

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA



**Diseño de módulo para la navegación autónoma de un vehículo
aéreo no tripulado en espacios interiores**

**TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO
ELECTRÓNICO**

AUTOR

Diego Ricardo Bueno Pacheco

ASESORES:

Andrés Flores Espinoza

Carlos Saito Villanueva

Lima, mayo 2019

Resumen

En la actualidad, la necesidad por automatizar procesos en diferentes áreas ha conducido al uso de vehículos aéreos no tripulados para la realización de tareas de forma autónoma y ágil. En este trabajo se desarrolla un método para la navegación autónoma de un vehículo aéreo no tripulado en espacios interiores privados de servicio de geolocalización. Para dicho método se diseña un módulo electrónico capaz de procesar imágenes para detectar códigos QR con lo cual tener información para realizar un plan de vuelo autónomo. La implementación se logra gracias a una computadora embebida (Raspberry Pi 3) que es capaz de realizar las tareas de comunicación y procesamiento en tiempo real para la navegación. También se hace uso del controlador de vuelo Pixhawk, un LIDAR para el control de la altitud de vuelo. En cuanto a software se hace uso de las librerías OpenCV y Dronekit para el lenguaje de programación Python. En los resultados se muestran el desempeño del módulo y las configuraciones necesarias para el óptimo funcionamiento del método de navegación.



Para todos aquellos que me apoyaron durante todos estos años de sacrificio

Sin ustedes no podría haber logrado este trabajo

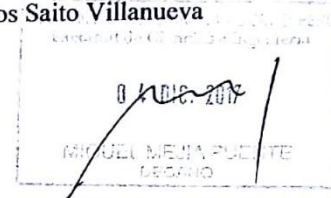
Por su amor y enseñanzas

¡Muchas gracias!



TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título : Diseño de módulo para la navegación autónoma de un vehículo aéreo no tripulado en espacios interiores
Área : Circuitos y Sistemas
Asesor : Ing. Andrés Flores Espinoza, Ing. Carlos Saito Villanueva
Alumno : Diego Ricardo Bueno Pacheco
Código : 20120858
Fecha : 13/09/17



1419

Descripción y Objetivos

A nivel mundial la navegación autónoma de vehículos aéreos no tripulados (VANT), conocidos comúnmente como drones, ha sido aplicada en diferentes industrias e investigaciones. En el caso del Perú, se presenta la necesidad de aplicar la navegación autónoma en espacios interiores donde el vehículo carece de localización por sistema global de navegación por satélite (en inglés Global Navigation Satellite System, GNSS) para aplicaciones como la revisión de molinos en la mina.

La tesis tiene como objetivo general el desarrollo de un módulo electrónico que permite la navegación de un VANT mediante el uso de códigos de respuesta rápida (en inglés Quick Response code, QR) en espacios interiores. Se hará uso del procesamiento digital de imágenes para encontrar una alternativa viable a la localización y posterior vuelo autónomo de un VANT en dicho entorno. Adicionalmente, la implementación será de manera embebida y el procesamiento en tiempo real.

Los objetivos específicos de la tesis son:

- Estudiar técnicas de procesamiento digital de imágenes aplicadas a códigos QR en conjunto con el lenguaje de programación de alto nivel - Python
- Integrar las técnicas de procesamiento digital de imágenes con las características de un código QR para obtener un método de navegación en espacios interiores
- Diseñar y desarrollar un módulo que procese la información de manera embebida, en tiempo real y que sea de un tamaño y peso adecuado para poder ser montado en un VANT
- Realizar simulaciones y pruebas para determinar los resultados de la navegación del VANT y del desempeño del módulo

MÁXIMO 50 PÁGINAS

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

M. Sc. Ing. WILLY CARRERA SORIA
Coordinador de la Especialidad de Ingeniería Electrónica



TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título : Diseño de módulo para la navegación autónoma de un vehículo aéreo no tripulado en espacios interiores

Índice

Introducción

1. Limitaciones y retos del vuelo de un vehículo aéreo no tripulado en espacios interiores
2. Tecnologías para la localización y vuelo en espacios interiores
3. Diseño y desarrollo de módulo electrónico para la navegación autónoma de un VANT en espacios interiores
4. Simulaciones, pruebas y resultados

Conclusiones

Recomendaciones

Bibliografía

Anexos

INDICE GENERAL

Capítulo 1 LIMITACIONES Y RETOS DEL VUELO DE UN VANT EN ESPACIOS INTERIORES.....	1
1.1. Definición espacio interior	1
1.2. Las limitaciones del servicio de localización GPS en espacios interiores	2
1.3. Los retos de la localización y navegación autónoma de un VANT en espacios interiores	4
Capítulo 2 PRESENTES TECNOLOGÍAS PARA LA LOCALIZACIÓN Y VUELO EN ESPACIOS INTERIORES.....	6
2.1. Estado del arte de las tecnologías para el posicionamiento en espacios interiores.....	6
2.2. Método de navegación y localización mediante el uso de sensores de distancia	8
2.3. Métodos de navegación y localización mediante visión por computadora sin el uso de marcadores visuales.....	9
2.4. Métodos de navegación y localización mediante visión por computadora con el uso de marcadores visuales.....	11
2.5. Síntesis y matriz de comparación entre tecnologías para la navegación autónoma	14
2.6. Modelo teórico del sistema de navegación autónoma.....	16
2.7. Comparación de procesadores embebidos para el uso en aplicaciones con VANTs	17
Capítulo 3 DISEÑO Y DESARROLLO DEL MÓDULO ELECTRÓNICO PARA LA NAVEGACIÓN AUTÓNOMA DE UN VANT EN ESPACIOS INTERIORES.....	20
3.1. Objetivo general y objetivos específicos	20
3.2. Herramientas para el procesamiento de imágenes	21
3.3. Método del procesamiento de imágenes para códigos QR.....	22
3.4. Desarrollo de código para la decodificación del código QR	23
3.5. Cálculo de distancia entre la cámara y el código QR	25
3.6. Información codificada en el código QR	26
3.7. Hardware para la implementación.....	27
3.8. Configuración y modos de vuelo en el Pixhawk	29
3.9. Comunicación entre la computadora y el controlador de vuelo.....	31
3.10. Hilos: programación para múltiples tareas	32
3.11. Simulador SITL – Software in the loop	33
Capítulo 4 SIMULACIONES y RESULTADOS	36
4.1. Resultados de la detección y decodificación de un código QR en una imagen	36
4.2. Resultado del método de medición de distancia del código QR a la cámara.....	41

4.3. Calibración y medición con el LIDAR.....	45
4.4. Simulación del vuelo con el software Gazebo	46
4.5. Implementación del módulo.....	48
Conclusiones.....	50
Recomendaciones	52
Bibliografía	53
Anexos	55



INDICE DE TABLAS

Tabla 1.1 factores de error en la señal GPS [2]	3
Tabla 2.1 Comparación de las fuentes especificadas	15
Tabla 2.2 comparación de plataformas de procesamiento embebido	18
Tabla 3.1 características del Raspberry PI 3 modelo B [12]	27
Tabla 3.2 características del Raspberry PI camara v1.3 [14].....	28
Tabla 3.3 características Pixhawk [15]	29
Tabla 3.4 características Lightware SF30/B [16].....	30
Tabla 4.1 FPS promedio para la toma y muestra de cuadros	38
Tabla 4.2 FPS promedio para la toma de cuadros	38
Tabla 4.3 FPS promedio aplicando el algoritmo de decodificación y la impresión de los cuadros	39
Tabla 4.4 FPS promedio aplicando el algoritmo de decodificación	39
Tabla 4.5 FPS promedio aplicando el algoritmo de decodificación sin interfaz gráfica.....	39
Tabla 4.6 constantes proporcionales a la distancia focal	42
Tabla 4.7 distancias máximas y mínimas posibles en la medición	42

INDICE DE FIGURAS

Figura 1.1 Recepción de señales satelitales en interiores [3]	3
Figura 2.1 medición de distancia [7]	8
Figura 2.2 Funnel lane producido por el VANT (UAV) y un punto característico [7]	9
Figura 2.3 Pistas de colores en el piso.	11
Figura 2.4 ejemplos de marcadores AR [9]	11
Figura 2.5 Algoritmo para el reconocimiento del marcador [10].....	13
Figura 2.6 Método para el cálculo del área [10]	13
Figura 2.7 Relación entre los componentes del sistema.....	16
Figura 3.1 Patrones de un código QR [20]	23
Figura 3.2 Transformada de perspectiva [20].....	23
Figura 3.3 Imagen para la calibración de cámara	25
Figura 3.4 Línea de manejo de imágenes [13]	28
Figura 3.5 Simulación de vuelo – SITL [18].....	33
Figura 3.6 Ambiente de laboratorio simulado en Gazebo	34

Figura 4.1 Resolución 320x240	37
Figura 4.2 Resolución 640x480	37
Figura 4.3 Resolución 1296x736	37
Figura 4.4 Pruebas de FPS aplicando el algoritmo de decodificación e impresión de cuadros	40
Figura 4.5 pruebas de FPS aplicando el algoritmo de decodificación	40
Figura 4.6 pruebas de FPS aplicando el algoritmo de decodificación sin la interfaz gráfica de usuario	41
Figura 4.7 verificación de distancia a 30 cm de la cámara	43
Figura 4.8 medición distancia con imagen 320x240	43
Figura 4.9 medición distancia con imagen 640x480	43
Figura 4.10 medición distancia con imagen 1296x736	44
Figura 4.11 Porcentaje de aciertos en la decodificación a 320x240	45
Figura 4.12 Porcentaje de aciertos en la decodificación a 640x480	45
Figura 4.13 simulación de vuelo en Gazebo	47
Figura 4.14 Simulación envío de comandos a partir de los códigos QR	47
Figura 4.15 distribución componentes en el VANT	48
Figura 4.16 Prueba de vuelo en exteriores.....	49
Figura 4.17 Prueba de vuelo en interiores.....	49

Introducción

En los últimos años se ha presentado un aumento significativo en el uso y aplicación de vehículos aéreos no tripulados (VANT), este incremento se debe a que los VANT presentan diferentes características que los hacen útiles para diversas aplicaciones. No obstante, existe también una necesidad de aplicar las características de un VANT en espacios interiores; como por ejemplo: centros comerciales, almacenes, estacionamientos, entre otros. Es por esta razón que en el presente trabajo se estudiarán y aplicarán soluciones para el vuelo autónomo de VANTs en espacios interiores.

La navegación autónoma de un VANT necesita de la señal de geolocalización para que el vehículo pueda seguir un plan de vuelo establecido. En el caso de ambientes exteriores se hace uso del sistema global de navegación por satélite (Global Navigation Satellite System, GNSS), el cual es capaz de proporcionar la ubicación del vehículo con un sistema de coordenadas en latitud y longitud, además de la altitud a la que se encuentra. Sin embargo, cuando se quiere utilizar el sistema de navegación por satélite en un espacio interior no es posible garantizar la correcta recepción de las señales de los satélites debido al error de multi trayecto. Entonces, el problema que se presenta es: ¿cómo lograr la navegación autónoma de un VANT en un espacio donde no es posible el uso del sistema GNSS?

El presente tiene como objetivo general solucionar el problema de la navegación en interiores mediante el diseño de un módulo electrónico que permita el vuelo del VANT. Para ello, se propone el uso de procesamiento de imágenes de códigos QR con el fin de lograr la navegación mediante comandos codificados en los códigos. Adicionalmente, se tiene como objetivos específicos: el estudio de técnicas de procesamiento digital de imágenes aplicadas a códigos QR, la integración de las características de los códigos con las técnicas de procesamiento de imágenes, lograr que el procesamiento sea en tiempo real, implementar el módulo de forma embebida, buscar la eficiencia en el peso y tamaño del módulo, y finalmente realizar pruebas para verificar la navegación del VANT, así como el desempeño del módulo.

El trabajo podrá servir para futuras aplicaciones como la grabación de videovigilancia, control de inventario, revisiones por imagen, entrega de paquetes o búsqueda de personas en espacios interiores.

CAPÍTULO 1

LIMITACIONES Y RETOS DEL VUELO DE UN VANT EN ESPACIOS INTERIORES

En el presente capítulo se explicará la problemática que se presenta en la navegación en espacios interiores. Además, se mostrarán las limitaciones del servicio de localización GPS y los posibles errores que disminuyen la precisión del servicio. Finalmente, se verán los retos que presenta la localización y vuelo autónomo de un VANT en espacios interiores.

1.1. Definición espacio interior

La definición más básica es la que hace referencia a lo que está dentro de un edificio o estancia. También se afirma que está vinculado inherentemente a un espacio definido arquitectónicamente; por lo tanto, es parte de la estructura y lo que está dentro de ella [1]. Sin embargo, la definición de espacio interior está especificada de diferente manera para cada disciplina en particular. Por lo que, en el caso del presente trabajo se define al espacio interior como todo ambiente donde no se puede contar con el servicio de geolocalización GPS. Es por ello que al tener una necesidad de realizar aplicaciones como: grabación de

videovigilancia, control de inventario, revisiones por imagen, entrega de paquetes o búsqueda de personas en espacios interiores; se debe buscar una solución a la navegación en espacios interiores.

1.2. Las limitaciones del servicio de localización GPS en espacios interiores

El principal problema de la navegación autónoma de un VANT dentro de interiores es no poder contar con el servicio de localización GPS debido a las limitantes que se presentan en el contexto de un espacio cerrado. A continuación, se tratará estos problemas y la relación con las dificultades de la navegación dentro de interiores.

Cuando un VANT se encuentra en exteriores es posible trazar planes de vuelo y conocer la ubicación del mismo en tiempo real gracias al servicio GPS. Sin embargo, cuando se trata de la navegación en interiores, la recepción de las señales satelitales se complica debido a que estas se transmiten en dos frecuencias pertenecientes a la banda L (1-2 GHz). Estas dos bandas son: 1.57542 y 1.2276 GHz. Debido a la naturaleza de las frecuencias de las ondas, no es posible atravesar objetos sólidos como el concreto. Adicionalmente, las señales satelitales tienen que viajar grandes distancias antes de llegar a la superficie de la tierra, durante todo ese trayecto existen diferentes factores que disminuye la precisión de la localización para los usuarios. Los factores que hacen menos exacta la localización son: errores del reloj de los satélites, error de efemérides, retrasos en la ionósfera, retrasos en la tropósfera, multi trayecto, retrasos debido al reloj y ruido en el receptor y la dilución geométrica de precisión. [2]

En primer lugar, se tiene el error de reloj del satélite, el cual se debe a la dificultad de sincronizar 20 satélites simultáneamente, por lo que el error en el tiempo puede llegar a ser de hasta 1 ms. Pudiendo reducirse a 3 ns con una corrección matemática. El error de efemérides se origina cuando la información de posición es transmitida de los satélites a las estaciones en tierra con el fin de calcular la información orbital de los satélites y poder devolver la información de efemérides a los satélites. El error de ionosfera es originado debido a que las ondas de radio que se propagan por la ionosfera experimentan un alejamiento de 1 a 10 m; sin embargo, gracias a que la señal es transmitida en dos frecuencias se puede calcular el retraso que se produce y minimizar el error. El error de tropósfera se origina por el cambio en la propagación de la velocidad de la luz debido a la refracción que existe en esa capa de la atmósfera; este error se reduce con el uso del GPS

diferencial. El error debido al multi camino se origina debido a que las señales de los satélites toman diferentes caminos dependiendo de los obstáculos presentes en el camino, estas señales se pueden unir de forma constructiva y/o destructiva lo que ocasiona errores de diferente magnitud; este error es el que más está presente cuando se usa el servicio de localización GPS dentro de interiores como se muestra en la Figura 1.1. Además, se pueden encontrar otros errores debido al ruido existente en el espacio y en el mismo receptor, que también puede presentar retrasos en el reloj debido a la dificultad de sincronización con múltiples satélites. A continuación, se presenta una tabla donde se resumen los factores de errores que. Si estos son considerados como fuentes Gaussianas, la suma RMS es 15 metros. [2]

Tabla 1.1 factores de error en la señal GPS [2]

Factor	Error (m)
Ionosfera	10
Multi Trayecto	10
Troposfera	3
Reloj del Receptor	3
Reloj del Satélite	1
Efemérides	1
Ruido del Receptor	1

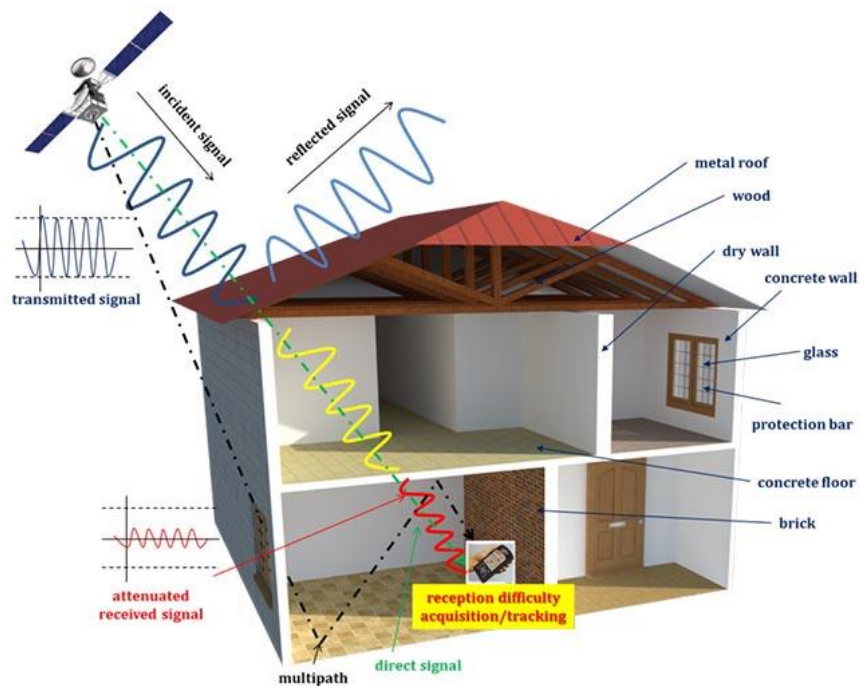


Figura 1.1 Recepción de señales satelitales en interiores [3]

La precisión tiene un error medio de 15 metros y cuando se trata de ambientes interiores el error puede incrementarse debido al error de multi trayecto causando mediciones totalmente erráticas. Como consecuencia de los diferentes errores mencionados anteriormente con las señales satelitales del GPS, se hace imposible el uso del mismo para la localización dentro de espacios interiores. Frente a esto se tiene la necesidad de buscar otros medios que reemplacen el servicio GPS en la navegación autónoma de un VANT. Este medio debe ser preciso y debe ser capaz de poder guiar al VANT mientras realiza el plan de vuelo. Por lo tanto, se debe tener en cuenta la necesidad de estimar las coordenadas de posicionamiento en tiempo real con el fin de generar la señal de control apropiadamente.

1.3. Los retos de la localización y navegación autónoma de un VANT en espacios interiores

La localización y navegación autónoma en espacios interiores presenta retos para lograr su correcta aplicación. Uno de ellos es la implementación de un software capaz de recibir la información de posición y decidir que comando de vuelo se debe enviar al VANT en el menor tiempo posible. Luego, es necesario contar con un hardware capaz de procesar los datos en tiempo real con el fin de evitar error en el orden de ejecución de los comandos de vuelo. Por otro lado, se necesita tener un sistema de control a la medida con la capacidad de responder inmediatamente ante señales de entrada de los sensores para evitar alguna colisión y permitir la estabilización del vuelo ante algún posible disturbio. Para ello se recomienda aplicar un sistema de control no lineal que ayude a mejorar el rendimiento y la robustez del sistema de control de vuelo.

Las características de un VANT que hacen eficiente el vuelo en espacios interiores son: ser lo más livianos posible, ser de bajo consumo energético y tener una relación precio–rendimiento adecuada [4]. Por ello, se presenta el reto de encontrar el correcto balance entre rendimiento y eficacia, puesto que el módulo debe contar con la menor cantidad de componentes para no sobrepasar la capacidad de carga del VANT y obtener el máximo tiempo de vuelo. Cabe resaltar que a medida que la carga del vehículo sea mayor, el tiempo máximo de vuelo se verá reducido. Debido a lo mencionado, se debe considerar las

opciones con menor uso de recursos energéticos y que estén a la medida de una implementación embebida. Entonces, se puede inferir que dependiendo del algoritmo a utilizar y de la cantidad de información que se procese, se podrá optar por un procesador de tamaño y peso reducido o por uno de mayores proporciones. Consecuentemente, la exigencia del procesamiento y el algoritmo utilizado se verá reflejado en la capacidad de carga que necesite el VANT. Sin embargo, en el caso se opte por realizar el procesamiento en un computador exterior, se debe tener en cuenta el rango de cobertura de la señal de comunicación entre la computadora y el VANT, así como la estabilidad del enlace inalámbrico de transmisión y recepción de datos.

No obstante, a pesar de que el procesador en tierra pueda ser más veloz; el procesamiento embebido ofrece más ventajas que el enlace con una estación de control en tierra, entre las ventajas podemos encontrar: mayor autonomía del VANT, un sistema con mayor robustez, e independencia de enlace con alguna estación de control en tierra. Debido a lo anterior surge el problema de encontrar un computador embebido que pueda satisfacer las siguientes necesidades: velocidad de procesamiento para una aplicación en tiempo real, bajo consumo energético, volumen reducido y peso reducido.

Con el fin de lograr identificar que procesador es más adecuado para el algoritmo utilizado es necesario probar el algoritmo en cada procesador para tener las mediciones de tiempo de procesamiento correctas. Pero como no es posible probar el algoritmo en todos los procesadores, se debe tener una referencia de trabajos afines para poder escoger el mejor procesador posible. [5]

CAPÍTULO 2

PRESENTES TECNOLOGÍAS PARA LA LOCALIZACIÓN Y VUELO EN ESPACIOS INTERIORES

En este capítulo se desarrollará algunas de las tecnologías que dan solución al posicionamiento y navegación de un VANT en espacios interiores. Se mostrarán las metodologías a seguir según diferentes autores y se comparará las propuestas de solución. Finalmente se explicará un trabajo de comparación de computadoras embebidas para el procesamiento digital de imágenes en VANTs.

2.1. Estado del arte de las tecnologías para el posicionamiento en espacios interiores

Gracias al constante desarrollo de diferentes tecnologías relacionadas con el control, el procesamiento de información y la reducción del tamaño de componentes electrónicos ha sido posible desarrollar diversas aplicaciones para el uso de VANTs. Dentro de estas ha surgido el interés por el uso de VANTs en espacios interiores para diferentes aplicaciones

tales como: entregas dentro de edificios, inventario automático, juegos basados en VANTs o hasta operaciones de búsqueda y rescate.

Sin embargo, muchas de las aplicaciones desarrolladas han sido aplicadas a exteriores o espacios abiertos debido a la facilidad en el uso del servicio de geo localización – GPS. Contrariamente, cuando se trata de aplicaciones en espacios interiores nos encontramos en un contexto donde no es posible obtener una buena recepción de las señales satelitales. El presente estudio se enfocará en soluciones con el uso de sensores de distancia y el uso del procesamiento de imágenes digitales. Se puede encontrar otros métodos, como la triangulación gracias a redes inalámbricas o el uso de realidad aumentada, que también son útiles para localizar algún dispositivo electrónico dentro de interiores; no obstante, son menos viables debido a la complejidad en la implementación y uso. Por lo tanto, estas no se consideran en el estudio.

Adicionalmente se estudiarán los algoritmos necesarios para el control de vuelo del VANT. Se verán diferentes tipos de control de vuelo que permitan tener una respuesta adecuada para la movilidad del vehículo dentro de espacios reducidos.

Finalmente se verá los métodos que se utilizan para guiar o localizar al VANT en un entorno establecido. Para ello se presentará diferentes métodos que incluyen el análisis de: marcadores visuales, características del entorno, diferencias entre superficies u objetos, esquinas detectadas en imágenes y hasta pistas de colores predefinidas.

La navegación autónoma de VANTs dentro de interiores requiere diferentes tipos de parámetros para poder controlar el vuelo. Estos parámetros son: posición del VANT en el espacio, altitud, ángulos de navegación del VANT (de dirección, elevación y alabeo), velocidad del VANT, entre otros. Para el control de todos estos parámetros se precisa de un sistema que responda en tiempo real y correctamente respecto a los parámetros obtenidos. Para ello se presentan métodos basados en diferentes tecnologías: haciendo uso de sensores de distancia, mediante el procesamiento digital de imágenes o visión por computadora al entorno y a marcadores visuales.

2.2. Método de navegación y localización mediante el uso de sensores de distancia

Nonami, Kendoul, Suzuki, Wang y Nakazawa [6] presentan un método de navegación y aterrizaje de un MAV - *micro aerial vehicle* mediante el uso de cuatro sensores infrarrojos y un sensor de ultrasonido de peso y tamaño reducido debido a la limitada capacidad de carga de un MAV. El sistema propuesto calcula la posición horizontal gracias a la medición de distancias hacia objetos que se encuentran en el plano horizontal tales como paredes. Sin embargo, cuando no se encuentran objetos cerca, el sistema usa un algoritmo que calcula la posición horizontal detectando los bordes de los objetos que se encuentran debajo del MAV. Además, se usa el sensor de ultrasonido para calcular la posición vertical del MAV. El principio básico de funcionamiento del sistema se basa en la rotación de los cuatro sensores infrarrojos gracias a cuatro servo motores. En un principio se asume que el MAV está encima de algún objeto, luego de esto los sensores infrarrojos rotan sobre los ejes hasta que se encuentre algún borde. Entonces, debido a la discontinuidad de la superficie se obtiene un cambio brusco en la medición de la distancia hacia el objeto, es ahí cuando se detecta el borde del objeto o superficie que se encuentre debajo del vehículo. Adicionalmente se mide la distancia vertical entre el MAV y el objeto con lo que se puede calcular la distancia horizontal desde el MAV al borde del objeto como se muestra en la figura 2.1.

$$L = h \tan \tilde{\alpha} + \frac{1}{2}w$$

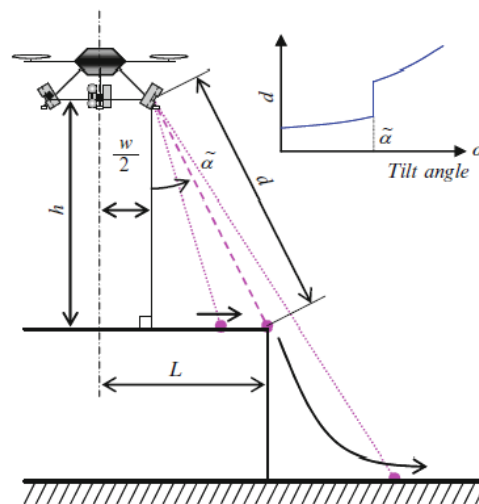


Figura 2.1 medición de distancia [7]

2.3. Métodos de navegación y localización mediante visión por computadora sin el uso de marcadores visuales

El método presentado por: Nguyen, Mann y Gosine [7] propone una solución mediante el procesamiento digital de imágenes y la técnica de navegación visual de aprendizaje y repetición (*teach-and-repeat*) para el reconocimiento de características de esquinas en la imagen – o el método: ‘Kanade-Lucas-Tomasi’. Además, se propone el uso de la teoría del carril de embudo o *Funnel Lane Theory* para el cálculo de la navegación dentro de interiores. El *funnel lane* define un conjunto de posibles ubicaciones donde el VANT debe ser controlado para poder volar directo hacia un punto característico del entorno. En la figura 2.2 se muestra las ubicaciones mencionadas en la teoría como una pirámide roja con superficie plana respecto al VANT y al punto de destino. En el proceso de detección de imágenes se tienen numerosos puntos de referencia, los cuales producen un *funnel lane* cada uno, por lo tanto, cuando el VANT se encuentre dentro de esta intersección podrá volar hacia adelante y en caso contrario deberá regresar a la intersección de los carriles para poder continuar con el vuelo. Cabe resaltar que, en el caso propuesto el VANT solo necesita un punto de referencia para asegurar la confiabilidad del sistema.

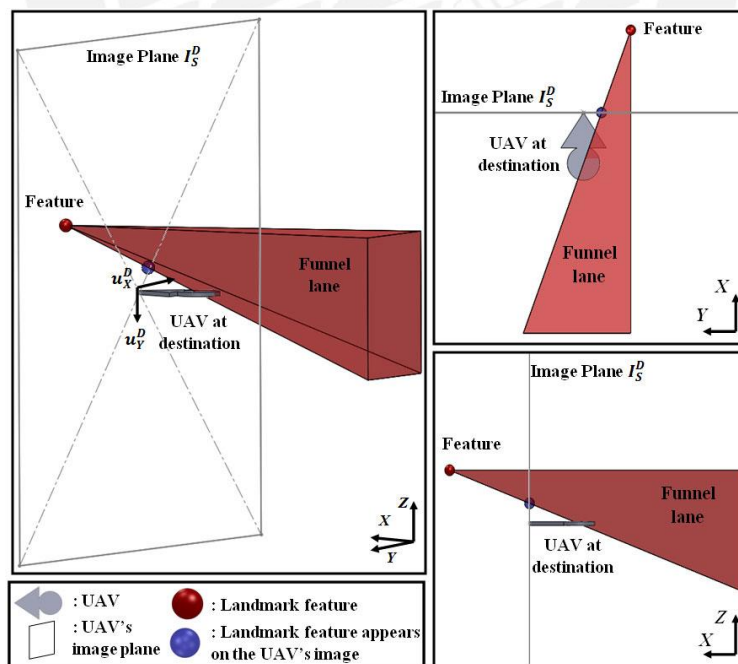


Figura 2.2 Funnel lane producido por el VANT (UAV) y un punto característico [7]

Luego de definir el concepto de *Funnel Lane* se construye un camino de referencia por donde volará el VANT, este camino se dividirá en diversos segmentos continuos que tendrán imágenes de comienzo y final de segmento. Además, se tiene un conjunto de imágenes en la posición actual del vehículo que se comparan con las anteriores para identificar el inicio o fin de un segmento. En la fase de aprendizaje se construye el camino visual que deberá seguir el VANT mediante la recolección de varias imágenes de referencia a lo largo del camino. Y en la fase de repetición se realiza el recorrido del camino gracias al cálculo de navegación usando la teoría *Funnel Lane*.

Algunos posibles inconvenientes con este método son: la posición inicial de vuelo tiene que ser lo más cercana posible a la posición original de la etapa de aprendizaje, la transición entre segmentos es muy aproximada debido al criterio de error cuadrático medio usado y finalmente el control utilizado es PID, el cual produce un sobre impulso considerable y es sensible al ruido del ambiente. No obstante, a pesar de los inconvenientes el sistema fue capaz de realizar el vuelo en los experimentos realizados.

King Phang, Jie Ong, Chen, Yeo y Lee [8] presentan una solución embebida utilizando también procesamiento de imágenes para el reconocimiento de bordes en un camino de tres colores como el de la figura 2.3, esto con la finalidad de que las pistas den la información de dirección, sentido y trayectoria al VANT. El algoritmo de procesamiento de imagen utilizado es el de rápida detección del camino que tiene como innovación el especial cuidado que se pone en los píxeles más cercanos en los marcos de las pistas. La característica de la imagen que se usa para la detección son los bordes entre los segmentos de diferentes colores. Para esto se utiliza una detección de bordes adaptativa con el fin de dividir el borde de las líneas de colores en segmentos y así poder usar una función que clasifica los píxeles como posibles píxeles de un borde y otros píxeles gracias al uso de la distancia euclidiana entre dos píxeles adyacentes en un espacio de color RGB.

Luego de aplicar la función tenemos clasificados los píxeles en píxeles potenciales de ser un borde y otros píxeles que no cumplen con el valor límite de distancia euclidiana y que por lo tanto no son parte de un borde, inmediatamente se pasa a la clasificación de colores en los segmentos utilizados y no utilizados en toda la imagen.

Finalmente, con la segmentación de detección de bordes y la clasificación de colores de cada segmento se puede detectar el camino buscando la secuencia correcta de colores y calculando las coordenadas del punto medio del camino entre las secciones alta y baja del propio camino en la imagen. Al conocer estos puntos se puede calcular el ángulo de dirección y la distancia al camino de color para poder realizar el vuelo.



Figura 2.3 Pistas de colores en el piso.
El orden de los colores determina la dirección de vuelo [8]

2.4. Métodos de navegación y localización mediante visión por computadora con el uso de marcadores visuales

Los trabajos de: Boudjit y Larbes en [9] y de Kim, Suk Lee y Han en [10] presentan ambos un método de navegación mediante el rastreo de marcadores visuales como códigos QR o marcadores AR - *Augmented Reality* como los mostrados en la Figura 2.4.



Figura 2.4 ejemplos de marcadores AR [9]

Boudjit y Larbes [9] utilizan un sistema de control de vuelo mediante lógica difusa para que el VANT pueda acercarse a un marcador determinado desde diferentes posiciones. Los autores apuestan por un control de lógica difusa debido a que este resuelve los problemas de precisión e incertidumbre que padecen los sistemas de control reales. El controlador de lógica difusa es uno de los métodos más activos de inteligencia artificial ya que este está basado en el pensamiento y toma de decisión humano.

En el trabajo realizado se busca resolver el problema del rastreo automático de un marcador visual como un código QR o AR. Para ello primero se garantiza que el VANT pueda reconocer el objetivo correctamente, luego se busca que el VANT vuele a la posición deseada cerca del objetivo y finalmente se calcula la posición relativa de las coordenadas respecto al objetivo para que el vehículo pueda acercarse controladamente.

Para el reconocimiento del marcador se utiliza el software *ROS Package 'ar_track_alvar'* el cual reconoce las esquinas del marcador fácilmente con una sola cámara o una sola perspectiva. Luego se calcula la homografía de la superficie del marcador. Entonces, cuando la cámara esta calibrada y se conoce el tamaño del marcador es posible calcular la posición del marcador en distancias reales. No obstante, el método presentado tiene limitaciones en el reconocimiento de marcadores, no es posible detectar el marcador a menos de medio metro de distancia respecto al marcador y distancias mayores a tres metros debido a que se presenta más ruido en las imágenes.

Se usa además códigos QR para obtener información de posición, orientación e identificación del código, estos datos sirven de entrada al controlador de lógica difusa y así este puede tomar acciones frente a la información recolectada. Para la decodificación del código QR se utilizó la librería *Zbar* en Linux.

La secuencia de acción que se tiene es la siguiente: si el marcador no es visible se busca el mismo, luego si el marcador es visible se alinea el vehículo con el objetivo y finalmente si está visible y alineado el vehículo se acerca al objetivo aplicando el control difuso.

Kim, Suk Lee y Han [10] proponen un sistema de vuelo que rastrea un marcador y mantiene una distancia adecuada con el objetivo. El algoritmo de reconocimiento del marcador usado calcula el área del marcador mediante un modelo geométrico del mismo. Este modelo se obtiene gracias a la detección de los cuatro vértices del marcador. Además, el trabajo hace uso de un celular con plataforma *Android* para recibir las imágenes capturadas por el VANT, procesarlas y finalmente enviar los comandos de vuelo al controlador de vuelo del VANT.

Adicionalmente, se tiene un algoritmo para el reconocimiento y comparación del marcador con una base de datos de todos los marcadores y las codificaciones, con lo cual se puede

obtener información del marcador rastreado. El algoritmo de reconocimiento del marcador usado en el sistema es mostrado en la figura 2.5.

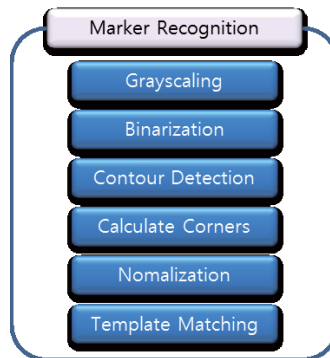


Figura 2.5 Algoritmo para el reconocimiento del marcador [10]

El método para mantener la distancia entre el vehículo y el objetivo se basa en el cálculo de distancia de los cuatro vértices para luego poder dividir el cuadrado en dos triángulos mediante una línea diagonal entre dos vértices como se muestra en la figura 2.6. Luego se calcula el área de los dos triángulos mediante las coordenadas de los vértices. Al tener las áreas de los triángulos calculadas se suman para obtener el área total del marcador. Entonces, sabiendo el tamaño del marcador calculado en la imagen y el tamaño real predefinido del mismo se puede calcular la distancia entre el vehículo y el marcador. Si el tamaño del marcador detectado es mayor que el real, el sistema manda un comando para que el VANT retroceda y en caso contrario se manda un comando para que el VANT avance.

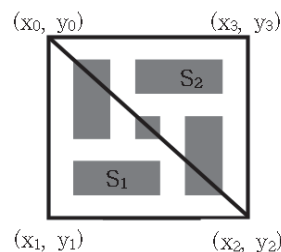


Figura 2.6 Método para el cálculo del área [10]

Conjuntamente se propone un método para alinear el centro de las coordenadas del marcador con el centro de la imagen obtenida por el VANT con el fin de que el sistema de rastreo tenga la precisión adecuada y pueda funcionar correctamente. Entonces, si el centro del marcador está fuera de la posición requerida, el VANT busca corregir la posición del centro de la imagen para que coincida con el centro del marcador.

Claudio E. Palazzi [11] propone un sistema de auto localización de un VANT en espacios interiores. Para ello hace uso de múltiples marcadores artificiales de los cuales se puede hallar las coordenadas de las posiciones en la imagen y a partir de ello se utiliza un algoritmo para calcular las coordenadas de la cámara respecto al marcador. Y finalmente conociendo las coordenadas globales de los marcadores se puede calcular las coordenadas globales de la cámara. Los resultados mostrados en el trabajo tienen un error de 20 cm aproximadamente lo que hace que sea posible utilizar este método para lograr aplicaciones dentro de ambientes interiores.

2.5. Síntesis y matriz de comparación entre tecnologías para la navegación autónoma

Se presentan diferentes métodos de solución para la navegación autónoma de VANT con diferentes niveles de complejidad y de eficiencia. Dentro de las soluciones se pueden encontrar el uso de sensores para medir distancias y estimar posición, el uso de procesamiento de imágenes con diferentes matices en cuanto al objetivo o referencia de la imagen que se procesa – desde todos los puntos característicos del entorno hasta un camino predefinido dibujado en el suelo – y finalmente el uso de marcadores visuales para la obtención de información relevante para el plan de vuelo del VANT.

Encontramos que el uso de sensores de distancia como el sensor infrarrojo y el sensor de ultrasonido son métodos que requieren un control adicional que maneje el movimiento de los sensores para que estos puedan hacer un barrido al entorno y así puedan detectar obstáculos o discontinuidades que simbolizan algún borde o esquina. Además, este método presenta el requerimiento de uso de distintos sensores que trabajen en conjunto y en sincronía.

Dentro de los métodos que utilizan el procesamiento digital de imágenes encontradas se distinguen dos grupos: los que hacen uso de marcadores visuales proporcionar información para el vuelo y los que no hacen uso de los marcadores visuales obteniendo puntos característicos en las imágenes que sirven de guías para poder realizar la navegación.

En cuanto al grupo que utiliza marcadores visuales encontramos que estas técnicas tienen una versatilidad en la implementación ya que los marcadores se pueden colocar en

diferentes posiciones de un ambiente, con lo que se puede lograr dar información de vuelo cada cierto tramo del recorrido.

En cambio, en el caso del grupo que no usa marcadores visuales, tenemos sistemas de mayor complejidad en la implementación. Sin embargo, estos métodos poseen independencia en el uso debido a que no instan de algún tipo de guía o patrón previamente definido para la correcta navegación. No obstante, para el uso de estos sistemas es necesario pasos previos al vuelo como lo son la calibración adecuada de las cámaras o una etapa de aprendizaje en donde se reconozcan y almacene datos importantes en la ruta de vuelo que seguirá el VANT en la etapa de repetición o vuelo autónomo.

Tabla 2.1 Comparación de las fuentes especificadas

Autores	Método de navegación utilizado	Método de control utilizado	Donde se realiza el procesamiento	Objeto o característica guía
Kenzo Nonami et al [6]	Uso de sensores infrarrojos y de ultrasonido	Control de espacio - estado	Sistema embebido	Entorno en general y objetos debajo del VANT
Trung Nguyen et al [7]	Procesamiento de imágenes al entorno en general	PID	Sistema externo – Laptop	Entorno en general
Swee King Phang et al [8]	Procesamiento de imágenes a un camino predefinido	PID	Sistema embebido	Camino con pistas de colores
Kamel Boudjit et al [9]	Procesamiento de imágenes a marcadores	Lógica difusa	Sistema externo – Laptop	Marcadores visuales
Jin Kim et al [10]	Procesamiento de imágenes a marcadores	No especifica	Sistema externo – celular con <i>Android</i>	Marcadores visuales

Lo más resaltante en la matriz presentada es la presencia de una mayor cantidad de alternativas propuestas basadas en procesamiento digital de imágenes y que dentro de ellas hay dos grupos definidos: el grupo que hace uso de marcadores visuales y el grupo que no hace el uso de estos. Adicionalmente, se puede observar que, dependiendo de la complejidad del algoritmo utilizado; el procesamiento se realiza externamente o de manera embebida.

2.6. Modelo teórico del sistema de navegación autónoma

La navegación autónoma de un VANT dentro de interiores requiere que el vehículo pueda estimar la posición y navegación por sí mismo. Para ello se puede usar: la visión por computadora, sensores infrarrojos, sensores de ultrasonido, sensor laser, entre otros. En el sistema de visión por computadora es necesario tener una cámara con la suficiente resolución y captura de cuadros por segundo para asegurar la mejor adquisición de imagen. Luego se requiere una plataforma o computadora en la cual se realice el procesamiento de la información para que en un siguiente paso se pueda enviar los comandos al controlador de vuelo y que este al mismo tiempo pueda mandar las señales de control a los motores del VANT. Finalmente se tiene un elemento externo al VANT que son los marcadores visuales. Estos permitirán la estimación de la posición y además servirán como contenedores de información para la ruta de vuelo. La relación entre estos elementos es mostrada gráficamente en la figura 2.7.

En el caso de la implantación del sistema es necesario tener métodos de captura de imágenes, de identificación de los marcadores, decodificación del marcador, procesamiento de la información obtenida del marcador, estimación de la posición del VANT, comunicación entre la computadora y el controlador de vuelo, y finalmente la ejecución del vuelo mediante un control de las señales para el movimiento. Todos estos métodos deben de ser integrados para formar el sistema en conjunto.

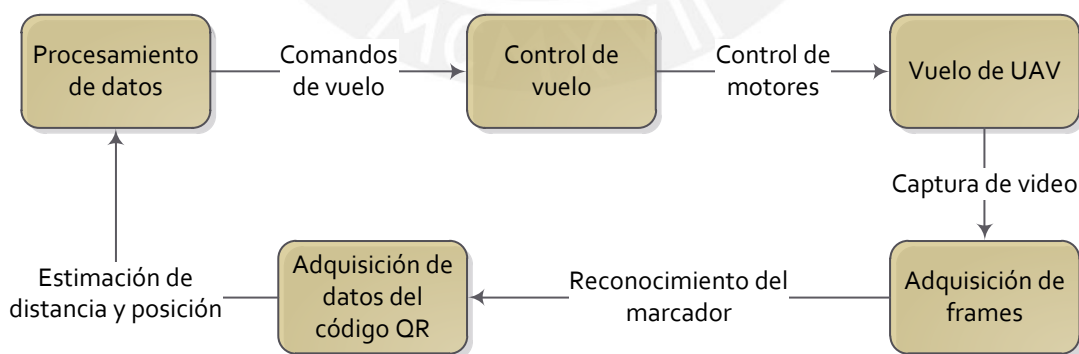


Figura 2.7 Relación entre los componentes del sistema

Según sea el algoritmo que se utilice se tiene una mayor o menor demanda de procesamiento requerido, esto lleva a la consideración de optar por tener un sistema embebido para el control del VANT o por tener un mecanismo de transmisión y recepción

de datos en el mismo VANT con el fin de enviar y recibir información de una computadora exterior y así esta pueda procesar la información que necesita el VANT en el vuelo. Es aquí donde la técnica de reconocimiento de marcadores puede ser más viable para la implementación en un sistema embebido, mientras que el reconocimiento sin el uso de marcadores es más exigente en el procesamiento por lo que se utiliza una computadora externa para la implementación.

Sin embargo, independientemente de la técnica que se utilice el resultado esperado debe ser la obtención de la posición del VANT dentro de un espacio interior para que luego sea posible trazar un plan de vuelo en una ruta definida y que el VANT sea capaz de realizarlo sin necesidad de operación de algún usuario.

Por lo tanto, se tiene que verificar el sistema con simulaciones por computadora y pruebas de vuelo reales en las cuales veremos: el rango de distancia en la cual se puede reconocer el marcador, el tiempo que se demora en recolectar la información del marcador, el tiempo que le toma al VANT realizar el plan de vuelo y finalmente la precisión con la que realiza el plan de vuelo. Para este último se tendrá que verificar si el vehículo llegó a los puntos donde se encuentran los marcadores visuales.

2.7. Comparación de procesadores embebidos para el uso en aplicaciones con VANTs

Hulens, Verbeke y Goedemé [5] hacen un estudio para determinar la mejor plataforma de procesamiento embebido para tratar imágenes digitales a bordo de VANTs. Las plataformas que se comparan en el estudio se muestran en la tabla 2.2 además se mencionan las características más resaltantes.

En el estudio se definen como referencias en la comparación cinco parámetros principales: velocidad de procesamiento, disipación de potencia, tiempo estimado de vuelo, volumen y peso del computador. Adicionalmente, se tiene en cuenta el nivel de complejidad del algoritmo con el son comparados los diferentes computadores. El nivel de complejidad de un algoritmo se aproxima mediante una relación lineal entre la velocidad de ejecución del algoritmo y el grado de complejidad del mismo. Para ello se tiene como nivel de complejidad mínimo la aplicación de un filtro mediano 3×3 en una imagen a color de 640×480 píxeles,

y como nivel máximo la detección de personas usando el descriptor HOG en una imagen con resolución 640 x 426. Teniendo estos límites se realiza una aproximación lineal para hallar la complejidad de un algoritmo y la velocidad que se conseguiría en cada una de las plataformas.

Luego de que se calcula las velocidades promedio del procesamiento, se pasa a calcular la potencia que se disiparía de acuerdo con la velocidad de procesamiento. En este caso también se hace una aproximación lineal de la potencia disipada con la velocidad de procesamiento. Con el fin de tener una referencia para la relación lineal se toma en cuenta la velocidad de procesamiento como la tasa de cuadros por segundo (FPS) necesaria para la aplicación y la velocidad máxima como la tasa de cuadros por segundo máxima que se puede conseguir con el algoritmo. Teniendo así una relación entre el consumo máximo y el consumo mínimo necesario. Seguidamente, se calcula la potencia total disipada por los motores y por el procesador. Esto se consigue teniendo los parámetros de eficiencia de motor, número de hélices, el peso total del vehículo, entre otros.


Finalmente teniendo la potencia total consumida, se estima el tiempo de vuelo de acuerdo con la capacidad de la batería (Ah) a un 75%.

Tabla 2.2 comparación de plataformas de procesamiento embebido

Plataforma	Procesador	Memoria RAM	Peso (gramos)	Potencia (Watts)	Volumen (cm³)
Computadora de escritorio	Intel i7-3770	20 GB	740	107	4500
Raspberry PI	ARM1176JZF-S	512 MB	69	3.6	95
Odroid U3	Samsung Exynos	2 GB	52	6.7	79
Odroid XU3	Samsung Exynos	2 GB	70	11	131
Jetson	Cortex A15	2 GB	185	12.5	573
mini-ITX atom	Intel Atom D2500	8 GB	427	23.5	1270
mini-ITX i7	Intel i7-4770S	16 GB	684	68	1815
Nuc	Intel i5-4250U	8 GB	550	20.1	661
Brix	Intel i7-4500	8 GB	172	26	261

En el estudio se analizan los resultados de la aplicación de dos algoritmos de implementación real, el primero es la detección de rostros mediante el uso del detector de Viola y Jones en OpenCV y el segundo es un algoritmo para la detección del camino en un huerto haciendo uso de la transformada de Hough para la búsqueda de líneas rectas del corredor y el filtro de Kalman para calcular el punto de fuga de la imagen del camino en el huerto. Luego de las pruebas, el estudio concluye que los siguientes parámetros: complejidad del algoritmo utilizado, velocidad de procesamiento o la tasa de FPS requerido, y el peso del computador; tienen un impacto considerable en la potencia total disipada y por consecuencia se obtiene un tiempo de vuelo más reducido según varíen estos parámetros. Por lo tanto, cuando se requiere hacer uso de VANTs de tamaño reducido es vital la optimización del consumo de energía del sistema para poder alcanzar los tiempos de vuelo requeridos y realizar el procesamiento adecuadamente. Cabe resaltar que las plataformas que mejores tiempos de vuelo (mejor rendimiento) obtuvieron fueron: Odroid U3 y Odroid XU3.





CAPÍTULO 3

DISEÑO Y DESARROLLO DEL MÓDULO ELECTRÓNICO PARA LA NAVEGACIÓN AUTÓNOMA DE UN VANT EN ESPACIOS INTERIORES

En el presente capítulo se mostrará el diseño y desarrollo de la propuesta de solución para la navegación autónoma de un VANT.

3.1. Objetivo general y objetivos específicos

La tesis tiene como objetivo general el desarrollo de un módulo electrónico que permite la navegación de un VANT mediante el uso de códigos QR en espacios interiores o cerrados.

Los objetivos específicos de la tesis son:

- Estudiar técnicas de procesamiento digital de imágenes aplicadas a códigos QR en conjunto un lenguaje de programación de alto nivel como Python

- Integrar las técnicas de procesamiento digital de imágenes con las características de un código QR para obtener un método de navegación en espacios interiores
- Diseñar y desarrollar un módulo que procese la información de manera embebida, en tiempo real y que sea de un tamaño y peso adecuado para poder ser montado en un VANT. En este caso el concepto de tiempo real tendrá que ser tomado como la capacidad de poder enviar comandos de navegación al controlador de vuelo mientras se procesa las imágenes respectivas con el fin de mantener la mínima velocidad de vuelo (aproximadamente 0.5 m/s) y mantener una distancia prudente hacia el código QR
- Realizar simulaciones y pruebas para determinar los resultados de la navegación del VANT y del desempeño del módulo

A continuación, se mostrarán los elementos necesarios para el diseño del módulo electrónico.

3.2. Herramientas para el procesamiento de imágenes

- OpenCV

OpenCV (*Open Source Computer Vision*) es un grupo de librerías de código abierto que tienen implementadas funciones y algoritmos especializados en visión por computadora. Estos conjuntos de librerías han sido desarrollados para obtener una alta eficiencia computacional y poder realizar aplicaciones en tiempo real. Esta escrito y optimizado en lenguaje C/C++ para poder sacar ventaja del multiprocesamiento en las diferentes plataformas. Además, la librería cuenta con diferentes funciones útiles para el cálculo de distancia entre una imagen y la cámara, la detección de esquinas o bordes, el escalamiento de imágenes, entre otros. Finalmente, las funciones pueden ser implementadas en diferentes lenguajes de programación como: C, C++, Python o Java. Para el desarrollo del presente trabajo se implementará las funciones en Python.

- Zbar

Zbar es una librería de código abierto que contiene funciones especializadas en la detección, pre procesamiento y decodificación de códigos de barra y códigos QR. Puede ser implementado en C/C++ o Python y es compatible con el uso de OpenCV. Además, la librería ha sido optimizada para obtener una alta velocidad en la detección y decodificación

de los códigos. Debido a que no se realizan operaciones de punto flotante y a que el consumo de memoria es reducido, la librería es ideal en el uso en aplicaciones en tiempo real.

3.3. Método del procesamiento de imágenes para códigos QR

El objetivo del presente trabajo es hacer uso de códigos QR para lograr obtener información para la navegación del VANT. Para ello es necesario tratar la imagen original capturada por el sensor con una serie de técnicas de procesamiento. Dentro este proceso se puede resaltar los siguientes pasos:

1. Conversión de imagen a escalas de grises: la imagen capturada de tres canales RGB tiene información redundante para la decodificación de un código QR. Adicionalmente una imagen en escala de grises requiere menos memoria y tiene un menor costo computacional. El método que se utiliza para convertir una imagen RGB a escala de grises tiene como finalidad obtener una relación entre la sensibilidad del ojo humano y los colores de la imagen. Las tonalidades verdes son más sensibles en el ojo humano, mientras que las tonalidades azules no son tan sensibles. Es por lo que se sigue la siguiente fórmula para la conversión:

$$F(x, y) = 0.30R(x, y) + 0.59G(x, y) + 0.11B(x, y) \quad \text{Ecuación 1}$$

2. Umbralización de la imagen: luego de tener una imagen en escala de grises, esta pasa a ser una imagen solo en blanco y negro (1 y 0). Debido a que un código QR solo necesita dos colores para la decodificación. La selección apropiada del valor límite es importante para la correcta decodificación. Debajo de este valor todos los píxeles serán considerados como negros y encima del valor como píxeles blancos. Para ello la librería Zbar calcula el valor límite dependiendo de las características del cuadro tomado. El valor límite que se tome dependerá en gran medida de la imagen original que se tenga. En esta selección se tendrá que tomar en cuenta la iluminación presente en la imagen original, mientras más uniforme sea la iluminación mejor se podrá realizar la umbralización del cuadro sin que se pierda las formas en la imagen.

La fórmula utilizada en la umbralización es la siguiente:

$$F(x, y) = \begin{cases} 1, & G(x, y) > T \\ 0, & G(x, y) < T \end{cases} \quad \text{Ecuación 2}$$

3. Detección y localización de patrones del código QR: en este paso es necesario encontrar patrones característicos del código QR en la imagen mediante herramientas para la detección de bordes. Los patrones necesarios para ello son: 3 patrones de búsqueda y un patrón de alineamiento. Una vez calculada las posiciones de los 3 patrones de búsqueda, mediante el trazado de líneas paralelas a los patrones es posible hallar el patrón de alineamiento en la parte inferior derecha del código. Finalmente, cuando se conocen estos cuatro patrones en la imagen es posible aplicar la transformada de perspectiva para corregir la distorsión geométrica que puede presentarse en la captura del código QR. Las figuras 3.1 y 3.2 muestran los patrones del código QR y la transformada de perspectiva para los mismos.

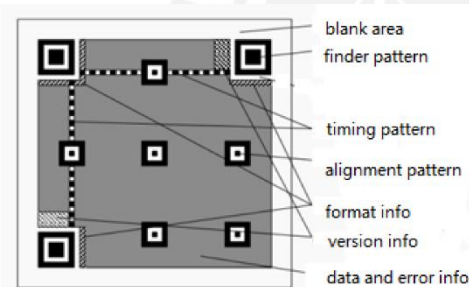


Figura 3.1 Patrones de un código QR [20]

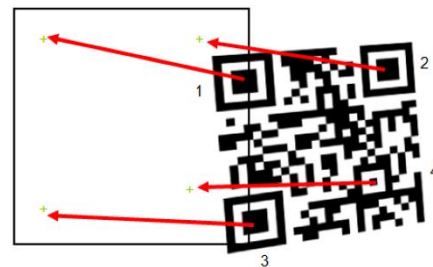


Figura 3.2 Transformada de perspectiva [20]

Luego de aplicar los anteriores pasos se obtiene la matriz de puntos en la cual ya se puede aplicar el algoritmo para decodificar la información del código QR. Para ello se utiliza la librería Zbar que realiza los dos últimos pasos mencionados anteriormente y además aplica el algoritmo de decodificación.

3.4. Desarrollo de código para la decodificación del código QR

En esta parte se mostrarán las funciones o atributos necesarios para la decodificación de códigos QR a partir de la obtención de las imágenes o cuadros en una secuencia de video. Luego de obtener las matrices sin procesar del cuadro actual, se realizan los siguientes pasos:

- Función para transformar la imagen a una en escala de grises
`cv2.cvtColor(Imagen de entrada, código de conversión de color)`
esta función convierte una imagen de entrada en un espacio de color a otro. En el caso de una imagen a color se debe especificar el orden de los canales de color. OpenCV trabaja con el orden BGR y cada uno de los canales pueden tener un rango de 8, 16 o 32 bits.
- Atributo de dimensión de una imagen
`frame.shape` -> devuelve las dimensiones x,y de una matriz
mediante este atributo podemos obtener las dimensiones de una imagen.
- Función para transformar la matriz en un formato para la numeración en Python
`zbar.Image(tamaño x , tamaño y, formato de color, conversión matriz a bytes)`
con el uso de esta función la matriz de números de la imagen es transformada en una matriz con formato adecuado para la librería Zbar.
- Crear objeto escáner de la librería y aplicar el método scan para buscar códigos en la imagen
`zbar.ImageScanner()`
`escaner.scan(imagen en formato Zbar)`
mediante estas funciones es posible crear un objeto que recopile la información necesaria de los códigos presentes en la imagen, además al usar el método scan se aplican los algoritmos de detección y decodificación de los códigos.
- Atributo de datos decodificados
`codigo_qr.data` -> devuelve la información decodificada del código
este atributo contiene la información decodificada luego de usar el método scan.
- Atributo de localización del código
`codigo_qr.location` -> devuelve los vértices del código QR
este atributo contiene las posiciones de los vértices del código QR en la imagen.

Finalmente, se obtiene la información decodificada del código, así como la posición de los vértices del código con los cuales se pueden obtener el tamaño en píxeles en la imagen, la orientación en la imagen y el tamaño real del código QR. Estos datos serán necesarios para posteriormente poder calcular la distancia de la imagen a la cámara.

3.5. Cálculo de distancia entre la cámara y el código QR

Una vez ubicado el código QR en la imagen, es posible extraer información de la geometría del código a partir del procesamiento digital y la información decodificada. En el presente trabajo se utilizará el principio de similitud de triángulos para calcular la distancia entre el sensor de la cámara y el marcador visual, en este caso el código QR. La relación que existe entre la distancia focal – distancia entre el sensor y el lente de la cámara – y la distancia real es igual a la relación entre el ancho (o largo) de un objeto en la imagen y el ancho (o largo) real de la imagen. Entonces se define la relación de la siguiente manera:

$$Distancia\ real = \frac{Constante\ proporcional\ a\ la\ distancia\ focal * Ancho\ Conocido}{Ancho\ en\ Píxeles} \quad \text{Ecuación 3}$$

De igual manera, para hacer uso de la relación presentada anteriormente es necesario aproximar el valor de la constante proporcional a la distancia focal de la cámara utilizada. Para ello es posible hallar este valor experimentalmente mediante el procesamiento a una imagen con los valores de dimensiones conocidos. La imagen de referencia permitirá calcular la distancia focal, esta imagen debe ser capturada de tal forma que se conozca la distancia entre la cámara y el marcador. Por lo tanto, teniendo las dimensiones reales del marcador y calculando la distancia en píxeles en la imagen podemos hallar la distancia focal aproximada. Se muestra a continuación en la figura 3.3 un código QR tomado a una distancia de 30 centímetros. Además, se halla las posiciones de los vértices del código para hallar el ancho en píxeles del código QR.

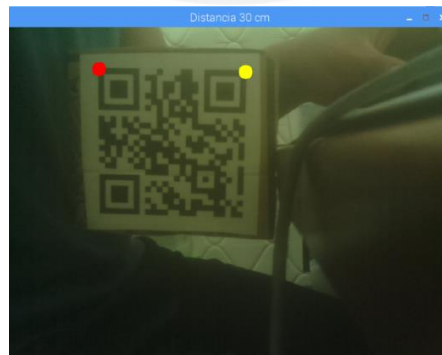


Figura 3.3 Imagen para la calibración de cámara

Una vez encontrada la distancia focal se realiza un proceso similar en el cual se calcula también el ancho en píxeles del marcador y con ello se puede aplicar la similitud de triángulos para encontrar la distancia real al marcador.

Cabe resaltar que la distancia focal encontrada previamente es válida únicamente para el sensor y resolución que se utilizó al capturar la imagen de referencia.

Debido a la forma de calcular la distancia entre el sensor y el lente es importante definir una resolución adecuada para la medición de la distancia. Es por lo que se debe encontrar la relación entre tres factores claves para la aplicación: la resolución de la imagen, la tasa de cuadros por segundo necesaria y la distancia máxima que será posible calcular debido al tamaño del marcador en la imagen. Estos tres parámetros están directamente relacionados entre sí, se entiende que a mayor resolución mayor será la distancia que se pueda detectar; sin embargo, será menor la tasa de cuadros por segundo que se logre en la línea de procesamiento. Entonces, se deberá definir cual resolución es la ideal para la aplicación que se proponga.

Dependiendo de la aplicación, la velocidad de movimiento del VANT proporcionará el valor adecuado de cuadros por segundo necesarios para reconocer satisfactoriamente un marcador. En el siguiente capítulo se verificará la tasa de cuadros por segundo obtenida con el algoritmo de reconocimiento de los códigos junto con el cálculo de distancia al marcador y se comprobará si los resultados obtenidos son adecuados para la decodificación de los códigos en el vuelo. De la misma manera, se busca obtener una distancia de detección del código de al menos 90 cm para lograr un vuelo seguro para el VANT. Para ello se harán las pruebas de decodificación con diferentes resoluciones de imágenes y se escogerá la resolución más adecuada para realizar la navegación.

3.6. Información codificada en el código QR

Como se vio anteriormente es necesario conocer el tamaño del marcador visual para el cálculo de la distancia. Es por lo que el primer dato que se codificará en el código QR es el tamaño del código QR en centímetros. Adicionalmente, es preciso obtener la dirección de vuelo hacia el siguiente código QR. Se codificará la dirección: izquierda o derecha, y también en caso sea necesario un giro en el eje de guiñada del VANT. Finalmente, es útil

saber las coordenadas de los códigos QR en el ambiente en el que se encuentren, por lo que se podrá codificar las coordenadas a las que se encuentra el código QR y mediante ello poder calcular las coordenadas del VANT.

3.7. Hardware para la implementación

La selección del computador embebido para la navegación a bordo de un VANT se hace considerando los criterios mencionados en el capítulo anterior. En el presente trabajo se resalta los criterios de peso y consumo de energía del hardware ya que el vehículo en el que se implemente el módulo tendrá una capacidad de carga limitada y por lo tanto a menor carga mayor será la duración de la batería. Por lo tanto, se debe hacer uso de un computador embebido que cumpla con el procesamiento en tiempo real para el algoritmo propuesto y que además sea suficientemente liviano con el fin de optimizar la duración de la batería para el vuelo.

- Raspberry Pi 3 modelo B

Esta versión de la línea de Raspberry presenta características mejoradas que la antecesora - Raspberry Pi modelo B – la computadora embebida que se comparó en el estudio de Hulens et al [5]. Las características de esta computadora son análogas a las del Odroid U3, por lo que se puede esperar obtener resultados similares para las pruebas hechas en el estudio [5]. La tabla 3.1 muestra las características más relevantes del Raspberry Pi 3.

Tabla 3.1 características del Raspberry PI 3 modelo B [12]

CPU	GPU	Memoria (SDRAM)	Consumo promedio	Dimensiones y peso
1.2GHz 64-bit quad-core ARMv8	400MHz Broadcom VideoCore IV	1GB (compartidos con la GPU)	800mA, (4.0 W)	85.60mm x 53.98mm 45 g

- Raspberry Pi Cámara v1.3

Esta cámara es ideal en la captura y transmisión de video cuando se trabaja con el Raspberry Pi. En el caso de la selección de la cámara es importante tener en cuenta carga de procesamiento que esta supondrá en la demanda del CPU y también el ancho de banda que se tendrá en la transmisión de datos entre el controlador del sensor de la cámara y el

SoC – *System on a chip* – del procesador. Para ello, la Raspberry Pi cámara ofrece una conexión dedicada a la transmisión de datos mediante el protocolo CSI – *Camera Serial Interface*. Esta transmisión de datos ofrece un ancho de banda hasta 2Gbps lo cual es ideal al momento de realizar una captura continua de cuadros. Además, la conexión se hace directamente con el GPU del sistema lo que significa una menor carga de procesamiento en el CPU en las tareas de preprocesamiento de las imágenes. Finalmente, el GPU trabaja con un RTOS – *Real Time Operate System* – capaz de manejar las líneas de los cuadros en tiempo real. Gracias a ello es posible realizar tareas de acondicionamiento del sensor y calibración de los cuadros en otro plano. Entre las tareas que se realizan se tiene: el ajuste de la ganancia del sensor, el tiempo de exposición, el balance de blancos, entre otros [13]. La tabla 3.2 muestra algunas características de la cámara y el proceso que siguen las imágenes al ser llamadas por la librería *picamera* en la figura 3.4.

Tabla 3.2 características del Raspberry PI camara v1.3 [14]

Máxima resolución	Modos de video	Sensor	Dimensiones y peso
5 megapíxeles	1080p30, 720p60 y 640 x 480p60/90	OmniVision OV5647	25 x 24 x 9 mm 3g

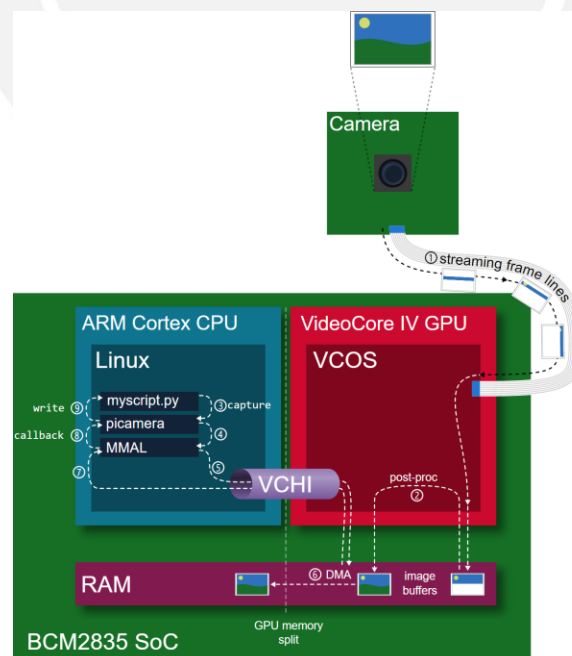


Figura 3.4 Línea de manejo de imágenes [13]

- Pixhawk

El computador abordo es el encargado del procesamiento de las imágenes y de la generación de los comandos de vuelo. Sin embargo, también es preciso contar con un controlador de vuelo encargado de traducir los comandos de vuelo en señales PWM para el control de los motores sin escobillas de la aeronave. Este controlador de vuelo necesita información de los parámetros de vuelo tales como: velocidad, inclinación, y orientación; por lo tanto, es preciso que cuente con sensores como: acelerómetros, giroscopios y magnetómetros. La tabla 3.3 presenta las características del controlador de vuelo Pixhawk.

Tabla 3.3 características Pixhawk [15]

CPU	Memoria	Sensores	Dimensiones y peso
32-bit ARM Cortex M4	168 MHz/256 KB RAM/2 MB Flash	MPU6000 – acelerómetro y giroscopio principales ST Micro 16-bit giroscopio ST Micro 14-bit acelerómetro/compas MEAS barómetro	Ancho, alto y largo: 50x15.5x81.5 mm Peso: 38 g

3.8. Configuración y modos de vuelo en el Pixhawk

En esta sección se verán los criterios que se deben tener en cuenta en la configuración del Pixhawk, las pautas para poder realizar el vuelo en interiores y los modos de vuelo adecuados para un contexto en el cual no es posible usar la señal GPS.

El controlador de vuelo realiza una serie de pruebas a los sensores y puertos de comunicación para verificar que funcionen correctamente antes de iniciar un vuelo. Algunos de los elementos que se verifican son: receptor de radio control, barómetro, compas, GPS, acelerómetros y giroscopios. Luego de verificar que funcionen correctamente el vehículo puede ser armado para que inicie el vuelo. Sin embargo, en el caso del vuelo dentro de interiores es necesario desactivar las verificaciones de la señal satelital GPS para que el vehículo pueda ser armado correctamente.

En el caso de la navegación en interiores la señal satelital GPS no puede dar la información de altitud y el barómetro posee un error considerable; por lo tanto, es necesario tener un sensor que mida la altitud del VANT con una precisión mayor. Debido a ello se utilizará un sensor de distancia LIDAR. Para el presente trabajo se utilizará el sensor Lightware SF30/B, el cual puede enviar la información de distancia mediante comunicación serial o mediante un voltaje proporcional a la distancia (analógico). En el presente trabajo se utilizará la salida analógica con el conversor análogo – digital de 3.3 V del Pixhawk. La tabla 3.4 muestra las principales características del sensor:

Tabla 3.4 características Lightware SF30/B [16]

Rango medición	Precisión	Tipo de salida	Ratio de actualización	Dimensiones y peso
0 – 50 m	±0.10 m	Serial y analógica	18317 lecturas por segundo (máximo)	30 x 56.5 x 50 mm 35 g

Luego de definir los elementos necesarios para el vuelo del VANT en interiores, es preciso definir los modos de vuelo que se utilizarán en el contexto. Los modos de vuelo que no necesitan de la señal GPS son: Stabilize, Alt Hold, Acro, Sport, Land y Guided NoGPS [17]. Sin embargo, los modos de vuelo que se utilizarán serán los dos últimos. A continuación, se explicará porque se prefiere estos modos de vuelo.

- Guided NoGPS

Este modo es una variación del modo Guided [17] con la diferencia que no es necesario el uso del GPS para el control del vehículo, lo cual lo hace ideal para el vuelo en interiores. A diferencia del modo Guided, en el cual se controla al VANT mediante el control de posición o velocidad, este modo solo acepta comandos de control de los ángulos de inclinación del VANT. Los ángulos de inclinación son tres: alabeo, cabeceo y guiñada. El alabeo es el ángulo que se forma con el eje de la aeronave que va desde la cabeza a la cola del mismo, este ángulo permite el movimiento hacia la izquierda o derecha. El cabeceo es ángulo de inclinación de la cabeza de la aeronave; cuando esta mira hacia arriba o hacia abajo, es decir, permite al vehículo moverse para adelante o hacia atrás. Y finalmente la guiñada es el ángulo que se obtiene del eje perpendicular al plano en el que se encuentra la aeronave, este ángulo permite cambiar de orientación la orientación del VANT. Entonces, mediante el

control de estos ángulos y el tiempo en el cual se realiza la inclinación, se puede realizar el control de la posición del vehículo.

- Land

Este modo de vuelo se utiliza para el aterrizaje del VANT, para ello se puede definir el parámetro de velocidad de aterrizaje el cual es por defecto: 50 cm/s. Al llegar a tierra el vehículo se apagará automáticamente y los motores se desarmarán si las velocidades de los mismos llegaron al mínimo. Sin embargo, este modo solo controla la posición vertical por lo que la posición horizontal se podrá controlar mediante el control de los ángulos de cabeceo y alabeo.

3.9. Comunicación entre la computadora y el controlador de vuelo

La comunicación entre el computador embebido – Raspberry Pi 3 – con el controlador de vuelo – Pixhawk es necesaria para lograr el vuelo autónomo del VANT. Para tal propósito se usará la interfaz de programación de aplicaciones – Dronekit, la cual permite crear aplicaciones que se ejecutan en computadoras embebidas y comunicar la computadora con el controlador de vuelo mediante la traducción de código en Python a mensajes con el protocolo de comunicación para vehículos aéreos - MAVLINK. A continuación, se explica la función de la librería Dronekit y el protocolo de comunicación MAVLINK.

- Dronekit

Este API permite realizar scripts que se comuniquen con el vehículo mediante comandos de MAVLINK. Gracias a ello es posible controlar los movimientos y la operación del vehículo, y además se puede recibir información de los parámetros y el estado del mismo. Asimismo, es posible la obtener información de los sensores conectados al controlador de vuelo y poder utilizar esta información para el control del VANT.

- MAVLINK

Es una librería de mensajes de encabezado que se utiliza para mandar comandos al controlador de vuelo y recibir información de este. Para ello se hace uso de paquetes de estructuras escritas en lenguaje C que pueden ser transmitidas mediante canales seriales con una gran eficiencia.

La comunicación física entre el controlador de vuelo y la computadora se realizará mediante comandos MAVLINK sobre una conexión serial entre ambos dispositivos. Para ello es necesario configurar el protocolo de comunicación serial en el controlador de vuelo y la tasa de baudios que tendrá dicha conexión.

Mediante las herramientas mostradas previamente es posible comunicar y controlar el Pixhawk con el Raspberry Pi 3 de forma en que sea posible el control del VANT en tiempo real. Gracias a ello se puede implementar el método de navegación en interiores propuesto previamente. Sin embargo, para realizar la implementación de la forma más óptima posible es necesario realizar las tareas de procesamiento de imágenes y navegación del VANT de forma simultánea. Por tal motivo en la siguiente sección se mostrará una alternativa para la ejecución de múltiples tareas.

3.10. Hilos: programación para múltiples tareas

En esta sección se explicará la técnica con el cual se puede realizar las diferentes tareas necesarias para la navegación del VANT. El método que se utilizará para alcanzar tal propósito es el uso de multi hilos. Estos permiten ejecutar múltiples funciones dentro del mismo proceso. Para el presente caso se usará el módulo *threading* en Python.

Las ventajas que se logran con una implementación paralela son significativas puesto que hay tareas que no deben esperar a que otras terminen de ejecutarse. Además, gracias a los multi hilos es posible sacar provecho de los 4 núcleos que posee la Raspberry Pi 3, ventaja que no se obtiene con una programación en serie.

Las tareas que se realizarán en paralelo son: captura de cuadros desde la cámara, procesamiento de cada cuadro recibido, obtención de parámetros del VANT y envío de comandos de vuelo al VANT. Por lo tanto, las tareas que requieran una operación continua; como es el caso del envío de comandos de vuelo, se deben hacer cada cierto periodo de tiempo para asegurar el correcto funcionamiento. Además, gracias al procesamiento en paralelo no se afectará la realización de las demás tareas como la obtención y el procesamiento de imágenes.

Finalmente, se tiene las herramientas necesarias para la implementación del módulo que permita la navegación en interiores. No obstante, antes de realizar una implementación física, es recomendable realizar simulaciones de vuelo. El software de simulación permite tener todos los componentes físicos del VANT en un entorno virtual en el cual se puede realizar pruebas con el computador embebido. Además, es posible recrear los espacios en los cuales se realizarán las pruebas, así como también los parámetros físicos tales como viento o fuerza de gravedad. En la siguiente sección se detallará más acerca del software de simulación.

3.11. Simulador SITL – Software in the loop

El simulador mostrado en la figura 3.5 es capaz de probar el comportamiento de los vehículos sin necesidad de tener algún controlador de vuelo. El simulador puede ejecutarse en cualquier computadora sin el uso de algún otro hardware debido a que el Firmware del controlador de vuelo tiene la versatilidad de instalación en múltiples plataformas.

Cuando el simulador está ejecutándose la información de los sensores proviene de un modelo de vuelo dinámico de algún simulador de vuelo. En este caso se utilizará el simulador Gazebo.

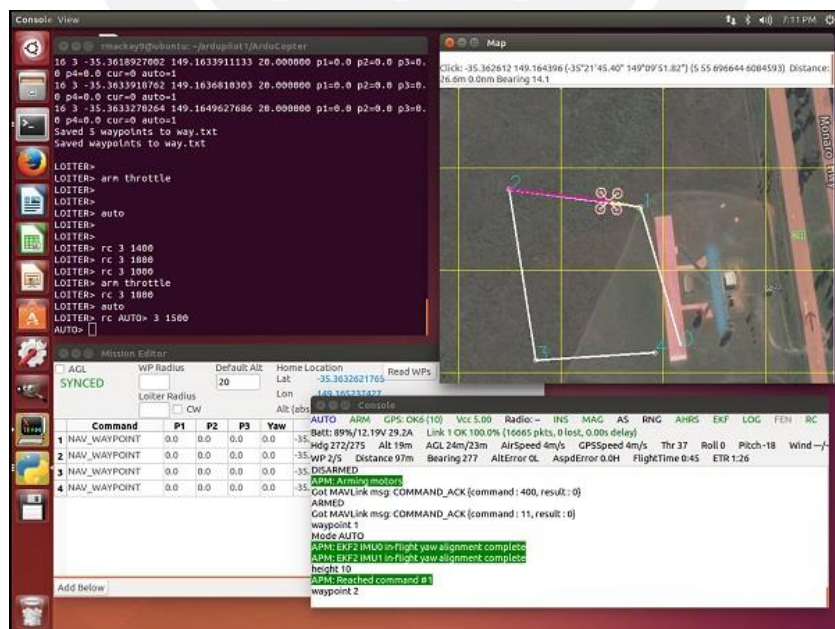


Figura 3.5 Simulación de vuelo – SITL [18]

- Gazebo

Es un simulador de robots que posee diferentes herramientas para probar algoritmos, y el rendimiento de los mismos. El sistema ofrece escenarios realistas y es capaz de simular ambientes interiores con versatilidad. En el presente diseño se simulará el espacio de un laboratorio de forma rectangular el cual posee ventanas y puertas en las paredes. La figura 3.6 muestra el entorno simulado en Gazebo.

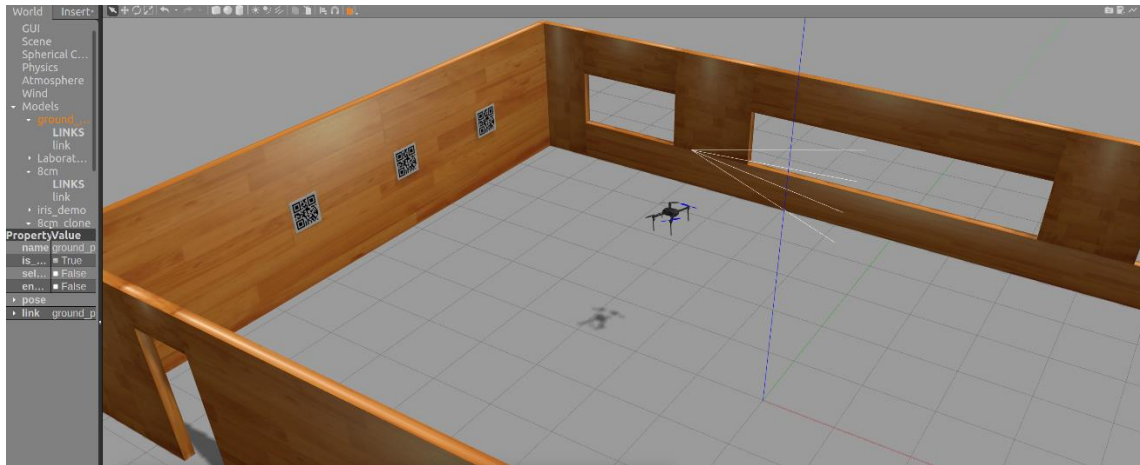


Figura 3.6 Ambiente de laboratorio simulado en Gazebo

El uso del simulador SITL es muy similar al uso del vehículo real. Es posible la conexión del vehículo simulado a diferentes estaciones de control que pueden ser una laptop o también el computador embebido. Además, es posible despegar, cambiar de modo de vuelo, cargar misiones o aterrizar tal como se haría en el vehículo real. No obstante, la principal diferencia es la posibilidad de configurar parámetros de la simulación como, por ejemplo: la velocidad y dirección del viento, la posible falla del GPS, la falla del receptor de radio control, entre otros. Adicionalmente, es posible añadir parámetros para hacer uso de sensores virtuales como un LIDAR o un sensor de flujo óptico. A continuación, se mostrará los parámetros que se deben tomar en cuenta para poder simular la navegación en interiores.

Los parámetros son modificados mediante comandos insertados en consola o línea de comandos del simulador.

- Prueba de fallo de GPS para interiores

En este caso podemos simular un fallo en el GPS para obtener las mismas condiciones de un espacio interior, para ello se debe cambiar el siguiente parámetro:

```
param set SIM_GPS_DISABLE 1
```

- Añadir un sensor virtual de distancia

Es posible simular un sensor analógico de distancia para la medición de la altura del VANT. Para ello se puede definir: el rango máximo de las mediciones, el rango de voltaje en el que trabaja el sensor, y la escala/resolución de medición que se tendrá. En este caso se usará un rango máximo de 30 metros, un rango de voltaje de 0 a 5 V y una escala de 10. Los parámetros que se deben modificar son los siguientes:

```
param set SIM_SONAR_SCALE 10
param set RNGFND_TYPE 1
param set RNGFND_SCALING 10
param set RNGFND_PIN 0
param set RNGFND_MAX_CM 3000
```

- Abrir una conexión UDP para la comunicación del simulador con el Raspberry Pi 3
- El simulador puede conectarse con el Raspberry Pi 3 mediante el envío y recepción de paquetes UDP en una red local. Para ello se debe especificar la dirección IP del computador – en este caso el Raspberry Pi 3 y también el puerto UDP que se quiere abrir. El parámetro que se debe introducir es:

```
output add 192.168.14.82:14550
```

donde la dirección IP depende de la red en la que se encuentre y el puerto que se desee elegir.

En el siguiente capítulo se mostrarán los resultados de las pruebas realizadas para la navegación real y simulada. Además, se verificará si los resultados fueron los que se esperaban.

CAPÍTULO 4

SIMULACIONES Y RESULTADOS

En este capítulo se mostrarán los resultados de las pruebas hechas en el procesamiento de imágenes y la navegación del VANT en un entorno simulado, así como también en pruebas reales. En el caso del procesamiento de imágenes de códigos QR se verificará la distancia máxima a la que pueden ser detectados, se comprobará la tasa de cuadros por segundo alcanzada con el algoritmo y se verán las optimizaciones realizadas para alcanzar una mayor tasa. Las simulaciones comprobarán la efectividad de la navegación en un entorno similar al real, pero sin el riesgo de algún accidente físico. Además, se podrá verificar la conectividad entre el Raspberry Pi 3 y el controlador de vuelo simulado. Se comprobará la navegación autónoma con el procesamiento de códigos QR. Asimismo, se verá las pruebas para la configuración correcta del LIDAR con salida analógica.

4.1. Resultados de la detección y decodificación de un código QR en una imagen

En esta etapa se mostrarán los resultados obtenidos de la detección y decodificación de los códigos QR mediante el uso del siguiente software: Sistema operativo – Raspbian

(distribución de GNU/Linux), lenguaje de programación – Python, librerías – OpenCV y Zbar; además del siguiente hardware: Raspberry Pi 3 y cámara Raspberry.

En primer lugar, se definen tres resoluciones de las imágenes con las cuales se trabajarán: 320x240, 640x480 y 1296x736 píxeles, las cuales corresponden a las figuras 4.1, 4.2 y 4.3 respectivamente. A continuación, se muestra las imágenes tomadas con la cámara Raspberry en una posición estática a una escala de 1/3:



Figura 4.1 Resolución 320x240

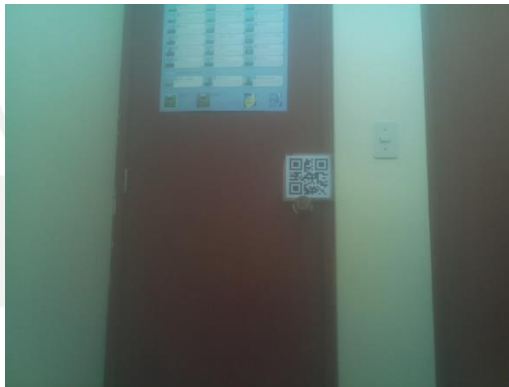


Figura 4.2 Resolución 640x480

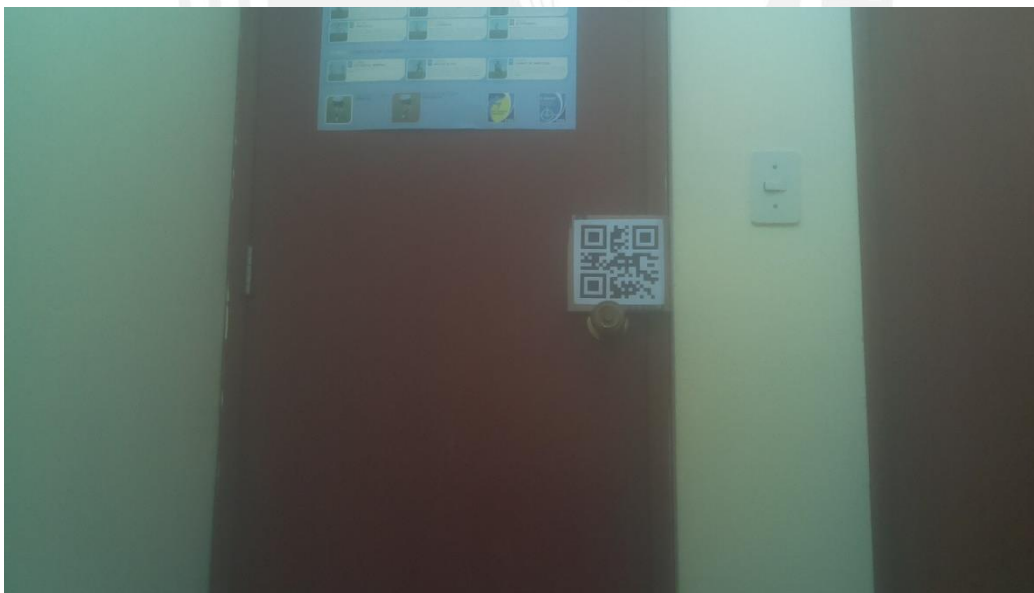


Figura 4.3 Resolución 1296x736

La primera prueba que se realizó es una captura continua de cuadros a las resoluciones mencionadas anteriormente. En esta prueba se hizo una medición aproximada de la tasa de cuadros por segundo que se puede conseguir capturando imágenes y mostrándolas en

pantalla. Para poder lograr la medición se realizó un programa que tomó una cantidad establecida de cuadros, en este caso 10 cuadros, y se midió el tiempo entre el inicio de la toma del primer cuadro hasta el fin de la toma del décimo cuadro. Entonces, se divide los 10 cuadros entre la cantidad de segundos transcurridos en la medición para aproximar la tasa de cuadros por segundo que se obtienen. Se hizo la toma de 10 cuadros 25 veces, con lo que se halló la tasa de FPS promedio para dichas condiciones. La tabla 4.1 muestra los resultados para cada tamaño de imagen:

Tabla 4.1 FPS promedio para la toma y muestra de cuadros

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
FPS promedio	31.4	12.7	6.3

Sin embargo, al realizar la misma prueba sin la impresión de la imagen en pantalla la tasa de FPS aumenta en comparación con la prueba anterior. Esto debido a que se requiere un tiempo de procesamiento para imprimir cada pixel en pantalla, con lo cual a mayor cantidad de píxeles mayor será el tiempo que le tome al procesador para imprimir todos los píxeles. Es por lo que se puede obtener un aumento de hasta dos cuadros por segundo más en resoluciones más grandes que en resoluciones más pequeñas en donde el aumento solo es de hasta un cuadro por segundo. Los resultados se muestran en la tabla 4.2.

Tabla 4.2 FPS promedio para la toma de cuadros

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
FPS promedio	32.2	13.9	8.4

No obstante, la cantidad de cuadros por segundo que se pueden procesar con el algoritmo de detección y decodificación de los códigos QR es menor debido al tiempo de procesamiento que se tiene entre la captura del cuadro hasta la decodificación del mismo. Además, al igual que las pruebas anteriores se tiene que a mayor resolución menor es la cantidad de cuadros por segundo que se pueden procesar, siendo una relación directa mas no lineal para este caso. La tabla 4.3 presenta los resultados de la prueba del algoritmo de decodificación con la muestra de los cuadros en pantalla.

Tabla 4.3 FPS promedio aplicando el algoritmo de decodificación y la impresión de los cuadros

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
FPS promedio	13.4	4.0	1.7

De la misma forma que en el caso anterior, al evitar la impresión de los píxeles en pantalla se logra un aumento en la en la tasa de cuadros por segundo procesado. La tabla 4.4 muestra los resultados de las pruebas del algoritmo de decodificación sin la impresión de cuadros.

Tabla 4.4 FPS promedio aplicando el algoritmo de decodificación

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
FPS promedio	14.0	5.7	2.6

No obstante, se puede obtener una mejora en la tasa de cuadros por segundo al ejecutar el programa sin la presencia de la interfaz gráfica de usuario del sistema operativo (GUI), es decir al ejecutar el programa directamente de la línea de comandos sin procesos de la interfaz gráfica que se ejecuten en segundo plano. La tabla 4.5 muestra las mejoras del caso.

Tabla 4.5 FPS promedio aplicando el algoritmo de decodificación sin interfaz gráfica

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
FPS promedio	20.1	7.2	3.1

Cabe resaltar que el error de las pruebas es de +/- 1.5 FPS debido a factores de pre procesamiento y medición de los tiempos. Además, se puede inferir que la relación entre resolución y la tasa de FPS es directa pero no lineal. Se puede observar los resultados y la relación entre ellos en las figuras 4.4, 4.5 y 4.6:

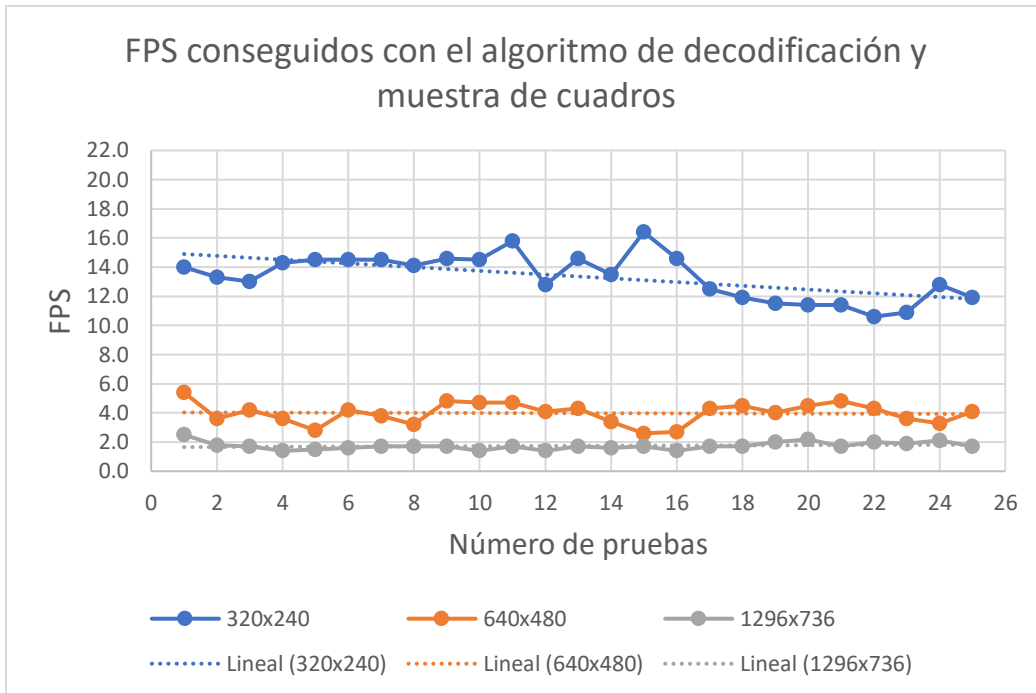


Figura 4.4 Pruebas de FPS aplicando el algoritmo de decodificación e impresión de cuadros

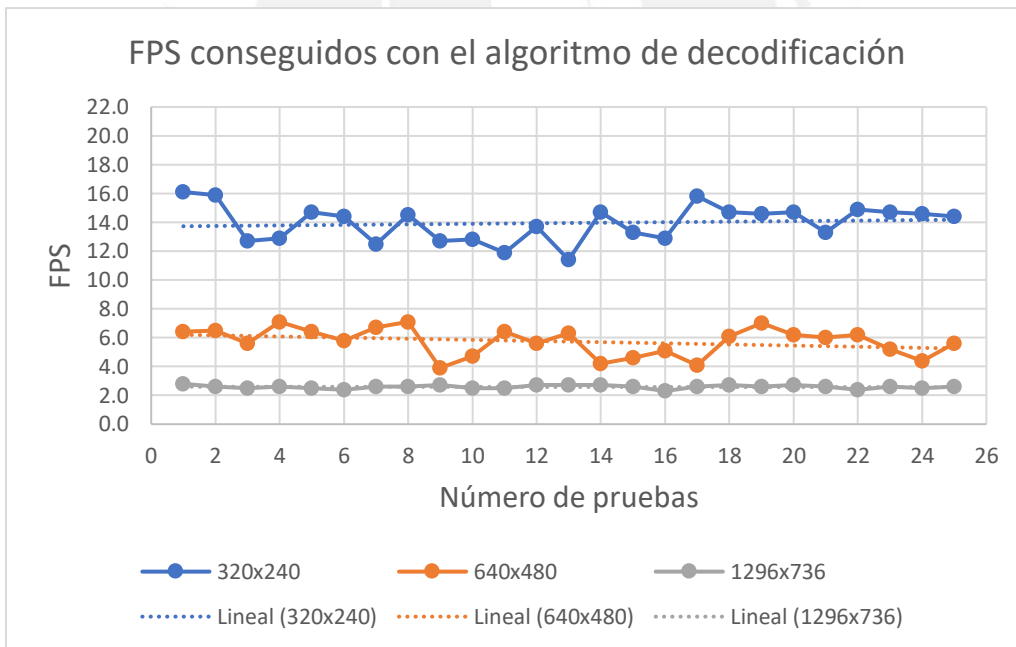


Figura 4.5 pruebas de FPS aplicando el algoritmo de decodificación

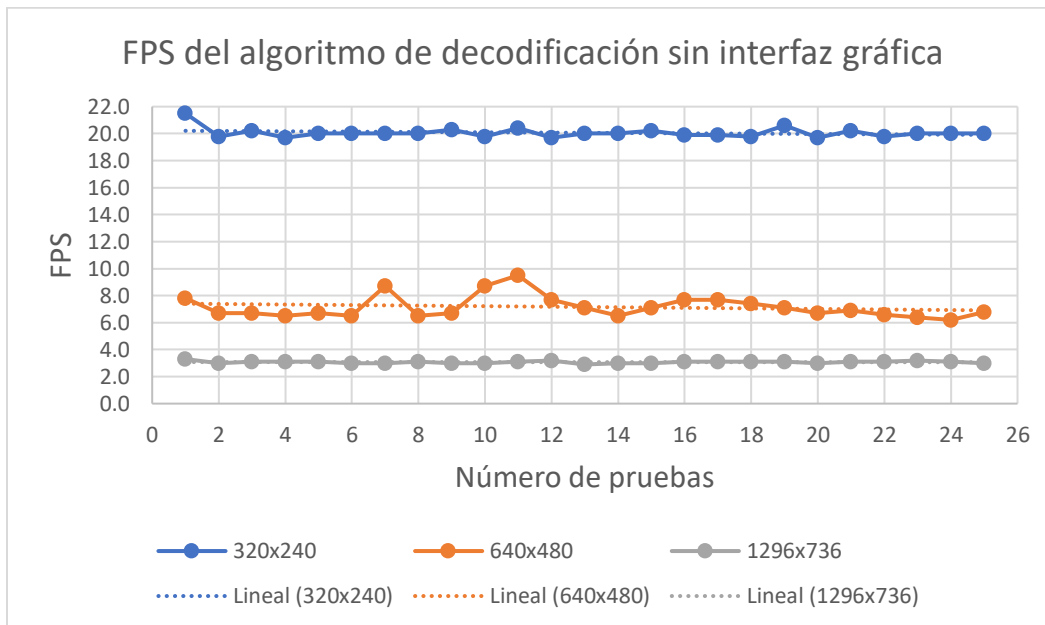


Figura 4.6 pruebas de FPS aplicando el algoritmo de decodificación sin la interfaz gráfica de usuario

En los gráficos se puede observar la separación entre las líneas de tendencia las cuales muestran una separación mayor entre la resolución 320x240 y 640x480 lo cual indica que la relación entre la tasa de cuadros por segundo y la cantidad total de píxeles no es lineal, pero si mantiene una tendencia definida. Además, se puede observar que el gráfico de las pruebas sin la interfaz gráfica de usuario, los resultados tienen menos variación que en los anteriores casos.

4.2. Resultado del método de medición de distancia del código QR a la cámara

Se toma las mediciones con las tres resoluciones mostradas anteriormente, para ello es necesario calcular los valores de las constantes proporcionales a la distancia focal para cada resolución. Es por eso que se debe realizar el proceso de calibración para los tres casos considerados anteriormente. La tabla 4.6 muestra los valores obtenidos para las constantes.

Esta calibración es aproximada puesto que se debe medir la distancia a la que se toma la primera imagen de forma manual y según esa distancia calcular el valor de la constante mediante el ancho del objeto real y el ancho en píxeles calculado en la imagen.

Tabla 4.6 constantes proporcionales a la distancia focal

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
Constante proporcional a la distancia focal	315	630	1260

Luego de tener los valores de las constantes se realiza el mismo proceso de medición con la diferencia de que teniendo el valor constante ya es posible calcular la distancia real al objeto con los demás valores conocidos. Las pruebas de medición de distancia fueron realizadas en las mismas condiciones de iluminación y calibración de los parámetros de captura de la cámara: exposición, brillo, velocidad de obturador, entre otros. Luego de realizar pruebas con códigos QR de: 8, 10 y 12 centímetros, se decidió hacer uso de un código QR de 12 centímetros de longitud debido al balance necesario entre la distancia máxima de detección y la distancia mínima de detección. Se debe aclarar que mientras más grande sea el código QR existirá una distancia mínima para que este entre en el cuadro de la imagen. En seguida se muestra las distancias mínimas y máximas que se lograron en las pruebas.

Tabla 4.7 distancias máximas y mínimas posibles en la medición

Resolución o tamaño de imagen – píxeles	320x240	640x480	1296x736
Distancia máxima (cm)	86	157	193
Distancia mínima (cm)	17	17	20

Se debe aclarar que las condiciones de iluminación en la que se realizan las pruebas son importantes para mejorar la detección de los códigos QR. Es decir que mientras la iluminación sea mejor el ruido en la imagen será menor con lo cual será posible detectar a una mayor distancia el código QR, pudiendo aumentar el rango de distancias hasta en un 10%. Además, la cantidad de píxeles que tiene la imagen es proporcional a la distancia máxima que se puede detectar el código, así como también el tamaño del código QR. Las figuras 4.7, 4.8, 4.9 y 4.10 muestran la verificación de la decodificación de la información para el cálculo de la distancia hacia la cámara.

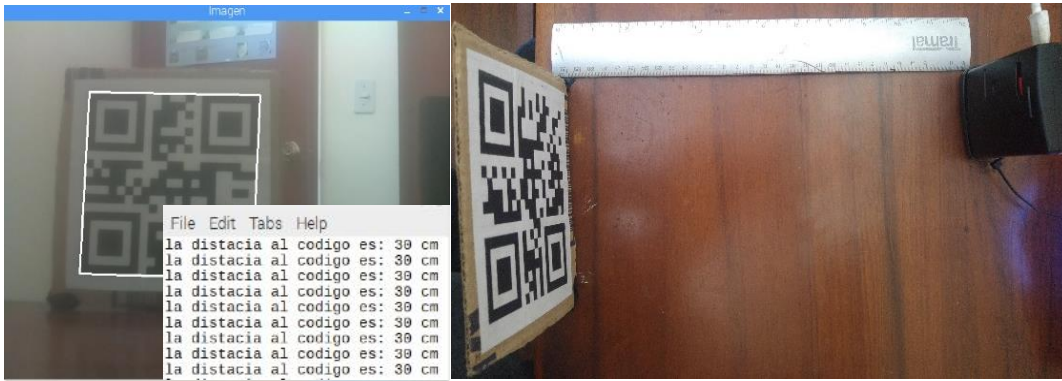


Figura 4.7 verificación de distancia a 30 cm de la cámara



Figura 4.8 medición distancia con imagen 320x240

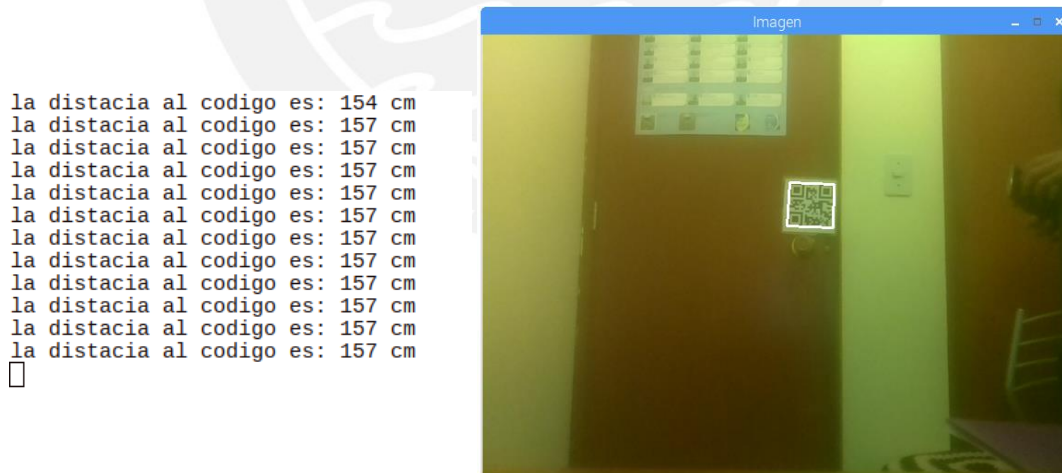


Figura 4.9 medición distancia con imagen 640x480

```
la distancia al codigo es: 210 cm
la distancia al codigo es: 193 cm
la distancia al codigo es: 184 cm
la distancia al codigo es: 159 cm
la distancia al codigo es: 159 cm
la distancia al codigo es: 180 cm
la distancia al codigo es: 180 cm
la distancia al codigo es: 182 cm
la distancia al codigo es: 182 cm
la distancia al codigo es: 182 cm
la distancia al codigo es: 180 cm
la distancia al codigo es: 175 cm
la distancia al codigo es: 177 cm
la distancia al codigo es: 175 cm
la distancia al codigo es: 180 cm
la distancia al codigo es: 182 cm
la distancia al codigo es: 180 cm
```

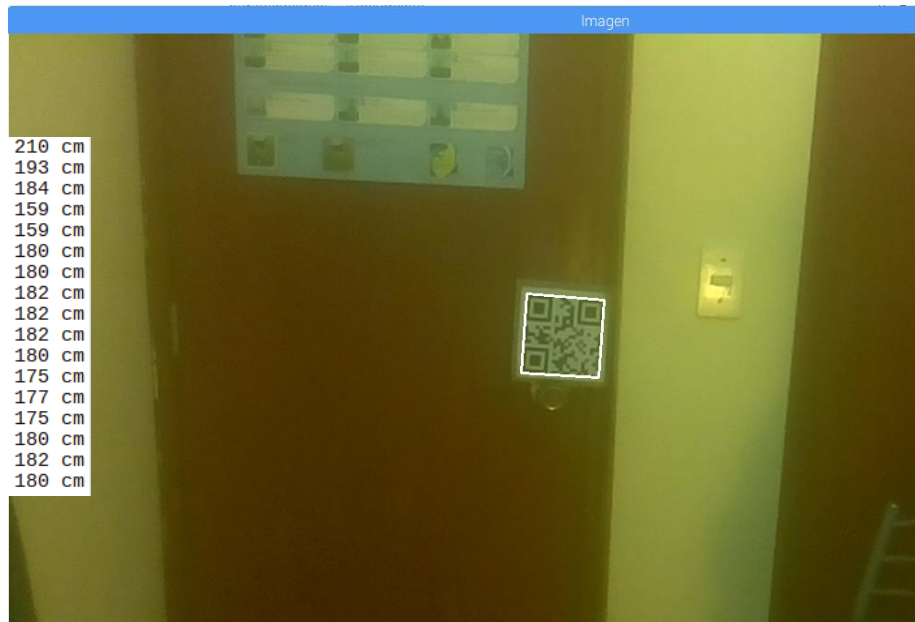


Figura 4.10 medición distancia con imagen 1296x736

Luego de tener los resultados de las tasas de cuadros por segundo y las distancias máximas alcanzadas con las distintas resoluciones, se debe escoger el balance adecuado entre el rendimiento de cuadros por segundo que se desea obtener y la distancia máxima de detección que se desea lograr. Para los objetivos que se proponen las resoluciones de 320x240 y 640x480 son las más indicadas con el hardware seleccionado. Sin embargo, se necesita hacer las pruebas con la cámara montada en el VANT y este en movimiento para lograr asegurar que los códigos puedan ser detectados con eficiencia.

En las figuras 4.11 y 4.12 se muestran los resultados de las pruebas para determinar la eficiencia de la medición de distancia. Estas muestran el porcentaje de cuadros que pudieron ser decodificados exitosamente. Se puede observar que a medida que la distancia aumenta el porcentaje de aciertos disminuye drásticamente a partir de un valor límite. Además, cuando las pruebas se realizaron en vuelo se tuvo menos porcentaje de aciertos debido al movimiento y vibración que se presenta en este caso. También se considera que la distancia mínima prudente que debe existir entre el código y el VANT es de 70 cm puesto que la medición de la distancia es desde el centro del vehículo por lo que se debe tomar en cuenta las medidas de las hélices del mismo.

Así mismo, se obtiene que el error de la medición de distancia es de +/- 2 cm por lo que no influye en el objetivo de mantener la distancia prudencial al código QR.

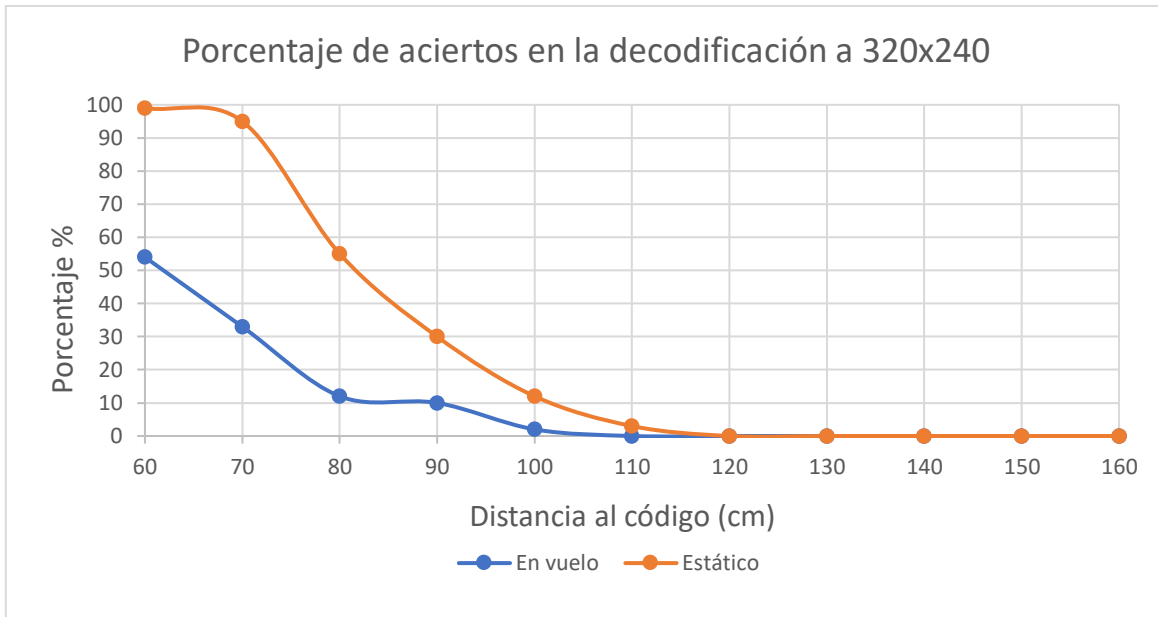


Figura 4.11 Porcentaje de aciertos en la decodificación a 320x240

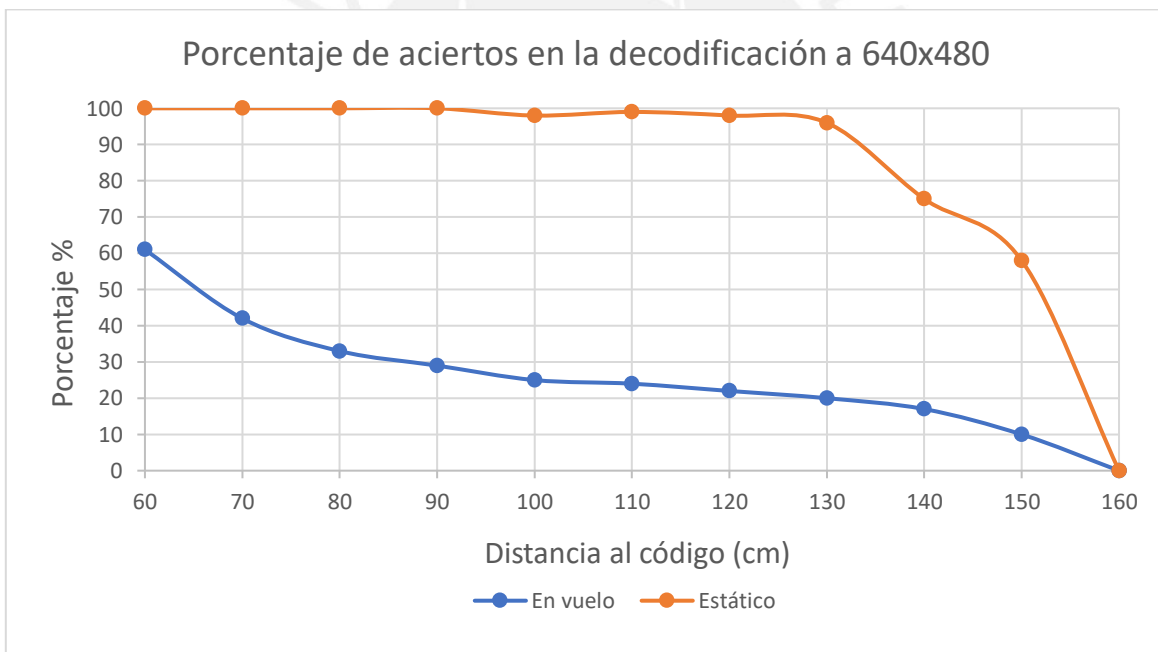


Figura 4.12 Porcentaje de aciertos en la decodificación a 640x480

4.3. Calibración y medición con el LIDAR

Las mediciones con el LIDAR Lightware SF30 se realizaron mediante la salida analógica que posee el sensor y se leyeron mediante el convertor análogo – digital de 3.3V del Pixhawk. Para tal fin es necesario ajustar los parámetros del ADC. En primer lugar, se debe ajustar el valor de offset a 0.08 voltios, el valor en el cual se mide una distancia cero con el

conversor. Luego, se necesita ajustar la ganancia del conversor a 3.3 Metros/Voltios para tener una escala adecuada con respecto a la configuración del sensor. En el caso del LIDAR es necesario modificar algunos parámetros como la ratio de actualización del puerto analógico el cual tiene una relación directa con la precisión del sensor. En este caso al ser más importante la precisión de la medición que la frecuencia en la cual se actualiza las mediciones se escogió una ratio de 1665 mediciones por segundo, lo cual nos permite tener una precisión de hasta 3 cm. Además, es necesario seleccionar el máximo rango analógico al que trabajará el sensor. En este caso al tratarse de una aplicación en interiores se requiere una medición de altitud de hasta 3 metros para la mayoría de los casos. Por lo tanto, se escogió el menor rango de valores: hasta 8 metros de distancia para un rango de 2.56 Voltios a 0.08 Voltios.

4.4. Simulación del vuelo con el software Gazebo

Antes de implementar el módulo en el mismo vehículo se hicieron simulaciones de vuelo con un vehículo virtual en un ordenador como se muestra en la figura 4.13. Las partes del vehículo simuladas fueron: el vehículo, el controlador del VANT (Pixhawk), el sensor LIDAR y el ambiente. El ordenador se comunica con el Raspberry Pi vía conexión UDP y procesa las imágenes capturadas con la cámara. Entonces, gracias a la simulación se puede verificar la correcta comunicación para enviar comandos al controlador de vuelo luego del procesamiento de imágenes con el Raspberry como se muestra en la figura 4.14.

En las simulaciones se hicieron pruebas de vuelo en las cuales mediante la decodificación de diferentes códigos QR se pudo realizar un plan de vuelo con el VANT virtual. Además, se verificó que el procesamiento de las imágenes se realizó en paralelo al envío de comandos al ordenador que en este caso era el controlador de vuelo.

Sin embargo, al realizar la navegación del VANT se comprobó que la estabilidad del vuelo no era la adecuada, puesto que la inercia del movimiento no permite mantener la posición del VANT. Debido a este problema se optó por simular un sensor óptico de flujo para garantizar el control inercial del vehículo. Gracias a ello fue posible mantener la posición del vehículo luego de recibir comandos de vuelo. Entonces, con la estabilización de posición y los comandos de navegación decodificados de los códigos QR se pudo realizar un plan de vuelo en la siguiente forma: el VANT despegar hasta una altura adecuada (1.5 m),

mantiene la altura y posición por unos segundos, avanza hasta encontrar un código QR, luego mantiene la posición hasta decodificar la información del código, responde al comando decodificado y vuela hasta el siguiente código, finalmente llega hasta el marcador que con el comando de aterrizaje y termina su vuelo.

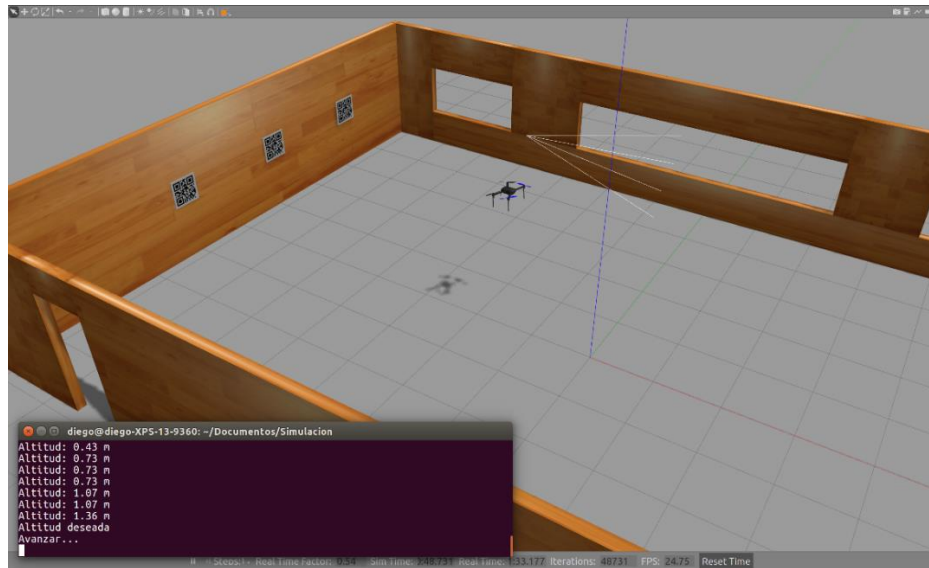


Figura 4.13 simulación de vuelo en Gazebo

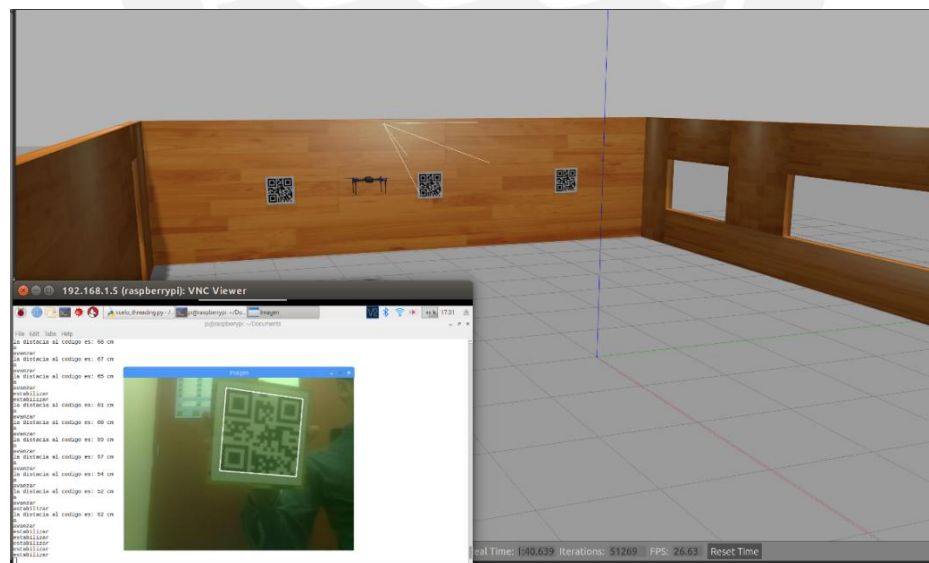


Figura 4.14 Simulación envío de comandos a partir de los códigos QR

4.5. Implementación del módulo

En la implementación del módulo se realizó las conexiones de los diferentes componentes. En principio, se la conexión entre el Raspberry Pi 3 y el Pixhawk se hizo mediante conexión serial. Luego se conectó el LIDAR al ADC de 3.3v del Pixhawk. Además, se utilizaron dos diferentes reguladores de voltaje para alimentar al Raspberry y al Pixhawk gracias a una batería LiPo. El VANT que se utilizó es un cuadricóptero con hélices ubicadas en forma de x. Para ello se utilizó también 4 motores sin escobillas y un controlador de velocidad electrónico (Electronic speed control - ESC). En la figura 4.15 se puede observar la distribución de los componentes del módulo en el VANT.

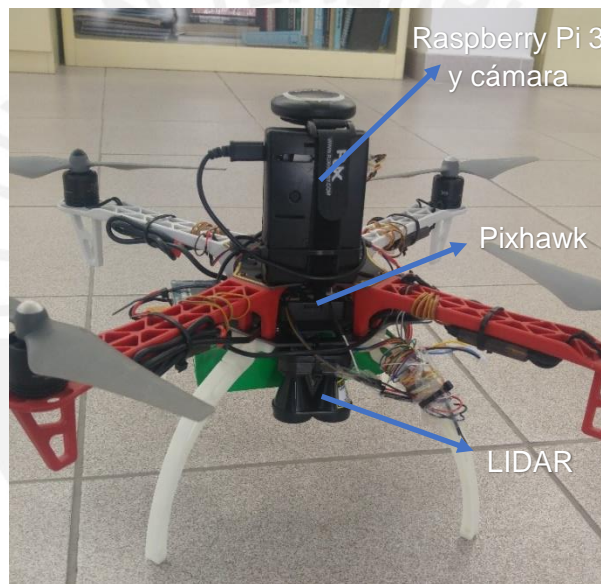


Figura 4.15 distribución componentes en el VANT

Las pruebas que se hicieron con la implementación del módulo mostraron resultados similares a los resultados de las simulaciones. Sin embargo, al no contar con una realimentación de la posición y velocidad horizontal, no fue posible la estabilización del VANT en la implementación. La decodificación de comandos y la respuesta en tiempo real también se logró en la implementación. Es decir, el VANT pudo responder a los comandos decodificados en los códigos QR.



Figura 4.16 Prueba de vuelo en exteriores



Figura 4.17 Prueba de vuelo en interiores

En las pruebas en exteriores como la mostrada en la figura 4.16, también fue posible comprobar los comandos de vuelo decodificados a partir de la decodificación de códigos QR. En estas pruebas en comparación a las pruebas en interiores como la mostrada en la figura 4.17, se pudo diferenciar la falta control de estabilización inercial para realizar el plan de vuelo a partir de los comandos de vuelo. Entonces, ante la ausencia de una señal de localización es necesario optar por una alternativa como un sensor de flujo óptico para lograr estabilizar al vehículo luego de cada tramo recorrido de la ruta de vuelo. Finalmente, los resultados de las pruebas lograron demostrar que es posible la decodificación de los códigos QR para velocidades de vuelo cercanas a la mínima requerida.

Conclusiones

La navegación autónoma de un VANT en espacios interiores puede ser realizada mediante procesamiento digital de imágenes de marcadores visuales como en este caso códigos QR. Este método ofrece versatilidad en su implementación y la posibilidad de realizar el procesamiento de manera embebida en el módulo electrónico gracias al algoritmo utilizado.

En cuanto a las técnicas de procesamiento digital de imágenes, se logró entender y aplicar diversas técnicas para la detección y decodificación de códigos QR. Se estudió el proceso de decodificación desde la obtención de la imagen original hasta la transformación de cada cuadrado del código en un valor de un bit (0,1). Además, se pudo implementar estas técnicas gracias a un lenguaje de programación como Python el cual ofrece eficiencia para los requerimientos.

Por otro lado, se pudo integrar las características de los códigos QR tales como: codificación de datos, decodificación rápida, capacidad de corrección de errores, entre otros; al método de navegación que se propuso. Gracias a la capacidad de codificación de información se pudo automatizar el método de medición de distancias puesto que el código QR brinda el dato del tamaño del mismo y la información del plan de vuelo del VANT. Adicionalmente, la rápida decodificación hizo posible el uso de los códigos QR para una aplicación en tiempo real en la cual se pueda obtener la información en el momento del vuelo.

Se desarrolló un módulo para la navegación autónoma de un VANT con siguientes componentes principales: Raspberry Pi 3, Pixhawk y sensor LIDAR. Estos componentes satisfacen los requerimientos de tamaño y peso necesarios para la navegación en interiores. Por otra parte, se llegó a un balance entre la distancia máxima de detección de un código QR y la resolución adecuada para obtener un tiempo de procesamiento eficiente para la respuesta en tiempo real. Para ello se eligió que la resolución apropiada es 640x480, puesto que permite un equilibrio entre la tasa de cuadros por segundo conseguida al aplicar el algoritmo y la distancia máxima a la que se puede detectar el código QR en un espacio interior. Cabe resaltar que se hicieron optimizaciones para lograr un promedio de 7.2 FPS en el rendimiento del procesamiento con la resolución mencionada y con el hardware establecido. También se minimizó el error en el cálculo de distancia al marcador a +/- 2 cm con lo cual el método de cálculo de distancia es adecuado para mantener una distancia prudente hacia el mismo marcador. En cuanto a la obtención del procesamiento en tiempo

real, a pesar de no contar con un sistema operativo en tiempo real, se pudo hacer uso de la técnica de multi hilos para realizar tareas en paralelo y así poder hacer el procesamiento de imágenes mientras se establece la comunicación entre el Raspberry y el Pixhawk, con lo que se logró alcanzar el objetivo de tiempo real propuesto gracias a las optimizaciones que se aplicaron al omitir tareas como la impresión de cuadros en pantalla y de la interfaz gráfica de usuario.

Finalmente, para probar el rendimiento del módulo se hicieron simulaciones de vuelo en un espacio virtual con el fin de probar los comandos de forma segura antes de llevar a la práctica la ruta de vuelo. Gracias a las simulaciones se pudo corroborar el correcto funcionamiento de la decodificación de códigos QR y el envío y recepción de comandos al controlador de vuelo. Además, se comprobó que es posible navegar en interiores gracias a los comandos que se envían a partir de la información de los códigos. No obstante, en cuanto a la implementación, se puede apreciar el porcentaje de aciertos de decodificación disminuye en el vuelo debido a que existe un problema en la estabilización del VANT, por lo que la estabilización y la vibración del vehículo suponen un problema para la decodificación de los códigos. Además, la falta de estabilización hace que el vehículo no pueda mantener la posición y así seguir un plan de vuelo de un punto hacia otro. Es por eso que surge la necesidad de aplicar un control inercial a los movimientos del VANT, ya que luego de recibir un comando y realizar algún movimiento, no es posible llegar al punto destino y mantener la posición debido a la inercia que el movimiento previo genera. El control inercial no se puede realizar puesto que no se tiene una señal de realimentación que permita obtener la posición horizontal. En cambio, en el caso de la posición vertical del VANT si es posible mantener la altura puesto que la señal del LIDAR proporciona una realimentación que permite controlar en todo momento la distancia entre el suelo y el VANT. Los vuelos reales mostraron resultados similares a los simulados, con la diferencia que el control inercial pudo lograrse en la simulación mas no en la implementación. Lo que supone el mayor reto para el desempeño del método de navegación propuesto.

Recomendaciones

Es posible mejorar el método de navegación propuesto mediante una realimentación de la velocidad o posición de la posición horizontal del VANT. Se propone adicionar un sensor de flujo óptico a la implementación para lograr la estabilización de la posición del VANT. Ante la imposibilidad de poder garantizar la recepción de las señales satelitales con el módulo GPS, se debe buscar la forma de obtener otra fuente que proporcione la velocidad y posición de manera confiable. Por tal motivo el sensor óptico de flujo es una opción adecuada para la obtención de esta señal de realimentación. Este sensor puede suplir la información de posición y velocidad que se necesita en ciertos modos de vuelo como LOITER. Este modo de vuelo permite mantener automáticamente la posición, altitud y dirección del vehículo. Entonces, mediante el uso del modo de vuelo LOITER con información del sensor óptico de flujo y el modo GUIDED_NOGPS con información de comandos decodificados de los códigos QR; será posible realizar la navegación autónoma en interiores de forma eficiente y con el menor riesgo de colisión del vehículo.

En cuanto al procesamiento digital de imágenes para la decodificación, será posible obtener una mejora en cuanto a la tasa de FPS conseguidos con el uso de un sistema operativo en tiempo real. Un sistema operativo en tiempo real no ejecuta procesos en segundo plano, lo cual permite destinar todos los recursos computacionales al programa de procesamiento de imágenes y navegación. Adicionalmente, es posible mejorar el rendimiento al usar un “banco de imágenes” al momento de la captura de cuadros. Esto quiere decir que mientras se capturan las imágenes del sensor, estas se almacenen en memoria hasta que algún hilo pueda procesarlos. Estos hilos podrán ser ejecutados desde distintos núcleos del procesador, lo que permitirá sacar provecho de toda la capacidad de procesamiento.

Además, con el fin de mantener la tasa de aciertos de decodificación durante el vuelo, se recomienda utilizar un computador embebido con mejores prestaciones de velocidad de procesamiento. Gracias a esto se podrá obtener una mayor tasa de cuadros por segundo y también trabajar a una mayor resolución con las imágenes procesadas. Esto permitirá obtener más aciertos a pesar de la vibración y movimiento del UAV.

Finalmente, se propone realizar un trabajo futuro para la aplicación de este método de vuelo en aplicaciones como: grabación de videovigilancia, control de inventario, revisiones por imagen, entrega de paquetes, entre otros.

BIBLIOGRAFÍA

- [1] J. Power, «Espacio interior: representación, ocupación, bienestar e interioridad,» *Temas de disseny*, vol. I, nº 30, pp. 10-19, 2014.
- [2] S. Goswami, «Global Positioning System,» de *Indoor Location Technologies*, New York, Springer-Verlag New York, 2013, pp. 51-63.
- [3] R. J. Landry, M. Sahmoudi y M. Andrianarison, «An Indoor Positioning System using Cognitive Radio,» 4 Julio 2016. [En línea]. Available: <http://substance-en.etsmtl.ca/indoor-positioning-cognitive-radio/>. [Último acceso: 11 11 2017].
- [4] P. Suwansrikham y P. Singkhamfu, «Indoor Vision Based Guidance System for Autonomous Drone and Control Application,» de *2017 International Conference on Digital Arts, Media and Technology (ICDAMT)*, Thailand, 2017.
- [5] D. Hulens, J. Verbeke y T. Goedemé, «Choosing the Best Embedded Processing Platform for On-Board UAV Image Processing,» de *Communications in Computer and Information Science*, vol 598, Cham, 2016.
- [6] K. Nonami, F. Kendoul, S. Suzuki, W. Wang y D. Nakazawa, «Autonomous Indoor Flight and Precise Automated-Landing Using Infrared and Ultrasonic Sensors,» de *Autonomous Flying Robots*, Tokio, Springer Japan, 2010, pp. 303-322.
- [7] T. Nguyen, G. K. I. Mann y R. G. Gosine, «Vision-based qualitative path-following control of quadrotor aerial vehicle,» de *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, Orlando, 2014.
- [8] S. King Phang, J. Jie Ong, B. M. Chen, R. T. C. Yeo y T. H. Lee, «Autonomous Mini-UAV for indoor flight with embedded on-board vision processing as navigation system,» de *2010 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering (SIBIRCON)*, Irkutsk, 2010.
- [9] K. Boudjit y C. Larbes, «Detection and target tracking with a quadrotor using fuzzy logic,» de *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, Algiers, 2016.
- [10] J. Kim, Y. Suk Lee y S. S. Han, «Autonomous flight system using marker recognition on drone,» de *2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, Mokpo, 2015.

- [11] C. E. Palazzi, «Drone Indoor Self-Localization,» de *DroNet '15 Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, Florence, 2015 .
- [12] R. P. FOUNDATION, «RASPBERRY PI,» [En línea]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Último acceso: 15 Octubre 2017].
- [13] D. Jones, «Picamera Documentation,» [En línea]. Available: <https://picamera.readthedocs.io/en/release-1.13/fov.html>. [Último acceso: 15 Octubre 2017].
- [14] R. P. FOUNDATION, «Documentation camera module,» [En línea]. Available: <https://www.raspberrypi.org/documentation/hardware/camera/>. [Último acceso: 15 Octubre 2017].
- [15] A. D. Team, «Ardupilot - Pixhawk Overview,» [En línea]. Available: <http://ardupilot.org/copter/docs/common-pixhawk-overview.html>. [Último acceso: 15 Octubre 2017].
- [16] LightWare Optoelectronics, «LightWare SF30/B,» [En línea]. Available: <http://lightware.co.za/shop2017/sense-and-avoid/33-sf30b-50-m.html>. [Último acceso: 15 Octubre 2017].
- [17] ArduPilot Dev Team, «Copter Documentation - Flight Modes,» [En línea]. Available: <http://ardupilot.org/copter/docs/flight-modes.html>. [Último acceso: 15 Octubre 2017].
- [18] ArduPilot Dev Team, «SITL Simulator,» [En línea]. Available: <http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html>. [Último acceso: 17 Octubre 2017].
- [19] K. Nonami, F. Kendoul, S. Suzuki, W. Wang y D. Nakazawa, «Introduction,» de *Autonomous Flying Robots*, Tokio, Springer Japan, 2010, pp. 1-29.
- [20] . C. Qichao, . D. Yaowei, L. Risan y T. Yumin, «Fast QR Code Image Process and Detection,» de *IOT Workshop 2012*, Berlin, 2012.

ANEXOS

Tabla A1

FPS en captura y muestra de cuadros		
320x240	640x480	1296x736
21.4	13.8	8.3
32.6	13.3	7.7
31.4	16.1	5.4
26.8	16.3	5.8
32.6	11.1	6.3
32.0	14.8	6.2
32.0	11.9	6.4
31.1	11.1	6.3
32.9	11.6	6.2
31.9	12.0	5.8
31.9	11.2	6.3
32.0	11.6	6.1
32.9	10.7	6.2
31.3	11.1	6.2
32.6	13.4	6.4
31.4	12.4	6.3
32.5	13.7	6.0
31.0	12.8	6.0
31.9	11.9	6.1
32.5	10.6	6.5
32.1	12.1	5.7
32.0	14.9	6.4
31.6	12.5	6.3
31.9	13.9	6.0
32.5	11.8	6.2

Tabla A2

FPS en captura de cuadros		
320x240	640x480	1296x736
36.1	13.8	10.3
32.0	14.5	7.8
32.8	12.9	8.2
31.5	26.8	8.6
32.0	11.5	8.1
32.6	12.3	7.3
32.0	12.2	9.0
31.2	27.0	9.0

32.8	19.4	7.7
31.5	10.8	8.3
31.7	11.5	8.8
32.0	13.3	8.7
32.3	12.3	7.6
31.9	11.5	7.8
32.6	14.2	7.8
31.5	12.4	8.4
31.7	12.2	9.0
32.9	10.8	9.4
32.0	13.1	8.0
32.4	11.6	8.4
30.6	13.2	9.0
32.5	13.3	8.5
32.0	12.3	9.0
32.5	10.8	7.6
31.3	13.2	8.9

Tabla A3

FPS con el algoritmo y muestra de cuadros		
320x240	640x480	1296x736
14.0	5.4	2.5
13.3	3.6	1.8
13.0	4.2	1.7
14.3	3.6	1.4
14.5	2.8	1.5
14.5	4.2	1.6
14.5	3.8	1.7
14.1	3.2	1.7
14.6	4.8	1.7
14.5	4.7	1.4
15.8	4.7	1.7
12.8	4.1	1.4
14.6	4.3	1.7
13.5	3.4	1.6
16.4	2.6	1.7
14.6	2.7	1.4
12.5	4.3	1.7
11.9	4.5	1.7
11.5	4.0	2.0
11.4	4.5	2.2
11.4	4.8	1.7

10.6	4.3	2.0
10.9	3.6	1.9
12.8	3.3	2.1
11.9	4.1	1.7

Tabla A4

FPS solo con el algoritmo		
320x240	640x480	1296x736
16.1	6.4	2.8
15.9	6.5	2.6
12.7	5.6	2.5
12.9	7.1	2.6
14.7	6.4	2.5
14.4	5.8	2.4
12.5	6.7	2.6
14.5	7.1	2.6
12.7	3.9	2.7
12.8	4.7	2.5
11.9	6.4	2.5
13.7	5.6	2.7
11.4	6.3	2.7
14.7	4.2	2.7
13.3	4.6	2.6
12.9	5.1	2.3
15.8	4.1	2.6
14.7	6.1	2.7
14.6	7.0	2.6
14.7	6.2	2.7
13.3	6.0	2.6
14.9	6.2	2.4
14.7	5.2	2.6
14.6	4.4	2.5
14.4	5.6	2.6

Tabla A5

FPS con algoritmo sin entrono grafico		
320x240	640x480	1296x736
21.5	7.8	3.3
19.8	6.7	3.0
20.2	6.7	3.1
19.7	6.5	3.1
20.0	6.7	3.1
20.0	6.5	3.0

20.0	8.7	3.0
20.0	6.5	3.1
20.3	6.7	3.0
19.8	8.7	3.0
20.4	9.5	3.1
19.7	7.7	3.2
20.0	7.1	2.9
20.0	6.5	3.0
20.2	7.1	3.0
19.9	7.7	3.1
19.9	7.7	3.1
19.8	7.4	3.1
20.6	7.1	3.1
19.7	6.7	3.0
20.2	6.9	3.1
19.8	6.6	3.1
20.0	6.4	3.2
20.0	6.2	3.1
20.0	6.8	3.0

Tabla A6

Aciertos para 640x480	
Distancia cm	Porcentaje de aciertos
160	0
150	58
140	75
130	96
120	98
110	99
100	98
90	100
80	100
70	100
60	100

Tabla A7

Aciertos para 320x240	
Distancia cm	Porcentaje de aciertos
120	0
110	3
100	12
90	30

80	55
70	95
60	99
50	100
40.0	100.0
30.0	100.0
20.0	100.0

