Control por visión de un Cuadricóptero Utilizando ROS

Maldonado Andrade Diego, Calderón María Teresa, Danilo Chávez
Ingeniería Eléctrica y Electrónica, Escuela Politécnica Nacional. Quito, Ecuador

***Resumen*- En el presente trabajo se implementa un sistema inteligente para el seguimiento autónomo a objetos basándose en su propiedad física de color o de forma que lo conforma, los mismos que son detectados mediante un sistema óptico localizado a bordo de un drone y trasmitidos a un punto de monitoreo y control para que puedan ser procesados y analizados mediante técnicas de visión por computador, obteniéndose como resultado final la determinación de la posición del objeto dentro de un sistema de referencia. El monitoreo tiene como punto de control un software desarrollado dentro del entorno del sistema operativo ROS (Robot Operating System), contando con controladores PID aplicado a cada grado de libertad, en el eje x (roll) y el eje z (yaw) para compensación de la caracterización matemática del drone**

Palabras clave—ROS, Tracking, Visual Servoing, Control PID, Filtro del Kalman.

I. INTRODUCCIÓN

Una de las aplicaciones más relevantes dentro de la robótica es el seguimiento de objetos mediante diferentes principios, técnicas de búsqueda y control. Esta aplicación de seguimiento está enfocada en buscar y perseguir objetos que tengan una característica relevante en su aspecto físico, ya sea por su forma geométrica o su color, minimizando en lo posible los recursos de procesamiento necesarios para la búsqueda de dichas características en comparación a la búsqueda de objetos más definidos, y enfocarse casi siempre en el proceso de persecución más que en el de búsqueda. La implementación del seguimiento autónomo se basa en la técnica de Visual Servoing o Visual Servo Control; contando con un cuadrotor dotado de una cámara y con la capacidad de transmitir video e información de su navegación así como de recibir los comandos de vuelo en tiempo real y de un computador cargado con un software específico conteniendo la lógica de procesamiento y control.

Para conseguir un seguimiento eficiente es necesario integrar todo un conjunto de acciones cruciales en cada una de sus etapas, empezando por la comunicación entre el Drone y el puesto de control, el envío y recepción de imágenes para procesamiento de datos y finalmente el retorno de las respectivas órdenes de vuelo; es por ello que sobre el ente encargado del control recae una gran responsabilidad al tener las funciones de administrar, ordenar y controlar al sistema en conjunto, motivo por el cual se lo ha desarrollado en su totalidad en la plataforma robótica ROS, al serun marco de código abierto que proporciona una serie de servicios y librerías que simplifican la creación de aplicaciones complejas en la robótica.

A. ROS (Robot Operating System)

ROS es un sistema operativo de código abierto que promueve el desarrollo y ejecución de aplicaciones de la robótica en general [2]. Provee herramientas y bibliotecas para la obtención, construcción, escritura y ejecución de código en varios equipos. Además implementa varios estilos de comunicación, incluida la comunicación sincrónica de estilo RPC a través de los servicios, transmisión asincrónica de datos a través de los temas, y el almacenamiento de datos en un Servidor de Parámetros [5]. ROS tiene dos "lados" básicos: el lado del sistema operativo ROS como se describió anteriormente y ros-pkg, una suite de paquetes contribuidos por el usuario que implementan funciones como localización simultánea y mapping, etc. [7].

Su intercomunicación es marco de aplicación distribuido de procesos (también conocido como Nodos) que permite a los ejecutables ser diseñados individualmente y tener un acoplamiento flexible en el tiempo de ejecución. Estos procesos se pueden agrupar en packages y stacks, que pueden ser fácilmente compartidos y distribuidos. Este diseño, permite decisiones independientes sobre desarrollo e implementación, pero todos pueden ser llevados juntos con herramientas de infraestructura ROS [5], [6].

B. Visual Servoing

El concepto de visual Servoing se refiere al uso de los datos de visión por computador para el control del movimiento de un robot, en este caso, el control de vuelo del AR.Drone [11].

Dentro del control visual las técnicas se pueden clasificar en tres tipos principales: IBVS (Control Visual basado en imágenes), PBVS (Posición - A base de control visual) y un enfoque híbrido. En este caso en específico se ha implementado en base al control IBVS ya que se encuentra basado en la determinación del error entre las características deseadas y actuales del frame de la cámara. Las mediciones de las imágenes en 2D son utilizadas para estimar directamente el movimiento deseado del robot [2]. Es más robusto, no sólo con respecto a la cámara sino también a los errores de calibración del robot.

C. El Cuadricóptero: Parrot AR.Drone 2.0

Es un vehículo aéreo no tripulado de uso recreativo civil. Funciona propulsado por cuatro motores eléctricos en configuración cuadricóptero. Cuenta con un microprocesador y una serie de sensores entre los cuales se incluyen dos cámaras, más un conector WiFi integrado. Es posible un control directo del cuadricóptero desde un dispositivo móvil, mientras se reciben a la vez imágenes y

datos de telemetría que los sensores del AR.Drone receptan.

Fig.1. AR.Drone Parrot 2.0 [2].

Se eligió este dispositivo por su robustez tanto en la comunicación como en la navegación, siendo fácil de pilotear pues existen drivers para conectarse con el AR.Drone directamente y enviarle comandos de velocidad o posicionamiento. Dispone de un IMU (Inertial Measurement Unit): que registra los datos de velocidad y orientación, usando una combinación de acelerómetros y girómetros permitiendo un vuelo y "estado de levitación" mucho más estables [4].

II. DEx|SARROLLO DE ALGORITMOS PARA RECONOCIMIENTO DE IMAGEN

El procesamiento de imágenes se usa para encontrar las características en el cuadro o imagen que serán utilizadas para reconocer un objeto o varios puntos de interés. Esta información extraída de la imagen varía de estructuras simples, como puntos o bordes, a estructuras más complejas como objetos completos. La mayor parte de las características utilizadas como referencia son puntos de interés; puntos en una imagen que tienen una posición bien definida. Algunos de estos puntos son *esquinas* formadas por la intersección de dos *bordes*[10].

A. Reconocimiento de imágenes por color

El procesamiento de color se basa en la segmentación del color deseado mediante la *sustracción de fondo* de la imagen dejando al descubierto las zonas de interés, representadas en color blanco, el mismo que se lo logra aplicando una serie de filtros y algoritmos en cada una de las imágenes [12].

En la Fig.2a se presenta el diagrama de flujo con los pasos de procesamiento implementados para obtener la imagen deseada de tal suerte que sea posible el reconocimiento y su posterior seguimiento según sus características de color. Y a la par en la Fig.2b, de manera gráfica puede notar cual es el resultado después de pasar cada imagen a través de su respectivo proceso.

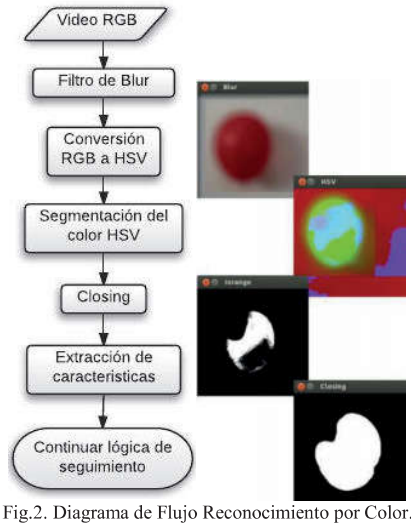

Fig.2. Diagrama de Flujo Reconocimiento por Color.

B. Reconocimiento por forma

El seguimiento por forma viene definido en encontrar figuras geométricas de determinado número de lados como son los triángulos, rectángulos y círculos, que al igual que el seguimiento por color es necesario pre-procesar la imagen aplicándose una serie de filtros y algoritmos de visión artificial entre los principales el filtro de Canny y procesamiento Morfológico; la gran diferencia en este caso es que la búsqueda estácentrada en notar la intensidad de los pixeles que se forman en las siluetas de los objetos permitiendo marcar segmentos, y si, en el caso de que dichos segmentos se intersecan y se cierran entre sí, es posible determinar un polígono. A continuación en la Fig.3a se muestra el diagrama de flujo implementado y en la Fig.3b el resultado del procesamiento de las imágenes a través de los diferentes algoritmos.

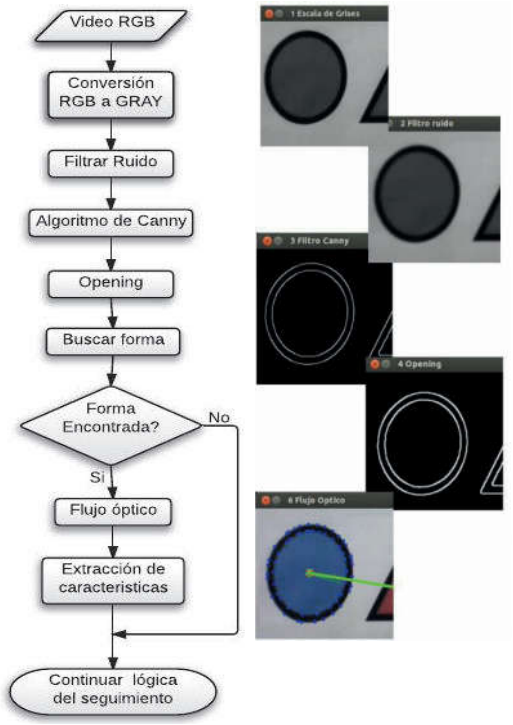

Fig.3. Diagrama de Flujo Reconocimiento por Forma.

Con el objetivo de optimizar el seguimiento y procurar que a lo largo de este no se vaya a perder el objetivo por confusiones con el entorno, problemas con el movimiento, cuestiones de iluminación, etc., se aplica otro concepto, el de *Flujo Óptico,* entendido como el movimiento aparente de los píxeles de una imagen de un cuadro a otro dentro de una secuencia de video. Estos puntos muestran la presencia de una textura o borde; garantizan la presencia de objetos sin la necesidad de detectarlos nuevamente, es decir reducen significativamente el tiempo empleado para el procesamiento de la búsqueda y se concentra en el seguimiento. No obstante, se puede dar la circunstancia donde el número rasgos antes identificados del objeto se vayan perdiendo, ya sea por el propio movimiento del tanto del Drone como del objeto a perseguir, cambios de intensidad de la iluminación y perdida de pixeles; para todo ello cuando aparezca dicha opción es necesario ejecutar nuevamente la instrucción de búsqueda para continuar con el seguimiento.

III. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

En el seguimiento en todo momento se trata de ubicar al punto centro del objeto perseguido, en el centro del campo visual de la cámara del cuadricóptero, para esto se estima primero la posición del objetivo y posteriormente se generan y se envían las respectivas órdenes de vuelo hacia el cuadricóptero. La Fig.4 nos muestra el diagrama de bloques con la arquitectura de control en lazo cerrado implementado con sus deferentes componentes.

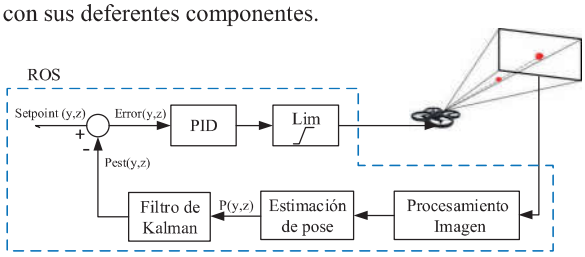

Fig.4. Diagrama de bloques del Proceso de Reconocimiento y Seguimiento.

A. Estimación de Pose

En la Fig.5 se muestra el área del campo visual de la cámara frontal del cuadricóptero, en la cual el objeto a ser seguido debe visualizarse e identificarse claramente para poder procesar las respectivas órdenes, caso contrario si el objeto buscado ya sea por forma o color no se encuentra dentro de esta área, el AR.Drone entra en modo *Hovering* o de *Planeamiento*.

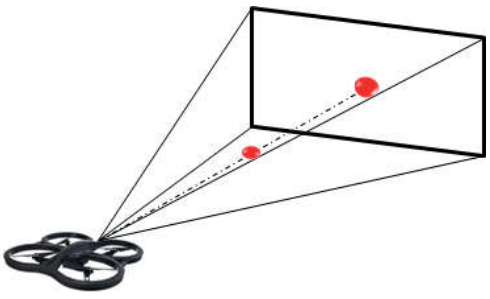

Fig.5. Campo visual de la cámara.

Para mejorar la velocidad de procesamiento de las imágenes, se las trabaja con la mitad de su resolución original de la cámara, es decir con una resolución de 640 x 360 pixeles, siendo esta, el área focal de búsqueda y seguimiento, como se muestra en la Fig.6.

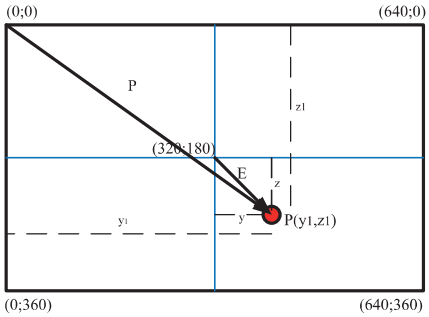

Fig.6.Estimación de Pose.

Es necesario conocer el error de pose que existe entre la ubicación actual del centro objeto y setpoint que viene definido por el centro del campo visual de la cámara.A modo de ejemplo, suponiendo que el objeto buscado se encuentra en el punto $P(y_1, z_1)$ respecto a la esquina superior izquierda, el error de pose se obtiene mediante la ecuación (1).

$$E(y,z) = P(y_1,z_1) - P(320,180) \qquad (1)$$

En el seguimiento, el punto centro del objeto no tiene una trayectoria definida sino más bien una irregular que varía por la suman de los efectos internos y externos del seguimiento, causando que el comportamiento del sistema se vuelva inestable. Para compensar y mejorar considerablemente los efectos antes mencionados y lograr una trayectoria suave y continua aplica un Filtro de Kalman.

B. Filtro de Kalman

El Filtro de Kalman se incorpora para conseguir la estabilización del cuadricóptero después de implementar el algoritmo de estimación de pose suavizando la trayectoria que se sigue, corrigiendo los errores de estimación en la misma [13].

El filtro es una ecuación de vectores que describe la evolución del estado con el tiempo. Para desarrollar el filtro, hay que diseñar un *observador,* lo que implica estimar el vector de estado a partir de las salidas del sistema. El vector de estado viene definido por la posición y su respectiva variación de velocidad en cada eje de desplazamiento, es decir,$[P_y \quad P_z \quad \Delta V_y \quad \Delta V_z]$ quedando de la siguiente manera.

$$X_K = F.X_{K-1} + W_k \qquad (2)$$

$$\begin{bmatrix} Py_k \\ Pz_k \\ \Delta Vy_k \\ \Delta Vz_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Py_{k-1} \\ Pz_{k-1} \\ \Delta Vy_{k-1} \\ \Delta Vz_{k-1} \end{bmatrix} + W_{k-1} \qquad (3)$$

Donde:

X_{K-1}, es el vector de estado (posición-velocidad)
F, es la matriz de transición de estados del sistema
W_k, es el vector que representa el ruido asociado al modelo del sistema.

A continuación, se muestra la estabilización que se consigue con el filtro de Kalman, en la fig.7 en color azul es

la trayectoria original del centro del objeto, y en color verde la trayectoria resultante con el filtro.

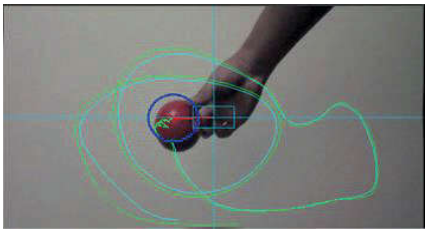

Fig.7.Filtro de Kalman en la trayectoria del objeto.

C. Modelación del cuadricóptero

El modelo escogido en base a la literatura analizada; presenta un "modelo" para el control de la velocidad del AR.Drone de forma aproximada, considerando un marco de referencia inercial y un marco de referencia fijo en el cuadricóptero, que se asume como un cuerpo rígido en el espacio, con una fuerza principal y tres momentos [14].

Fig.8. Sistema de referencia para la modelación

Donde se tiene la matriz rotacional R que representa el sistema

$$\begin{bmatrix} \cos\psi\cos\theta & \cos\psi\text{sen}\theta\text{sen}\phi - \text{sen}\psi\cos\phi & \cos\psi\text{sen}\theta\cos\phi + \text{sen}\psi\text{sen}\phi \\ \text{sen}\psi\cos\theta & \text{sen}\psi\text{sen}\theta\text{sen}\phi + \cos\psi\cos\phi & \text{sen}\psi\text{sen}\theta\cos\phi - \cos\psi\text{sen}\phi \\ -\text{sen}\theta & \cos\theta\text{sen}\phi & \cos\theta\cos\phi \end{bmatrix} \quad (4)$$

Y después de todo el análisis se presenta el modelo matemático en funciones de transferencia para cada uno de sus ejes, tomando para este caso el desplazamiento en los ejes de altura y el de ladeo.

Eje Yaw Eje Roll

$$G(S) = \frac{1}{m.S^2} \quad G(S) = \frac{l}{I_y.S^2} \qquad (5),(6)$$

Donde:

m,es la masa del cuadrotor
l, la longitud del brazo de palanca
I_y, el momento de inercia específico

El modelo tomado representa a un sistema de segundo orden siendo su comportamiento oscilatorio, para lo cual se implementa un controlador PID discreto, por su rápida respuesta correctora del error.

D. Controlador PID

El principio de un PID es reducir el error que existe entre el valor medido y el valor que se desea obtener, basándose en la calibración de tres parámetros o componentes para aplicar una acción correctora que ajuste el proceso, con base en esto se diseñaron los controladores y se obtuvieron las diferentes constantes. Con la ayuda del MATLAB, se observó el tipo de comportamiento; y a continuación en las fig.9 y fig.10, se muestra la respuesta a una entrada paso, sin controlador y con controlador para cada uno de los ejes.

Altura

$$G(S) = \frac{1}{m.S^2}C(S) = \frac{3.5\,S^2+1.085\,S+0.00105}{S} \quad (7)$$

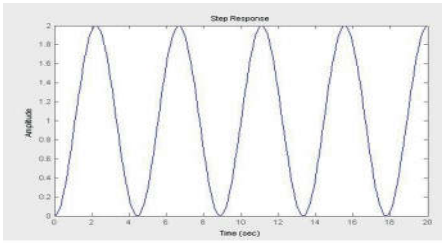

Fig.9a. Respuesta original del eje

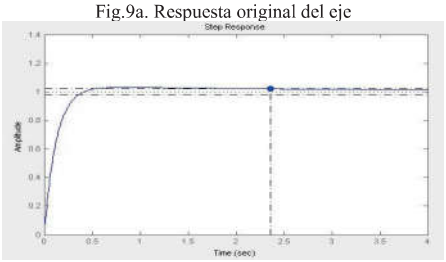

Fig.9b. Respuesta con compensación PID

Ladeo

$$G(S) = \frac{L}{I_y.S^2}C(S) = \frac{2S^2+0.62\,S+0.006}{S} \quad (8)$$

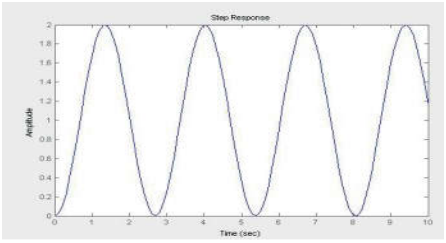

Fig.10a. Respuesta original del eje

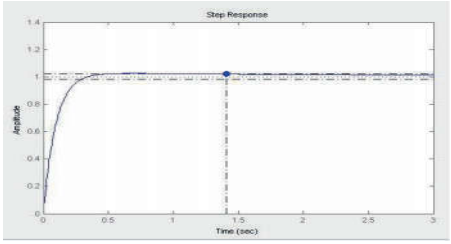

Fig.10b. Respuesta con compensación PID

E. Corriendo el Sistema en ROS

Toda la implementación de las acciones de control se las ha realizado dentro de plataforma de ROS como se lo ha indicado anteriormente, Aplicación que para ejecutarla es

necesario correr los respectivos archivos de configuración e inicialización *launch* utilizando el comando *roslaunch* en el terminal de Ubuntu, lo que permite desplegar la interfaz de usuario formada por ventanas que muestran las imágenes de cada etapa del seguimiento, los estados de vuelo del drone, y los comandos que son enviados, es decir, todos los datos de interés para que el usuario manipule de manera eficiente el seguimiento en todo tiempo. En la Fig. 11 se muestra los datos de consola que son presentados al usuario.

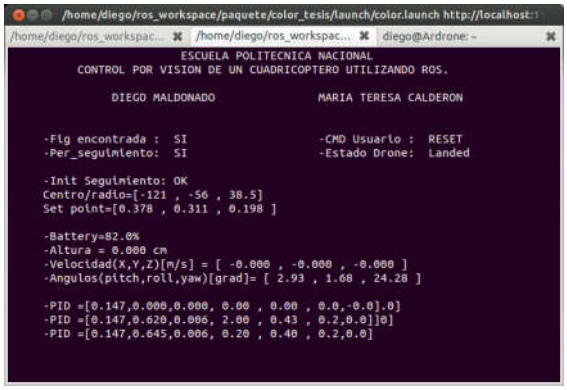

Fig.11. Captura de pantalla de la Interfaz de Usuario.

Para la mejor comprensión de la comunicación interna utilizada por ROS, en la Fig.12, se presenta el flujo con la comunicación establecida mediante sus tópicos entre los tres nodos principales del proceso:

- /Seguimiento: Conteniendo el procesamiento de imágenes, búsqueda de objetos y envió de ordenes
- /ardrone_driver: Aquí recibe los estados de navegación del drone y envía comandos
- /ardrone/front/image_proc: permite la conversión del formato de las imágenes desde CvBridge (característico de ROS) a el formato OpenCV IplImage (Utilizado por las librerías de OpenCV)

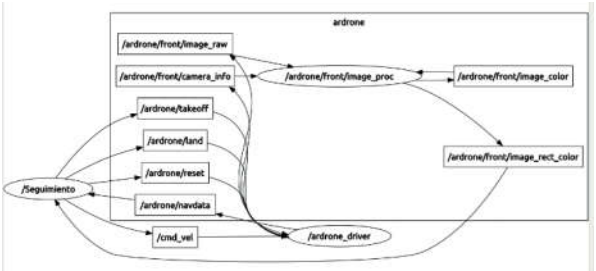

Fig.12. Captura de pantalla de la Interfaz de Usuario.

IV. PRUEBAS Y RESULTADOS

Los datos que proporcionan estas pruebas se utilizan para extraer conclusiones y proponer soluciones y/o recomendaciones para los problemas que se presentaron en el desarrollo del proyecto. Por otro lado, es importante mencionar que las pruebas se hicieron únicamente considerando dos grados de libertad del cuadricóptero bajo condiciones de un ambiente semi-estructurado.

Existen ciertos aspectos que podrían influir de forma negativa en el desarrollo de esta aplicación, situaciones como problemas de iluminación, presencia de sombras, demasiada luz en el entorno, ambientes complejos, muchos objetos en la escena, entre otros. Problemas que dificultan el reconocimiento del objetivo e impiden un buen desempeño de la parte de seguimiento, a continuación se presenta las pruebas realizadas en cada punto crítico del control.

A. Reconocimiento de objetos

Los principales problemas que aparecen dentro de la visión artificial para seguimiento y que se deben superar son, cambios de iluminación, cambios de estructura (forma o tamaño) de un objeto debido a la posición relativa del observador, presencia de muchos objetos moviéndose, etc.

Todos estos factores convergen en que resulta incierto el hecho de reconocer un solo objetivo sin experimentar confusiones con objetos de similares características. La complejidad de los algoritmos obedece a que éstos deben detectar cambios, determinar las características del movimiento del observador y los objetos, caracterizar los movimientos, recuperar la estructura de los objetos y poder decidir si en verdad los rasgos obtenidos son legítimos y suficientes para iniciar el seguimiento.

Demostrando lo antes mencionado, en la Fig.13, se puede notar la confusión de tonalidades debido a los cambios de intensidades de la iluminación en el seguimiento por color.

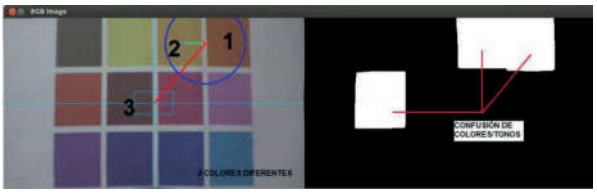

Fig.13. Confusión de tonalidad por efectos ambientales.

La siguiente imagen fue tomada en condiciones de seguimiento, es decir, en pleno movimiento del Drone y del objeto perseguido, y tal como se puede notar existe una tendencia a buscar una figura geométrica, pero esta no se encuentra bien definida ya que los segmentos que la conforman no logran cerrarse completamenteentre sí,en consecuencia, el sistema determina que no se ha encontrado la figura solicitada.

Fig.14. Errores de reconocimiento de formas geométricas por movimiento de objetos.

Dentro de un ambiente semi-estructurado, con una intensidad de luminosidad continua, limitados objetos, sin perturbaciones ambientales, y con la calibración adecuada de cada uno de los algoritmos de visión artificial el resultado obtenido en el procesamiento de imágenes es el siguiente para cada uno de los casos.

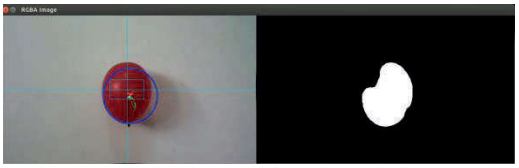

Fig.15. Resultado del Procesamiento de Seguimiento por color.

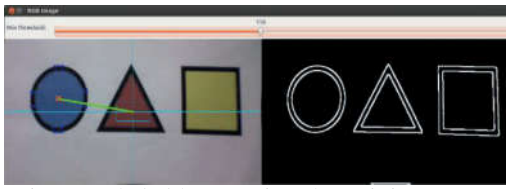

Fig.16. Resultado del Procesamiento de Seguimiento por Forma.

B. *Seguimiento o tracking*

Poniendo a prueba el filtro de Kalman, se sometió al objeto seguido a cambios bruscos de su posición perfilándose en una trayectoria irregular para así lograr verificar la eficacia con la que el filtro estima y determina una trayectoria suave donde los cambios de posición se dan de manera progresiva, generando la sensación de estabilidad, tal cual como se muestra en las trayectorias marcadas de color verde(para el eje yaw) y azul (para el eje roll) en la Fig.17.

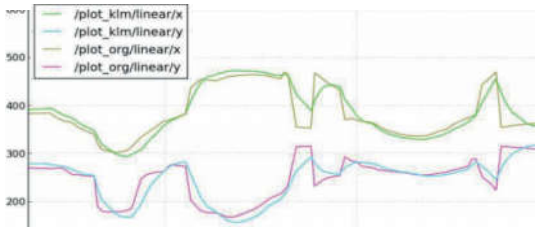

Fig.17. Eficacia de filtro de Kalman.

Finalmente se efectúala integración de cada uno de los componentes que conforman el sistema en el seguimiento, comprobándose de manera visual la eficacia de la persecución al tratar de que en todo momento el objeto deseado (ya sea por su característica de color o forma) se encuentre dentro del campo visual de la cámara. Y para un mejor análisis se corrobora mediante las gráficas variación de señal del setpoint y la señal del estado actual del seguimiento en función del tiempo para algunas condiciones.

La primera prueba fue implementada en modo Hovering en donde se ubica al objeto dentro del centro visual de la cámara del drone y se ordena que inicie el seguimiento para comprobación de la estabilización del mismo, ya que tomando en cuenta el comportamiento oscilatorio característico del modelo matemático del drone, el sistema sea compensado a través del control PID y no tienda a comportarse de manera oscilatoria.

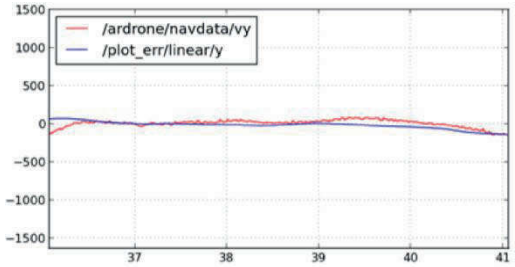

Fig.18. Seguimiento en modo Hovering.

La segunda prueba, sedesarrollópara un cambio de dirección en un eje de movimiento, obteniendo un resultado tal como se muestra en la Fig.19

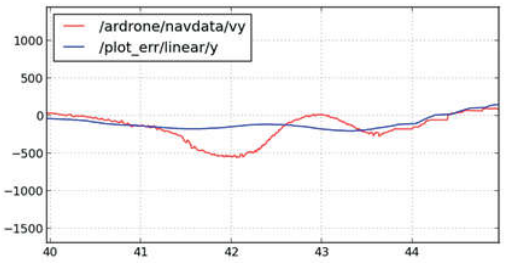

Fig.19. seguimiento con cambio de dirección del objeto seguido.

La prueba final y más completa, en sí, tiene a lugar bajo condiciones de seguimiento continuo y prolongado, expuesto a todos efectos internos y externos que se puedan suscitar incluyendo los cambios de direcciones en la persecución. La precisión del seguimiento viene demostrado en la siguiente fig.20, donde se puede ver claramente la tendencia a que el error sea cero con un cierto margen de error.

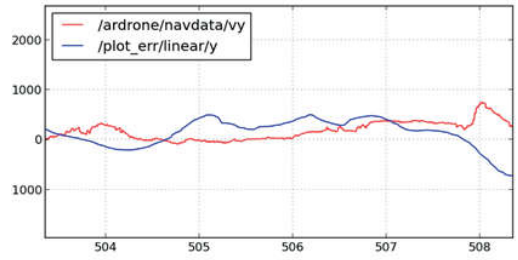

Fig.20. seguimiento en estado continuo.

V. CONCLUSIONES

- Es posible lograr la autonomía del seguimiento mediante la integración de diferentes dispositivos como sensores y actuadores que son comandados por algoritmos de control, obteniéndose un sistema robusto capaz de identificar y perseguir al objeto deseado en base a sus características físicas de color y forma.
- Mediante el uso de los recursos y herramientas del framework ROS, es posible desarrollar diferentes aplicaciones de seguimiento a objetos por medio del visual servoing dentro de ambientes semi-estructurados con una alta eficiencia en sus resultados.
- ROS se caracteriza por apoyar a la reutilización de código, es decir permite una amplia integración entre diferentes aplicaciones de código abierto para formar sistemas robóticos más complejos, pero así mismo hay situaciones negativas, como el intentar integrar repositorios no oficiales, los resultados obtenidos dependerán solo de la habilidad y experiencia que tenga el desarrollador.
- Para conseguir un seguimiento eficiente se utiliza el sistema de control visual basado en imagen para determinar de la posición del cuadricóptero respecto del objetivo que se desea seguir, sin embargo, existe la dificultad de no poder predecir la trayectoria exacta que seguirá el AR.Drone hasta el final, con lo que es necesario en lo posible estimar y predecir aplicando algoritmos de control.

VI. RECONOCIMIENTOS

Un reconocimiento y agradecimiento especial al MSc. Sandro Jua por todo su apoyo, su colaboración y guía durante el desarrollo de este proyecto.

VII. REFERENCIAS

[1] Steve Rossius, "Reconocimiento de objetos mediante WebCam en tiempo Real", Universidad Politécnica de Valencia.

[2] Jara Forcadell Ortiz, Marc Sureda Codina, "Development of autonomous manoeuvres in a quadcopter", Universidad de Girona, Febrero 2013. Berlin, Germany: Springer, 1989, vol. 61.

[3] Inkyu Sa, Hu He, Van Huynh, Peter Corke. "Monocular Vision based Autonomous Navigation for a Cost-Effective Open-Source MAVs in GPS-denied Environments. Paper.

[4] Daniele Iasella, Stefano Fossati. "Monocular Autonomous Exploration in Unknown Environment with Low-Cost Quadrotor". Politécnico Di Milano, 2012-2013.

[5] ROS, "Introduction", [Online]. Disponible http://wiki.ros.org/ROS/...

[6] Álvaro García Cazorla. "ROS: Robot Operating System", Universidad Politécnica de Cartagena, Septiembre 2013.

[7] "Sistema Operativo Robótico", [Online]. Disponible: http://es.wikipedia.org/wiki/Sistema_Operativo_Rob%C3%B3tico

[8] "OpenCV", [Online]. Disponible: http://es.wikipedia.org/wiki/OpenCV

[9] Gary Bradsky, Adrian Kaehler, "Learning OpenCV. Computer Vision with the OpenCV Library". O'REALLY.

[10] Pascual Campoy, Iván Mondragón, Miguel Olivares, Carol Martínez. "Visual Servoing for UAVs". Universidad Politécnica de Madrid.

[11] François Chaumett, Seth Hutchinson. "Visual Servo Control Part 1: Basic Approaches". IEEE, Tutorial.

[12] P. Gil, F. Torres, F. Ortiz, "Detección de Objetos por Segmentación Multinivel combinada de espacios de Color", Universidad de Alicante, 2004.

[13] Sergio Pereira Ruiz. "Localización de robots mediante filtro de Kalman".

[14] Barquín Murguía, Alberto Isaac. "Implementación de un controlador externo en un cuadricóptero comercial", Marzo 2014.

VIII. BIOGRAFÍAS

María Teresa Calderón, nació en la ciudad de Quito, el 28 de Agosto de 1990. Realizó sus estudios secundarios en el Colegio Nacional "Hipatia Cárdenas de Bustamante". Terminó la carrera de Ingeniería Electrónica y Control en la Escuela Politécnica Nacional en diciembre de 2013. Áreas de Interés: Domótica, Edificios Inteligentes, Generación de Energías Renovables, Ecología y Medio Ambiente, Automatización de Procesos.

Diego Maldonado, nació el 12 de Noviembre de 1990 en la ciudad de Tulcán. Realizó sus estudios secundarios en el Instituto Tecnológico Superior Bolívar de la cuidad de Tulcán. Graduado Cum Laude de la carrera de Ingeniería Electrónica y Control en la Escuela Politécnica Nacional en Octubre 2014.Área de Interés: Sistemas Microprocesador, Robótica, FPGA's y Control Industrial. Actualmente desempeña el cargo de Docente Ocasional 2 en el Departamento de Automatización y Control Industrial de la Escuela Politécnica Nacional.

Danilo Chávez García, nació el 1 de Abril de 1977, en Quito, Ecuador. Doctor en Ingeniería de Sistemas de Control en la Universidad Nacional de San Juan, Argentina. Máster en Domótica en la Universidad Politécnica de Madrid. Hizo una pasantía corta de Investigación, Liebniz Universitat Hannover, Alemania e Ingeniero en Electrónica y Control en la Escuela Politécnica Nacional. Áreas de Interés: Sistemas de Control, Edificios Inteligentes, Sistemas Hombre-Máquina, Desarrollo Sostenible, Energías Alternativas.