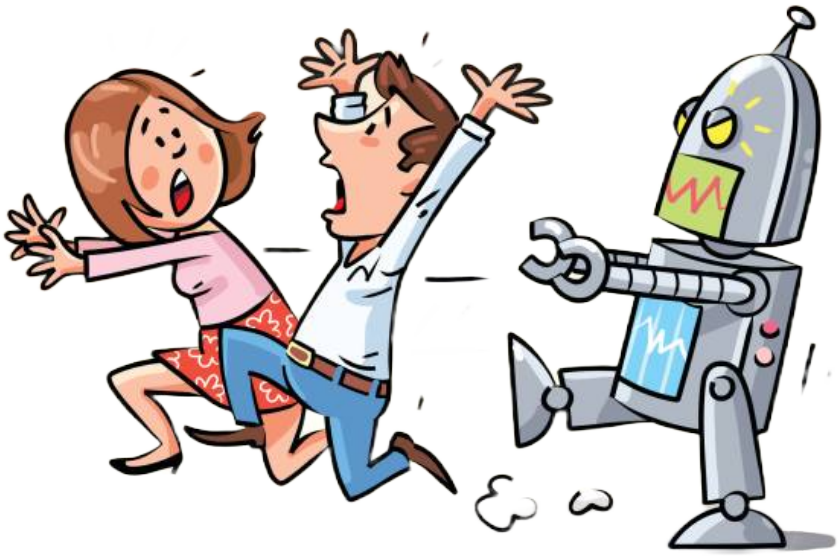
Universidad de Alicante

Ingeniería Robótica

Desarrollo de un sistema de seguimiento automático para jugar al pilla pilla con un turtlebot



Autores:

Álvaro Martínez Martínez y Marina Villanueva Pelayo

15 de enero de 2023

Índice

1. Objetivo

El objetivo de la práctica final es la programación de un robot móvil que sea capaz de resolver una tarea compleja y completa con todo lo aprendido durante el curso. En este caso, la tarea que se cumple es el seguimiento de una persona que posee un distintivo haciendo uso de visión artificial, evitando obstáculos y dirigiéndose hacia un objetivo. La orientación que se le ha dado ha sido un "PillaPilla", donde el humano deberá huir del robot y el robot, cuando a través de su cámara detecte el distintivo, seguirá al humano hasta alcanzarlo.

2. Sensores utilizados

Para poder implementar todo el ejercicio, es muy importante que el turtlebot cuente con los tres sensores que permiten su funcionamiento.

(a) Sensor laser (b) Cámara (c) Base

Figura 1: Sensores necesarios

2.1. Sensor láser Hokuyo ust-10lx

El Hokuyo UST-10LX es un sensor láser de escaneo de alta precisión que se utiliza comúnmente en robótica y automatización. Es capaz de medir la distancia a objetos en un ángulo de 270 grados con una resolución de 0.25 grados y una precisión de hasta 40 mm. Además, tiene un rango de medición de 0,1 a 10 metros y puede medir hasta 8192 puntos por escaneo.

El sensor utiliza un láser de clase 1 que emite un haz de luz infrarroja inofensivo para el ojo humano. El haz se escanea mediante un espejo giratorio para medir las distancias a los objetos dentro del ángulo de medición. Los datos recopilados se pueden utilizar para crear mapas tridimensionales del entorno, evitar obstáculos o navegar de forma autónoma.

El Hokuyo UST-10LX es compatible con varios sistemas operativos, incluyendo Windows, Linux y ROS (Robot Operating System). Además, existen paquetes de ROS compatibles con el UST-10LX que facilitan su integración en proyectos de robótica.

2.2. Cámara Orbeec Astra

La cámara Astra es una cámara 3D producida por Orbbec. Esta cámara utiliza tecnología de estructura de luz (ToF) para capturar imágenes 3D en un rango de 0.4 hasta 8 metros. La cámara Astra es capaz de capturar imágenes a una resolución de 640 x 480 a 30 frames por segundo y tiene un ángulo de visión de 60 grados.

2.3. Base Kobuki

La base Kobuki es lo que permite el movimiento del turtlebot. Cuenta con dos ruedas convencionales y una rueda de equilibrio que permiten una gran movilidad y estabilidad. Además, tiene un sistema de navegación por odometría, un sistema de carga y descarga de baterías, un sistema de detección de colisiones y un sistema de control de velocidad.

3. Desarrollo de las tareas

El juego del pilla pilla contra un turtlebot funciona de la siguiente manera.

1. Al ejecutar el programa, se hace una cuenta atrás de 3 segundos para que cada uno de los participantes se coloque en su puesto. Para poder jugar, es importante que las personas que quieran jugar lleven un distintivo de color azul en la pierna, a la altura de debajo de la rodilla aproximadamente.

2. Al acabar la cuenta atrás, el turtlebot empieza a moverse de forma arbitraria por el espacio. En este momento, va deambulando evitando los obstáculos del entorno.

    a) Si detecta una persona, deja de deambular y su movimiento se dirige hacia ella. Si consigue acercarse mucho, gana un punto el turtlebot.

    b) Si una persona se acerca antes al turtlebot y le toca el sensor del bumper, es la persona la que gana punto.

3. Cada vez que se gana un punto, aparece la puntuación en el terminal.

En este trabajo, se distinguen dos tareas que conforman el ejercicio final.

3.1. Subtarea de movimiento

La subtarea más simple que se ha de realizar es el propio movimiento del robot. Esta puede realizarse en dos modos:

El primero de ellos se produce cuando la cámara no detecta el distintivo en la imagen, de modo que no habría un objetivo a seguir. En este caso el robot irá vagando por la zona evitando los obstáculos hasta que en la cámara aparezca el color objetivo, y es entonces que cambia a su segundo modo de movimiento.

El segundo de los modos de movimiento es el que se emplea cuando la cámara detecta un objetivo. En este modo el robot va tras la persona, aunque también evita obstáculos y paredes, para ello recibe las coordenadas del centroide del identificador en la imagen, y a partir de estas trata de avanzar hacia delante mientras lo mantiene en el centro de la imagen. Por razones de seguridad la velocidad máxima que el robot podía alcanzar fue limitada debido al poco espacio disponible en el laboratorio. Por otro lado, para ajustar la velocidad angular se tenía en cuenta la distancia de la coordenada X al centro de la imagen, de modo que cuanto más centrado se encuentre el identificador menor velocidad angular se le daría al robot, mientras que cuanto más alejado se encontrase del centro mayor velocidad se le daría para así centrarlo lo antes posible.

3.2. Subtarea de detección

La segunda de las subtareas que el robot realiza es la detección de un identificador que el humano lleva en la pierna. Para realizar esta tarea se creó un nuevo nodo cuya función era únicamente sacar las coordenadas del centroide con ayuda de la librería OpenCV. Una vez obtenidos los valores necesarios se ha de poder acceder a esta información desde el nodo que hará mover al robot, para ello se hace uso de un topic y un mensaje creado exclusivamente para esta tarea.

El mensaje creado para poder enviar la información está formado por los siguientes componentes:

- **X** : Coordenada X del identificador.
- **Y** : Coordenada Y del identificador.
- **Widht** : Ancho de la imagen
- **Height** : Alto de la imagen
- **Detected** : Variable booleana con la que saber si hay detección o no

La variable booleana "Detected.es necesaria porque de esta depende si el robot actúa en el primer modo de movimiento o en el segundo. Además, las dimensiones de la imagen son usadas para saber en que zona de la imagen se encuentra la coordenada X y así poder ajustar la velocidad angular.

La detección del identificador se realiza aislando el color del este respecto al resto de colores, de modo que se obtiene una imagen binaría en la que solo aparecen coloreados los colores del identificador. Para eliminar pequeñas zonas que suelen aparecer por error se calcula el área de todos los contornos que aparecen en la imagen, de modo que se eliminan todos excepto al mayor, que es el que interesa. Una vez obtenido este contorno es simple obtener su centroide utilizando técnicas matemáticas de visión artificial.

4. Paquete pilla-pilla

Para este proyecto se ha creado desde cero un paquete en ROS con los dos nodos que envían y reciben la información necesaria para jugar al pilla-pilla. El contenido del paquete es el siguiente:

- **msg**: En esta carpeta se encuentran definidos los mensajes que se envían de un nodo a otro.
- **src**: Dentro de esta carpeta están los scripts de los nodos que hacen funcionar el programa.
  - **detect.py**: En este código se lee la información que detecta la cámara. Cuando la cámara detecta el color azul, publica en el topic */topic_posicion* los mensajes con la información de la región de interés.
  - **follow.py**: Este nodo está suscrito al sensor láser para que el robot pueda moverse, al bumper para detectar si el humano ha ganado punto, y a la cámara para saber donde está el humano. Inicialmente va deambulando hasta que el otro nodo le envía la información de que se ha detectado una zona de color azul. Es entonces cuando cambia su rumbo a esa dirección. Este programa ha sido implementado mediante una máquina de estados que se explica a continuación.

4.1. Programación con máquina de estados

Una máquina de estados es un modelo formal de programación que describe cómo un sistema puede cambiar entre diferentes estados y cómo responde a eventos externos en cada estado. Proporciona una manera estructurada y fácil de entender para diseñar sistemas complejos, ya que permite dividir el sistema en estados y transiciones claramente definidos.

En el caso del robot pilla-pilla, se han definido los siguientes estados:

- **Estado 0:** Es el estado base. En él se hace la cuenta atrás para comenzar a jugar y se asegura de tener espacio suficiente para moverse.
- **Estado 1:** En este caso el robot avanza en línea recta. Va acelerando progresivamente hasta llegar a la velocidad máxima indicada. En caso de detectar paredes cercanas, se intentan evitar modificando ligeramente la velocidad angular antes de pasar a los siguientes estados.
- **Estado 2:** Es un estado de decisión, donde se comprueba a que lado es mejor girar porque no hay opción de seguir en línea recta.
- **Estado 3:** En este caso el robot gira al lado derecho decelerando hasta que la velocidad lineal es 0. Cuando detecte suficiente espacio libre hacia el frente, vuelve al estado 1.
- **Estado 4:** En este caso el robot gira al lado izquierdo decelerando hasta que la velocidad lineal es 0. Al igual que en el caso anterior, cuando detecte suficiente espacio libre hacia el frente, vuelve al estado 1.
- **Estado 5:** Es el caso de detectar el distintivo de color azul. Comienza a acelerar en línea recta y la velocidad angular dependerá de en qué lado detecte la región de interés. Por supuesto en este caso también tiene en cuenta el entorno y sigue evitando los obstáculos igual que antes, como se ha explicado en el apartado 3.1.

Los cambios que producen los cambios de los estado son las distancias detectadas por el sensor láser y la detección o no del área azul.

5. Demostración

El resultado del turtlebot jugando al pilla-pilla se puede ver en el siguiente vídeo: `https://youtube.com/shorts/ZLqL8G-TDak`

Además, todo el material está subido a Github: `https://github.com/mariinaVillanueva/pilla-pilla.git`

6. Conclusión

En conclusión, el paquete de ROS desarrollado para jugar al juego “pilla pilla” con un TurtleBot ha demostrado ser un éxito en la utilización de técnicas de robótica móvil y visión artificial. El desarrollo de este paquete ha permitido al TurtleBot navegar de manera autónoma en un entorno

desconocido, evitando obstáculos y buscando objetos. Además, la implementación de visión artificial ha permitido al TurtleBot detectar y seguir a un objeto en movimiento, lo que ha sido esencial para el juego “pilla pilla”.

En general, este proyecto ha demostrado la capacidad del TurtleBot para realizar tareas complejas de robótica móvil y visión artificial, y haciendo uso de ROS como marco de trabajo, ha permitido una fácil integración de diferentes componentes y algoritmos avanzados.