CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Programa de

Sistemas Autónomos de Navegación Aérea y Submarina

"Navegación de un cuadrirrotor utilizando visión artificial en ambientes exteriores para evadir obstáculos"

TESIS

Que presenta

DONOVAN FLORES MEZA

Para obtener el grado de

MAESTRO EN CIENCIAS

En

Sistemas Autónomos de Navegación Aérea y Submarina

Directores de la Tesis:

Dr. Hugo Romero Trejo

Dr. Pedro Castillo García

México, D.F. **Junio, 2015**

Agradecimientos

Principalmente quisiera agradecer a mis papás que me han apoyado en todas mis decisiones y mis proyectos, a toda mi demás familia de Tlaxcala que siempre ha mostrado interés y apoyo a lo largo de mi trayectoria académica, también quiero agradecer a mis compañeros y amigos del laboratorio por su apoyo y disposición cuando surgieron dudas.

Agradezco también a mis asesores de tesis gracias por todo lo que me enseñaron, gracias porque mis conocimientos en esta área eran escasos pero siempre hubo disposición para aclarar aspectos del proyecto.

Gracias al CONACYT por la beca otorgada, fue el principal sustento económico durante mi estancia en el CINVESTAV.

Gracias al CINVESTAV, por todas las facilidades en cuanto a servicios y apoyo.

Resumen

La utilización de la visión artificial es muy útil para la navegación cuando el entorno de vuelo suele ser muy incierto o irregular como puede ser el hecho de evadir ramas o troncos de arboles, es por eso que este trabajo está enfocado en la navegación utilizando diferentes métodos de visión artificial como el algoritmo de rastreo Haartrainning, el flujo óptico y la segmentación por color para el reconocimiento de obstáculos y la evasión de los mismos.
El flujo óptico es utilizado para mejorar la estabilización del cuadrirotor y para la estimación de la velocidad traslacional real, mientras que los algoritmos de Haartrainning y segmentación son empleados para la detección de los obstáculos. El campo de la visión artificial está teniendo un gran auge en la tecnología reciente ya que tiene un gran número de aplicaciones no sólo en el área científica sino que también en la parte de entretenimiento,seguridad y más.

Abstract

The application of the computer vision is very useful to the navigation when the flight enviroment is irregular or unknown like when you like avoid branches and tree stems,that is why this project focus on the navigation using several computer vision methods like Haar-trainning algorithm, optical flow and color segmentation to track and avoid obstacles.
The optical flow is using to improve the quadrotor stabilization and traslation velocity estimation, while the Haartrainning and segmentation is using to detect the obstacles. The field of computer vision reciently is growing about technology since has numerous of aplications not only in scientific area but also in the entertainment, security and more.

Índice

Glosario

- **I2C**: bus de comunicaciones en serie.
- **Visión artificial**: es un subcampo de la inteligencia artificial que tiene el propósito de programar un dispositivo para reconocer característica en una imagen.
- **Haartrainning**: algoritmo de visión artificial para el reconocimiento de objetos.
- **Cuadricóptero**: helicóptero de cuatro rotores.
- **Motor brushless**: motor eléctrico sin escobillas
- **Acelerómetro**: sensor capaz de medir los ángulos de inclinación en los tres ejes.
- **Giróscopo**: sensor capaz de medir las velocidades angulares.
- **Giroscopio**: otro nombre con el que se conoce al giróscopo.
- **Autopiloto**: es un sistema electrónico usado para guiar un vehículo sin la ayuda de un ser humano.
- **Gumstix**: sistema de programación embebido.
- **Sensor ultrasónico**: sensor de distancia que funciona a partir del rebote de ondas de sonido.
- **Multirotor**: helicóptero de más de un rotor.
- **Cuadrirotor**: otro nombre con el que se le conoce a un cuadricóptero.
- **Yaw**: ángulo de rotación en el eje z
- **Pitch**: ángulo de rotación en el eje y
- **Roll**: ángulo de rotación en el eje x

- **Firmware**: bloque de instrucciones para propósitos específicos, que establece la lógica de mas bajo nivel que controla los circuitos electrónicos.
- **Frame rate**: tasa de captura de imágenes.
- **PSeye**: cámara de PlayStation 3
- **SolidWorks**: software de diseño de piezas mecánicas.
- **Pixel**:unidad básica en que se divide una imagen digital.
- **Dilatación**: operación morfológica que adiciona pixeles en las fronteras de la imagen.
- **Erosión**: operación morfológica que remueve pixeles de las fronteras de la imagen.
- **Distancia focal**: es la distancia entre el centro óptico de la lente y el foco o punto focal.

  enditemize

Lista de figuras

Abreviaturas

- **IPN**: Instituto Politécnico Nacional.
- **UAV**: Unmanned Aerial Vehicle.
- **PCB**: Printed Circuit Board.
- **LiPo**: Lithium Polymer.
- **ARM**: Advanced RISC Machine.
- **DSP**: Digital Signal Processor.
- **SCI**: Serial Communication Interface.
- **PWM**: Pulse-Width Modulation.
- **HDMI**: High-Definition Multimedia Interface.
- **USB**: Universal Serial Bus.
- **Hz**: Hertz.
- **Seg**: Segundos.
- **UART**: Universal Asynchronous Receiver-Transmitter.
- **PLA**: Polylactic acid.
- **RGB**: Red-Green-Blue

Capítulo 1

Introducción

1.1 Objetivo General

Realizar el vuelo de un cuadrirrotor de forma autónoma con el uso de algoritmos de visión artificial, estos deben ser capaces de reconocer obstáculos para poder evadirlos.

1.2 Objetivos Particulares

- Estimar la velocidad lineal del cuadrirrotor a partir del flujo óptico.
- Realizar un algoritmo de visión artificial que detecte obstáculos.
- Calcular una ley de control dedicada a la evasión de obstáculos, considerando la distancia al objeto y la velocidad de aproximación, además de aplicar una estrategia discriminante para enfocarse en el más cercano en caso de obstáculos múltiples detectados.

1.3 Justificación

En razón de que los entornos de navegación pueden ser desconocidos e impredecibles es necesario contar con un buen sensor y estrategia de navegación para asegurar un buen desempeño de la plataforma robótica. En algunos de los casos se ha optado por el uso de GPS, sin embargo es susceptible a fallas y bloqueos por lo que una alternativa realista es la utilización de la visión artificial para asegurar una buena navegación. Tomando el enfoque de la visión artificial para reconocimiento de patrones que pudieran representar obstáculos para el helicóptero y así tener la capacidad de saber esquivarlo o bien evitar rutas por donde existan objetos que dificulten el vuelo, para ello también resulta muy útil saber la velocidad a la que se viaja, ya que dependiendo de ella se elige la ruta más segura y una acción pertinente.

1.4 Organización de actividades

Para la realización del proyecto se programaron tareas para lograr el objetivo de hacer un vuelo autónomo con visión y las fases fueron las siguientes:

Primero se realizó la tarea de revisar el estado del arte para poder delimitar los alcances y la metodología del proyecto, analizando las diferentes alternativas de solución del problema y elegir la mejor. A partir del análisis se estableció cuales serias los algoritmos a utilizar que nos permitan cumplir con el objetivo principal del proyecto. Posteriormente se eligieron los trabajos más relevantes y acordes con nuestro propósito para que sirvieran como guía y apoyo en sus respectivas fases del proyecto. Una vez elegidos los algoritmos, lo siguiente fue probar el desempeño de estos. Como primer paso fue necesario probar primero los algoritmos en la computadora de escritorio con OpenCV corriendo sobre algún compilador de Python. De ser satisfactorio el desempeño en computadora la siguiente fase sería intentar hacerlo funcionar en un dispositivo embebido para poder montarlo en el cuadrirrotor, de no ser bueno el funcionamiento en la computadora se procedería a buscar una posible alternativa que resuelva el problema. Primero se realizan pruebas con cualquier tipo de imagen para posteriormente examinar el algoritmo con imágenes de paisajes que realmente podrían presentarse en vuelos de prueba, de existir discrepancias en los resultados con respecto a las primeras pruebas se requiere ajustar el algoritmo a las condiciones reales en vuelo.

Dando seguimiento al proyecto se prosigue a la elección del dispositivo embebido que se va a ocupar; cuando se trata de procesamiento de imágenes el costo computacional suele ser elevado y hay que tomarlo en cuenta a la hora de la elección de la plataforma de desarrollo. Otro aspecto a considerar para la elección es el tamaño y el peso ya que no se cuenta con mucho espacio sobre el cuiadrirrotor y además tiene una capacidad de carga útil limitada, teniendo incluidos ya los pesos de la estructura, motores y batería. Después de haber elegido la plataforma se procede a aprender a utilizar el dispositivo

y aprender a explotar al máximo sus características. Una vez que es posible programar el dispositivo se empieza a programar las funciones básicas de funcionamiento de la tarjeta y habilitar los puertos de comunicación y otros periféricos necesarios.
Ya que se tienen todo funcionando correctamente lo siguiente es empezar a probar los algoritmos de visión de forma embebida y revisar si la plataforma es lo suficientemente capaz de realizar el tratamiento de forma satisfactoria, se prueban primero los algoritmos de forma separada y ya que funcionan correctamente, se incorporan en uno solo para comprobar su buen desempeño y factibilidad desde el punto de vista computacional.

Una vez logrado que los algoritmos de visión funcionan de forma correcta todos juntos, se empieza a calcular la ley de control seleccionada a probar y se acopla con las lecturas de los algoritmos. Por último se realizan pruebas en vuelo para observar su desempeño real y corregir errores para tener el mejor comportamiento y desempeño posibles.

1.5 Estado del arte

Las innumerables aplicaciones de vehículos aéreos impulsa la necesidad de que estos puedan trabajar con autonomía, que solamente baste con indicarle cual va a ser su tarea, que ésta la realice de forma correcta y de forma secuencial, pero que pasaría cuando el entorno no siempre es el mismo, es ahí donde entra la visión artificial para ofrecer información visual del entorno donde se encuentra y así poder decidir cual es la mejor forma de realizar su tarea. En trabajos como [16], [5], [21] y [19] utilizan el flujo óptico para ayudar a la estabilización del helicóptero ya que a veces la aeronave suele tener desplazamientos laterales de deriva cuando se encuentra en vuelo estacionario y por medio de este herramienta es posible detectar y corregir a tiempo esos desplazamientos. Sin embargo solo en [16] y [5] se realizan vuelos en trayectorias de forma autónoma, además de que en [16] se utiliza la lectura del flujo óptico para realizar una estimación de la velocidad traslacional real del helicóptero, de los trabajos antes mencionados se elige utilizar el método de calculo de flujo óptico de Lucas y Kanade.

En [7] se presenta una aplicación de un UAV capaz de hacer vuelo estacionario, donde se utiliza un DSP que es el que se encarga de los cálculos de los algoritmos de control y el video grabado por la cámara es enviado a una estación en tierra directamente donde se hace el procesamiento de imágenes y se envían los resultados de vuelta a la aeronave.

En más aplicaciones de visión en UAV´s se ha utilizado la parte de segmentación como principal algoritmo para la evasión de obstáculos de manera autónoma como en [1] donde se utilizan marcas en portales como puntos de referencia para la navegación, la segmentación se hace por valor en el espacio de color HSV.

En [2] se emplea un enfoque diferente de flujo óptico más denso y basado en segmentación por color para encontrar un objeto en un video, se realiza un preprocesamiento donde se segmenta la imagen en regiones de color homogéneo, después el algoritmo extrae un grupo de etiquetas para cada región y sobre éstas el flujo óptico es obtenido.

Otros trabajos acerca de navegación utilizando flujo óptico como en [20] se prueba

una segmentación de imagen basada en flujo óptico que suele llamarse segmentación de movimiento. A partir del flujo óptico es posible determinar la velocidad real de un vehículo aéreo requiriendo conocer la distancia de los objetos que aparecen en la imagen y hoy en día se han hecho avances al respecto como el sensor de flujo óptico descrito en [10] que integra un cámara de rápida adquisición y un sensor ultrasónico en un solo dispositivo para poder calcular la velocidad. Además Honegger a trabajado con otros campos de la visión artificial, siendo en [9] donde además de utilizar flujo óptico usa también disparidad de imágenes para la estimación de velocidad en un vehículo aéreo. Este dispositivo debido a su eficiencia ya se a implementado en varios proyectos como en [4], donde se apoyan de él para estabilizar el helicóptero y realizar despeje-aterrizaje autónomo y seguimiento de trayectorias, además este proyecto trabaja sobre la plataforma de desarrollo gumstix.

Existen otras formas para la localización de un objeto de nuestro interés en una imagen, se pueden utilizar algoritmos de rastreo por medio de entrenamiento, este tipo de algoritmo se le proporciona como parámetro de entrada una imagen del objeto a detectar, ejemplo de éste es el que se propone en [13] como un algoritmo de detección basado en cascadas de clasificadores de característica simple para la detección de rostros. Otro proyecto similar es el de [17] donde propone un método basado en puntos característicos para hacer el rastreo más rápidamente.

Se han explorado diferentes tipos de algoritmos de rastreo basados en diferentes técnicas, en [6] se introduce un algoritmo llamado bosques Hough que está basado como su nombre lo dice en la transformada de Hough pero con un mejor desempeño para reconocimiento de objetos. En [15] se propone otra estrategia de rastreo de objetos, está enfocado especialmente para vehículos aéreos ya que el algoritmo es robusto ante cambios de apariencia, movimientos en los 3 planos y situaciones donde el objeto rastreado está fuera del campo visible de la cámara. Ya existen diferentes plataformas de software dedicado a la visión artificial y que además son de libre uso como OpenCV y la información para su uso es extensa como en [3], donde se explica los algoritmos de las

funciones más ocupadas de la librería y explica como utilizarlas con el código incluido. Otro ejemplo que intenta introducir el uso de la librería es [11] y en [8] se ven más fondo el origen de los algoritmos. En [12] se explora una alternativa de segmentación de imagen primero por que ésta se hace basado en la luz visible fusionado con luz infrarroja.

Capítulo 2

Plataforma experimental

2.1 Control

2.1.1 Modelado del cuadricóptero

Para un helicóptero de cuatro rotores se emplean las ecuaciones de Euler-Lagrange como en [14].

Las coordenadas generalizadas del vehículo pueden escribirse como:

$$q = (\xi, \eta)^T = (x, y, z, \psi, \theta, \phi)^T \in \mathbb{R}^6, \tag{2.1}$$

donde:

$\xi = (x, y, z) \in \mathbb{R}^3$: denota la posición del centro de masa del helicóptero con respecto al marco inercial.
$\eta = (\psi, \theta, \phi) \in \mathbb{R}^3$: son los ángulos de Euler.

ψ es el ángulo de yaw, θ el ángulo de pitch y ϕ el ángulo de roll que representan

la orientación del vehículo.

Se define el Lagrangiano como:

$$L(q, \dot{q}) = T_{trans} + T_{rot} - U, \tag{2.2}$$

donde:

$T_{trans} = \frac{m}{2}\dot{\xi}^T\dot{\xi}$: es la energía cinética translacional.
$T_{rot} = \frac{1}{2}\dot{\omega}^T\mathbf{I}\dot{\omega}$: es la energía cinética rotacional.
$U = mgz$: es la energía potencial del sistema.
z : representa la altura de vehículo.
m : es la masa.
ω : es la velocidad angular.
$\mathbf{I}$: es la matriz de inercia.
g : es la aceleración gravitacional.

El vector ω respecto a los ejes de coordenadas del cuerpo se relaciona con las velocidades generalizadas, en donde los ángulos de Euler son válidos, utiliza la siguiente relación cinemática.

$$\dot{\eta} = W_\eta^{-1}\omega \tag{2.3}$$

donde:

$$W_\eta^{-1} = \begin{bmatrix} -sin\theta & 0 & 1 \\ cos\theta sin\psi & cos\psi & 0 \\ cos\theta sin\psi & -sin\psi & 0 \end{bmatrix}. \tag{2.4}$$

$$\omega = \begin{bmatrix} \dot{\phi} - \dot{\psi} sin\theta \\ \dot{cos}\phi + \dot{\psi} cos\theta sin\phi \\ \dot{\psi} cos\theta cos\phi - \dot{\theta} sin\phi \end{bmatrix} \tag{2.5}$$

Se define $\mathbb{J}(\eta) = W_\eta^T \mathbf{I} W_\eta$ tal que

$$T_{rot} = \frac{1}{2} \dot{\eta}^T \mathbb{J} \dot{\eta}. \tag{2.6}$$

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.7}$$

La matriz $\mathbb{J}$ actúa como la matriz de inercia para la energía cinética rotacional del helicóptero, expresada en términos de coordenadas generalizadas η. El modelo dinámico completo del helicóptero se obtiene de las ecuaciones de Euler-Lagrange con fuerzas externas generalizadas.

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \begin{bmatrix} F_\xi \\ \tau \end{bmatrix} \tag{2.8}$$

donde F_ξ es la fuerza traslacional aplicada al vehículo debido a la entrada de control principal, entonces se puede expresar lo siguiente: $\hat{F} = [0\ 0\ u]$ y $u = f_1 + f_2 + f_3 + f_4$, donde

$$f_i = k_i \omega_i^2 \, \forall i = 1, ..., 4 \, k_i > 0, \tag{2.9}$$

k_i es una constante y ω_i es la velocidad angular del motor i-ésimo. Por lo tanto $F_\xi = R\hat{F}$ donde R denota la matriz rotacional $R(\psi, \theta, \phi)$ que representa la orientación del cuadrirotor relacionada al eje de referencia fijo.

Los momentos o pares generalizados son:

$$\tau = \begin{bmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} \triangleq \begin{bmatrix} \sum_{i=1}^{4} \tau M_i \\ (f_2 - f_4)l \\ (f_3 - f_1 l) \end{bmatrix}, \tag{2.10}$$

donde l es la distancia entre los motores y el centro de gravedad, y τM_i es el momento producido por el motor M_i, $i = 1, ..., 4$, alrededor del centro de gravedad del vehículo.

Debido a que el Lagrangiano no contiene términos en la energía cinética combinando $\dot{\xi}$ con $\dot{\eta}$, las ecuaciones de Euler-Lagrange pueden ser divididas en las dinámicas para las coordenadas de ξ y las coordenadas de η.

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_\xi, \tag{2.11}$$

$$\mathbb{J}\ddot{\eta} + \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T \mathbb{J}\dot{\eta}) = \tau. \tag{2.12}$$

Definiendo los términos de coriolis que contiene los efectos giróscopos y centrífugos asociados a η como:

$$C(\eta, \dot{\eta})\dot{\eta}) = \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T \mathbb{J}\dot{\eta}), \tag{2.13}$$

Finalmente se obtiene:

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_\xi. \tag{2.14}$$

$$\mathbb{J}\ddot{\eta} = -C(\eta, \dot{\eta})\dot{\eta} + \tau.$$

Con el objetivo de simplificar el modelo se propone el siguiente cambio de variable:

$$\tau = C(\eta, \dot{\eta})\dot{\eta} + \mathbb{J}\tilde{\tau}, \tag{2.15}$$

donde $\tilde{\tau} = [\tilde{\tau}_\psi,\ \tilde{\tau}_\theta,\ \tilde{\tau}_\psi]$ son las nuevas entradas y por lo tanto,

$$\ddot{\eta} = \tilde{\tau}. \tag{2.16}$$

El sistema puede reescribirse como:

$$\begin{aligned} m\ddot{\xi} + mgE_z &= F_\xi \\ \ddot{\eta} &= \tilde{\tau} \end{aligned} \tag{2.17}$$

donde F_ξ está definida como $F_\xi = \begin{bmatrix} -f\sin\theta \\ f\sin\phi\cos\theta \\ f\cos\phi\cos\theta \end{bmatrix}$

Finalmente al multiplicar se obtiene:

$$\begin{aligned} m\ddot{x} &= -u\sin\theta \\ m\ddot{y} &= u\cos\theta\sin\phi \\ m\ddot{z} &= u\cos\theta\cos\phi - mg \\ \ddot{\varphi} &= \tilde{\tau}_\phi \\ \ddot{\theta} &= \tilde{\tau}_\theta \\ \ddot{\psi} &= \tilde{\tau}_\psi. \end{aligned} \tag{2.18}$$

donde x y y son las coordenadas en el plano horizontal, z es la posición vertical, y $\tilde{\tau}_\psi$, $\tilde{\tau}_\theta$, y $\tilde{\tau}_\phi$, son los momentos de yaw, pitch y roll respectivamente, los cuales están relacionados con los momentos generalizados τ_ψ, τ_θ y τ_ϕ de la ecuación (2.15).

2.1.2 Aplicación de un controlador PID

Para implementar este controlador en la plataforma, nuevamente se utilizó el modelo reducido (2.18) como en [14], y se dividió en subsistemas, posteriormente se aplicó una cancelación de no-linealidades.

Subsistema z:

Considere el subsistema correspondiente a la posición vertical:

$$m\ddot{z} = u\cos\theta\cos\phi - mg.$$

Aplicamos el siguiente control con el objetivo de cancelar las no-linealidades:

$$u = m\left(u_1 + g\right)\left(\cos\theta\cos\phi\right)^{-1} \tag{2.19}$$

con $\cos\theta\cos\phi \neq 0$, θ, $\phi \in (-\frac{\pi}{2}, \frac{\pi}{2})$

y obtenemos un nuevo sistema lineal de la forma:

$$\dot{x_z} = A_z x_z + B_z u_1, \tag{2.20}$$

donde las matrices $A_z = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B_z = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}$.

Entonces podemos implementar un controlador PD y para ello fijamos la referencia en cero y con $m = 1$. Por lo que podemos proponer u_1 como

$$u_1 = -kp_z x_z - kd_z \dot{x_z}. \tag{2.21}$$

donde:

kp_z : ganancia proporcional del subsistema z

kd_z : ganancia derivativa del subsistema z

Se obtiene la determinante de $[sI-(A+BK)]$, donde los valores de $k_{p,z} = 30$, $kd_z = 120$ fueron elegidas experimentalmente para la plataforma utilizada:

$$P(s) = s^2 + 120s + 30 \tag{2.22}$$

Las raíces obtenidas son:

$$s_1 = -119.7495,$$
$$s_2 = -0.2505,$$

Las raíces se encuentran en la parte real negativa por lo que podemos concluir que el sistema se comporta en forma estable.

Subsistema ψ:

$$\ddot{\psi} = \tau_\psi, \tag{2.23}$$

Cuya representación en espacio estado es:

$$\dot{x}_\psi = A_\psi x_\psi + B_\psi \tau_\psi \tag{2.24}$$

donde

$$A\psi = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \; x_\psi = \begin{bmatrix} x_{1_\psi} \\ x_{2,\psi} \end{bmatrix} = \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix}, \; B_\psi = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.25}$$

y se puede elegir un controlador Proporcional Derivativo, suponiendo que la referencia es cero,

$$\tau_\psi = -kp_\psi x_{1,\psi} - kd_\psi x_{2,\psi}, \tag{2.26}$$

donde:

$kp_\psi = 72$

$kd_\psi = 92$

Para verificar la colocación de las raíces, se obtiene la determinante de $[sI-(A+BK)]$,

$$P(s) = s^2 + 92s + 72 \tag{2.27}$$

cuyas raíces son:

$$\begin{aligned} s_1 &= -119.2106, \\ s_2 &= -0.7894, \end{aligned} \tag{2.28}$$

Las raíces están en la parte real negativa por lo que se puede decir que el subsistema es estable.

Subsistema $y - \phi$:

Considere el subsistema:

$$\begin{aligned} m\ddot{y} &= u\cos\theta\sin\phi \\ \ddot{\phi} &= \tau_\phi \end{aligned} \tag{2.29}$$

sustituyendo el control (2.19) se obtiene lo siguiente:

$$\begin{aligned} m\ddot{y} &= mg\tan\phi \\ \ddot{\phi} &= \tau_\phi \end{aligned} \tag{2.30}$$

con $m = 1$. Suponiendo que τ_ϕ puede estabilizar al sistema se puede asumir que $\tan\phi = \phi$ y entonces obtenemos una nueva aproximación lineal del sistema.

$$\dot{x}_{y,\phi} = A_{y,\phi}x_{y,\phi} + B_{y,\phi}\tau_\phi \tag{2.31}$$

donde $A_{y,\phi} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $x_{y,\phi} = \begin{bmatrix} y \\ \dot{y} \\ \phi \\ \dot{\phi} \end{bmatrix}$, $B_{y,\phi} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

La referencia está fijada en cero y se considera $g = 9.8$, se puede definir el control τ_ϕ como:

$$\tau_\phi = -kp_1 y - kd_1 \dot{y} - kp_2 \phi - kd_2 \dot{\phi} \tag{2.32}$$

donde:

$kp_1 = 0.1$
$kd_1 = 0.98$
$kp_2 = 30$
$kd_2 = 60$

son las ganancias para el controlador diseñado, éstas fueron elegidas experimentalmente.

Utilizando las ganancias anteriormente planteadas podemos hallar el siguiente polinomio característico:

$$P(s) = s^4 + 60s^3 + 30s^2 + 9.604s + 0.98, \tag{2.33}$$

y las raíces obtenidas son:

$$\begin{aligned} s_1 &= -59.4985 + 0.0000i \\ s_2 &= -0.1744 + 0.2783i, \\ s_3 &= -0.1744 - 0.2783i, \\ s_4 &= -0.1527 + 0.0000i, \end{aligned} \tag{2.34}$$

De acuerdo al lugar de las raíces se puede concluir que el sistema se comportará de manera estable, note que una raíz es cero debido al criterio de rango explicado anteriormente, sin embargo puede no ser cero, esto puede verse claramente utilizando un método óptimo como la función *lqr* de matlab, donde ese polo es muy pequeño pero sigue siendo negativo.

Subsistema $x - \theta$:

Considere la siguiente ecuación:

$$\begin{aligned} m\ddot{x} &= u \sin\theta \\ \ddot{\theta} &= \tau_\theta \end{aligned} \tag{2.35}$$

Se sustituye el control (2.19) y de manera similar al subsistema anterior se obtiene una aproximación lineal:

$$\dot{x}_{x,\theta} = A_{x,\theta} x_{x,\theta} + B_{x,\theta} \tau_\theta \tag{2.36}$$

donde $A_{x,\theta} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $x_{x,\theta} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$, $B_{x,\theta} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Se puede definir el control τ_θ cuando la referencia se fija en 0, con $g = 9.8$ de manera siguiente:

$$\tau_\theta = -kp_1 x - kd_1 \dot{x} - kp_2 \theta - kd_2 \dot{\theta} \tag{2.37}$$

donde:

$$kp_1 = 0.1$$
$$kd_1 = 0.99$$
$$kp_2 = 30$$
$$kd_2 = 60$$

se eligieron experimentalmente.

Con las ganancias presentadas anteriormente se puede obtener el polinomio característico siguiente:

$$P(s) = s^4 + 60s^3 + 30s^2 + 9.7s + 0.98, \tag{2.38}$$

donde las raíces obtenidas son:

$$\begin{aligned} s_1 &= -59.4985 + 0.0000i \\ s_2 &= -0.1760 + 0.2414i \\ s_3 &= -0.1760 - 0.2414i \\ s_4 &= -0.1495 + 0.0000i \end{aligned} \tag{2.39}$$

2.2 Descripción de la plataforma

La plataforma empleada para este trabajo consiste en una estructura de la marca Diatone modelo Q450 V3, impulsada con cuatro motores brushless turnigy multistar, cuatro controladores de velocidad para motor turnigy de 40 amperes, el acelerómetro y el giroscopio de 3 ejes cada uno para obtener la orientación y velocidad angular vienen dentro de la tarjeta del autopiloto Pixhawk de la marca 3DRobotics. Para el tratamiento y procesamiento de imágenes se ocupa un sistema de procesamiento embebido Gumstix Overo Fire el cual se comunica con una estación en tierra vía Wi-Fi, para la adquisición de imágenes se emplearon en primera instancia dos cámaras PSeye de la marca PlayStation y además contaban con un sensor ultrasonico SRF10 para la medición de la altitud de la aeronave en vuelo. Para protección de todos los dispositivos se diseño una protección hecha de fibra de carbono y plástico PLA que es ligera,

tomando en cuenta que el margen de peso a agregar a la plataforma es limitado.

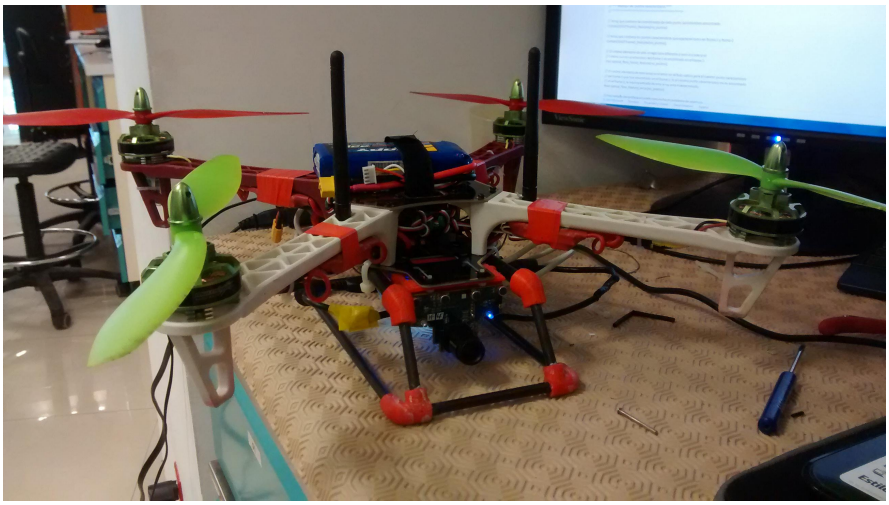


Figura 2.1: Prototipo

2.2.1 Estructura Diatone

La estructura principal es de la marca Diatone de modelo Q450 V3 y consta de cuatro brazos de nylon poliamida unidos a dos placas de fibra de vidrio con una PCB sobre ella como se puede ver en 2.2. Tiene una dimensión de 45 cm de largo, 45 cm de ancho y una altura de 6 cm de altura sin contar los motores.

Figura 2.2: Estructura Diatone Q450 V3

2.2.2 Motor

Los motores utilizados son de la marca Turnigy de la linea Multistar y tiene una corriente de trabajo nominal de 17 amperes con una máxima de 22 amp. Tiene una constante de velocidad del motor es de 880 RPM/V y cada uno tiene un peso de 65g. Para este tipo de motor se ocuparon hélices de 10 pulgadas de largo. Se recomienda el uso de baterías LiPo de 3 celdas.

Figura 2.3: Turnigy Multistar 4220-880

2.2.3 Controlador de velocidad

Para la variación de velocidad de los motores se utilizó el controlador Turnigy PLUSH 40A para motores con demanda de corriente con picos de hasta 40 amperes. Para comodidad este modelo cuenta con una bocina que facilita la sincronización de los motores y el radio. Se requiere sincronizar los cuatro controladores para configuraciones de multirotores. El proceso de sincronización y calibración de los controladores es necesario debido a que los motores deben empezar a girar todos al mismo tiempo para un despegue exitoso; además la calibración establece el valor limite de la señal del radiocontrol.

Figura 2.4: ESC Turnigy PLUSH 40A

2.2.4 Autopiloto Pixhawk

El controlador Pixhawk es de la marca 3DRobotics y puede ser configurado para manejar diferentes configuraciones de vehículos desde aéreos hasta terrestres como cuadrirrotores, hexarrotores, octarrotores, trirrotores, aviones, y estos a su vez en sus diferentes variantes. Internamente cuenta con un giroscopio y un acelerómetro de tres ejes (roll,pitch,yaw), su procesador es un ARM Cortex M4 de 32 bits, además cuenta con barómetro y diferentes puertos para poder conectar otros diferente periféricos via UART, I2C o CAN. Tienes además LEDs una bocina que sirve para avisar de algún desperfecto o señal de alarma o aviso, para seguridad se suma un botón que sirve para pre-habilitar los motores. Si se quiere guardar datos se cuenta con una memoria microSD de 4GB ya incluida con el autopiloto.

Figura 2.5: Pixhawk

2.2.5 Gumstix

Esta tarjeta esta basada en un sistema de Texas Instrument con un chip OMAP3530 que tiene un procesador ARM CortexA8 corriendo a 720 MHz, además de esto el Gumstix cuenta con un procesador digital de señales(DSP) C64x+ que trabaja a 520 MHz. Este dispositivo en conjunto con la tarjeta de expansión Summit cuenta con diversos puertos de entrada y salida como puertos I2C, lector de tarjeta microSD, HDMI, puertos SCI, puertos ADC, PWM, conexión via USB. Es posible tener comunicación con la tarjeta de forma inalámbrica ya que este modelo cuenta con módulos bluetooth y Wi-Fi. Este dispositivo es de sofware libre haciéndolo un dispositivo muy versátil.

Figura 2.6: Gumstix Overo Fire

2.2.6 Cámara PSEye

Es una cámara digital USB similar a una webcam principalmente usada en conjunto con el PlayStation 3, tiene un frame rate estándar de 60 Hz a una resolución de 640x480, a 120 Hz a una resolución de 320x240, pero puede llegar a un máximo de 187 Hz a 320x240 o 75 Hz a 640x480. La PSeye además cuenta con un arreglo de cuatro micrófonos y el lente de la cámara puede hacer un acercamiento de forma manual. La velocidad de adquisición es adecuada para el proyecto además de que es de tamaño adecuado y peso ligero.

Figura 2.7: PSEye

2.2.7 Sensor ultrasónico SRF10

Este sensor funciona a partir de un sonar, calculando el tiempo que tarda en retornar una onda que rebota con alguna superficie, en particular este modelo presenta algunas ventajas con respecto a otros ya que su tamaño es más compacto que los demás, tiene un mayor rango de sensado ya que puede llegar a detectar objetos hasta a 10 metros de distancia, además de que la salida de datos es por medio de protocolo I2C con la dirección 0xE0 y de ser necesario se puede modificar por otras 16 direcciones en caso de utilizar varios sensores o que la dirección ya esté ocupada.

Figura 2.8: SRF10

2.2.8 Sensor de flujo óptico

Este es un dispositivo que lleva integrado el sensor ADNS30380, una de sus características principales es que captura fotos con una resolución de 30 x 30 pixeles lo cual parecería una característica no muy buena, pero gracias a esto la tasa de captura se eleva considerablemente con respecto a otros tipos de cámara convencionales ya que la tasa de captura va desde los 2000 a los 6400 capturas por segundo. Además ésto sirve para que el procesamiento se haga más rápido y se actualicen los datos casi instantáneamente.

La interfaz de comunicación de este dispositivo es por protocolo SPI, se alimenta a 5V cuenta con un lente de 8mm que puede ser remplazado de ser requerido. Alguno de sus inconvenientes es que es susceptible al tipo de iluminación del entorno, tiene mejores lecturas en ambientes con buena iluminación y enfocado a superficies contrastantes o variadas.

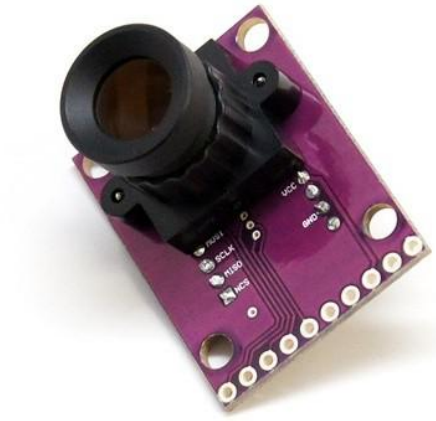


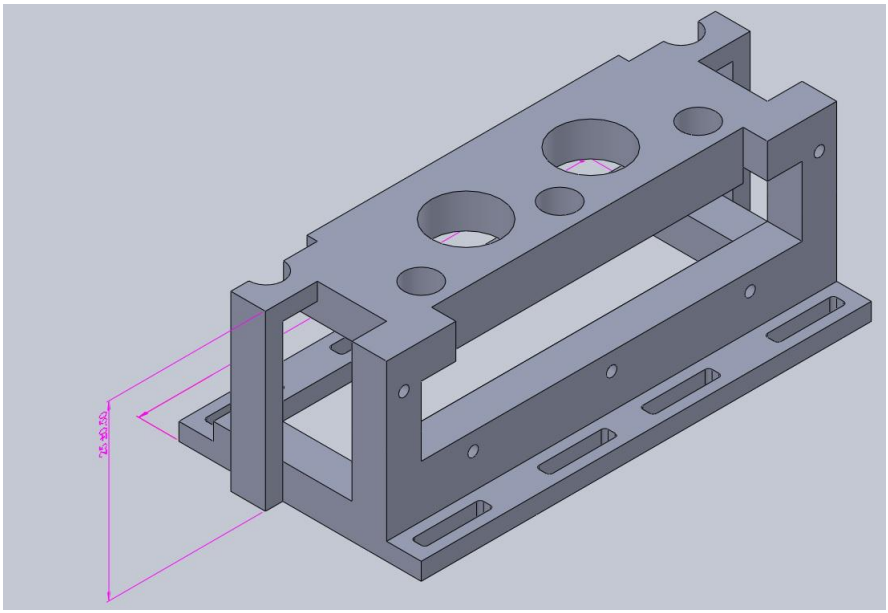
Figura 2.9: Sensor de flujo óptico

2.2.9 Estructura Camara-ultrasónico-camara

Para el montaje de las cámaras y el sensor ultrasónico se requirió diseñar una estructura compatible con la base principal del cuadrirrotor que logrará tener una cámara en el frontal y la otra en la parte inferior sin que algún objeto interfiera con el angulo de captura de las cámaras y cerca del foco de la cámara inferior lleva el sensor ultrasónico también orientado hacia abajo para medir la altura. Para la manufactura de la pieza se un utilizó material PLA y el diseño fue hecho en el software SolidWorks.

Esta estructura va ubicada en la parte inferior del helicóptero, un lugar estratégico para hacer las mediciones, ya que las cámaras no tienen un mejor angulo y posición para

una medición más precisa y útil.



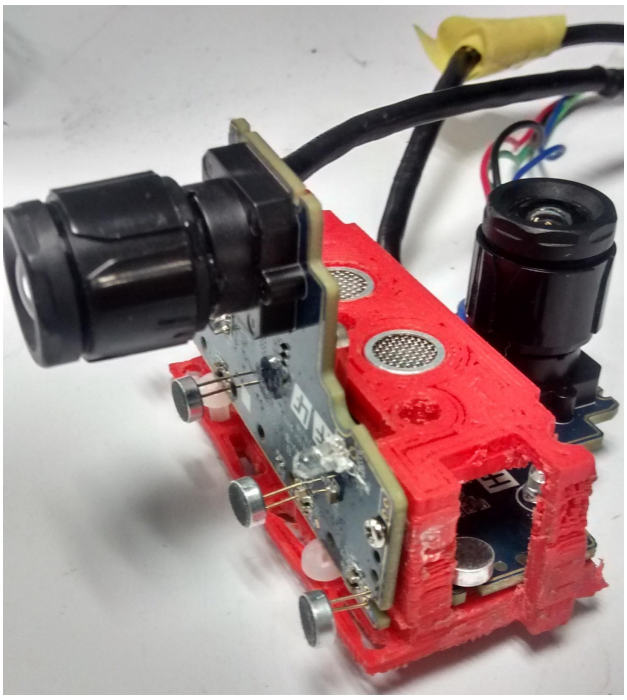


Figura 2.10: Diseño de estructura en SolidWorks

2.2.10 Estructura de protección

Debido a que al montar la estructura de las cámaras la elevación no es suficiente para que la cámara inferior quede fija en una posición correcta se procedió a diseñar una estructura que incrementara la altura, además de hacerlo más alto tiene la ventaja de que ofrece protección para la cámara frontal ya que quedaba muy expuesta y en caso de tener que hacer una aterrizaje brusco podría sufrir daños serios.Considerando que el peso total del helicóptero es un factor clave a tomar en cuenta, se pensó que la estructura fuera lo más ligera posible es por eso que se elaboró con varillas de fibra de carbono con uniones diseñadas en SolidWorks. Posteriormente al diseño de la pieza se procedió a imprimir en una impresora 3D en material PLA.

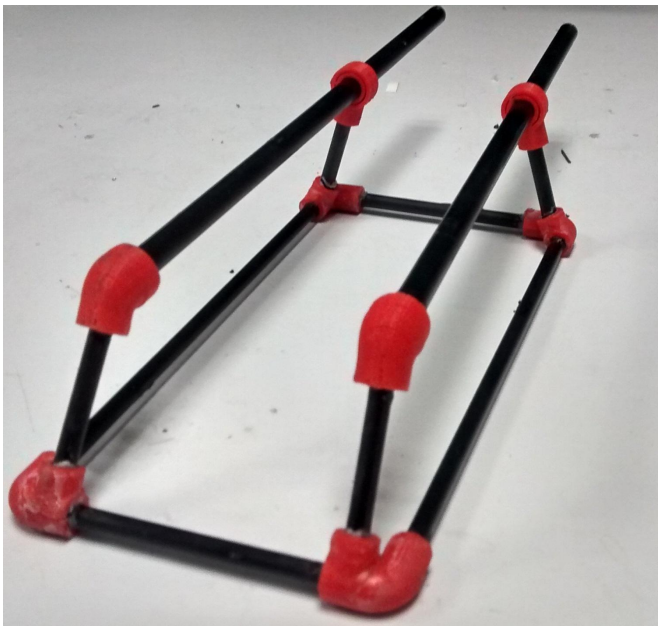


Figura 2.11: Protección de fibra de carbono

Capítulo 3

Algoritmos de visión

3.1 Haartrainning

Consiste en un algoritmo que reconoce la forma de un objeto a partir de un procedimiento que se le conoce como entrenamiento para crear un clasificador cascada. Para ello primero es necesario hacer dos definiciones importantes:

Imágenes positivas: Estas son imágenes donde se encuentra el objeto que se desea detectar sin importar el angulo o el tamaño.

Imágenes negativas: Estas son imágenes donde no se encuentra el objeto de interés, pudiendo tener cualquier cosa excepto el objeto.

Para las imágenes positivas se puede incluir en el caso de que no tenga una forma definida en concreto las posibles variantes que pudiera tener, y de ser posible desde diferentes ángulos, con diferentes fuentes de iluminación y con varios fondos, esto sirve para hacer al algoritmo más robusto y le sea más fácil detectar el objeto. Una vez teniendo las suficientes imágenes positivas se tiene que marcar exactamente en que

parte de la imagen se encuentra el objeto y esto se hace con un programa con el que se recortan las imágenes positivas utilizando el cursor, generalmente es necesario un mayor número de imágenes positivas que de imágenes negativas, al ir marcando las imágenes positivas se crea un archivo que contiene las coordenadas del objeto de interés dentro de las imágenes positivas; para el caso de las imágenes negativas se ejecuta otro programa que enlista los nombres de todas las imágenes negativas en un solo archivo.

Se prosigue creando un archivo de muestras para el entrenamiento esto se logra ejecutando un programa que genera un archivo de salida .vec y su parámetro de entrada es el archivo con las coordenadas del objeto buscado y el numero de imágenes positivas. La siguiente parte es donde se realiza la parte del entrenamiento también ejecutando el correspondiente programa con los parámetros de entrada con: el archivo de muestras .vec, el archivo con la lista de imágenes negativas y el número de imágenes positivas y negativas.
El parámetro de salida del algoritmo de entrenamiento es un archivo con extension .xml que es el clasificador cascada y también es el parámetro de entrada de el algoritmo de rastreo que si se utiliza OpenCV la función a utilizar es cvHaarDetectObjects que es el realiza la búsqueda en la imagen actual.

La cascada puede ser vista como un mecanismo que enfoca su atención en un objeto en especifico que provee garantías estadísticas de que regiones que no sean parecidas o no contengan el objeto de interés serán descartadas.

Figura 3.1: HaarTrainnig para detección de rostros

Desde el punto de vista matemático este algoritmo se apoya de diferentes técnicas para realizar el entrenamiento y el rastreo.

Como primer aspecto importante se emplea una representación especial de la imagen que es llamada imagen integral la cual permite evaluar más rápidamente las características usadas por el detector, esta representación de la imagen puede ser calculada usando algunas operaciones por pixel. Se realiza tal como dice su nombre una integral de los valores del pixel a traves de toda la imagen, es decir que el pixel con coordenadas (x, y) de la imagen integral tiene el valor de la sumatoria de todos los pixeles arriba hasta la izquierda de la imagen original en ese punto. Se expresa como:

$[ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x^{'}, y^{'})]$

donde $ii(x,y)$ es la imagen integral y $i(x,y)$ es la imagen original.
Para poder realizar el calculo de esta imagen de forma más eficaz, se puede hacer el calculo de la siguiente forma:

$$s(x,y) = s(x,y-1) + i(x,y)$$

$$ii(x-1,y) + s(x,y)$$

Donde $s(x,y)$ es la sumatoria de los valores de la columna, los casos de $s(x,-1)$ y $ii(-1,y)$ se consideran igual a 0. En pocas palabras lo que esto significa es que para realizar el calculo más rápido no es necesario estar realizando toda la sumatoria en cada paso sino que basta con sumar el valor del pixel anterior mas las sumatoria del valor de la columna mas el valor actual. Si se requiere hacer la sumatoria de algún área rectangular cualesquiera de la imagen se puede realizar un criterio similar como a continuación.

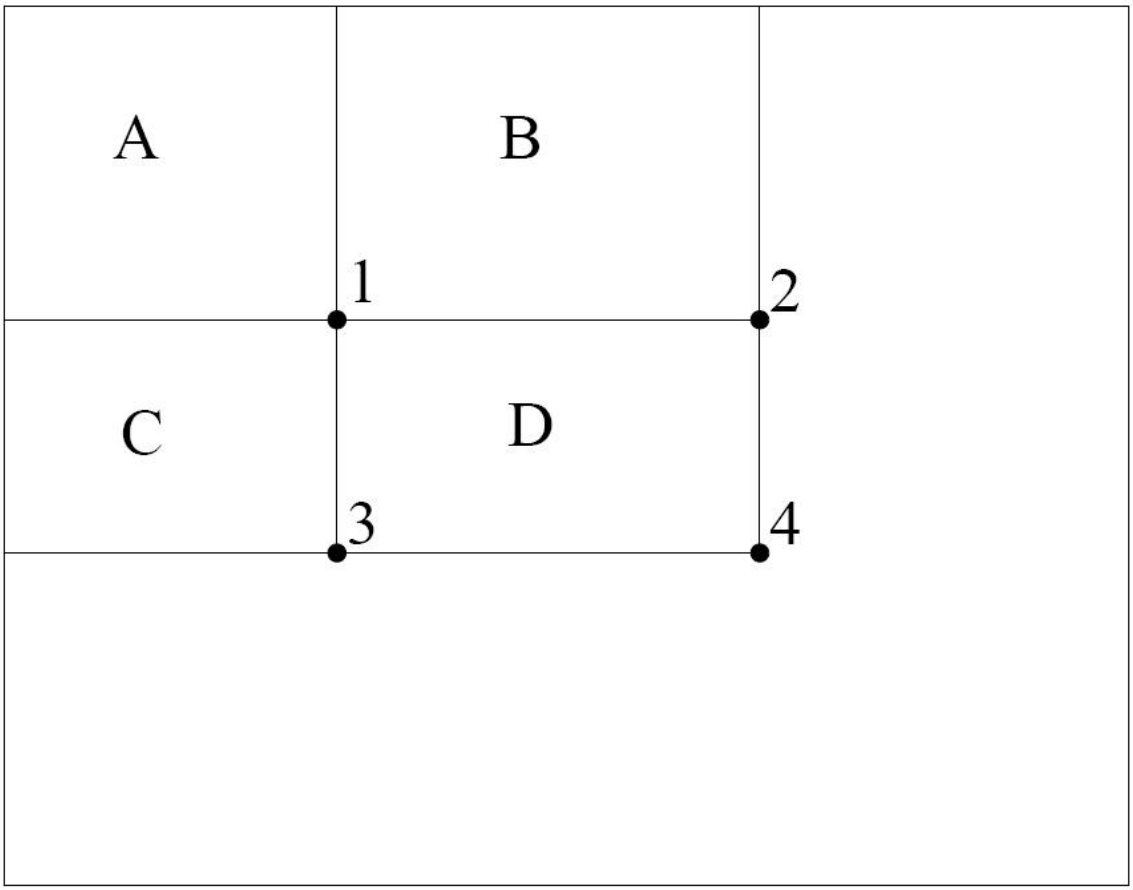


Figura 3.2: Sumatoria de un área rectangular D

De la anterior imagen se puede ver que para obtener el valor de la sumatoria de los valores del área D la manera más fácil es teniendo los valores de las áreas A, B y C, y a partir del enfoque del valor del pixel de la imagen integral podemos decir que el valor de la sumatoria de los valores contenidos en el rectángulo A es igual al valor del punto 1 de la imagen integral, por lo que el valor del punto 2 es A + B, el de el punto 3 es A + C y el de el punto 4 es A + B + C + D; tomando en cuenta esto se puede deducir que para hallar el valor del rectángulo D basta con calcular el valor de la operación de los puntos 4 + 1 - (2 + 3).

El procedimiento para la detección del objeto clasifica imágenes basándose en los val-

ores de rasgos simples. Estos rasgos simples son rectángulos fraccionados, donde cada fracción de área corresponde a una sumatoria de los valores dentro y estas áreas se suman o restan dependiendo el tipo de rasgo simple, algunos tipos se pueden ver a continuación:

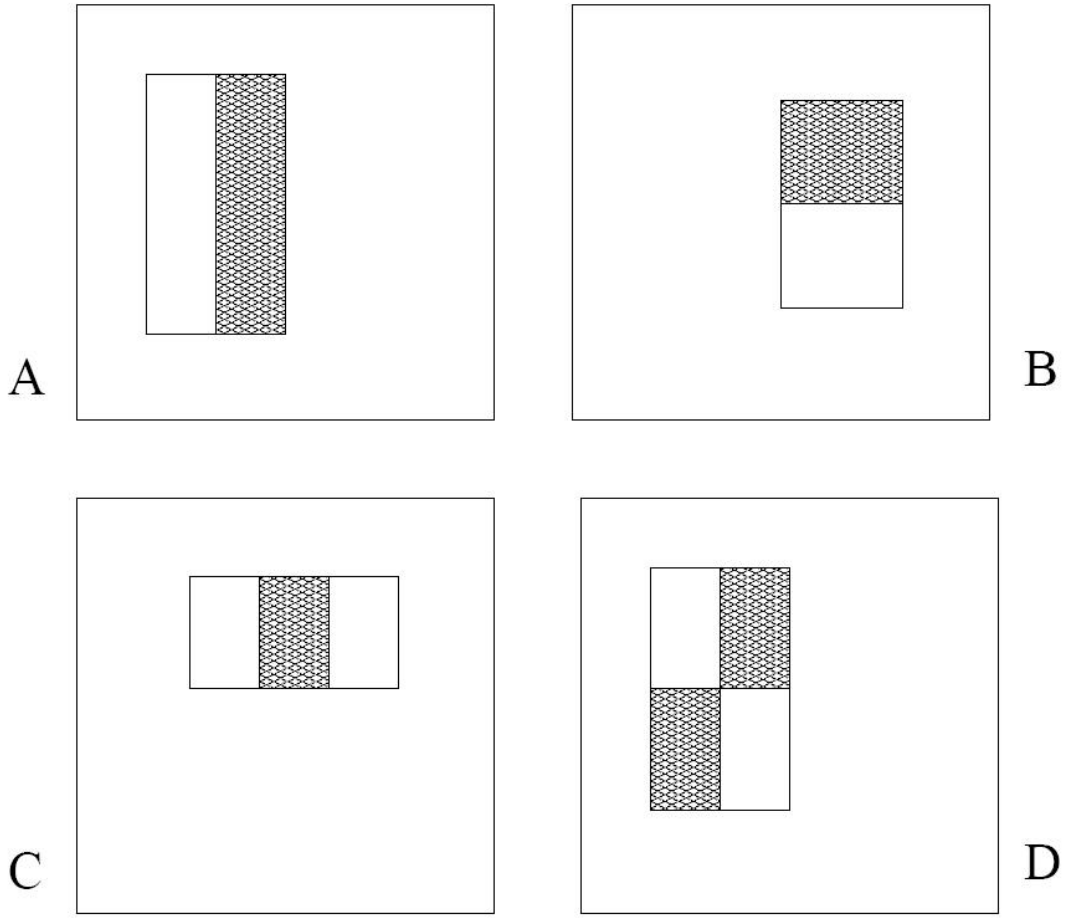


Figura 3.3: Ejemplos de rasgos rectangulares

Los sistemas basados en rasgos operan mucho más rápido que los sistemas basados en pixeles. Generalmente se usan tres tipos de rasgos simples; los primeros son los de dos rectángulos y el valor del rasgo es igual a la diferencia entre la suma del valor de los pixeles dentro delas dos regiones; los siguiente son los rasgos de tres rectángulos, éste

calcula la suma de los dos rectángulos extremos y lo resta al valor de la suma de los valores del rectángulo del centro; par cerrar están los rasgos de cuatro rectángulos, en estos su valor sale de la resta de los pares diagonales de rectángulos.
El algoritmo después junta lo ante visto de los rasgos con lo de la imagen integral y calculo rápido de cualquier rectángulo dentro de este imagen.

La siguiente fase es la del algoritmo de aprendizaje y este esta basado en un algoritmo ya existente llamado AdaBoost el cual selecciona un número de rasgos críticos de un grupo más grande y crea clasificadores eficientes. Debido a que el número de rasgos rectangulares posibles depende directamente de la resolución del detector que depende del objeto buscado el numero de posible de combinaciones es muy grande, a veces incluso más grande que el numero de pixeles de la imagen; es por esto que se hace la hipótesis de solo algunos pocos de estos rasgos pueden ser combinados para formar clasificador efectivo.
Es así como el algoritmo de aprendizaje esta diseñado para seleccionar los rasgos rectangulares simples que mejor separen ejemplos positivos y negativos. Para cada rasgo el algoritmo de aprendizaje determina función de clasificación del umbral optima de modo que el mínimo numero de ejemplos sea mal clasificados. Posteriormente de crear los clasificadores se construye lo que se conoce como una cascada de clasificadores que es parecido a lo que es una maquina de estados, esto incrementa el desempeño de detección además de que reduce el tiempo de calculo. Un resultado positivo en el primer clasificador permite pasar a ser evaluado al segundo clasificador y de pasar este sigue otro, y todos los niveles se van evaluando con un clasificador cada vez más complejo. De obtener algún resultado en cualquiera de los niveles la sub ventana es desechada.

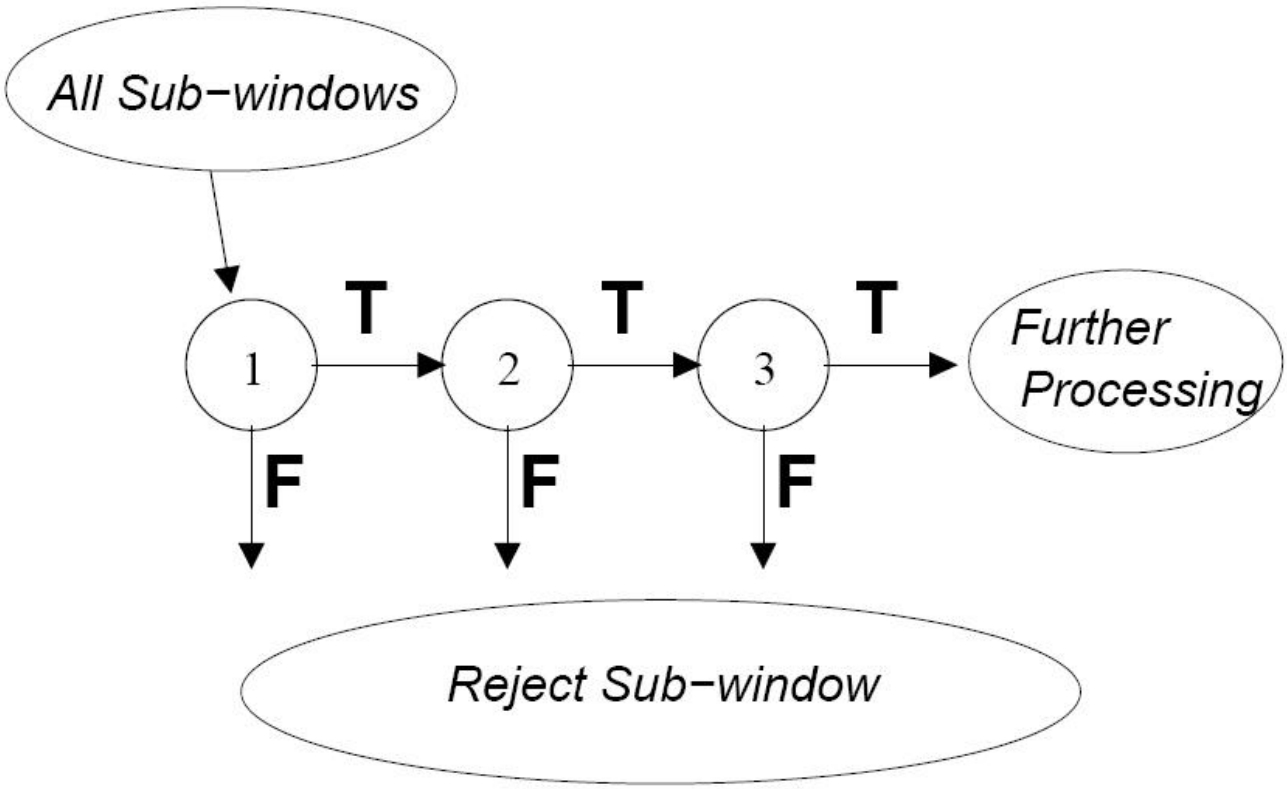


Figura 3.4: Diagrama de funcionamiento en cascada

3.2 Segmentación

Consiste en dividir o separar una imagen digital en varias partes u objetos de interés. El objetivo es simplificar la representación de una imagen en una más significativa y más fácil de analizar, se usa tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Existen diferentes métodos de segmentado, puedes ser por color, por textura, por movimiento, por bordes, etc.

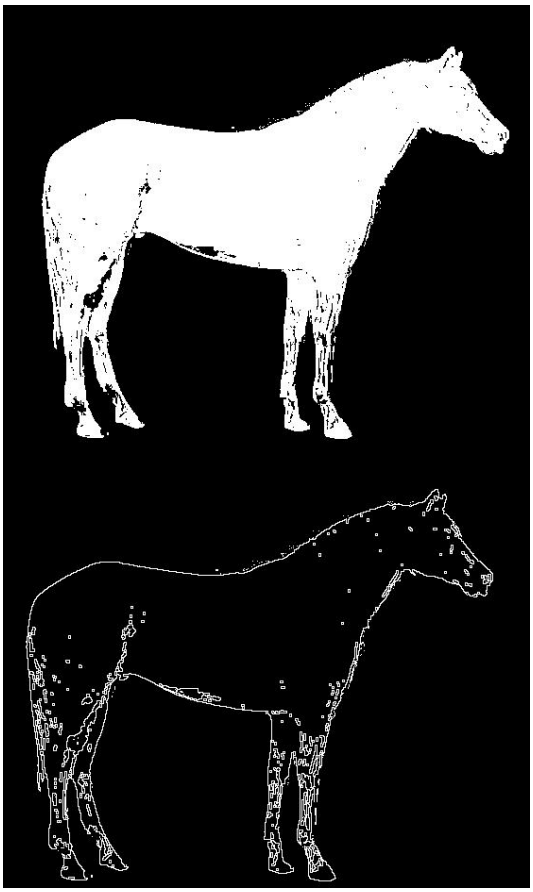


Figura 3.5: Segmentación utilizando bordes

3.2.1 Segmentación por color

Los pixeles que compartan una etiqueta parecida también tendrán ciertas características visuales similares como para este caso en especifico los valores de sus canales RGB o escala de gris, lo que indicaría que pertenece al mismo objetos de interés. Se prosigue creando una imagen binarizada de los pixeles que potencialmente son de nuestro interés, después se recomienda hacerla una operación de dilatación y una de erosión para mejorar la imagen y quitarle el ruido que pudiera tener y poder visualizar correctamente un objetivo real.Con las áreas de interés ya definidas lo que sigue ya depende mucho de para que se desee aplicar, ya que a partir de las zonas detectadas es posible determinar

que es un objeto por medio de su área ya sea por su forma o tamaño y así poder indicar que el objeto buscado se encontró y se puede marcar en la imagen con alguna figura.

Figura 3.6: Segmentación por color en plantaciones

3.3 Flujo Óptico

Es el patrón de movimiento aparente de la escena entre dos imágenes de la misma tomadas en instantes diferentes y es causado por el movimiento relativo entre el observador y la escena.

Si bien existen diferentes técnicas de calculo del flujo óptico como métodos diferenciales, correlación, métodos basados en frecuencia, etc. Para este trabajo se emplea el algoritmo de Lukas y Kanade.

Empezamos suponiendo que las imágenes a ocupar son de un solo canal por lo que se utilizan en escala de grises, se expresa como

$I(x_{img}, y_{img}, t)$

y se define como:

$$I(x_{img}, y_{img}, t) = I(x_{img} + \Delta x_{img}, y_{img} + \Delta y_{img}, t + \Delta t) \tag{3.1}$$

donde I denota el valor de intensidad del pixel, (x_{img}, y_{img}) representan la coordenada del pixel dentro de la imagen y $(\Delta x_{img}, \Delta y_{img}, \Delta t)$ son los desplazamiento con respecto al tiempo en la segunda imagen. Ahora, considerando las notaciones de que

$$I = I(x_{img}, y_{img}, t), I_x = \frac{\partial I(x_{img}, y_{img}, t)}{\partial x_{img}}, I_y = \frac{\partial I(x_{img}, y_{img}, t)}{\partial y_{img}}, I_t = \frac{\partial I(x_{img}, y_{img}, t)}{\partial t}$$

entonces podemos descomponer la parte derecha de la ecuación en series de Taylor como:

$$I(x_{img} + \Delta x_{img}, y_{img} + \Delta y_{img}, t + \Delta t) = I + I_x \Delta x + I_y \Delta y + I_t \Delta t + T.M.A.O. \tag{3.2}$$

ignorando lo términos de más alto orden(T.M.A.O), sustituyendo (3.2) en (3.1) y ya

que ∂t es con respecto al tiempo llegamos a que:

$$I_x \Delta x + I_y \Delta y + I_t = 0 \tag{3.3}$$

con Δx y Δy que son las velocidades del punto con respecto a la imagen en los ejes x y y podemos definir los componentes del vector flujo óptico (u, v), expresado vectorialmente $V_{OF} = (u, v) = [\Delta x, \Delta y]^T$. Y definimos el gradiente espacial como $\nabla I = [I_x, I_y]$. Y así llegamos a la ecuación de restricción de flujo óptico:

$$\nabla I \cdot V_{OF} + I_t = 0 \tag{3.4}$$

3.3.1 Método de Lucas y Kanade

Dado que la ecuación anterior tiene dos incógnitas en una sola ecuación es necesario proponer otra condición para poder hallar una solución. Es por eso que Lucas y Kanade proponen que el flujo óptico es constante sobre una región definida por una ventana de p x p donde $p > 1$ y centrada en el pixel del cual queremos calcular el desplazamiento. Entonces de lo anterior podemos encontrar el valor de los componentes del flujo óptico (V_{OF}) con:

$$V_{OF} = [u \quad v]^T = A^T A^{-1} A^T I_t \tag{3.5}$$

donde

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}, \; I_t = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}$$

y $n = p^2$

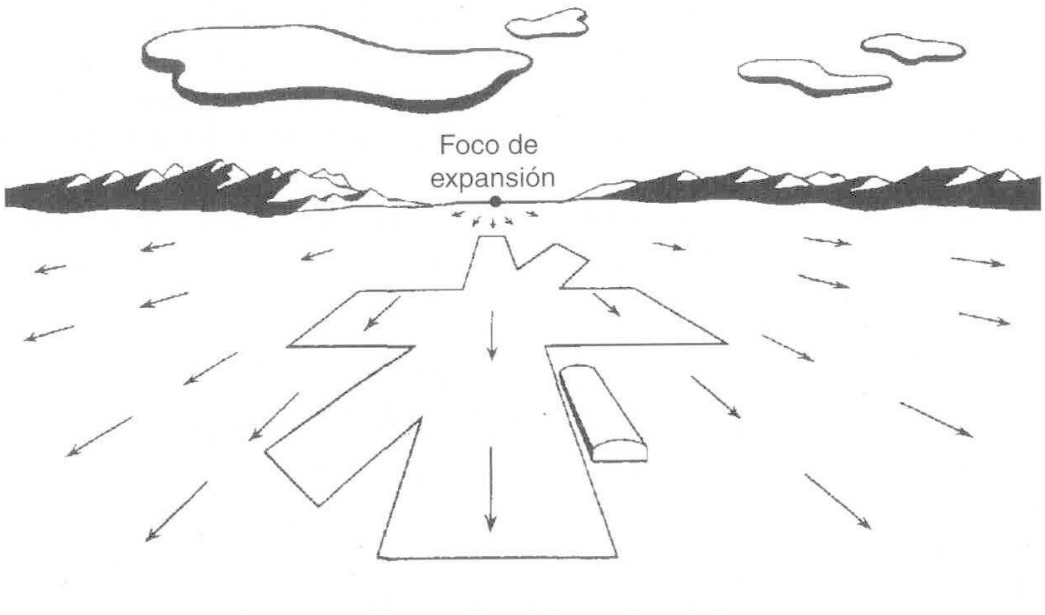


Figura 3.7: Flujo óptico

Existen otros diferentes enfoques del flujo óptico como se puede ver en [18].

3.3.2 Velocidad traslacional

Una vez calculadas las componentes del flujo óptico con 3.4 es posible hacer una estimación de la velocidad traslacional de un vehículo aéreo siempre y cuando la cámara

utilizada este fija de una forma que toda la imagen tenga el mismo movimiento, esto quiere decir que todo lo que se proyecta esta a la misma distancia, siendo así el flujo óptico calculado es directamente relacionado con la velocidad traslacional del vehículo de la siguiente forma:

$$V_{OF_x} = -\frac{\mathsf{F}\dot{x}}{z}$$
$$V_{OF_y} = -\frac{\mathsf{F}\dot{y}}{z} \tag{3.6}$$

donde $\dot{x}$ y $\dot{y}$ son las velocidades del vehículo en el plano $x - y$, z es la altitud de la cámara y F define la distancia focal.

Capítulo 4

Resultados

Por la parte de segmentación se realizaron varias pruebas de segmentación por color utilizando diferentes espacios de color, fue aplicado sobre una imagen de un paisaje de un bosque debido a que se pretende que sea útil para identificar troncos de arboles y así poder evitar colisiones, estos primeros experimentos se realizaron en una computadora de escritorio con OpenCV corriendo en Python; la imagen de prueba fue la siguiente.

Figura 4.1: Imagen de prueba

Después de realizar un suavizado a la imagen se realizaron la pruebas en los diferentes espacion de color para observar cual tiene mejor desempeño; la primera prueba se aplicó con la imagen en el espacio de color HSV para el cual debido a que el color del tronco no es uniforme se eligió un umbral de valores de pixel para cada canal de la imagen, así que primero se hizo el proceso de binarización.

Figura 4.2: Binarización para imagen HSV

La imagen binarizada muestra que los pixeles de color blanco corresponden a los valores que en ese espacio de color son equivalente o cercanos a lo de los troncos y ramas de los arboles, para el espacio HSV los valores de los umbrales fueron:

H(Matiz)=50-120
S(Saturación)=35-120
V(Valor o brillo)=0-140

Figura 4.3: Segmentación de arboles por HSV

También se realizaron pruebas para la espacios de color HLS y RGB; el proceso de binarización y segmentación es el mismo para estos dos casos, los valores de los umbrales para el espacio HLS fueron los siguientes: H(Matiz):50-120

L(Luminosidad):35-120

S(Saturación):0-100

Figura 4.4: Binarización de arboles por HLS

Figura 4.5: Segmentación de arboles por HLS

Para las pruebas en espacio RGB los valores de los umbrales son:

R(Rojo):50-120

G(Verde):35-120

B(Azul):0-100

Figura 4.6: Binarización de arboles por RGB

Figura 4.7: Segmentación de arboles por RGB

A partir de los resultados anteriores se puede observar que el que tiene mejores resultados es el HLS además de que este campo tiene la propiedad de ser uno de los que tiende a ser mas robusto ante cambios de iluminación lo que es de mucha utilidad.

El flujo óptico fue evaluado al igual que la segmentación en la computadora de escritorio, sin embargo debido a la aplicación se necesita funcionar en la plataforma embebida por lo que se cargo una version ligera de OpenCV en la plataforma gumstix overo fire y utilizando la función de calculo de flujo óptico de Lukas Kanade se realizaron pruebas

obteniendo los siguientes resultados.

Figura 4.8: Flujo óptico

Sin embargo se puede observar algunos puntos mal unidos por lo que se realizaron más pruebas reajustando los parámetros y probando cambios de iluminación.

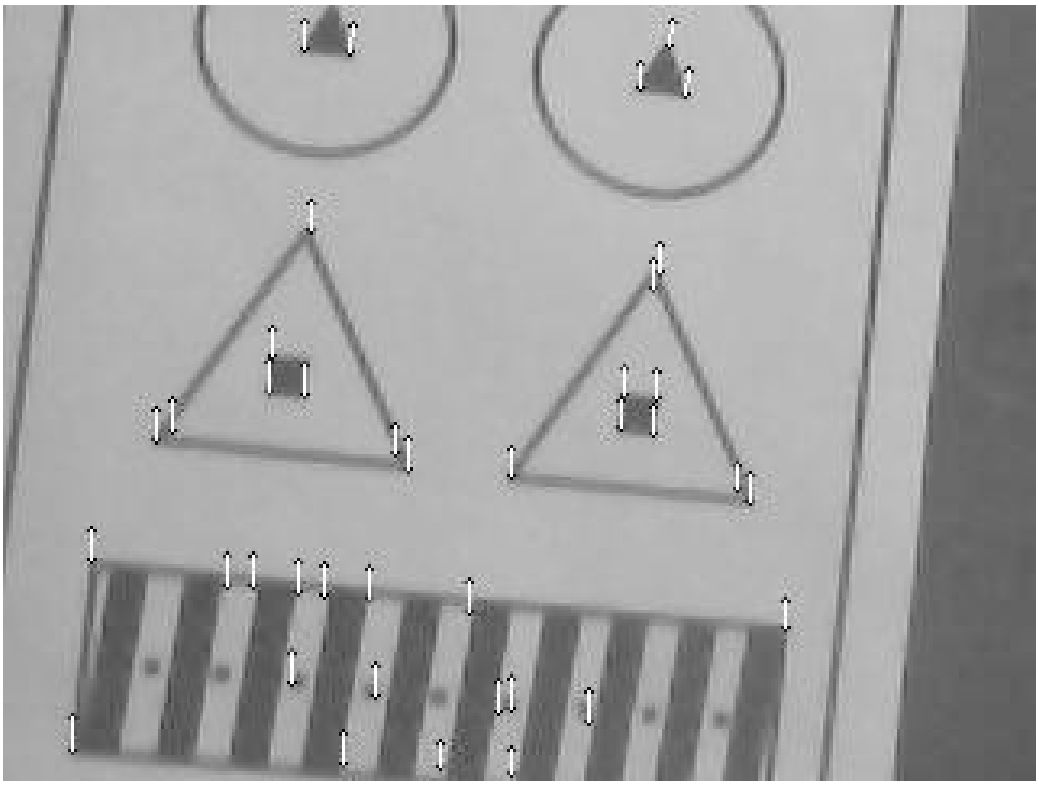


Figura 4.9: Flujo óptico mejorado

Posteriormente se probo que detectara la dirección del movimiento debido al flujo y a que velocidad en pixeles por segundo.

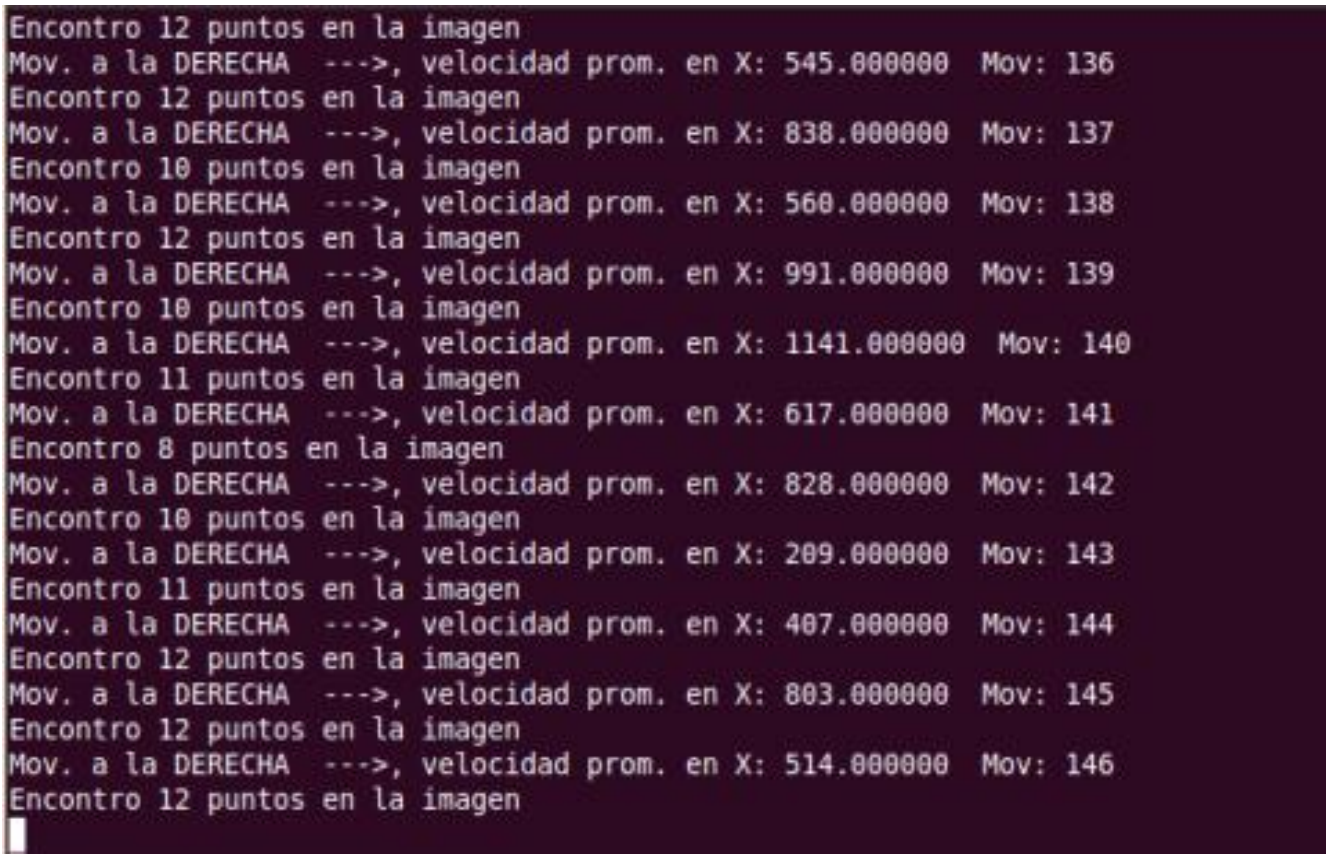


Figura 4.10: Detección de dirección de movimiento con flujo óptico

Al reajustar los parámetros se perdió un poco de sensibilidad en la lectura pero los datos fueron todos correctos.

Con respecto al algoritmo de rastreo Haar training, después de probar con diferentes patrones se encontró uno en el que mejoró el desempeño, el archivo de entrenamiento se creo a partir del siguiente patrón.

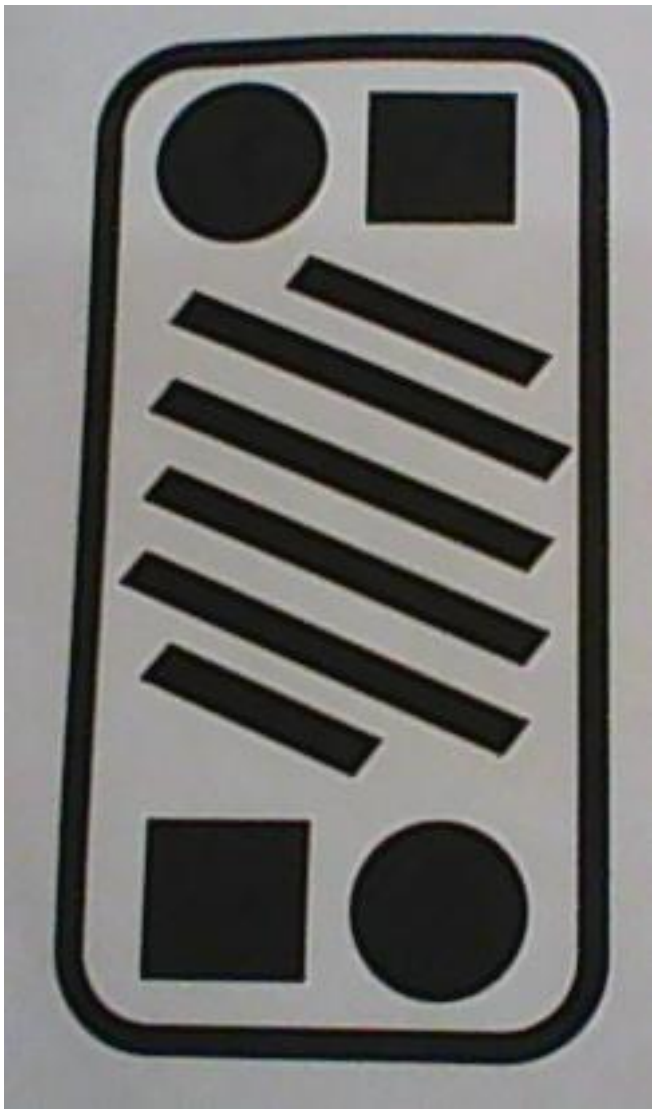


Figura 4.11: Patron de entrenamiento

Para el entrenamiento se ocuparon 350 imágenes positivas con diferentes inclinaciones y condiciones de luz, el entrenamiento se realizó en la computadora de escritorio con visual studio 2008. El archivo de salida se probo en el sistema embebido obteniendo los siguientes resultados.

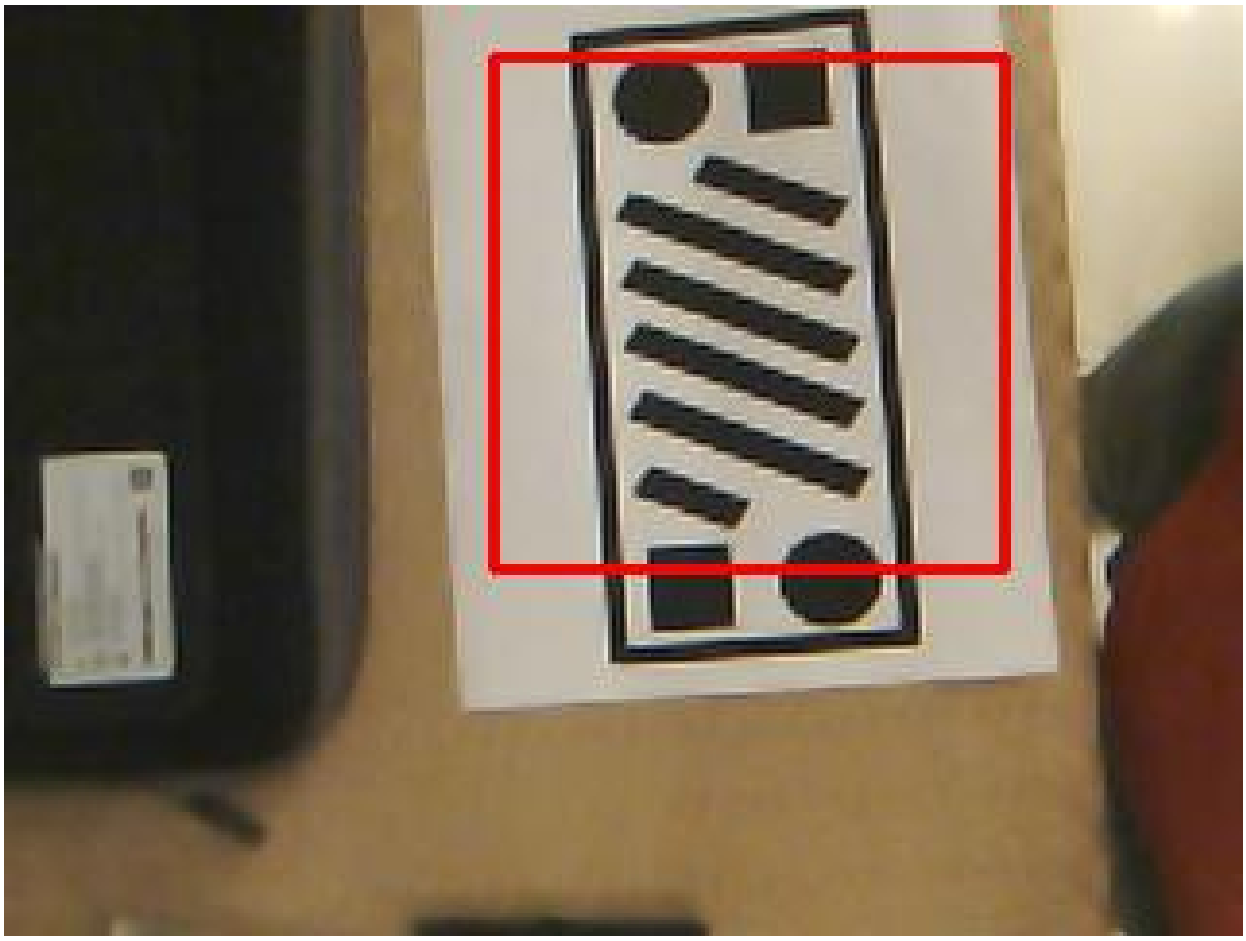


Figura 4.12: Objeto detectado por HaarTrainnig

Con esto se comprobó que logra detectar el patrón de forma correcta, lo siguiente fue probar si era capaz de detectar varios patrones al mismo tiempo, obteniendo lo siguiente.

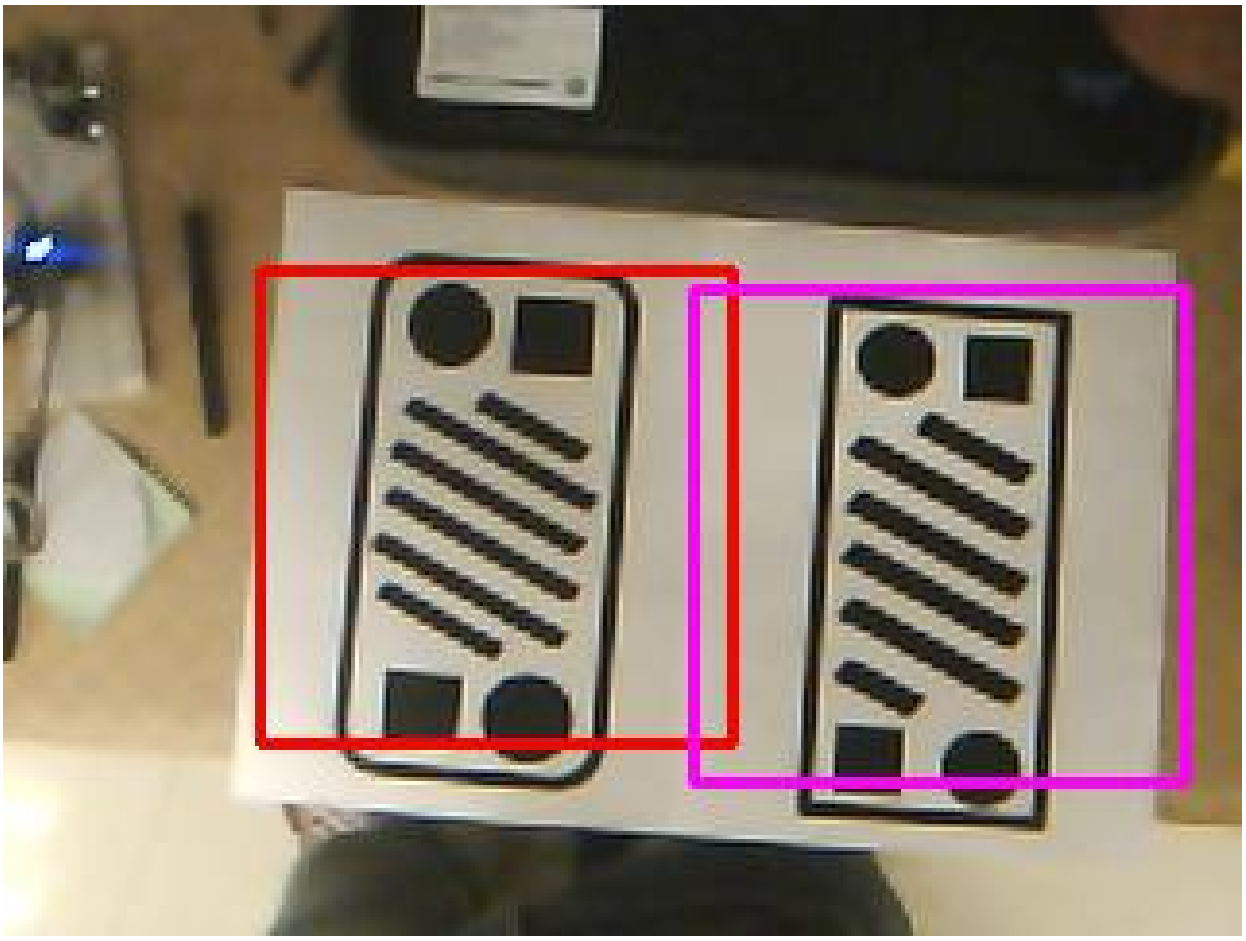


Figura 4.13: Varios objetos detectados por HaarTrainnig

Una vez comprobado que puede detectar varios patrones a la vez, se tiene que hacer que el algoritmo logre detectar el mas cercano y dado que nuestro patrón de entrenamiento tiene una medida estándar se puede deducir que el obstáculo más cercano es el que detecta mas grande e incluso a partir de un polinómio se puede calcular la distancia a la que se encuentra.

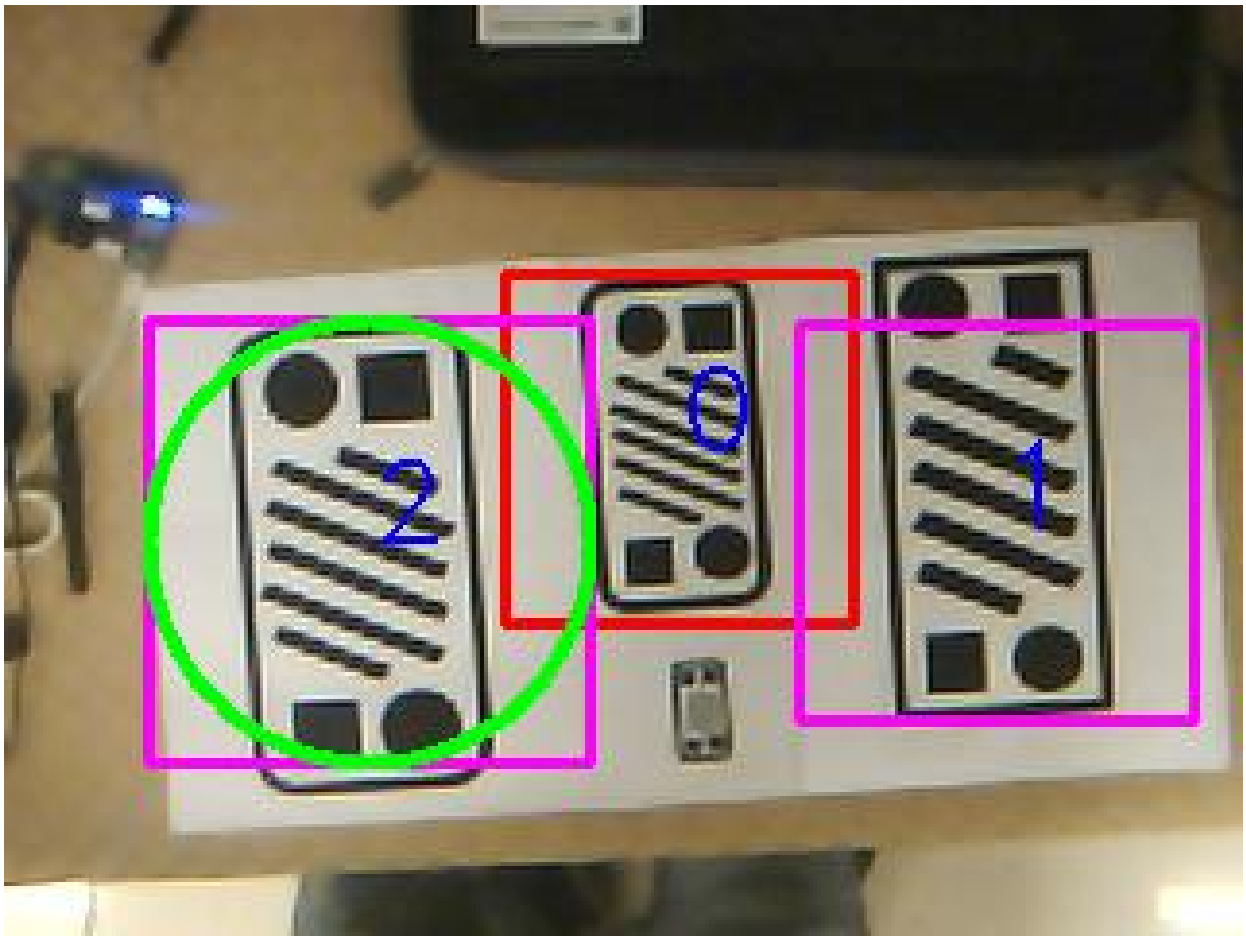


Figura 4.14: Objeto mas grande detectado como prioridad

Como se puede observar de los tres patrones detectado se enfoca en el mas grande que seria en patrón 2 (encerrado en circulo verde) a partir de esto le dará prioridad ya que es el obstáculo que encuentra más cerca.

Una vez comprobados los dos algoritmos principales se procedió a unirlos en uno solo dentro de la plataforma embebida sin embargo el hecho de adquirir por dos cámaras diferentes al mismo tiempo, mas el procesamiento de flujo óptico y además el procesamiento del rastreo hizo que el proceso en general fuera demasiado lento por lo que concluyó que habría de quitarle carga al procesador y eliminar un algoritmo del proceso

que realiza el gumstix y compensarlo de otro modo.

Fue por esta razón que se quedo embebido en el gumstix solo el algoritmo de rastreo y se busco remplazar la parte de flujo óptico de alguna otra forma, es aquí donde se incorpora el uso del sensor de flujo óptico ADNS8030.

Este sensor nos ofrece la ventaja que en calculo lo hace dentro del mismo circuito de la tarjeta del sensor y solo se envían las lecturas hacia el autopiloto.

A continuación se muestran gráficas de simulación de estabilización en modo estacionario.

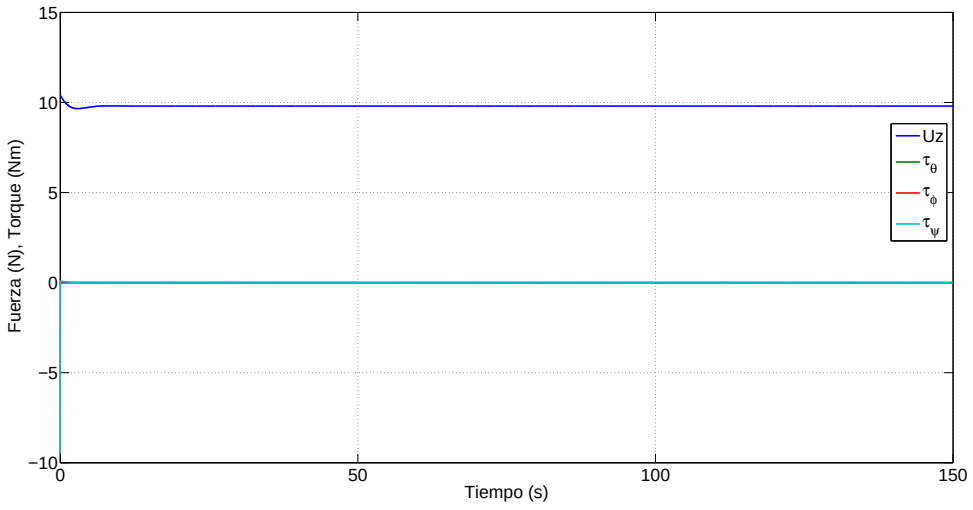


Figura 4.15: Controles

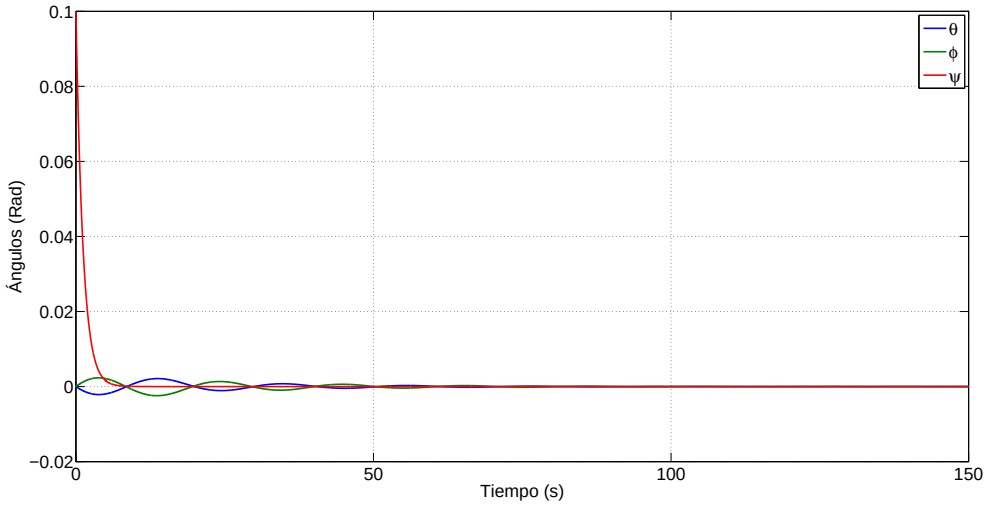


Figura 4.16: Orientación

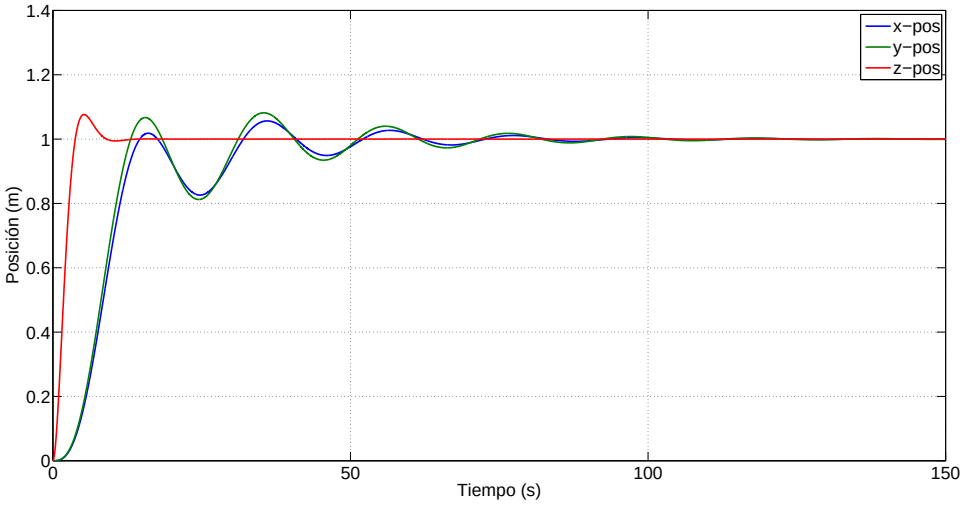


Figura 4.17: Posición

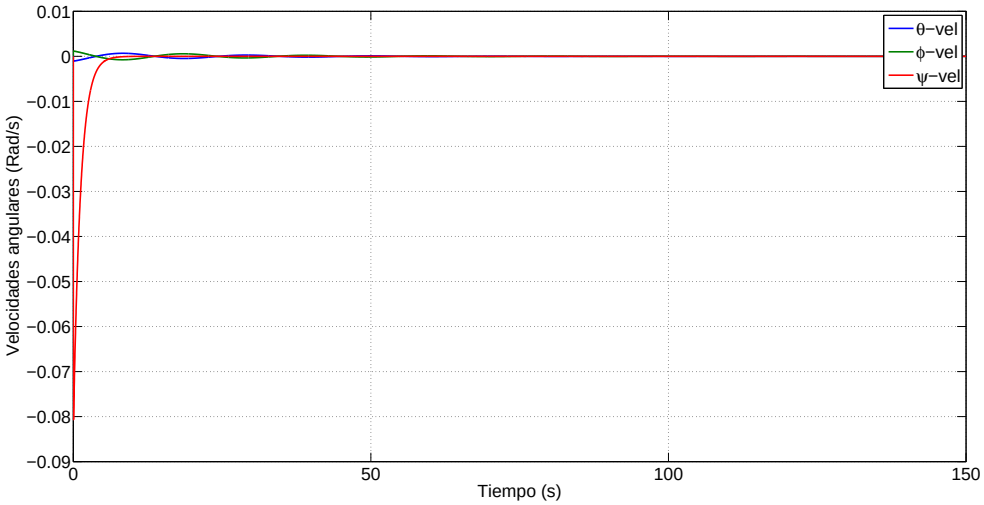


Figura 4.18: Velocidad angular

En vuelos de prueba para controlar la altura y el avance se utiliza el radio, esto debido a que el control de altura que se intento emplear no es lo suficientemente eficaz para mantenerlo fijo a medio metro del suelo,y el avance por seguridad en caso de que no detectara el obstáculo evitar una colisión.

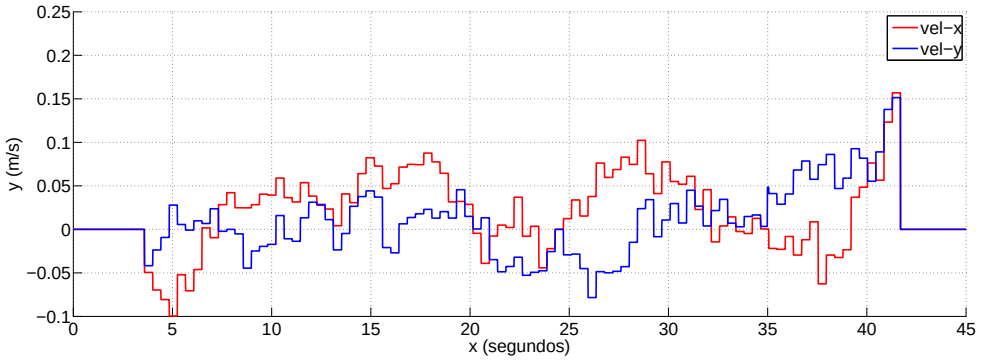


Figura 4.19: Velocidad traslacional

Las lecturas anteriores se obtienen del sensor de flujo óptico, y como se puede observar empieza con una velocidad en cero después se empieza a mover lentamente y cerca del

segundo 35 detecta el obstáculo por lo que va aumentando su velocidad en el eje y.

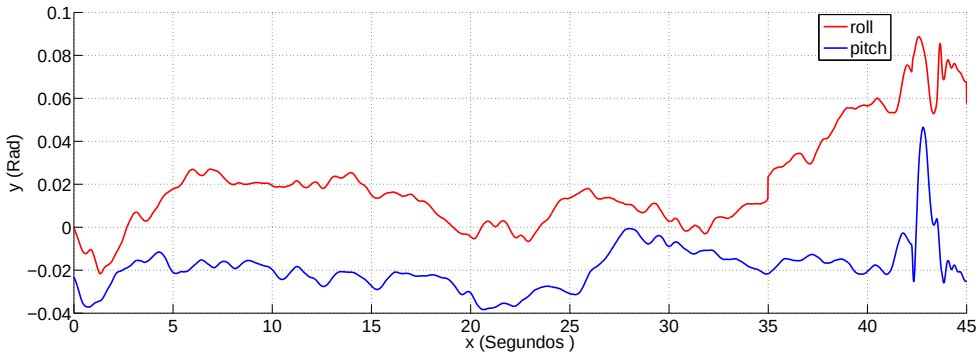


Figura 4.20: Pocisión angular en Pitch y Yaw

Se puede corroborar lo anterior con la posición angular en pitch y yaw, tambien se puede observar el mismo patrón que en la gráfica de la velocidad traslación, cerca del segundo 35 se empieza a notar una inclinación mayor en el angulo de roll para lograr evitar el obstáculo

Capítulo 5

Conclusiones y trabajo futuro

Se concluye que los algoritmos aquí implementados ya son capaces de detectar los obstáculos para poder evadirlos ya que ademas de detectarlos ya se es posible hacer una estimación de la distancia a la que se encuentran, sin embargo por ahora solo sirve para estos obstáculos específicamente, si que quisiera detectar otros diferentes se necesitaría realizar el entrenamiento para los posibles diferencia de estos o bien implementar la parte de segmentación de ser viable, tomando en cuenta que se tiene que tener alguna idea de la forma y el color de los obstáculos para poder emplear esta técnica.
Con el flujo óptico ya es posible estimar la velocidad y la dirección de los desplazamientos apoyado de OpenCV, pero debido a la cuestión de la carga de procesamiento del gumstix no es posible sostener los procesamientos de rastreo y flujo al mismo tiempo, por lo que se concluye que es necesario remplazar un algoritmo.

El uso del sensor de flujo óptico ADNS3080 tiene una tasa de captura muy rápida, sin embargo la sensibilidad es pobre y la es demasiado vulnerable a los cambios y falta de iluminación por lo que no compensa del todo bien el desempeño del algoritmo corriendo en gumstix.

Como trabajo futuro se podría buscar otro buscar otro sensor de flujo óptico con un mejor desempeño para tener una mejor lectura de la velocidad traslacional, ademas se le podría agregar un sistema de orientación de posición como un GPS para trazar una trayectoria, otra cosa que haría seria un control de altura eficiente ya que el que se tiene ahora va derivando por la lectura del barómetro incluido en el Pixhawk.

```
if(aux_2>2000)
  {
  if(flag_alt==1){
      z_des = baro_alt;
      flag_alt = 0;
  }
  errorz = z_des - baro_alt;
  errorz_dot = - delta_alt;
  u_z = kp_z * errorz  + kd_z * errorz_dot;
  tau_z = (u_z /(cos(pitch)*cos(roll)));

  if(tau_z > 60)
     tau_z = 60;
  if(tau_z < - 40)
     tau_z = - 40;
  }
  else
  {
  flag_alt=1;
  tau_z=0;
  z_des=0;
  }
```

Figura 5.1: Código control de altura

```
if(dist_obst_m > 0)
{
    if(lado>110)
    {
    tau_avoid = dist_obst_m*10;
    if(tau_avoid>15){
        tau_avoid = 15;
    }
    }else
    {
    tau_avoid = -dist_obst_m*10;
    if(tau_avoid<(-15)){
        tau_avoid = -15;
    }
    }
}else
{
    tau_avoid = 0;
}
```

Figura 5.2: Código control de evasión

```
for( i=0; i <(ventana ? ventana->total : 0); i++)
{
    r1=(CvRect*)cvGetSeqElem(ventana, i);
    cvRectangle(frame, cvPoint(r1->x, r1->y), cvPoint(r1->x+r1->width, r1->y+r1->height), CV_RGB(255, 0, i*255), 2, 8, 0);
    sprintf(buffer,"%d",i);
    cvPutText(frame,buffer,cvPoint(r1->x+r1->width/2,r1->y+r1->height/2),&font,cvScalar(255));
    //obj=1;
    printf("Objeto %d \n",i);
    printf("Largo: %d  Ancho: %d \n",r1->width,r1->height);
    area_obj=(r1->width)*(r1->height);
 if(area_obj>area_mayor) mayor=i;
 if(i==(ventana->total-1))
                {
     s1=(CvRect*)cvGetSeqElem(ventana, mayor);
     cvCircle(frame,cvPoint(s1->x+s1->width/2,s1->y+s1->height/2), s1->width/2 ,CV_RGB(0, 255, 0),3);
     distancia=0.0087*((s1->width)*(s1->width))-3.1656*(s1->width)+328.98;
     printf("Distancia: %f \n",distancia);
     if(distancia<50) obj=1;
                }
}
```

Figura 5.3: Código detección de obstáculos

Bibliografía

[1] Franz Andert, Florian-M Adolf, Lukas Goormann, and Jörg S Dittrich. Autonomous vision-based helicopter flights through obstacle gates. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 259–280. Springer, 2010.

[2] Michael Bleyer, Margrit Gelautz, and Christoph Rhemann. *Colour segmentation-based computation of dense optical flow with application to video object segmentation.* Citeseer, 2005.

[3] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[4] Václav Endrych. Řízení a stabilizace bezpilotní helikoptéry sledující dynamicky se měnící trajektorii. 2014.

[5] Nils Gageik, Michael Strohmeier, and Sergio Montenegro. An autonomous uav with an optical flow sensor for positioning and navigation. *INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS*, 10, 2013.

[6] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2188–2202, 2011.

[7] Nicolas Guenard, Tarek Hamel, and Robert Mahony. A practical visual servo control for an unmanned aerial vehicle. *Robotics, IEEE Transactions on*, 24(2):331–340, 2008.

[8] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[9] Dominik Honegger, Pierre Greisen, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5177–5182. IEEE, 2012.

[10] Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1736–1741. IEEE, 2013.

[11] Adam Kozłowski and Aleksandra Królak. Teaching image processing and pattern recognition with the intel opencv library. In *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2009*, pages 750205–750205. International Society for Optics and Photonics, 2009.

[12] Liu Kun, Guo Lei, Li Huihui, and Chen Jingsong. Fusion of infrared and visible light images based on region segmentation. *Chinese Journal of Aeronautics*, 22(1):75–80, 2009.

[13] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Pattern Recognition*, pages 297–304. Springer, 2003.

[14] Rogelio Lozano. *Unmanned aerial vehicles: Embedded control*. John Wiley & Sons, 2013.

[15] Carol Martínez, Iván F Mondragón, Pascual Campoy, José Luis Sánchez-López, and Miguel A Olivares-Méndez. A hierarchical tracking strategy for vision-based applications on-board uavs. *Journal of Intelligent & Robotic Systems*, 72(3-4):517–539, 2013.

[16] LE Munoz, P Castillo, and P Garcia. Observer-control scheme for autonomous navigation: Flight tests validation in a quadrotor vehicle. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 795–804. IEEE, 2013.

[17] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking.

[18] Morimichi Nishigaki. Color segmentation-based optical flow computation and motion segmentation.

[19] Hugo Romero, Sergio Salazar, and Rogelio Lozano. Real-time stabilization of an eight-rotor uav using optical flow. *Robotics, IEEE Transactions on*, 25(4):809–817, 2009.

[20] Mario Sarcinelli-Filho, HJA Schneebeli, and Eliete MO Caldeira. Using optical flow to control mobile robot navigation. In *Proceedings of the*, volume 15, 2002.

[21] Z Zamudio, R Lozano, J Torres, and E Campos. Stabilization of a helicopter using optical flow. In *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, pages 1–6. IEEE, 2011.