

Odometría

LUIS ALBERTO BALLADO ARADIAS

Cinvestav Unidad Tamaulipas

luis.ballado@cinvestav.mx

8 de marzo 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**

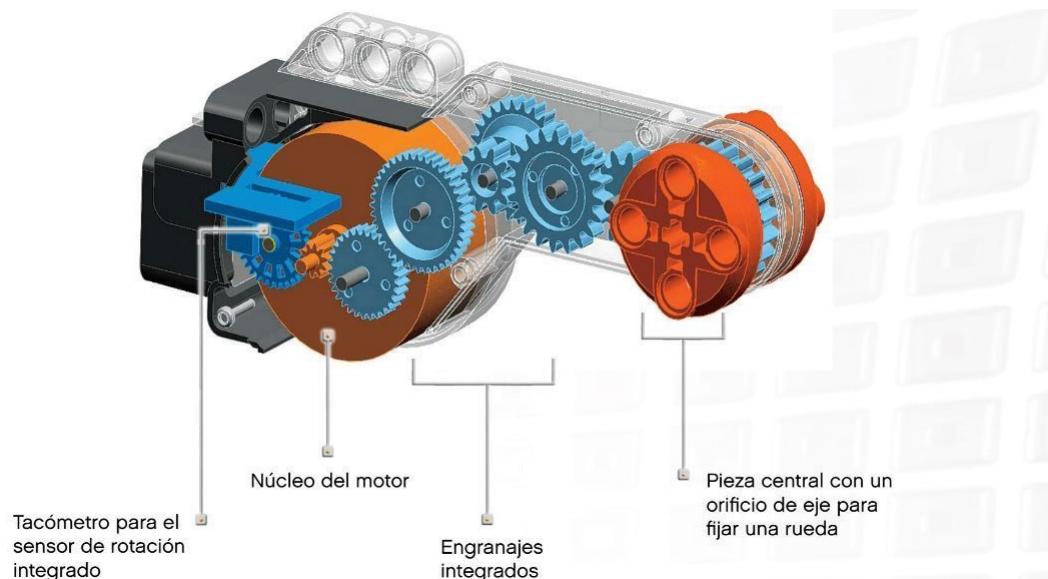
Es posible que haya notado que el compilador debe ejecutarse como superusuario para que funcione. Esto se debe a la forma en que Linux maneja los archivos del dispositivo y será necesario

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 la siguiente función:

III. Modelo Cinemático

El programa del odómetro debe consistir en las siguientes instrucciones:

- 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

IV. Pseudocódigo

La parte fundamental

```
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 Tacómetro;  
end  
Wait(50ms);
```

Algorithm 1: Odómetro

v. Pruebas



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. En recorridos de unos 8 metros, tiene un error de entre unos 300 y 400 mm. Lo que más error produce son las trayectorias curvas, mayor velocidad angular es necesario peor, 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.

Trayecto de 4 metros en linea recta: En 10 pruebas realizadas, el error medio en distancia fue de 15 cm.

Giro sobre si mismo de 360° En 5 pruebas realizadas, el error medio en grados era de 3°.
Trayecto de 3 metros con giro: En 10 pruebas realizadas, el error medio en distancia fue de 24 cm.

IV. DISCUSIÓN Y ANÁLISIS

REFERENCIAS