Navegación autónoma de un vehículo aéreo por referencia visual

*Adrián Chouza, Raul Hernandez, Jose Luis Jimenez, Ulises Orozco-Rosas**

CETYS Universidad, Centro de Innovación y Diseño (CEID), Av. CETYS Universidad No. 4. El Lago, C.P. 22210, Tijuana B.C., México
[adrian.chouza, raul.hernandez, joseluis.jimenez]@cetys.edu.mx, ulises.orozco@cetys.mx

Resumen

Los vehículos aéreos no tripulados (UAV) han tenido un auge durante los últimos años con diversas aplicaciones como envío de correo, apoyo en desastres naturales, sensado de información meteorológica, entre muchas otras. Es por ello por lo que en este trabajo se explora la misión de que un cuadricóptero, a partir de referencias visuales, pueda navegar a través de un marco de referencia representado por un cuadro rojo, de manera totalmente autónoma. Entre las técnicas utilizadas se encuentre el procesamiento de imagen a través del método de Otsu, Canny y difuminación Gaussiana, además de métodos de control como el controlador proporcional-integral-derivativo (PID). El cuadricóptero utilizado es el Parrot Bebop 1, el cual contiene una variedad de sensores como acelerómetro, cámara frontal, giroscopio, ultrasónico y magnetómetro, los cuales permiten a este estimar su posición actual, información que es transmitida a un sistema de procesamiento que después de identificar la distancia al marco de referencia, es procesada para determinar la velocidad adecuada que es enviada de vuelta al vehículo para realizar los movimientos correspondientes. Utilizando estas técnicas se logra el objetivo inicial de forma satisfactoria, sin embargo, se discuten algunos problemas del algoritmo tal como no detectar de forma completa el marco de referencia, errores en la odometría del cuadricóptero y problemas de iluminación dado el contexto de prueba. Finalmente se ofrecen las conclusiones del proyecto junto con una breve descripción del trabajo a realizar en un futuro.

Palabras clave—Controlador PID, Navegación autónoma, Procesamiento de imagen, ROS, Vehículos aéreos.

Abstract

Unmanned aerial vehicles (UAVs) have suffered an increase in popularity in recent years with various applications such as mail delivery, natural disaster support, meteorological information detection, among many others. That is why this paper explores the mission that a quadcopter, given visual references, can fly through a reference frame described by a red frame, in a totally autonomous way. Among the techniques used are image processing through the method of Otsu, Canny, and Gaussian blur, in addition to control methods such as the proportional-integral-derivative (PID) controller. The quadcopter used is the Parrot Bebop 1, which contains a variety of sensors such as accelerometer, front camera, gyroscope, ultrasonic and magnetometer. That helps to estimate the vehicle's current position, information that is transmitted to a processing system that after calculating the distance to the reference frame, that is processed to determine the appropriate speed that is sent back to the quadcopter to perform the corresponding movements. Using these techniques, the initial objective is satisfactorily achieved; however, some problems of the algorithm are discussed, such as not fully detecting the reference frame, errors in the quadcopter odometry module and illumination problems are given the test context. Finally, the conclusions of the project are offered along with a brief description of the work to be done in the future.

Keywords— *Aerial vehicles, Autonomous navigation, Image processing, PID Control, ROS.*

1. INTRODUCCIÓN

Ante la ausencia de un piloto en un vehículo aéreo no tripulado (UAV), o dron, este puede ser controlado mediante un operador humano localizado en tierra, o bien, poseer sensores y algoritmos los cuales le permitan realizar movimientos de manera autónoma.

El desarrollo de sistemas autónomos ha tenido un auge en los últimos años. En el caso de los drones, estos sistemas han sido desarrollados para diferentes propósitos como el desarrollo de sistemas autónomos de envío de correo [1], sistemas de grabado multimedia en eventos deportivos, sistemas de apoyo en desastres naturales, sistemas de sensado en investigación meteorológica [2], entre muchos otros.

* Autor para la correspondencia (*Corresponding author*)

En el presente trabajo se pretende solventar el problema de movimiento autónomo de un cuadricóptero mediante la identificación de un marco de referencia, representado por un marco rojo con medidas de 1m por lado, colocado a una altura de 50cm (de esta forma, el centro del marco se encuentra a 1m de altura).

La misión se describe de la siguiente forma: de manera totalmente autónoma el cuadricóptero debe despegar con su visión perpendicular al plano formado por el marco de referencia, no necesariamente frente al centro de este. Posteriormente deberá realizar las maniobras adecuadas para atravesar el marco por su centro y al terminar, aterrizar del otro lado de este.

Para lograr la misión de forma satisfactoria, en la Sección 2 se presentan bases teóricas necesarias para comprender la

propuesta de la Sección 3, donde se describe el algoritmo separado en dos partes: el control del cuadricóptero a partir de un controlador PID y el procesamiento de imagen para la detección del marco de referencia.

2. FUNDAMENTOS

En esta sección se ofrece un marco teórico de los distintos conocimientos y herramientas utilizados en la propuesta de este artículo. Los temas cubiertos abarcan desde terminología básica del cuadricóptero, los métodos de control y de procesamiento de imagen utilizados.

2.1 Navegación de un cuadricóptero

Un cuadricóptero es capaz de moverse en un entorno tridimensional gracias a sus 6 grados de libertad [3]. De estos 6 grados, 3 corresponden a movimientos traslacionales a través de los ejes *x*, *y* y *z*, y los otros 3 corresponden a movimientos rotacionales a través de los mismos ejes. Cada uno de los movimientos rotacionales posee un nombre especial, los cuales son:

1. *Roll*: Esta rotación se encarga del alabeo del cuadricóptero, siendo el eje de rotación el que va del frente hacia la parte trasera del cuadricóptero, permitiendo que este se pueda mover de derecha a izquierda.
2. *Pitch*: Esta rotación le permite al cuadricóptero realizar movimientos hacia adelante y hacia atrás, siendo el eje de rotación el que va del lado derecho del cuadricóptero al izquierdo.
3. *Yaw*: Esta rotación le permite a un cuadricóptero realizar rotaciones en sentido de las manecillas del reloj y opuesto, por lo cual esta rotación se realiza a través del eje *y* en el espacio.

2.2 Controlador PID

Un controlador proporcional-integral-derivativo (PID) [4] engloba una ecuación integro diferencial que permite ajustar los valores que se deben ingresar a un actuador para controlar un sistema. La ecuación es la que sigue:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad [1]$$

El nombre que se le da al control depende de los valores que se le otorgue a cada una de las constantes K_p, K_i y K_d. En caso de que sean *1*, *0*, *0*, respectivamente, entonces solo es un controlador P. De ser 1,1,0, respectivamente, entonces es un controlador PI. El primer término de la función representa la parte proporcional. La función $e(t)$ calcula el error actual del sistema dado en un tiempo t. Por lo tanto, el término aporta un factor proporcional al error actual del sistema, modificado por la constante K_p.

El segundo término es la parte integral. En este se acumula el error del sistema a lo largo del tiempo. El objetivo que tiene es reducir la oscilación de un sistema causado por el constante cambio en la parte proporcional. Sin embargo, utilizado sin cuidado, el error que acumule puede ser muy grande y causar efectos no deseados.

Finalmente, el último término es el factor derivativo. Para evitar el fenómeno de *overshoot* (que el sistema sobrepase su objetivo), se calcula la derivada del error (la cual debería de ser negativa si el error disminuye), la cual reduce el valor $u(t)$ final, suavizando el movimiento del sistema.

2.3 Binarización, filtrado y detección de ejes

Para procesar las imágenes para su uso, primero se deben asegurar algunas variables. Se conoce que la computadora es capaz de procesar colores de algunas maneras como RGB utilizando tres valores que varían de 0 a 255. Sin embargo, en cuestión de vídeo en vivo es mejor usar el formato HSV (*Hue*, *Saturation*, *Value*) [5].

2.3.1 Método de Otsu

El método de Otsu [6] consiste en reducir una imagen de escala de grises a una imagen binaria, para esto se tienen dos clases de píxeles que se consideran como de fondo y de frente y Otsu calcula la probabilidad de que un píxel pertenezca a una clase discriminando así a los píxeles en blancos y negros.

Su mayor ventaja recae en que puede hacer su binarización en grupos llamados *Clusters* lo cual permite un filtrado más rápido. Si bien, este método tiene variaciones que optimizan su filtrado estas son computacionalmente más costas y no se aplicaron para mantener un nivel de respuesta óptimo en el cuadricóptero.

2.3.2 Método de Canny

En cuestión de detección de ejes existe el método Canny [7] el cual reconoce las aristas de la figura. Este funciona mediante tres principios: la detección para maximizar la probabilidad de detectar ejes verdaderos y minimizar la de no detectar ejes, mientras que la localización consiste en considerar que los ejes reales y los detectados deben de estar lo más cerca posible. Finalmente, un eje real solo debería producir un eje detectado.

2.3.3 Difuminación Gaussiana

Una difuminación Gaussiana [8] consiste en un método de difuminación basado en la función Gaussiana, con el propósito de limpiar la imagen y eliminar ruido. Ya que una imagen es de dos dimensiones se utiliza una función Gaussiana de las mismas dimensiones, definida por la fórmula:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \quad [2]$$

donde σ indica que tan amplios serán las funciones kernel, mientras que x y y definen la posición de un píxel dentro de la imagen.

3. PROPUESTA

Para lograr el objetivo de este artículo, se desarrolló una propuesta formada de distintos módulos, principalmente, un

sistema de control y de procesamiento de imagen. En esta sección se describen los detalles de implementación además del hardware y software utilizado.

Fig. 1. Cuadricóptero Parrot Bebop 1.

3.1 Características del cuadricóptero

El cuadricóptero por utilizar en esta propuesta se llama Parrot Bebop [9], elegido por su accesibilidad de precio y los sensores integrados para facilitar el vuelo. Se puede apreciar una imagen de este en la Fig. 1, donde el protector se encuentra ensamblado.

El cuadricóptero Parrot Bebop 1 cuenta con conectividad a dispositivos por medio de una antena Wi-Fi [10] y una cámara frontal con lentes *Fisheye* de 180°, 14 megapixeles y definición de video de 1920x1080p a 30 cuadros por segundos. Esta resolución baja a 856x480 mientras envía información por medio de Wi-Fi. Su batería ofrece un tiempo de vuelo aproximado de 12 minutos y cuenta con 6 sensores distintos:

- Magnetómetro de 3 ejes
- Giroscopio de 3 ejes
- Acelerómetro de 3 ejes
- Sensor de flujo óptico
- Ultrasónico
- Presión

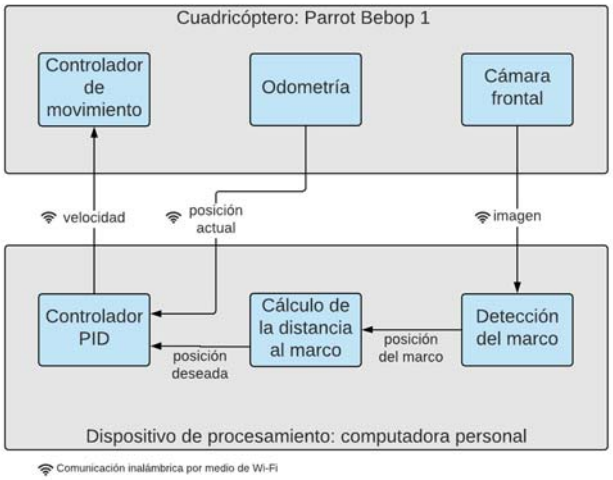
Finalmente, tiene dimensiones 28x32x3.6 cm sin el protector y 33x38x3.6 cm con protector, además de un peso de 380g y 400g, respectivamente.

3.2 ROS

La programación y comunicación con el cuadricóptero se llevó a cabo con ROS [11] (*Robot Operating System*), el cual ofrece una gran variedad de herramientas para comunicar varios dispositivos entre sí y al mismo tiempo tiene una gran base de usuarios, ocasionando que el gran soporte de la comunidad sea una de sus grandes ventajas.

Entre el gran soporte que ofrece la comunidad de ROS se encuentra el paquete *bebop autonomy* [12], que ofrece una interfaz sencilla para la comunicación entre el hardware del cuadricóptero Parrot Bebop 1 y 2, junto con una amplia documentación para realizar los movimientos deseados. Al ser una biblioteca de alto nivel, elimina la necesidad de pensar en la física detrás de la potencia de cada rotor del

Fig. 2. Sistema de navegación autónoma implementado en el vehículo aéreo.



cuadricóptero y requiere solo la velocidad a la que se desea mover en cada uno de sus grados de libertad.

3.3 Descripción del sistema

Para que el cuadricóptero sea capaz de identificar su entorno y moverse alrededor de él, es necesario que se sitúe en un sistema de ejes coordenados global y pueda moverse de un punto *A* a un punto *B*. Esta se logra a partir de dos partes: el control PID y los mensajes de odometría del cuadricóptero. Este publica constantemente mensajes a la red inalámbrica sobre su posición actual utilizando como referencia su primera posición de despegue. Estos datos pueden ser procesados junto con los controladores PID para determinar la velocidad que debe tener el cuadricóptero para llegar a su destino.

De la misma forma, en cuestión de conocer su entorno el cuadricóptero debe de poder reducir la imagen y concentrarse en lo que se desea ver. Así que el algoritmo escucha los mensajes visuales publicados por el cuadricóptero, procesa la imagen para enfocar los colores rojos, a partir de eso binariza la imagen y busca el contorno que pertenezca a los cuadrados y así reconocer la distancia a su objetivo.

El sistema de navegación autónoma implementado se puede visualizar en la Fig. 2. Mediante la imagen generada por la cámara frontal del cuadricóptero, el dispositivo de procesamiento realiza la detección del marco, calcula la distancia hacia este y determinar la posición deseada junto con los mensajes de odometría del cuadricóptero, la cual es procesada por el controlador PID para determinar las velocidades que deben ser enviadas de vuelta al controlador de movimiento del cuadricóptero para llevar a cabo el objetivo. Este procedimiento es ciclado para actualizar la meta del cuadricóptero cada vez que una imagen nueva es

recibida. Toda la comunicación entre el dispositivo de procesamiento y el cuadricóptero se lleva a cabo por medio de Wi-Fi. Los módulos observados dentro del dispositivo de procesamiento serán explicados en las subsecciones 3.4, 3.5 y 3.6 de esta sección.

3.4 Control de movimiento del cuadricóptero

Para implementar el control del sistema, fue necesario leer constantemente los datos de odometría del cuadricóptero, el cual brinda la posición actual del cuadricóptero, obteniendo la posición la posición x, y, z y θ. Para el objetivo del artículo solo las primeras tres serán de importancia.

Después, se crean tres controladores PID distintos para cada coordenada en el espacio. La implementación de los controles PID se realizó a partir del paquete *simple-pid* [13] para Python3 [14]. A cada uno de estos controladores les serán asignadas las mismas constantes K: $K_p = 0.4$, $K_i = 0$ y $K_d = 1$.

Para determinar la velocidad que debe ser asignada al cuadricóptero bastaba con calcular el error de distancia para cada uno de los ejes y utilizar este valor para calcular la función del respectivo controlador PD (controlador proporcional-derivativo dado que $K_i = 0$) interpretando el valor que regresan como la velocidad que debe tener el sistema para cada eje. Debido a que este puede ser muy grande, se limitaran a una velocidad máxima de 0.15 ms^{-1}.

Finalmente, estos valores son enviados al cuadricóptero para que realice los movimientos. Para determinar cuándo ha llegado al punto destino, se utiliza una tolerancia de error de 10 centímetros, además de que la velocidad total del cuadricóptero debe ser menor a 0.1 ms^{-1} para asegurar que este no se pase del punto de destino a pesar de estar a menos de 10 cm de distancia.

3.5 Procesamiento de imagen

Para reconocer los ejes de la figura primero se debe hacer un preprocesamiento. Todo preprocesamiento fue realizado con la biblioteca OpenCV [14] para Python3. El inicio de este consiste en hacer un difuminado para que los píxeles se encuentren mejor definidos y sea más fácil reconocerlos para el proceso de binarización, utilizando una difuminación Gaussiana.

En este caso también debe reconocerse solo el color rojo. Para esto se usaron dos distintos rangos de rojo cuatro distintos valores, que representan variaciones del color en sus versiones opaca y brillante respectivamente, en HSV (*Hue*, *Saturation*, *Value*) con los valores (0, 50, 50) a (10, 255, 255) y (170, 50, 50) a (180, 255, 255) respectivamente. Estos se aplicaron a la imagen difuminada, con el propósito de filtrado de rojos, y una vez que se aplica esta máscara solo quedan las partes rojas de la imagen y se utiliza otra difuminación, a partir de eso se convierte a escala de grises y se binariza utilizando el método de Otsu. A partir de eso se aplica el método Canny para reconocer las aristas de la imagen y otro para buscar los contornos de manera jerárquica de la biblioteca utilizada, una vez que se obtienen los contornos de las figuras disponibles en el vídeo en vivo se seleccionan aquellas que tengan cuatro lados, al igual que el marco de referencia. Esto genera información del tamaño del marco y la posición del centro de este, siendo los pixeles de la imagen la unidad de medida.

3.6 Cálculo de distancia al marco de referencia

Una vez obtenida la información del marco en la imagen, se puede conocer la distancia a la que se encuentra el cuadricóptero al objetivo (sobre el eje *x*). Se requieren tres datos: el tamaño real del marco ($h = 1m$), el campo de visión del cuadricóptero ($\alpha = 50°$ en el campo vertical) y el tamaño en píxeles de la imagen ($H = 480\ px$). De esta manera, si el campo de visión abarca la longitud del marco de forma exacta, podemos determinar una distancia d de referencia para el resto de los cálculos a partir de una ley de senos:

$$\frac{\frac{h}{2}}{\sin(\alpha)} = \frac{d}{\sin(90-\alpha)} \rightarrow d = \frac{h \cdot \sin(90-\alpha)}{2 \cdot \sin(\alpha)} \approx 1.07\ m \quad [3]$$

Conociendo cuanto mide el lado del marco en píxeles (h_{px}), es posible determinar la distancia D a la que se encuentra el cuadricóptero a partir de una semejanza:

$$\frac{H}{D} = \frac{h_{px}}{d} \rightarrow D = \frac{H \cdot d}{h_{px}} \quad [4]$$

De esta manera, con las medidas obtenidas con el procesamiento de imagen, es posible calcular la distancia al plano del marco. Este cálculo no solo sirve para calcular la profundidad (en metros) a la que se encuentra el marco, sino que se puede utilizar un método similar para centrar el cuadricóptero.

Del procesamiento de imagen se conoce el centro del marco en pixeles, con lo cual es posible calcular su diferencia con el centro de la imagen (en pixeles) y convertirlo a metros utilizando la misma razón $\frac{h_{px}}{d}$. Sea x_{px}, y_{px} la distancia en pixeles horizontal y vertical, respectivamente y sean x, y sus respectivas distancias en metros. De esta manera podemos generar las siguientes ecuaciones

$$\frac{x_{px}}{x} = \frac{h_{px}}{d} \rightarrow x = \frac{x_{px} \cdot d}{h_{px}} \quad [5]$$

$$\frac{y_{px}}{y} = \frac{h_{px}}{d} \rightarrow y = \frac{y_{px} \cdot d}{h_{px}} \quad [6]$$

Así, ya se conocen las distancias en los tres ejes del espacio para llegar al objetivo. El siguiente paso es alimentar el controlador PD con estos datos para generar las velocidades necesarias que posteriormente son enviadas al controlador de movimiento del cuadricóptero para ejecutar su movimiento.

4. RESULTADOS

Los resultados del proyecto fueron satisfactorios, sin embargo, hay mucho espacio de mejora aún. El cuadricóptero fue capaz de completar la misión de forma consistente. Ocasionalmente se presentaban algunos problemas.

En la Fig. 3 se puede observar un ejemplo de cómo el cuadricóptero está detectando el marco de referencia, calculando su centro y calculando la distancia hacia este punto para los tres diferentes ejes. En la Fig. 4 se observa la misma escena, en esta ocasión mostrando la posición del cuadricóptero.

Fig. 3. Vista en primera persona del cuadricóptero, reconociendo el marco de referencia y calculando la distancia a este mismo en los distintos ejes.

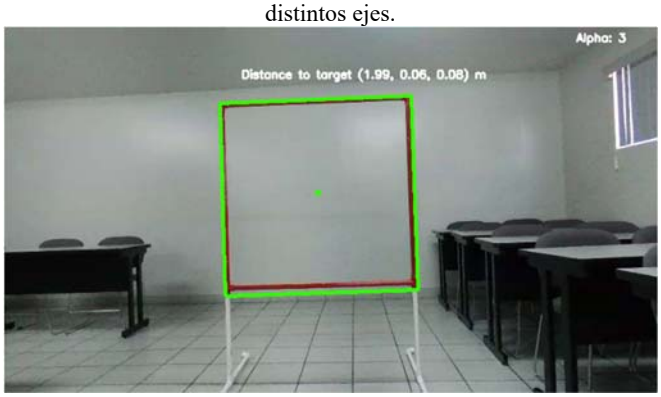


Fig. 4. Fotografía de la posición del cuadricóptero al mismo momento que está observando la escena de la Fig. 3.

El primer problema que se observó es que, al momento de acercarse al objetivo, el cuadricóptero podía avanzar al punto de perder el marco completo de su rango de visión, como se muestra en la Fig. 5, ocasionando que no se detecte (ya que el algoritmo requiere ver el objetivo completo) y comience a avanzar sin encontrarse necesariamente en el centro, llevando a una posible colisión.

El otro problema se debe a las condiciones de prueba no uniformes. Estas fueron realizadas en distintas áreas de CETYS Universidad, como salones de clases de distintos tipos o el auditorio de la escuela los cuales ofrecen diferentes tipos de iluminación y ruido de fondo. Los salones presentan un fondo más claro y limpio, con buena iluminación mientras que el auditorio tiene una variedad más amplia de colores e iluminación más pobre, generando imágenes oscuras. Este problema ocasionó que el rango de detección del objetivo disminuyera de forma considerable y era necesario iniciar la misión en una posición más cercana al marco para ser completada de forma satisfactoria.

Fig. 5. Vista en primera persona del cuadricóptero frente a un lado del marco, sin ser capaz de detectarlo, llevando a una colisión.

5. CONCLUSIÓN

La programación de vehículos autónomos no pilotados es una actividad cuya importancia ha estado en aumento, demostrado por la variedad de aplicaciones mencionadas en la introducción. Esto genera la motivación de realizar propuestas como la descrita en este artículo, en el cual se buscó que un cuadricóptero, de forma totalmente autónoma, pudiera despegar, detectar un marco frente a él, realizar las maniobras adecuadas para atravesarlo y finalmente aterrizar.

Las técnicas utilizadas para lograr el objetivo involucraron procesamiento de imagen a través de métodos como Otsu, Canny y difuminación Gaussiana a través de la biblioteca OpenCV para Python, junto con un sistema de control basado en un controlador PD y el módulo de odometría del Parrot Bebop 1 (cuadricóptero) para recibir información en tiempo real sobre su posición actual, las cuales fueron unidas a través de ROS, software especializado para la programación de robots.

Estas técnicas resultaron satisfactorias para completar la misión deseada, mostrando consistencia en las pruebas realizadas, sin embargo, aún existen problemas en la implementación. El primero está fuertemente ligado al método de visión, el cual requiere que el vehículo pueda captar el marco de referencia completo dentro de su rango de visión para detectarlo. En caso contrario, el cuadricóptero asume que no hay nada delante de él, y si se encontraba muy cerca de este podía llegar a estrellarse o rodear el objetivo. El segundo problema depende de las condiciones de prueba, ya que en algunos casos la iluminación y ruido de fondo es muy buena, mientras que en otros la luz es pobre y la variedad de colores aumenta, creando imágenes más oscuras y disminuyendo el rango de detección del cuadricóptero.

Finalmente, para trabajo futuro se busca ampliar la misión a completar, utilizando más de un marco de referencia para cruzarlos todos de forma totalmente autónoma. Esto tiene

algunos problemas con la propuesta actual, debido al ruido que causan los marcos extra. Además, para este nuevo objetivo y solucionar el primer problema descrito se ha considerado el uso de una red convolucional neuronal para que el modelo aprenda los movimientos que debe realizar a partir de lo que está observando. Otra característica que se considera agregar es implementar no solo el sistema de traslación del cuadricóptero, sino permitir que rote también para completar circuitos. Esto último ha sido probado algunas veces ya y ha generado problemas en la precisión de los movimientos por lo que requiere un mayor estudio.

6. REFERENCIAS

[1] A. Pattrick, G. Utama, A. Santosa, A. Chonawa, J. Suroso, R. Shofyanti, and W. Budiharto, "GNSS-based navigation systems of autonomous drone for delivering items," *Journal of Big Data,* 2019.

[2] A.R. Jah, *Theory, Design, and Applications of Unmanned Aerial Vehicles,* CRC Press, 2017.

[3] G. Strimel, S. Bartholomew, and E. Kim, "Engaging Children in Engineering Design through the World of Quadcopters," *Children's Technology and Engineering*, vol. 21, no. 4, May 2017.

[4] S. Dormido, A. Visioli, PID Control, In: Baillieul J., Samad T. (eds) Encyclopedia of Systems and Control, Springer, London, 2015.

[5] I. Rynning, Hue, Saturation and Value (HSV) Image Manipulation and Evaluation Through Numerical Techniques: A Review of the Tools Thereby, University of Colorado Colorado Springs (UCCS).

[6] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979.

[7] J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[8] I. Kamanga, "An Adaptive Approach to Improve Canny Method for Edge Detection," International Journal of Science and Research, vol. 6, pp. 164-168, 2017.

[9] Paparazzi UAV, The Free Autopilot, https://wiki.paparazziuav.org/wiki/Bebop

[10] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012), pp.1-3534, Dec. 2016.

[11] ROS, "About ROS," https://www.ros.org/about-ros

[12] Bebop Autonomy, "bebop_autonomy - ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 & 2.0," https://bebop-autonomy.readthedocs.io/en/latest/#bebop-autonomy-ros-driver-for-parrot-bebop-drone-quadrocopter-1-0-2-0

[13] Pypi. simple-pid, https://pypi.org/project/simple-pid/

[14] Python Software Foundation, Python Language Reference, version 3.7, http://www.python.org