Follow me

Duckietown Engineering Chile

Equipo ejecutor Josué Celis Muñoz

Valeria Palma

Gabriela Urbina

Profesional responsable Eduardo Jorquera
Gonzalo Olave
Ignacio Romero Aravena
Felipe San Martín

Fecha 18 de diciembre de 2022

Duckietown
Engineering
Chile

1. Problemática

En la ciudad de Duckietown, se ha visto la necesidad de mayor presencia de transporte público, debido al aumento de habitantes y con esto la cantidad de patos que necesitan del servicio, además, el transporte no contaba con servicio nocturno por la poca iluminación de la ciudad, dificultando así la movilidad de los patos desde que anochece.

Al mismo tiempo, el poco transporte público existente podía llegar a ser bastante inseguro para los habitantes, ya que muchas veces los pato-choferes no respetaban o mantenían la distancia segura establecida para el manejo de vehículos y como consecuencia de esto, se registró una importante cantidad de accidentes por alcance entre duckiebots.

Para resolver esta problemática, se desarrolló un sistema de seguimiento capaz de mantener una distancia segura de manera autónoma y que pueda funcionar en la oscuridad. El cual opera de la forma en que cada duckiebot de transporte público siga al vehículo de adelante manteniendo una distancia segura y una velocidad prudente entre vehículos a partir del tamaño de área detectada dependiente de la distancia entre ellos. Si llegara a perder o no identificar al vehículo delantero, realizaría una rotación hasta volver a identificarlo y poder seguir su ruta. De esta forma mejoraría la situación del transporte público y la calidad de vida de los habitantes de Duckietown.

2. Objetivos

Objetivo general:

- Desarrollar e implementar un programa de seguimiento entre duckiebots.

Objetivos específicos:

1. Implementar un sistema de detección por colores.
2. Estimar la posición del objeto.
3. Crear un nodo de ROS para mover las ruedas manteniendo una distancia segura con el objeto.
4. Generar un movimiento de rotación desde que no identifica al objeto hasta volver a detectarlo.
5. Adecuar sistema en la oscuridad o en lugares con cambios de luz.

3. Desarrollo del proyecto

Marco teórico

- Image Processing: Es el proceso de transformar una imagen en forma digital y realizar ciertas operaciones para obtener información útil de ella. [1]
- Computer Vision: Campo de la IA que permite que las computadoras y sistemas obtengan información significativa de entradas visuales, y tomen acciones basadas en esa información. [2]
- Blob (Binary Large Object): Colección de datos concentrados que se almacenan en un archivo en una base de datos o en un programa específico. [3]
- Apriltags: Sistema visual en que los objetivos se pueden crear con una impresora ordinaria, y el software de detección calcula la posición, la orientación y la identidad 3D precisas de las etiquetas en relación con la cámara.[4]

¿Qué hicimos?

Para comenzar con el proyecto usamos como base los 2 códigos creados durante las capacitaciones 4 y 5, en las cuales se usaron herramientas de visión 3D y la librería de Python OpenCV para lograr que los Duckiebots se detengan al detectar patos a una cierta distancia. A partir de los archivos duck_detector.py y controller.py creamos la parte central del proyecto, los archivos bot_detector.py y FUN_FOLLOW.py respectivamente. Cuyas funciones se describen a continuación:

- bot_detector.py: En este código se usa la detección por colores de duck_detector.py ajustando el rango de colores para poder detectar el color verde del disco que se adhiere al duckiebot guia (en vez del amarillo de los patos), encerrando el circulo de color en un rectángulo. Cabe destacar que, en un inicio la idea era detectar el color azul característico de los duckies pero, debido a que es un color bastante común en el fablab, el robot detectaba objetos no deseados en el entorno imposibilitando su uso.
  Además, en este nodo se publica el mensaje “dis” del tipo “point” en “duckiebot/posicion_bot”. En su primera y segunda componente se indica el centro del blob en el eje x e y respectivamente. La tercera componente entrega el área de la detección, calculada como el ancho por el alto del rectángulo que contiene al blob. Junto con eso, se publica una imagen con el blob detectado en “dis_bot”, pero es netamente para saber qué es lo que ve el robot por lo que no tiene mayor relevancia una vez finalizada la fase de calibración.

- FUN_FOLLOW.py: Este nodo está suscrito al tópico donde publica el nodo anterior y se procesa la información que tiene el mensaje “dis”. El primer componente se usa para calcular la variable “self.sep”, correspondiente a la separación entre el centro de la imagen y el centro del rectángulo de detección (en el eje x). El centro de la imagen en una primera instancia, por recomendación de un tutor, se fijó calculando el centro en píxeles a partir de la

resolución que tiene la cámara. Sin embargo, resultó más eficiente encontrar el centro “manualmente”, realizando un print del centro del blob en el eje x y posicionado el círculo frente a la cámara en el centro de la imagen para así fijar esa posición (87 píxeles) como el centro de la imagen. El tercer componente (de la cual se realiza un pint) permite estimar qué tan lejos está el objeto y establecer una distancia segura entre estos (área de 2000 pixeles), pues a mayor área menor es la separación entre los duckies. Inicialmente se iba a usar la distancia en el eje z en vez del área, pero como la detección por colores no es muy estable se obtiene una pésima aproximación de esta separación. Cabe mencionar que, como en Duckietown no hay pendientes, para este proyecto la segunda componente del mensaje “dis” es innecesaria.

Con lo anterior, se pudo regular la velocidad mediante una regla de proporcionalidad. Para la velocidad angular se estableció esta relación con “self.sep” y se usó la constante de proporcionalidad “k” que varía para ciertos rangos del valor de la cantidad mencionada (“k” crece si la separación aumenta). La definición de la variable (“self.sep”) se hizo de modo que su signo entregue de forma adecuada el sentido de giro.

La velocidad lineal es proporcional al área normalizada por 10 (debido a que los valores de esta última variable eran muy grandes) con constante “c” que varía para rangos determinados del área (“c” decrece si el área es mayor para ir más lento si se está más cerca). Cabe destacar que, cuando no hay detecciones el área es nula y en este caso se asignan valores específicos para “self.sep” y “k” de modo que el Duckiebot gire hasta que pueda realizar otra detección.

Se debe considerar que, las constantes dependen de la superficie en la que se moverá el robot y se calibraron mediante un procedimiento coloquialmente conocido como el “método de prueba y error”.

Además, para seguir con la forma de trabajo usada en capacitaciones anteriores, FUN_FOLLOW.py se suscribe al tópico donde publica el nodo “pato” del archivo pato.py para procesar la velocidad del Duckiebot. Sin embargo, esta suscripción es prácticamente innecesaria y se puede eliminar realizando algunas modificaciones en el archivo.

Algunas complicaciones

Si bien intentamos solucionar todos los problemas que se presentaron en el camino hay ciertos conflictos que son inherentes al algoritmo utilizado. En este caso, la detección por colores no es estable y presenta problemas con la luminosidad del ambiente, por esa razón puede ocurrir que, en el mismo camino, el robot detecte al Duckiebot guia solo en uno de los sentidos de la pista. Para solucionarlo, probamos poner una linterna en nuestro “pato-auto” para que todo lo que observa se mantenga en un cierto rango de luminosidad. Además, esto le permite al duckie funcionar de noche y en recintos con escasa o nula iluminación.

Resultados

En resumen, como resultado de todo el trabajo realizado en clases y tardes extras, se obtuvieron dos códigos python esenciales; el código *bot_detector.py* que tiene como objetivo principal encerrar en un rectángulo rojo el blob detectado por el color verde del disco, y finalmente *FUN_FOLLOW.py* que a grandes rasgos, controla la velocidad de las ruedas según el área del blob detectado.

Luego, juntando todo lo desarrollado se obtiene un sistema autónomo de seguimiento por detección del color. El cual permite que los duckiebots a los que se les implemente este programa mantengan una distancia segura entre sí, y un control de velocidad sobre las ruedas dependiendo si se acerca o se aleja mucho del objetivo. Además, como el sistema de reconocimiento de color es sensible a la luminosidad del sector en el que se encuentra, se le añadió una linterna para que pueda detectarlo de mejor forma y así se puede ocupar de noche. Sin embargo, esta última mejora concluyó en la fase de prueba.

4. Conclusiones y trabajo futuro

En conclusión, se lograron los objetivos mencionados al inicio, ya que se desarrolló exitosamente un programa de seguimiento entre dos duckiebots por detección de colores, el cual funciona a pesar de las variaciones de luz si es que se continúa con la implementación de la linterna. No obstante, estos objetivos no son exactamente iguales a los que se establecieron al comienzo y las variaciones realizadas fueron para aterrizar el proyecto para que así sea realizable en el tiempo disponible.

Por otro lado, como trabajo futuro se recomienda crear un “trencito” de duckiebots, implementar apriltags y/o añadir una nueva cámara del estilo OAK-D, la cual entregaría una mayor precisión. También recomendamos que los futuros trainees tengan en consideración que la detección de colores es sensible al cambio de iluminación, si eligen ese camino para desarrollar su proyecto.

Por lo tanto, a modo de cierre, el trabajo futuro que se mencionó anteriormente es una misión que quedará pendiente y se espera que se cumpla por futuros equipos trainee.

5. Bibliografía

[1] Simplilearn. (2022, noviembre 23) “What Is Image Processing? Overview, Applications, Benefits, and Who Should Learn It” [Online]. Disponible en: https://www.simplilearn.com/image-processing-article#what_is_image_processing

[2] IBM. “¿Qué es la Visión Artificial?” [Online]. Disponible en: https://www.ibm.com/cl-es/topics/computer-vision

[3] Canto. (2021, febrero 10) “Binary large object: How to store large data” [Online]. Disponible en: https://www.canto.com/blog/binary-large-object/

[4]April robotics laboratory at the University of Michigan. (2010) “AprilTags Visual Fiducial System” [Online]. Disponible en: https://april.eecs.umich.edu/software/apriltag

6. Anexos

Repositorio de videos en YouTube:
https://www.youtube.com/@grupo4duckietownfcfm153/featured

Repositorio de GitHub:
https://github.com/jodcm/Follow-me-G4