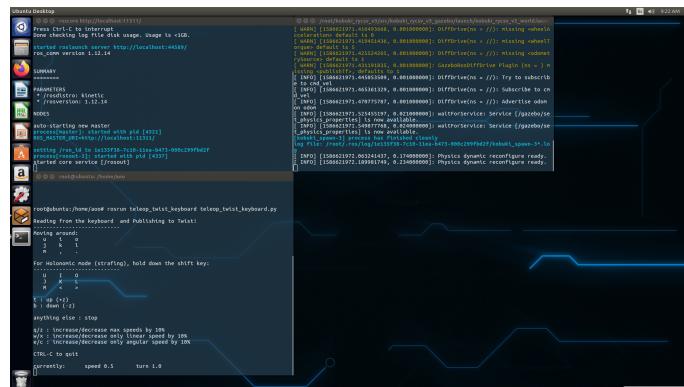


Figura 10: Control del robot por teclado.

Finalmente podemos observar que se tiene las tres ventanas de mando y el simulador. Este proceso puede hacerse en simultáneo con Rviz, y el código implementado también optimiza la ventana para correr roscore puesto que el archivo launch del paquete kobuki_rycsv_v3_gazebo ya lanza una sola vez todos estos comandos.

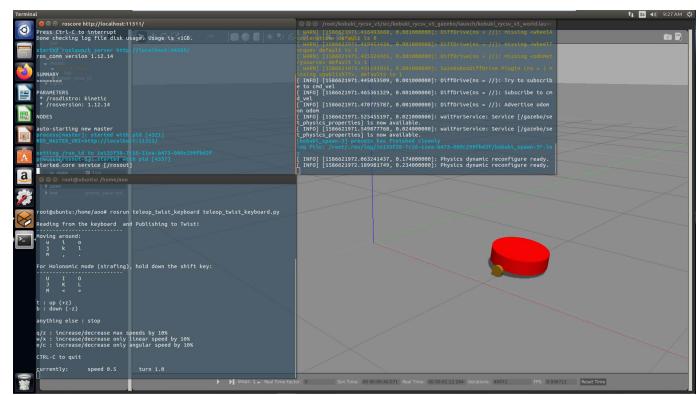


Figura 11: Simulación completa.

Toda la información encontrada aquí podrá observarla en el siguiente link:

<https://github.com/aojedao/rycsv>

Referencias

- [1] <http://kobuki.yujinrobot.com/about2/>.
- [2] <https://servicerobotics.eu/en/robotinor/>
- [3] A Simple and Inexpensive 3D Scanning of Remote Objects by Robot.