

Odometría

LUIS ALBERTO BALLADO ARADIAS

Cinvestav Unidad Tamaulipas

luis.ballado@cinvestav.mx

30 de abril de 2023

Resumen

El presente trabajo describe el uso de odometría para un robot móvil de tipo diferencial y su implementación utilizando un robot LEGO NXT bajo el lenguaje NXC (Not eXactly C). La odometría es una técnica utilizada en robótica móvil diferencial para estimar la posición y orientación de un robot mientras se mueve en un entorno. En robots móviles diferenciales, la odometría se basa en la medición de las velocidades de las ruedas y el diámetro de las mismas. La integración de las velocidades permite obtener una estimación de la trayectoria seguida por el robot, pero los errores acumulativos pueden afectar la precisión de la estimación.

I. INTRODUCCIÓN

La odometría es el estudio de la estimación de la posición de vehículos con ruedas durante la navegación. Para realizar esta estimación se usa información sobre la rotación de las ruedas para estimar cambios en la posición a lo largo del tiempo. El término se usa a veces para referirse a la distancia que ha recorrido uno de estos vehículos.

La palabra **odometría** se compone por las palabras griegas **hodos** (viajar,trayecto) y **metron** (medida). La idea fundamental es la integración de información incremental del movimiento a lo largo del tiempo.

A pesar de las limitaciones, es una parte importante del sistema de navegación de un robot debido a su bajo costo y esfuerzo de implementación, debe usarse con medidas del posicionamiento absolutas para proporcionar una estimación de la posición más fiable.

Hay casos, cuando no hay referencias externas, la odometría es la única información de navegación utilizando un modelo matemático que tiene en cuenta las características del movimiento del robot y la geometría del entorno.

I. Pasos de la odometría

La metodología para realizar odometría en robots móviles depende del tipo de robot y de los sensores utilizados, pero en general, se pueden seguir los siguientes pasos:

1. Caracterización de las ruedas: se debe medir el radio de las ruedas y distancia entre el punto de contacto entre cada rueda.
2. Adquisición de datos: se deben medir las velocidades de las ruedas y la dirección de giro del robot. Esto se puede realizar mediante encoders en las ruedas o mediante sensores iniciales.
3. Integración de las velocidades: se deben integrar las velocidades medidas para calcular la posición y orientación del robot.

-
- 4. Corrección de errores: se deben corregir los errores que pueden afectar la precisión de la odometría, como el deslizamiento de las ruedas, la fricción con el suelo o la deformación de los neumáticos.
 - 5. Evaluación de la precisión: se deben evaluar la precisión y el error de la odometría. Para ellos, se pueden comparar las estimaciones obtenidas con la posición real del robot o con la información obtenida por otros sensores.

II. MÉTODOS

I. Instalación

Comenzaremos con la instalación de NXC (Not Exactly C)

La configuración del compilador se realiza para un ambiente **LINUX Ubuntu 20.04LTS**

- 1. Verificar si podemos reconocer el bloque NXT con el siguiente comando

```
$ lsusb | grep Lego
```

```
Bus 002 Device 002: ID 0694:0002 Lego Group Mindstorms NXT
```

Si no obtenemos una salida similar, es probable que tengamos algo mal configurado o algún error en la comunicación USB ó el Ladrillo Lego está apagado.

Podemos instalar la biblioteca si no contamos con ella bajo el siguiente comando:

```
$ sudo apt-get install libusb-dev
```

- 2. Para poder compilar y descargar nuestros programas NXC o NBC, necesitamos instalar parte de actual del proyecto **nbc-compiler** mantenido por el usuario pierre-24 de github <https://github.com/pierre-24/nbc-compiler>

Uno de los requisitos es la instalación del compilador de pascal **fpc** (free pascal compiler) Para instalarlo haremos uso del siguiente comando:

```
$ sudo apt-get install fpc
```

- 3. Una vez cubiertos los requisitos para la instalacion del compilador nbc, hacemos una copia del repositorio **nbc-compiler** con los siguientes comandos: debemos tener instalado el paquete de git (`$ sudo apt-get install git-all`)

```
$ git clone https://github.com/pierre-24/nbc-compiler.git
```

Una vez clonado el repositorio entramos a la carpeta clonada y ejecutamos el siguiente comando:

```
$ make all
```

Si todo sale bien, se contará con un ejecutable dentro de la carpeta
Hacemos un pequeño test para comprobar si todo esta en orden usando `$./nbc test/bools.nbc`

se deberá tener una salida similar a:

```
# Status: NBC compilation begins
# Status: Compiling for firmware version 128, NBC/NXC enhanced = FALSE
# Status: Loading NBC system files
# Status: Running NBC Preprocessor
# Status: Include path = /home/pierre/code/nbc-compiler;/usr/local/include/nbc/;tests/
# Status: Processing include: NBCCommon.h
# Status: Compiling NBC source code
# Status: Finished compiling NBC source code
# Status: Finalizing dependencies
# Status: Optimizing at level 1
# Status: Build codespace references
# Status: Optimize mutexes
# Status: Compact the codespace
# Status: Remove unused labels
# Status: Compact the dataspace
# Status: Sort the dataspace
# Status: Generate raw dataspace data
# Status: Fill clump and codespace arrays
# Status: Update executable file header
# Status: Write file header to executable
# Status: Write dataspace to executable
# Status: Write clump data to executable
# Status: Write code to executable
# Status: Write optimized source to compiler output
# Status: Finished
```

4. Movamos el compilador clonado a la ruta `/usr/local/bin/` para hacer uso de el fuera de la carpeta clonada con el siguiente comando

```
$ sudo mv nbc /usr/local/bin
```

Comprobar que se encuentra bien ejecutando lo siguiente

```
$ which nbc
/usr/local/bin/nbc
```

5. Ahora podremos compilar y correr nuestros programas en el ladrillo bajo el siguiente comando:

```
$ sudo nbc -d miprograma.nxc
```

La bandera **-d** indica descargar el binario compilado. Para conocer más comando disponibles se puede ejecutar el help de la siguiente manera: **\$ nbc -help**

Es posible que haya notado que el compilador debe ejecutarse como superusuario para que funcione. Esto se debe a la forma en que Linux maneja los archivos del dispositivo y será necesario crear una regla udev para permitir que se ejecute con los permisos normales.

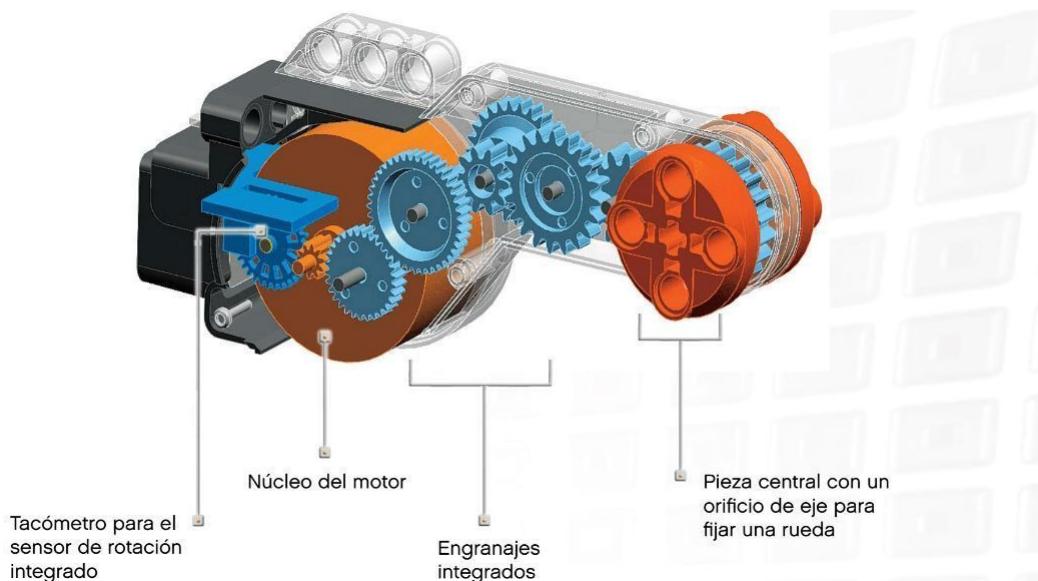
udev es la utilidad a cargo de los dispositivos dinámicos (por ejemplo, dispositivos USB que deben permitir la conexión de forma rápida). En los sistemas Linux, cuando se conecta un nuevo dispositivo, busca a través de un conjunto de reglas predefinidas para decirle qué hacer con el dispositivo y quién puede acceder a él. Los detalles de cómo configurar la regla udev necesaria son demasiado largos para incluirlos aquí, pero hay excelente instrucciones en este sitio.

<https://bricxcc.sourceforge.net/nbc/doc/nxtlinux.txt>

II. Encoders y Cinemática Diferencial

Los encoders son sensores, podemos encontrarlos basados en diferentes tecnologías, y miden las revoluciones de un eje. Es decir, un encoder dice cuantas vueltas ha dado un eje en un intervalo de tiempo dado.

En robots móviles, los encoders se suelen situar en los ejes de las ruedas.



Un robot diferencial es aquel que solo cuenta con dos ruedas motrices sobre un mismo eje. Cada una de estas ruedas tiene su propio motor que la hace girar perpendicular al eje que une ambas ruedas.

En nuestro robot hay dos ruedas (izquierda y derecha) decimos que estan unidas a por un eje ya que se encontraran paralelas al mismo nivel una de otra. Estas ruedas tienen su propio motor y encoders que miden las revoluciones. Las ruedas giran en el plano perpendicular al eje que une a ambas ruedas, cada rueda sólo puede ir hacia adelante o hacia atrás.

Además de las ruedas motrices, también existe una rueda loca que simplemente se utiliza para darle estabilidad al robot móvil. Esta rueda loca no es motorizada ni es importante que tenga encoders.

Al trabajar con el kit de robótica de LEGO Mindstorm NXT, los encoders se encuentran ya incluidos y disponibles para su uso bajo, hacemos uso de la función **MotorRotationCount**, que obtiene un valor de posición relativa respecto a una salida específica. La salida la obtenemos en grados.



Figura 1: Robot Lego utilizado

III. Modelo Cinemático

El modelo cinemático es el estudio de movimiento de sistemas mecánicos sin considerar las fuerzas que afectan dicho movimiento. Para el robot móvil diferencial, su principal propósito es representar la velocidad del robot en función a las velocidades de las ruedas conjuntamente a los parámetros geométricos del robot.

Un robot móvil con ruedas se debe moverse sobre una superficie mediante la acción de ruedas montadas en él.

- El robot móvil se mueve sobre una superficie plana horizontal, es decir es constante
- Los ejes de referencia son perpendiculares al suelo
- No existen elementos flexibles en la estructura del robot
- El contacto entre cada rueda y el suelo se reduce a un solo punto
- No existe deslizamiento

El modelado del robot se puede representar como la siguiente imagen
Donde:

- θ : Ángulo de orientación del robot
- ω_l : Velocidad angular rueda izquierda
- ω_r : Velocidad angular rueda derecha
- \vec{v}_l : Velocidad rueda izquierda
- \vec{v}_r : Velocidad rueda derecha
- l : Distancia entre las dos ruedas medidas haciendo referencia a su punto de apoyo

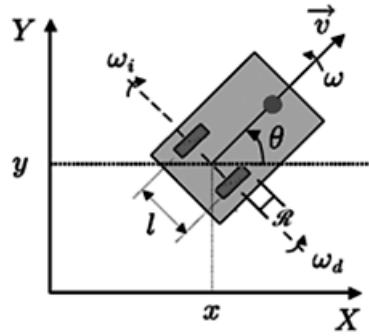


Figura 2: Modelo Robot

- R : Radio del robot
- x, y : Posición x, y en el plano
- D_L : Distancia recorrida rueda izquierda
- D_R : Distancia recorrida rueda derecha
- D_c : Distancia recorrida por el robot

Partiendo de la ubicación del centro de nuestro robot, considerando el punto medio como el punto que corta entre la línea de acción l y una perpendicular a la rueda motriz, considerando un robot móvil de direccionamiento diferencial no holonómico que se mueve en un plano horizontal, se describe su cinemática con las ecuaciones:

$$x' = \frac{dx}{dt} = v * \cos\theta \quad y' = \frac{dy}{dt} = v * \sin\theta \quad \dot{\theta} = \omega$$

$$p = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

es el vector que nos indica la posición y orientación del robot

Partiendo del gráfico del modelo del robot, podemos observar que existe una relaciones donde hacemos uso de funciones trigonométricas **Seno** y **Coseno**, para conocer la posición x, y de nuestro robot a medida que recopilamos datos del encoder. Pero antes debemos contar con el vector resultante de nuestro robot con los datos conocidos de la rueda izquierda y derecha.

Distancia Derecha - partiendo de las mediciones de rotación obtenidas en grados, las convertimos a radianes

$$D_L = radio * (LecturaRuedaIzquierda * (\frac{\pi}{180})) \quad D_R = radio * (LecturaRuedaDerecha * (\frac{\pi}{180}))$$

La distancia recorrida por nuestro robot será el promedio de ambas distancias:

$$D_{ROBOT} = \frac{(D_R + D_L)}{2}$$

Una vez obtenidos los parámetros anteriores, podemos calcular nuestro ángulo de giro ω

$$\omega = \frac{(DR - DL)}{\text{longitud entre ruedas}}$$

Nuestro ángulo de giro obtenido de la ecuación anterior nos dan los grados en radianes, para obtenerlo en grados multiplicamos por $\frac{180}{\pi}$, de esta forma podremos hacer uso de las primeras ecuaciones para conocer las nuevas coordenadas x,y en el plano.

$$x = x + \text{Distancia Robot} * \frac{\cos(\theta)}{100} \quad y = y + \text{Distancia Robot} * \frac{\sin(\theta)}{100}$$

iv. Programa

El programa del odómetro debe consistir en las siguientes fases dentro de un ciclo infinito

- Leer tacómetros
- Comparar con lectura anterior
- Calcular las diferencias ΔX
- Calcular las diferencias ΔY
- Calcular las diferencias $\Delta\theta$
- Actualizar odómetro ($X + \Delta X, Y + \Delta Y, \theta + \Delta\theta$)
- Actualizar mediciones

v. Pseudocódigo

Durante la realización del odómetro, el interprete no soporta asignaciones con expresiones largas, es por eso que en el código la asignación para las nuevas coordenadas x,y se dividieron para así contar con los datos correctos.

```
Leer tacometros;  
Comparar con lectura anterior;  
if supera umbral then  
     $\Delta X \leftarrow X + \Delta X;$   
     $\Delta Y \leftarrow Y + \Delta Y;$   
     $\Delta\Theta \leftarrow \Theta + \Delta\Theta;$   
    Actualizar Odómetro;  
    Actualizar Tacometro;  
end  
Wait(50ms);
```

Algorithm 1: Odómetro

La implementación quedará dentro de una tarea (task), ya que el kit de LEGO NXT haciendo uso de NXC soporta multiples hilos de ejecución de tareas. Se incluyen más tareas como un seguidor de linea básico, e ir mostrando el desplazamiento del robot en la pantalla del ladrillo LEGO. La ejecución de los hilos la realiza el hilo principal con ayuda un organizador de una lista de tareas **Precedes** con el siguiente orden:

Precedes(odometria,sigue_lineas,make_draw)

vi. Pruebas

Durante las pruebas se pueden observar errores, dichos errores sistemáticos los podemos considerar agenos al modelo dichos errores pueden ser variados como:

-
- Los diámetros de las ruedas no son iguales
 - Medida errónea de los diámetros de las ruedas
 - Mal alineación de las ruedas
 - Irregularidad del terreno
 - Derrape, patinaje

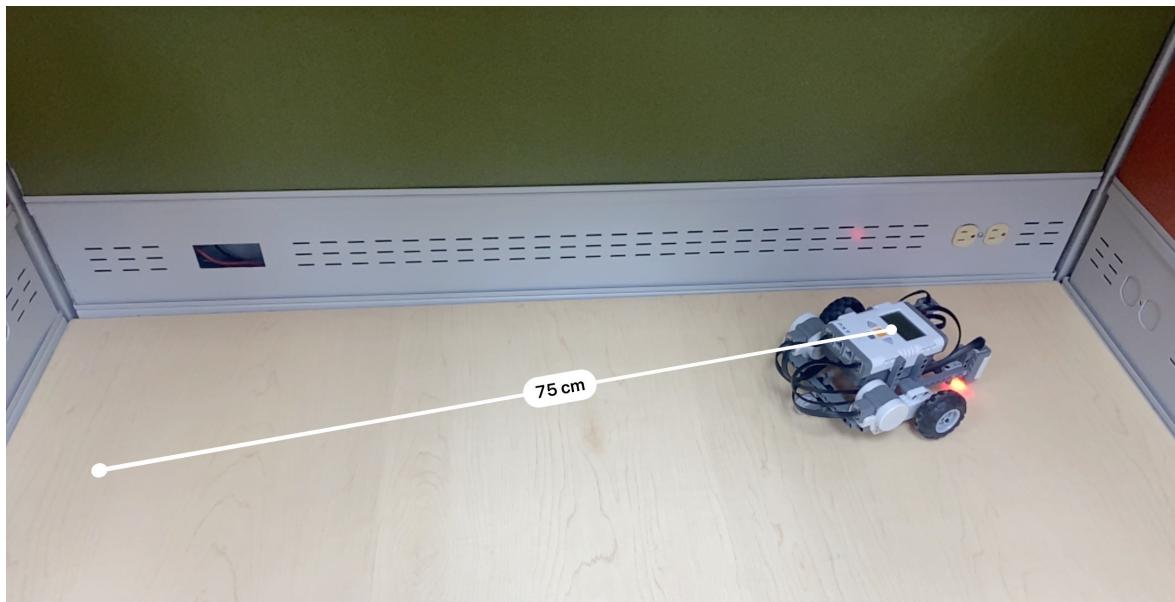
Los errores no sistemáticos los podemos corregir con una calibración. A continuación se muestran tres errores de tres corridas diferentes.



En la primera imagen después de un desplazamiento aproximado de 86 cm, los datos de desplazamiento del robot fueron: 88.49 cm y una inclinación de 10.38°



En la segunda prueba después de un desplazamiento aproximado de 75 cm, los datos de desplazamiento del robot fueron: 79.48 cm y una inclinación de 18.32°

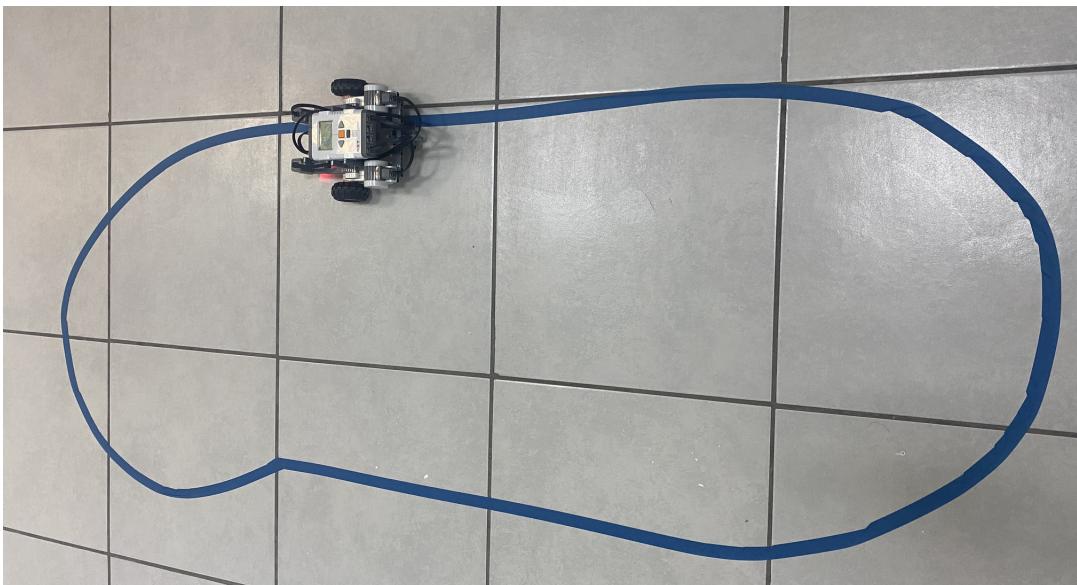


En la tercera prueba después de un desplazamiento aproximado de 86 cm, los datos de desplazamiento del robot fueron: 87.56 cm y una inclinación de 18.65°



VII. Prueba con el siguelineas

Al tener calibrado nuestras mediciones del odómetro, lo integramos dentro de la rutina de sigue-líneas.



Al superponer los recorridos, donde se observa el error por la oscilación originado de la colocación de los sensores que es amplio.

III. RESULTADOS

Los resultados de la estimación de la posición están dentro de lo esperado, teniendo en cuenta la precisión de los sensores y motores. Al cabo de varios metros recorridos el error empieza a ser considerable. Lo que más error produce son las trayectorias curvas, ya que los rozamientos hacen que los cálculos hechos con las lecturas de los tacómetros de los motores introduzcan demasiado error cuando el robot está girando.

El código se puede ver en la siguiente liga Github

IV. CONCLUSIONES

En conclusión, la odometría es un método comúnmente utilizado para estimar la posición y orientación de un robot móvil en un entorno desconocido. Sin embargo, como cualquier técnica de medición, la odometría tiene errores que pueden afectar la estimación de la posición y orientación del robot.

Los errores sistemáticos en la odometría pueden deberse a una variedad de factores, como el desgaste de las ruedas, la fricción en la superficie del suelo y la falta de precisión en los sensores utilizados para medir la velocidad de las ruedas. Estos errores pueden acumularse con el tiempo y afectar la capacidad del robot para navegar en un ambiente desconocido.

Para reducir los errores, es importante realizar una calibración de las mediciones de los encoders y el mantenimiento/ajuste del robot.

En resumen, aunque los errores sistemáticos son un desafío importante en la odometría, existen formas de minimizar su impacto y mejorar la precisión de la navegación del robot móvil. Es clave comprender y atender los errores de manera efectiva para lograr un rendimiento óptimo en la navegación autónoma del robot.

REFERENCIAS

- [1] Instalación NXC en LINUX, <http://ubuntudaily.blogspot.com/2011/03/using-lego-mindstorms-nxt-with-ubuntu.html>
- [2] Repositorio Compilador NBC utilizado, <https://github.com/pierre-24/nbc-compiler>
- [3] Documento para evitar sudo en NXC, <https://bricxcc.sourceforge.net/nbc/doc/nxtlinux.txt>
- [4] Comando para instalar libusb-dev, <https://howtoinstall.co/en/libusb-dev>
- [5] Presentación Odometría, <http://www.kramirez.net/Robotica/Material/Presentaciones/Odometria.pdf>
- [6] Modelo Cinemático de un robot móvil tipo diferencial y navegación a partir de la estimación odometrífica, <https://www.redalyc.org/pdf/849/84916680034.pdf> VALENCIA V., JHONNY A.; MONTOYA O., ALEJANDRO; RIOS, LUIS HERNANDO

-
- [7] Modelo cinemático de un robot móvil implementado con LEGO NXT para un sistema de localización indoor diseñado en Labview, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiu0_So28_9AhViDkQIHR1XDtcQFnoECBQQAQ&url=https%3A%2F%2Frevistas.udistrital.edu.co%2Findex.php%2FTecnura%2Farticle%2Fdownload%2F6810%2F8394%2F30717&usg=A0vVaw2PsCrkFGkk_nGN-G084B11
 - [8] Notas de clase, Robótica Móvil Inteligente, Dr. José Gabriel Ramírez Torres, Enero-Abril 2023