DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10, No. Especial 4 (2022) 92–99

Resolución de tareas con drones en simuladores
Solving drone tasks in simulators

A. Cureño-Ramirez [a,*], A. Juárez-Terrazas [b], R. S. Arellano-Aguilar [b], J. M. Ibarra-Zannatha [a]

[a] *Centro de Investigación y de Estudios Avanzados del IPN, Av. IPN 2508, San Pedro Zacatenco Alcaldía GAM, 07360 Ciudad de México, CDMX.*
[b] *Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas del IPN Av IPN 2580, La Laguna Ticomán, Alcaldía GAM, 07340 Ciudad de México, CDMX.*

Resumen

En este trabajo se presentan soluciones a varias de las pruebas incluidas en las competiciones para drones, en particular se consideran las pruebas que se tienen en el Torneo Mexicano de Robótica de la Federación Mexicana de Robótica en la categoría virtual de drones. Es en este contexto que en este trabajo se muestra el desarrollo de varias aplicaciones de tipo académico que se han venido incluyendo dichas competiciones a nivel nacional. Así, las pruebas incluidas en este trabajo son: vuelo sobre punto específicos de una ruta, reconstrucción 3D de objetos en el entorno, vuelo entre ventanas y vuelo en formación. Para resolver estas pruebas se deben utilizar técnicas de visión artificial, localización y mapeo simultáneos, control tradicional, reconstrucción 3D, planificación de rutas y aprendizaje de máquina. Todas ellas se han estado trabajando de manera virtual en distintos simuladores como Gazebo y AirSim bajo ROS.

Palabras Clave: Drones autónomos, Visión por computadora, Simuladores de drones, ROS

Abstract

In this paper we present solutions to some of the tests included in drone competitions, in particular we consider the tests held in the Mexican Robotics Tournament of the Mexican Federation of Robotics in the virtual drone category. In this context, this work shows the development of some academic applications that have been included in these competitions at national level. Thus, the tests included in this work are: flight over specific points along a path, 3D reconstruction of objects in the environment, flight between windows and formation flying. To solve these tests, computer vision, simultaneous localisation and mapping, traditional control, 3D reconstruction, route planning and machine learning techniques must be used. All of them have been working virtually in different simulators such as Gazebo and AirSim under ROS.

Keywords: Autonomous Drones, Computer vision, Drone simulators, ROS

1. Introducción

Las aplicaciones de los drones multirrotor son cada vez más variadas y complicadas. En la Agricultura de Precisión se usan para detectar hierbas indeseadas y utilizar herbicidas tan solo en la zona afectada, ahorrando agroquímicos y disminuyendo sus impactos ecológicos; también son capaces de hacer levantamientos topográficos o de hacer la fotogrametría en sitios arqueológicos o bien en zonas urbanas modernas. También se utilizan en labores de evaluación, supervisión y análisis en zonas de desastres, así como en la búsqueda y auxilio de víctimas. Todo esto representa una ventana de oportunidad para la investigación en áreas como el Control Automático y la Percepción Visual entre otras. Es en este contexto que, con el fin de entrenar especialistas y de generar soluciones a muchos de los problemas técnicos que implican estas aplicaciones, se han creado diversas competiciones, tanto nacionales como internacionales, en donde el mundo académico propone metodologías, algorítmicas y otras soluciones a problemas de tipo académico.

1.1. *Competiciones*

El objetivo del laboratorio es la formación de recursos humanos del más alto nivel posible en las áreas que ahí se cultivan. Para alcanzar este objetivo hemos considerado varias estrategias entre las que destacan la realización de proyectos

*Autor para correspondencia: andres.cureno@cinvestav.mx
Correo electrónico: andres.cureno@cinvestav.mx (Andres Cureño-Ramirez), aaronjt@outlook.com (Aaron Juárez-Terrazas), a.a.r.samuel99@gmail.com (Rudyard Samúel Arellano-Aguilar), jibarra@cinvestav.mx (Juan Manuel Ibarra-Zannatha).

Historial del manuscrito: recibido el 16/05/2022, última versión-revisada recibida el 18/08/2022, aceptado el 29/08/2022, publicado el 05/10/2022. **DOI:** https://doi.org/10.29057/icbi.v10iEspecial4.9273

de investigación de actualidad y la participación en diversas competiciones, en particular con los estudiantes de pregrado que realizan sus tesis de licenciatura o su Servicio Social en el laboratorio.

La elección de la primera de estas dos estrategias resulta evidente, mientras que la segunda vale la pena que sea explicada. Las competiciones, en efecto, proporcionan un marco de referencia en el que los estudiantes se ven obligados a aportar soluciones a diversos problemas tecnológicos, a menudo originales, con fuertes limitaciones de presupuesto y, sobre todo, de tiempo. Dichas soluciones deben ser lo más eficientes que sea posible considerando, además, que el resto de los equipos participantes también quieren ganar. Así, las competiciones obligan a poner práctica el trabajo en equipo, la investigación, el desarrollo de soluciones tecnológicas y, resumiendo, ayudan a los estudiantes a adquirir muchas de las cualidades de un buen profesional en las áreas de la Robótica y la Visión Artificial. En el **Anexo 1** se presentan tres de las competiciones con drones que representan un mayor interés para el Laboratorio.

1.2. Estado del arte

Se han propuesto trabajos para simular drones como *Comprehensive simulation of quadrotor uavs using ros and gazebo* Meyer et al. (2012), *A high fidelity simulator for a quadrotor uav using ros and gazebo* Zhang et al. (2015) y también esta el simulador Sphinx Parrot (2022) pero se menciona en la documentación que ya no se le daba mantenimiento y el simulador presentaba fallas, por lo que se buscaron nuevas alternativas para trabajar.

1.3. Software utilizado

Sistema Operativo de Robots (ROS): es un framework flexible para escribir software de robots. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto a través de una amplia variedad de plataformas robóticas.

Gazebo: ofrece la posibilidad de simular de forma precisa y eficiente robots en entornos complejos, tanto interiores como exteriores. Se puede tener acceso a múltiples motores de física de alto rendimiento, también proporciona una representación realista de los entornos, incluyendo iluminación, sombras y texturas de alta calidad. Puede generar datos de sensores, tales como telémetro laser, cámaras 2D/3D, cámaras RGB-D como el Kinect, sensores de contacto, de fuerza, de torsión y más Gazebo (2022).

AirSim: es un simulador para drones y carros principalmente y está basado en el motor de juegos Unreal Engine EpicGames (2022) de Epic Games. Es de código libre, multiplataforma y puede trabajar con distintos controladores de vuelo como Autopilot PX4 (2022) y ArduPilot ArduPilot (2022) y contiene gráficos visuales realistas MicrosoftResearch (2022).

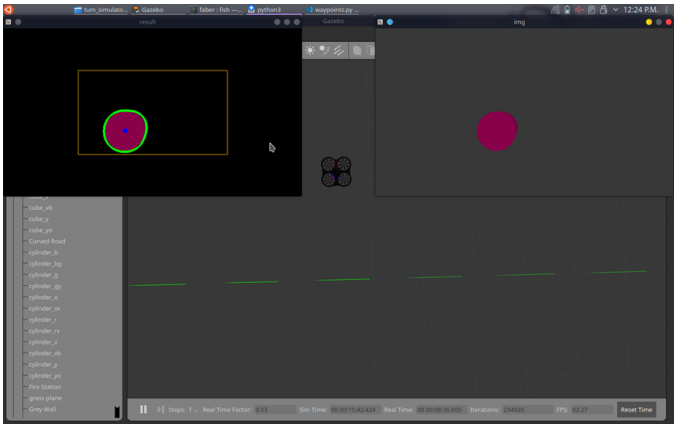

Figura 1: Dron sobrevolando uno de los marcadores antes de pasar al siguiente.

2. Resultados de las pruebas

Los drones son sistemas subactuados, pero los drones utilizados en los simuladores tienen como entradas de control las velocidades lineales en los ejes x, y, z y la velocidad angular sobre el eje z como se muestra en la figura 2. Por lo tanto solo se tienen cuatro entradas de control u_x, u_y, u_z y u_{yaw}, que toman valores positivos o negativos para indicar la dirección del movimiento o del giro.

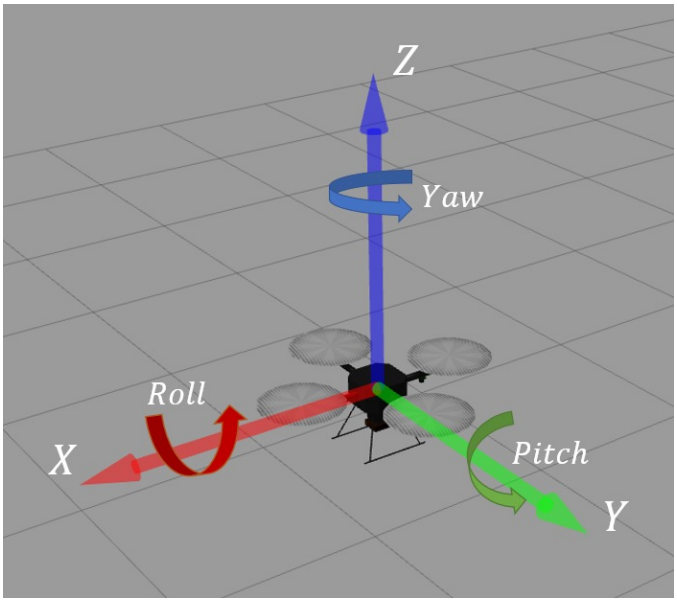

Figura 2: Marcos de referencia del dron.

Las ecuaciones de control se toman del trabajo Orozco-Soto et al. (2018), pero sin el vector de perturbaciones y son las siguientes:

$$u_{yaw} = K_{p\psi}(\psi_d - \psi_a) + K_{d\psi}(\dot{\psi_a}) \tag{1}$$

$$u_z = K_{pz}(z_d - z_a) + K_{dz}(\dot{z_a}) \tag{2}$$

$$u_y = \frac{K_{py}(y_d - y_a) + K_{dy}(\dot{y_a})}{u_z + mg} \tag{3}$$

$$u_x = \frac{K_{px}(x_d - x_a) + K_{dx}(\dot{x_a})}{u_z + mg} \quad (4)$$

Siendo ψ_d, z_d, y_d y x_d las posiciones y orientaciones deseadas y ψ_a, z_a, y_a y x_a las posiciones y orientaciones actuales, estas pueden ser obtenidas ya que dentro de los simuladores se generan datos de odometría o de posición global del dron dentro de los mundos virtuales.

2.1. *Vuelo sobre puntos específicos de una ruta*

En esta prueba se da una lista con las coordenadas de los puntos que debe visitar el dron, a partir de esta el dron debe moverse al marcador y sobrevolarlo por unos segundos antes de partir al siguiente. Originalmente, las marcas en el suelo debería ser identificadas de manera automática usando el sistema de visión embarcado, pero se simplificó.

Para que el dron visite cada una de las marcas en el suelo su movimiento se utilizan las ecuaciones (1) - (4), con los siguientes valores en $\psi_d = 0$ (siempre viendo al frente), $z_d = 2$ (que vuele a 2 metros de altura) y x_d y y_d van cambiando de acuerdo a los puntos de la lista. Al dron se le da una consigna de altitud; una vez alcanzada ésta se mantiene constante y el dron comienza a moverse hacia el primer marcador en la lista; mientras el dron se desplaza se utiliza la cámara inferior del dron para detectar los marcadores (círculos rojos). Cuando se detecta un marcador se calcula su centroide y el dron lo utiliza como consigna del controlador proporcional de movimiento, con o cual se desplaza hasta que el centroide del marcador se encuentre en el centro de la imagen tal como se observa en la figura 1. Una vez que termina por colocarse sobre el centro del marcador el dron se mantiene en esa posición por dos segundos y después se dirige al siguiente marcador. Una vez que ha visitado todos los marcadores el dron regresa a la posición de la cual despegó y aterriza.

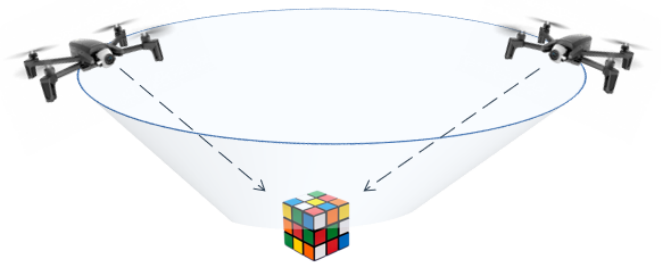

Figura 3: Representación de trayectoria circular del dron siempre mirando hacia el centro.

2.2. *Reconstrucción 3D de objetos en el entorno*

En esta prueba el dron debe sobrevolar el entorno a fin de obtener un mapa tridimensional de dicho entorno incluyendo los objetos ahí presentes. Se optó por obtener una nube de puntos de los objetos de interés con la técnica de ORBSLAM2 Mur-Artal et al. (2015) debido a que esta técnica se puede utilizar en tiempo real bajo ROS y que es de uso libre. ORBSLAM2 realiza la localización y mapeo de manera simultanea por medio de puntos clave en las imágenes obtenidas con el método de detección ORB, el cual se puede utilizar como una función de OpenCV. Una vez obtenida la nube de puntos, ésta se filtra con el objeto de eliminar aquellos puntos que no pertenecen a los objetos que se quieren reconstruir. Finalmente, teniendo ya la nube lo más limpia posible, se procede a realizar la reconstrucción del entorno (y los objetos ahí presentes) utilizando para ello uno de los siguientes dos métodos de reconstrucción 3D: PowerCrust Chen et al. (2018) que se basa en obtener el esqueleto Tagliasacchi et al. (2016), el cuál describe la topología y geometría de una forma y Screened Poisson Kazhdan and Hoppe (2013) se basa en que a cada punto se le puede asociar un vector normal a la superficie y construir un campo vectorial para la reconstrucción.

2.2.1. *Obtención de nubes con ORBSLAM2*

Para poder obtener un buen mapa de puntos con ORBSLAM2 será necesario que el dron vuele alrededor del objeto de interés con la condición de que siempre esté mirando hacia el suelo justo al debajo del centro de su trayectoria circular tal como se muestra en la figura 3. Para realizar esta trayectoria se utilizan las las ecuaciones (1) - (4), con los siguientes valores, para $z_d = cte$, se elige una altura constante dependiendo de la altura del objeto a inspeccionar, para $y_d = cte$, para que se mantenga realizando un movimiento lateral, para $x_d = r$ donde r es el radio deseado del circulo y para $\psi_d = \theta$ que es el ángulo en el que se encuentra el dron mientras se desplaza en el eje x y y, y permite ajustar la cámara del dron al centro del circulo.

En esta simulación el objeto de interés es el automóvil *Hatchback* que se muestra en la figura 4 disponible en los objetos predeterminados del simulador Gazebo. El objeto de interés y el dron con su cámara se colocan dentro de un mundo virtual en el simulaor Gazebo tal como se ve en la figura 5 y se hace que el dron siga la trayectoria circular predeterminada. Se captura la secuencia de imágenes producidas por la cámara (video) y se procesa mediante ORBSLAM2 para obtener en tiempo real la nube de puntos correspondiente, todo ello dentro del middleware ROS. En la figura 6 se muestran los resultados obtenidos al final de la trayectoria circular.

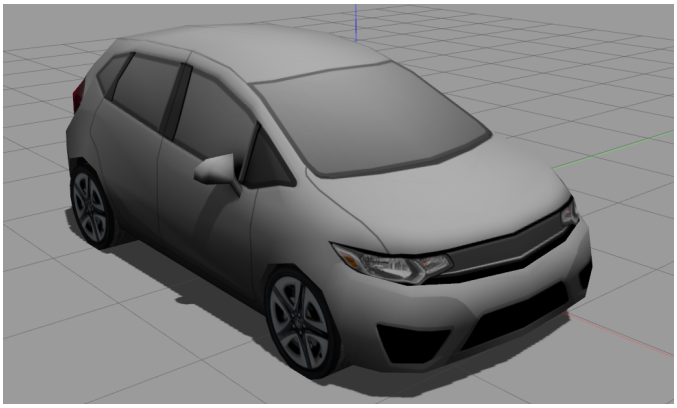

Figura 4: Automóvil Hatchback en Gazebo.

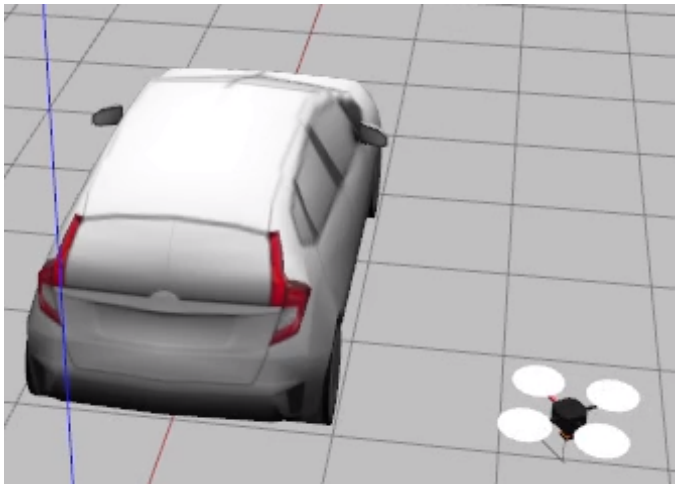

Figura 5: Automóvil Hatchback y dron en Gazebo.

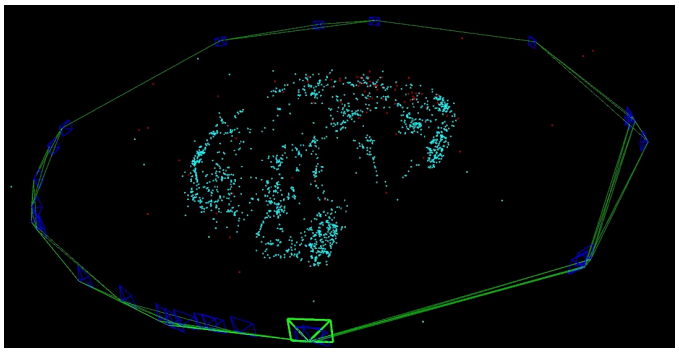

Figura 6: Nube de puntos en 3D obtenida con ORBSLAM2 del automóvil Hatchback.

2.2.2. *Reconstrucción 3D con filtrado*

Como se observa en la figura 6, la nube de puntos resultante tiene bastantes puntos que se encuentran fuera del objeto de interés, denominados *outlayers*, que se consideran como ruido, por lo que es necesario realizar un filtrado de la nube de puntos. Así, en la figura 7 se muestra la nube de puntos resultantes después de aplicar el filtrado, el filtrado se realiza de forma manual eligiendo un radio de una esfera que cubra al objeto, por lo tanto los puntos que se encuentran fuera de esta esfera se eliminan. Una vez que se tiene la versión filtrada de la nube de puntos se realiza la reconstrucción en 3D aplicando alguno de los dos métodos antes mencionados.

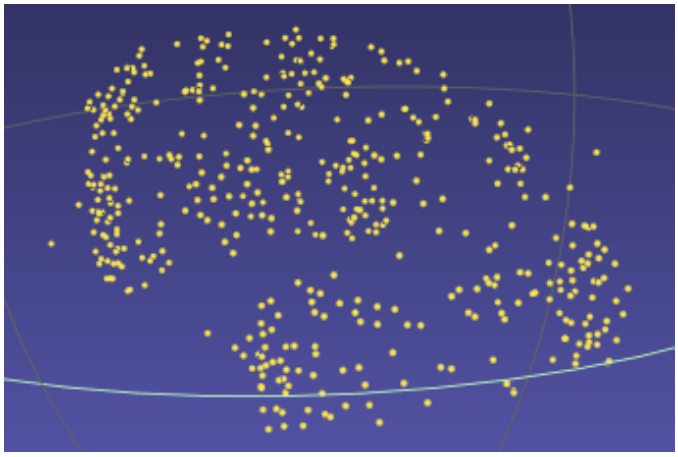

Figura 7: Nube de puntos del automóvil *Hatchback* obtenida con ORBSLAM2 filtrada.

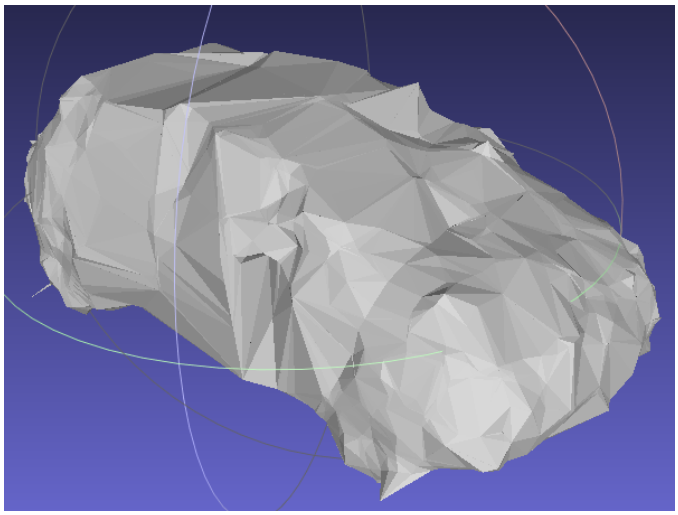

Figura 8: Reconstrucción de nube filtrada con Powercrust.

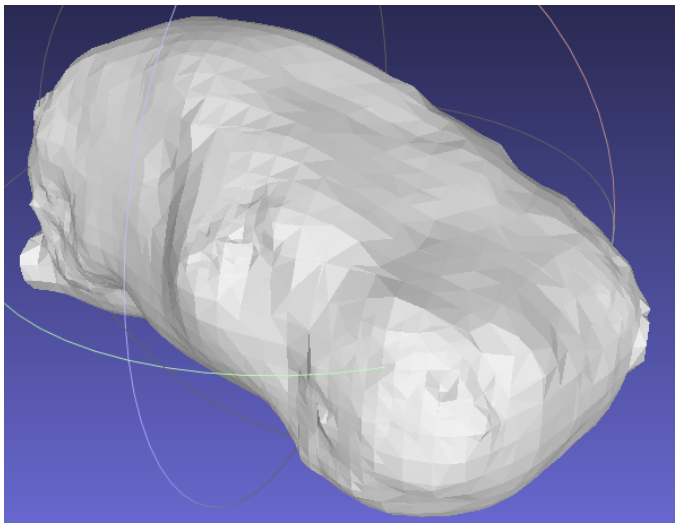

Figura 9: Reconstrucción de nube filtrada con Screened Poisson.

Las Figuras 8 y 9 muestran respectivamente las reconstrucciones resultantes de ambos métodos. Se puede observar que la reconstrucción obtenida con el método Powercrust tiene vértices más pronunciados en comparación con la que se obtiene con

el método de Poisson, además de que en esta última reconstrucción (Poisson) se aprecia mejor la forma del objeto original.

2.3. *Identificación de ventanas*

En esta prueba se busca que el dron vuele pasando a través de todas las ventanas del entorno lo más rápido posible.

La identificación de ventanas se llevó a cabo utilizando el simulador AirSim en Unreal, y se diseñó una red neuronal entrenada para identificar ventanas. Se creó una simulación y se añadieron las ventanas a utilizar en la prueba, lo cual se ilustra en la figura 11. Usando la API de AirSim es posible identificar objetos, así como obtener los rectángulos de encuadre de cada ventana dentro de la imagen que ve el dron. Esta información se almacena en formato *pascal voc* PASCAL2 (2022) que es un formato de anotación para detección de objetos y para el entrenamiento de redes neuronales. Esto se ilustra en la figura 10

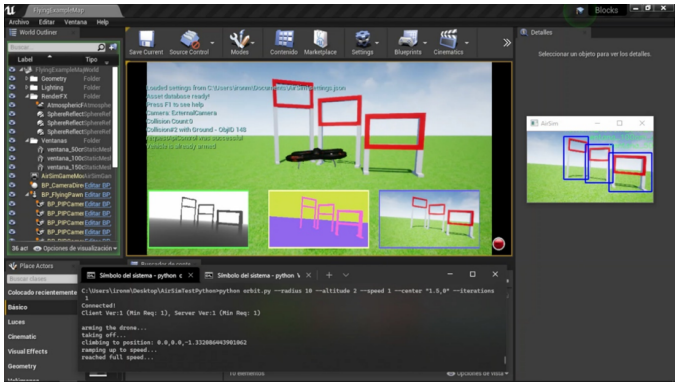

Figura 10: Identificación de ventanas usando la API de Unreal, con esta información se genera un conjunto de datos para entrenar una red neuronal.

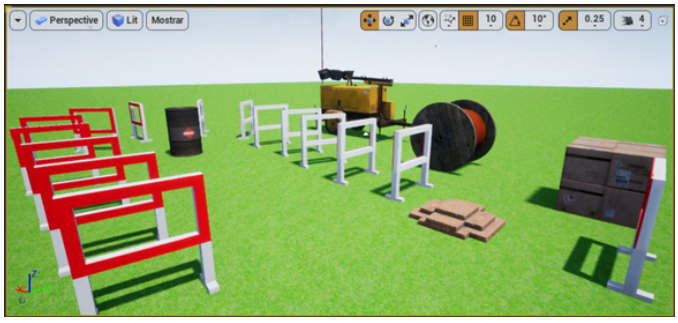

Figura 11: Entorno virtual creado para la recolección de datos en AirSim.

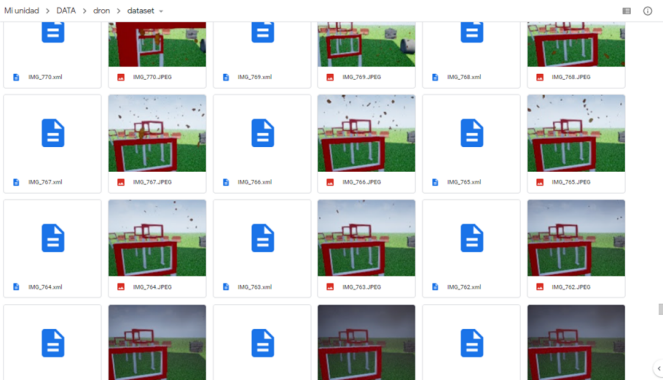

Figura 12: Datos recolectados para entrenamiento variando la iluminación y condiciones climáticas en el entorno.

2.3.1. *Recolección de datos*

Simulando distintas condiciones de iluminación e incluso de clima, con este simulador fue posible recolectar una gran cantidad de datos en poco tiempo, lo cual se ilustra en la figura 12. Los datos así obtenidos fueron usados más adelante para entrenar una red neuronal de tipo EfficientDet Tan et al. (2019) que está diseñada para la detección de objetos y propone varias optimizaciones clave para mejorar la eficiencia y está basada en una red piramidal bidireccional ponderada de características, la cuál puede ser ejecutada en una microcomputadora Raspberry Pi.

2.3.2. *Entrenamiento de la red*

La red neuronal antes mencionada fue entrenada en Google Colaboratory utilizando “EfficientDet-Lite”, una familia de modelos de detección de objetos optimizada para IOT y dispositivos móviles, la arquitectura deriva de EfficientDet Tan et al. (2019) propuesta por el equipo de investigación de Google, en particular se usó “EfficientDet-Lite0” la cual tiene una latencia promedio de 146 ms en una Raspberry Pi 4 usando únicamente 4 hilos del CPU.

2.3.3. *Pruebas y desempeño*

Se probó la capacidad de detección de la red neuronal con imágenes representativas de distintas condiciones de iluminación y de clima, obteniéndose predicciones bastante acertadas.

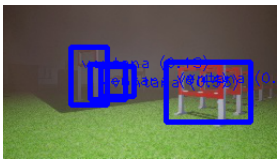

Figura 13: En las pruebas realizadas se observa que la red detecta más ventanas de las existentes debido al polvo y a la débil iluminación, sin embargo es posible retener únicamente las detecciones más grandes, es decir, las correspondientes a las ventanas más grandes.

2.4. *Vuelo en formación*

En esta prueba se involucran al menos dos drones, los cuales deben volar en conjunto y de manera sincronizada siguiendo la trayectoria del líder, la cual ha sido previamente definida.

Para poder realizar el vuelo sincronizado con dos drones dentro de Gazebo, se tuvo que modificar el paquete de drones *hector quadrotor* Johannes Meyer (2022) para que funcionara en ROS Noetic y además se pudiera ejecutar con dos drones. Una vez realizada esta modificación y habiendo logrado el despliegue de los dos drones, tal como se muestra en la figura 14, se procedió a elegir la estrategia de vuelo.

2.4.1. *Estrategia de vuelo*

Se eligió una estrategia en la cual existiera un dron líder y otro que se encargara de seguirlo, es decir, un seguidor.

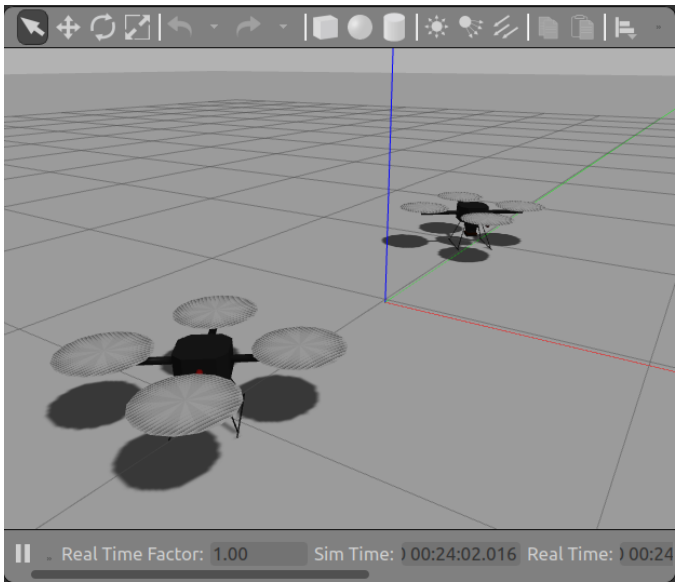

Figura 14: Hector Quadrotor adaptado a ROS Noetic con dos drones dentro de Gazebo.

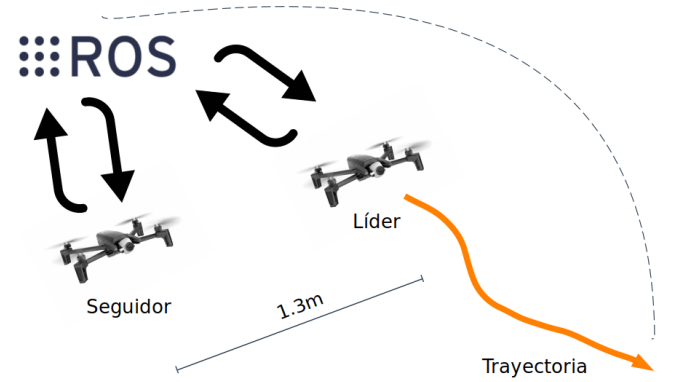

Figura 15: Representación de vuelo en formación con estrategia líder y seguidor.

En la estrategia del seguimiento se tuvo que acceder a la odometría de cada uno de los drones para poder aplicar un control proporcional. En cuanto a la posición y dirección de ambos drones se utilizó una matriz de rotación para poder rotar los marcos de referencia de cada dron a la dirección del punto que se quiera seguir, esto con la intención de que el dron solo se mueva hacia adelante o hacia los lados con el controlador proporcional. Una vez que se logra hacer el seguimiento del dron líder por parte del dron seguidor, se procede a generar las diferentes trayectorias que seguirá el dron líder y se utilizan las ecuaciones de control (1) - (4).

2.4.2. *Seguimiento de trayectorias*

Para realizar el seguimiento de trayectorias, primero se posiciona el dron seguidor a la derecha del dron líder a una distancia constante de 1.3 m, como se muestra en la figura 15, esta distancia permite hacer más seguro el vuelo de ambos drones y así evitar colisiones. En la figura 2 se muestran los marcos de referencia del dron dentro del simulador Gazebo. Posteriormente, ambos drones despegan simultáneamente y se elevan a la altitud deseada, luego, el dron líder se desplaza al punto inicial de la trayectoria seleccionada, en este caso se trata de una Lemniscata ilustrada en la figura 16, además, se ha definido también la trayectoria en espiral mostrada en la figura 17.

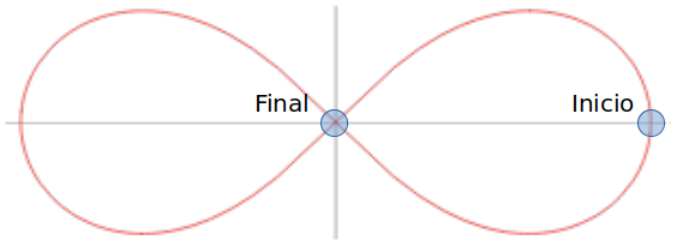

Figura 16: Representación de la primer trayectoria Lemniscata.

La trayectoria Lemniscata está definida por las siguientes funciones paramétricas:

$$x_d = \frac{a\cos(t)}{1+\sin^2(t)} \tag{5}$$

$$y_d = \frac{a\sin(t)\cos(t)}{1+\sin^2(t)} \tag{6}$$

Mientras que la altitud del dron líder será variable y estará definida por la siguiente función:

$$z_d = sin(t) \tag{7}$$

Por su parte, la trayectoria espiral está definida por las siguientes funciones paramétricas:

$$x_d = ae^{bt}\cos(t) \tag{8}$$

$$y_d = ae^{bt}\sin(t) \tag{9}$$

Las variables a y b cambian la amplitud de la trayectoria y t es el tiempo de vuelo. Para esta trayectoria la altitud de vuelo del dron líder será constante. Una vez concluido el vuelo sincronizado de ambos drones en cada una de las trayectoria elegidas, se realiza un aterrizaje en el cual las velocidades en los ejes x y z se definen como:

$$u_x = \sin(t) \tag{10}$$

$$u_z = -cte. \tag{11}$$

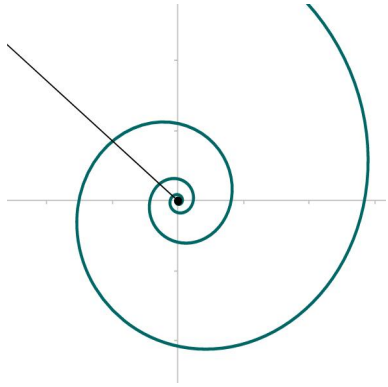

Figura 17: Representación de la segunda trayectoria Espiral.

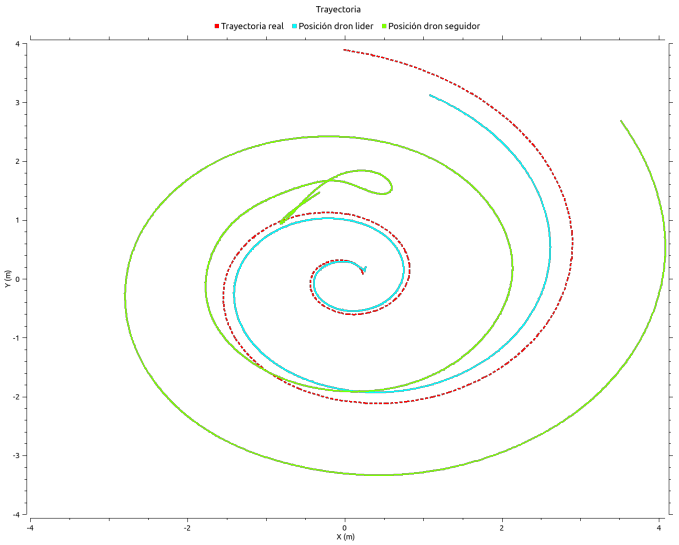

Figura 19: Resultados de la segunda trayectoria espiral.

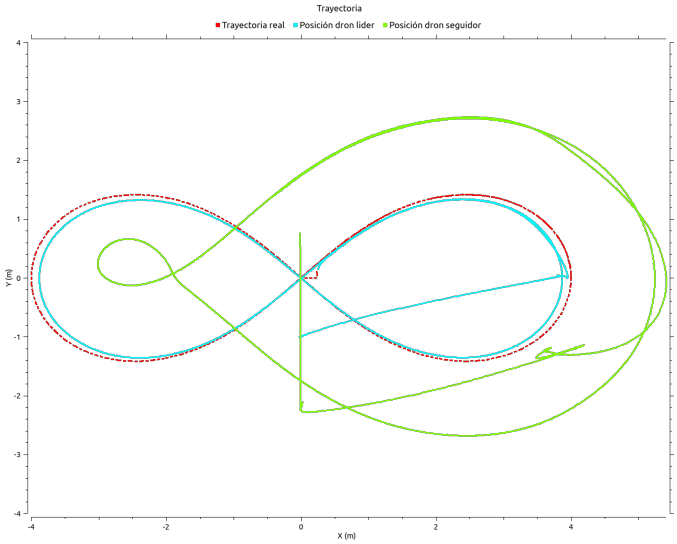

Figura 18: Resultados de la primer trayectoria Lemniscata.

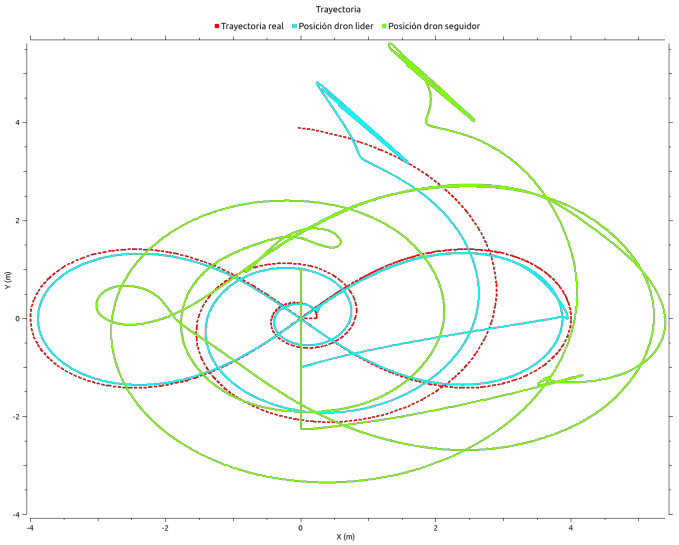

Figura 20: Resultado del recorrido completo.

2.4.3. Resultados de simulación

A continuación se muestran los resultados obtenidos cuando al dron líder se le asignan las trayectorias Lemniscata y espiral. Como se puede observar en la figura 18, el dron seguidor después de haberse posicionado a la derecha del dron líder, empieza a seguirlo al punto de inicio de la trayectoria Lemniscata, posteriormente en la figura 19 se muestra como comienza el seguimiento de la trayectoria espiral cercano al centro. Finalmente en la figura 20 se muestra el recorrido completo es decir, el posicionamiento a la derecha del dron líder, las dos trayectorias y el aterrizaje sinusoidal.

3. Conclusiones

Los simuladores son de gran ayuda para realizar pruebas que son complejas de llevar a cabo debido a la falta de espacio o drones. Es bueno poder realizar estas pruebas antes en un simulador, ya que la mayoría de los drones y sus refacciones tienen costos elevados. Una de las principales ventajas es que en los nuevos simuladores que están utilizando motores gráficos basados en videojuegos, es que podemos preparar los algoritmos para cambios de luz que se parezcan más al mundo real, esto brinda mayor confianza para llevar los algoritmos que se prueben en el simulador al mundo real. Como trabajo a futuro se continuará trabajando con el simulador AirSim para conocer mejor su integración con ROS o usarlo como un entorno de pruebas y después crear librerías para utilizarlas en cualquier entorno.

Referencias

ArduPilot (2022). Ardupilot - versatile, trusted, open. `https://ardupilot.org/`. Fecha de acceso: 2022-03-20.

Chen, L., Tang, W., John, N. W., Wan, T. R., and Zhang, J. J. (2018). Slam-based dense surface reconstruction in monocular minimally invasive surgery and its application to augmented reality. *Computer Methods and Programs in Biomedicine*, 158:135–146.

EpicGames (2022). The world's most open and advanced real-time 3d creation tool. `https://www.unrealengine.com/en-US/`. Fecha de acceso: 2022-03-20.

FIRA (2022). Fira air. `https://firaworldcup.org/category/leagues/fira-air/`. Fecha de acceso: 2022-03-20.

Gazebo (2022). Gazebo documentation. `https://gazebosim.org/docs`. Fecha de acceso: 2022-03-20.

IMAV (2022). International micro air vehicles, conferences and competitions. `http://www.imavs.org/`. Fecha de acceso: 2022-03-20.

Johannes Meyer, S. K. (2022). Hector quadrotor. `https://github.com/RAFALAMAO/hector-quadrotor-noetic`.

Kazhdan, M. M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13.

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and Stryk, O. v. (2012). Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International conference on simulation, modeling, and programming for autonomous robots*, pages 400–411. Springer.

MicrosoftResearch (2022). Welcome to airsim. `https://microsoft.github.io/AirSim/`. Fecha de acceso: 2022-03-20.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

Orozco-Soto, S. M., Ibarra-Zannatha, J. M., Malo-Tamayo, A. J., and Cureño-Ramirez, A. (2018). Active disturbance rejection control for uav hover using ros. In *2018 XX Congreso Mexicano de Robótica (COMRob)*, pages 1–5. IEEE.

Parrot (2022). What is parrot sphinx. `https://developer.parrot.com/docs/sphinx/`. Fecha de acceso: 2022-05-01.

PASCAL2 (2022). The pascal visual object classes homepage. `http://host.robots.ox.ac.uk/pascal/VOC/`. Fecha de acceso: 2022-03-20.

PX4 (2022). Open source autopilot for drone developers. `https://px4.io/`. Fecha de acceso: 2022-03-20.

Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., and Telea, A. (2016). 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, volume 35, pages 573–597. Wiley Online Library.

Tan, M., Pang, R., and Le, Q. V. (2019). Efficientdet: Scalable and efficient object detection. *CoRR*, abs/1911.09070.

TMR (2022). Drones autónomos. `https://www.femexrobotica.org/tmr2021/portfolio-item/drones-autonomos`. Fecha de acceso: 2022-03-20.

Zhang, M., Qin, H., Lan, M., Lin, J., Wang, S., Liu, K., Lin, F., and Chen, B. M. (2015). A high fidelity simulator for a quadrotor uav using ros and gazebo. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pages 002846–002851. IEEE.

Anexo 1

TMR: *Torneo Mexicano de Robótica* es un evento organizado por la Federación Mexicana de Robótica, A.C. en el cual se cuenta con la categoría de Drones Autónomos TMR (2022). Algunas de las pruebas propuestas en este torneo son: evasión y detección de obstáculos, vuelo entre ventanas, vuelo en formación, generación de mapas 3D de un entorno desconocido.

FIRA-Air: *Federation of International Sports Association* es la competición de fútbol de robots más antigua del mundo y ha ido creciendo y ahora tiene una categoría para drones que es FIRA-Air FIRA (2022). Algunas de las pruebas propuestas son vuelo entre ventanas, seguimiento de una ruta sobre una ciudad, detección de fuego en edificios y localización zonas de desastre y, sobre todo, de víctimas, en una ciudad a escala.

IMAV: *International Micro Air Vehicles* es un evento científico y tecnológico en el campo de la robótica aérea que se ha establecido como uno de los más importantes entre la comunidad internacional IMAV (2022). Algunas de las pruebas que ahí se han propuesto es la navegación de un dron dentro de un túnel con humo y detectar objetos dentro. También se tienen pruebas en donde el dron debe ser capaz de realizar un inventario de los objetos presentes en un conjunto de anaqueles. Además de estas pruebas en interiores se tienen pruebas en exteriores.