Polígonos Coloreados Simples como Instrucciones de Vuelo

Tomás Oelckers y Tarik Sáez

Ingeniería Eléctrica, Facultad de Ingeniería y Ciencias Aplicadas Universidad de los Andes, Santiago, Chile

toelckers@miuandes.cl - tsaez@miuandes.cl

Introducción

Programamos un dron DJI Tello en Python 3 utilizando la librería OpenCV para transmitir instrucciones de vuelos a través de figuras geométricas simples coloreadas. Para la detección de figuras se utilizaron funciones predefinidas en la librería, sin embargo nuestro proyecto se caracteriza por tener un rango de colores único, el cual es capaz de adaptar el vuelo para varios escenarios con diferentes luces y sombras. Adicionalmente combinamos más de una forma de reconocimiento de figuras para mejorar su consistencia en la detección.

Dron DJI Tello

El mini dron DJI Tello cuenta con una cámara frontal de 720p (HD) y conexión wifi 2.4 GHz.

Especificaciones:

- Peso: 80 gramos aproximadamente.
- Dimensiones: 98x92.5x41 mm.
- Distancia máxima de vuelo: 100 metros.
- Velocidad máxima: 8 m/s.
- Tiempo de Vuelo Máximo: 13 minutos.
- Altura máxima de vuelo: 30 metros.
- FOV: 82.6º.
- Video: 720p @ 30 FPS, formato MP4.
- Lenguaje: Python
- Librería: djitellopy.

Condiciones Ambientales

Las condiciones necesarias son:

1. Evitar los colores rojo, azul, verde y amarillo en las paredes, suelo y techo.
2. Evitar ambientes muy iluminados o muy oscuros. Las fuentes de luz no deben impactar directamente a ninguna figura.
3. Evitar espacios no cerrados para prevenir corrientes de viento.

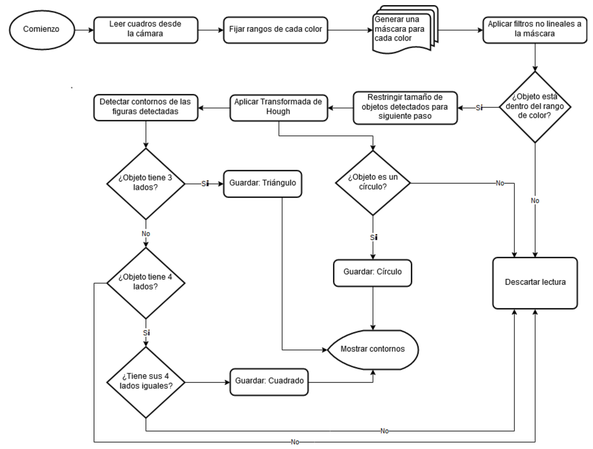
Algoritmo

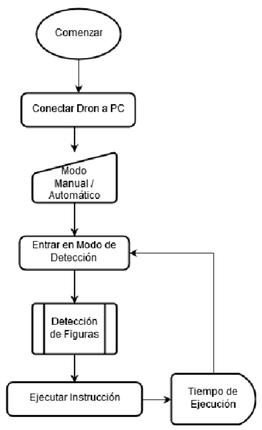
El algoritmo puede ser dividido en dos partes principales:

1. Detección: Se utiliza una máscara que filtra figuras según su tamaño, forma geométrica (dada por la cantidad de lados y su razón de proporción entre ellos) y su color, el cual está definido por rangos en el espacio HSV. De esta forma, si uno muestra un cuadrado de un color que no corresponde o bien una figura de un color utilizado pero que no es ninguna de las figuras predefinidas, se descarta su detección.

   - *findContours*: A través de esta función, contamos la cantidad de lados que tiene una figura. Si tiene 3 lados es un triángulo; si tiene 4 lados y estos son más o menos del mismo tamaño, es un cuadrado. Si no cumple con ninguno de las condiciones anteriores, se descarta la lectura.
   - *HoughCircles*: Utilizamos la función HoughCircles para detectar círculos según una búsqueda de contornos (CannyEdge) y un centro tentativo. Si el círculo detectado esta fuera de un rango de tamaños predefinido, se descarta la lectura.

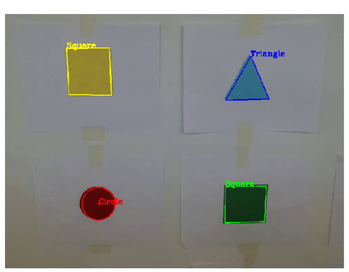
2. Ejecución de Instrucciones: En esta parte se incorpora al dron y su cámara; a partir de la detección de figuras, se definen instrucciones para cada figura, es decir, una figura representa una acción que el dron debe ejecutar y que ha sido preprogramada.

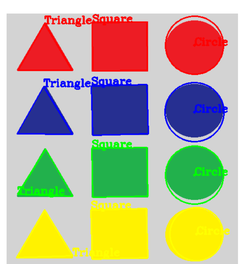




Resultados

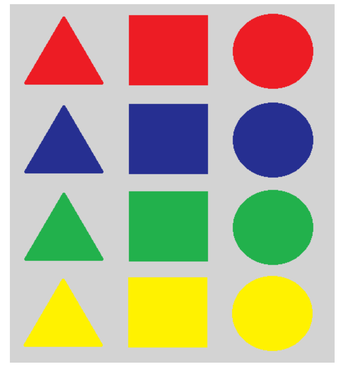
A partir de pruebas realizadas tanto con imágenes computarizadas (ambiente controlado ideal) e imágenes obtenidas directamente desde la cámara del dron (ambiente semi-controlado), nuestro algoritmo es eficaz a la hora de detectar las figuras deseadas, tal como se puede apreciar en los ejemplos presentados a continuación.





Señales Utilizadas

Las señales establecidas para la codificación de instrucciones fueron triángulos, cuadrados y círculos de colores rojo, azul oscuro, verde y amarillo. En total combinan hasta 12 instrucciones posibles de vuelo; suficientes para describir las funciones necesarias para recorrer un circuito visual en un ambiente semi-controlado.



Conclusiones

A partir de los resultados obtenidos de nuestros experimentos, podemos concluir que el algoritmo propuesto funciona para ejecutar una ruta de vuelo basada en instrucciones visuales (figuras). Sin embargo, cabe destacar también que el ambiente en el cual es ejecutado debe ser semi-controlado para evitar detectar falsos positivos y negativos.

Además extendemos que si se implementara una red neuronal para permitir el correcto funcionamiento en ambientes no controlados y para figuras más complejas.

Referencias

1. "DJI Tello Py", *Python Software Foundation*, `https://pypi.org/project/djitellopy/`.
2. "Hough Circle Transform", *OpenCV*, `https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html`.
3. "Structural Analysis and Shape Descriptors", *OpenCV*, `https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html`