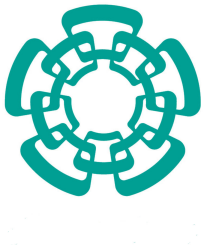


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Laboratorio de Tecnologías de Información

Interacción hombre-robot con vehículos aéreos no tripulados basada en visión

Tesis que presenta:

Daniel Soto Guerrero

Para obtener el grado de:

Maestro en Ciencias en Computación

Director de la Tesis:
Dr. José Gabriel Ramírez Torres

Cd. Victoria, Tamaulipas, México. Noviembre, 2012

La tesis presentada por Daniel Soto Guerrero fue aprobada por:

Dr. Pendiente

Dr. Pendiente

Dr. José Gabriel Ramírez Torres, Director

Cd. Victoria, Tamaulipas, México., 14 de Noviembre de 2012

Dedicatoria Pendiente

Agradecimientos

Agradecimientos

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Resumen

Interacción hombre-robot con vehículos aéreos no tripulados basada en visión

por

Daniel Soto Guerrero

Maestro en Ciencias del Laboratorio de Tecnologías de Información
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2012
Dr. José Gabriel Ramírez Torres, Director

En este trabajo de tesis se propone el desarrollo de una interfaz entre un vehículo aéreo no tripulado (UAV por sus siglas en inglés) y una persona. La interfaz realiza el rastreo de una persona y el reconocimiento de sus ordenes gestuales, por medio de visión por computadora. Generalmente, se emplea una estación de trabajo fija en tierra para implementar el procesamiento digital de imágenes, limitando la aplicación del vehículo al rango de comunicación entre ambas partes. Para evitar el uso de una estación fija en tierra para el control del dron, y lograr una mayor autonomía del uso de la interfaz, se propone que el procesamiento computacional de las imágenes se realice en un dispositivo móvil portado por el usuario.

El desarrollo de la interfaz propuesta en este trabajo, requirió que se diera solución a algunas dificultades técnicas, relacionadas tanto con la comunicación con el UAV como con las capacidades computacionales reducidas de la plataforma. Igualmente, se discute la unión sinérgica del decodificador de video con el procesamiento digital de imágenes, como estrategia para aumentar el rendimiento general de la interfaz.

Con este trabajo se pretende realizar un aporte al estudio de interacción hombre-robot, aplicado a vehículos aéreos no tripulados. La interfaz propuesta se basa en el reconocimiento de gestos que el usuario realiza con su dorso y brazos, capturados por una cámara de video montada en el vehículo aéreo. Se propone también el uso de un dispositivo láser para, por medio de triangulación visual, estimar la distancia entre el vehículo y el usuario.

Finalmente, se muestran y se discuten los resultados finales, los cuales prueban que la interfaz opera satisfactoriamente, siguiendo al usuario y clasificando sus gestos de manera correcta.

Abstract

Vision based human-robot interaction with unmanned aerial vehicles

by

Daniel Soto Guerrero

Master of Science from the Information Technology Laboratory
Research Center for Advanced Study from the National Polytechnic Institute, 2012
Dr. José Gabriel Ramírez Torres, Advisor

In this thesis work we propose the development of an interface between a human and a UAV (Unmanned Aerial Vehicle a.k.a. drone), based on visual gesture recognition and user's upper body tracking. Generally, all computational tasks run in ground-based work stations, which deliver performance but they are not meant for ubiquitous applications. In order to improve the usage of the interface, we propose a mobile platform, worn by the user, to perform all computational tasks. This document cover all technical difficulties solved during the development of this interface, including wireless communication troubleshooting and all strategies implemented to make good use of available processing performance. We describe how we mixed drone's video decoder and digital image processing algorithms to achieve this.

This research work complements what has been observed and reported recently in the field of human-machine interaction. This interface differs from others because of its platform and its visual recognition algorithms, which segment and classify user's upper body through the UAV's on board camera. We also tested one laser range finder to estimate the user's relative position to the drone. We show and discuss all final results that prove the usefulness of the proposed interface to track and recognize body gestures.

Nomenclatura

Acrónimos principales

UAV	Unmanned Aerial Vehicle.
IMU	Inertial Measurement Unit.
MEM	Micro Electrical Mechanism.
SLAM	Simultaneous Localization And Mapping.
GPU	Graphical Processing Unit.
SoC	System on Chip.
KIB	Kibibyte.
GOB	Group of Blocks.

Símbolos y notación

$f(\mathbf{X})$	Función objetivo
$\mathbf{F}(\mathbf{X})$	Función vectorial, o vector de funciones objetivo
$\mathcal{F}$	Región factible
M	Número de objetivos
n	Número de variables de decisión
N	Tamaño de la población (número de soluciones)
P^*	Conjunto de puntos de referencia en el espacio de las funciones objetivo
P	Población de soluciones candidatas
$\mathcal{U}$	Espacio de búsqueda
$\mathbf{X}$	Vector de diseño, o vector de variables de decisión
x	Variable de decisión
Z	Conjunto aproximación: soluciones encontradas por un optimizador multiobjetivo

1

Introducción

Este capítulo describe los antecedentes y motivaciones para desarrollar este trabajo de tesis. Abarca los objetivos generales y específicos, el planteamiento del problema y la hipótesis de trabajo que dio origen al presente desarrollo.

1.1 Antecedentes y motivación

El pionero en aviación francés Etienne Oehmichen comprobó que era posible la construcción del cuadricóptero[1] con la fabricación del Oehmichen No.2 en 1922. Dicha aeronave es el primer antecedente histórico de un cuadricóptero práctico. En la actualidad, los cuadricópteros son comúnmente diseñados para ser vehículos aéreos no tripulados[2]; por lo tanto, son de menor tamaño y pueden ser conducidos en interiores y exteriores. Los avances en tecnologías de semiconductores y MEMs[3] han hecho posible el desarrollo de pequeñas IMUs[4], útiles para estimar la orientación del vehículo aéreo en el espacio tridimensional. Dichas IMUs ya son incorporadas en las tarjetas de circuito impreso de

[1] Se le llama cuadricóptero por las 4 hélices que lo propulsan.
[2] También conocidos como UAV, acrónimo en inglés para: *Unmanned Aerial Vehicle*, vehículo aéreo no tripulado.
[3] Acrónimo en inglés: *Micro Electrical Mechanisms*, dispositivos micro electromecánicos.
[4] Acrónimo en inglés: *Inertial Measurement Unit*, unidad de medición inercial.

los controladores a bordo de los vehículos.

En los vehículos aéreos no tripulados, también denominados *drones*, el control se realiza de manera remota haciendo uso de alguna tecnología inalámbrica como: radiofrecuencia, zigbee, WiFi, etc. El operador toma el control de la aeronave a través de una interfaz, interpreta la información disponible y actúa en consecuencia. De lo expuesto, sobresalen tres puntos:

1. El vehículo aéreo no es completamente autónomo.

2. El vehículo aéreo no tiene conocimiento del ambiente que le rodea.

3. El usuario usa necesariamente una interfaz para controlarlo.

Con respecto al primer punto, en el área de vehículos terrestres autónomos han sido propuestas varias soluciones a la navegación autónoma en un ambiente inicialmente desconocido. Dichas soluciones son clasificadas como *Simultaneous Localization And Mapping* (SLAM) y han sido estudiadas, desarrolladas y probadas en años recientes en robots terrestres; no es el caso de los vehículos aéreos no tripulados o drones.

El segundo punto implica que el vehículo aéreo no procesa ninguna información sobre el ambiente que le rodea. Toda la información generada por los sensores que porta es enviada a la base de control para su interpretación. No existe ningún sistema de soporte para la toma de decisiones, independiente de la base de control en tierra, que lo haga adoptar un modelo de control proactivo.

Acerca del tercer punto, las interfaces con un usuario siempre son diseñadas tomando en cuenta simples principios de diseño para mejorar la interacción y la experiencia del usuario. Una interfaz debe ser simple y eficiente, así facilitará que el usuario cumpla su objetivo. Regularmente se desarrollan tomando como modelo alguna analogía con la que el usuario está familiarizado. Las mejores interfaces para la manipulación remota de aeronaves son de tipo hápticas[5] y se asemejan a una cabina de avión. Cabe mencionar que todas las decisiones son tomadas por el piloto en tiempo de vuelo.

Para este trabajo de tesis se usará el cuadricóptero modelo AR, desarrollado por la compañía Parrot (mostrado en la Figura 1.1). Hay dos escenarios de operación del dron ya desarrollados y

[5]En este contexto, una interfaz háptica es aquella que provee información al usuario no sólo de manera visual y auditiva, sino que provee información sensorial adicional como frío, calor, movimiento físico, fuerza, etc.

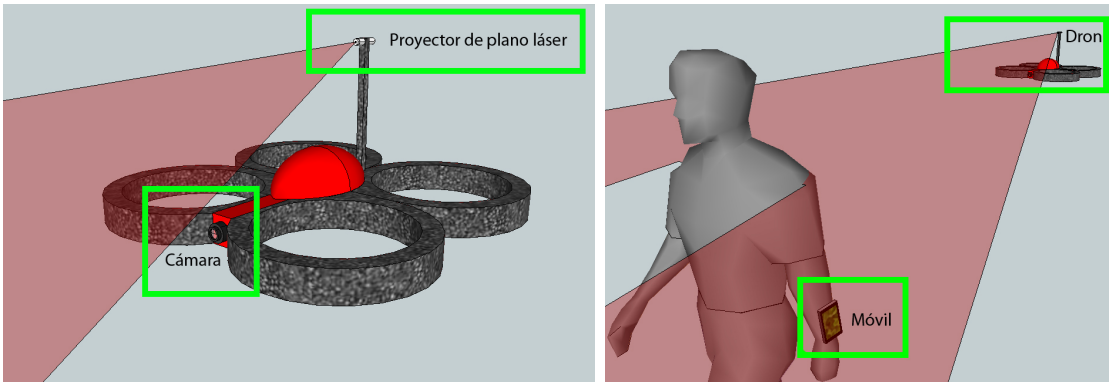
Figura 1.1: Dron AR, del fabricante Parrot.

reportados. En el primero y de propósito lúdico, el dron es controlado a distancia por el usuario con un dispositivo de comunicación móvil. En este escenario, el usuario está a cargo de todas las maniobras. En el segundo escenario, el dron es controlado por una estación de trabajo fija en tierra, donde la información es recibida y procesada por algoritmos con algún propósito de navegación. El rango de acción del vehículo depende entonces del alcance de la señal transmitida entre ambas partes.

Un tercer escenario, que no ha sido reportado previamente y que es el enfoque presentado en esta tesis, se muestra en la Figura 1.2; en este escenario, el dron se controla a través de órdenes gestuales percibidas por medio de la cámara a bordo del dron y procesadas en un dispositivo móvil. Bajo estas premisas y lo expuesto anteriormente, se propone una estrategia que desplace la carga de procesamiento a un dispositivo móvil basado en tecnología Android y lo separe de una estación de procesamiento fija en tierra; con el objetivo de incrementar la autonomía de esta interfaz y sólo limitarla por la carga de las baterías. Uno de los objetivos de la implementación de esta estrategia es tomar en cuenta las restricciones de procesamiento en una plataforma móvil, pues los recursos computacionales disponibles no se comparan con los disponibles en una computadora personal.

Actualmente, los dispositivos móviles basados en Android han aumentado en capacidades de cómputo y algunos fabricantes los han dotado de una unidad de procesamiento de gráficos o GPU[6]. El objetivo principal de dichos GPUs es mejorar la experiencia de navegación en internet y de videojuegos para los usuarios, por lo cual, poco trabajo se ha realizado en el procesamiento de propósito general

[6]GPU: *Graphical Processing Unit.*



(a) (b)

Figura 1.2: Tercer escenario de operación. (a) Dron con la cámara y proyector láser. (b) Escenario de operación.

usando GPUs de sistemas móviles. Sin embargo, el acceso a una *tablet* con un SoC de 4 núcleos de procesamiento y la implementación de varias estrategias enfocadas en la correcta administración de los recursos, mantuvo el rendimiento del rastreo en niveles cercanos a los 15 cuadros por segundo.

1.2 Planteamiento del problema

La sección anterior expone, de manera general, la operación del sistema bajo la estrategia planteada. Este escenario de operación representa la propuesta de este trabajo de tesis y se muestra en la Figura 1.2; en dicha ilustración, el dron no es controlado desde una estación de trabajo fija, sino a través de un dispositivo móvil que el usuario sujeta con su mano. El dron, con una cámara a bordo, enfoca la espalda del usuario y entrega al dispositivo móvil un flujo de video con las acciones del usuario. Los comandos gestuales que el usuario desee ejecutar, los deberá realizar dentro del campo visual de la cámara, para que sean capturados y enviados al dispositivo móvil. Su procesamiento e interpretación por el dispositivo móvil modificará el comportamiento del dron. Un análisis más profundo sobre los detalles de operación de esta interfaz, reveló las complejidades involucradas que serán descritas a continuación.

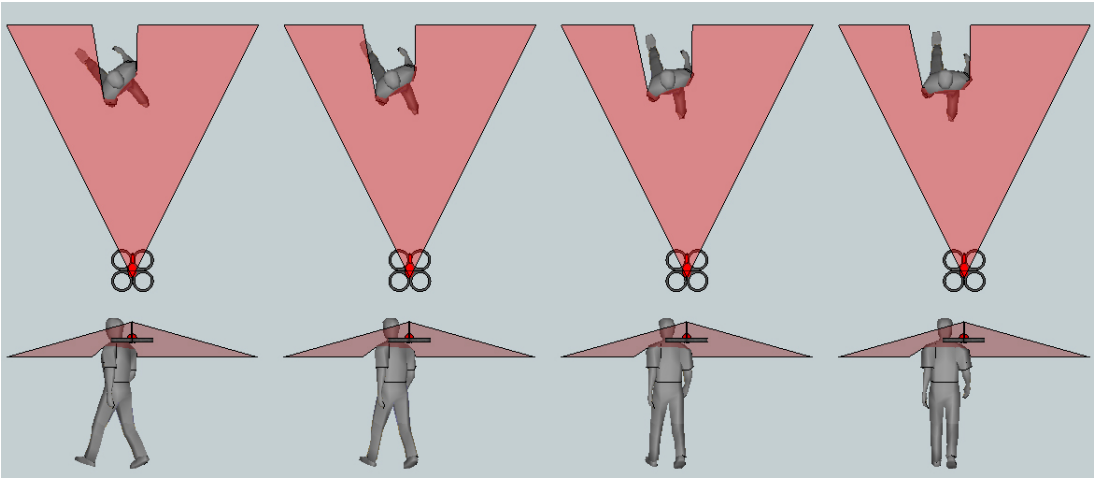


Figura 1.3: Plano láser proyectado en la persona.

1.2.1 Seguimiento del usuario

Para que la interfaz sea práctica, la cámara del dron debe estar siempre enfocada sobre el usuario. Solo así es posible obtener la mejor fotografía del gesto que el usuario realiza con su dorso y brazos. Para estimar la distancia y la orientación a la que se encuentre el usuario del dron, se propuso usar un proyector de plano láser desde el vehículo aéreo.

El plano láser se proyecta en la espalda del usuario, con el objetivo de estimar la distancia a la persona. Es una idea sencilla que se ilustra en la Figura 1.3. Como se puede observar, la orientación de la persona hace variar la inclinación de la línea proyectada en su espalda. La seguridad fue prioridad cuando se manejaron los dispositivos láser por su capacidad de destruir la retina del ojo humano. Aunque el láser empleado es de clase II[7], seguir al usuario desde atrás minimiza la posibilidad de causar un daño a sus ojos.

También es importante considerar tanto el movimiento propio del dron como el movimiento del usuario para asegurar que este último permanezca la mayor parte del tiempo en el encuadre de la cámara. Consecuentemente, la estrategia para seguir al usuario desde el aire contempló la dinámica del dron y la persona en movimiento.

[7]La potencia pico del láser de clase II, con el que se realizaron las pruebas, es de 10mW.

1.2.2 Comandos de la interfaz

Como se mencionó previamente, los comandos los presenta el usuario al dron dentro del campo visual de su cámara, mientras que el procesamiento, reconocimiento y ejecución se realizan en un dispositivo móvil que porta el usuario. El reconocimiento del comando se realiza con base en dos técnicas: (1) segmentación con procesamiento digital de imágenes y (2) clasificación de patrones.

Un comando del usuario está construido de dos eventos:

1. *Presentar gesto*: El usuario portando el dispositivo móvil, ubicado dentro del campo visual de la cámara, presenta uno de los gestos válidos usando su dorso y brazos.

2. *Procesamiento*: El dispositivo móvil procesa los cuadros del flujo del video y empieza su segmentación.

El usuario presenta un gesto usando su dorso y brazos, la silueta segmentada es interpretada como un gesto hasta que el dispositivo móvil compara la silueta contra todas las referencias existentes y la clasifique.

1.2.3 El problema

El problema abordado puede entonces plantearse de la siguiente manera:

Dado un vehículo aéreo no tripulado que cuenta con un sensor extereoceptivo láser y una cámara monoscópica, lograr el seguimiento e interacción gestual con un usuario humano, ejecutando en un dispositivo móvil los algoritmos de control del cuadricóptero y de clasificación de gestos.

1.2.4 La hipótesis

Es posible diseñar e implementar los algoritmos necesarios para que, desde un dispositivo móvil, se gobierne a un vehículo aéreo no tripulado para que interactúe de manera natural con un usuario, mediante gestos que éste realice con sus brazos.

1.3 Objetivos generales y específicos del proyecto

Esta sección abarca los objetivos de esta tesis, una breve descripción y su contextualización.

1.3.1 Objetivo general

El objetivo de esta tesis es contribuir al estado del arte en interacción hombre-máquina, con el desarrollo de una interfaz que permita a un usuario dar instrucciones gestuales a un vehículo aéreo no tripulado, a través de un sistema de visión por computadora y procesamiento digital de imágenes en dispositivos móviles.

En la Figura 1.4 se describe, con un diagrama de flujo, la operación del sistema descrito.

1.3.2 Objetivos particulares

Los puntos clave de avance son listados en esta sección. Se señalan algunos de los resultados por obtener durante el transcurso de esta investigación.

1.3.2.1. Comunicación y control del dron con un dispositivo móvil

Para que el sistema no dependa de una estación fija en tierra, se require de algoritmos de control que se ejecuten desde un dispositivo móvil, lo que hace necesario la interacción entre el dron y el dispositivo móvil a través de una conexión WiFi *ad hoc*. Para ello, hay que proponer un modelo de control que contemple la dinámica del usuario y del dron en el aire, con lo cual se logrará un rastreo eficaz del usuario.

Se cuenta con el SDK distribuido por el fabricante del dron para los siguientes sistemas operativos:

1. Windows.

2. iOS, para los dispositivos iPhone, iPad y iPod de Apple Inc.

3. Linux.

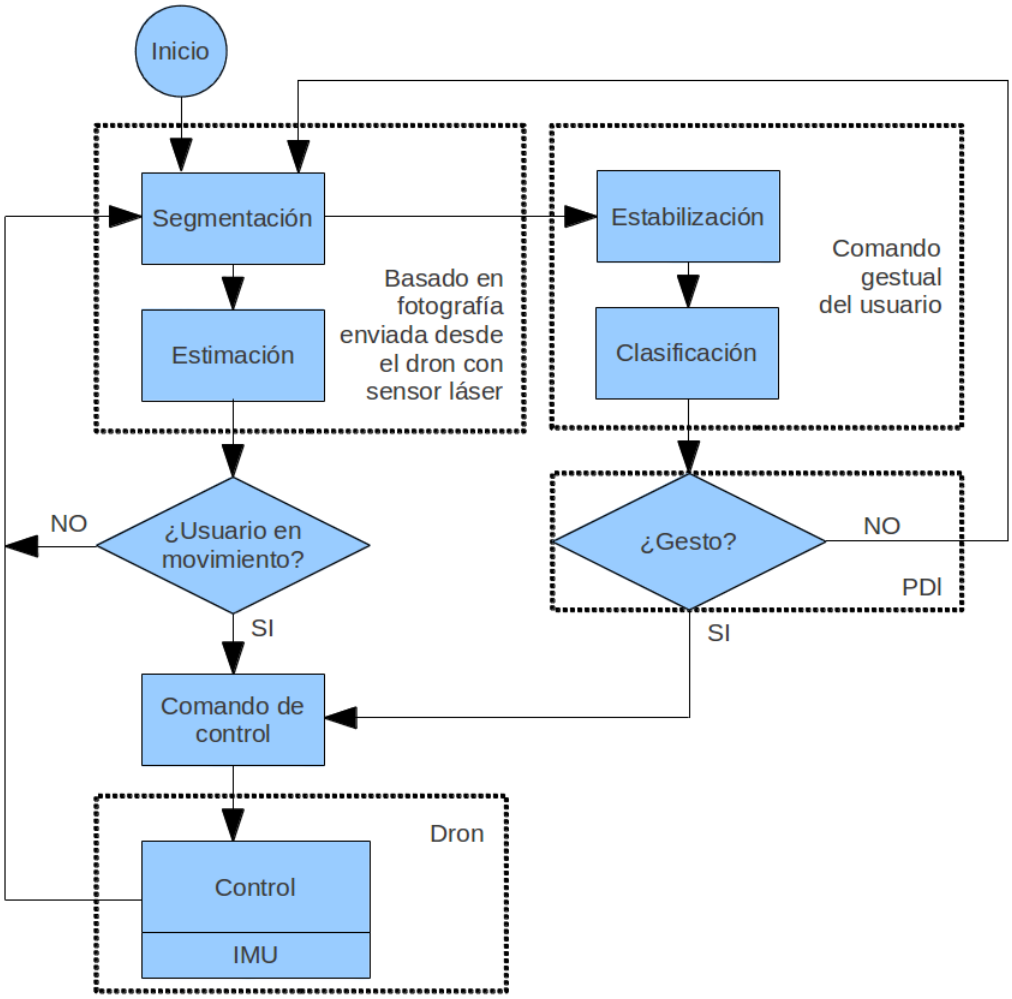


Figura 1.4: Diagrama de flujo de la operación del sistema propuesto.

4. Android.

De estas cuatro plataformas, sólo Android se ejecuta en dispositivos móviles y sus herramientas de desarrollo están disponibles de forma gratuita. Contrastablemente, Android es la plataforma para la cual el SDK ofrece un soporte inferior.

Se pretende contribuir a la comunidad de desarrolladores con una aplicación de código abierto y gratuita que pueda usarse libremente como base para construir otras aplicaciones; esto último, difiere de la política de Apple que obliga a todos los desarrolladores a adquirir una licencia para poder ejecutar su aplicación en un dispositivo con iOS.

Por las razones anteriores, en Android se desarrollará la aplicación base que servirá para alcan-

zar los objetivos de esta tesis y pueda compartirse después con la comunidad de desarrolladores e investigadores. Este desarrollo será una producción secundaria de este trabajo de tesis.

1.3.2.2. Segmentación y clasificación de comandos gestuales

Para conocer el comando que el usuario desea ejecutar, la imagen que se capture con la cámara a bordo del dron se segmentará y clasificará de acuerdo a la silueta que el usuario presenta con su dorso y brazos.

La segmentación de la silueta pictórica se realizará con la fusión de dos técnicas: segmentación por colores del objeto de interés (en este caso, la ropa del usuario) y con un filtro de partículas para incrementar la eficiencia de la detección del usuario en la imagen.

Existen estrategias probabilísticas para segmentación de extremidades del cuerpo humano basadas en el color de la piel. Son entrenadas con una gran gama de tonalidades de piel en distintas condiciones de iluminación y el modelo probabilístico obtenido puede usarse para segmentar áreas de piel en una imagen [23]. De una forma similar, la aplicación a desarrollar efectuará una inicialización del color a segmentar, empleando las capacidades táctiles del dispositivo móvil.

1.3.2.3. Implementación y caracterización de un sensor láser

Se desarrollará una estrategia eficaz para conocer la distancia y orientación a la que se encuentra la persona del cuadricóptero, basado en la proyección de un plano láser sobre la espalda del usuario.

En robótica, los sensores suelen clasificarse en extereoceptivos e intereoceptivos, según midan variables externas e internas del robot, respectivamente. El altímetro ultrasónico con el que cuenta el dron se clasifica como un sensor extereoceptivo, por medir la distancia al suelo en tiempo de vuelo. Por otro lado, la IMU es un sensor intereoceptivo por estimar la orientación en el espacio tridimensional del dron. El sensor del estado de carga de batería del dron es también un sensor intereoceptivo.

Como parte de este trabajo de investigación, se desea dotar al dron de un sensor extereoceptivo adicional, que sea capaz de proveer información referente al ambiente, la distancia entre el dron y el usuario específicamente. Para cumplir este objetivo, se montará en el dron un proyector de plano

láser, como se ilustra en la Figura 1.2a. El plano láser se proyecta en la espalda del usuario y la escena se retratará por la cámara montada en el dron. En la Figura 1.3 se ilustra el efecto que causa el cambio en la orientación del usuario en la proyección del láser sobre el dorso del usuario. Estos cambios serán registrados por la cámara frontal del dron y serán analizados por el dispositivo móvil.

Puesto que se trata de un sensor basado en triangulación y la espalda es una superficie irregular, lo capturado por la cámara no tendrá la geometría de una línea recta. Por lo tanto, el cálculo de la orientación de la persona y la distancia que existe entre la persona y el dron no estará libre de errores.

1.3.2.4. Rendimiento del procesamiento de imágenes

Otro de los objetivos de esta tesis será lograr una velocidad de procesamiento de imágenes que permita rastrear al usuario en tiempo real, manteniéndose dentro del presupuesto energético de los dispositivos móviles y contemplando las limitaciones computacionales de la plataforma móvil.

En los dispositivos móviles *high-end* de la actualidad, los fabricantes han concentrado sus esfuerzos en el desarrollo de arquitecturas más rápidas y de bajo consumo de energía. El propósito original de estas nuevas arquitecturas es mejorar el desempeño de los dispositivos móviles en navegación web y videojuegos; en procesamiento de carácter más general también ofrecen mejores resultados como es el caso de este trabajo de tesis. Técnicamente, el consumo de energía de un CPU depende directamente de su frecuencia y voltaje de operación; para una misma demanda computacional, un dispositivo multi núcleo consume menos energía que sus similares de un sólo núcleo. Estas arquitecturas emplean tecnología SoC con cualidades de procesamiento simétrico, cuyas principales características son:

1. Contar con 2 o más núcleos de procesamiento.

2. Compartir la memoria disponible del sistema y ejecutar el mismo sistema operativo en todos los núcleos.

3. Dependiendo de la carga de trabajo, cada núcleo es capaz de operar de forma distinta para responder eficientemente y también compartir la carga con otros núcleos.

Así, en circunstancias de alta demanda de procesamiento, los distintos núcleos comparten la carga y no requieren operar a toda su capacidad para responder eficientemente. En consecuencia, consumen menos energía manteniendo un rendimiento superior o igual a sus similares de un sólo núcleo. La plataforma que empleamos en este trabajo de tesis la desarrolló Nvidia para dispositivos móviles, el Tegra 3 cuenta con 4 núcleos de procesamiento y ofreció un buen rendimiento en el procesamiento digital de imágenes empleado.

1.4 Organización por capítulos

En los siguientes capítulos de esta tesis, se describirá la metodología técnica y los resultados a los que se llegaron después de la implementación y prueba de la interfaz. La estructura de esta tesis es como sigue:

1. El Capítulo 2 describe los trabajos relacionados con la línea de investigación de esta tesis y el estado del arte.

2. Los fundamentos teóricos y técnicos de este trabajo de tesis se describen en el Capítulo 3.

3. La implementación de las técnicas listadas en el Capítulo 3 y los problemas resueltos durante su desarrollo se describe en el Capítulo 4. Se incluye la descripción detallada de los métodos de procesamiento digital de imágenes.

4. El Capítulo 5 describe los resultados a los que se llegaron después de implementar y probar el funcionamiento de la metodología propuesta.

5. Finalmente, el Capítulo 6 describe todas las conclusiones a las que se llegaron y describe el trabajo futuro.

1.5 Conclusiones y organización del documento

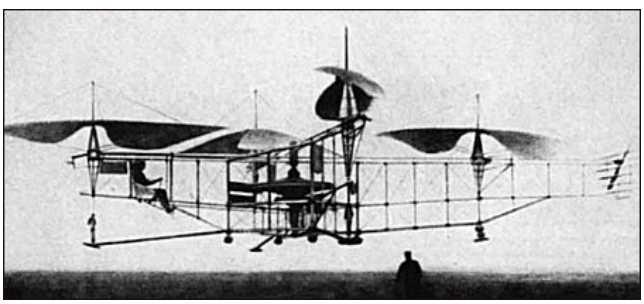
Esta sección dio una introducción al problema que se planteó al principio del desarrollo y la hipótesis que derivó de él. Complementariamente, describió la interfaz y los objetivos a alcanzar en su desarrollo.

2

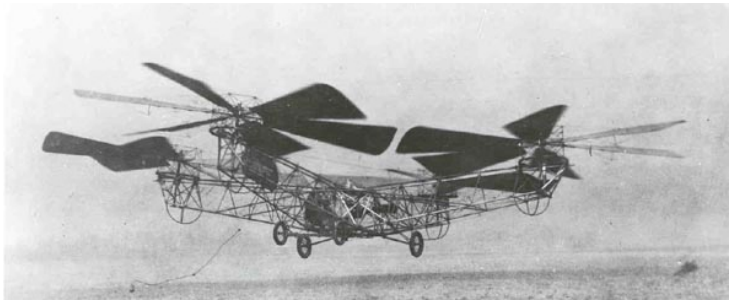
Trabajos relacionados

Un cuadricóptero es un vehículo aéreo propulsado por 4 rotores, capaz de despegar y aterrizar verticalmente. A diferencia de los helicópteros, las aspas de las hélices tienen un ángulo de ataque fijo y el movimiento del vehículo se logra variando la velocidad relativa entre los 4 rotores.

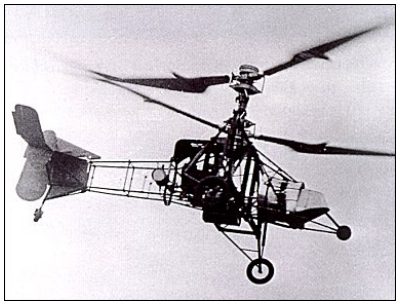
Los cuadricópteros actualmente se utilizan como vehículos no tripulados; han demostrado su utilidad en distintas aplicaciones, desde propósitos lúdicos hasta misiones de búsqueda y rescate. Se redujo su tamaño y peso significativamente, desde que el pionero francés de la aviación Etienne Oehmichen comprobara que era posible la construcción del helicóptero teórico, con la fabricación del cuadricóptero *Oehmichen No.2*, en 1922 (ver Figura 2.1a). Etienne Oehmichen experimentaba con alas rotatorias desde 1920, en su diseño los cuatro rotores del cuadricóptero eran propulsados por un solo motor y podían cambiar el ángulo de ataque; pero no eran lo suficientemente precisos para ser controlados independientemente, por lo cual usó otras 8 hélices para propulsión y estabilización. El *Oehmichen No.2* es considerado el primer cuadricóptero confiable capaz de realizar aterrizajes y despegues verticales. En 1923 era capaz de sostenerse inmóvil en el aire por varios minutos, estableció el primer récord para helicópteros al volar 360 metros y completó más de 1000 vuelos. El 4 de mayo de 1924 implantó un nuevo récord al volar 14 minutos en un recorrido circular de un



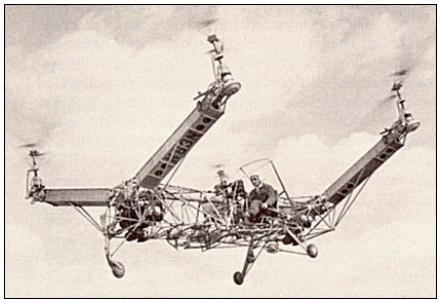
(a)



(b)



(c)



(d)

Figura 2.1: Primeras aeronaves con hélices (a) Oemichen No.2 (b) el cuadricóptero de George de Bothezat, (b) el *Gyroplane Laboratoire*, (c) Convertawings Modelo A.

kilómetro. En el continente de América, en enero del año 1922, la Armada de los Estados Unidos otorgó un contrato al equipo del Dr. George de Bothezat e Ivan Jerome para desarrollar un vehículo aéreo de propulsión vertical. Un año más tarde, en 1923, consiguieron elevar la aeronave de la Figura 2.1b de cuatro rotores y un peso de 1678 Kg, a una altura de 5 metros sobre el nivel del suelo. La nave transportaba a un pasajero, era mecánicamente compleja y susceptible a fallas.

Con el desarrollo de configuraciones *tandem* y del helicóptero coaxial (constituido de una ala rotatoria horizontal y una hélice en la parte posterior) los cuadricópteros no fueron del interés de los desarrolladores durante casi 30 años. El *gyroplane laboratoire*, construido en 1933 por el ingeniero francés Louis Breguet, es considerado el primer helicóptero coaxial práctico (ver Figura 2.1c). En esa época predominó en la aviación el desarrollo de helicópteros coaxiales y paralelamente, el desarrollo de aviones. No fue sino hasta mediados del siglo XX que el interés por los cuadricópteros resurgió con proyectos financiados por la Armada de los Estados Unidos, que buscaba el "jeep volador" como medio de transporte aéreo para tropas en zonas de guerra.

Figura 2.2: Cuadricóptero VZ-7

En 1956, el primero de los frutos de estos desarrollos fue el vuelo del cuadricóptero Convertawings Modelo A mostrado en la Figura 2.1d. Cada par de rotores está dispuesto en un arreglo *tandem* y unidos por un cuerpo tubular que porta un motor en cada extremo. Los motores están conectados al mecanismo que maneja los rotores mediante varias bandas, de tal forma que uno de los motores podría proveer la potencia para todos los rotores. El mecanismo de control es mucho más simple comparado con el de sus antecesores, basado en un dispositivo diferencial que equilibra los cambios en la fuerza de impulso de cada rotor.

El VZ-7 (mostrado en la Figura 2.2) es otro de los cuadricópteros desarrollados para la Armada de los Estados Unidos que mostró buena maniobrabilidad y facilidad de manejo. La compañía Curtis-Wright entregó dos de ellos a mediados de 1958 para realizar pruebas. El desempeño de la aeronave fue satisfactorio, pues se elevó a 60 metros sobre el nivel del suelo y se desplazó a 51 Km/h, pero no cumplió con los estándares de la armada y fue devuelta al fabricante en 1960. Ese mismo año, Curtis-Wright entregó otro prototipo de aeronave VTOL[1]: la X-19 contaba con dos alas dispuestas transversalmente al cuerpo de la aeronave, y en los extremos de cada ala se encontraba un rotor que podía girar 90°, permitiéndole a la aeronave despegar y aterrizar como un helicóptero. En noviembre de 1963 voló por primera vez y en agosto de 1965 se desplomó, lo que provocó la cancelación del proyecto.

Se pueden separar los diseños de los cuadricópteros en tres generaciones. Los vehículos presentados anteriormente pertenecen a la primera generación, ideada para transportar uno o más pasajeros;

[1]VTOL: *Vertical Take off and landing*.

Figura 2.3: E-volo, primer vehíclo aéreo VTOL eléctrico tripulado.

fueron los primeros que exitosamente lograron despegar y aterrizar verticalmente. Sin embargo, tenían un desempeño pobre y poca estabilidad, como se expuso con anterioridad. Los cuadricópteros de la segunda generación, que también se conocen como drones, son diseñados con un perfil de vehículos VTOL no tripulados. En esta generación, las baterías eléctricas son su principal fuente de alimentación, su vuelo es estable y cuentan con capacidades robustas de maniobra controladas desde algún dispositivo electrónico [13]. Los vehículos VTOL no tripulados, o drones, se abarcarán en la sección 2.1. La tercera generación y la más reciente, es la suma de las primeras dos, vehículos VTOL tripulados cuya fuente de poder es híbrida o puramente eléctrica.

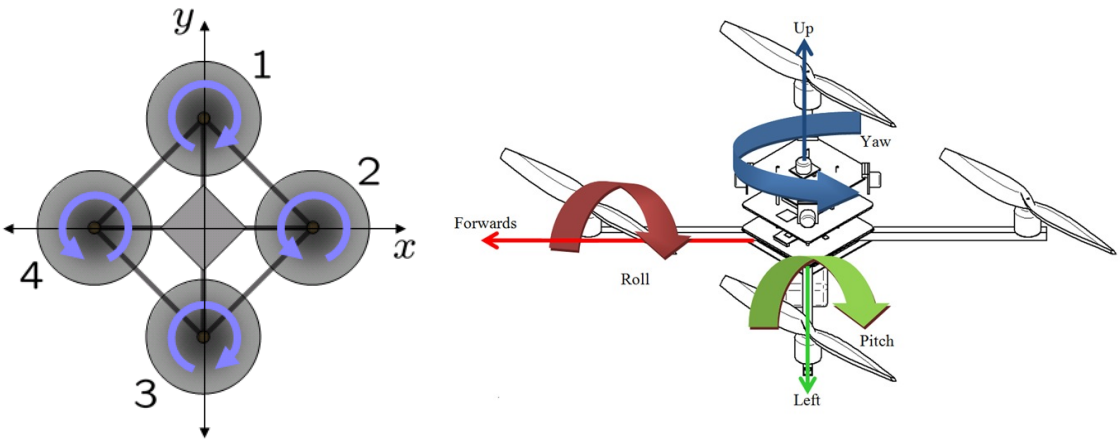
En lo que respecta a la tercera generación, a finales de octubre del 2011, un equipo alemán de tres ingenieros dirigido por Thomas Senkel, creó el primer vehículo VTOL eléctrico tripulado que denominaron *E-volo* (mostrado en la Figura 2.3). La disposición de los rotores es semejante al de un cuadricóptero actual, pero para poder proveer suficiente fuerza de levantamiento, cada rotor fue sustituido por 4 rotores, dando un total de 16 rotores en el cuerpo de la aeronave. Su vuelo duró un minuto 30 segundos y se elevó a poco más de 5 metros sobre el nivel del suelo. Manejar una aeronave en el aire de forma tan simple como un automóvil es, en opinión de los desarrolladores, el futuro en vuelos aéreos. Sus creadores desean comercializarlo y se encuentran desarrollando aspectos de seguridad, aprovechamiento de recursos y un modelo híbrido que use una fuente de poder con combustible fósil y otra eléctrica.

Figura 2.4: Dron AR, del fabricante Parrot.

2.1 Los cuadricópteros no tripulados (drones)

La ventaja de los cuadricópteros de segunda generación sobre los helicópteros coaxiales son dos. La primera, los cuadricópteros no requieren de uniones mecánicas para variar el ángulo de ataque de cada rotor. Se valen del control de velocidad de los rotores para desplazarse en el aire [30]. La segunda, comparados con un helicóptero, poseen menos energía cinética al desplazarse; esto se debe al menor tamaño en diámetro de sus cuatro rotores, comparado con el diámetro del rotor del helicóptero coaxial equivalente. Consecuentemente, se puede interactuar con ellos a una distancia menor [8], e incluso algunos drones comerciales usan su fuselaje para resguardar a los propulsores. Uno de ellos es el dron AR, del fabricante Parrot (ver Figura 2.4).

La Figura 2.5a muestra la disposición de los rotores de un cuadricóptero; el centro de gravedad (COG) de la aeronave se encuentra en el origen del sistema coordenado. El sentido de giro de los rotores está indicado con flechas circulares y los rotores están numerados del 1 al 4. Cada uno de los rotores del cuadricóptero genera una fuerza de empuje, un par de giro alrededor de su centro de rotación y una fuerza de arrastre contraria a la dirección de vuelo. Si todos los rotores giran con la misma velocidad angular, con rotores 1 y 3 girando en sentido horario y 2 y 4 en sentido anti-horario, la suma de pares de fuerzas alrededor del COG, es exactamente 0. Consecuentemente, la aeronave se encuentra flotando establemente en el aire. El giro alrededor del eje z, se logra disminuyendo la velocidad angular de alguno de los pares de motores (1 y 3 ó 2 y 4), esto causa un desequilibrio de los pares de fuerzas que actúan sobre el COG, haciendo girar la aeronave. Las variaciones de



(a) (b)

Figura 2.5: Cuadricópteros. (a) Disposición de los 4 rotores, (b) los 3 ángulos de orientación de la aeronave.

aceleraciones angulares en el eje de *pitch* y *roll* (ver Figura 2.5b) se logra aumentado y disminuyendo las velocidades angulares de cada par de motores [30]. El problema de la estabilidad y dinámica del cuadricóptero resulta complejo y para conocer las variables del sistema, es necesario el desarrollo de sensores inerciales, como giroscópios y acelerómetros, que permitan conocer el estado que guarda el cuadricóptero en todo momento.

2.2 IMU

Al principio de la década de los 70's ya existía la tecnología necesaria para obtener mediciones inerciales confiables usando acelerómetros y giroscopios dispuestos en una suspensión Cardan que se muestra en la Figura 2.6. Paralelamente, el desarrollo del giroscopio de anillo láser (RLG[2]) tomó 18 años, desde mediados de los 60's a principios de los 80's, pero no demostró superar en confiabilidad a sus contrapartes mecánicos. Algunos giroscopios contemporáneos a los RLGs, basados en un disco giratorio, tenían un tiempo promedio de uso entre fallas de decenas de miles de horas y sus mecanismos sufrían de un desgaste mínimo por su uso en ambientes aeronáuticos. Una IMU[3] basada en

[2]RLG: *Ring Laser Gyroscope*.
[3] Acrónimo en inglés: *Inertial Measurement Unit*, unidad de medición inercial.

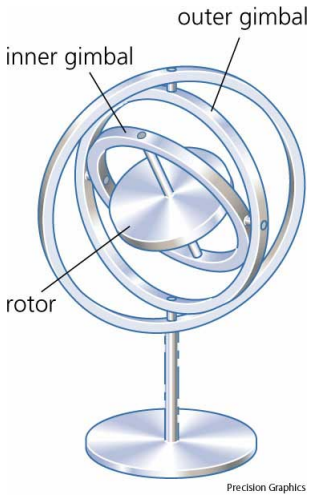


Figura 2.6: Suspensión Cardan.

RLG's y acelerómetros mide aproximadamente 178x178x279 milímetros, pesa alrededor de 10 Kg y requiere de una potencia de 50W [17]. A principios del siglo XXI, el desarrollo y venta al público de acelerómetros y giroscopios con tecnología MEM[4], hizo posible el desarrollo de pequeñas IMUs, útiles para estimar la orientación del vehículo aéreo en el espacio tridimensional. Dichas IMUs ya son incorporadas en las tarjetas de circuito impreso de los controladores a bordo de los vehículos aéreos, consumen menos de medio watt en operación y no ocupan una área mayor a 4 centímetros cuadrados. Este avance, entre otros, hizo posible que los cuadricópteros fueran más pequeños y estables, manteniendo bajos requerimientos computacionales para su control. La tecnología en microcontroladores de 8 bits ha alcanzado niveles de operación de hasta 1 MIPS por MHz y un consumo en potencia en el orden de miliWatts. Un microcontrolador AVR de 8 bits y una IMU de tecnología MEM son suficientes para operar el lazo de control de un cuadricóptero con una frecuencia de 1000Hz [13].

2.3 Telemetría y SLAM

En años recientes, la comunidad de investigadores en el área de robótica encontró en los cuadricópteros una nueva plataforma para resolver problemas de distintas áreas del conocimiento y migrar distintos problemas ya resueltos en plataformas terrestres. Algunos investigadores se formu-

[4]Acrónimo en inglés: *Micro Electrical Mechanisms*, dispositivos micro electromecánicos.

laron la pregunta de cómo equipar estos cuadricópteros con las habilidades requeridas para realizar vuelos autónomos. Dichas habilidades forman parte de un conjunto de estrategias para navegación autónoma conocidas como SLAM[5], que se han desarrollado y documentado con éxito para robots terrestres. En el SLAM, los robots construyen un mapa de un ambiente desconocido o actualizan el mapa de un ambiente conocido, mientras rastrean su ubicación en el mapa. La migración no es directa porque un robot terrestre es estable por desplazarse en un medio estable en esencia, lo cual no aplica a vehículos aéreos, donde comandos de estabilización deben ser enviados continuamente aunque el vehículo se encuentre en la posición deseada.

Controlar vehículos aéreos no tripulados es una tarea generalmente realizada desde una estación en tierra. Trabajos para realizar vuelos autónomos de UAV's se realizaron en años recientes [12, 37, 4, 5, 6]. En todos ellos, se emplean distintas estrategias y sensores para resolver un problema del tipo SLAM o de maniobras aéreas.

En estos trabajos, el cuadricóptero seleccionado para resolver este tipo de problemas cuenta con: unidad de cómputo capaz de ejecutar los algoritmos con restricciones de tiempo, sensores del tipo *laser range finder*, cámara monoscópica [12] o estereoscópica [4], IMU's y el conocimiento a priori del mapa. El cuadricóptero es capaz de generar un mapa local, estimar su ubicación en el mapa y generar una trayectoria para desplazarse en el ambiente. Una solución similar fue la estudiada en el Instituo Tecnológico de Massachusetts, con el uso de una cámara estereoscópica a bordo y de una estación en tierra para ejecutar los algoritmos de tipo SLAM [4].

Un aspecto que sobresale de los trabajos resumidos anteriormente lo constituye el sistema de telemetría láser, *laser range finder*. Son usados en la gran mayoría de las soluciones de tipo SLAM para cuadricópteros porque la percepción del entorno para un robot autónomo o semiautónomo es imprescindible. Los *laser range finders* estiman la distancia a un objeto midiendo el tiempo que le toma a la luz del diodo láser ser reflejada de vuelta [4, 33]. Estimar la distancia midiendo el tiempo de vuelo[6] de la luz, eleva el costo de la electrónica e imposibilita su uso para aplicaciones comerciales. Se han propuesto e implementado tecnologías de menor costo para estimar distancias a un objeto

[5] SLAM: *Simultaneous Localization And Mapping*.
[6] *Time-of-flight*.

con un láser y basadas en triangulación. Su éxito lo han alcanzado con características como: un costo de materiales menor a los 30 dólares estadounidenses, un rango de acción de 6 metros, 3 cms de precisión y un ancho de banda de 10 Hz [18].

En Japón se han diseñado, construído y calibrado *laser range finders* en el marco de un proyecto nacional para robots de búsqueda y rescate [20]. El sensor descrito es omnidireccional y puede capturar una fotografía al mismo tiempo que obtiene el mapa de distancias a los objetos que lo rodea. Cuenta con LED's que iluminan el área que desea retratar. El dispositivo se montó en el Souryu-IV, un robot inspirado en el movimiento de algunos reptiles con el propósito de construir mapas tridimensionales de lugares inaccesibles para el humano durante operaciones de rescate. El algoritmo genético que se usó para su calibración fue descrito posteriormente por los mismos autores [19] y desde las primeras etapas del diseño del *range finder*, el algoritmo de extracción de fondos se usó para segmentar la fotografía de la proyección del láser en algún objeto. Un *laser range finder* también se montó en un globo para escanear un monumento y poder preservar su modelo tridimensional en forma digital [16]. En ese trabajo se describe el método usado para fusionar la información capturada por el *range finder* y el movimiento del globo en el aire, el cual es susceptible al arrastre del viento, interpretado como ruido de baja frecuencia.

Las técnicas de fusión de datos provenientes de cámaras y sensores láser se analizaron y usaron en problemas SLAM tridimensionales. Por ejemplo, los datos proporcionados por una cámara estereoscópica y un *range finder* se pueden fusionar para que un dron navegue de forma autónoma en los interiores de un edificio [4]. De forma similar, la información del *range finder* puede ser usada para facilitar la extracción de características en un sistema de visión por computadora [33]. Una estrategia muy diferente a las mencionadas anteriormente, es obtener un modelo del entorno para generar una trayectoria. En este enfoque, el sistema de visión primero clasifica el entorno en el que se encuentra el UAV y con algoritmos de visión basados en perspectivas estiman la dirección deseada para volar [5].

Otros investigadores han resuelto problemas de aterrizaje y despegue automático sobre plataformas móviles o fijas. El problema se acrecienta al declarar la dinámica de la plataforma de aterrizaje como desconocida y/o provocada por eventos aleatorios, como es el caso de una plataforma flotan-

do sobre agua. En uno de estos trabajos se contempla un modelo dinámico del cuadricóptero para estimar su desplazamiento en el aire a partir de las cualidades inerciales del cuerpo de la aeronave. En la parte inferior del cuadricóptero se monta una cámara que apunta al suelo para capturar una imagen que es procesada por el algoritmo Lucas-Kanade para estimar, con flujo óptico, el desplazamiento de la plataforma para el aterrizaje [14]. En otro trabajo, se usa una cámara monocular y una homografía para estimar la pose del dron con respecto al área de aterrizaje; para posteriormente, con un filtro extendido de Kalman corrigir los errores ocasionados por la naturaleza de las mediciones. Para aterrizar, un controlador difuso analisa la información visual para generar comandos de altitud para el UAV [27].

Un conjunto de robots aéreos puede usarse para realizar un trabajo colectivo y crear un equipo de trabajo [22]. Pero su coordinación aún depende de un único algoritmo; las estrategias reportadas aún no involucran una interfaz del tipo propuesta en esta tesis. Similarmente, es posible usar un modelo reactivo-proactivo de control para realizar una tarea de colaboración entre un robot bípedo y un humano, permutando continuamente los roles de líder y subordinado entre los miembros del equipo [35]. La firma de arquitectos Gramazio & Kohler colaboró con el ingeniero Raffaello D'Andrea para crear la primera construcción multiagente de una maqueta arquitectural, a escala, de una ciudad vertical de 600 metros de alto[7]. En su página web describen la operación de varios cuadricópteros ensamblando una maqueta hecha de 1500 bloques prefabricados de poliestireno, que equivaldrían a bloques con 6 metros de alto y 3.5 metros de diámetro. La maqueta fue exhibida del 2 de diciembre 2011, al 19 de febrero 2012 en el centro Pompidou en París, Francia[11].

2.4 Interacción humano-robot

Una interfaz debe ser simple y eficiente, así facilitará que el usuario cumpla su objetivo. Regularmente se desarrollan tomando como modelo alguna analogía con la que el usuario está relacionado. Las mejores interfaces para la manipulación remota de aeronaves son de tipo hápticas[8] y se asemejan

[7] http://www.gramaziokohler.com/web/d/projekte/209.html

[8] En este contexto, una interfaz háptica es aquella que provee información al usuario no solo de manera visual y auditiva. Provee otra información sensorial como frío, calor, movimiento físico, etc.

a una cabina de avión. Cabe mencionar que todas las decisiones son tomadas por el piloto en tiempo de vuelo.

Hay un interés creciente en la investigación por la interacción humano-robot basado en un modelo sociológico. Para el caso de los vehículos aéreos no tripulados y de acuerdo a [29], se clasificaron los dispositivos de una interfaz, ideados originalmente para los videojuegos, en 5 grupos:

1. Dispositivos de salida: monitores, altavoces y dispositivos hápticos.

2. Métodos de representación de información: salida gráfica animada visual (gráficas generadas por computadora o provenientes de una cámara). Información adicional, mostrada en pantalla, relacionada con el estado de la aeronave (realidad aumentada). Por último, una salida no visual como audio.

3. Dispositivos de entrada de la interfaz: botones como teclados, dispositivos para señalamiento (punteros), *joysticks* y controladores especializados.

4. Métodos de entrada de la interfaz:

   - Comandos: una acción concreta del usuario genera un evento inmediato.
   - Lenguaje natural: consistente en una entrada del usuario mediante lenguaje oral o escrito.
   - Cursor: paradigma usado en interfaces humano máquina con dispositivos físicos (*mouse*).
   - Control de cámara: el usuario puede tener control total, parcial o nulo sobre la orientación de la cámara que le proporciona una vista parcial del ambiente.

5. Clasificación de entradas de la interfaz:

   - Simple: un botón presionado por el usuario genera un evento.
   - Contextual: una acción del usuario genera un evento sensible al contexto en el que se encuentre.
   - Combinacional: dos o más acciones simultáneas del usuario, generan un evento.

- Secuencial: dos o más eventos consecutivos, dentro de una ventana de tiempo, generan un evento.

La clasificación listada anteriormente fue desarrollada por la necesidad de seleccionar una interfaz apropiada para los usuarios no entrenados en el manejo de aeronaves no tripuladas, tomando como punto de partida el campo maduro de los videojuegos. El control de la aeronave se realiza desde una estación fija en tierra y reducir el tiempo de entrenamiento del personal es crucial [29]. De la clasificación, sobresale el apartado 4; en él se abarcan los métodos de entrada de la interfaz que pertenecen a la interacción natural.

En nuestra sociedad, es normal tener una comunicación directa con nuestro interlocutor si se encuentra a una distancia relativamente pequeña. Es raro que dos personas se comuniquen usando dispositivos electrónicos si se encuentran a una distancia menor a 3 metros una de la otra. Este trabajo de tesis plantea que una interfaz hombre-máquina, basada en modelos sociológicos, resulta ventajosa con respecto al paradigma clásico de control remoto.

En la Universidad de Texas A&M se montó la obra de teatro: *Un sueño de una noche de verano* [8]. En ella los personajes de la obra interactúan directamente con cuadricópteros y helicópteros que representan a hadas mágicas. No fue la intención original, pero la obra mostró ciertos aspectos imprevistos de la interacción hombre-máquina. Los actores, que no tenían conocimiento de la fragilidad de los helicópteros, los trataron con poco cuidado. Fue necesario que el encargado intercediera presentando a los helicópteros como "hadas bebes" que requieren cuidados; inmediatamente los actores cambiaron la actitud que tenían con los pequeños helicópteros. Incluso al desplomarse, los recogían con cuidado y al despegue tenían la precaución de orientarlos correctamente, acción que omitían cuando no se les había presentado a los helicópteros como hadas delicadas. Los helicópteros se desplomaban alrededor de 3 veces por presentación y la audiencia asimiló un rol similar al observar el patrón de comportamiento de los actores con los helicópteros; los trataban con la misma delicadeza, si se desplomaban cerca de ellos. En la misma obra, usaron un cuadricóptero de mayor tamaño y durante los ensayos, los actores despreciaron el posible daño que los rotores podrían infringirles, ya que el cuadricóptero no usa su fuselaje como una guarda para los rotores (ver Figura 2.7). Fue hasta

Figura 2.7: Dron AR-100, del fabricante Airobot. Usado en la obra de teatro de la universidad de Texas

que el encargado les mencionó a los actores su temido segundo nombre: "*La cortadora de césped voladora de la muerte*". Eso bastó para que los actores cambiaran su comportamiento y difundieran su pseudónimo en periódicos locales. Este ejercicio accidental de la Universidad de Texas prueba que el comportamiento de las personas que interactúan con cuadricópteros y que no han recibido entrenamiento previo puede tornarse peligroso para ambas partes.

En 2011, en la Universidad de Calgary, Canada, se planteó un problema de interacción hombre robot de distinta forma a la interacción a control remotro. En dicho esquema la cercanía entre ambas partes posibilita una interacción basada en gestos que el usuario realiza con sus brazos. Su motivación es la ausencia de trabajos relacionados con la interacción de robots voladores y humanos; además, la posibilidad de que los drones se conviertan algún día en un compañero habitual que espere instrucciones por parte del usuario requerirá el diseño de esquemas de interacción adecuados. El punto en el que se concentró la investigación fue el estudio de las reacciones y sensaciones que causa al usuario una interacción directa con un vehículo aéreo no tripulado. En el estudio utilizan el cuadricóptero AR y se infiere que usan el modelo de control a distancia. Ensayaron la estrategia de "El mago de Oz", una persona actuaba como "el mago" fuera de la vista del usuario, interpreta los gestos que la persona realizaba frente al dron y lo controla para que cumpla el comando. No cuenta con una manera automática de garantizar una distancia segura entre dron y usuario. Se usa una raqueta de badminton para mantener al dron a una distancia segura y como plataforma de despegue y aterrizaje. Se instruyó al usuario para que dirigiera comandos de control de vuelo y de

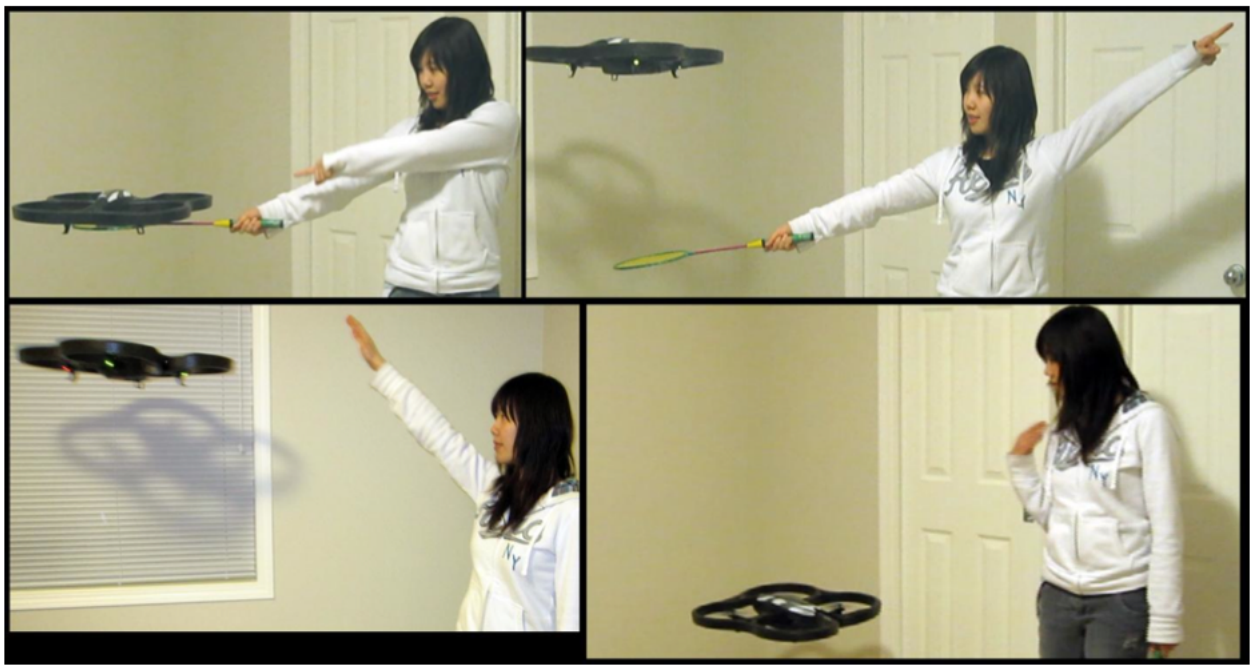
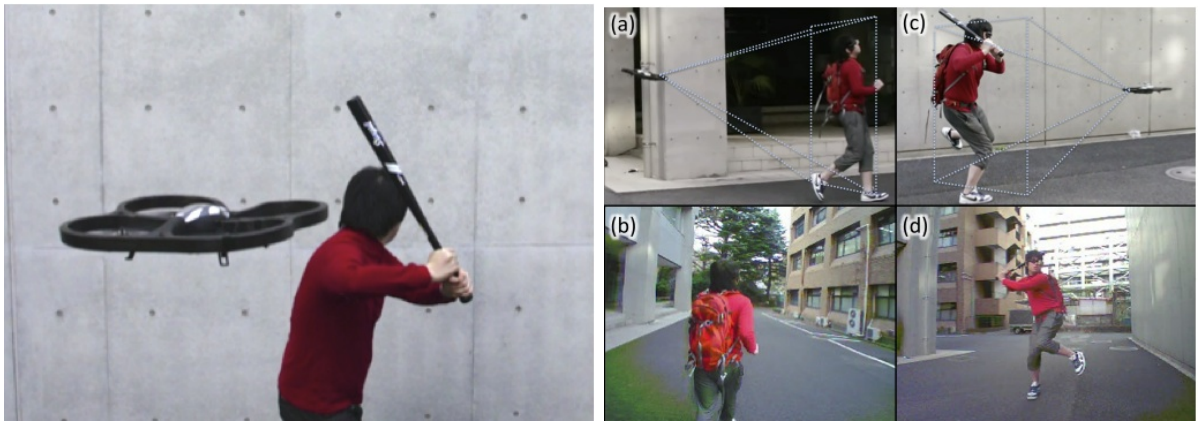


Figura 2.8: (Parte superior) El usuario realiza el gesto de despegue, (abajo a la izquierda) Gesto de elevación. (abajo a la derecha) Gesto de acercamiento.

interacción como: despegue, aterrizaje, acercamiento y giro (los gestos que la persona presenta al dron se muestran en la Figura 2.8). Concluyen que los comandos interpretados fueron aprendidos fácilmente por el usuario y que causaron un apego emocional del usuario hacia el dron, pues los usuarios trataban a los cuadricópteros como mascotas vivientes; la raqueta de badminton probó ser insuficiente como plataforma de aterrizaje y despegue [24].

De forma similar, en la Universidad de Tokio, usaron un cuadricóptero para seguir a un deportista y a través de la cámara que transportaba el dron, ofrecerle al deportista una vista en tercera persona (ver Figura 2.9a). Esta perspectiva le es útil al deportista para evaluar su estrategia y corregirla si así lo cree conveniente; él puede elegir con un dispositivo móvil, desde que punto de su vecindad quiere que sea retratado. El flujo de video se le presenta al deportista con un HMD[9]. Para seguir a la persona, usaron un filtro de partículas que rastrea el área roja de la chamarra que porta el usuario; la plataforma de procesamiento era una computadora personal o una laptop que el usuario transportaba en una mochila del mismo color (ver Figura 2.9b). Reportan que el deportista experimentaba desorientación

[9]HMD:*Head Mounted Display*.



(a) El dron retrata al deportista.

(b) El usuario lleva una mochila con la plataforma de procesamiento dentro y el dron logra rastrearlo.

Figura 2.9: Asistente para deportistas de la Universidad de Tokio

a causa del HMD y la perspectiva en tercera persona [15].

En el área de las ciencias computacionales, más específicamente en el desarrollo de interfaces hombre-máquina, se ha tratado de adoptar un modelo más natural para lograr interacción humano-máquina. Se han probado distintas estrategias, con resultados similares entre ellas. El modelo oculto de Markov (HMM[10]), aplicado a este tipo interfaces, tiene buenos resultados. Para describir la posición del usuario se usan las relaciones vectoriales entre una docena de puntos clave del cuerpo. El HMM se entrena con un algoritmo, como el Baum-Welch, para distinguir entre los gestos que no representaban información y 10 gestos más. La certidumbre de esta técnica es alta en reconocimiento de gestos si la calidad de la información capturada es buena, como la de la cámaras estereoscópicas del T-robot[11], con una velocidad de 60 cuadros por segundo [36].

En la universidad turca de Tuzla [21] estudiaron el mismo problema de reconocimiento de gestos con otro enfoque. El objetivo fue interactuar con un robot a través de gestos hechos con la mano y sus 5 dedos. Localizaban secciones potenciales de piel en una imagen con un modelo probabilístico cromático; después segmentaban la mano asumiendo que era el área más grande de la fotografía de pixeles, detectados como piel, que se encontraban 4-conectados. Una vez segmentada la mano, se

[10]HMM: *Hidden Markov Model*.
[11]Robot de servicio conocido como *Thinking Robot*.

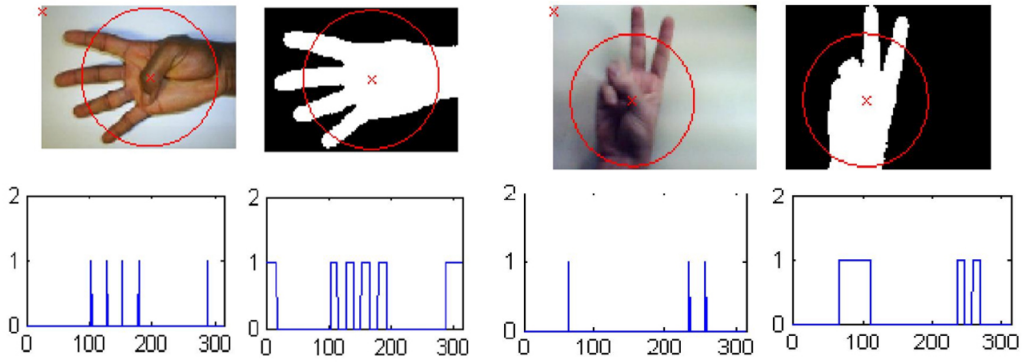


Figura 2.10: Resultado de la segmentación y generación de una función binaria para el reconocimiento de dos gestos [21].

localizaba su centro de gravedad (COG). Con el número de intersecciones del área segmentada de piel y un círculo, de radio 0.7 veces la distancia más grande del COG a uno de los dedos, generaban una señal binaria que indica el número de dedos usados en el gesto (la Figura 2.10 muestra la segmentación binaria y la señal binaria que resultaba de la superposición del círculo mencionado).

El estudio de gestos basado en acelerómetros ha proliferado gracias a que ya son usados en la mayoría de los dispositivos electrónicos comerciales y proveen una interfaz natural con el usuario. En esta área de estudio, se abordan dos escenarios distintos, uno dependiente del usuario y otro independiente de éste; generalmente los escenarios independientes del usuario reportan una menor certidumbre.

Diferentes estrategias de reconocimientos de gestos basadas en acelerómetros y DTW[12] han sido propuestas e implementadas con buenos resultados. El DTW es un algoritmo que mide la similitud entre dos secuencias de tiempo, de distinta duración. Puede ser usado en unión con otras técnicas [1] como:

1. Compresión temporal, algoritmo para disminuir el efecto del ruido en la medición del acelerómetro.

2. Propagación de afinidad, un algoritmo que considera a todos las muestras como datos ejemplares y recursivamente propaga mensajes verdaderos hasta que un clúster de datos válido

[12]DTW: *Dynamic Time Warping*.

emerge.

2.5 Dispositivos móviles

En los dispositivos móviles *high-end* actuales, los fabricantes han concentrado sus esfuerzos en el desarrollo de procesadores más rápidos y de bajo consumo de energía. El consumo de energía de un CPU depende directamente de la frecuencia y voltaje de operación del CPU. Por esta razón, fueron desarrollados algoritmos del tipo DVFS[13] que permiten aumentar la duración de la carga de la batería [10].

Actualmente, la tendencia de los fabricantes ha sido desarrollar tecnología SoC[14] con cualidades de procesamiento simétrico que cuentan con 2 o más núcleos de procesamiento (CPUs). Ahora, en circunstancias de alta demanda de procesamiento, los distintos CPUs comparten la carga y no requieren operar a toda su capacidad para responder eficientemente [25]. En consecuencia, consumen menor potencia.

Los dispositivos móviles han aumentado en capacidades de cómputo y algunos fabricantes los han dotado de una unidad de procesamiento de gráficos o GPU[15]. El objetivo principal de dichos GPUs es mejorar la experiencia de navegación en internet y de videojuegos para los usuarios, por lo cual poco trabajo se ha realizado en el procesamiento de propósito general usando GPUs de sistemas móviles.

En computadoras personales el GPU se ha convertido en un coprocesador de propósito general importante. Usan la arquitectura SIMD[16] y puesto que los algoritmos de visión realizan una misma operación en múltiples pixeles, es posible explotar su arquitectura para implementarlos eficientemente [9]. Sin embargo, estas aplicaciones han sido desarrolladas para plataformas en computadoras personales que soportan CUDA[17] y que actualmente no están disponibles en las más recientes generaciones de GPUs móviles [32]. No obstante se ha analizado, recientemente, el rendimiento de los

[13] *Dynamic Voltage and Frequency Scaling*.
[14] Circuitos integrados del tipo *System On Chip*.
[15] GPU: *Graphical Processing Unit*.
[16] SIMD: *Single Instruction Multiple Data*.
[17] CUDA: *Compute Unified Device Architecture*.

GPU's móviles con aplicaciones de reconocimiento de rostros [7], procesamiento digital de imagenes [32] y aplicaciones de realidad aumentada [34].

Para reconocer rostros, se representaron las cualidades del rostro con características basadas en filtros de Gabor [7]. Se han implementado algoritmos de ecualización del histograma [3]. Es posible implementar algoritmos de procesamientos de imágenes en un GPU portátil como: escalamiento de video, *rendering* con un estilo semejante al de la caricaturas y el algoritmo de detección de esquinas de Harris [32].

2.6 Conclusiones

Con base en la información presentada en este capítulo, se concluye que no hay un trabajo idéntico a lo que se pretende desarrollar en este trabajo de tesis. Los trabajos relacionados son bastantes y abarcan distintas áreas de conocimiento, pero usan metodologías diferentes. Uno de los estudios más similares, estudia la interacción humano robot con una estrategia que incluye a una segunda persona que aporta la personalidad del dron. La persona que personificaba al "mago" es quien interpretaba los comandos y manipula al dron de manera remota [24].

La telemetría láser es una área de desarrollo tecnológico madura y su utilidad se ha demostrado en aplicaciones de autonomía aérea. Sin embargo, se requiere de gran capacidad computacional para procesar toda la información que un sensor de este tipo provee. Por otro lado, la fusión de esta información con otra de una fuente distinta, amplifica la carga computacional de un desarrollo.

Los dispositivos móviles que cuentan con procesadores de doble núcleo y con un GPU, sólo se han utilizado masivamente en aplicaciones de videojuegos y navegación en internet. Se han realizado pocos estudios de su desempeño en aplicaciones menos específicas y paralelizables, como es el caso del procesamiento digital de imágenes.

3

Estrategia

Este capítulo describe el enfoque propuesto para abarcar los objetivos de este trabajo de tesis y las herramientas empleadas en hardware y software. Al respecto del hardware, se describe totalmente el proyecto, sus componentes y arquitectura; en cuanto a software, se describe la arquitectura de la aplicación desarrollada. La aplicación lleva como nombre *IHRVANT* por Interfaz Hombre-Robot con un Vehículo Aéreo No Tripulado. Además, el capítulo describe los escenarios de trabajo en los que la interfaz fue desarrollada y probada. Al final de este capítulo se encuentra el marco teórico del que se partió para desarrollar la estrategia planteada.

La Figura 3.1 contiene un diagrama a bloques de los elementos que componen la estrategia propuesta. Del lado izquierdo se encuentra el cuadricóptero y la tarjeta IOIO, que a través de una conexión Bluetooth, permite prender o apagar el láser de forma remota. Del lado derecho está el usuario que porta una chamarra de color y manga larga; transporta con él una *tablet* que ejecuta IHRVANT y cuenta con los periféricos necesarios para comunicarse con el dron (WiFi y Bluetooth).

El dron cuenta con una unidad de medidas inerciales para conocer su ubicación en el espacio, tal como una aeronave tripulada. Para medir la altitud de vuelo, el dron usa un sensor ultrasónico que apunta al suelo. Cuenta con dos cámaras: una frontal y una ventral; la frontal provee un flujo

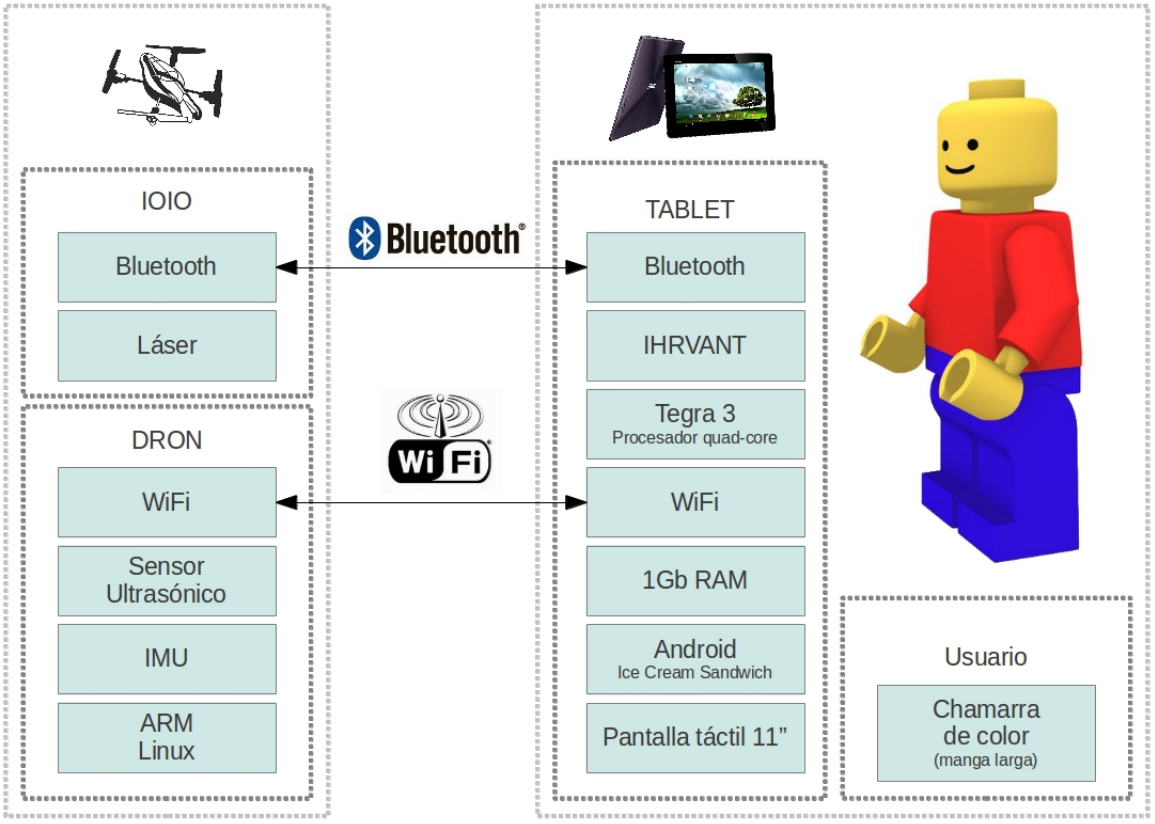


Figura 3.1: Esquema que describe las partes que integran la estrategia propuesta.

de video con resolución QVGA (320×240) y la ventral proporciona un flujo de video de tipo QCIF (176×144).

El dron encendido se convierte en un punto de acceso WiFi al que la tablet puede conectarse y, mediante DHCP, el dron le asigna una dirección IP. A través de esta conexión, la *tablet* puede solicitar al cuadricóptero el envío de toda la información de vuelo y el flujo de video. De forma similar, la *tablet* puede enviar comandos de control para despegar, manipular al dron durante el vuelo y aterrizar. La red WiFi, de acuerdo al fabricante, tiene un alcance de 50 metros línea de vista; el alcance de la red Bluetooth es de 10 metros, pero a una distancia mayor de 4 metros, la interfaz que propone esta tesis es impráctica y esta distancia no supera el alcance del Bluetooth. En conclusion, el usuario junto con la *tablet*, no pueden estar a una distancia mayor al alcance de la red Bluetooth (la de menor alcance) sin hacer impráctica la interfaz.

IHRVANT, a través de los periféricos de la *tablet*, hará que el cuadricóptero siga al usuario

mientras rastrea el color de su chamarra en el flujo de video frontal. Además, con la segmentación del color de la chamarra del usuario es posible distinguir su pose, que está relacionada con su movimiento al caminar o el evento de levantar sus brazos para dar un comando gestual. Estos elementos en conjunto y en un área de condiciones de iluminación controladas, harán que el cuadricóptero siga al usuario al mismo tiempo que clasifica la silueta de su chamarra.

3.1 Hardware

Esta sección describe los componentes de hardware que contempla la estrategia propuesta. Empieza por las características de la *tablet* y el dron; por último, el láser como sensor extereoceptivo.

3.1.1 ASUS Transformer Prime TF201

Para cumplir los objetivos de esta tesis, se escogió como plataforma de desarrollo la tableta de ASUS modelo TF201. Se trata de la primera tableta en incluir un Tegra 3 de Nvidia, un SoC de 5 núcleos ARM y un GPU GeForce de 12 núcleos. El quinto núcleo ARM solamente está disponible para el uso del sistema operativo en situaciones de poca demanda de procesamiento, el SO administra los recursos y acceso a los demás núcleos cuando una aplicación así lo demande. Los núcleos son de arquitectura ARMv7 con un coprocesador NEON, para realizar operaciones con el modelo SIMD. Cuenta con 1 Gb de memoria RAM y una pantalla táctil de 11".

La tableta ejecuta Android *Ice Cream Sandwich* y por lo tanto es compatible con todas las más recientes herramientas que ofrece Google y las de software libre actualizadas. Entre ellas se encuentran: Eclipse como ambiente de desarrollo, el ADK (*Android Development Kit*) y el NDK (*Native Development Kit*). Todas ellas están disponibles gratuitamente para los desarrolladores a través de internet. Eclipse y el ADK son las dos herramientas básicas y recomendadas para emprender un desarrollo en Android; el NDK contiene lo necesario para crear bibliotecas nativas en C.

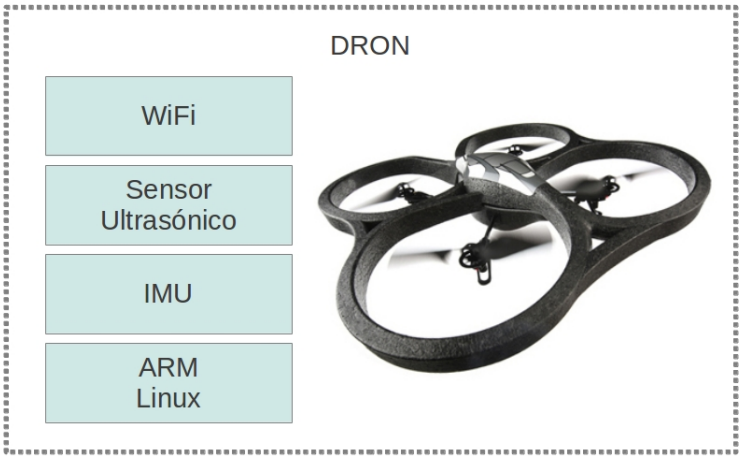


Figura 3.2: Diagrama del AR-Drone.

3.1.2 Parrot AR-Drone

El cuadricóptero que se usó fue el AR-Drone, del fabricante Parrot. Aunque tiene un propósito lúdico, ha sido bien recibido por la comunidad científica como plataforma de desarrollo para resolver otra gama de problemas que no están relacionados con la construcción del dron o su estabilidad de vuelo. Tiene dos tarjetas con circuitos electrónicos, una que incluye todos los sensores y la segunda con un procesador ARM ejecutando una versión reducida del kernel de Linux. Juntas, hacen posible que el cuadricóptero genere su propia red WiFi para enviar o recibir información entre el dron y un usuario adscrito a la red (el AR-Drone se muestra en la Figura 3.2).

3.1.2.1. Casos de uso

El paradigma de uso habitual comprende a un dispositivo móvil capaz de conectarse a una red WiFi y el cuadricóptero. La empresa Parrot, fabricante del cuadricóptero, distribuye la aplicación AR-Drone *Free Flight* como punto de partida, la cual ofrece al usuario la posibilidad de acceder a todos los recursos del dron y volarlo. Como es de suponer, una computadora puede remplazar al dispositivo móvil en el escenario anteriormente descrito. Esta opción predomina en trabajos de investigación con el AR-Drone, pues la computadora con todos sus recursos ejecuta algoritmos mucho más complejos.

3.1.2.2. Comunicación dron-dispositivo móvil

La comunicación se realiza a través de tres puertos UDP, cada uno con distintos propósitos. Desde el punto de vista del dispositivo adscrito a la red del cuadricóptero, los puertos son los siguientes:

1. Puerto UDP 5554: Recepción de los datos de navegación.

2. Puerto UDP 5555: Recepción del flujo de video.

3. Puerto UDP 5556: Envío de comandos de control al cuadricóptero.

La elección del protocolo UDP es habitual en este tipo de aplicaciones, pues no es necesario garantizar que no se pierda ningún paquete de información. En estas aplicaciones es mejor enviar el último y más reciente paquete de información a usar recursos para reenviar el último paquete que se perdió.

3.1.2.3. Cámaras a bordo

Como se mencionó anteriormente en este capítulo, el cuadricóptero cuenta con dos cámaras. La cámara frontal, de tecnología CMOS, tiene un lente gran angular (93°), su resolución es QVGA (320×240) y es capaz de enviar hasta 18 cuadros por segundo. La cámara ventral, de tecnología CMOS, tiene un lente gran angular (64°), su resolución es QCIF (176×144) y es capaz de enviar hasta 60 cuadros por segundo. La cámara frontal exhibe una aberración cromática en sus bordes; un tono verde cubre el contorno de cada cuadro del flujo de video. Este efecto es más evidente cuando la cámara captura objetos de color blanco. En el área central del cuadro de video, lugar en el que se mantiene al usuario al rastrearlo, la aberración cromática es menor y propició el éxito de la estrategia planteada.

3.1.2.4. Decodificador de video

El decoficador de video, empleado para reconstruir los cuadros del stream, usa un mecanismo de submuestreo en el espacio de color YC_bC_r. Dos codificaciones están disponibles: UVLC (semejante al

MPEG) y P264 (semejante al H264). Se usó el primer decodificador, por la disponibilidad de código fuente para su implementación en Java. Cuenta con las siguientes características:

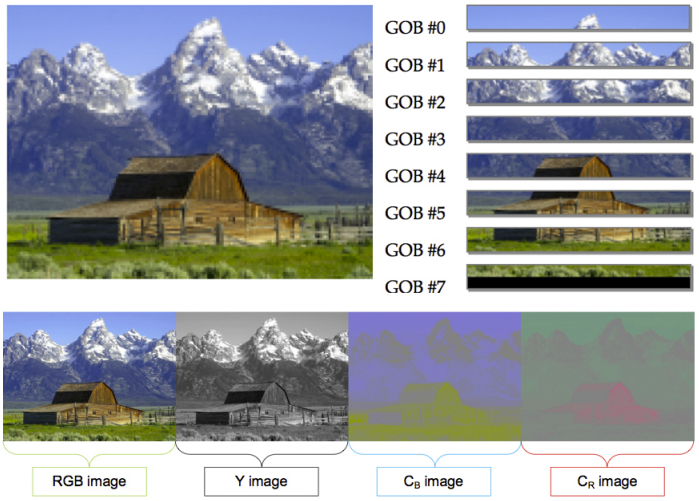
1. Espacio de color YC_bC_r.
2. Transformación DCT 8x8.
3. Codificación de entropía RLE+UVLC.

Cada cuadro de video es dividido en grupos de bloques (GOBs) que corresponden a 16 secciones horizontales de la imagen (ver Figura 3.3a). Cada GOB es dividido en 20 macro bloques que representan un área de 16x16 pixeles del cuadro original, en formato YC_bC_r, tipo 4:2:0 (ver Figura 3.3b). Finalmente, cada macro bloque se almacena en 6 bloques de 8x8 valores como se muestra en la Figura 3.3c que consisten en:

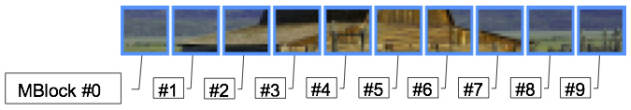
1. 4 bloques (Y_0, Y_1, Y_2, Y_3) para formar los 16x16 pixeles de la imagen Y del componente luminancia, la cual corresponde a la versión en tonos de grises de la imagen original en RGB.
2. 2 bloques (C_b, C_r) que contienen la información de ambos canales de crominancia calculados de la imagen original en RGB.

3.1.2.5. Control del cuadricóptero

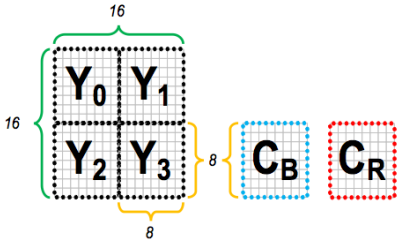
La Figura 3.4 ilustra las cuatro variables a manipular para controlar el vuelo del cuadricóptero. Los tres nombres de los ejes son también los nombres de los tres ángulos que se usan para describir la pose de una aeronave en el espacio. Si se modifica alguno de los tres ángulos *roll*, *pitch* o *yaw*, el cuadricóptero girará al rededor de sus ejes respectivos e implicará un movimiento en el aire. Por ejemplo, una modificación del ángulo *roll* hará que el cuadricóptero se desplace lateralmente a lo largo del eje *pitch*; una variación del ángulo *yaw* causará que el dron gire y cambie su orientación con respecto al norte gravitacional. La cuarta variable de control es el *gaz* y está relacionada con la altura de vuelo del cuadricóptero, una variación del *gaz* provocará que el cuadricóptero se eleve



(a) Codec



(b) Un GOB dividido en MacroBloques



(c) Estructura de un MacroBloque

Figura 3.3: Descripción de la estructura de una imagen.

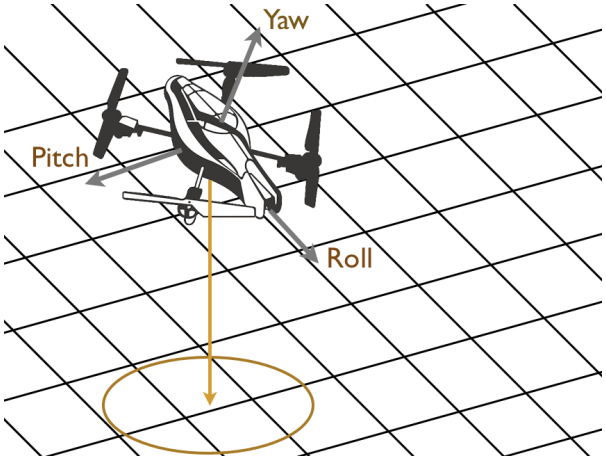


Figura 3.4: Direcciones de movimiento en el aire del AR-Drone.

o descienda; el sensor ultrasónico reporta la altura de vuelo constantemente y mediante el firmware del dron, la altura máxima de vuelo está limitada a 5m.

El control del cuadricóptero desde la tablet se realiza a través de comandos AT. Los comandos son cadenas de texto que se envían al dron para controlar sus acciones en paquetes UDP, puerto 5556. El fabricante recomienda enviar estos comandos cada 30ms para un vuelo suave y con un espaciamiento máximo de 2 segundos para que el dron no dé por perdida la conexión.

Las comandos AT se codifican en cadenas de caracteres ASCII de 8 bits, con un caracter *line feed* <LF> al final de la cadena ($10_{(10)}$). Cada comando comienza con tres caracteres: *AT** (*i.e.* tres palabras de 8 bits con valores: $41_{(16)}$,$54_{(16)}$, $2A_{(16)}$), le siguen el nombre de comando, un signo igual, un número de secuencia y opcionalmente, una lista de argumentos separados por comas cuyo significado depende del nombre del comando. Un sólo paquete UDP puede contener uno o más paquetes separados por caracteres *new line* ($0A_{(16)}$). No es posible dividir un comando AT en dos paquetes UDP y la longitud máxima de un comando no puede exceder 1024 paquetes, de otra forma, el dron no acepta el comando.

En la Tabla 3.1 hay un comando AT con nombre PCMD que el dron interpreta como un comando de movimiento. El número de secuencia es igual al número de secuencia del último comando enviado más 1. Específicamente, el comando PCMD acepta 5 argumentos, si el primero tiene un valor igual a cero, los siguientes 4 argumentos son interpretados como modificaciones solicitadas al *roll*, *pitch*, *yaw* y *gaz*; en ese orden y con un rango de valores válidos $[-1, 1]$. Si el primer argumento tiene un valor distinto a 0, el dron volará establemente sin desplazarce en alguna dirección. De forma similar, el comando REF puede hacer que el dron despegue o aterrice; ejemplo 3 y 4 de la Tabla 3.1, respectivamente. La guía de desarrollo que ofrece Parrot, junto al SDK, contiene una mejor descripción de todos los comandos y sus especificaciones.

3.1.2.6. Datos de navegación

Toda la información que generan los sensores (ángulos de orientación, altitud, velocidad y resultados de la realidad aumentada) a bordo del cuadricóptero es enviada a través de UDP, por el puerto 5554. La información está almacenada en formato binario y consiste de varios bloques que

Ejemplo	Nombre del Comando	# de Secuencia	Argumentos	*Line feed*
1	AT*PCMD=	21624	,0,0.1,0.1,0,0	<LF>
2	AT*PCMD=	21625	,1,0,0,0,0	<LF>
3	AT*REF=	21626	,290718208	<LF>
4	AT*REF=	21627	,290717696	<LF>

Tabla 3.1: Ejemplos de comandos AT.

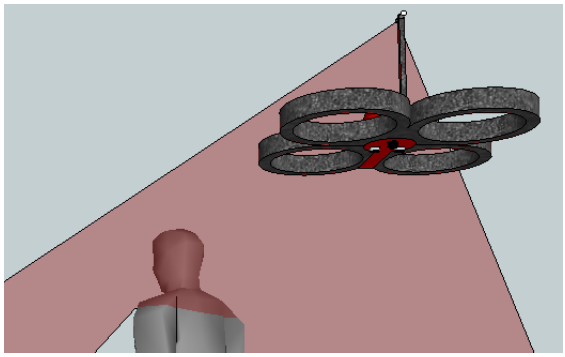


Figura 3.5: Proyección del láser sobre el usuario.

el fabricante denomina "opciones". Cada opción tiene un encabezado para su identificación y la información es almacenada en enteros de 32 bits en formato *little endian*. Incluye una sección *checksum* para que el cliente puede diagnosticar si el paquete es válido. IHRVANT interpreta el flujo de datos que arrivan al puerto 5554 para desplegar en la GUI, tales como: la carga de la batería, los ángulos de orientación y altitud. La guía de desarrollo que ofrece Parrot, junto al SDK, contiene las especificaciones técnicas del flujo de datos de navegación.

3.1.3 Láser

En el contexto de este trabajo de investigación, se desarrolló un sistema de triangulación de posición usando un plano láser. El objetivo del plano láser es estimar la distancia y la orientación de la persona relativa al cuadricóptero; el arreglo se puede describir como un sensor extereoceptivo basado en triangulación y procesamiento digital de imágenes. El plano láser se proyecta desde el cuadricóptero a 15.5 cms sobre el nivel de su cámara frontal, de tal forma que dibuje una línea horizontal en un objeto que se encuentra a 1.5 metros de distancia del dron y cuyo punto medio coincida con el centro del cuadro de la cámara frontal (la Figura 3.5 ilustra el escenario descrito).

El láser creará un cambio de iluminación en el objeto sobre el que se proyecte y por lo tanto, es posible segmentarlo parcialmente con alguno de los distintos métodos de detección de bordes, como Sobel o Canny. El caso de estudio tiene una particularidad importante: su objetivo es reconocer un cambio de iluminación vertical, pues se trata de una línea horizontal la que genera el plano láser. Por lo tanto, en caso de emplear Sobel, es posible usar solamente la máscara que detecta bordes en el eje vertical.

Es necesario remarcar que Sobel detecta un cambio en el gradiente de la imagen a partir de seis vecinos de cada pixel, lo que posibilita la detección de bordes no meramente verticales u horizontales. Por lo cual se decidió sólo tomar la información de la derivada discreta de cada una de las columnas verticales; esto se vuelve aún más coherente cuando se contempla la otra particularidad de este caso de estudio: su plataforma de procesamiento. Los recursos computacionales de la *tablet* no son equiparables a los de una computadora de escritorio y calcular solamente la derivada discreta vertical aligera la carga de trabajo.

Para segmentar aún mejor los pixeles detectados como bordes, escogeremos sólo los que se encuentren dentro de la silueta pictórica (usuario) que se obtiene de la segmentación del flujo de video, y que no esten cerca de su perímetro. Para medir la distancia de cada pixel detectado como borde al perímetro, usaremos el operado Chamfer. Si la distancia correspondiente a cada pixel detectado como borde sobrepasa un umbral definido previamente, será considerado un borde válido.

Desafortunadamente, no hay garantía de que todos los pixeles detectados como bordes con los criterios descritos anteriormente, son efectivamente bordes generados por la proyección láser. Por esta razón usaremos RANSAC para diferenciar entre *inliers* y *outliers*. Con RANSAC será posible encontrar un subconjunto de pixeles (*inliers*) que mejor ajuste a una línea recta, descartando otros pixeles (*outliers*) que podrían no pertenecer a los puntos generados por la proyección del láser. De RANSAC obtendremos los dos parámetros que definen una línea recta: su pendiente y su ordenada al origen; para que después, geométricamente, podamos estimar la orientación y distancia (relativa al dron) del objeto sobre el cual se proyectó el láser. En la Sección 3.3 se describen las herramientas teóricas de los pasos descritos.

3.2 Software

Como se mencionó, Parrot es el fabricante del AR-Drone y distribuye un SDK para crear aplicaciones para el SO Android, Linux, Windows y iOS. Esto representaba una ventaja al comenzar este trabajo de tesis, pues se pensaba que su puesta en marcha sería directa, sin contratiempos; desafortunadamente no fue así y se desechó la posibilidad de usar el SDK, así que se buscaron otras alternativas. Los problemas que se encontraron y como se resolvieron se comentan en las siguientes secciones de este capítulo.

3.2.1 SDK del fabricante Parrot

El SDK distribuido por Parrot sirve a los desarrolladores como base para sus aplicaciones específicas. Incluye un API que, funcionando correctamente, hace transparente el acceso a los puertos UDP y el API provee las funciones necesarias para controlar al drone, recibir datos de navegación y obtener el flujo de video. El SDK está escrito en lenguaje C, lo que posibilita su compilación para múltiples plataformas y reduce la cantidad de código específico para cada una.

Las plataformas que el fabricante reporta como soportadas son:

1. iOS, el sistema operativo de Apple para sus dispositivos móviles.

2. Cualquiera de las distribuciones Linux.

3. Windows, de Microsoft.

4. Andriod, de Google.

Se lograron compilar y ejecutar exitosamente las dos aplicaciones incluídas en el SDK para Linux. La primera aplicación lee comandos desde un *joystick* y muestra información enviada desde el dron en una instancia de la terminal. El código fue modificado para hacer uso del hardware disponible en el Cinvestav: un control de X-Box 360 y Linux Ubuntu v11.10.

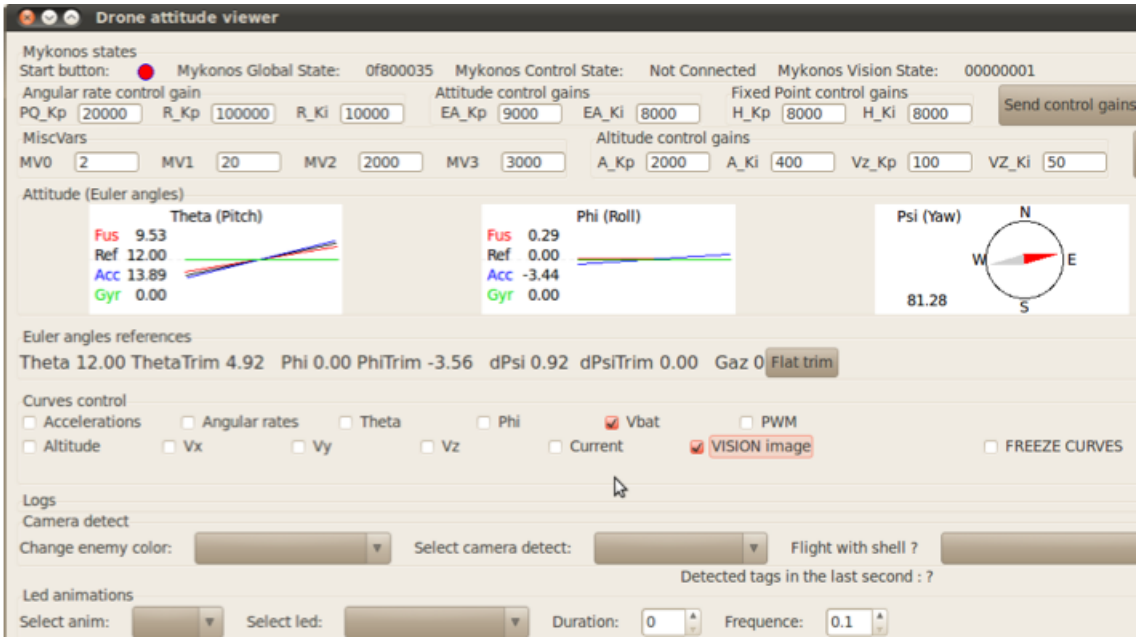


Figura 3.6: Attitude Viewer.

Primeramente, se reconocieron los valores únicos de identificación por dispositivo determinados en el protocolo USB. Cada dispositivo USB certificado, cuenta con dos valores que lo identifican, un *Vendor ID* y un *Device ID*. El *Vendor ID* es el que usa el fabricante para todos los dispositivos USB que fabrica (0x045E para Microsoft) y el *Device ID* es el identificador único que el fabricante le asigna a cada uno de sus productos con conectividad USB (0x0202 para el control de X-Box 360 que se usó). Siguiendo la guía de desarrollo de Parrot y modificando el código fuente, común entre ambas aplicaciones de Linux, se logró controlar el cuadricóptero desde una computadora ejecutando Linux y un *joystick*. En la Figura 3.6 se muestra la segunda aplicación, *Attitude Viewer*, puesta en marcha para Linux, la más completa de las dos, muestra gráficamente los valores enviados por el dron y es posible modificar algunos valores del sistema de control.

La aplicación demo, incluida en el SDK, usa un hilo de procesamiento por cada una de las principales actividades de la siguiente lista:

1. GUI.

2. Envío de de comandos de control al dron.

3. Recepción de datos de navegación.

4. Recepción y decodificación de video.

La Figura 3.7 describe la estructura de la aplicación demostrativa, incluida en el SDK. La aplicación, a través del SDK, crea 3 hilos para acceder a cada uno de los puertos UDP y las funciones de decodificación e interpretación las realiza el SDK, sólo para presentarlas a la aplicación a través de una función *callback*. La aplicación usa el acelerómetro del dispositivo móvil como entrada para enviarle comandos al dron en los ejes *roll* y *pitch*. El modelo de esta aplicación multi-hilo se tomó como referencia para construir la nueva solución, una vez que se decidió no usar el SDK.

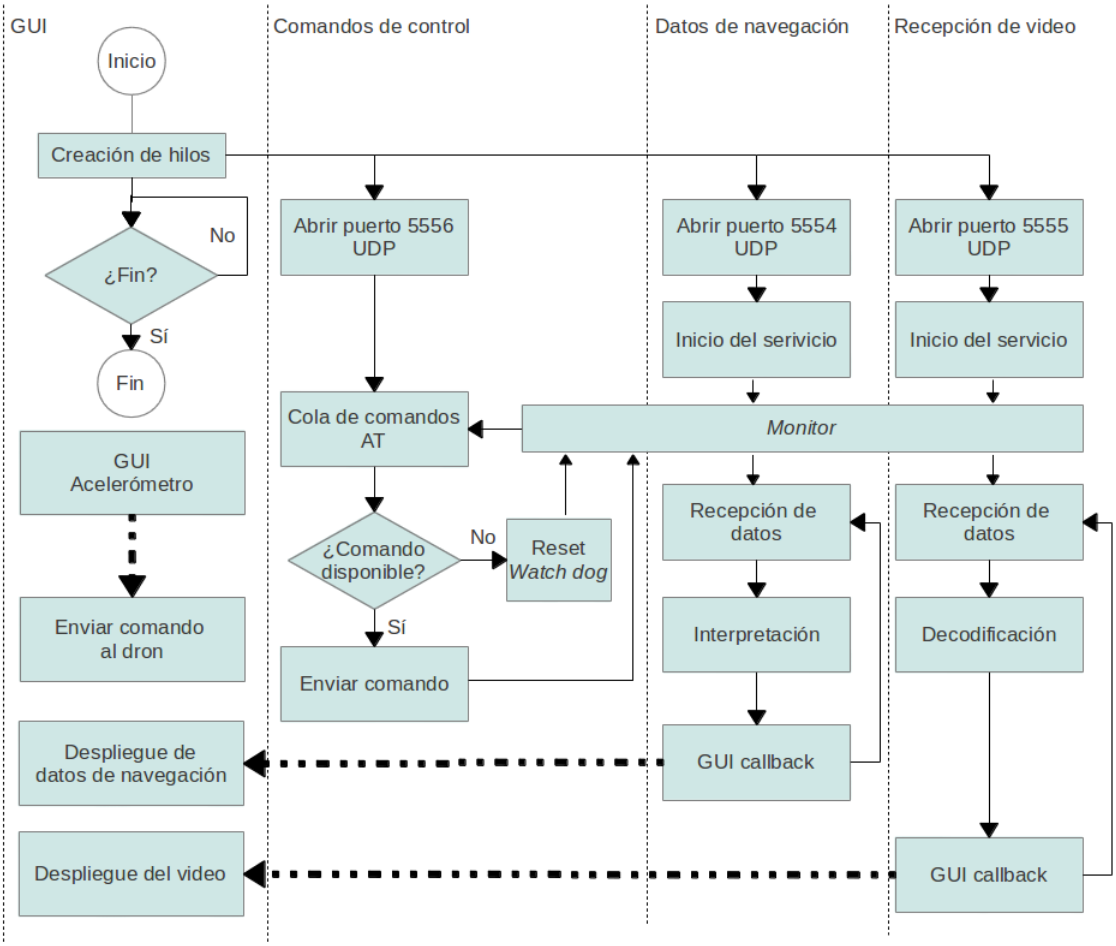


Figura 3.7: Diagrama de hilos de la aplicación incluída en el SDK.

3.2.2 JAVA AR-Drone

La empresa *CodeMinders* [1] desarrolló y distribuye bajo licencia GNU una aplicación escrita en el lenguaje Java para controlar al dron sin el SDK de Parrot.

La aplicación puede usarse como base para aplicaciones más específicas. Incluye las estructuras de datos y rutinas que hacen transparente el control del cuadricóptero, recepción de datos de navegación y despliegue del video. Está completamente escrita en JAVA, lo que hace posible su uso en múltiples plataformas. En la página de CodeMinders comparten vídeos donde su código, ejecutándose en una PC, controla al dron usando un joystick como dispositivo de entrada. Se trata, igualmente, de una aplicación multi hilo que atiende los tres puertos de comunicación UDP y la GUI; es una aplicación similar a la descrita en la Figura 3.7.

3.3 Marco teórico

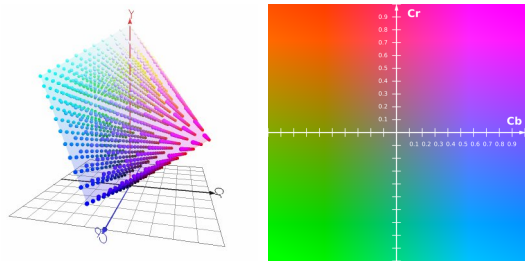
Hasta ahora, de la estrategia propuesta, sólo se han descrito los componentes y la forma de operar entre ellos; esta sección describe las herramientas teóricas en las que se basó este trabajo de tesis para segmentar la silueta del usuario y poder rastrearlo a través del flujo de video.

3.3.1 Segmentación por color

Existen distintas formas de segmentar el color, algunas son dependientes del espacio de color en la que se encuentre la imagen y varían en complejidad de operaciones. El primero consiste en crear un modelo probabilístico del color a segmentar y con dicho modelo, evaluar la probabilidad de que cada pixel sea parte del color de interés [21]. Este método requiere de un entrenamiento para definir el modelo probabilístico, lo cual no es práctico si se desea cambiar dinámicamente del color a segmentar.

El método que se describe a continuación, tiene dos característica beneficiosas: hace posible el cambio dinámico del color a segmentar y la similitud de cualquier pixel con el color a segmentar

[1] http://www.codeminders.com/



(a) Espacio Coordenado YC_bC_r (b) Plano C_bC_r

Figura 3.8: Espacio de color YC_bC_r

consiste en una distancia euclidiana. La operación es sencilla de implementar y más importante aún, la plataforma de procesamiento la puede realizar rápidamente.

Tomando en consideración la estructura del decodificador y de las imágenes, se decidió segmentar la información por color en el espacio YC_bC_r y no invertir tiempo de procesamiento en una conversión a otro espacio de color. Los espacios de color se diferencian por la estructura coordenada que emplean para representar la información cromática de un pixel, en el caso del espacio de color YC_bC_r, el eje coordenado z representa el componente luma (Y) y los ejes x, y representan los componentes C_b y C_r respectivamente, tal como se observa en la Figura 3.8a.

Para la segmentación del color de interés con coordenadas $(\varsigma_b, \varsigma_r)$ en el plano definido por los ejes C_b y C_r, se usa la distancia euclidiana cuadrada definida en la Ecuación 3.1. La distancia de cualquier punto dentro del planto C_bC_r con coordenadas (x, y) al color de interés $(\varsigma_b, \varsigma_r)$ se puede conocer con la Ecuación (3.1). Los tonos semejantes al color de interés caen dentro de una vecindad que es definida circular y de radio r. Por lo tanto, las distancias euclidianas cuadradas que sean menores a r^2 son considerados como segmentos en la imagen que son de interés.

$$D^2 = (x - \varsigma_b)^2 + (y - \varsigma_r)^2 \tag{3.1}$$

3.3.2 Filtro de partículas

Deseamos rastrear una región de interés en un cuadro de video, cuya silueta rectangular representa el dorso de una persona. La silueta pictórica de esta región no está definida y es variable en el tiempo; se requiere rastrearla en un flujo de video para estimar la ubicación del usuario respecto al dron. Existen distintas opciones para lograr este objetivo, todas basadas en distintas características como: color, bordes, flujo óptico y textura. Algunos de estos métodos, parten de la extracción de fondo; esto queda descartado, pues consideran que la cámara está fija, caso contrario a la cámara que lleva el cuadricóptero. Puesto que sólo deseamos estimar el estado de un sólo usuario, podemos elegir de entre el filtro de Kalman o el filtro de partículas. El primero se ha empleado extensivamente por la comunidad de visión para rastrear objetos o puntos de interés, pero tiene la limitante de asumir que las variables de estado tienen una distribución Gaussiana. Por lo tanto, el filtro de Kalman no puede estimar las variables de estado que no tienen una distribución paramétrica. Para compensar esto, usaremos un filtro de partículas con un modelo dinámico de segundo orden. El filtro de partículas puede estimar variables de estado que tienen una función de probabilidad no paramétrica [2].

3.3.2.1. Descripción

En estadística, un filtro de partículas es un método recursivo de Monte Carlo. Su objetivo es estimar recursivamente una secuencia de parámetros ocultos[2], a partir de observaciones previas de parámetros cuantificables de distinta naturaleza. Es una técnica sofisticada usada mayormente para estimar un modelo Bayesiano que cumple ciertas características como:

1. Las variables que definen su estado están conectadas por una cadena de Markov, pero el espacio de estados es continuo y no discreto como en un modelo oculto de Markov.

2. Dada la naturaleza del problema, el espacio de estados no está lo suficientemente restringido como para hacer posible una inferencia exacta. Por otro lado, en un *sistema dinámico lineal* el

[2]Por ocultos, entiéndase todos aquellos parámetros que no es posible cuantificar directamente con observaciones del sistema.

espacio de estados de las variables que lo definen está restringido a una distribución Gaussiana y por lo tanto, una inferencia exacta puede realizarse mediante el filtro de Kalman.

En este contexto, la operación del filtro de partículas es estimar recursivamente las funciones de distribución de probabilidad de las variables de estado en un tiempo específico, dadas todas o algunas observaciones anteriores de dichas variables. El filtro de partículas tiene este nombre por realizar la operación de filtrado usando un conjunto de "partículas" ponderadas, cuya población regularmente es de 1000 individuos.

El concepto de hipótesis *a priori* y *a posteriori* forma parte del marco teórico probabilista. La primera, denotada como $\bar{b}(x_t)$, representa una función de distribución de probabilidad que predice el estado x_t, de acuerdo a estados anteriores $x_{0:t-1}$. La segunda, denotada como $b(x_t)$, se calcula incorporando $\bar{b}(x_t)$ con observaciones del sistema $z_{1:t-1}$ y comandos de control ejecutados $u_{1:t-1}$.

3.3.2.2. Modelo dinámico

El orden del modelo para el filtro de partículas depende proporcionalmente del número de observaciones previas que se incluyen en la estimación de la observación *a priori*.

El filtro de partículas asume que el vector de estados x_t, observaciones previas $z_{1:t-1}$ y las acciones de control $u_{1:t-1}$, pueden ser modelados como una cadena de Markov. El cambio de estado, desde el inicial a los consecuentes $(x_0, x_1, x_2, ..., x_t)$, se estima a partir de una función de distribución de probabilidad que considera que x_{t-1} representa significativamente todas las mediciones y accciones de control previas, $z_{1:t-1}$ y $u_{1:t-1}$ respectivamente.

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{3.2}$$

Usar sólo x_{t-1} representa un modelo dinámico de primer orden, uno de segundo orden también contempla x_{t-2}. En nuestra implementación del filtro de partículas usamos uno de segundo orden que estima el estado x_t a partir de una función de distribución de probabilidad que contempla ambos

estados (Ecuación3.3).

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, x_{t-2}, u_t) \tag{3.3}$$

En la práctica, este modelo se expresa de manera matricial como la combinación lineal de los estados x_{t-1} y x_{t-2}, como se muestra en las Ecuaciones 3.4 y 3.5. Las matrices D, F y G definen al sistema dinámicamente y la matriz H es resultado de la caracterización del sensor empleado. Los sumandos w_t y v_t son valores aleatorios con distribución normal, media 0 y varianza 1. El factor σ es un valor que modica la desviación estándar de w_t y v_t; se ajusta a conveniencia.

$$x_t = D(x_{t-2}) + F(x_{t-1}) + G(u_t) + \sigma w_t \tag{3.4}$$

$$z_t = H(x_t) + \sigma v_t \tag{3.5}$$

3.3.2.3. Algoritmo del filtro de partículas

El algoritmo genérico del filtro de partículas, descrito en el Algoritmo 1, usa un conjunto de M hipótesis (las partículas) para estimar la función de distribución de probabilidad *a posteriori* $b(x_t)$. Recibe como entrada M partículas, la acción de control u_t y una medición z_t.

Primeramente estima $\bar{b}(x_t)$, dicho de otra forma, calcula un conjunto de hipóteis *a priori*, con un modelo de la forma descrita por la Ecuación 3.3. Este modelo predice la función de distribución de probabilidad $\bar{b}(x_t)$. La representación es aproximada, pero no es paramétrica y por lo tanto puede representar un mayor rango de distribuciones, no solo guassianas. La función de distribución de probabilidad $b(x_t)$ que incluye la hipótesis *a priori* y los valores u_t y z_t, representa el valor devuelto por el algoritmo.

En este filtro, las muestras de la distribución *a posteriori* son llamadas partículas y denotadas como: $X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$. Cada partícula $x_t^{[m]}(1 \leq m \leq M)$ es un instancia del estado en el tiempo t, o sea, una hipótesis del estado en el mundo real en el tiempo t. Aquí M representa el número de partículas y generalmente es un número grande, alrededor de 1000. En algunas implementaciones

M depende de t o de otras cantidades relacionadas con $b(x_t)$.

La intuición del algoritmo radica en aproximar $b(x_t)$ con el conjunto de partículas X_t a través de un proceso de muestreo ponderado. Idealmente, la probabilidad de incluir la hipótesis $x_t^{[m]}$ en el conjunto X_t debe ser proporcional a una función de distribución de probabilidad, como la de la Ecuación 3.6.

$$x_t^m \sim p(x_t|u_t, x_{t-1}^{[m]}) \tag{3.6}$$

Como sugiere la Ecuación 3.6, entre más densamente poblada una subregión de la función de distribución de probabilidad con muestras esté, mayor será la probabilidad de que el verdadero estado esté dentro de dicha subregión. La variante más básica del filtro de partículas se describe en el Algoritmo 1.

Algoritmo 1 Filtro de partículas (X_{t-1}, u_t, z_t)

1: $\bar{X}_t \Leftarrow 0$
2: $X_t \Leftarrow 0$
3: **for** $m = 1 \rightarrow M$ **do**
4: muestrear $x_t^m \sim p(x_t|u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} \Leftarrow p(z_t|x_t^{[m]})$
6: $\bar{X}_t \Leftarrow \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: **for** $m = 1 \rightarrow M$ **do**
8: tomar i con probabilidad $\propto w_t^{[i]}$
9: agregar $x_t^{[i]}$ a X_t
10: **return** X_t

El algoritmo toma como entrada un conjunto de partículas X_{t-1} de tamaño M, un comando de control u_t y una medición z_t. En el primer bucle, línea 4, para cada una de las partículas se calcula su estado hipotético *a priori* usando una función de distribución de probabilidad que describe el cambio de estado. En la línea 5 se calcula un factor de importancia de cada partícula muestreando de la función de probabilidad de la medición z_t dada por $w_t^{[m]} = p(z_t|x_t^{[m]})$. El paso más importante y valioso del filtro de partículas ocurre en la línea 8, donde un nuevo conjunto de partículas X_t del mismo tamaño se crea muestreando con remplazo del conjunto temporal de partículas $\bar{X}_t$. La probabilidad del muestreo de cada partícula está dada por el valor de su factor de importancia. Como

el remuestreo se realiza con remplazo, el conjunto de partículas resultado puede poseer muchas partículas duplicadas. Más importante aún resulta el hecho de que las partículas ausentes en X_t y presentes en $\bar{X}_t$, son las que obtuvieron un factor de importancia menor. Este algoritmo tiene como complejidad $O(2M)$.

3.3.3 Reconocimiento de gestos

Una de las primeras estrategias propuestas para clasificar la postura del usuario e identificarla como un gesto, es el uso de características que contribuyan por igual a una función de discriminación, y por tanto estén dentro de un mismo orden de magnitud.

En este trabajo de tesis se emplearon dos técnicas para la clasificación de la silueta pictórica del usuario que trabajan con imágenes binarias. La primera usa los siete momentos definidos por Hu, mientras que la segunda emplea los histogramas horizontal y vertical de la silueta para clasificarla. La cualidad más importante de los momentos de Hu es que son invariantes a la escala y a la rotación; mientras que los histogramas son susceptibles a ambas, por lo cual se empleó un método de acondicionamiento previo que será descrito en las secciones siguientes.

3.3.3.1. Clasificador por mínima distancia

La región segmentada puede ser clasificada usando los 7 momentos de Hu (1962) que son invariantes a la traslación, la rotación y el cambio de escala. Para definir los momentos de Hu, es necesario primero definir el momento de orden $(p + q)$ para una región (Ecuación 3.7).

$$m_{pq} = \sum_{x} \sum_{y} x^p y^q f(x, y) \tag{3.7}$$

Donde $f(x, y)$ es la intensidad del punto (x, y) y la suma se toma sobre todas las coordenadas espaciales (x, y) de puntos de la región. El *momento central* de orden $(p + q)$ viene dado por la Ecuación 3.8 y las coordenadas del centroide de una región por las Ecuaciones 3.9.

$$\mu_{pq} = \sum_{x}\sum_{y}(x-\bar{x})^p(y-\bar{y})^q f(x,y) \tag{3.8}$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad\qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{3.9}$$

Los momentos *centrales normalizados* de orden $(p+q)$ se definen como

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \qquad donde \qquad \gamma = \frac{p+q}{2}+1 \qquad\qquad para(p+q) = 2,3,... \tag{3.10}$$

El siguiente conjunto de *momentos invariantes* propuesto por Hu, se puede obtener usando únicamente los momentos centrales normalizados de órdenes 2 y 3.

$$\phi_{_1} = \eta_{_{20}} + \eta_{_{02}} \tag{3.11}$$

$$\phi_{_2} = \left(\eta_{_{20}} - \eta_{_{02}}\right)^2 + 4\eta_{_{11}}^2 \tag{3.12}$$

$$\phi_{_3} = \left(\eta_{_{30}} - 3\eta_{_{12}}\right)^2 + \left(3\eta_{_{21}} - \eta_{_{03}}\right)^2 \tag{3.13}$$

$$\phi_{_4} = \left(\eta_{_{30}} + \eta_{_{12}}\right)^2 + \left(\eta_{_{21}} + \eta_{_{03}}\right)^2 \tag{3.14}$$

$$\begin{aligned}\phi_{_5} = {} & \left(\eta_{_{30}} - 3\eta_{_{12}}\right)\left(\eta_{_{30}} + \eta_{_{12}}\right)\left[\left(\eta_{_{30}} + \eta_{_{12}}\right)^2 - 3\left(\eta_{_{21}} + \eta_{_{03}}\right)^2\right] + \\ & \left(3\eta_{_{21}} - \eta_{_{03}}\right)\left(\eta_{_{21}} + \eta_{_{03}}\right)\left[3\left(\eta_{_{30}} + \eta_{_{12}}\right)^2 - \left(\eta_{_{21}} + \eta_{_{03}}\right)^2\right]\end{aligned} \tag{3.15}$$

$$\begin{aligned}\phi_{_6} = {} & \left(\eta_{_{20}} - \eta_{_{02}}\right)\left[\left(\eta_{_{30}} + \eta_{_{12}}\right)^2 - \left(\eta_{_{21}} + \eta_{_{03}}\right)^2\right] + \\ & 4\eta_{_{11}}\left(\eta_{_{30}} + \eta_{_{12}}\right)\left(\eta_{_{21}} + \eta_{_{03}}\right)\end{aligned} \tag{3.16}$$

$$\begin{aligned}\phi_{_7} = {} & \left(3\eta_{_{21}} - \eta_{_{03}}\right)\left(\eta_{_{30}} + \eta_{_{12}}\right)\left[\left(\eta_{_{30}} + \eta_{_{12}}\right)^2 - 3\left(\eta_{_{21}} + \eta_{_{03}}\right)^2\right] + \\ & \left(3\eta_{_{12}} - \eta_{_{03}}\right)\left(\eta_{_{21}} + \eta_{_{03}}\right)\left[3\left(\eta_{_{30}} + \eta_{_{12}}\right)^2 - \left(\eta_{_{21}} + \eta_{_{03}}\right)^2\right]\end{aligned} \tag{3.17}$$

Para que todos los momentos contribuyan por igual en la función de discriminación fue necesaria una normalización (Ecuación 3.18). De tal forma, para una silueta pictórica que represente algún gesto y para cualquier otra silueta, definimos un vector de características $\alpha^* = \left(\phi_{_1}, \phi_{_2}, \phi_{_3}, \phi_{_4}, \phi_{_5}, \phi_{_6}, \phi_{_7}\right)$.

En el caso particular de este trabajo de tesis, definimos la galería de gestos como el conjunto de n vectores, donde cada elemento representa un gesto del usuario $\lambda = (\alpha_1^*, \alpha_2^*, ..., \alpha_n^*)$. Para detectar un gesto, evaluamos la similitud entre α y todos los vectores que constituyen a λ con la distancia euclidiana para espacios vectoriales de 7 dimensiones (Ecuación 3.19). La menor distancia d_i indicará cuál es el gesto que el usuario presentó a la cámara del dron.

$$\phi_n = |\ln(|\phi_n|)| \tag{3.18}$$

$$d_i = \sqrt{\sum_{n=1}^{7} (\alpha_n^* - \alpha_n)^2} \tag{3.19}$$

3.3.3.2. Clasificación por histogramas horizontal y vertical

Los histogramas horizontal y vertical de la imagen binaria estabilizada que contiene la silueta pictórica se pueden usar para clasificar la imagen completa. En el siguiente capítulo se describirá como la aplicación IHRVANT estabiliza el cuadro de la silueta pictórica para su clasificación. Por otro lado, la similitud de cada uno de los histogramas normalizados la determina la Ecuación 3.20. Donde N representa el número de *buckets* con el que fue calculado el histograma y h_i^* representa el *i-ésimo bucket* del histograma de referencia, de forma similar h_i representa el valor del *i-ésimo bucket* del histograma calculado a partir del cuadro estabilizado actual.

$$d_i = \sum_{n=1}^{N} (h_i^* - h_i)^2 \tag{3.20}$$

3.3.4 Análisis de los clasificadores de gestos

Ambos clasificadores serán evaluados de acuerdo al número de verdaderos-positivos (VP), verdaderos-negativos (VN), falsos-positivos (FP) y falsos-negativos (FN) que detecten. Dichas cantidades defi-

nirán la sensibilidad, especificidad y precisión del clasificador (Ecuaciones 3.21, 3.22, 3.23).

$$\textit{Sensibilidad} = \frac{VP}{VP + FN} \tag{3.21}$$

$$\textit{Especificidad} = \frac{VN}{VN + FP} \tag{3.22}$$

$$\textit{Precisión} = \frac{VP}{VP + FP} \tag{3.23}$$

3.3.5 Transformada de distancia

La transformada de distancia proporciona una medición de la separación existente entre dos puntos dentro de una imagen. Dados dos pixeles $p(x, y)$ y $q(s, t)$; se puede definir una función de distancia D si se cumple: $D(p, q) \geq 0$, $D(p, q) = 0$ si $p = q$ y $D(p, q) = D(q, p)$. En la Figura 3.2a, el punto D se encuentra justo en el centro y la distancia euclidiana se calcula de D a todos los demás pixeles.

Las funciones de distancia comúnmente usadas son: distancia euclidiana y distancia de tablero de ajedrez. La distancia euclidiana entre p y q está dada por $D_E(p, q) = \sqrt{(x - s)^2 + (y - t)^2}$. La distancia de tablero de ajedrez: en donde se observa que los 4-vecinos están a una distancia unitaria del píxel central; si se desea que los 8-vecinos estén a la misma distancia se toma: $D(p, q) = Max(x - s, y - t)$. En la Figura 3.2b, el punto D se encuentra justo en el centro y la distancia de ajedrez se calcula de D a todos los demás pixeles.

$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
$\sqrt{5}$	$\sqrt{2}$	2	$\sqrt{2}$	$\sqrt{5}$
2	1	0	1	2
$\sqrt{5}$	$\sqrt{2}$	2	$\sqrt{2}$	$\sqrt{5}$
$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$

(a) Distancia Euclidiana.

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

(b) Distancia tablero de ajedrez.

8	7	6	7	8
7	4	3	4	7
6	3	0	3	6
7	4	3	4	7
8	7	6	7	8

(c) Chamfer.

4	3	4
3	0	3
4	3	4

(d) Máscara de chamfer

Tabla 3.2: Cálculo de distintas distancias para una misma imagen de 5×5.

También existe la distancia Chamfer, en la Figura 3.2c se muestra un mapa de distancias $D(x, y)$ de 5×5 pixeles que fueron calculados con la máscara de la Tabla 3.2d. Al principio todos los pixeles tienen un valor de ∞, exceptuando los pixeles a partir de los cuales se desea medir una distancia, esos tienen el valor de 0. La máscara se sobrepone en cada elemento del mapa y sólo se modificara su valor si la máscara coincide con por lo menos, un pixel distinto a ∞. Se asigna a $D(x, y)$ el mínimo de las sumas de cada elemento de la máscara con el pixel que sobrepone. Esta operación se repite hasta que no haya elementos con valores iguales a ∞.

3.3.6 Detección de bordes y operadores de primera derivada

Un borde, en procesamiento digital de imágenes, se define como una diferencia significativa entre la tonalidad de cada pixel y sus 8 vecinos. La Figura 3.9 representa el mapa de bits original, el recuadro rojo enmarca una línea de exploración vertical y a su costado derecho se encuentra el perfil de intensidad a lo largo de dicha línea. De manera ilustrativa, la función de los cambios en tonalidades $f(x)$ se muestra con una línea roja, a su derecha su derivada $f'(x)$ y por último su valor absoluto $|f'(x)|$. Los bordes (transición de oscuro a claro o viceversa) se modelan como una rampa en lugar de hacerlo como un cambio brusco de intensidad. La primera derivada es cero en todas las regiones de intensidad constante y tiene un valor constante en todas las transiciones de intensidad. Por tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada y presenta un paso por cero, es decir se produce un cambio de signo en su valor.

El gradiente de una imagen $f(x, y)$ en un punto (x, y) se define como un vector bidimensional dado por la Ecuación 3.24, siendo un vector perpendicular al borde donde le vector $\mathbf{G}$ apunta en la dirección de variación máxima de f en el punto (x, y) por unidad de distancia con la magnitud y dirección dadas por las Ecuaciones 3.25. Es una práctica habitual aproximar la magnitud del gradiente con valores absolutos (Ecuación 3.26). Esto se hace porque el valor de la magnitud del gradiente no es tan importante como la relación entre diferentes valores. Es decir, se va a decidir si un punto es de borde según que la magnitud del gradiente supere o no un determinado umbral, pues bien sólo es cuestión de ajustar dicho umbral para que el resultado de la extracción de bordes sea el mismo tanto

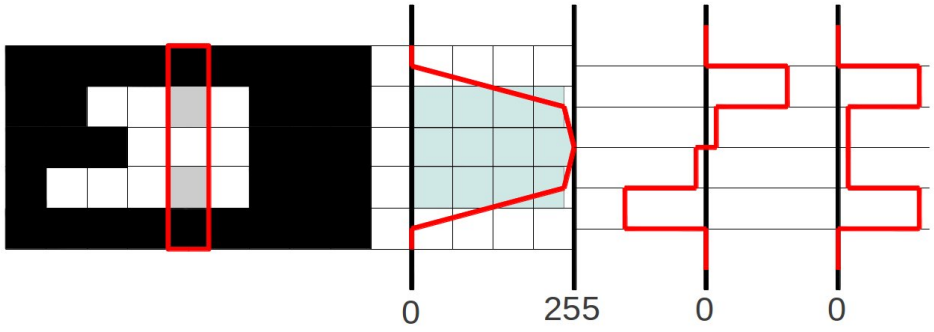


Figura 3.9: Mapa de bits I mostrando una función $f(x)$ de intensidad a lo largo de una línea de exploración vertical, su primera derivada $f'(x)$ y valor absoluto $|f'(x)$.

si se calcula la magnitud del gradiente (Ecuación 3.25) como si se hace mediante la aproximación por valores absolutos (Ecuación 3.26).

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta}{\delta x} f(x,y) \\ \frac{\delta}{\delta y} f(x,y) \end{bmatrix} \tag{3.24}$$

$$|\mathbf{G}| = \sqrt{G_x^2 + g_y^2} \qquad \phi(x,y) = \tan^{-1} \frac{G_y}{G_x} \tag{3.25}$$

$$|\mathbf{G}| \approx |G_x| + |G_y| \tag{3.26}$$

Se puede aproximar la derivada por medio de las diferencias de primer orden entre dos pixeles adyacentes (Ecuación 3.27). Esta es la forma más elemental de obtener el gradiente en un punto. Para obtener una imagen binaria puede usarse la Ecuación 3.28 donde T es un valor de umbral no negativo. Sólo los pixeles de borde cuyo gradiente excedan del valor T se consideran importantes. Así la ecuación se puede ver como un procedimiento que extrae sólo aquellos pixeles caracterizados por transiciones de intensidad significativas (dependiendo de T).

$$G_x = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \qquad G_y = \frac{f(y+\Delta y) - f(y-\Delta y)}{2\Delta y} \tag{3.27}$$

$$g(x,y) = \begin{cases} 1 & \text{si } G[f(x,y)] > T \\ 0 & \text{si } G[f(x,y)] \leq T \end{cases} \tag{3.28}$$

3.3.7 RANSAC

El algoritmo RANSAC (*RANdom SAmple Consensus*) servirá para ajustar una línea al conjunto de coordenadas cartesianas $\mathbf{P_N}$. El algoritmo evalúa K consensos, creando una línea recta entre dos puntos pertenecientes a $\mathbf{P_N}$ y midiendo la distancia ortogonal de todos los N puntos a dicha línea. Clasifica a un subconjunto de M puntos $\mathbf{I_M} \subseteq \mathbf{P_N}$ como *inliers* si la distancia ortogonal es menor a un umbral d_t, los demás puntos se consideran *outliers*. El mejor consenso de los K evaluados, será el conjunto de puntos que maximice M y los dos parámetros que definen la recta a partir de $\mathbf{I_M}$, podrán conocerse con las ecuaciones de ajuste de rectas por mínimos cuadrados.

El Algoritmo 2 describe las operaciones de RANSAC para ajustar una línea recta, toma como entrada un conjunto de puntos $\mathbf{P_N}$, el número de consensos a evaluar K, la distancia umbral d_t y una razón $r \in (0,1)$ que definirá el número mínimo de *inliers* necesarios para evaluar como válido el *k-ésimo* consenso.

Al principio RANSAC inicializa 2 variables: la variable max y el número mínimo de *inliers* necesarios para considerar válido un consenso a partir de r y N. Por cada iteración del ciclo *for* (línea 3) RANSAC toma dos puntos aleatoriamente de $\mathbf{P_N}$ (línea 5) y mide la distancia ortogonal d_i (línea 10) de todos los puntos a la recta definida por dichos puntos. Si la distancia d_i es menor a la distancia de umbralado d_t, el *i-ésimo* punto formará parte del conjunto $\mathbf{I}$ (línea 12). El tamaño de $\mathbf{I}$, denotado por M, deberá ser mayor a I_t para considerar que se trata de un consenso válido que contiene un número mínimo de *inliers*; si lo es, con las ecuaciones de mínimos cuadrados se conocerán los parámetros m y b que definen la recta que mejor ajustan al conjunto de puntos de $\mathbf{I}$ (líneas **??** y 19). Por último, se calculan los dos puntos $\mathbf{p_1}$ y $\mathbf{p_2}$ a partir de las abscisas menor y mayor con la ecuación de la recta

$y = mx + b$ (líneas 21 y 20).

Hipotéticamente, los puntos $\mathbf{p_1}$ y $\mathbf{p_2}$ representan los extremos de la recta que mejor ajusta a la proyección del láser sobre la silueta pictórica del usuario. Sólo resta explicar el modelo geométrico con el que se extraerá información a partir de la recta estimada con la metodología descrita anteriormente. La orientación y la distancia a la que se encuentra el usuario del dron serán las dos variables a estimar con dicho modelo geométrico. Este análisis será el objeto de un párrafo del siguiente capítulo.

Algoritmo 2 RANSAC $(\mathbf{P_N}, K, d_t, r)$

1: $max \Leftarrow 0$
2: $I_t \Leftarrow \lfloor r \times N \rfloor$
3: **for** $m = 1 \rightarrow K$ **do**
4: $\quad \mathbf{I} \Leftarrow \emptyset$
5: $\quad (\mathbf{p_1}, \mathbf{p_2}) \Leftarrow$ muestreo aleatorio$(\mathbf{P_N})$
6: $\quad \mathbf{w} \Leftarrow \mathbf{p_2} - \mathbf{p_1}$
7: $\quad \mathbf{D} \Leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|}$
8: $\quad \eta \Leftarrow -\mathbf{D}^{-1}$
9: $\quad$ **for** $i = 1 \rightarrow N$ **do**
10: $\quad\quad d_i \Leftarrow \eta \times (\mathbf{P_i} - \mathbf{p_1})$
11: $\quad\quad$ **if** $d_i < d_t$ **then**
12: $\quad\quad\quad \mathbf{I} \Leftarrow \mathbf{I} \cup \mathbf{p_i}$
13: $\quad M \Leftarrow$ número de puntos en $\mathbf{I}$
14: $\quad$ **if** $M < I_t$ **then**
15: $\quad\quad$ **continue**
16: $\quad$ **if** $M > max$ **then**
17: $\quad\quad max \Leftarrow M$
18: $\quad\quad m \Leftarrow \frac{M \sum \mathbf{I_x I_y} - \sum \mathbf{I_x} \sum \mathbf{I_y}}{M \sum \mathbf{I_x^2} - \sum \mathbf{I_x}^2}$
19: $\quad\quad b \Leftarrow \frac{\sum \mathbf{I_y} \sum \mathbf{I_x^2} - \sum \mathbf{I_x} \sum \mathbf{I_x I_y}}{M \sum \mathbf{I_x^2} - \sum \mathbf{I_x}^2}$
20: $\mathbf{p_1} \Leftarrow (min(\mathbf{I_x}), m \times min(\mathbf{I_x}) + b)$
21: $\mathbf{p_2} \Leftarrow (max(\mathbf{I_x}), m \times max(\mathbf{I_x}) + b)$
22: **return** $(\mathbf{p_1}, \mathbf{p_2})$

3.3.8 Control de vuelo

El procesamiento de imágenes aportará información para estimar la posición relativa del usuario con respecto al dron. Las variables medibles de la silueta pictórica que servirán para estimar la posición de usuario, que es una parámetro *oculto* en el proceso de observación, serán:

- El área de la silueta pictórica. Intuitivamente, a mayor área de la silueta, menor será la distancia del dron al usuario y viceversa.
- El centroide de la silueta pictórica. Para mantener al usuario dentro del cuadro del video, la magnitud y orientación de un vector polar trazado desde el centro del cuadro al centroide, indicará la acción correctiva necesaria para ubicar al usuario al centro del cuadro de video.
- Las coordenadas de las fronteras del rectángulo envolvente y la proporción entre alto y ancho será el primer indicador para estimar la pose del dorso del usuario. Si el usuario levanta las manos para realizar un gesto, las proporciones de la caja envolvente se verán alteradas.

Por otra parte, el dron cuenta con sensores a bordo que le indican su altura a nivel del suelo y su pose en el aire (*pitch, yaw y roll*). Estos datos son compartidos a través de un puerto UDP y junto a los parámetros mencionados anteriormente, hará posible la creación de un control difuso, en la tablet, que enviará de forma automática comandos de vuelo al dron.

Un comando de control para el dron, lo constituye 4 variables en el rango $[-1, 1]$. Los primeros tres hacen variar el *pitch, yaw* y *roll* del cuadricóptero; el cuarto hace que se eleve o descenda. Por lo tanto, la salida del controlador a implementar debe tener esos mismos 4 parámetros de salida y los tres parámetros listados en la Sección 3.3.8.

Por tratarse de un controlador multi-entrada y multi-salida, se propone un controlador difuso, que en base a la entrada y reglas de inferencia, decida la salida requerida para los cuatro parámetros de control del dron.

3.4 Comprobación de resultados

La primera prueba de cualquier interfaz es que sea práctica y la interfaz que se propone en esta tesis no debe ser la excepción.

En [15], los usuarios interactúan con un cuadricóptero, pero no describen como logran mantener una distancia apropiada entre el usuario y el dron. En otro trabajo usan una raqueta para alejar al cuadricóptero y mantenerlo a una distancia segura [24]. Por otro lado, en [8], se describe el riesgo

potencial de la interacción con un dron de 4 aspas. En el caso del AR-Drone se puede usar el fuselaje pensado para interiores, para proteger al usuario. Falta mencionar que basado en observaciones, es necesario que el cuadricóptero se mantenga a una distancia mínima de 1 metro del usuario para que la toma del video alcance a abarcar el dorso y brazos del usuario. Si la interfaz logra mantener esta distancia, será funcional.

El tiempo útil de la carga de la batería a bordo del cuadricóptero es de 10 minutos de vuelo. Si la interfaz logra seguir al usuario durante esos diez minutos de vuelo, manteniéndolo dentro del cuadro de video, se considerará funcional.

En [26], reportan una precisión superior al 79 % en la detección de 4 gestos y solo rastreando la parte superior del cuerpo, la máxima precisión reportada es de 98 %. Pero a diferencia de este trabajo de tesis, usan una cámara fija para obtener una fotografía del fondo y con una substracción con la imagen del usuario obtienen la silueta pictórica. Se considerará práctica la interfaz si se reconoce en un 80 % los gestos del usuario en condiciones controladas de iluminación.

3.5 Listado de actividades

En esta sección se ordenan los pasos a seguir para resolver el problema propuesto en este protocolo.

1. Investigación de trabajos previos referentes a esta tesis y planteamiento del problema a resolver.

2. Redacción y revisión del protocolo de tesis, donde se establece el marco teórico y una perspectiva completa del trabajo a desarrollar.

3. Redacción y revisión del documento que describa el estado del arte del área de estudio de esta tesis.

4. Modificar el dron para montar el proyector de plano láser en la parte alta del fuselaje.

5. Desarrollar la comunicación WiFi entre el dron y el dispositivo móvil. Se usará el SDK del fabricante Parrot y su aplicación muestra para la plataforma Android como punto de partida.

6. Desarrollar la primera parte de la aplicación Android. Será capaz de reconocer el evento de *llamada de atención*. En ella se desplegará la imagen capturada por la cámara a bordo del dron. Se realizará la calibración de la cámara y implementarán los medios para guardar las fotografías en un medio no volátil.

7. Se calibrará el sensor extereoceptivo con el plano láser para obtener mediciones fiables. Esta primera calibración sólo verificará la distancia a la que se encuentran los objetos.

8. Desarrollar un algoritmo para obtener estimaciones fiables de la orientación del objeto sobre el cual se proyecta el plano láser.

9. Diseñar un esquema de control adecuado para poder dirigir el dron en la dirección deseada con la velocidad deseada. Se basará en la información proveída por el sensor láser y su objetivo será seguir al usuario adecuadamente y de manera autónoma.

10. Entrenar el clasificador que será usado para distinguir entre los distintos gestos que el usuario realice con su mando frente al dron.

11. Implementar un algoritmo para segmentar la mano de la fotografía. Usará el marcador proyectado en la pantalla del dispositivo móvil para estimar el área en la fotografía donde se encuentra la mano del usuario y por ende, la instrucción.

12. Una serie de pruebas y mejoras de todo el sistema, al término de todas las etapas de desarrollo.

13. En todas las fases se obtendrán datos e información valiosa que debe ser analizada para incluirla en el documento de tesis que será redactada a la par de los avances proyectados.

14. La tesis será revisada en dos ocasiones por el profesorado del CINVESTAV para realizar correcciones de cualquier tipo.

15. Por último, se presentará un examen de grado, donde se defenderá el trabajo de tesis realizado.

3.6 Conclusiones

En este capítulo se describieron la herramientas técnicas propuestas para alcanzar los objetivos de este trabajo de tesis y que prueban su factibilidad.

Las implementaciones de las estrategias aquí planteadas deben tomar en cuenta las limitaciones del cómputo móvil, pues la capacidad de procesamiento de la tablet impactará directamente en el rendimiento de la aplicación. Hay que considerar las restricciones de tiempo, la usabilidad de la interfaz y obtener provecho de la programación multi hilos. La programación multi-hilos con una sincronización adecuada y aprovechamiento de recursos disponibles permitirá que la interfaz responda en un tiempo aceptable.

El dron usa un algoritmo de compresión con pérdidas para enviar el flujo de video y el bitmap en el espacio YC_bC_r, decodificado en la tablet, es la información inmediata al *codec* después de ser transmitido *over-the-air*. Uno de los principales cimientos que propiciará el buen rendimiento de la aplicación es el acceso directo al *codec* de video. En la Sección 3.3.1 se hizo mención de la técnica de codificación y el uso del espacio de color YC_bC_r, resulta beneficioso adaptar las técnicas descritas en este capítulo a dicho espacio de color y evitar el *overhead* de convertir el mapa de bits que se muestra en la pantalla en RGB a cualquier otro espacio de color (como el HSV). Además, con el acceso al codec, también se obtiene una edición submuestreada de los canales C_bC_r, lo cual reduce el tamaño de la imagen a procesar, de QVGA (320×240) a una cuarta parte.

El filtro de partículas es el algoritmo que toma más tiempo producir un resultado. Su utilidad ha sido comprobada [28] en el espacio HSV y siguiendo regiones de proporciones fijas, aunque de distinta escala.

El sensor láser puede ofrecer una manera complementaria al segmentado con el filtro de partículas para estimar la distancia existente entre el dron y el usuario. Resta comprobar su exactitud e implementación en la aplicación IHRVANT.

Su implementación y resultados se mostrarán en capítulos posteriores.

4

Desarrollo e Implementación

Este capítulo describe la implementación de la estrategia propuesta en el capítulo anterior y presenta algunos resultados preliminares. La plataforma de procesamiento (mencionada en la Sección 3.1.1) consiste en una *tablet* con Android como sistema operativo, por lo cual la aplicación IHRVANT fue desarrollada mayoritariamente en el lenguaje de programación JAVA. Pese a esto, el procesamiento digital de imágenes se implementó en C, mediante JNI[1] y el NDK[2] provisto por Google. Este capítulo describe como la metodología y la arquitectura del decodificador de video definieron la arquitectura de IHRVANT.

4.1 Puesta en marcha del SDK y Java AR-Drone

La Sección 3.2.1 describe el primer acercamiento con el SDK de Parrot, pese a seguir las instrucciones descritas en la guía del desarrollador que se ofrece junto con el SDK, su compilación no fue directa. Afortunadamente, los problemas ya estaban registrados y solucionados en un foro de

[1] JNI: *Java Native Interface*.
[2] NDK: *Native Development Kit*.

internet por la comunidad de desarrolladores[3].

Construidas y probadas las aplicaciones para Linux descritas en la Sección 3.2.1, que demostraban el acceso a todas las funciones del dron, se procedió a construir la aplicación para Android. Los problemas técnicos enfrentados para usar el SDK en Android fueron demasiados y, desafortunadamente, no se logró con éxito compilar esta aplicación, pese a solicitar soporte a través del foro, correo electrónico y vía telefónica con la empresa Parrot, en Francia. Por políticas de la empresa, no fue posible hablar con ninguno de los ingenieros de software de Parrot involucrados en el SDK.

El problema principal estaba relacionado con el hilo de recepción de datos de navegación. Por razones desconocidas, no lograba establecer comunicación con el dron y esto era considerado un error que hacía culminar la aplicación de forma abrupta. Al parecer, un *timer* culminaba su cuenta antes de que se recibiera algún dato de navegación.

Por otro lado, se probó el proyecto Java AR-Drone. Dicho código corrió en una aplicación para el SO Android con poco éxito. Usando el código de *CodeMinders*, el cuadricóptero al despegar daba media vuelta en el aire para chocar contra el suelo. La recepción del video era deficiente, en el mejor de los casos, sólo un cuadro se presentaba en la interfaz; la recepción de datos de navegación probó ser funcional al mostrarlos en la GUI de manera fluida.

Probadas ambas alternativas con poco éxito, se optó por construir una aplicación reutilizando código del proyecto Java AR-Drone y dejar en el olvido al SDK de Parrot. La arquitectura base es idéntica a la mostrada en la Figura 3.7 y representa los cimientos de IHRVANT. Primero, se construyó una interfaz con el usuario para desplegar toda la información necesaria: datos de navegación, video y controles de entrada (*joysticks*). El hilo de comandos de control se construyó desde cero para evitar que el cuadricóptero vuelva a dar una vuelta en el aire y se cause algún daño. El segundo hilo, el receptor de datos de navegación fue tomado directamente de Java AR-Drone. El tercer hilo está dividido en dos partes: el receptor de paquetes UDP y el decodificador del cuadro de video. El receptor de video fue construído desde cero y el decodificador de video fue tomado del proyecto Java AR-Drone. Este último, es del cual parten todos los demás hilos de procesamiento digital de imágenes.

[3]Foro en línea: https://projects.ardrone.org/

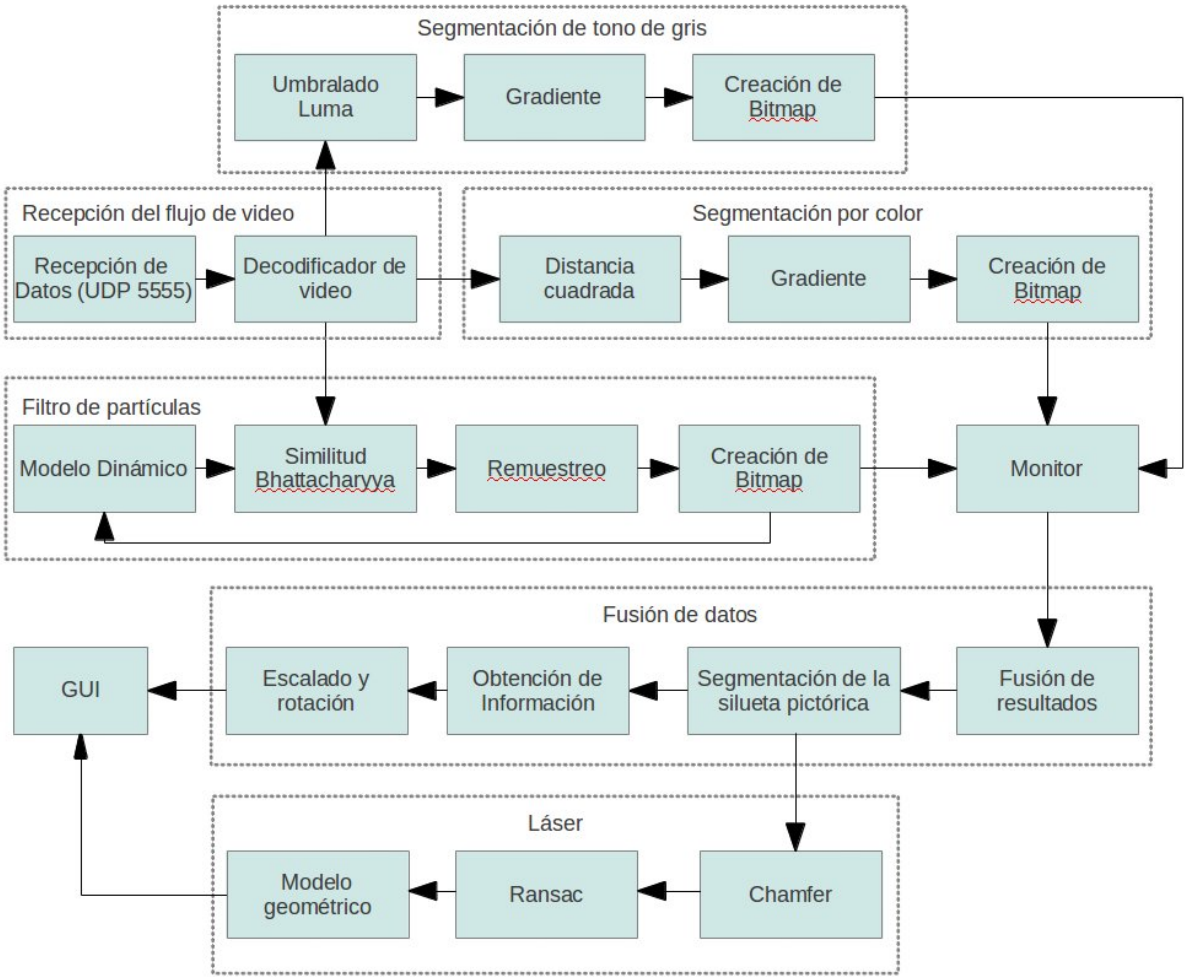


Figura 4.1: Esquema de los Hilos de PDI

En la figura 4.1, se ilustra la interacción que existe entre los seis hilos de los que consta el procesamiento digital de imágenes de IHRVANT. El hilo más elemental es el encargado de la recepción del flujo de video a través del puerto UDP 5555 y su decodificación. El decodificador entrega los dos canales de información cromática (C_b,C_r) al hilo de Filtro de partículas y al de segmentación por color; el canal Luma sólo se entrega al de segmentación por tono de gris. Cada uno de los hilos realiza operaciones específicas y entrega sus resultados en forma de mapa de bits para que de forma sincronizada lleguen al quinto hilo que fusiona los datos, obtiene la silueta pictórica del usuario y calcula algunas estadísticas. El hilo láser estima con un modelo geométrico y la silueta pictórica, la pose del usuario. Al igual que la fusión de datos, su resultado lo envía a la GUI para presentarla al

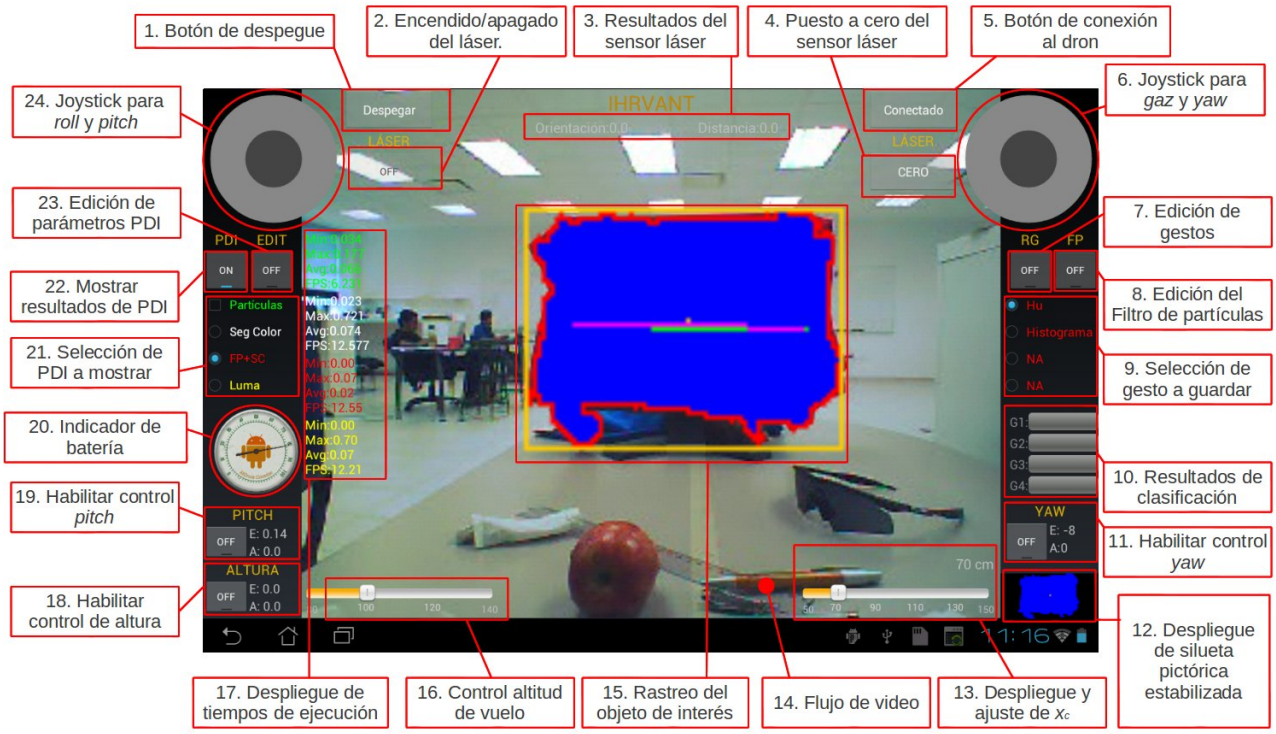


Figura 4.2: Interfaz de IHRVANT

usuario.

En este capítulo primero se describirá la interfaz de IHRVANT, después cada una de los hilos de procesamiento digital de imágenes y sus implementaciones; empezando por el decodificador de video, seguido de la segmentación por color y tono de gris, filtro de partículas, la fusión de datos y al final, el hilo de segmentación del láser.

4.2 Descripción de la interfaz de IHRVANT

La Figura 4.2 muestra la interfaz del proyecto IHRVANT en la pantalla de la *tablet*. Como se puede observar, la mayor parte de la pantalla la ocupa la proyección del flujo de video y a continuación se listan las descripciones de cada componente.

1. Botón de despegue: cuando es presionado y el cuadricóptero está en línea, el cuadricóptero despega.

2. Encendido/apagado del láser: con este botón se puede prender o apagar el láser de forma remota a través de la conexión Bluetooth con la tarjeta IOIO.

3. Resultados del sensor láser: despliega el resultado de la estimación de orientación y distancia al objeto de interés.

4. Puesto a cero del sensor láser: con este botón el sensor láser calcula su referencia a partir de la cual, estima la distancia a la que se encuentra el objeto de interés.

5. Botón de conexión al dron: establece la comunicación con el dron e inicializa todos los hilos de los que está compuesto IHRVANT.

6. Joystick para controlar *gaz* y *yaw*: con un evento *touch* el usuario puede manipular uno o ambos grados de libertad del dron.

7. Edición de gestos: si es activado, el usuario puede definir cada uno de los gestos en los cuatro bancos disponibles.

8. Edición del filtro de partículas: si es activado, el usuario puede usar su dedo para elegir el histograma a rastrear por el filtro en el cuadro de video.

9. Selección de gesto a guardar: estos son los cuatro bancos que pueden usarse para almacenar un gesto o para seleccionar el método para clasificar la silueta pictórica.

10. Resultados de clasificación: cuando dos o más bancos están inicializados, IHRVANT los compara continuamente con la silueta pictórica actual. Los resultados de la comparación se ilustran de forma gráfica en esta área.

11. Habilitar control *yaw*: si es activado, IHRVANT controla el ángulo *yaw* automáticamente.

12. Despliegue de silueta pictórica estabilizada. El área segmentada es escalada y trasladada al centro de este bitmap.

13. Despliegue y ajuste de x_c: Con este control el usuario puede asignar un valor a x_c y poner a cero el sensor láser (ver Sección 4.11).

14. Flujo de video: IHRVANT despliega en esta área cada cuadro del flujo de video.

15. Rastreo del objeto de interés: el resultado del procesamiento digital de imágenes se ilustra sobrepuesto al cuadro de video.

16. Control altitud de vuelo. Con este control deslizable, el usuario puede elegir la altura a la que volará el cuadricóptero mientras lo persigue desde el aire.

17. Despliegue de tiempos de ejecución. Muestra estadísticas del tiempo de ejecución de los distintos hilos de procesamiento digital de imágenes: cuadros por segundo y tiempos de ejecución (promedio, mínimo y máximo).

18. Habilitar control altura: si es activado, IHRVANT mantiene la altura de vuelo igual a la especificada por el control deslizable de forma automática.

19. Habilitar control *pitch*: si es activado, IHRVANT controla el ángulo *pitch* automáticamente.

20. Indicador de batería. De forma gráfica se muestra el nivel de carga de la batería del dron.

21. Selección de PDI a mostrar. Se usa para elegir qué proceso de segmentación es mostrado, sólo las partículas se pueden mostrar simultáneamente con algún otro.

22. Mostrar resultados de PDI: si es activado, IHRVANT muestra el resultado de la segmentación sobrepuesto al cuadro de video.

23. Edición de parámetros PDI: si es activado, el usuario puede inicializar el método de segmentación deseado: segmentación por color o tono de gris.

24. Joystick para controlar *roll* y *pitch*: con un evento touch el usuario puede manipular uno o ambos grados de libertad del dron.

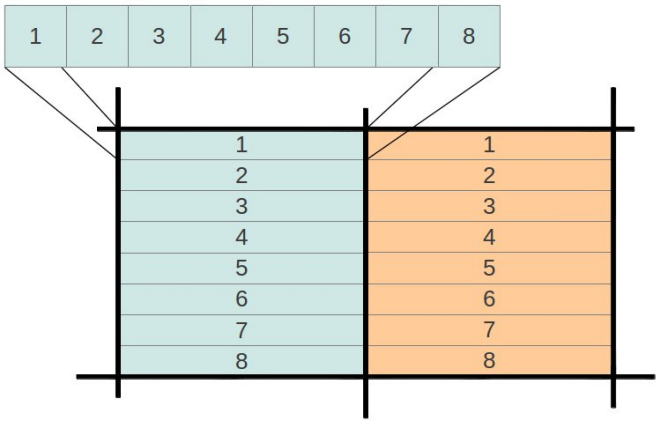
4.3 Cualidades técnicas del decodificador de video

Con la Figura 3.7 se describió la arquitectura base de la aplicación IHRVANT y en la Sección 3.3.1 se hizo mención de la arquitectura del decodificador de video. La resolución del flujo de video proveniente de la cámara frontal es QVGA (320×240 pixeles) y la relación de submuestreo empleada es $4:2:0$, por lo tanto el tamaño del cuadro de los canales cromáticos (C_b,C_r) es de una cuarta parte del QVGA, o sea 160×120 pixeles. Contemplando lo anterior se pueden definir algunas constantes:

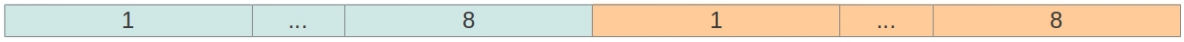
- El número de GOB's en un cuadro de video está dado por el cociente de la resolución vertical y el alto de un macro bloque: $\psi_x = \frac{240}{16} = 15$.
- El número de macro bloques por GOB está dado por el cociente de la resolución horizontal y el ancho de un macro bloque: $\psi_y = \frac{320}{16} = 20$.
- El número total de macro bloques (ψ_t) en un cuadro de video está dado por el producto de ψ_x y ψ_y, por lo tanto $\psi_t = 300$.

Cada bloque de los 6 que constituye un macrobloque consta de 64 pixeles representados en memoria por una variable de tipo short (2 bytes), por lo tanto para almacenar en memoria un cuadro de video se requieren 37.5 KIB y 150 KIB para almacenar cada uno de los cuadros de crominancia y de luminancia, respectivamente.

La sincronización que existe entre el decodificador y los hilos de procesamiento digital de imágenes está basada en el modelo productor-consumidor y semáforos; este modelo ha sido estudiado ampliamente en los sistemas informáticos. El decodicador produce GOB's a consumir por los hilos de procesamiento digital de imágenes. Cada uno de los hilos que desea acceder a esta información cuenta con un búfer, fijo en memoria, para almacenar la cantidad de información que le sea pertinente y también cuenta con una interfaz que garantiza exclusión mutua al acceder a este recurso. Técnicamente, el hilo del decodificador copia los datos del cuadro en turno en los búfers de los hilos de procesamiento digital de imágenes, para después desbloquear el hilo de procesamiento agregando un permiso al semáforo por el que estaba esperando. Se escogió el modelo productor-consumidor, por



(a) 2 bloques contiguos y su disposición en un cuadro de video.



(b) Almacenamiento en memoria de dos bloques contiguos.

Figura 4.3: Disposición en memoria de dos bloques contiguos, cada bloque con 8 renglones de alto consta de 8 pixeles a lo ancho.

que en el caso que la segmentación demore más tiempo que la decodificación, sólo disminuirá la frecuencia de entrega de resultados de la segmentación, mientras el hilo de decodificación podrá seguir decodificando el video a 18 Hz.

El decodificador almacena cada bloque de la imagen de forma lineal. Por ejemplo, para el caso de dos bloques contiguos (ver Figura 4.3a) que representan un área de 8×8 pixeles cada uno, en memoria son almacenados en un arreglo lineal como se ilustra en la Figura 4.3b. Por tanto, para almacenar un GOB de un cuadro del canal C_b cuya resolución es 160×120, se requiere de un arreglo lineal de 1280 elementos del tipo short, o 2.5KIB. De forma similar, para almacenar un GOB del canal de luminancia, se requiere un arreglo de 5120 elementos o 10 KIB.

Como se mencionó, el hilo de la recepción del flujo de video recibe uno a uno, los ψ_x GOBs para reconstruir un cuadro de video. Cuando recibe un GOB, lo comparte inmediatamente con los tres hilos de segmentación: por tono de gris, por color y el filtro de partículas. Las siguientes secciones describen como proceden cada uno de estos hilos de procesamiento y al final se describirá como se fusionan los datos en su respectivo hilo.

4.4 Segmentación por color

El hilo del decodificador de video es el responsable de copiar la información requerida por cada uno de los hilos de procesamiento digital de imágenes a sus búfers correspondientes y así iniciar el proceso de cada hilo. Específicamente con el hilo de segmentación por color, cada GOB decodificado se entrega haciendo una copia de los 1280 pixeles por cada canal cromático. Posteriormente se calcula la distancia cromática de cada uno de los pixeles del GOB con el color de referencia, como está descrito por la Ecuación 3.1. Queda por aclarar la selección del color de referencia con coordenadas $(\varsigma_b, \varsigma_r)$ y el umbral r^2.

Para la selección del color de referencia y el umbral se usaron las cualidades *touch* de la *tablet*. El usuario puede elegir el color que la aplicación debe segmentar tocando sobre la pantalla dentro del cuadro de video. El flujo de video con resolución QVGA es desplegado en un área de 1000×752 pixeles en la pantalla de 11 pulgadas, cuya resolución es de 1280×800 pixeles y es capaz de detectar hasta 10 eventos *touch*. Dadas las dimensiones del área en la que se despliega el video y la resolución QVGA, se usó un factor de 0.32 para escalar las coordenadas (x, y) del evento *touch* a coordenadas a la escala del flujo de video (h, k).

Para hacer más certera la selección del color, usamos una máscara gaussiana cuadrada de L pixeles de lado para estimar el color que represente a los valores de la vecindad del evento *touch* con coordenadas (h, k). La máscara gaussiana (Ecuación 4.1) usa dos medias (μ_x, μ_y) y dos varianzas (σ_x, σ_y) que fueron aproximadas a partir del ancho de la ventana de muestreo L.

$$N(x, y) = \exp\left(-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)\right) \tag{4.1}$$

Para un ancho impar L de la ventana de muestreo elegimos a μ_x y μ_y como $\mu_x = \mu_y = \frac{L-1}{2} + 1$. Por otro lado, sabemos que el rango $[\mu - 3\sigma, \mu + 3\sigma]$ incluye a un conveniente $99.7\,\%$ de muestras en una distribución normal, entonces definimos $3\sigma_x = 3\sigma_y = \frac{L-1}{2}$. En la Figura 4.4, se ilustra la función normal que se centra en las coordenadas del evento *touch* (h, k) para estimar el valor de ς_b y ς_r con las Ecuaciones 4.2 y 4.3. Las variables C_b y C_r son matrices que contienen la información

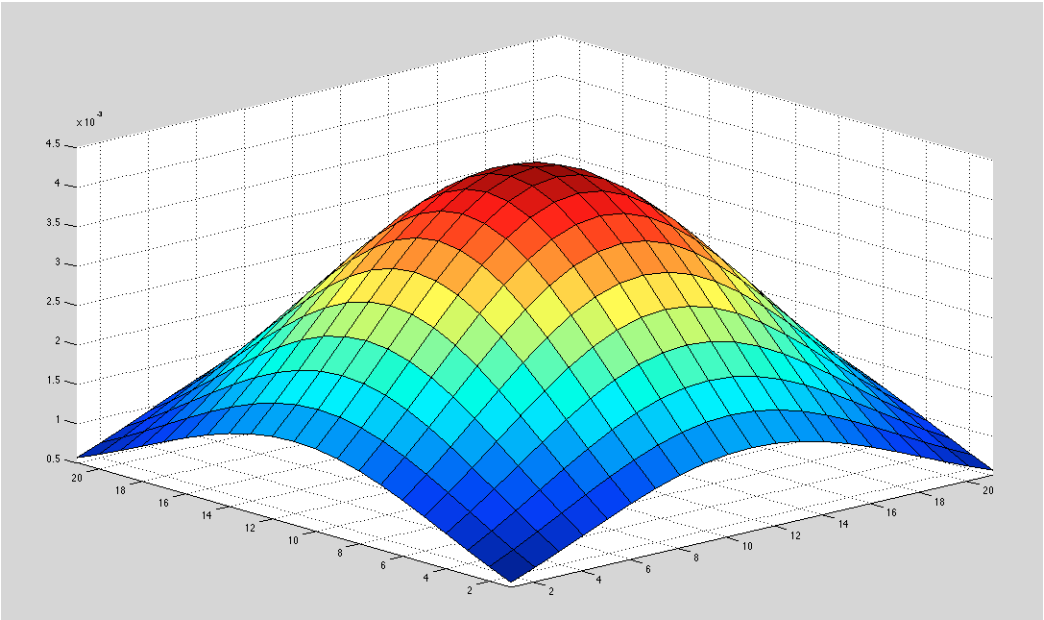


Figura 4.4: Función normal empleada para la selección del color con $L = 21$

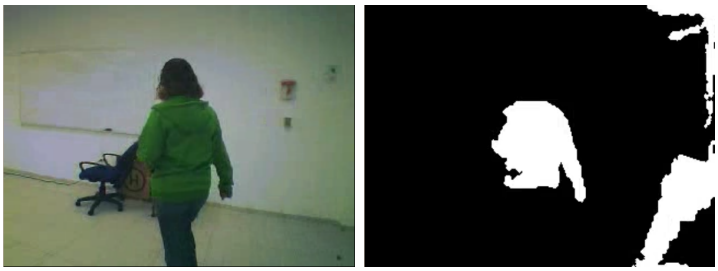
cromática del cuadro de video.

$$\varsigma_b = \sum_{x=0}^{L-1}\sum_{y=0}^{L-1} C_b(x+h-\frac{L-1}{2}, y+k-\frac{L-1}{2}) \times N(x+1, y+1) \tag{4.2}$$

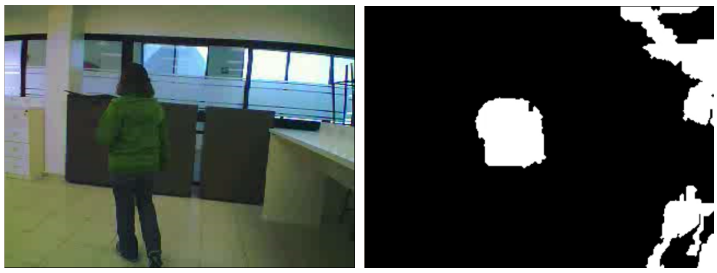
$$\varsigma_r = \sum_{x=0}^{L-1}\sum_{y=0}^{L-1} C_r(x+h-\frac{L-1}{2}, y+k-\frac{L-1}{2}) \times N(x+1, y+1) \tag{4.3}$$

La selección del umbral se realiza con un evento *touch* múltiple, inicialmente el umbral u tiene el valor de $r = 13$. El primer dedo, con coordenadas $P_1(x_1, y_1)$, lo usamos para elegir el color a segmentar y con un segundo dedo con coordenadas $P_2(x_2, y_2)$ se define un factor proporcional al cambio en la distancia euclidiana entre ambos puntos, dado por $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Al detectarse el segundo evento *touch*, se almacena el valor d en una variable independiente ($\varrho = d$) y a cualquier cambio de d, con respecto a los dos puntos (P_1,P_2), se calculará el nuevo umbral como $u = \frac{d}{\varrho} \times r$.

En las Figuras 4.5a y 4.5c, se muestran dos cuadros de una secuencia de video que se grabó con la cámara del cuadricóptero y una aplicación para Android. Es evidente la aberración cromática de



(a) Cuadro original (b) Segmentada por color



(c) Cuadro original (d) Segmentada por color

Figura 4.5: Cuadros de video segmentados por color

la cámara que aumenta la cantidad de falsos positivos en el perímetro del cuadro al segmentar. En estos cuadros, el color de interés es el verde de la chamarra del usuario y el área tenía paredes y suelo blanco. La segmentación por color no es suficiente para rastrear el color verde de la chamarra que portaba el usuario, tal como se muestran en las Figuras 4.5b y 4.5d, pues no provee una manera de distinguir entre el área de interés y los falsos positivos en el borde del cuadro de video.

Esta técnica de segmentación por color se incluyó de forma nativa en la aplicación IHRVANT con un buen desempeño. Como se explicó en la Sección 3.1.2.4, el submuestreo de los canales de crominancia 4:2:0, hace posible que sólo sea necesario procesar un cuadro cuyo tamaño es una cuarta parte del QVGA (resolución del flujo de video). Este hilo, sincronizado con el decodificador de video, recibe cada GOB y lo procesa paralelamente a la decodificación del video. De esta forma, el decodificador continúa su operación normal, mientras el hilo de segmentación procesa un GOB.

La existencia de falsos positivos en la segmentación por color justifica el uso del filtro de partículas, que permite discriminar entre el objeto de interés y los falsos positivos en una secuencia de video. Además, la segmentación por tono de gris aportará información relevante al contraste de los objetos y

(a) Cuadro 1 (b) Cuadro 2 (c) Cuadro 3

Figura 4.6: IHRVANT desplegando el resultado de la segmentación por tono de gris.

se podrá distinguir aún mejor el área a segmentar. La fusión del filtro de partículas y la segmentación por color y tono de gris permitirá estimar la silueta pictórica del usuario, a través del flujo de video.

4.5 Segmentación por tono de gris

El tercer canal que provee el decodificador es el de luminiscencia con resolución QVGA. Para la selección del tono de gris a segmentar (ς_y) también se usa una máscara gaussiana con un valor de L distinto y centrada en las coordenadas (h, k) donde el usuario colocó su dedo (Ecuación 4.1). La segmentación por tono de gris toma un rango de luminiscencia $[\varsigma_b \pm u]$ para elegir los pixeles que son de interés (Ecuación 4.5). En la Figura 4.6, se ilustran los resultados que despliega IHRVANT al segmentar por tono de gris. El usuario puede elegir con su dedo, el área cuyo tono de gris desea segmentar.

$$\varsigma_y = \sum_{x=0}^{L-1}\sum_{y=0}^{L-1} Y(x+h-\frac{L-1}{2}, y+k-\frac{L-1}{2}) \times N(x+1, y+1) \tag{4.4}$$

$$g(x,y) = \begin{cases} 1 & \text{si } (\varsigma_y - u) \leq Y(x,y) \leq (\varsigma_y + u) \\ 0 & \text{de otra forma.} \end{cases} \tag{4.5}$$

4.6 Filtro de partículas

El algoritmo descrito en la Sección 3.3.2 puede adaptarse para rastrear un objeto en un flujo de video basado en información cromática y de eso trata esta sección. El hilo del filtro de partículas está sincronizado con el decodificador de video de forma similar que con el hilo de segmentación por color. La diferencia radica en que, del cuadro de video en turno acepta ψ_x GOBs, antes de continuar su operación normal; es así porque el algoritmo requiere ambos cuadros de crominancia para realizar las operaciones descritas en la siguiente sección.

4.6.1 Adaptación del filtro de partículas

El punto de inicio es un modelo de espacio de estados de Markov, donde una hipótesis *a priori* es condicionalmente independiente a un proceso de observación. Donde un vector de estado oculto $\mathbf{x_t}$, representa la coordenada de la persona en un cuadro de video que constituye, a su vez, la observación $\mathbf{z_t}$ en el tiempo t. Desafortunadamente en problemas de seguimiento visuales, las hipótesis *a priori* no se pueden aproximar de manera lineal, pero recursivamente en un marco de trabajo secuencial de Monte Carlo es posible.

La hipótesis $b(x_t)$ se aproxima con un número finito de M muestras, las partículas. El vector de estado $\mathbf{x_t}$ para cada partícula se define como en la Ecuación 4.6. Los componentes de $\mathbf{x_t}$ son escaleres que definen las coordenadas (x, y) en el marco de referencia del cuadro del video en el tiempo t y $t-1$.

$$\mathbf{x_t} = \begin{pmatrix} x_t & y_t & x_{t-1} & y_{t-1} \end{pmatrix}^T \tag{4.6}$$

La hipótesis *a priori* $\bar{b}(x_t)$ se aproxima con un modelo que mueve a todas las partículas indepen-

dientemente con un modelo de transición de estado de segundo orden (Ecuación 4.7).

$$\mathbf{x_{t+1}} = \begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{x_t} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \mathbf{v_t}, \mathbf{v_t} \sim N(0, \Sigma) \tag{4.7}$$

$$\mathbf{x_{t+1}} = A\mathbf{x_t} + C\mathbf{v_t}, \mathbf{v_t} \sim N(0, \Sigma) \tag{4.8}$$

El siguiente paso en el algoritmo del filtro de partículas es el de ponderación, que asigna un peso a cada una de ellas, para que después se realice un remuestreo basado en dicha ponderación.

La ponderación de la m-ésima partícula $p(z_t|\mathbf{x_t^{[m]}})$ se basa en el coeficiente de similitud de Bhattacharyya. Este coeficiente define una distancia D de similitud entre dos histogramas bidimensionales divididos en N *buckets*. El primer histograma se escoge previamente como la referencia y se infiere que se trata del color a rastrear, su histograma se expresa como $\mathbf{q}^* = q^*(n)_{n=1\ldots N}$ con $\sum_{n=1}^{N} q_t(n; \mathbf{x}) = 1$. El segundo $\mathbf{q_t}(\mathbf{n}, \mathbf{x})$ representa el histograma de la vecindad cuadrada W de la partícula ubicada en coordenadas $\mathbf{x} = (x, y)$, dentro del cuadro de video y en un tiempo t. La vecindad W mide w pixeles por lado y debe ser impar. La distancia D entre los dos histogramas divididos en N *buckets* está dada por la Ecuación 4.9.

$$D[\mathbf{q}^*, \mathbf{q_t}(\mathbf{x})] = \left[1 - \sum_{n=1}^{N} \sqrt{q^*(n) q_t(n; \mathbf{x})}\right]^{\frac{1}{2}} \tag{4.9}$$

En este trabajo, el coeficiente de similitud se basa en el propuesto en [28], donde del estudio de distintas corridas exitosas de seguimiento se observó un comportamiento exponencial consistente para la distancia cuadrada (D^2). Se infirió que $p(\mathbf{z_t}|\mathbf{x_t}) \propto p(D^2[\mathbf{q}^*, \mathbf{q_t}(\mathbf{x})])$ y se propuso que la función de distribución de probabilidad estuviera dada por la Ecuación 4.10. En [28], se aclara que no hay manera exacta para determinar el valor de λ, por lo cual, se le asignó un valor fijo de 20.

$$p(\mathbf{y_t}|\mathbf{x_t}) \propto exp\left(-\lambda D^2[\mathbf{q}^*, \mathbf{q_t}(\mathbf{x_t})]\right) \tag{4.10}$$

El Algoritmo 3 describe la adaptación del filtro de partículas de la Sección 3.3.2.3. Recibe como argumentos de entrada un conjunto de partículas $\mathbf{X_t}$ de tamaño M, el mapa de bits de los canales de crominancia $B_{C_bC_r}$ y un histograma referencia $\mathbf{q}^* = q^*(n)_{n=1...N}$ del objeto a rastrear. En la línea 4, se calcula una hipótesis *a priori* con el modelo descrito en la Ecuación 4.8 para después, en la línea 5, calcular el histograma de la vecindad W de cada partícula $\tilde{\mathbf{x}}_{\mathbf{t+1}}^{[\mathbf{m}]}$, tomando la información de $B_{C_bC_r}$. Se le asigna un peso a cada partícula comparándolo con el histograma de referencia $\mathbf{q}_{\mathbf{i}}^*(n)$ (línea 8). El remuestreo con remplazo se realiza en el segundo ciclo *for* del algoritmo, para generar un conjunto de partículas $\mathbf{X_t}$ de tamaño M. Los factores K y J de las líneas **??** y 8 son tales que normalicen la funciones de distribución de probabilidad correspondientes. Este algoritmo tiene complejidad $O(2Mw^2)$.

Algoritmo 3 Filtro de partículas$(B_{C_bC_r}, \mathbf{X_t^{(i)}}, \mathbf{q_i^*}(n))$

1: $\bar{X}_t \Leftarrow 0$
2: $X_t \Leftarrow 0$
3: **for** $m = 1 \rightarrow M$ **do**
4: $\quad \tilde{\mathbf{x}}_{\mathbf{t+1}}^{(\mathbf{i})[\mathbf{m}]} \Leftarrow A\mathbf{x}_{\mathbf{t}}^{(\mathbf{i})[\mathbf{m}]} + C\mathbf{v_t}$
5: $\quad$ **for all** $u \in W(\tilde{\mathbf{x}}_{\mathbf{t+1}}^{(\mathbf{i})[\mathbf{m}]})$ **do**
6: $\quad\quad q_{t+1}(\lfloor B_{C_bC_r}(u)/n \rfloor; u) + +$
7: $\quad q_{t+1}(n; \tilde{\mathbf{x}}_{t+1}^{(i)[m]}) = Jq_{t+1}(n; \tilde{\mathbf{x}}_{t+1}^{(i)[m]})$
8: $\quad \tilde{w}_{t+1}^{[m]} \Leftarrow Kexp\left(\sum_{n=1}^{N} \lambda \sqrt{q_i^*(n) q_{t+1}(n; \tilde{\mathbf{x}}_{t+1}^{(i)[m]})}\right)$
9: **for** $m = 1 \rightarrow M$ **do**
10: $\quad$ draw i with probabily $\propto \tilde{w}_{t+1}^{[i]}$
11: $\quad$ add $\mathbf{x}_{t+1}^{[i]}$ to $\mathbf{X_t}$
12: **return** $\mathbf{X_t}$

4.6.2 Implementación del filtro de partículas

En la Figura 4.7, se ilustra de forma somera el algoritmo del filtro de partículas, pero ejemplifica el funcionamiento del hilo de procesamiento. De derecha a izquierda, la operación comienza con el uso del modelo dinámico para estimar la hipótesis *a priori* de cada partícula (Ecuación 4.7); paso seguido, se asigna un peso a cada partícula (Ecuación 4.10) comparando el histograma de la vecindad

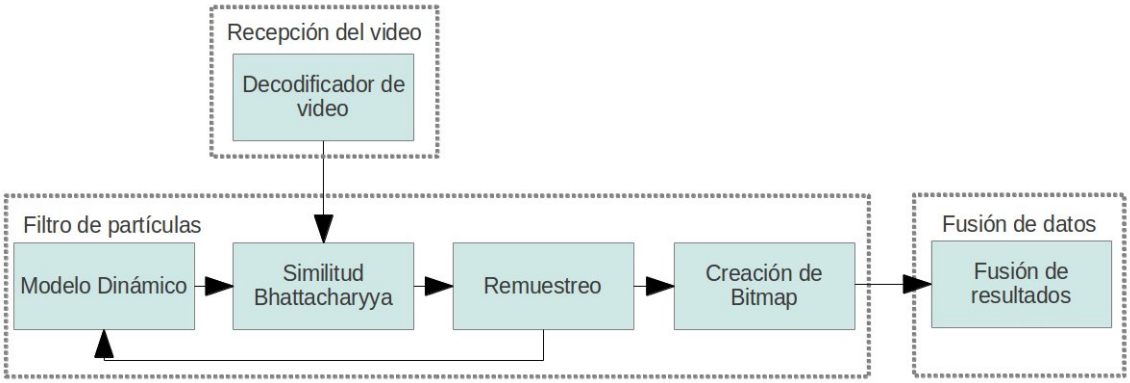


Figura 4.7: Hilo de procesamiento del filtro de partículas.

de la partícula $\mathbf{q_t}(\mathbf{x})$ con el de referencia $\mathbf{q}^*$. El remuestreo de las partículas con remplazo, donde la posibilidad de una partícula de ser muestreada es proporcional al peso asignado, hace que sólo las partículas con mayor peso subsistan. Finalmente, las coordenadas de cada partícula son usadas para crear un mapa de bits que ilustra su ubicación en el cuadro de video.

Para inicializar el histograma de referencia $\mathbf{q}^*$ también usamos las cualidades *touch* de la *tablet*. Las coordenadas (h, k) del evento *touch* las usamos para calcular el histograma de referencia a partir de una vecindad cuadrada de tamaño w pixeles por lado. Si el usuario así lo desea, puede deslizar su dedo a través de la pantalla, para que múltiples histogramas sean promediados y obtener una mejor aproximación de $\mathbf{q}^*$.

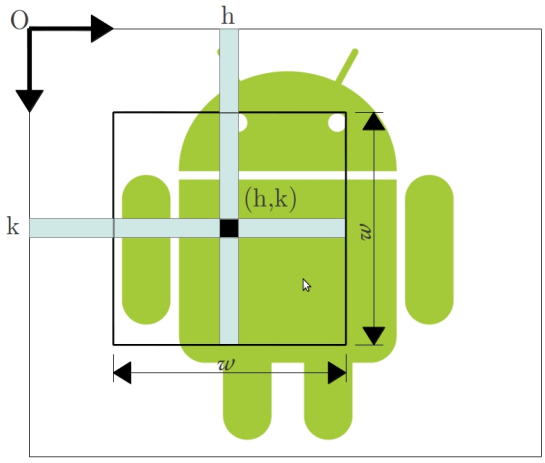
Habiendo descrito algunas cualidades técnicas de la implementación de este filtro de partículas, ahora mencionaremos los parámetros de ajuste:

- M Número de partículas usadas por el filtro.
- N^2 Número de $buckets$ para el cálculo del histograma.
- w Tamaño de la vecindad para el calculo de $\mathbf{q_t}(\mathbf{x})$.
- Σ Varianza del factor aleatorio en el modelo de segundo orden (Ecuación 4.7).

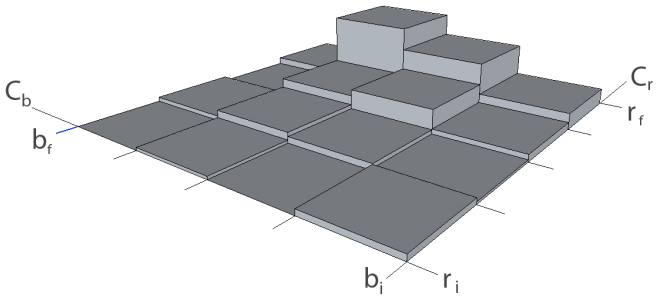
Cabe mencionar que entre más grandes sean los primeros tres parámetros, menor será el rendimiento computacional del filtro de partículas. El cuarto parámetro determinará el sumando probabilístico en el modelo dinámico de segundo orden (Ecuación 4.7).

El número de partículas M en algunas aplicaciones radica en el orden de 1000 [28]. En esta

implementación, cada una de ellas evalúa la similitud del histograma de su vecindad con un histograma de referencia $\mathbf{q}^*$. En la Figura 4.8a, se muestra una vecindad centrada en coordenadas (h, k) y por lado w; se elige un valor de w impar para que la distancia horizontal y vertical (en pixeles) del centro a los bordes sea entera.



(a) Representación gráfica de una vecindad centrada en las coordenadas (h, k) de un mapa de bits.

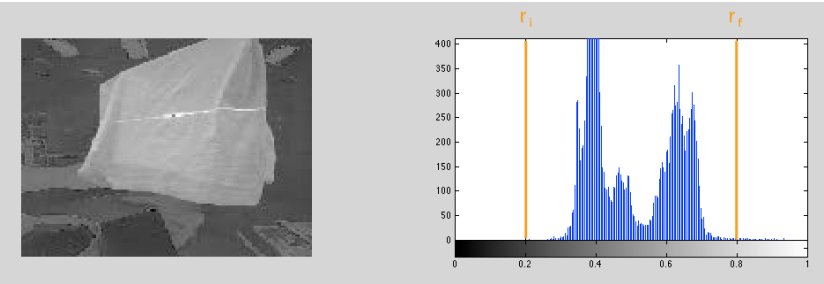


(b) Histograma bidimensional, con $N = 4$

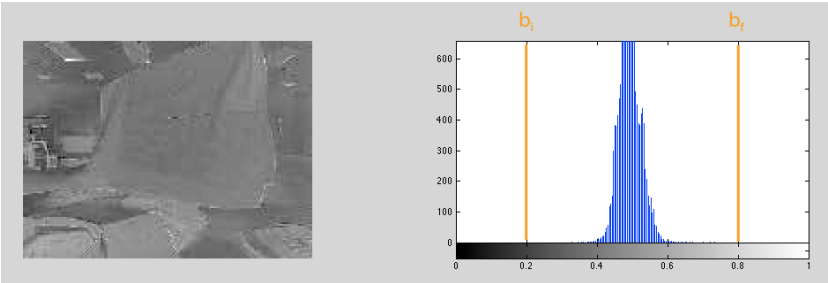
Figura 4.8: Representación de la vecindad de una partícula ubicada en coordenadas (h, k) y un histograma bidimensional.

El segundo parámetro N^2, representa el número de *buckets* para calcular el histograma de la vecindad de tamaño w. El histograma se calcula de manera bidimensional, tal como se ilustra en la Figura 4.8b. Los rangos dinámicos de cada canal cromático comprendidos entre $[b_i, b_f]$ para el canal C_b está dividido en N segmentos iguales: sucede lo mismo con el canal C_r, pero se usan los límites $[r_i, r_f]$.

El valor trivial, pero el no más acertado para el rango dinámico de ambos límites $[r_i, r_f]$ y $[b_i, b_f]$, es $[0, 255]$. La Figura 4.9a muestra un cuadro del canal cromático C_r y su sesgado histograma con el rango dinámico normalizado. Aún más sesgado se encuentra el histograma normalizado del canal C_b para el mismo cuadro (ver Figura 4.9b). En 10 cuadros en diferentes condiciones de iluminación el patrón se repetía, por lo cual se delimitó el rango dinámico de los histogramas a los mostrados en la Figura 4.9. La utilidad de disminuir el rango dinámico de los canales, para el cálculo del histograma bidimensional, radica en la concentración de zonas de mayor energía. Así cada *bucket* contará con



(a) Cuadro del canal C_r y su histograma con límites en $r_i = 0.2$ y $r_f = 0.8$



(b) Cuadro del canal C_b y su histograma con límites en $b_i = 0.2$ y $b_f = 0.8$

Figura 4.9: Energía en los histogramas de ambos canales cromáticos

valores más significativos que mejor representen a la vecindad de cada partícula.

En la Figura 4.10, hay dos cuadros sobrepuestos del flujo de video con el mismo usuario en diferentes poses. El resultado de la segmentación por color son todos los pixeles de color blanco, mientras que las partículas están sobrepuestas de color rojo. La figura ilustra dos problemas importantes. El primero es distinguir los falsos positivos que resultan de la segmentación por color que aparecen en el área de las piernas y en el fondo del cuadro de video. Es importante descartarlos para que no generen datos erróneos que puedan resultar en el mal funcionamiento de la interfaz. El segundo problema ocurre cuando dos o más áreas son segmentadas por su tonalidad cromática. ¿Cómo distinguir una de la otra? Con el filtro de partículas se puede distinguir entre ellas, pues las partículas se encontrarán sobrepuestas al área de interés. Mediante la fusión con operaciones morfológicas de la segmentación por color y el filtro de partículas es posible resolver los dos problemas mencionados anteriormente, para obtener correctamente la silueta pictórica del usuario. Por último, para poner a prueba el algoritmo del filtro de partículas, se probó si lograba rastrear al usuario exitosamente a través del flujo de video que recibe IHRVANT del dron. La Figura 4.11 muestra una secuencia de

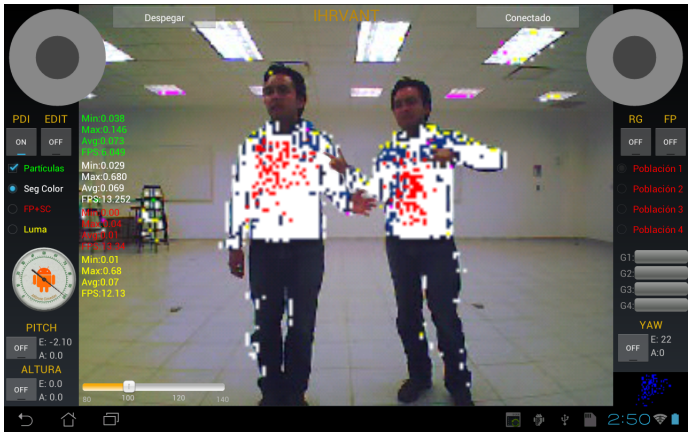


Figura 4.10: Filtro de partículas y segmentación por color en la aplicación IHRVANT. Se trata del mismo usuario en dos cuadros distintos del video.

cuadros pertenecientes a un video que se tomó del algoritmo en operación ya integrado a IHRVANT. En cada cuadro, se muestra de color blanco el resultado de la segmentación por color y en color rojo las partículas rastreando el usuario. El cuadro de la Figura 4.11b muestra como el algoritmo rastrea al usuario durante un salto; de la Figura 4.11d a la 4.11i, el usuario trota describiendo un círculo frente a la cámara del dron y el algoritmo también logra rastrearlo. En los últimos tres cuadros de la figura, el usuario camina frente a la cámara y es rastreado de la misma forma.

4.7 Silueta pictórica

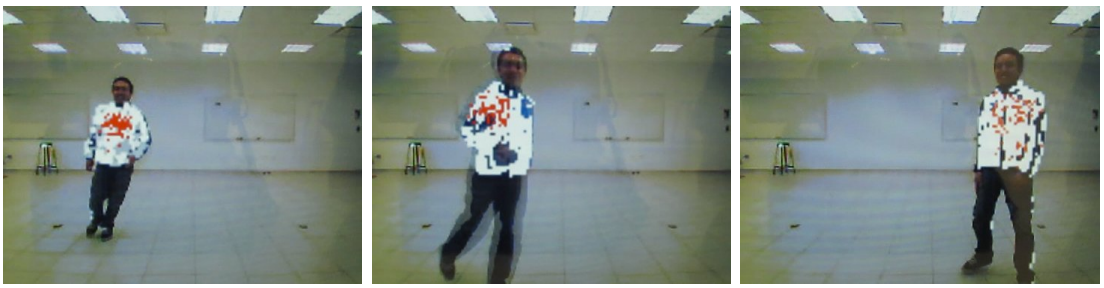
En este apartado se describen las técnicas empleadas para la obtención de la pose del usuario basándose en los resultados de pasos anteriores (filtro de partículas y segmentación por color), pues de esto dependerá la manera en que será clasificada la silueta para definir instrucciones o comandos para esta interfaz.

4.7.1 Fusión de datos

Para fusionar los datos que cada una de las técnicas descritas en la metodología arrojan, se eligieron varias técnicas bien conocidas en el área de procesamiento digital de imágenes y el resultado final, será la obtención de la silueta pictórica del dorso del usuario en el cuadro de video.

(a) Cuadro 1 (b) Cuadro 2 (c) Cuadro 3

(d) Cuadro 4 (e) Cuadro 5 (f) Cuadro 6



(g) Cuadro 7 (h) Cuadro 8 (i) Cuadro 9

(j) Cuadro 10 (k) Cuadro 11 (l) Cuadro 12

Figura 4.11: Rastreo del usuario con el filtro de partículas. El resultado de la segmentación por color está sobrepuesta al usuario de color blango y de color rojo, todos los pixeles que representan a las partículas.

Partimos de dos cuadros de video, uno resultado de la segmentación por color (ver Figura 4.12a) y el segundo, del doble de resolución que el anterior, resultado de la segmentación en un rango de tonalidades en el canal de luminiscencia del decodificador (ver Figura 4.12b). El primer paso es fusionar ambos cuadros verificando que hallan sido segmentados al menos tres de los cuatro pixeles en el cuadro de la Figura 4.12b, que traslapan a cada uno de los pixeles del cuadro en la Figura 4.12a; los pixeles que cumplen con esta condición cambian de color a verde en la Figura 4.12c.

El paso descrito anteriormente fue crucial para distinguir falsos positivos que consistían en áreas de la misma tonalidad cromática que el área de interés, por ejemplo la chamarra azul del usuario como área de interés y una ventana con tonalidades del azul como falso positivo. Fusionando la información de contraste que existe entre las áreas de la misma tonalidad cromática, evitamos estos falsos positivos. Considere como ejemplo de un falso positivo con estas cualidades al pixel marcado con un asterisco rojo en el cuadro de la Figura 4.12a.

Paso siguiente, realizamos una dilatación del área marcada con color verde con un elemento estructural del tipo rombo de 3 pixeles por lado (ver Figura 4.12d). A manera ilustrativa en el cuadro resultante, hay dos áreas segmentadas por el mismo método sin ninguna forma de distinguir entre ellas cuál es la importante. La utilidad del filtro de partículas en esta estrategia es indicar cuál es el área de interés superponiendo las partículas a lo segmentado por color (ver Figura 4.12e). Sólo resta usar el algoritmo *bush fire* para usar las partículas como puntos de inicio del incendio (ver Figuras 4.12f y 4.12i).

De esta forma todo los puntos que no hayan sido marcados con azul serán descartados y sólo permanecerá el área perteneciente a la silueta pictórica, para poder entonces obtener las dimensiones de su caja envolvente y su centroide (ver Figura 4.12h). La caja envolvente puede verse como un bitmap cuyas dimensiones son descritas por un vector con cuatro entradas: (x_{bb}, y_{bb}, w_{bb}, h_{bb}) que representan las dos coordenadas de la esquina superior izquierda, su ancho y largo; por otro lado, el centroide de la silueta tiene coordenadas (c_x, c_y).

El problema que aún puede presentarse depende de circunstancias inherentes al vuelo del cuadricóptero, que por tratarse de un vuelo ligeramente inestable, hace al área a segmentar susceptible a perturbaciones. Gracias a que el cuadricóptero provee los datos que genera la IMU que lleva a

bordo, es posible estabilizar el área de la silueta pictórica con sencillas transformaciones geométricas de tipo translación y rotación. Es posible conocer cualquier cambio en la inclinación del horizonte a través de los datos de navegación y corregir la inclinación en el video para mantener la percepción del horizonte. La manera en la que se estabiliza el cuadro de video es descrita en el siguiente apartado.

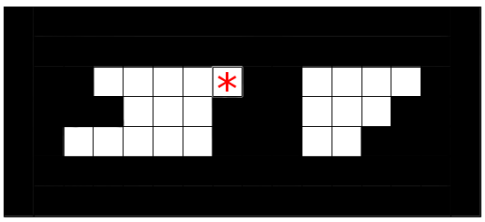
4.8 Estabilización del área segmentada

Para poder definir el área a segmentar del cuadro de video que mejor aproxime las proporciones y dimensiones del dorso humano, es necesario adoptar un modelo antropomórfico. En artes gráficas son populares dos modelos del cuerpo humano basados en las dimensiones de la cabeza humana, la cual funciona como unidad de medida u. El primero modelo, de proporciones ideales, resalta la divinidad o heroísmo de algún personaje; la altura del cuerpo humano en este modelo tiene 8 cabezas de alto ($8u$). El segundo modelo es más realista y contempla que una persona mide $7\frac{1}{2}u$ de alto (ver Figura 4.13). En la misma figura observamos que la distancia que existe entre el cuello y la cintura es de $2\frac{1}{2}u$ y que es igual a la distancia entre los hombros y muñecas de la persona. De forma similar, el modelo postula que la distancia horizontal entre ambos hombros es de $\frac{4}{3}u$. Sabiendo lo anterior, se estima la distancia que existe entre ambas muñecas cuando una persona mantiene los brazos extendidos horizontalmente como $6\frac{1}{3}u$.

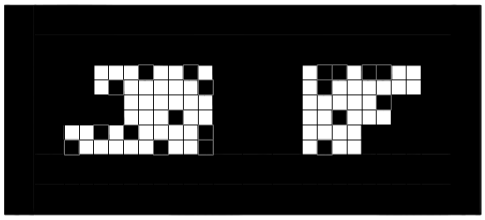
Si usamos el alto de la caja envolvente (h_{bb}) como una estimación de la distancia que existe entre las muñecas y los hombros de la persona, podemos definir que el ancho máximo del área a segmentar (w_{seg}), cuando una persona mantiene los brazos extendidos de forma horizontal, está dada por la Ecuación 4.11.

$$w_{seg} = \frac{6\frac{1}{3}u}{2\frac{1}{2}u} h_{bb} = 2\frac{8}{15} h_{bb} \sim 2\frac{1}{2} h_{bb} \tag{4.11}$$

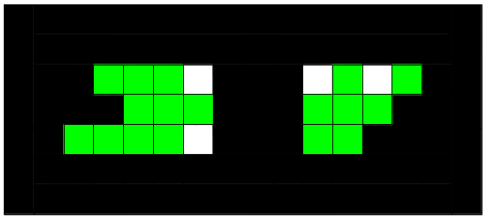
Si estimamos w_{seg} a partir de h_{bb}, conoceremos las dimensiones de un rectángulo con proporciones $2:5$, que inscribe al dorso y el área donde se pueden encontrar los brazos del usuario. Esta subárea B_0 del cuadro de video es de donde se tomará la información para estabilizar la silueta pictórica



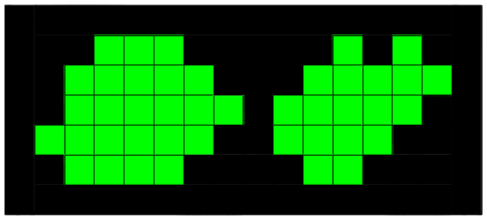
(a) Cuadro de segmentación por color.



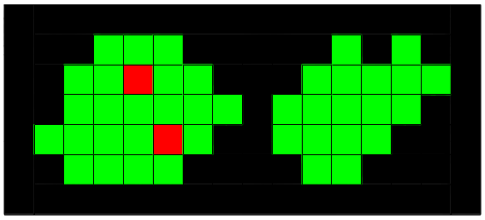
(b) Cuadro de segmentación por tono de gris.



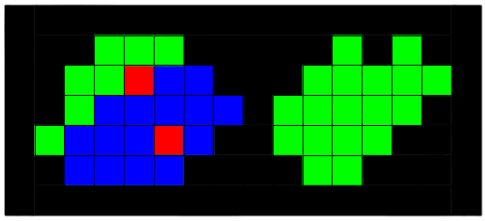
(c) Fusión de 4.12a y 4.12b.



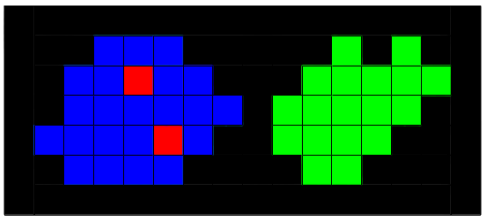
(d) Dilatación de 4.12c.



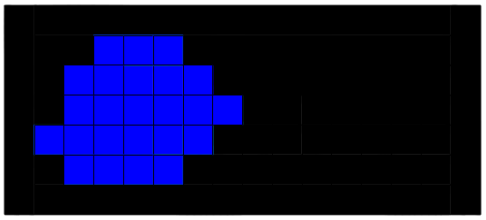
(e) Traslape con el filtro de partículas.



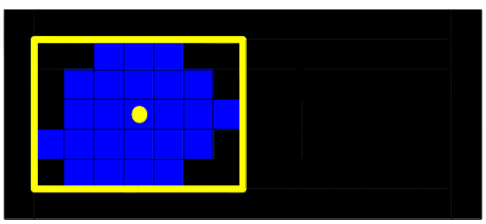
(f) Uso del primer *bush fire*



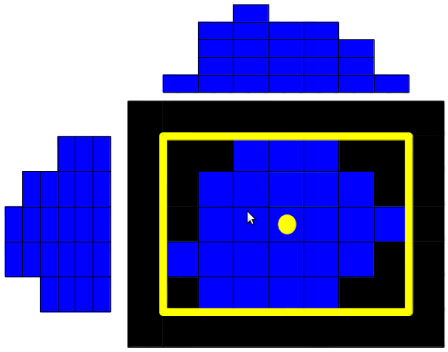
(g) Uso del segundo *bush fire*



(h) Eliminación de los residuos



(i) Centroide y caja envolvente.



(j) Cálculo del histograma horizontal y vertical del área segmentada.

Figura 4.12: Método de fusión de datos

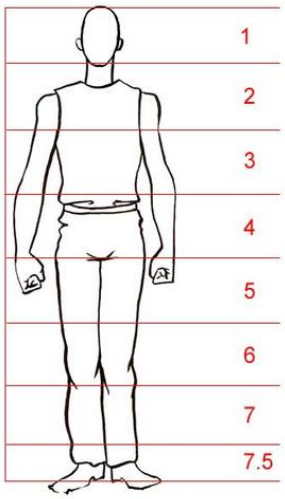


Figura 4.13: Proporciones humanas. 7 cabezas y media de alto.

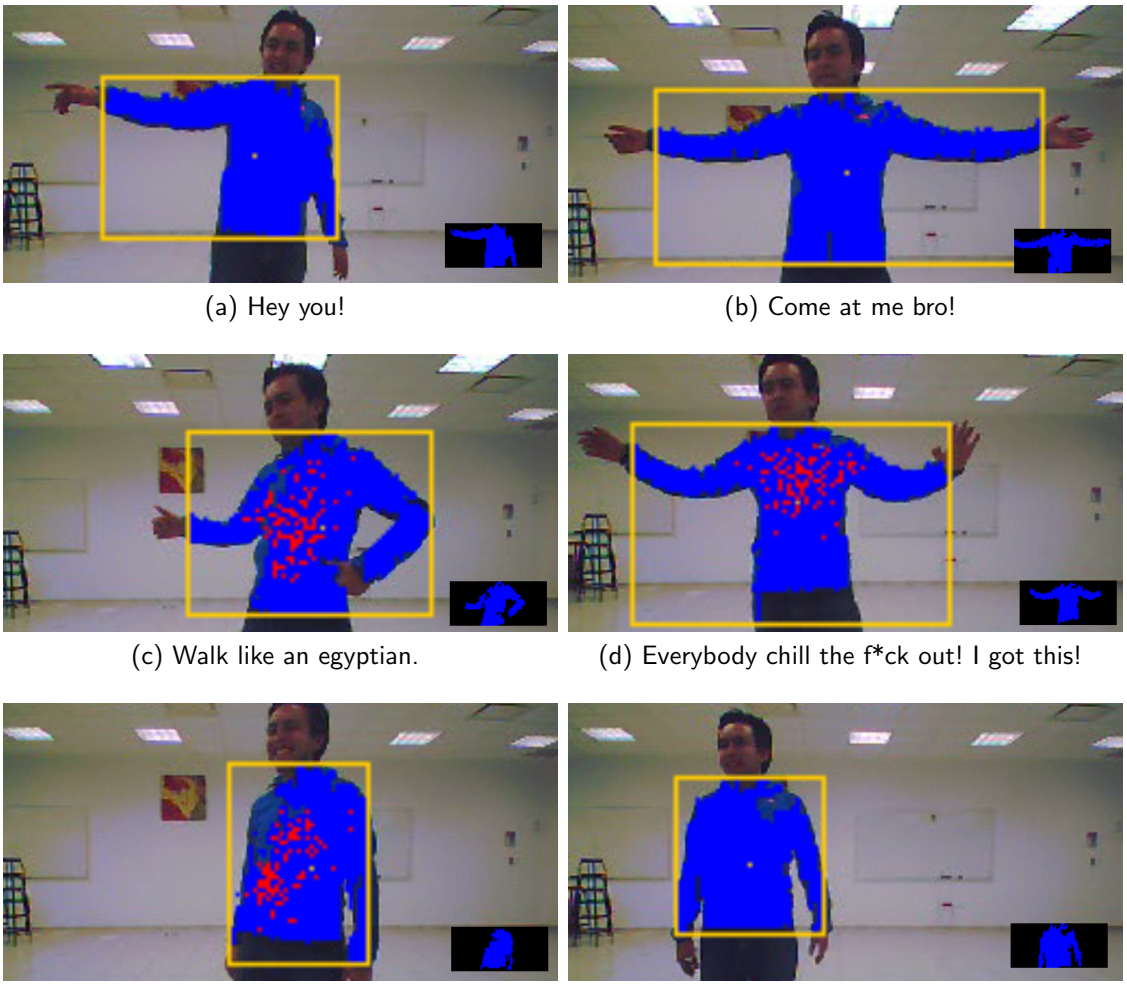
considerando cambios de rotación y traslación.

Para la representación de la silueta pictórica estabilizada, en la GUI se usó un bitmap (B_s) de proporciones $2:5$ con dimensiones h_s y w_s, de alto y ancho, respectivamente. A cada uno de los pixeles de B_s les fue asignado un valor, tomados de su correspondiente en B_0 con el usó la Ecuación 4.12; dicha ecuación contempla los factores de escala y la traslación de la caja envolvente para que coincida con las dimensiones de B_0.

Complementariamente, para corregir el ángulo *roll* con el que fue tomado el cuadro de video y rotar la silueta pictórica alrededor de un pivote $\mathbf{p}$, se usó la Ecuación 4.13. Geométricamente, las coordenadas del pivote están dadas por $p_x = w_s/2$ y $p_y = \frac{c_x}{s_y}$, mientras que el ángulo θ es el reportado por la unidad de medidas inerciales embarcado en el dron.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2\frac{1}{2}h_{bb} & 0 \\ 0 & \frac{h_{bb}}{h_s} \end{pmatrix} \begin{pmatrix} x_s \\ y_s \end{pmatrix} + \begin{pmatrix} c_x - 1\frac{1}{4}h_{bb} \\ y_{bb} \end{pmatrix} \tag{4.12}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.13}$$



(a) Hey you! (b) Come at me bro!

(c) Walk like an egyptian. (d) Everybody chill the f*ck out! I got this!

(e) Too cool to stand straight. (f) Are you talking to me?

Figura 4.14: Segmentación y escalado de la silueta pictórica del usuario.

La principal aportación de la estabilización del área segmentada, es el incremento en la certidumbre del clasificador de la silueta. La Figura 4.14 ilustra varias de las poses segmentadas que la aplicación IHRVANT muestra en el bitmap B_s en la esquina inferior derecha. IHRVANT es capaz de mostrar el resultado del sementado del filtro de partículas y la silueta pictórica simultáneamente.

La Figura 4.15 muestra la detección de dos gestos cuando el cuadricóptero se encuentra inclinado 10° al rededor del eje *roll*. La orientación del brazo del usuario es paralela al horizonte, pero está inclinado con referencia al borde horizontal de la imagen un ángulo proporcional al valor de *roll*. Para clasificar esta silueta pictórica, primero se estabiliza y después se compara contra todas las

referencias. El bitmap estabilizado B_s también se muestra por cada uno de los incisos de la figura, en él, el brazo es paralelo al borde horizontal de la imagen y es clasificado correctamente por IHRVANT.

(a) Gesto 2 detectado. (b) Gesto 3 detectado.

Figura 4.15: Detección de gestos con un ángulo de *Roll*$= 10^{\circ}$. Cuadro estabilizado y barras de detección, en rojo se encuentra la etiqueta del gesto ganador.

4.9 Distancia al usuario

La primera aproximación para estimar la distancia a la que se encuentra el usuario relativa al dron consistió en estudiar la altura de la caja envolvente (h_{bb}), que es proporcional a la distancia d a la que se encuentra el usuario del cuadricóptero. Para obtener una estimación lo más cercana a la realidad se eligieron tres colores distintivos para segmentar la silueta pictórica: rojo, verde y azul; paso seguido, hicimos distintas observaciones ubicando a la persona a una distancia conocida con respecto al dron y registrando la altura en pixeles de su silueta pictórica. La altura de la silueta pictórica con respecto a la distancia del usuario relativa al dron mantienen una relación semejante a una hipérbola y para aproximarla usamos un ajuste polinomial de tercer orden.

El ajuste polinomial que mapea la altura de la caja envolvente con la distancia a la que se encuentra el usuario y las muestras tomadas de distintos cuadros, se ilustra en la Figura 4.16. En la misma figura, el eje horizontal representa la altura en pixeles de la caja envolvente y el eje vertical la distancia estimada en metros entre el dron y el usuario. Por otro lado el polinomio de tercer grado

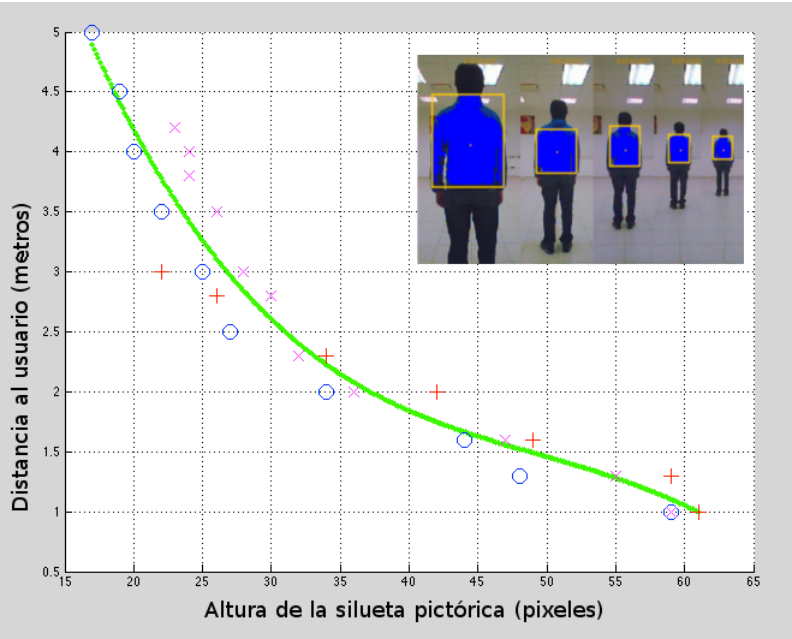


Figura 4.16: Mapeo de la altura de la silueta pictórica a la distancia entre el dron y el usuario. Silueta pictórica segmentando el color rojo ($\circ$), verde ($+$) y azul ($\times$). Con verde en la gráfica se ilustra la curva del ajuste polinomial. A menor la altura de la silueta pictórica, mayor es la distancia al usuario.

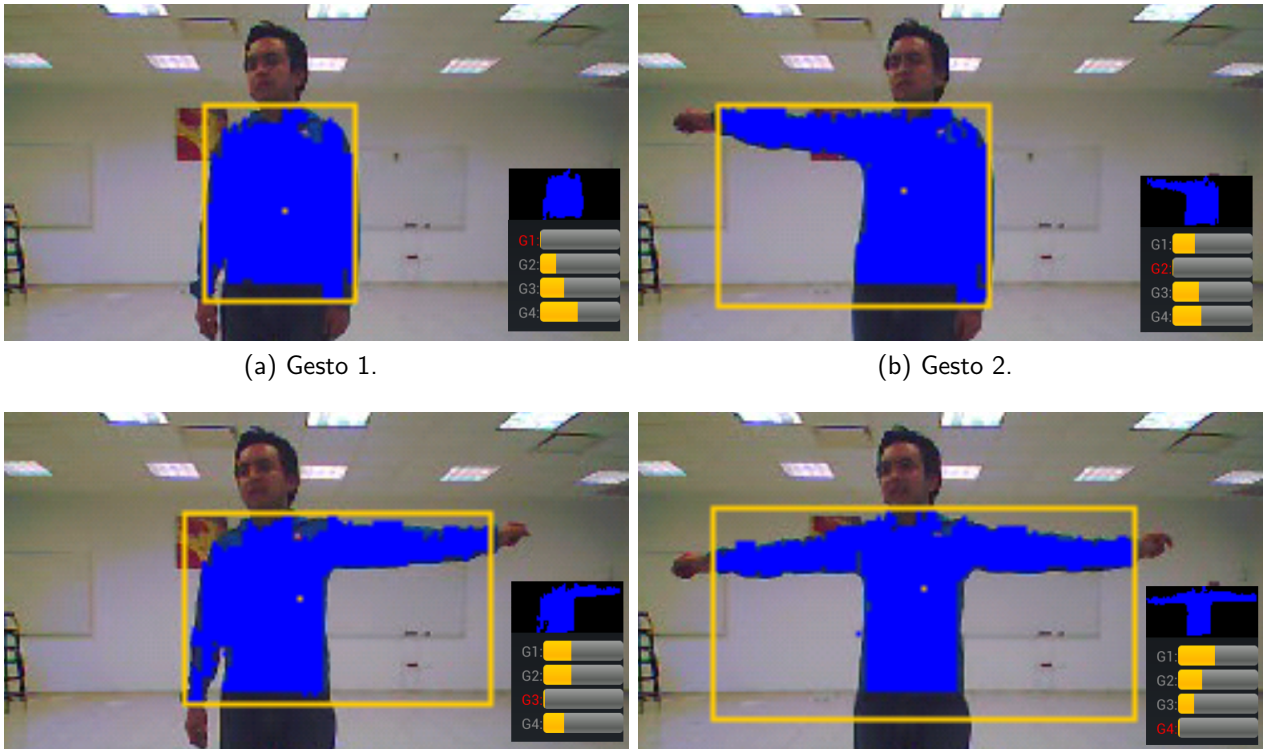
que mejor las ajusta es el de la Ecuación 4.14.

$$d = -7.13 \times 10^{-5} h_{bb}^3 + 1.05 \times 10^{-2} h_{bb}^2 - 5.48 \times 10^{-1} h_{bb} + 11.15 \tag{4.14}$$

4.10 Clasificador

IHRVANT es capaz de usar los dos clasificadores de la silueta pictórica de B_s descritos en la Sección 3.3.3. El primero calcula los momentos de Hu de la silueta pictórica a partir de B_s y mide la distancia euclidiana con los almacenados en los bancos gestuales. El segundo, calcula el error cuadrático entre el histogramas normalizados horizontal y vertical de B_s (ver Figura 4.12) contra los bancos gestuales que existan. IHRVANT realiza el entrenamiento de los clasificadores promediando 31 muestras del área segmentada cada 1.2 segundos; la *tablet* emite un sonido cada vez que toma una muestra, el usuario debe presentar el gesto continuamente frente a la cámara del dron. Es

necesario iniciar el modo edición y elegir el banco a modificar para que el entrenamiento de inicio. Los entrenamientos deben realizarse mientras el dron no este volando. En la Figura 4.17, hay algunos de los resultados del clasificador; en cada inciso, al centro está la segmentación de la silueta pictórica, el bitmap B_s a la derecha y en la esquina inferior derecha la evaluación del gesto actual contra los 4 bancos existentes. IHRVANT representa al gesto ganador cambiando el color a rojo de la etiqueta a la izquierda de cada barra.



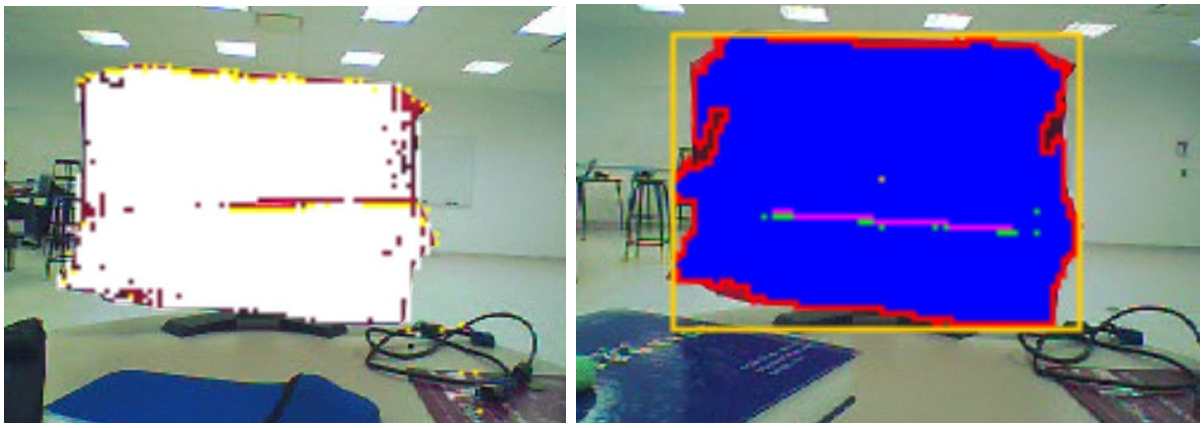
(a) Gesto 1. (b) Gesto 2.

(c) Gesto 3. (d) Gesto 4.

Figura 4.17: Clasificación de las siluetas pictóricas.

4.11 Plano Láser

IHRVANT segmenta el rayo láser usando algunas de las herramientas descritas en el marco teórico (ver Sección 3.3). La Figura 4.1 ilustra como el hilo de segmentación láser toma los resultados de la segmentación de la silueta pictórica que incluye el cálculo del gradiente realizado por el hilo de



(a) Canales de Crominancia, en color amarillo los bordes detectados

(b) Representación del perímetro en color rojo. Los puntos borde detectados en color verde y la línea estimada en color magenta.

Figura 4.18: Cálculo de bordes y estimación de la recta

segmentación por color. Hasta este punto, sólo están presentes los pixeles cuyo gradiente superó un umbral T y están dentro de la silueta pictórica.

El hilo de segmentación por color entrega con un color distintivo todos los puntos que al evaluar la Ecuación 3.24 toman el valor de 1 (ver Figura 4.18a). El hilo de fusión de datos determina el perímetro de la silueta pictórica y dibuja de color verde todos los pixeles que representan algún borde y posiblemente hayan sido generados por la incidencia del láser sobre el objeto de interés. Este mismo hilo entrega las coordenadas de dichos puntos al hilo láser para que sean procesados por RANSAC y estime la recta que mejor ajuste al conjunto de puntos. Paso siguiente, con la recta y el modelo geométrico (Sección 4.11.1) se estima la pose del objeto de interés. En la Figura 4.18b, la línea que mejor ajusta al conjunto de puntos detectados dentro de la silueta pictórica es de color magenta.

El usuario puede realizar la calibración del sensor láser colocando un objeto a una distancia conocida x_c y ajustarla con el control deslizante de "Despliegue y ajuste de x_c". Para después, presionar el botón "Puesto a cero del sensor láser" para que IHRVANT calcule el valor de las variables y_f, z_f y y_L a partir de la última recta estimada por RANSAC. Todas las variables mencionadas están definidas en la siguiente sección. Una vez puesto a cero, el sensor empieza a entregar la orientación y la distancia del objeto de interés a la GUI.

4.11.1 Orientación y distancia del usuario, con respecto al dron

Para estimar las dos variables relativas al usuario (orientación y distancia) con respecto al dron, usaremos un modelo basado en geometría proyectiva (ver Figura 4.19). Para esto, el sensor debe calibrarse con alguna referencia y a partir de dicha referencia estimar un resultado.

El objetivo del modelo geométrico es ubicar un punto tridimensional $I(x', y', z')$, a partir de un punto bidimensional en el campo visual de la cámara $C(u, v)$. El origen del sistema coordenado del punto I se encuentra en la cámara frontal del cuadricóptero. El lugar geométrico de donde se proyecta el láser tiene coordenadas $L(d, h_L, 0)$; el punto $I_L(u_L, v_L)$ representa el lugar de incidencia del láser en el plano de calibración ($\mathbf{N_c}$), que se encuentra a una distancia conocida d_c del dron.

Las dimensiones de u_f y v_f son las dimensiones del campo visual, en metros, de la cámara al momento de poner a cero el sensor (Ecuación 4.15). Estas dimensiones representan el ancho y el alto de la superficie que registra la cámara y que deben ser proporcionalmente divididos entre los pixeles horizontales y verticales de los canales de crominancia. Ambas dimensiones están relacionadas geométricamente con d_c y los ángulos de visión de la cámara: α y θ; que a su vez, están directamente relacionados con el ángulo de apertura del lente de la cámara.

$$v_f = 2x_c \tan\theta \qquad u_f = 2x_c \tan\alpha \tag{4.15}$$

Si el láser incide en un plano N_i que se acerca o se aleja a partir de la distancia de referencia d_c, es posible estimar la distancia a la que se encuentra el plano si y solo si la cámara registra el punto de incidencia del láser en un cuadro de video $I_L(u_L, v_L)$. Como restricción, el modelo propuesto contempla que todos los planos son paralelos entre sí ($\mathbf{N_c} \parallel \mathbf{N_{yz}} \parallel \mathbf{N_i}$). Este modelo geométrico ajusta a una recta la proyección del láser para que con su pendiente (Ecuación 4.16) y las coordenas del punto L, se defina la ecuación de la recta con la cual se pueda estimar x' a partir de v (Ecuación 4.17). Calcular x' representa la primera aproximación para conocer la distancia, en metros, entre el dron y el objeto de interés; pero un punto no es suficiente y por eso emplearemos al menos dos para

después promediarlos y estimar la distancia χ (ver Figura 4.19).

$$m = \frac{y' - h_L}{d_c - d} \quad \text{donde} \quad y' = \frac{v_f}{2} - \frac{v_L \times v_f}{120} \tag{4.16}$$

$$x' = \frac{y' - h_L}{m} + d \tag{4.17}$$

Por otro lado, a la distancia x' de la cámara, el ancho del campo visual en metros puede estimarse con una relación trigonométrica $u_t = 2x' \tan \alpha$. Puesto que u_t debe distribuirse proporcionalmente en los 160 pixeles de ancho, con la Ecuación 4.18 se puede estimar z'.

$$z' = \frac{u_t(x - 80)}{160} \tag{4.18}$$

Con esta metodología un punto dentro del cuadro de la cámara $C(u, v)$ puede ser mapeado a coordenadas tridimensionales $I(x', y', z')$; por lo cual, proponemos usar los dos extremos de la línea ajustada por el algoritmo RANSAC (Sección 3.3.7) para calcularla las coordenas tridimensionales de dos puntos en el espacio $I'_L(x', y', z')$ y $I''_L(x'', y'', z'')$, tal como lo muestra la Figura 4.19. Hipotéticamente estos puntos pertenecen a los extremos del láser proyectado sobre el objeto de interés. Finalmente, conociendo las coordenadas tridimensionales del los dos puntos, I'_L e I''_L, se puede calcular la distancia al objeto de interés y su orientación con las Ecuaciones 4.19.

$$\beta = \arctan\left(\frac{x'' - x'}{z'' - z'}\right) \quad \chi = \frac{x' + x''}{2} \tag{4.19}$$

4.12 Control

Para manipular de forma automática al cuadricóptero en el aire, propusimos un esquema de control como el que se muestra en la Figura 4.20. Se trata de un controlador de lazo cerrado que toma como entrada algunos datos de la GUI, como la altura a la que el usuario definió que sobrevuele el cuadricóptero y el área a segmentar. El controlador, de tipo PID, genera una señal correctiva para que sea enviada al cuadricóptero en forma de comando AT a través de WiFi. El lazo de control se

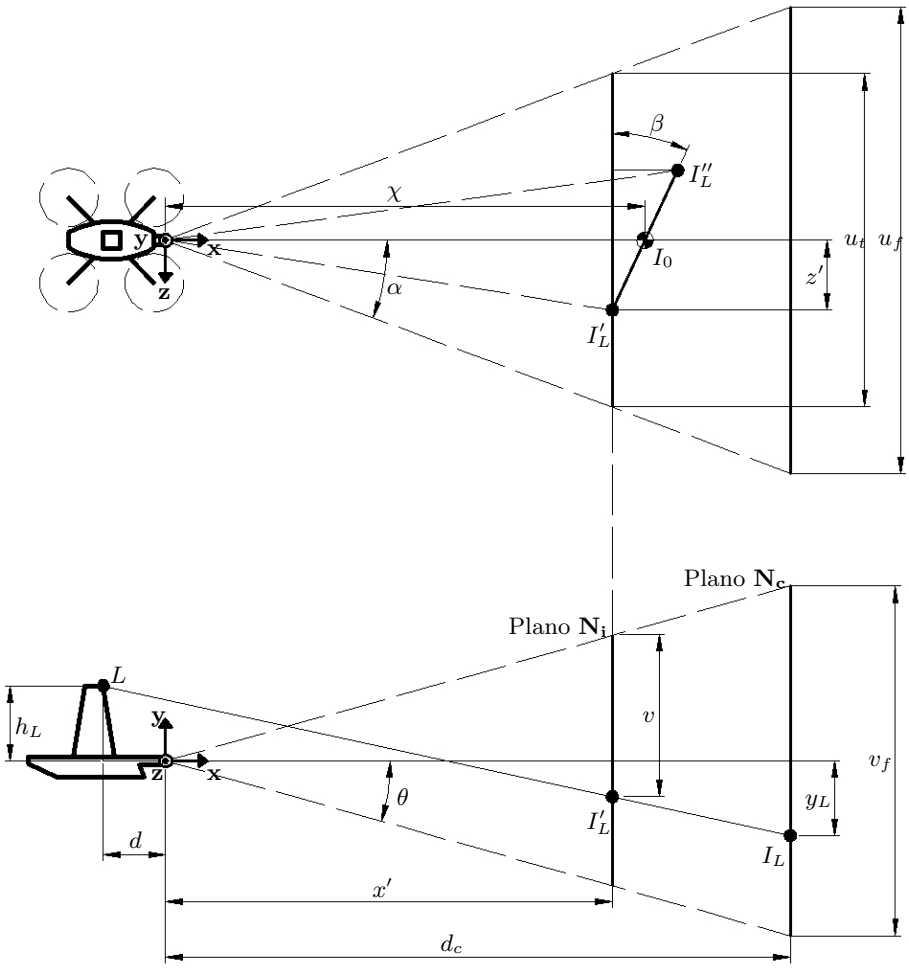


Figura 4.19: Modelo geométrico del láser.

cierra con el procesamiento digital de imágenes que segmenta la silueta pictórica y estima la posición del usuario en el cuadro de video.

El cuadricóptero tiene 4 grados de libertad que definen la postura del dron en el aire; los primeros tres, representan los ángulos con respecto a la horizontal de los tres ejes cartesianos que en aeronáutica se les llama *yaw*, *pitch* y *roll*. El cuarto grado de libertad consiste en la altura del cuadricóptero con respecto al suelo.

Para modificar el valor de algunos de los cuatro grados de libertad enviamos un comando de control al cuadricóptero definiendo el movimiento a realizar con el valor de 4 variables distintas

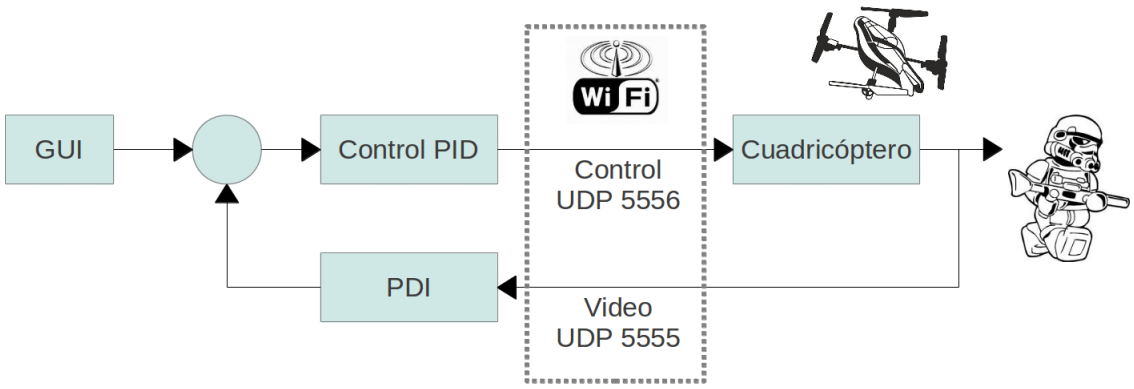


Figura 4.20: Control PID

cuyos rangos válidos son todos $[-1, +1]$. Pese a esto, IHRVANT no envía comandos de control con algunas de las variables fuera del rango $[-0.2, 0.2]$, pues probó ser suficiente y hace menos probable que la aeronave, en control automático y por algún mal funcionamiento, sufra algún daño.

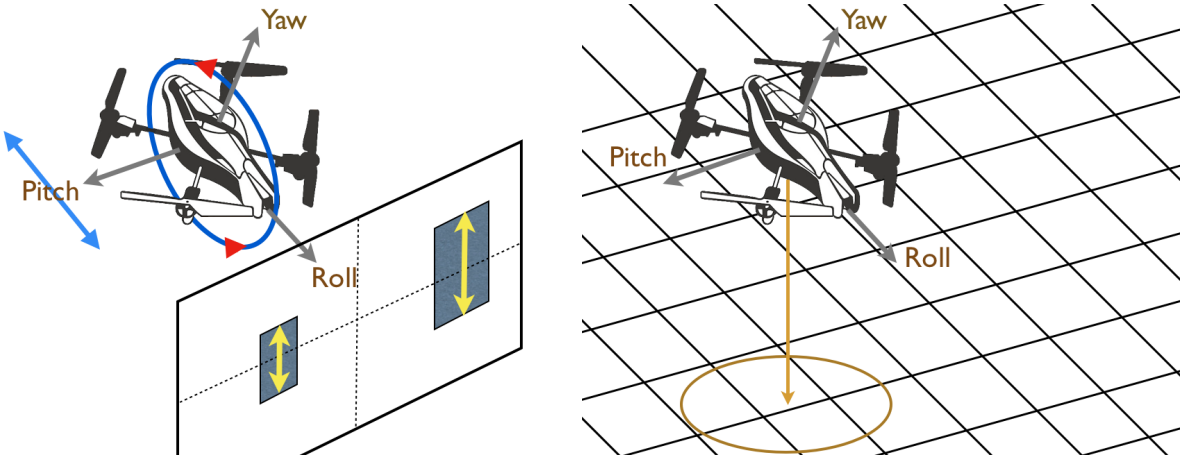
Estimando continuamente la ubicación del usuario en el cuadro de video a través de la posición y las dimensiones de la silueta pictórica del usuario, obtenidas por medio del procesamiento digital de imágenes, generamos una acción de control para cada uno de los cuatro grados de libertad con un esquema como el que se ilustra en la Figura 4.20; con esto logramos dar seguimiento al usuario al mismo tiempo que se clasifica su silueta pictórica. Cada uno de los controladores y su funcionamiento se describirán en las siguientes subsecciones.

El control para cada una de las 4 variables de movimiento del dron puede ser habilitado o deshabilitado en tiempo de vuelo a través de la interfaz de IHRVANT. Por otro lado, cada controlador fue sintonizado de manera empírica.

4.12.1 Altura de vuelo

La altitud del vuelo la recibe la plataforma de procesamiento a través del puerto UDP que envía los datos de navegación. El usuario puede elegir la altitud deseada con el control deslizable en la interfaz y la diferencia con la altitud de vuelo definirá el error que será la entrada al controlador.

El dron cuenta con un sensor ultrasónico en la parte inferior que le permite conocer la distancia



(a) Control *pitch*. (b) Control de altura de vuelo.

Figura 4.21: Control del ángulo *pitch* y altura de vuelo del cuadricóptero.

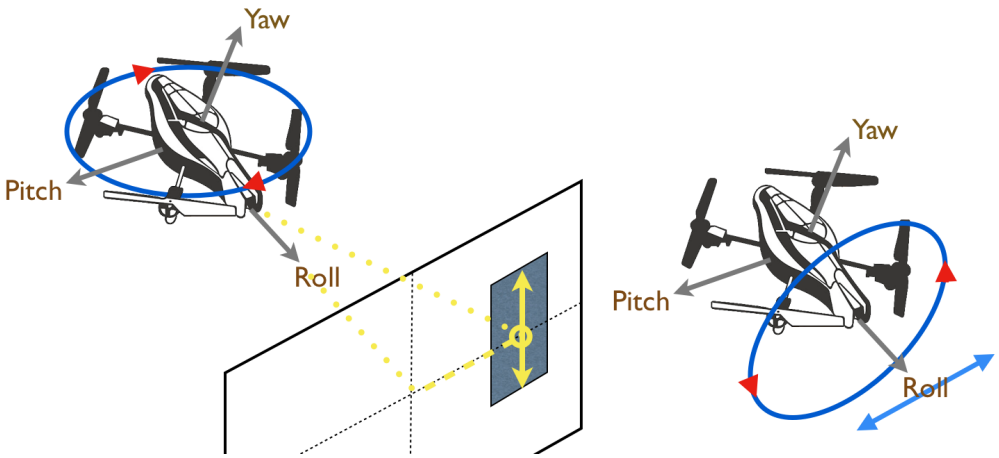
a la que se encuentra volando sobre el nivel del suelo. Este valor se mantiene constante, asumiendo que el usuario no subirá o bajará por alguna escalera. El modelo para lograr esto, es un control proporcional que tome como entrada la altitud de vuelo del dron y envíe una acción correctiva en forma de comando AT.

Si definimos a h_o como la altura objetivo a la que debe volar el cuadricóptero y a h_d como la altura estimada a la que vuela el cuadricóptero, se define el error como la diferencia que existe entre ambas variables $e = h_o - h_d$. La acción correctiva C será proporcional a e multiplicado por una constante K_p conocida como ganancia proporcional $c = K_p \times e$. El valor c puede ser escalado a un rango de valores que el comando de vuelo admita, para entonces formar parte directamente de un comando de control.

Una vez activado el control automático de altura, el cuadricóptero mantiene una altura de vuelo constante e igual a la asignada en la interfaz del usuario.

4.12.2 Pitch

Al cambiar el ángulo *pitch* el cuadricóptero avanzará o retrocederá durante el vuelo (ver Figura 4.21a). Esta acción hace que el cuadricóptero se aleje o se acerque a la persona a partir de la distancia estimada entre ambos.



(a) Control *Yaw*. (b) Control *Roll*

Figura 4.22: Control del ángulo *Yaw* y *Roll* del cuadricóptero.

La segmentación de la silueta pictórica y obtención de algunas de sus características como las dimensiones de la caja envolvente (h_{bb}) son usadas con el modelo definido por la Ecuación 4.14 para estimar la distancia a la que se encuentra el usuario del dron. El rango de la distancia objetivo a mantener entre el cuadricóptero y la persona es $[1.0m, 1.5m]$ y el usuario no lo puede modificar por razones de seguridad.

El error que sirve de entrada para este controlador es la diferencia entre un valor de referencia y la distancia entre el usuario y el cuadricóptero estimada.

4.12.3 Yaw

Dinámicamente, al variar el valor del ángulo *yaw*, el cuadricóptero gira a la derecha o a la izquierda manteniéndose estático en el plano horizontal. Lo que se tomó como entrada a este controlador es la distancia horizontal que existe, en pixeles, entre el centro del cuadro del video y la ubicación del centroide de la silueta pictórica. Un error nulo y por lo tanto una acción de control nula, sucedería si el centroide de la silueta pictórica se encuentra justo a la mitad del ancho del cuadro de video.

La Figura 4.22a ilustra la distancia horizontal entre el centro del cuadro del video y el centroide de la silueta pictórica. Una vez activado el control automático, IHRVANT tratará de mantener el centroide de la silueta pictórica al centro del cuadro de video.

4.12.4 Roll

La manipulación del ángulo *roll* implica un movimiento lateral a la derecha o a la izquierda del cuadricóptero (ver Figura 4.22b). La acción combinada de los controladores *yaw* y *pitch* son suficientes para que la aeronave siga al usuario, por lo que para nuestra aplicación no es necesario ejercer algún control sobre esta variable.

4.13 Modo de uso de IHRVANT

El usuario debe inicializar su rastreo para que IHRVANT empiece la segmentación y seguimiento a través del flujo de video. Con IHRVANT desplegando el flujo de video, el usuario debe:

1. Ajustar la altura de vuelo a la que debe volar el dron con el *slider* de control de altitud.

2. Hacer visible los resultados de PDI.

3. Seleccionar los procesos, uno a la vez, a editar: Segmentación por color o por tono de gris.

4. Iniciar el modo de edición y con su dedo seleccionar el color o tono de gris colocando su dedo sobre la ventana del flujo de video.

5. Iniciar el modo de edición del filtro de partículas y recorrer con su dedo el área con el tono de color a rastrear.

6. Saliendo del modo de edición del Filtro de partículas, hacer visible el resultado: "FP+SC". IHRVANT desplegará el resultado de la segmentación sobrepuesto al flujo de video.

7. Asegurarse de que la tarjeta IOIO está conectada a la fuente de alimentación y prender el láser.

8. Colocar un objeto frente al dron a una distancia conocida y asignar dicha distancia con el *slider* de ajuste de x_c; asegurarse de que IHRVANT segmenta el láser y poner a cero el sensor.

9. Seleccionar uno de los dos clasificadores disponibles y colocarse dentro del flujo de video. IHRVANT desplegará la silueta pictórica del usuario si la segmenta correctamente en la esquina inferior derecha de la interfaz.

10. Iniciada la edición de gestos, presentar el gesto frente a la cámara y elegir uno de los cuatro bancos para guardarlo. IHRVANT emitirá 31 sonidos indicando el muestreo del gesto presentado. La operación se realiza 31 veces para obtener una muestra representativa de los descriptores del gesto, esto concuerda con los postulado por el teorema del límite central.

11. La referencia para el clasificador se promedia tantas veces como se realice el entrenamiento del paso anterior. Si uno de los bancos está ocupado, IHRVANT borrará el gesto para almacenar el nuevo.

12. Despegar el cuadricóptero.

13. Encender controles automáticos para YAW, PITCH y altura.

14. La interfaz, a través de IHRVANT, empezará su funcionamiento normal.

4.14 Conclusiones

Este capítulo describió las diferentes estrategias técnicas usadas para que IHRVANT logre los objetivos planteados en este trabajo de tesis. Paralelamente, mostró algunos resultados parciales de dichas estrategias y como son desplegados en la GUI de IHRVANT.

Pese a las limitaciones de procesamiento intrínsecas en la plataforma de procesamiento móvil, la unión sinérgica del decodificador con los demás algoritmos de segmentación, en hilos independientes, probaron ser un enfoque eficaz en este tipo de aplicaciones. En el próximo capítulo, se presentan algunas estadísticas de tiempo de procesamiento de IHRVANT que complementan esta conclusión.

Hubo detalles del decodificador de video que no habían sido tomados en cuenta al inicio del presente trabajo, los cuales debieron ser analizados a profundidad el desarrollo de la aplicación. Las

estrategias adoptadas rindieron frutos que repercutieron directamente en la velocidad de procesamiento. Por mencionar algunos: el submuestreo de los canales cromáticos, la adición de un sesgo para el cálculo de los histogramas, búfers estáticos y la estabilización del área segmentada.

El procesamiento digital de imágenes logra detectar el láser y a partir de él, se estima la pose del usuario. En el capítulo siguiente hablaremos de la caracterización de este sensor. A través de estrategias de PDI, mostramos que es posible estimar la pose del usuario y su posición relativa al cuadricóptero procesando el flujo de video en un dispositivo móvil y, como se mostrará en el siguiente capítulo, establecer un control funcional sobre el cuadricóptero basado en esta información.

5

Resultados

Este capítulo describe los resultados de este trabajo de tesis y emplea distintos enfoques para el analizar por separado cada elemento del que está compuesta la interfaz. Más adelante en este capítulo y en una condición normal de vuelo, analizaremos los resultados de la operación de todos los componentes funcionando en conjunto de forma cualitativa y cuantitativa. Los datos expuestos en este capítulo, constatarán que esta interfaz es funcional.

5.1 Clasificación de gestos

Como se mencionó en la Sección 4.8, el clasificador toma como entrada el bitmap B_s que resultó de la segmentación de la silueta pictórica. IHRVANT es capaz de usar un clasificador a la vez, de dos posibles. El primero calcula los momentos de Hu y los compara con alguna referencia; el segundo usa los histogramas horizontal y vertical de B_s para medir una similitud con una referencia. El gesto 1 simula la pose del usuario al mantenerse estático o al caminar, los gestos 2 y 3 consisten en mantener un sólo brazo de forma horizontal; el cuarto gesto consiste en mantener ambos brazos de forma horizontal, estos gestos se muestran en la Figura 5.1.

(a) Gesto 1 (b) Gesto 2 (c) Gesto 3 (d) Gesto 4

Figura 5.1: Gestos

En modo edición, IHRVANT puede entrenar el clasificador promediando 31 muestras de los histogramas y momentos de Hu cada 1.2 segundos para posteriormente almacenarlo en uno de los 4 bancos disponibles. IHRVANT considera como gesto válido todo aquel que haya sido clasificado durante 2 segundos continuos. Por lo tanto, el clasificador en turno debe clasificar correctamente un gesto durante 2 segundos para que IHRVANT actúe en consecuencia. Cualquier variación en la clasificación hace que el conteo de los 2 segundos reinicie.

5.1.1 Montaje del experimento

Para comprobar que el clasificador realice su tarea, se montó al cuadricóptero sobre una plataforma giratoria con el objetivo de emular las condiciones de vuelo. El dron, al girar, captura al usuario en distintas escenas mientras IHRVANT clasifica su silueta pictórica. Adicionalmente, el cuadricóptero fue inclinado para medir la robustez del clasificador al cambio en el ángulo de inclinación $roll$ del cuadricóptero. Las condiciones de luz fueron controladas y sólo había luz fluorescente en la sala.

En la Figura 5.2 se representa gráficamente el montaje del experimento. Al centro, el cuadricóptero gira sobre una plataforma y se detiene en distintas posiciones para que el usuario presente alguno de los cuatro gestos que IHRVANT soporta.

Con los clasificadores entrenados y el cuadricóptero girando, el usuario presentó 31 veces cada uno de los cuatro gestos con $roll = 0°$ (para un total de 124 gestos). Después se aumentó el ángulo de inclinación en 5°, por cada vez que el usuario presentaba los 4 gestos 31 veces, hasta que $roll = 10°$. En la Tabla 5.1 están los resultados de clasificar los cuatro gestos de la interfaz con $roll = 0°$. Un gesto presentado (g) puede ser (✓) o no ser ($\times$) detectado por IHRVANT, esto también aplica para un gesto que no sea presentado ($\bar{g}$). Un gesto g correctamente detectado cuenta como un VP (Verdadero Positivo), los gestos presentados que no sean detectados cuentan como FN

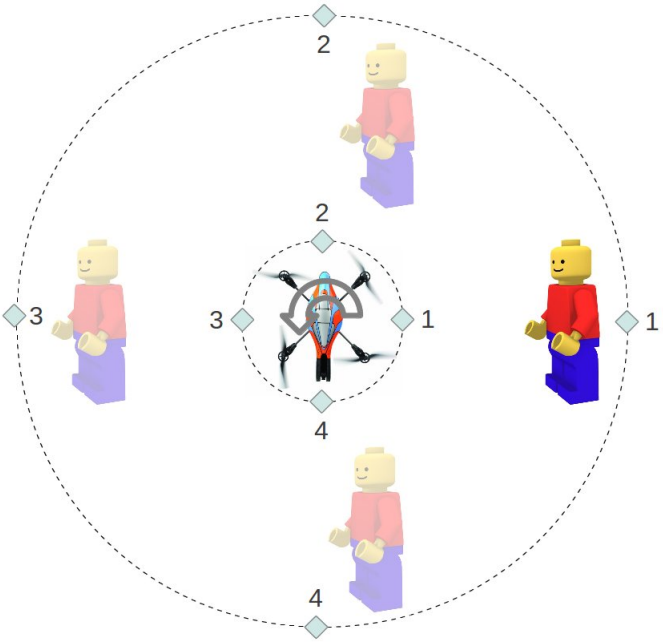


Figura 5.2: Experimento, el Dron gira al rededor del eje *Yaw* mientras el usuario le presenta gestos.

(Falsos Negativos). Un gesto $\bar{g}$ que sea detectado cuenta como FP (Falso Positivo) y el que no sea detectado cuenta como VN (Verdadero Negativo).

Las Tablas 5.2 y 5.3 muestran los resultados para otras variaciones del ángulo $roll$; la Tabla 5.4 ilustra los resultados en conjunto. Suman en total 372 pruebas, presentando 4 gestos y con tres distintos ángulos de inclinación $roll$.

En estadística, la sensibilidad, especificidad y precisión relacionan las cantidades VP, VN, FV y FN. La Tabla 5.5 contiene, en porcentajes, los valores calculados para las distintas variaciones de $roll$ y en conjunto. La sensibilidad es la más afectada por variar el ángulo $roll$; la especificidad y precisión permanecen casi invariantes. Al realizar las pruebas, la causa principal del fallo del clasificador fue la incorrecta segmentación de la silueta pictórica.

	g	$\bar{g}$
✓	31	0
×	0	93

(a) Gesto 1

	g	$\bar{g}$
✓	31	0
×	0	93

(b) Gesto 2

	g	$\bar{g}$
✓	30	0
×	1	93

(c) Gesto 3

	g	$\bar{g}$
✓	26	0
×	5	93

(d) Gesto 4

Tabla 5.1: Resultados de la clasificación de 4 gestos $roll = 0°$.

	g	$\bar{g}$
✓	31	1
×	0	93

(a) Gesto 1

	g	$\bar{g}$
✓	29	0
×	2	93

(b) Gesto 2

	g	$\bar{g}$
✓	26	0
×	5	93

(c) Gesto 3

	g	$\bar{g}$
✓	29	0
×	1	93

(d) Gesto 4

Tabla 5.2: Resultados de la clasificación de 4 gestos $roll = 5°$.

	g	$\bar{g}$
✓	29	0
×	2	93

(a) Gesto 1

	g	$\bar{g}$
✓	28	0
×	3	93

(b) Gesto 2

	g	$\bar{g}$
✓	25	0
×	6	93

(c) Gesto 3

	g	$\bar{g}$
✓	27	0
×	4	93

(d) Gesto 4

Tabla 5.3: Resultados de la clasificación de 4 gestos $roll = 10°$.

	g	$\bar{g}$
✓	91	1
×	2	279

(a) Gesto 1

	g	$\bar{g}$
✓	88	0
×	5	279

(b) Gesto 2

	g	$\bar{g}$
✓	81	0
×	12	279

(c) Gesto 3

	g	$\bar{g}$
✓	82	0
×	9	279

(d) Gesto 4

Tabla 5.4: Resultados de la clasificación de 4 gestos en total.

	Sensibilidad				**Especificidad**				**Precisión**			
$roll$	0°	5°	10°	Total	0°	5°	10°	Total	0°	5°	10°	Total
Gesto 1	100	100	93	97	100	98	100	99	100	96	100	98
Gesto 2	100	93	90	94	100	100	100	100	100	100	100	100
Gesto 3	96	83	80	87	100	100	100	100	100	100	100	100
Gesto 4	83	96	87	90	100	100	100	100	100	100	100	100

Tabla 5.5: Estadísiticas del clasificador (%).

La inestabilidad del clasificador con momentos de Hu hizo imposible que IHRVANT diera como detectado algún gesto en periodos de 2 segundos. Por lo cual, no fue posible presentar ningún resultado de la misma naturaleza al de clasificación por histogramas.

5.2 Sensor láser

En esta Sección se describen los actividades realizadas para montar el láser en el dron y cuales fueron los resultados que derivaron de esta actividad.

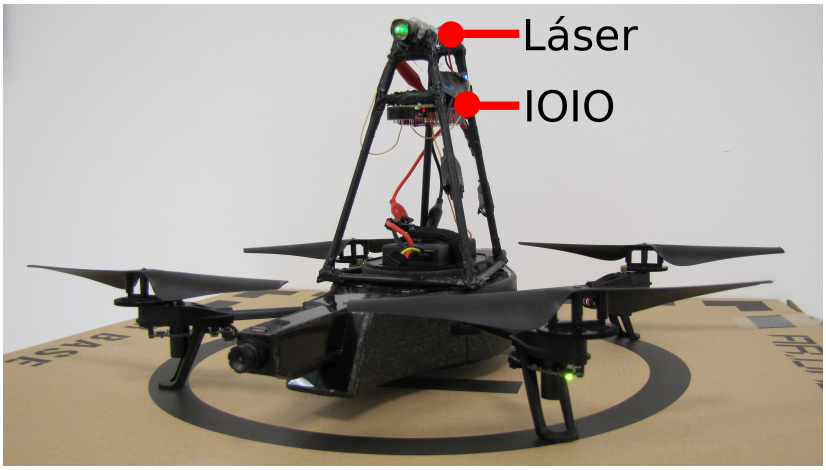


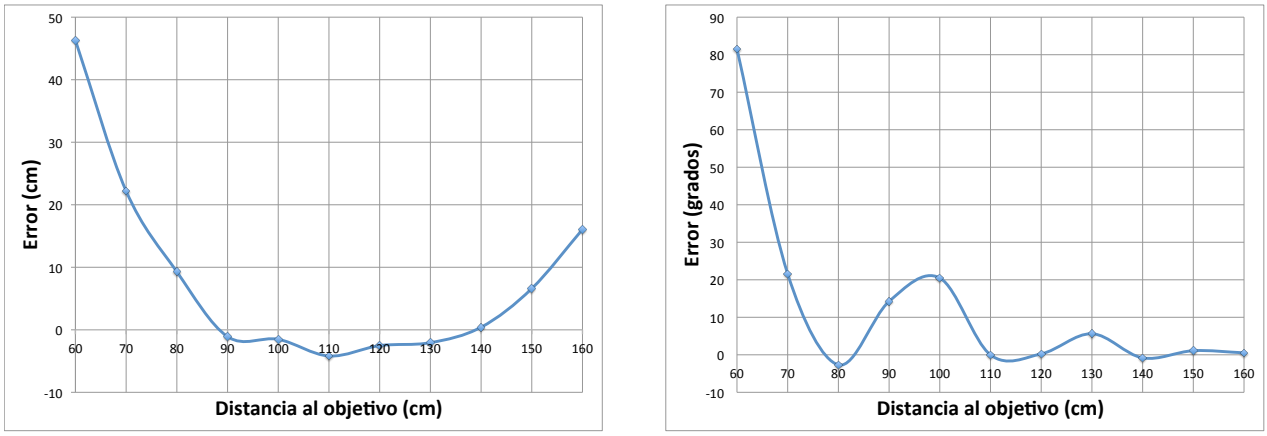
Figura 5.3: Montaje del láser en el dron.

5.2.1 Montaje del láser

Tomando como referencia la Figura 4.11, el láser se montó en las coordenadas $d = -15.5cm$ y $h_L = 15.5cm$; justo por encima de la cámara frontal del dron. Se usó una estructura liviana, hecha de madera, para fijar la tarjeta IOIO y el láser en la parte superior del dron como se muestra en la Figura 5.3. El lente cilíndrico que está fijo al diodo láser, diverge la luz para proyectar una línea sobre el objeto en el que incida el haz de luz.

La tarjeta IOIO cuenta con un adaptador bluetooth para comunicarse con el dispositivo móvil; se alimenta de la batería principal del dron a través de un par de conexiones caimán. El pin número 10 de la tarjeta IOIO opera como salida digital para activar o desactivar el láser por medio de un MOSFET que actúa como switch.

Desafortunadamente, el dron no fue capaz de elevarse en el aire junto con la estructura donde se encontraba el láser. Por lo cual, los resultados que aquí se muestran fueron obtenidos con el dron en tierra y variando la distancia del objeto de interés al dron. La estructura pesa 106 gramos y junto con la altura, ejercían un par de torsión en el cuerpo del cuadricóptero que no pudo compensar.



(a) Distancia del objetivo Vs error.

(b) Distancia del objetivo Vs Error.

Figura 5.4: Errores del sensor láser al estimar la distancia a un obetivo con una inclinación de 0°

5.2.2 Caracterización del láser

El sensor láser fue caracterizado para conocer la certidumbre de la posición estimada del usuario con respecto al dron. Para la segmentación del láser usamos $T = 18$ como umbral para determinar si un pixel representa un incremento signficativo en el gradiente vertical (Ecuación 3.28). De estos pixeles, no todos entraban directamente al algoritmo RANSAC. Escogimos sólo los pixeles que se encontraban dentro de la silueta pictórica y alejados de la periferia 8 unidades en distancia Chamfer (Sección 3.3.5). Estos pixeles, hipotéticamente, representan la proyección del láser en el objeto de interés. Sólo resta ajustarlos con el algoritmo RANSAC, para el cual usamos $K = 50$ iteraciones y 3 pixeles como distancia umbral para determinar si un pixel pertenecía al conjunto de *inliers* del consenso. Sólo los consensos que incluían al menos a un 40 % ($r = 0.4$) del total de los puntos era ajustado por mínimos cuadrados (Sección 3.3.7). La prueba se realizó acercando el objeto de interés 10 centímetros a la vez y anotando la medición, esta operación se realizó 31 veces para poder obtener un muestra estadística significativa del funcionamiento del láser.

En la Figura 5.4a se puede observar que la estimación al objetivo es suficientemente buena cuando el objeto de interés se encuentra a una distancia de entre 90 y 140 cms. De la Figura 4.19, se puede inferir que a menor distancia entre el objeto de interés y el dron, menor será . Para

	Distancia		Orientación	
rango	$\bar{e}$	σ_e	$\bar{e}$	σ_e
$[90-140]$	-1.8302	1.5339	6.6222	8.8465
$[60-160]$	8.1315	15.2568	12.8912	24.3988

Tabla 5.6: Estadísticas del sensor láser.

distancias mayores a 140 cms, IHRVANT no lograba segmentar el láser con la misma regularidad que a distancias menores. A distancias menores de los 90 cms, la deformación de la lente impedía que se estimara correctamente la distancia al dron. De forma complementaria, en la Figura 5.4b se puede observar el error al estimar la orientación del objeto de interés con una posición de 0°. Las estadísticas de la Tabla 5.6 se muestran en dos rangos, para el rango $[90-140]$ donde el sensor láser es más preciso y exacto y el rango completo de la prueba.

5.3 Rastreo del usuario

La interfaz hombre-máquina debe rastrear al usuario a través del flujo de video para que se considere funcional. Graficaremos el error con respecto al tiempo que sirve de entrada a los controladores de los distintos grados de libertad del cuadricóptero y observaremos como IHRVANT intenta minimizar el error. Esta acción implica que el cuadricóptero mantuvo constantemente al usuario al centro del cuadro de video y que el usuario fue rastreado exitosamente. Por otro lado, hablaremos de las situaciones en las que el procesamiento digital de imágenes falla al seguir el usuario.

Hablando de los parámetros empleados en cada uno de los algoritmos, para la selección del color IHRVANT usa una campana gausiana bidimensional con $L = 21$ y un umbral inicial de $D^2 = 13^2$ (Sección 3.3.1). El usuario puede cambiar el valor D^2 mediante la interfaz táctil de IHRVANT (Sección 4.4). El filtro de partículas (Sección 4.6.2) opera con: $M = 200$ partículas, $N^2 = 4^2$ número de *buckets*, $w = 11$ pixeles de vecindad para el cálculo de $\mathbf{q_t}(\mathbf{x})$ y $\Sigma = 2$ como varianza en el modelo dinámico de segundo orden. Todos estos parámetros se determinaron experimentalmente.

IHRVANT, gracias a la segmentación de la silueta pictórica, soporta ligeros cambios de iluminación mediante el reajuste del tono de gris a segmentar. El ajuste lo realiza muestreando la vecindad

al rededor del centroide de la silueta pictórica.

La colección de gráficas de la Figura 5.5 representa la bitácora de vuelo de IHRVANT, durante una de las pruebas que se realizó con la interfaz. La prueba se realizó en un espacio cerrado con iluminación fluorescente. Cada vez que la interfaz es puesta en marcha, todos los datos generados se almacenan en un archivo de formato CSV[1]. Específicamente, la Figura 5.5a ilustra la detección de gestos a lo largo del tiempo, mientras la batería proveyó la energía suficiente (ver Figura 5.5b). Los demás incisos de la Figura (5.5c, 5.5d, 5.5e) representan el error estimado de cada uno de los grados de libertad que IHRVANT trata de minimizar.

La mejor labor la realiza el controlador *Gaz* que durante toda la prueba mantuvo al dron estable a una altura de 1 metro sobre el nivel del suelo (ver Figura 5.5e). En el costado izquierdo de la gráfica, con un círculo punteado, se ilustra el encendido del control automático y como a partir de ese momento el error se reduce y se mantiene estable durante la operación de la interfaz. De forma similar, el controlador del eje *Yaw* actúa para mantener al usuario en el centro del cuadro de video como se muestra en la Figura 5.5c. El ángulo *pitch* es el más disperso de todos. Los cambios de iluminación en la chamarra que porta el usuario impacta directamente en la calidad de la segmentación y por ende, la estimación de la distancia al usuario es muy variante.

El dron despega y aterriza con el comando 2. Durante el vuelo, IHRVANT detecta acertada y repetidamente el gesto 1 (brazos pegados al cuerpo). Esto se debe a que, por los cambios de iluminación, la silueta pictórica no es segmentada correctamente y no puede clasificarse como alguno de los 4 gestos. Al mejorar la segmentación, el gesto 1 vuelve a detectarse y así sucede en distintas ocasiones (ver Figura 5.5a).

Para los ejes de movimiento *gaz* y *Yaw* se emplearon controladores de tipo proporcional con ganancias definidas experimentalmente; para *gaz* se usó $K_p = 0.8$ y para *yaw* $K_p = 0.012$. El lazo de control se refresca con un período de $120ms$, es decir, cada vez que se reciben 2 paquetes de datos de navegación. La entrada al controlador *Pitch* es muy susceptible a variaciones en la calidad de la segmentación, puesto que se calcula restando un valor de referencia y la distancia estimada al usuario (Sección 4.9). Por esta razón, el eje de movimiento *Pitch* cuenta con un controlador del

[1]CSV: Comma Separeted Values.

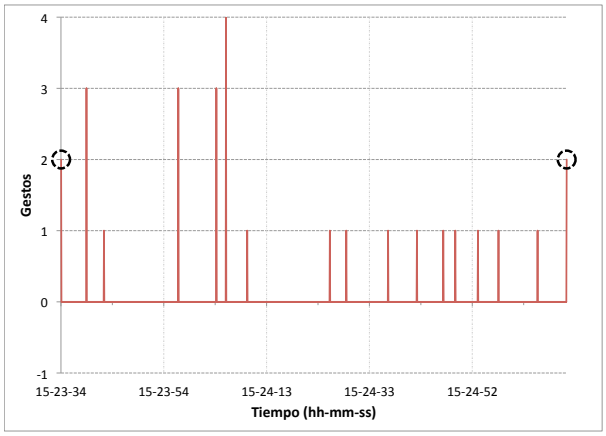
tipo Proporcional-Derivativo y sus ganancias, también definidas experimentalmente, son $K_p = 0.30$ y $K_d = 0.03$. El controlador derivativo, por oponerse al cambio del error, absorve variaciones de la distancia al usuario estimada que pudieran volver inestable el vuelo del dron.

5.4 Rendimiento de IHRVANT en Android

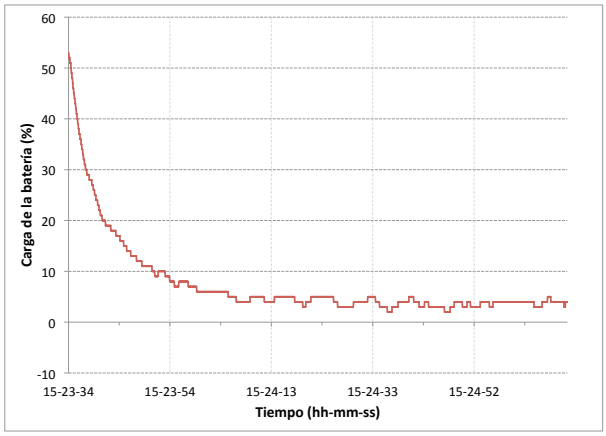
La plataforma de procesamiento empleada es la *tablet Transformer Prime* TF201 que cuenta con el procesador Tegra 3 de Nvidia y 1 GB de memoria RAM. Ejecuta como sistema operativo Android v4.0.3 y su pantalla es de 11 pulgadas. La plataforma soporta tres modos de rendimiento: *power saving*, *balanced* y *performance*; que van relacionados directamente con la rapidez de procesamiento de la *tablet*. En cada apartado de la Tabla 5.7 los acrónimos empleados refieren a los hilos de procesamiento siguientes: **SC** segmentación por color (Sección 4.4), **FP** filtro de partículas (Sección 4.6.2), **FP+SC** fusión de datos del filtro de partículas y la segmentación por color (Sección 4.7.1)y por último **Luma** segmentación por tono de gris (Sección 4.5).

5.5 Situaciones de fallo

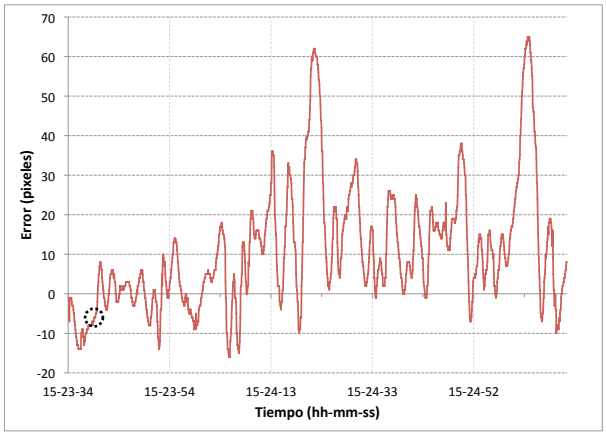
La velocidad de avance del usuario y los cambios de iluminación representan los dos problemas más grandes en el proceso de segmentación de la silueta pictórica. Si el usuario avanza demasiado rápido IHRVANT no es capaz de rastrearlo y un cambio de iluminación puede resultar en un fallo de la segmentación por color. Con respecto a los cambios de iluminación, el codificador a bordo del dron cuenta con una característica denominada "adaptive video" que compensa los cambios de iluminación en el flujo de video de manera automática y su acción intempestiva puede provocar que la segmentación por color falle. Cabe mencionar que Parrot no provee soporte para modificar el *firmware* del cuadricóptero y no es posible que IHRVANT actúe para compensar la acción del "adaptive video".



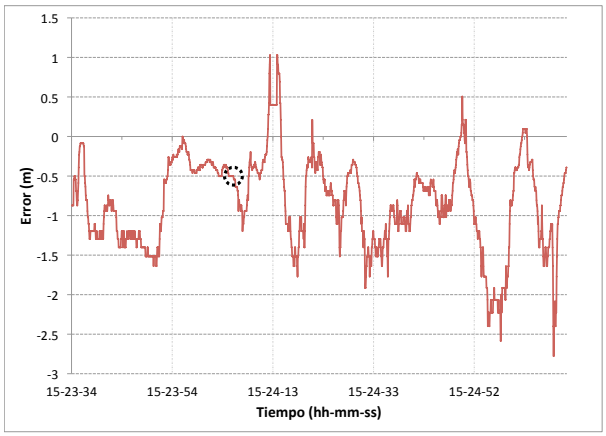
(a) Gestos



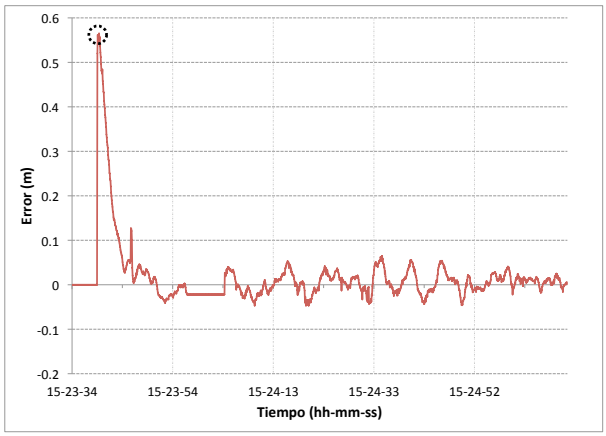
(b) Batería



(c) Yaw



(d) Pitch



(e) Gaz

Figura 5.5: Rastreo del usuario

	SC	FP	FP+SC	Luma
Tiempo mínimo	0.042	0.065	0.010	0.030
Tiempo máximo	0.531	0.211	0.070	0.530
Tiempo promedio	0.108	0.120	0.030	0.090
FPS	9.240	4.170	9.160	9.100

(a) Modo *power saving*.

	SC	FP	FP+SC	Luma
Tiempo mínimo	0.020	0.034	0.010	0.010
Tiempo máximo	0.245	0.155	0.050	0.250
Tiempo promedio	0.065	0.060	0.010	0.060
FPS	13.90	6.850	13.85	13.80

(b) Modo *balanced*.

	SC	FP	FP+SC	Luma
Tiempo mínimo	0.017	0.029	0.010	0.010
Tiempo máximo	0.487	0.145	0.060	0.840
Tiempo promedio	0.066	0.068	0.010	0.060
FPS	14.20	7.100	14.05	14.50

(c) Modo *performace*.

Tabla 5.7: Estadísiticas de rendimiento de IHRVANT.

5.6 Conclusiones

Evaluar el clasificador con una mayor inclinación en el ángulo $roll$ es impráctico, pues dicho ángulo está directamente relacionado con el movimiento lateral del vehículo. Si el ángulo $roll$ sobrepasara los 10°, implicaría que el dron está desplazándose lateralmente a una velocidad mayor que la cadencia normal de una persona.

Con la estrategia que usa IHRVANT del umbral de 2 segundos para detectar un gesto, clasificar los gestos mediante los momentos de Hu es imposible, por las variaciones en la segmentación de la silueta pictórica, aún despúes del proceso de estabilización. Otro modelo puede usarse para clasificar el gesto basado en los momentos de Hu; dichos modelos están descritos en el siguiente capítulo, en la Sección de trabajo futuro.

Es posible estimar la distancia entre el objeto de interés y el dron con un láser. Desafortunadamente, el dron no soporta la carga extra y no fue posible probarlo durante el vuelo. El rango útil

del sensor láser que se implementó en este trabajo de tesis se encuentra entre los $[90 - 140]cm$. El incremento en el error cuando el objeto se acerca al dron puede deberse a la deformación geométrica de la cámara, pero su corrección se dejará como trabajo futuro.

IHRVANT incorpora una serie de estrategias que hicieron posible alcanzar un rendimiento aceptable del procesamiento digital de imágenes con los recursos computacionales de la *tablet* y sin hacer uso de los GPUs. De los tres modos de desempeño disponibles en la *tablet*, el cambio más abrupto en tiempos registrado por IHRVANT fue al pasar de *power saving* a *balanced*. El tercer modo no elevó considerablemente los tiempos registrados.

El mejor de los controles de los grados de libertad del dron es el de *Gaz* y el más inestable es el de *Pitch*. El primero toma como entrada la información proveniente del sensor ultrasónico que porta el dron, mientras que el segundo toma la estimación de la distancia entre el dron y el usuario con procesamiento digital de imágenes (Sección 4.11.1). A pesar de que la segunda se ve afectada por los cambios de iluminación, se logró parcialmente mantener la distancia entre el usuario y el dron.

6

Conclusiones

Este Capítulo describe las conclusiones a las que se puede llegar después de realizar este trabajo de tesis.

Parrot merece una mención especial, pues su soporte para el sistema operativo Android, hasta la versión del SDK 1.8, es casi nulo. El imprevisto causó un retraso significativo en la elaboración de este trabajo de tesis, no fue posible cumplir con la programación de actividades planteadas en el protocolo. Sin demeritar el trabajo de los desarrolladores de Parrot, es bien sabido que el soporte del producto en manos del usuario final es más importante que las ventas mismas. En el caso del autor, esto viene a confirmar la advertencia latente en todo desarrollo técnico: la inexperiencia puede hacer que todo se demore. Aún así, después de una serie de dificultades técnicas, se cumplió con uno de los principales objetivos particulares de esta tesis: Comunicación y control del dron con un dispositivo móvil. Primero, sustituyendo el SDK con software propio y emulando la aplicación del fabricante donde el usuario dicta el comportamiento del dron; segundo, mediante controladores y procesamiento digital de imágenes, dar un sentido de autonomía al dron.

El paradigma de Parrot ofrece a los desarrolladores la posibilidad de construir aplicaciones en

Windows, Linux, iOS y Android, tomando como plataforma el SDK[1] que distribuyen de forma gratuita. Por lo cual, los trabajos tomados del estado del arte usan computadoras personales fijas en tierra o laptops para procesar el flujo de video y están restringidos, inevitablemente, por el alcance de la red inalámbrica del dron y el abastecimiento de energía eléctrica.

Al sustituir el SDK por software propio, se conoció la arquitectura del decodificador y no fue posible usar bibliotecas ya existentes para el procesamiento digital de imágenes (ver Sección 4.3); pero aún así, este trabajo de tesis logró contribuir al estado de arte de las interfaces hombre máquina con la realización de un interfaz con un vehículo aéreo no tripulado y realizando todas las operaciones de PDI en un dispositivo móvil. Esto último, forma parte del objetivo general de esta tesis y su realización se cumplió gracias a la implementación de las estrategias descritas en los Capítulos 3 y 4.

Las siguientes secciones describirán como se abarcaron los demás objetivos particulares y algunos otros detalles dignos de mencionarse.

6.1 Unión sinérgica del decodificador de video con el PDI

La unión sinérgica entre el decodificador y los distintos hilos del PDI, consistió en la sincronización y distribución de la información cromática y lumínica desde el codec a los hilos de PDI. Esta unión fue producto del abandono del SDK, pues el desarrollo de la infraestructura necesaria para la recepción del flujo de video y su decodificación (UVLC $YC_bC_r\ 4:2:0$) dio pie a la implementación de distintas iniciativas.

La configuración del codificador submuestrea los canales de crominancia; si el cuadro del video es VGA, la información cromática puede representarse con una resolución QVGA. El esquema de segmentación propuesto requiere trabajar con información cromática (Capítulo 4) y el submuestreo de la información a procesar aligeró la carga de procesamiento para la *tablet*. Esta estrategia de implementación hizo que IHRVANT alcanzara un rendimiento aceptable con los recursos disponibles en el dispositivo móvil y que coincide con uno de los objetivos particulares de este trabajo de tesis: lograr una velocidad de procesamiento de imágenes compatible con una aplicación de tiempo real,

[1] Hasta la versión 1.8 del SDK, Android era el sistema operativo con menor soporte.

manteniéndose dentro del presupuesto energético de los dispositivos móviles. Cabe mencionar que un esquema de submuestreo para los algoritmos de PDI puede hacer factible el aumento de su rendimiento.

6.1.1 Segmentación por color

Cuando IHRVANT sólo incluía el decodificador y la segmentación por color (implementada en C y JNI) la recepción, decodificación y segmentación se realizaba con una frecuencia de 18Hz; *i.e.* el procesamiento era suficientemente rápido para presentar un resultado cuadro a cuadro.

Se sabe de la existencia de las instrucciones NEON (disponibles en JNI), que siguen el modelo SIMD[2] y con una sola instrucción realiza una misma operación sobre múltiples datos. De acuerdo a ARM, el fabricante, la tecnología NEON puede acelerar algoritmos de procesamiento de señales y multimedia, por lo menos 3 veces, comparados con una arquitectura ARMv5 y por lo menos 2 veces contra una arquitectura ARMv6 SIMD. Por lo tanto, si fuera necesario aumentar el rendimiento de este proceso o de cualquier otro, se puede implementar una estrategia que use operaciones NEON.

6.1.2 Manejo de memoria

El tamaño mínimo de una localidad de memoria en la arquitectura empleada es de 32 bits, cualquier intento de acceder a localidades de menor tamaño resulta en operaciones adicionales. Para hacer un uso eficiente de los accesos a memoria, en los algoritmos donde era posible, se usaron escrituras de 32 bits y se comprobó que este es un factor a considerar si la eficiencia es un objetivo de la implementación.

Por otra parte y de manera complementaria, las operaciones para reservar y liberar memoria deben reducirse al mínimo. Los lenguajes como Java facilitan dichas operaciones, pero impactan directamente en el rendimiento de la aplicación. En una edición temprana de IHRVANT, los búfers eran reservados y liberados por cada cuadro de video procesado; al migrar a una estrategia con búfers fijos en memoria el rendimiento mejoró.

[2]SIMD *Single Instruction-Multiple Data*.

6.1.3 Filtro de partículas

El filtro de partículas tiene distintas aplicaciones y es posible adaptarlo a distintos problemas. El filtro de partículas de IHRVANT se usa para validar el área de segmentado por color que pertenece al objeto de interés. Cada una de las partículas es independiente y no rastrean una área y un factor de escala [28]; además, un rastreo similar no ofrece la posibilidad de obtener la silueta pictórica del objeto de interés. Sería necesario una técnica adicional para obtener la silueta una vez que se conoce el área donde se encuentra el usuario [26].

Por otro lado, la información cromática que usa el filtro de partículas es la que provee el codificador en el espacio de color YC_bC_r, a diferencia de otros trabajos que usan el espacio de color HSV [28]. Ambos espacios de color son igual de útiles, pero dada la información del decodificador en YC_bC_r, no era necesario invertir recursos en cambiar de espacio de color. Además, el filtro de partículas resiste la inestabilidad inherente del vuelo del cuadricóptero, lo que hace posible su rastreo a través del flujo de video.

6.2 Clasificadores

De los clasificadores implementados en IHRVANT, el clasificador basado en histogramas probó ser el que mejores resultados entrega debido en gran parte a la estabilización de la silueta pictórica. La idea detrás de la estabilización de la silueta pictórica (Sección 4.8) fue acondicionar la entrada para el clasificador de gestos y mejorar su rendimiento.

Los momentos de Hu y los histogramas, que sirven de entrada para los clasificadores, se calcularon a partir de escalar y rotar la segmentación de la silueta pictórica en el cuadro de flujo de video. El cuadricóptero puede verse como una cámara que provee información de su orientación, lo cual hizo posible compensar la variación del ángulo *roll* en el aire. Esto fue crucial para mejorar el rendimiento del clasificador por histogramas, pues en condiciones normales de vuelo, el usuario puede ubicarse en distintos lugares dentro del cuadro de video y el cuadricóptero tener distintas orientaciones. Sin la estabilización, un clasificador basado en histogramas hubiera sido imposible de implementar, pues el

usuario debería adoptar una convención que lo ubique en cierta parte del cuadro de video para que funcione correctamente, lo que haría la interacción con el dron menos natural.

6.3 Láser

El láser puede usarse como un sensor extereoceptivo para estimar la distancia al objeto de interés, de tal manera que se puede corregir la deformación geométrica de la cámara con el objetivo de obtener mediciones más fiables; como por ejemplo, un modelo de la cámara que corrija su deformación radial y el efecto del lente de ángulo amplio que monta la cámara frontal.

Es necesario aligerar el peso de la estructura, tal vez omitiendo la tarjeta IOIO, escoger un diodo láser de la misma potencia pero de menor peso y omitir cualquier componente no necesario. Por supuesto, hay que vigilar la distribución de peso sobre el dron para facilitar su vuelo; el balance del montaje debe cuidarse para evitar un par de fuerzas que pudieran hacer girar al dron en el aire.

6.4 Conclusion general

Este trabajo de tesis contribuyó al estado del arte de las interfaces hombre-máquina empleando un dron como robot y la cámara que lleva a bordo para rastrear el usuario e interpretar sus gestos. Para aumentar la funcionalidad de la interfaz y el rango de autonomía del dron, el procesamiento digital de imágenes se realizó en un dispositivo portátil con Android como SO. La aplicación que ejecuta el dispositivo móvil, IHRVANT, es capaz de procesar los cuadros del flujo de video lo suficientemente rápido como para rastrear al usuario y seguirlo, controlando el cuadricóptero de manera autónoma. La interfaz resultó ser práctica, empleando únicamente procesamiento digital de imágenes y pese a que el cuadricóptero no pudo volar con la carga extra del sensor láser.

6.5 Trabajo futuro

Durante el desarrollo y análisis de resultados de este trabajo de tesis es inevitable pensar en mejoras a la arquitectura o técnicas empleadas. En esta sección discutiremos todo este cúmulo de ideas que creció durante el desarrollo de IHRVANT.

De acuerdo a Google, el desarrollador de Android, implementar algoritmos con JNI[3] no significa una mejora inmediata en rendimiento. Por lo cual, el autor considera interesante la posibilidad de implementar los algoritmos de procesamiento digital de imágenes en JNI y Java para medir su rendimiento por separado para saber si vale la pena usar JNI o con JAVA basta.

Uno de las principales debilidades de los sistemas de visión por computadora son las condiciones de iluminación. IHRVANT podría caracterizar las condiciones de iluminación a través de un LDR[4] conectado a la tarjeta IOIO que usa como interfaz entre software y hardware. De esta forma, hipotéticamente, se podría contrarestar la acción de los cambios de iluminación.

Recientemente Parrot lanzó el AR-Drone 2.0, con una cámara de alta definición al frente (720p) y la habilidad de realizar marometas en el aire. Parte del trabajo futuro consiste en modificar a IHRVANT para soportar el dron 2.0 y que sea capaz de realizar marometas como uno de los comandos de la interfaz.

Otros tipos de clasificadores pueden probarse con IHRVANT. Como en [31], se emplea una arquitectura con filtros de partículas y HMMs para rastrear las coyunturas de las extremidades del dorso y brazos humanos. Los gestos son interpretados con redes neuronales MLP[5] y RBF[6]. Por otra parte, en [26] usan un filtro de partículas para rastrear la pose del usuario a partir de una biblioteca de poses llave y con un HMM clasifican el gesto. Es decir, pueden usarse un conjunto de técnicas y sus combinaciones para clasificar gestos.

Manteniendo una estrategia similar al que se propone en el Capítulo 4, sería interesante tomar el problema de segmentación por color como un problema de optimización. El problema puede consistir

[3] Java Native Interface
[4] Light Dependent Resistor.
[5] Multi Layer Perceptron
[6] Radial Base Functions

en maximizar el área segmentada ajustando los parámetros de segmentación $[D^2, h, k]$ y manteniendo una restricción de similitud con alguno de los gestos llave de IHRVANT.

La cámara frontal del cuadricóptero probó tener aberración cromática que es indeseable en los procesos digitales de imágenes. Hace falta compensar los efectos de la aberración cromática de manera que sea minimizada y hay que modelar la cámara para corregir la deformación geométrica. Con la corrección de la deformación hay que probar la exactitud y precisión del sensor láser. Si así el sensor láser no funciona correctamente, habrá que proponer una nueva estrategia para estimar la distancia al objeto de interés.

El último, pero no más sencillo trabajo futuro es construir un cuadricóptero con elementos disponibles comercialmente y en él montar los componentes necesarios para abarcar y superar los objetivos de este trabajo de tesis. Pueden incluirse cámaras infrarojas o termográficas, un sensor láser *range-finder*, un GPS etc. Esto podría permitirnos aumentar el tiempo de vuelo y el grado de autonomía del cuadricóptero, además de aumentar el rango de posibles problemas a resolver como el SLAM [4, 37].

IHRVANT en la *tablet* TF201 de Asus registró tiempos de procesamiento satisfactorio. El desempeño podría variar en otras plataformas con distintos recursos computacionales y la posibilidad de ejecutar IHRVANT en otras plataformas y comparar su desempeño queda como trabajo futuro.

Bibliografía

[1] A. Akl and S. Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation; compressive sensing. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2270 –2273, march 2010.

[2] Omar Javed Alper Yilmaz and Mubarak Shah. Object tracking: A survey. In *ACM Computing Surveys*, volume 38, pages 661–675, December 2006.

[3] A. Bachoo. Using the cpu and gpu for real-time video enhancement on a mobile computer. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 405 –408, oct. 2010.

[4] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096 –1097, may 2010.

[5] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5776 –5783, may 2011.

[6] S. Bosch, S. Lacroix, and F. Caballero. Autonomous detection of safe landing areas for an uav from monocular images. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5522 –5527, oct. 2006.

[7] Kwang-Ting Cheng and Yi-Chu Wang. Using mobile gpu for general-purpose computing; a case study of face recognition on smartphones. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1 –4, april 2011.

[8] B.A. Duncan, R.R. Murphy, D. Shell, and A.G. Hopper. A midsummer night's dream: Social proof in hri. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 91 –92, march 2010.

[9] J.-P. Farrugia, P. Horain, E. Guehenneux, and Y. Alusse. Gpucv: A framework for image processing acceleration with graphics processors. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 585 –588, july 2006.

[10] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid. Power emulation based dvfs efficiency investigations for embedded systems. In *System on Chip (SoC), 2010 International Symposium on*, pages 173 –178, sept. 2010.

[11] Gramazio and Kohler et Raffaello D'Andrea. Flight assembled architecture. HYX, Orléans, february 2012.

[12] S. Grzonka, G. Grisetti, and W. Burgard. A fully autonomous indoor quadrotor. *Robotics, IEEE Transactions on*, PP(99):1 –11, 2011.

[13] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 361 –366, april 2007.

[14] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto. The landing problem of a vtol unmanned aerial vehicle on a moving platform using optical flow. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1600 –1605, oct. 2010.

[15] Keita Higuchi, Tetsuro Shimada, and Jun Rekimoto. Flying sports assistant: external visual imagery representation for sports training. In *Proceedings of the 2nd Augmented Human International Conference*, AH '11, pages 7:1–7:4, New York, NY, USA, 2011. ACM.

[16] Y. Hirota, T. Masuda, R. Kurazume, K. Ogawara, K. Hasegawa, and K. Ikeuchi. Flying laser range finder and its data registration algorithm. In *Robotics and Automation, 2004. Proceedings.*

ICRA '04. 2004 IEEE International Conference on, volume 3, pages 3155 – 3160 Vol.3, april-1 may 2004.

[17] A.D. King. Inertial navigation - forty years of evolution. *GEC REVIEW*, 13(3):140–149, 1998.

[18] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah. A low-cost laser distance sensor. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3002 –3008, may 2008.

[19] M. Kurisu, H. Muroi, and Y. Yokokohji. Calibration of laser range finder with a genetic algorithm. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 346 –351, 29 2007-nov. 2 2007.

[20] M. Kurisu, H. Muroi, Y. Yokokohji, and H. Kuwahara. Development of a laser range finder for 3d map-building in rubble; installation in a rescue robot. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pages 2054 –2059, aug. 2007.

[21] A. Malima, E. Ozgur, and M. Cetin. A fast algorithm for vision-based hand gesture recognition for robot control. In *Signal Processing and Communications Applications, 2006 IEEE 14th*, pages 1 –4, april 2006.

[22] Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30:73–86, 2011. 10.1007/s10514-010-9205-0.

[23] Shu Mo, Shihai Cheng, and Xiaofen Xing. Hand gesture segmentation based on improved kalman filter and tsl skin color model. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 3543 –3546, july 2011.

[24] Wai Shan Ng and E. Sharlin. Collocated interaction with flying robots. In *RO-MAN, 2011 IEEE*, pages 143 –149, 31 2011-aug. 3 2011.

[25] The Benefits of Multiple CPU Cores in Mobile Devices. Nvidia Corporation, 2010.

[26] Chi-Min Oh, M.Z. Islam, and Chil-Woo Lee. Pictorial structures-based upper body tracking and gesture recognition. In *Frontiers of Computer Vision (FCV), 2011 17th Korea-Japan Joint Workshop on*, pages 1 –6, feb. 2011.

[27] M.A. Olivares-Mandndez, I.F. Mondrag andn, P. Campoy, and C. Mart andnez. Fuzzy controller for uav-landing task using 3d-position visual estimation. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1 –8, july 2010.

[28] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *In Proc. ECCV*, pages 661–675, 2002.

[29] J. Richer and J.L. Drury. A video game-based framework for analyzing human-robot interaction: characterizing interface design in real-time interactive multimedia applications. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on human-robot interaction*, pages 266–273. ACM, 2006.

[30] A.L. Salih, M. Moghavvemi, H.A.F. Mohamed, and K.S. Gaeid. Modelling and pid controller design for a quadrotor unmanned air vehicle. In *Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on*, volume 1, pages 1 –5, may 2010.

[31] M. Sigalas, H. Baltzakis, and P. Trahanias. Gesture recognition based on arm tracking for human-robot interaction. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5424 –5429, oct. 2010.

[32] N. Singhal, In Kyu Park, and Sungdae Cho. Implementation and optimization of image processing algorithms on handheld gpu. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 4481 –4484, sept. 2010.

[33] Chhay Sok and M.D. Adams. Visually aided feature extraction from 3d range data. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2273 –2279, may 2010.

[34] Gabriel Takacs, Vijay Chandrasekhar, Natasha Gelfand, Yingen Xiong, Wei-Chao Chen, Thanos Bismpigiannis, Radek Grzeszczuk, Kari Pulli, and Bernd Girod. Outdoors augmented reality

on mobile phone using loxel-based visual feature organization. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, MIR '08, pages 427–434, New York, NY, USA, 2008. ACM.

[35] Anand Thobbi, Ye Gu, and Weihua Sheng. Using human motion estimation for human-robot cooperative manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2873 –2878, sept. 2011.

[36] Hee-Deok Yang, A-Yeon Park, and Seong-Whan Lee. Human-robot interaction by whole body gesture spotting and recognition. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 774 –777, 0-0 2006.

[37] Tianguang Zhang, Ye Kang, M. Achtelik, K. Kuhnlenz, and M. Buss. Autonomous hovering of a vision/imu guided quadrotor. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pages 2870 –2875, aug. 2009.