

Control de movimiento en un Drone mediante visión artificial

Roderick Aparicio	8-916-593
Isaac Bethancour	8-917-2263
Alan Castro	8-912-1890
Jesús De Gracia	8-1086-1646
Efrain Escobar	8-772-1697
Guillermo Espino	8-925-2235
Daniel González	8-907-1404
Allan Marín	EC-20-12127
Elvin Marín	EC-20-12126
David Morán	2-741-87

Lourdes Moreno	8-920-640
Kaiser Obaldía	8-898-703
Yeny Ortega	8-923-1263
Fermin Povaz	8-932-1661
Kevin Ruedas	8-910-1800
Krisbel Sanjur	2-738-644
Alexander Santana	8-933-1167
Stephanie Tejeira	8-924-2239
Victor Valenzuela	8-931-2246
Ana Villarreal	8-917-2049

Abstract— In this article we will try to create a controller that, through the webcam, can detect the objects that are around it and can operate in said environment.

For the development of this laboratory, the knowledge acquired on the operation of the nodes in ros (ros publisher, ros subscriber, among other features) was used to combine their functions together with the gazebo simulator and the ardrone autonomy. Subsequently, a controller for a webcam was created using the c++ language, where artificial vision techniques were applied together with OpenCV, to achieve that through the camera it recognized a circular object, and once it was detected that through this the ardrone in the simulator could move in different ways.

Resumen

En este artículo intentaremos crear un controlador que a través de la cámara web, pueda detectar los objetos que se encuentran a su alrededor y se pueda desenvolver en dicho entorno.

Para el desarrollo de este laboratorio se utilizaron los conocimientos adquiridos sobre los funcionamientos de los nodos en ros (ros publisher, ros subscriber, entre otras características), para combinar sus funciones junto al simulador gazebo y el ardrone autonomy. Posteriormente se creó un controlador para una webcam utilizando el lenguaje c++, en donde se aplicaron técnicas de visión artificial junto a OpenCV, para lograr que a través de la cámara reconociera un objeto circular, y una vez detectada que por medio de esta el ardrone en el simulador pudiera moverse de diferentes formas.

Palabras Claves

Gazebo, Controlador, Ubuntu, Ros Kinect, Inteligencia Artificial, Detectar.

Introducción

Los drones se están volviendo cada vez más populares en estos días. Hay múltiples tareas e ideas que se pueden implementar. Uno de ellos es la idea que se presentará en el documento.

Se pretende mostrar una simulación de un AR drone para la detección de objetos mediante una cámara web, utilizando paquetes de ROS Kinect que es la versión necesaria para la instalación del drone, el uso del mismo simulador para implementar el gazebo y pueda ser controlado por medio de los movimientos.

Objetivo

Crear un controlador que por medio de una cámara web pueda detectar el objeto circular para indicarle al Drone AR qué movimientos debe realizar como por ejemplo: moverse hacia arriba, hacia abajo, izquierda, derecha o retroceder.

Materiales y Métodos

A. Visión Artificial: son técnicas que permiten por medio de métodos de procesamiento, de análisis y comprensión de imágenes en el mundo real, lograr que un computador pueda reconocer que es lo que hay en esas imágenes. Estas técnicas a menudo utilizan diferentes algoritmos matemáticos para poder que la computadora pueda extraer información de dichas imágenes.

B. Gazebo Simulator

Gazebo es un simulador de múltiples robots en un mundo abierto al aire libre. es capaz de simular una serie de robots, sensores y objetos en un mundo tridimensional. Esto puede generar retroalimentación de forma realista para sensores cuando se interacciona con objetos del mapa digital.

Gazebo, tiene las siguientes características técnicas:

- ❖ Simulación dinámica: acceso a múltiples motores de física de alto rendimiento.
- ❖ Gráficos 3D avanzados: usando OGRE puede generar entornos y renderizar formas realistas, texturas, luces, sombras, etc.
- ❖ Plugins: para que los desarrolladores puedan personalizar robots, sensores y entornos de control gracias a su API.

Existen requisitos específicos del sistema:

- Sistema operativo Linux (Ubuntu Trusty o posterior).
- Una GPU dedicada (las tarjetas Nvidia funcionan bien).
- CPU Intel Core i5, i7
- 50 GB de espacio libre en disco.
- 16 GB de Ram.

C. Ros

Robot Operating System (ROS) es una colección de marcos de software para desarrollo de software robot. ROS proporciona servicios de sistema operativo estándar, como abstracción de hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y administración de paquetes.

D. OpenCV

Es una biblioteca de código abierto que incluye varios cientos de algoritmos de visión por computadora.

La biblioteca OpenCV se usa ampliamente debido a su amplia cobertura de las tareas de visión por computadora y su disponibilidad para involucrarla en varios proyectos, incluido el aprendizaje profundo. Normalmente, OpenCV se utiliza con C++.

E. Parrot

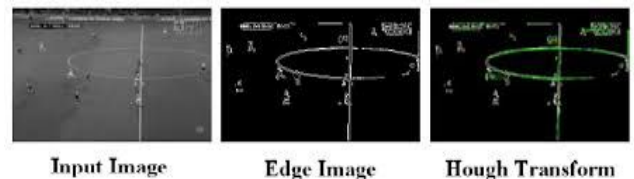
Parrot AR. Drone es un quadcopter volador a control remoto construido por la compañía francesa Parrot. El dron está diseñado para ser controlado por sistemas operativos móviles o tabletas como el iOS o Android compatible dentro de sus respectivas aplicaciones o el software no oficial disponible para dispositivos Windows Phone, Samsung BADA y Symbian.

F. Desenfoque Gaussiano

Esta es una operación que se puede hacer mediante OpenCV, en el cual una imagen se convoluciona con filtros gaussianos. El filtro gaussiano es un filtro de bajo paso que elimina componentes de alta frecuencia que se pueden llegar a producir. Esto aplicado en el laboratorio, nos permite por medio de la cámara conocer qué círculo es el correcto en la detección del objeto.

G. Transformación Hough

La transformación Hough es una técnica de extracción de características utilizada en el análisis de imágenes, la visión por ordenador y el procesamiento de imágenes digitales. El propósito de la técnica es encontrar casos de objetos dentro de una determinada clase de formas por un procedimiento de votación.



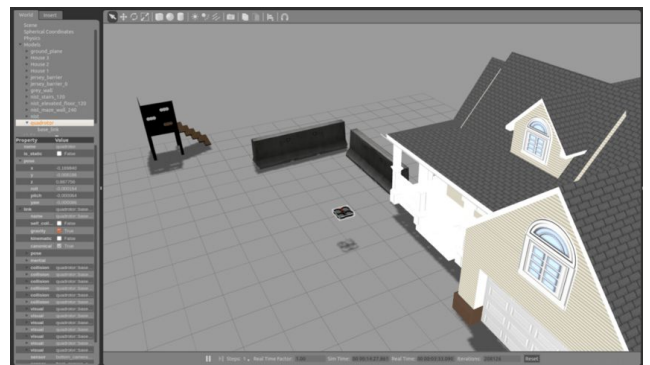
Ejemplo en la detección de círculos.

H. CvBridge

CvBridge es una biblioteca ROS que proporciona una interfaz entre ROS y OpenCV. CvBridge se puede encontrar en el paquete cv_bridge en la pila vision_opencv. Todo el análisis de imágenes se basa en convertir imágenes ROS en imágenes OpenCV.

I. Tum Simulator

Tum Simulator es una implementación de Gazebo Simulator y Paquete de autonomía Ardrone. La instalación de este paquete permite ejecutar el dron en un mundo artificial en Gazebo Simulator.



Simulación Tum en un dron virtual.

J. Geometry_msgs

Geometry_msgs proporciona mensajes para primitivas geométricas comunes como puntos, vectores y poses. Estas primitivas están diseñadas para proporcionar un tipo de datos común y facilitar la interoperabilidad en todo el sistema.

K. Algoritmo del Controlador

Para que el drone pueda moverse se implementó el siguiente algoritmo:

- ❖ Si el objeto circular se encuentra muy cerca de la cámara (el drone irá hacia atrás), si está a una distancia más lejos pero dentro del rango de la cámara el drone se moverá hacia adelante.
- ❖ Si la distancia es apropiada (tomando en cuenta los ajustes que se le dieron al código) analizando la posición vertical, el drone subirá o bajará.
- ❖ Si la posición o eje vertical es la indicada, se procederá a analizar el eje horizontal, una vez analizado, se le dará la orden al drone de moverse hacia la derecha o izquierda.

Metodología

1. El primer paso para llevar a cabo este proyecto, es tener instalada la versión ubuntu 16.04. Y posteriormente descargar el distro correcto de ros, en nuestro caso fue el Ros Kinetic.

2. Luego, un paso fundamental antes de proceder a ejecutar el controlador de la webcam, es llamar a los diferentes paquetes(ardrone autonomy, cv_bridge, roscpp, geometry_msgs, etc), esto se debe producir dentro del archivo CMakeLists.txt. Este archivo debe estar localizado en la carpeta ardrone_autonomy.

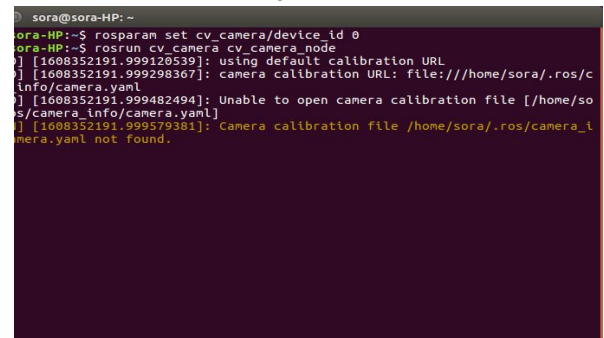
3. Dentro de la carpeta mencionada anteriormente, nos ubicamos en la carpeta drone, dentro de ella nos ubicamos en la carpeta src y colocamos nuestro script llamado main.cpp. Este script contiene el algoritmo del controlador de la cámara, como también las librerías requeridas de OpenCV y las funciones que permiten el reconocimiento del objeto de prueba.

4. Luego de haber añadido los archivos CMakeLists.txt y main.cpp. Procedemos a abrir 4 terminales en ubuntu, en el cual ejecutarán las siguientes funciones:

- ❑ **Roscore:** en la primera terminal ejecutaremos roscore, que permitirá enlazar los nodos necesarios de los diferentes componentes del simulador, y de la máquina.
- ❑ **Rosnode de la cámara:** en esta terminal se prepara el controlador de la cámara por medio de Ros y OpenCV. Este último es importante ya que permite detectar la imagen y sus diferentes contrastes de colores de la cámara del computador. Esto lo podemos observar en la **Figura 1**.
- ❑ **Roslaunch en gazebo:** por medio de esta

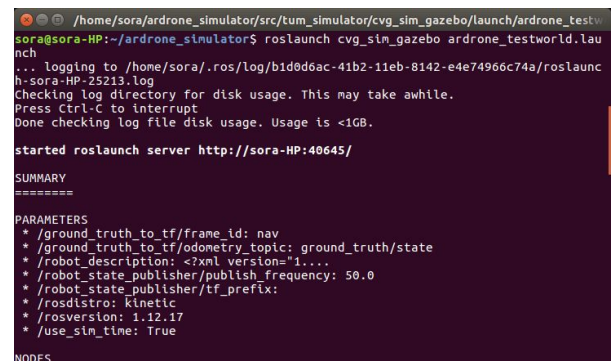
terminal nos permitirá cargar el ambiente en el cual se hará la prueba del controlador de la cámara web, y el ardrone que se desenvolverá en el mundo testworld, como se muestra en la **Figura 2** y **Figura 3**.

- ❑ **Rosrun del script main:** Lleva a cabo varias acciones que unen los pasos previos como: suscribirse al nodo de la cámara publicado para procesar la información que se captura, iniciar el vuelo del drone, controlar los movimientos del drone mediante el radio y dirección del objeto.



```
sora@sora-HP: ~  
sora-HP:~$ roscpp set cv_camera/device_id 0  
sora-HP:~$ roscpp cv_camera_node  
[1608352191.999120539]: using default calibration URL  
[1608352191.999298367]: camera calibration URL: file:///home/sora/.ros/c  
info/camera.yaml  
[1608352191.999482494]: Unable to open camera calibration file [/home/s  
s/camera_info/camera.yaml]  
[1608352191.999579381]: Camera calibration file /home/sora/.ros/camera_l  
amera.yaml not found.
```

Figura 1. Ros Node de cámara web



```
/home/sora/ardrone_simulator/src/tum_simulator/cvg_sim_gazebo/launch/ardrone_test  
sora@sora-HP:~/ardrone_simulator$ roslaunch cvg_sim_gazebo ardrone_testworld.lau  
nch  
... logging to /home/sora/.ros/log/b1d0d6ac-41b2-11eb-8142-e4e74966c74a/roslauch  
h-sora-HP-25213.log  
Checking log directory for disk usage. This may take awhile.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://sora-HP:40645/  
  
SUMMARY  
=====  
PARAMETERS  
* /ground_truth_to_tf/frame_id: nav  
* /ground_truth_to_tf/odometry_topic: ground_truth/state  
* /robot_description: <?xml version="1.0...  
* /robot_state_publisher/publish_frequency: 50.0  
* /robot_state_publisher/tf_prefix:  
* /roscpp: kinetic  
* /rosversion: 1.12.17  
* /use_sim_time: True  
  
NODES
```

Figura 2. Lanzamiento de Gazebo AR Drone

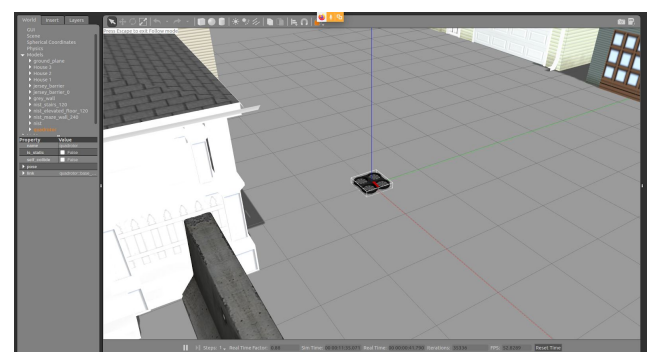


Figura 3. Ejecución del Gazebo

Resultados

Como se observa en la **Figura 4** se logra realizar la detección del objeto mediante la aplicación de la transformación Hough con openCV. Esto nos permite controlar el drone basado el movimiento y dirección del objeto.

Al mover el objeto en dirección vertical(arriba y abajo) y horizontales(izquierda y derecha) el drone

realiza los movimientos satisfactoriamente, sin embargo, al ordenarle las instrucciones al dron de moverse hacia adelante la detección del objeto se distorsiona generando múltiples círculos en otros objetos y la instrucción de moverse hacia atrás se perdía la detección del objeto; afectando en la ejecución de los movimientos del dron.

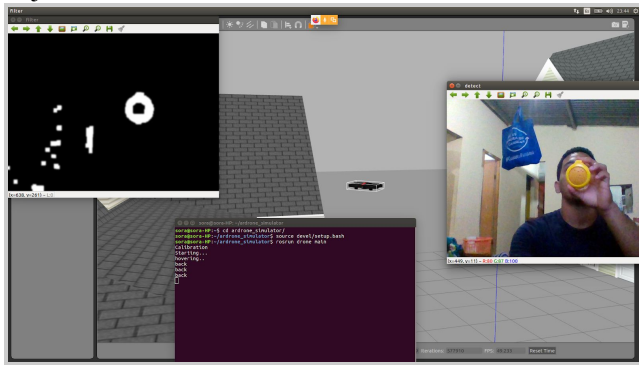


Figura 4. Reconocimiento de objeto

Conclusiones

En conclusión, las condiciones del entorno donde se realizan las pruebas de detección de un objeto para controlar acciones de un dron, deben estar reguladas para una fases iniciales, ya que si existe un sobreexceso de luz sobre el objeto, habrá una saturación en el color afectando a la captura adecuada de información, en el caso contrario de carencia de luz, la cámara web se le dificulta la detección.

Por otra parte, se realizaron pruebas de este proyecto mediante el uso de máquinas virtuales en diferentes computadoras buscando la posibilidad de utilizar otras cámaras web, lo cual no se consiguió ningún resultado ya que al lanzar el simulador gazebo mediante la instrucción de roslaunch contenía muchos errores en diversos archivos internos y no aparecía el dron en el mundo para poder realizar las pruebas.

También, para el correcto funcionamiento del simulador de gazebo con **ROS**, se requiere instalar paquetes como *hector* y *ardrone-autonomy*, ya que sin estos al lanzar el simulador junto con ros, el dron tiene un comportamiento no correcto, subiendo indefinidamente.

Por lo tanto, como trabajo futuro, se requeriría llevar el proyecto a versiones posteriores de ubuntu y ros, ya que el ubuntu 16.04 presenta ciertos problemas de compatibilidad con hardware de varios actuales.

Bibliografía

- alpinmaarif. (2020). *Webots-DJI-Mavic-2-Pro-PID*. Obtenido de Controlador. <https://github.com/alpinmaarif/Webots-DJI-Mavic-2-Pro-PID-Controller>
- Anwar, T. (2020, Diciembre 14). *Localización. Clasificación con localización.* <https://www.learnopencv.com/>
- Ciberbótica. (2020). *Obtenido de Cámara*. Manual de referencia de Webots. https://www.cyberbotics.com/doc/reference/camera#wb_camera_has_recognition
- Doxygen. (2020, Diciembre 18). *OpenCV*. OpenCV. <https://docs.opencv.org/master/d1/dfb/intro.html>
- illusion Soft. (2020, Abril 22). *Obtenido de Camera in Webots Simulator*. Visual Object Recognition. https://www.youtube.com/watch?v=JHAWZmfY8zA&list=PLt69C9MnPchkLuNNc4q9SeMFA96_v4THJ&index=1
- Lewis, J. (2014, Noviembre 15). *Color Tracking Drone*. ColorTracking-ARDrone2.0-Python. <https://github.com/JJLewis/ColorTracking-ARDrone2.0-Python>
- mmmmmmwei. (2019, Julio 14). *Drone Target*. target_tracking_drone. https://github.com/mmmmmmmwei/target_tracking_drone