BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

TESIS DE MAESTRÍA

Seguimiento de objetos con Vehículos Aéreos No Tripulados

Autor:
Osvaldo ESCOBAR DÍAZ

Asesor:
Dr. Abraham SÁNCHEZ LÓPEZ

Que para obtener el grado de
Maestro en Ciencias de la Computación
presenta
OSVALDO ESCOBAR DÍAZ

en

Movis Research Group
Facultad de Ciencias de la Computación

Noviembre 2015

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Introducción

Facultad de Ciencias de la Computación
Maestría en Ciencias de la Computación

Tesis

por Osvaldo ESCOBAR DÍAZ

Los robots aéreos son dispositivos que se encuentran cada vez más presentes en nuestro entorno, gracias a la miniaturización de los componentes electrónicos que los conforman, haciendo posible utilizarlos en situaciones variadas como pueden ser aplicaciones de servicios: localizar personas, filmación de eventos o inspección de zonas de difícil acceso, como es el caso de la inspección de los reactores nucleares dañados en Fukushima en 2011, o la inspección sobre la ciudad fantasma de Chernobyl en 2014, que sufrió las consecuencias del desastre nuclear de 1986, entre otras.

Los micro-vehículos aéreos son aeronaves no tripuladas con un tamaño que va en el orden de los centímetros, los cuales pueden o no ser autónomos. En este documento se hará referencia a ellos por las siglas MAV – del inglés Micro Aerial Vehicle – ya que son las más utilizadas en la literatura. En años recientes, los MAVs han llegado a ser una herramienta importante no solo en el dominio militar, sino también en ambientes civiles.

El seguimiento de objetos con MAVs ha creado un gran interés entre la comunidad científica en los últimos años, sobre todo por sus aplicaciones como la de inspección de objetos (grabación) que se mueven, sin la necesidad de controlar el MAV desde una estación remota. Es por ello que en este documento se describe el objetivo principal de este trabajo, que es el del seguimiento mediante un MAV y la metodología con la que se puede lograr.

Agradecimientos

Doy gracias a mi esposa Carolina Angélica, que durante todo el trayecto de la Maestría recibí su apoyo y comprensión, con su amor, éste ha sido un camino agradable. Caro siempre me ha impulsado a lograr lo que quiero.

Agradezco a mis padres, Bárbara y Enrique, por su incondicional y más fuerte soporte, ellos han estado en los momentos más difíciles de éste grado que obtengo, en los desvelos y los momentos de mayor preocupación. Su ayuda y cariño me han mantenido en pie todo el tiempo.

Y no menos importante, agradezco todas las facilidades de mi asesor de Tesis, Dr. Abraham, por su conducción en éste y otros proyectos, así como por su incomparable amistad.

Índice

Introducción I

Agradecimientos II

Índice III

Índice de Figuras V

Índice de Tablas VI

Abreviaciones VII

Símbolos VIII

1. Estado del Arte **1**
1.1. Vehículos aéreos no tripulados 1
1.2. Visión Artificial 5
1.2.1. Seguimiento de objetos 6
1.2.1.1. Filtro de partículas 7
1.2.2. Reconocimiento de objetos 10
1.2.2.1. Segmentación 11
1.2.2.2. Histograma de Gradientes Orientados 12
1.3. Los MAVs para el seguimiento de objetos 13
1.4. Organización de la Tesis 14

2. El AR.Drone 2.0 y su control **15**
2.1. Introducción 15
2.2. Movimientos del AR.Drone 2.0 15
2.3. Sensores de movimiento 17
2.4. Video streaming 17
2.5. Servicios de comunicación entre el AR.Drone 2.0 y el dispositivo cliente 17
2.6. El Control del AR.Drone 2.0 18
2.6.1. Plataforma para control 18
2.6.1.1. Arquitectura del software 19
2.6.1.2. La biblioteca AR.Drone 19
2.6.1.3. Administración del vídeo 20
2.6.1.4. Flujo de vídeo 20

2.6.2. ROS 22
2.6.2.1. ardrone_autonomy 22
2.6.3. RT_ARDrone 23

3. Operación y Administración de las Funciones del MAV **24**
3.1. Movimiento del ARDrone 2.0 24
3.2. Captura de navegación 27
3.3. Captura de video 28

4. Visión por computadora con el MAV **30**
4.1. Umbralización 30
4.2. Histograma de Gradientes Orientados 31
4.3. Filtro de partículas 36

5. Resultados **38**
5.1. Pruebas 38
5.2. Restricciones del problema 40

6. Conclusiones **42**
6.1. Trabajo futuro 43

Referencias **44**

Índice de Figuras

1.1. Ejemplos de MAVs 2
1.2. AR.Drone de Parrot 4
1.3. Pesos de Partículas en FDP 8
1.4. Filtro de Partículas 9
1.5. Evolución de Partículas 11
1.6. Proceso de umbralización 12
1.7. Proceso HOG 13
1.8. Dron MeCam (AlwaysInnovating) 13
1.9. Dron Mind4 (AirMind) 14

2.1. AR.Drone 2.0 15
2.2. AR.Drone 2.0 Motores 16
2.3. AR.Drone 2.0 Movimientos 16
2.4. Arquitectura 19

4.1. Umbralización con MAV 31
4.2. Gradientes de color 32
4.3. Histograma de una imagen 33
4.4. Bloques y Celdas del HOG 33
4.5. Divisiones de las Orientaciones 34
4.6. Proceso HOG 35
4.7. Etapas del HOG 35
4.8. HOG con el MAV 36
4.9. Filtro de partículas desde el MAV 37

5.1. Reconocimiento de persona 39
5.2. Reconocimiento de varias personas 39
5.3. Seguimiento de una persona por el MAV 40
5.4. Visión por Computadora por el MAV 40

Índice de Tablas

3.1. Comandos AT para mover el MAV 25
3.2. Teclas para mover el MAV 26

Abreviaciones

CMOS	**C**omplementary **M**etal-**O**xide-**S**emiconductor
DGAC	**D**irección **G**eneral de **A**eronáutica **C**ivil
DOF	**D**egrees **O**f **F**reedom (Grados de Libertad)
FPS	**F**rames **P**er **S**econd (Imágenes por Segundo)
FTP	**F**ile **T**ransfer **P**rotocol (Protocolo de Transferencia de Archivos)
HOG	**H**istogram of **O**riented **G**radients (Histograma de Gradientes Orientados)
IMU	**I**nertial **M**easurement **U**nit (Unidad de Medición Inercial)
MAV	**M**icro **A**erial **V**ehicle (Micro Vehículo Aéreo)
MEMS	**M**icro-**E**lectro-**M**echanical **S**ystems
ROS	**R**obot **O**perative **S**ystem
SCT	**S**ecretaría de **C**omunicaciones y **T**ransportes
SDK	**S**oftware **D**evelopment **K**it (Kit de desarrollo de software)
SIFT	**S**cale-**I**nvariant **F**eature **T**ransform
SVM	**S**upport **V**ector **M**achine (Máquina de Soporte Vectorial)
TCP	**T**ransmission **C**ontrol **P**rotocol
UAV	**U**nmanned **A**erial **V**ehicle (Vehículo Aéreo No Tripulado)
UDP	**U**ser **D**atagram **P**rotocol

Símbolos

$kbps$	Kilobit por segundo	kbit/s
$Mbps$	Megabit por segundo	Mbit/s
m	Metros	
ms	Milisegundos	

Capítulo 1

Estado del Arte

En el presente Capítulo se verá la evolución de los Vehículos Aéreos No Tripulados en la historia y la mención de algunos trabajos que incluyen el uso de estos dispositivos, así como el marco teórico de algunas de las técnicas utilizadas para realizar el seguimiento de objetos.

1.1. Vehículos aéreos no tripulados

Los vehículos aéreos no tripulados (Unmanned Aerial Vehicles, UAV), en los últimos años han visto niveles sin precedentes de crecimiento en aplicaciones de dominio militar y civil para vigilancia, reconocimiento, la cartografía, patrullas fronterizas, la inspección, seguridad nacional, búsqueda y rescate, detección de incendios, monitoreo de tráfico, entre otros dominios de aplicación [1].

Anteriormente, los UAV eran propulsados por motores de combustión, lo que impedía su miniaturización por debajo de cierto tamaño. Conforme cambió esa situación y aparecieron los motores sin escobillas de alta eficiencia y con el incremento de la densidad de energía almacenable por la tecnología de las nuevas baterías, se ha hecho posible la miniaturización de estos vehículos. También, la reducción imparable de la electrónica en tamaño, consumo y costo, ha propiciado la transición de los aviones o helicópteros operados por línea de visión directa a complejas plataformas autónomas con procesadores y sensores de todo tipo. Con la ventaja de los motores sin escobillas y ya con la posibilidad de incluir controladores digitales a bordo, surgieron nuevas configuraciones mecánicas con varias alas rotativas, vehículos que tienen múltiples

capacidades dinámicas, llamados multirrotores, como [2]. De acuerdo al número de alas rotativas, reciben el nombre de trirrotores, cuatrirrotores, hexarrotores, etc. En los últimos años, es cuando se comienza a adoptar el término micro-vehículo aéreo, MAV, – del inglés *Micro Aerial Vehicle* – para designar en general a los dispositivos voladores no tripulados, que tienen proporciones que les permiten ser transportadas por alguna persona a pie, como los de la Figura 1.1 Otras definiciones son Aeronave no Tripulada, UA, – del inglés *Unmanned Aircraft* –; Sistema de Aeronave no Tripulada, UAS, – del inglés *Unmanned Aircraft System* – o Aeronave Pilotada a Distancia, RPA, – del inglés *Remote Piloted Aircraft* –. Los MAVs han atraído un interés mayor en la investigación durante la última década [3].

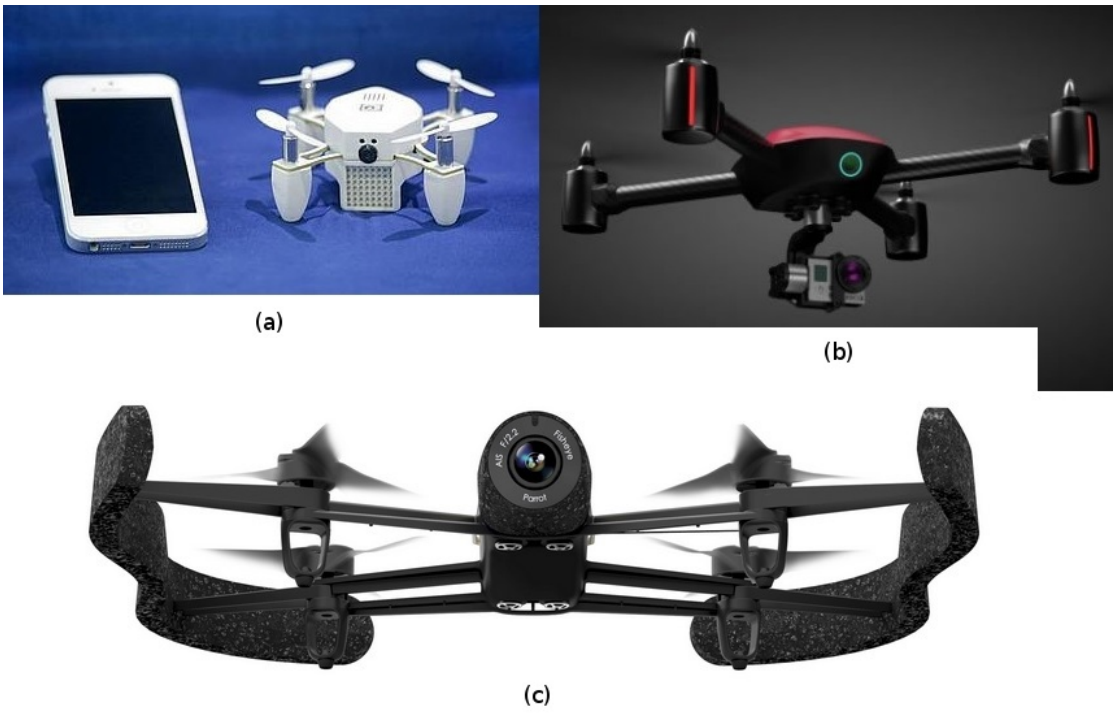

FIGURA 1.1: Ejemplos de Microvehículos aéreos. **(a)** Zano, un prototipo de MAV muy pequeño. **(b)** Mind4, un MAV que se encuentra en desarrollo para seguimiento de objetos. **(c)** Bebop, MAV que graba en Alta Definición.

Ya que los MAV son no tripulados, se necesita de algún cierto grado de autonomía, donde un MAV con autonomía total es capaz de interactuar o realizar sus funciones sin intervención humana, mientras que en el caso más común, un operador le manda instrucciones remotamente de algunas actividades a realizar, siendo el MAV el que ejecuta las tareas más comunes, como puede ser la estabilización de la verticalidad, que por lo general son tareas que requieren menor grado de inteligencia. Por ejemplo, el MAV puede incluir algún ciclo de control para mantener o corregir una posición estática, mientras el operador decide a donde dirigirse a continuación.

Según la complejidad de las acciones automatizadas, se tiene mayor o menor grado de autonomía.

Los cuatrirrotores o más comúnmente llamados cuadricópteros han llegado a ser más populares, en especial por sus propósitos de exploración y observación en interiores y exteriores, pero también por su colección de datos, manipulación de objetos o simplemente como juguetes de alta tecnología. Con un comportamiento similar a un helicóptero tradicional, un cuadricóptero es capaz de despegar y comenzar su vuelo verticalmente, permanecer estable en el aire y moverse en cualquier dirección en cualquier momento, sin tener que girarlo primero. Esto permite a los cuadricópteros – al contrario que los aviones tradicionales – poder maniobrarlos en espacios interiores como pasillos u oficinas.

El concepto de una aeronave con cuatro rotores alineados horizontalmente fue propuesto en 1922 [4], aunque el diseño rápidamente desapareció y fue sustituido por el mucho más común helicóptero de dos rotores. Esto dado que el cuadricóptero es inherentemente inestable y por lo tanto difícil de controlar – no contaba con ayuda de los sistemas de control electrónico avanzado y rutinas de estabilización -. Ahora, con el importante crecimiento de los MAVs, el diseño del cuadricóptero ha sido popular de nuevo. Es mecánicamente más simple que el helicóptero normal. Además, los cuatro rotores pueden ser protegidos por un marco, como [5], protegiéndolos en colisiones y permitiéndoles vuelos más seguros en interiores y ambientes con varios obstáculos.

Para poder navegar, los MAVs modernos pueden contar con una amplia variedad de sensores. Además de una unidad de medición inercial (IMU), la medición de la orientación y la aceleración de la aeronave, una unidad ultrasónica que mide la altura a la que se encuentra, o un sistema de navegación GPS que es frecuentemente utilizado para determinar la posición absoluta del MAV. Algunos sensores son utilizados para medir valores simples como la velocidad de rotación de los motores.

Una de las tareas más importantes de un sistema autónomo o vehículo de cualquier tipo es adquirir conocimiento acerca de su entorno. Esto es hecho utilizando uno o más sensores para realizar mediciones y extrayendo información valiosa. Una de las formas más directas de recolectar información acerca del ambiente son las cámaras ópticas, ya que proveen una amplia cantidad de información y son sensores comparativamente baratos, pequeños y ligeros, y esto permite situarlos comúnmente en los MAVs. Sin embargo, está demostrado que el tratamiento

de ésta gran cantidad de información de datos visuales es una tarea muy desafiante que requiere de gran capacidad computacional.

En los últimos años, han aparecido en el mercado modelos de MAV para aficionados con precios considerablemente bajos comparados con los modelos profesionales. Después se empezaron a desarrollar modelos más sofisticados dirigidos a usuarios más técnicos. Estas aeronaves se han popularizado en los últimos años dentro del ámbito académico por su bajo costo, su relativa facilidad de construcción y su reducida cantidad de partes móviles, lo que permite que varias universidades del mundo las utilicen [6].

FIGURA 1.2: AR.Drone de Parrot®.

En enero de 2010, fue introducido al mercado el cuadricóptero AR.Drone de Parrot® (Figura 1.2), originalmente diseñado como un juguete sofisticado para juegos de realidad aumentada. Esto para controlarlo por un piloto utilizando un teléfono inteligente o una tableta. Sin embargo, este MAV rápidamente captó la atención de las universidades y las instituciones de investigación, y hoy en día es utilizado en muchos proyectos de investigación en el campo de la robótica, inteligencia artificial y la visión por computadora [7]. Además, este modelo de MAV ofrece la posibilidad de desarrollar software de control gracias a un kit del fabricante [8], el cual han ido enriqueciendo poco a poco mediante la entrega de un *SDK* y la creación en junio de 2015 de un foro más amigable que el que anteriormente existía [9].

Ya que se puede desarrollar software para los MAVs, entre más compleja es una aplicación, se hace necesario la adopción de alguna arquitectura que resuelva los problemas básicos de computación y que ayude a organizar la estructura de los programas. Es así que los desarrolladores pueden enfocarse en solucionar problemas de más alto nivel y no en reinventar lo que ya se ha realizado. En este sentido, existe la arquitectura de software de PIXHAWK [10, 11], un framework para MAVs de ETH Zurich, pero que no incluye soporte para el AR.Drone. Por otra parte, existe un controlador de software para el AR.Drone [12] que funciona sobre el sistema

operativo ROS (Robot Operating System), el cual es un sistema que proporciona un mecanismo de intercomunicación entre componentes basado en mensajes.

El auge que han tenido los MAVs en estos últimos años, ha sido realmente impresionante. Actualmente, la mayoría de las personas en México saben o tienen una idea de lo que es un 'dron', como se le llama normalmente a un microvehículo aéreo (MAV), a diferencia de hace dos años, cuando comenzó este trabajo de tesis, donde muy pocos entendían que era un dron y lo que era capaz de hacer. Así es como en abril de 2015, la DGAC de la SCT del gobierno mexicano, hace la regulación sobre el `Uso de Aeronaves No Tripuladas (Drones)`, las cuales llama "Sistema de Aeronave Pilotada a Distancia, (RPAS)" [13], la cual establece principalmente mediante la masa del MAV ciertas restricciones como la necesidad de contar con un permiso de la DGAC, y especificaciones de uso, como su operación a cierta distancia de aeropuertos y helipuertos. Así mismo, en el ámbito científico, debido al interés que ha causado, en marzo de 2015 se realizó el ExpoDrone MX[14], el primer evento de su tipo en México, donde se presentaron diversas marcas de MAVs, conferencias de líderes del sector donde mostraron los campos de aplicación de esta industria y pruebas donde los asistentes podían pilotar un dron [15], evento que tendrá más ediciones.

1.2. Visión Artificial

La inteligencia artificial (IA) es un área multidisciplinaria que estudia la creación y diseño de entidades capaces de razonar por sí mismas utilizando como paradigma la inteligencia humana.

Tan general y amplio como eso, es que reúne varios campos (robótica, sistemas expertos, entre otros), todos los cuales tienen en común la creación de máquinas que pueden pensar. En ciencias de la computación se denomina inteligencia artificial a la capacidad de razonar de un agente no vivo. John McCarthy acuñó la expresión "inteligencia artificial" en 1956, y la definió así: "Es la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes"[1].

La visión artificial, también conocida como Visión Computacional o Visión por Computadora (del inglés *computer vision*), es un subcampo de la inteligencia artificial. Su propósito es programar una computadora para que "entienda" una escena o las características de una imagen.

[1] Basic Questions; http://www-formal.stanford.edu/jmc/whatisai/node1.html

Los objetivos típicos de la visión artificial incluyen la detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).

Un sistema de Visión Artificial actúa sobre una representación de una realidad que le proporciona información sobre brillo, colores, formas, etcétera. Estas representaciones suelen estar en forma de imágenes estáticas, escenas tridimensionales o imágenes en movimiento.

La Visión Artificial en un intento de reproducir el comportamiento del ser humano, define tradicionalmente cuatro fases principales:

- La primera fase, que es puramente sensorial, consiste en la *Captura* o *Adquisición* de las imágenes digitales mediante algún tipo de sensor.
- La segunda etapa consiste en el tratamiento digital de las imágenes, con objeto de facilitar las etapas posteriores. En esta etapa de *Preprocesamiento* es donde, mediante filtros y transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma.
- La siguiente fase se conoce como *Segmentación*, y consiste en aislar los elementos que interesan de una escena para comprenderla.
- Por último se llega a la etapa de *Reconocimiento* o *Clasificación*. En ella se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

Estas cuatro fases no se siguen siempre de manera secuencial, sino que en ocasiones deben retroalimentarse hacia atrás. Así, es normal volver a la etapa de segmentación si falla la etapa de reconocimiento, o la de preproceso, o incluso a la de captura, cuando falla alguna de las siguientes.

1.2.1. Seguimiento de objetos

El seguimiento de objetos es el proceso de estimar en el tiempo la ubicación de uno o más objetos móviles mediante el uso de una cámara. La rápida mejora en cuanto a calidad y resolución de los sensores de imagen, juntamente con el dramático incremento en cuanto a la potencia de cálculo en la última década, ha favorecido la creación de nuevos algoritmos y aplicaciones mediante el seguimiento de objetos.

El seguimiento de objetos puede ser un proceso lento debido a la gran cantidad de datos que contiene un vídeo. Además, la posible necesidad de utilizar técnicas de reconocimiento de objetos para realizar el seguimiento incrementa su complejidad.

1.2.1.1. Filtro de partículas

El *Filtro de partículas*, es un método secuencial de Monte Carlo[2] aplicable a cualquier transición de estados o modelo de medida, y en la Visión Artificial se utiliza para el seguimiento de objetos en secuencias de imágenes.

El *Filtro de partículas* fue introducido originalmente en el área de la Visión Computacional como el algoritmo Condensation (**CON**ditional **DENS**ity Propag**ATION**), que fue presentado por Michael Isard y Andrew Blake para seguir contornos y lo llamaron ICONDENSATION. J. MacCormick y A. Blake lo han ampliado desde entonces [16].

El Algoritmo de *Filtro de partículas* proporciona una forma simple y efectiva de modelar procesos estocásticos con funciones de densidad de probabilidad arbitrarias (no necesariamente normales o gausianas) por aproximación numérica de éstas. Las partículas son estados posibles del proceso que se pueden representar como puntos en el espacio de estados de dicho proceso.

El *Filtro de partículas* fue propuesto por N. Gordon, D. Salmond y A. Smith en 1993 [17], como *Filtro Bootstrap* para implementar *Filtros bayesianos recursivos*. La densidad requerida del vector de estados era representada como un conjunto de muestras aleatorias, las cuales se actualizan y propagan por el algoritmo. El *filtro de partículas* supone una nueva manera de representar y generar recursivamente una aproximación del estado de la Función de Densidad de Probabilidad (FDP). La idea central es representar la FDP requerida como un conjunto de muestras aleatorias, más que como una función sobre el espacio de estados. La aproximación Bayesiana se utiliza para construir la FDP, que define el estado del objeto en el instante actual.

El *Filtro de Partículas* representa la densidad a posteriori mediante una distribución de partículas (muestras) en el espacio de estados. Las partículas son estados posibles del proceso, que se pueden representar como puntos en el espacio de estados de dicho proceso. Este enfoque se ha desarrollado de forma independiente en los últimos años en campos como la estadística, la economía o la visión artificial. Los nombres con los que se le ha denominado son: Método Secuencial de

[2]Método que utiliza una ruleta como un generador simple de números aleatorios que da solución a una gran variedad de problemas matemáticos posibilitando la realización de experimentos con muestreos estadísticos en una computadora.

Monte Carlo (Sequential Monte-Carlo Methods), Algoritmo Condensation (CONDENSATION Algorithm), Filtro Bootstrap (Bootstrap Filter), Filtro de la Supervivencia del Más Apto, aunque últimamente se está utilizando el término Filtro de Partículas (Particle Filter) para todos ellos. Todos estos algoritmos son propuestas similares que propagan las partículas (muestras de la función de densidad a posteriori), utilizando el *modelo de movimiento* $P(x_t|x_{t-1}, a_t)$ y el *modelo de verosimilitud* $P(z_t|x_t)$ de forma que el peso combinado de las partículas de una región aproxima la integral de la función de densidad a posteriori en esa región. En concreto, el *Filtro de Partículas* representa la densidad a posteriori mediante un conjunto discreto de N partículas $\{m_1, \ldots, m_N\}$ y sus probabilidades asociadas $\{\pi_1, \ldots, \pi_N\}$.

Inicialmente, el conjunto de partículas se escoge a partir de la distribución a priori $p(x_0)$. Si no existe información a priori, entonces las partículas se distribuyen uniformemente por el espacio de estados. Posteriormente, en cada instante de tiempo t, se actualizan las N partículas en función de la acción anterior a_{t-1} y la observación actual z_t. Para ello, se aplica el *modelo de movimiento* $p(x_t|x_{t-1}, a_{t-1})$ a cada una de las N partículas, generando un nuevo conjunto de partículas. Las partículas nuevas representan la predicción de la variable de estado, sin considerar la observación, se obtiene el peso π^i asociado a cada partícula. El conjunto de pesos de cada partícula es proporcional a la probabilidad de su estado y a la suma normalizada de sus pesos. La densidad de los pesos es igual al producto de la densidad previa (del muestreo) y la probabilidad (de los pesos). En la Figura 1.3 se representa un muestreo discreto de una función de densidad de probabilidad continua mediante partículas, cuyo tamaño hace referencia al peso asignado a las mismas.

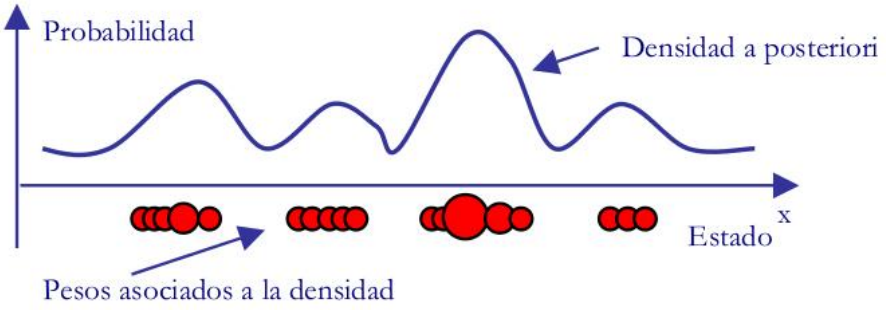

FIGURA 1.3: Muestreo y pesos obtenidos de las partículas al aplicar el Filtro.

En un último paso, se remuestrea el conjunto de partículas, extrayendo (con reemplazo) N partículas del conjunto actual, proporcional al peso de cada una. En este nuevo conjunto tendrán más

probabilidad de desaparecer aquellas partículas para las que no hay evidencia de verosimilitud o, lo que es lo mismo, que tenga menor peso. Una vez construido el nuevo conjunto de partículas, según la probabilidad de éstas se asocia un peso a cada una. Este nuevo conjunto de partículas constituye una representación muestral de la probabilidad a posteriori.

En la Figura 1.4, se observa de manera gráfica la evolución de las partículas en cada fase, suponiendo que las partículas están estimando un único parámetro, distribuido en el eje horizontal. El área de los círculos representa el peso de cada partícula.

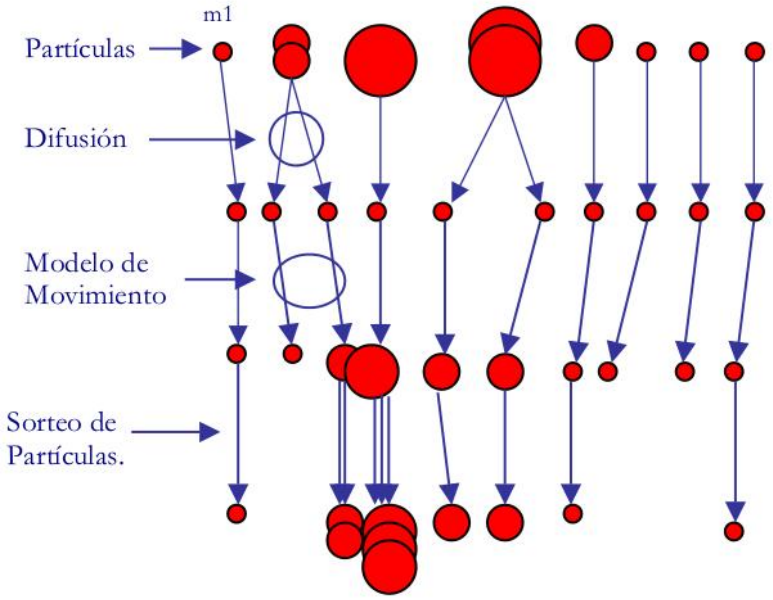

FIGURA 1.4: Funcionamiento del Filtro de Partículas.

Los elementos que se consideran en el método del *Filtro de Partículas* son:

- Los *Vectores de medidas* Z, en este caso, serán provenientes de las imágenes. Las medidas dependen del estado del objeto, y el estado del objeto se deriva estadísticamente de las medidas.
- El *Modelo de Movimiento* F, se utiliza para predecir la posición del objeto en el instante actual, a partir de la densidad de probabilidad del instante anterior $(x_{t+1} = F(x_t))$.
- El *Modelo de Verosimilitud* $P(Z_t|x_t)$, la estimación de la función de probabilidad condicional a posteriori $P(Z_t|x_t)$ define la verosimilitud de la medida observada dado un punto en el espacio de estados (es decir, dado un estado del sistema).

Para poder implementar el algoritmo de actualización es necesario un conjunto bien ponderado de partículas en el instante t con pesos iguales $\frac{1}{N}$, y así actualizar este conjunto para reflejar las nuevas medidas obtenidas en el instante $t + \delta t$. El algoritmo de actualización queda como sigue:

1. Propagar cada partícula m_i en el tiempo utilizando el *modelo de movimiento* F del objeto para obtener un conjunto actualizado de partículas $\{m_i^*\}$.

2. Obtener un nuevo vector de medidas Z y evaluar la *densidad de probabilidad posterior* π_i^* para cada m_i^*, $\pi_i^* = p(m_i^*|Z)$ que cuantifica la verosimilitud de (un estado) m_i^* dado un *vector de medidas* Z. Esto puede ser escrito usando la *regla de Bayes* descrita en la Ecuación 1.1.

$$Pr(m_i^*|Z) = \frac{p(Z|m_i^*)p(m_i^*)}{p(Z)} \tag{1.1}$$

   Donde:

   $p(Z)$ es la probabilidad a priori de la medida que se asume constante y conocida.

   $p(m_i^*) = \frac{1}{N}$, por lo que: $p(m_i^*|Z) = Kp(Z|m_i^*)$ donde $p(Z|m_i^*)$ puede ser calculado sin inversión de las ecuaciones de medida.

3. Volver a muestrear a partir del conjunto $\{m_i^*\}$ con probabilidades π_i^* y generar un nuevo conjunto bien ponderado $\{m_i^{'}\}$ con pesos iguales $(\frac{1}{N})$ para cada partícula. El muestreo de cada partícula consiste en un estado, un peso y otra información posible (una covarianza por instante).

4. Repetir los pasos 1 y 3 para instantes sucesivos.

En la Figura 1.5, se observa la forma en que evolucionan las partículas en el tiempo durante una ejecución del *Filtro de Partículas*.

1.2.2. Reconocimiento de objetos

El Reconocimiento de Objetos es la tarea que consiste en encontrar e identificar objetos en una imagen o secuencia de vídeo. Los humanos reconocemos una multitud de objetos en imágenes con poco esfuerzo, a pesar del hecho que la imagen del objeto puede variar un poco en diferentes puntos de vista, en diferentes tamaños o escala e incluso cuando están trasladados o rotados.

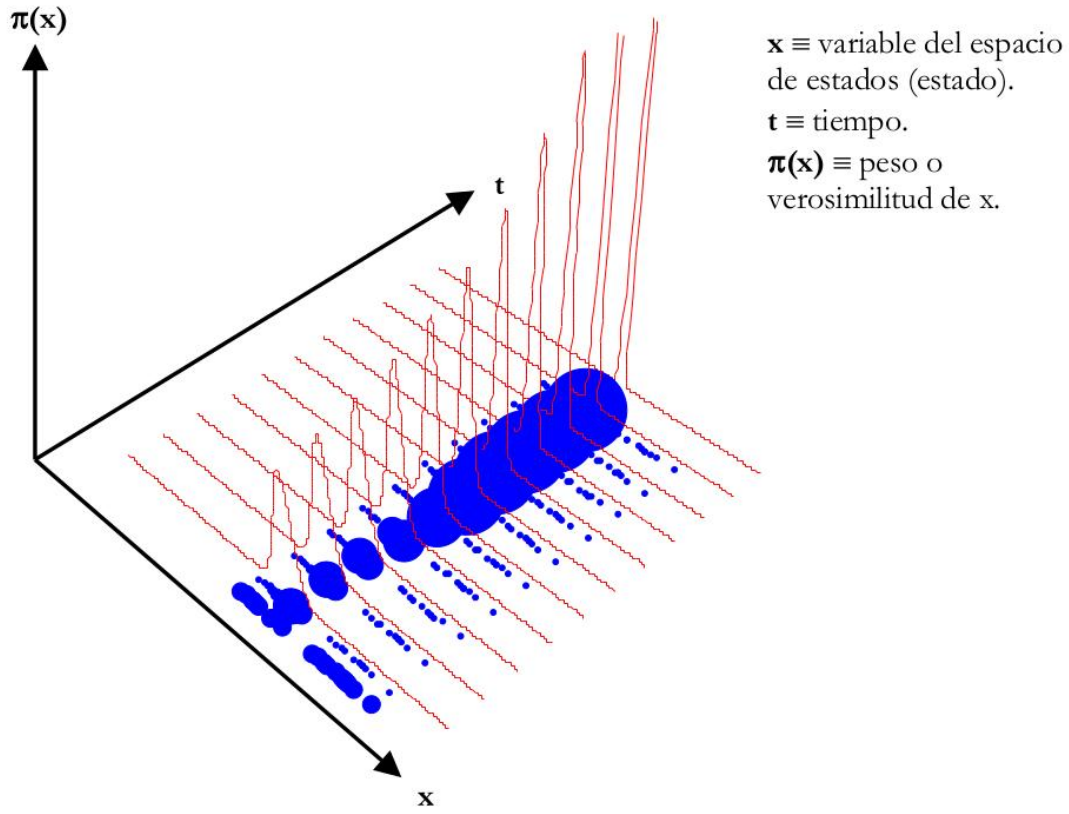

FIGURA 1.5: Evolución de la densidad de probabilidad y de su modelado a través de partículas.

Los objetos pueden ser reconocidos cuando están parcialmente obstruidos desde una vista. No obstante esta tarea es un desafío para los sistemas de visión por computadoras.

1.2.2.1. Segmentación

La Segmentación es un proceso que se encarga de dividir una imagen digital en varias partes. Es utilizada tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Concretamente, es un proceso de asignación de una etiqueta a cada píxel de la imagen. Se ha generado una gran cantidad de trabajos que presentan diversas técnicas, modelos y algoritmos. Éstas técnicas se dividen en cuatro grandes grupos [18]:

- Técnicas de segmentación basadas en los valores de píxel.
- Técnicas de segmentación basadas en el área.
- Técnicas de segmentación basadas en orillas.
- Técnicas de segmentación basadas en la física.

En la Figura 1.6 (b) se aprecia una segmentación basada en los valores de píxel de la que se encuentra en la Figura 1.6 (a), donde se colorean de negro los píxeles que están cercanos al azul y al blanco.

(a) (b)

FIGURA 1.6: **(a)** Imagen real **(b)** Imagen *(a)* segmentada

1.2.2.2. Histograma de Gradientes Orientados

El algoritmo se usa en ámbitos como la visión por computadora y el procesamiento de imágenes con el propósito de detectar objetos en una imagen. La esencia de dicho algoritmo es que la forma de un objeto en una imagen puede ser descrito por medio de la distribución de los gradientes. Fue introducido en 2005 por Navneet Dalal y Bill Triggs [19].

El objetivo principal de esta técnica es la extracción de características de una imagen, las cuales identifiquen la silueta de una persona o algún otro objeto. Las características son extraídas teniendo en cuenta los bordes. El proceso de obtener información de los bordes presentes en una imagen, se consigue calculando los gradientes y las orientaciones de los píxeles (Figura 1.7).

La idea principal es que la apariencia y la forma de los objetos pueden ser caracterizados de mejor manera por la distribución de gradientes de intensidad locales, inclusive sin conocimiento del gradiente correspondiente. Esto se hace mediante la división de la imagen en varias regiones pequeñas (celdas), cada una de las cuales tiene un histograma de las direcciones de su gradiente o también llamada orientación de borde de los píxeles de la celda.

Las entradas de histograma combinadas forman la representación. Para un mejor desempeño cuando no se tienen condiciones de iluminación o sombras, se recomienda hacer una normalización en contraste antes de procesarlas. Esto se hace acumulando una medida local de “energía”

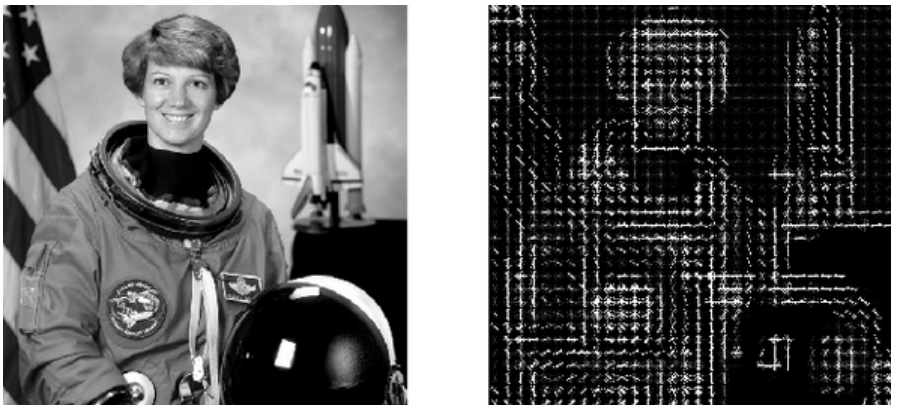

FIGURA 1.7: Histograma de Gradientes Orientados (HOG) aplicado a una imagen en escala de grises

sobre regiones más grandes (bloques) y utilizando los resultados para normalizar todas las celdas en el bloque. Este descriptor de normalización se conoce como descriptor de *Histograma de Gradientes Orientados* (HOG). Dividiendo la imagen con una densa maya (que de hecho se superponen) de descriptores HOG y utilizando el vector de características combinadas en una Máquina de Soporte Vectorial (SVM) convencional es lo que hace que se pueda realizar detección de seres humanos.

1.3. Los MAVs para el seguimiento de objetos

Debido a los últimos avances en la tecnología de los drones, se han desarrollado muchas aplicaciones que permiten realizar diversas tareas, como es el caso del seguimiento de objetos.

Incluso, entre las tareas del seguimiento de objetos se encuentra el seguimiento de personas.

FIGURA 1.8: MeCam de AlwaysInnovating

AlwaysInnovating[20], una empresa de consumibles electrónicos fabricó el dron MeCam (Figura 1.8), el cual graba y transmite el contenido directamente al teléfono inteligente para poderlo transmitir a las redes sociales, y que puede seguir a la persona utilizando comandos de voz ya que no cuenta con control remoto [21].

FIGURA 1.9: Mind4 de AirMind

Un caso muy reciente es el Mind4 (Figura 1.9), fabricado por AirMind [22], el cual puede ser utilizado por esquiadores, personas que practican snowboard, ciclistas y similares, para grabar tomas aéreas de sus actividades desde una altura de 20 m.[23].

1.4. Organización de la Tesis

El resto del presente trabajo de Tesis está organizado de la siguiente manera:

- El **Capítulo 2** presenta las características físicas del MAV y su arquitectura para permitir la comunicación entre la estación de trabajo y el dron utilizado en este proyecto.
- El **Capítulo 3** habla de la manera en la que se puso en funcionamiento la plataforma para operar el MAV aprovechando las propiedades que se describen en el Capítulo 2.
- El **Capítulo 4** explica la aplicación de las técnicas de Visión por computadora descritas en este Capítulo para lograr el seguimiento de un objeto por el dron.
- El **Capítulo 5** nos permite conocer los resultados que se obtienen de la estrategia propuesta en el Capítulo 4, así como las restricciones de este proyecto en particular y el trabajo futuro.

Capítulo 2

El AR.Drone 2.0 y su control

2.1. Introducción

El AR.Drone 2.0 (Figura 2.1) es un cuadricóptero desarrollado por Parrot®, su estructura está conformada por cuatro motores unidos en forma de cruz donde el hardware de radio frecuencia y la batería se encuentran en el centro.

FIGURA 2.1: Cuadricóptero AR.Drone 2.0.

Cada par de motores opuestos giran de la misma forma. Un par gira en el sentido de las manecillas del reloj y el otro en el sentido contrario (Figura 2.2).

2.2. Movimientos del AR.Drone 2.0

El AR.Drone 2.0 tiene cuatro movimientos distintos que hacen posible un sinfín de maniobras en el aire, entre las que destacan las siguientes:

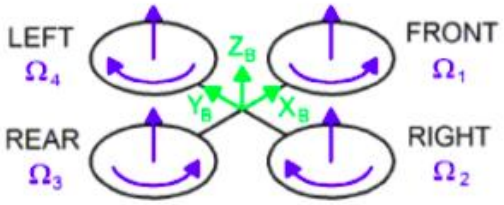

FIGURA 2.2: Motores del AR.Drone 2.0 y el sentido de su giro.

1. Elevación (Throttle)
2. Giro lateral (Roll)
3. Giro hacia adelante (Pitch)
4. Giro sobre el mismo eje (Yaw)

En la Figura 2.3, podemos apreciar los cuatro tipos de movimientos, donde a) eleva o baja el cuadricóptero, b) lo mueve hacia la derecha o lo mueve hacia la izquierda, c) lo mueve hacia adelante o hacia atrás y d) hace que gire sobre su mismo eje en sentido horario o sentido antihorario.

De esta manera, las maniobras son obtenidas cambiando los ángulos de Pitch, Roll y de Yaw.

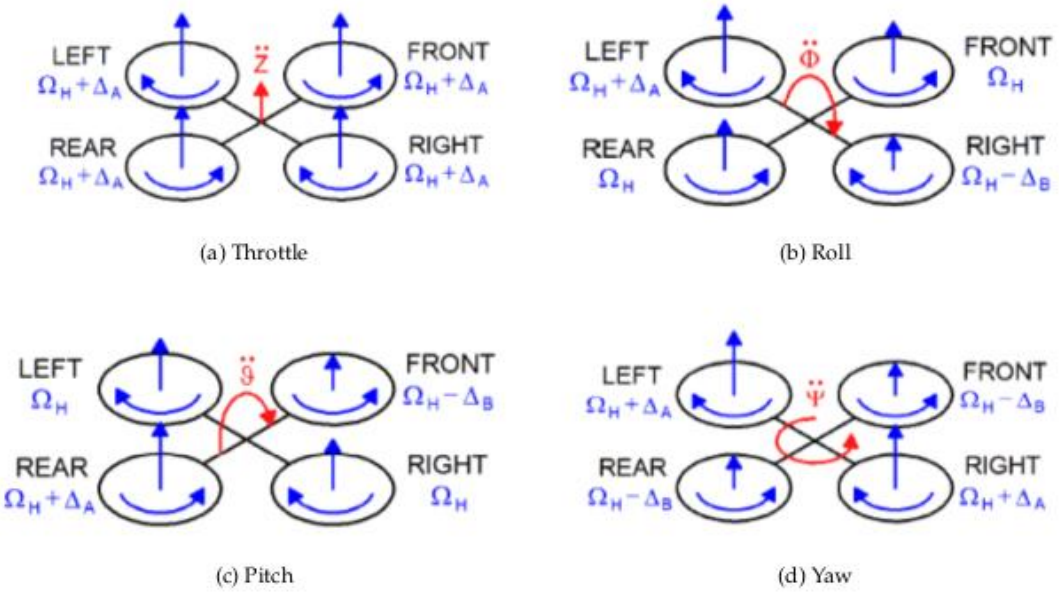

FIGURA 2.3: Movimientos del AR.Drone 2.0 y los cambios en su velocidad de giro.

Variando la velocidad de los motores izquierdo y derecho de forma opuesta (es decir, a uno se le aumenta y al otro se le disminuye) se logra un cambio en el ángulo de Roll y hace que el cuadricóptero se mueva hacia su derecha o su izquierda. Variando la velocidad de los motores delantero y trasero de forma opuesta se logra un movimiento hacia adelante o hacia atrás. Variando la velocidad de cada par de motores de forma opuesta, se logra un giro sobre su propio eje hacia la izquierda o hacia la derecha.

2.3. Sensores de movimiento

El AR.Drone 2.0 tiene varios sensores de movimiento localizados en su receptáculo central.

Cuenta con una IMU, basada en MEMS, de 9 DOF para su estabilización automática con un magnetómetro de 3 ejes.

Tiene un sensor ultrasónico que provee de medidas de altitud para la estabilización y control de la velocidad vertical asistida.

2.4. Video streaming

La cámara frontal es un sensor CMOS con un lente con un ángulo de 90 grados.

El AR.Drone 2.0 codifica y envía las imágenes entrantes al dispositivo huésped. Utiliza imágenes con una resolución de 360p (640×360 píxeles) o de 720p (1280×720 píxeles). La frecuencia de actualización puede ser ajustada entre 15 y 30 FPS y una tasa de transferencia entre 250 $kbps$ y 4 $Mbps$.

2.5. Servicios de comunicación entre el AR.Drone 2.0 y el dispositivo cliente

El AR.Drone 2.0 puede ser controlado por cualquier dispositivo que soporta WiFi.

El control del AR.Drone 2.0 es realizado por tres servicios de comunicación principales.

El control y la configuración del dron es realizado por medio del envío de comandos AT en el puerto UDP 5556. La latencia de transmisión de los comandos de control son críticos para la

experiencia de usuario. Estos comandos tienen que ser enviados de manera regular (usualmente 30 veces por segundo).

La información acerca del dron, como el estado, su posición, velocidad de rotación de los motores, llamado *navdata* (de *navigation data*), son enviados por el dron a su cliente en el puerto UDP 5554.

La secuencia de vídeo es enviada por el AR.Drone 2.0 al dispositivo cliente por el puerto TCP 5555. Las imágenes de este vídeo son codificadas de acuerdo al formato que es discutido en la sección 2.6.1.4 (Encapsulamiento).

Otro canal de comunicación, llamado *control port*, puede ser establecido por el puerto TCP 5559 para enviar información crítica, en oposición a los otros canales de comunicación donde la información puede ser perdida sin efectos peligrosos. Es utilizado para recuperar datos de configuración, y el envío de configuraciones.

2.6. El Control del AR.Drone 2.0

2.6.1. Plataforma para control

En primera instancia, se debe entender como Control del AR.Drone 2.0 a la acción de controlar el cuadricóptero por medio de una plataforma que permita ejecutar las actividades más rutinarias. Esto de acuerdo a los movimientos explicados en la sección 2.2.

Dado que el objetivo principal de este documento es hacer el seguimiento de una persona, no se necesitan modificar los movimientos, ya que con ellos es suficiente el movimiento mediante el envío de los comandos AT.

Una plataforma para hacer las maniobras del AR.Drone 2.0 es mediante [8], la cual permite sin hacer gran modificación del código, controlar remotamente el cuadricóptero desde algún dispositivo programable con una tarjeta de red WiFi con TCP/UDP/IP. El SDK no permite un acceso directo al hardware del drone.

2.6.1.1. Arquitectura del software

La arquitectura del software se encuentra en capas. En la Figura 2.4 se muestran las capas de la arquitectura de una aplicación montada sobre el SDK del AR.Drone 2.0.

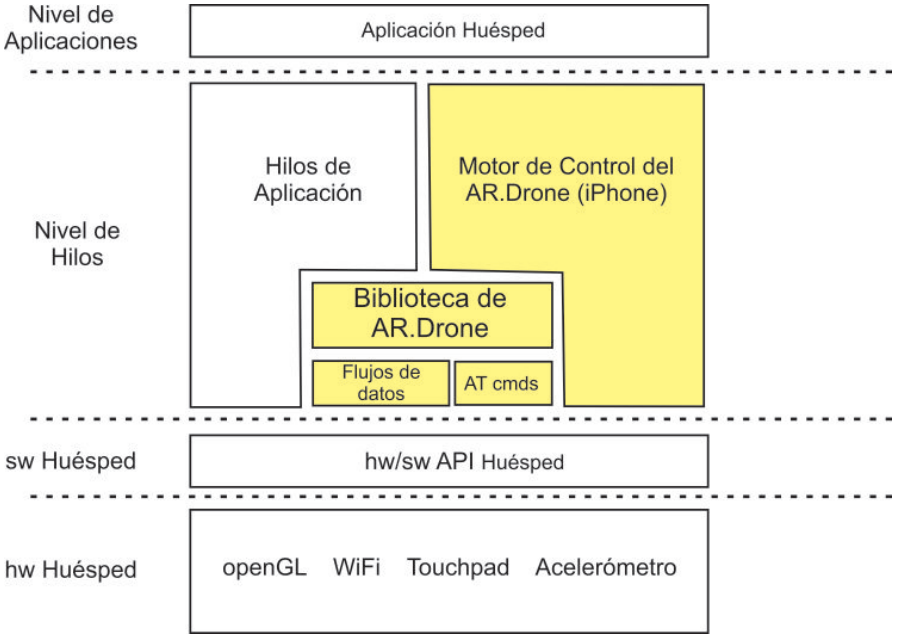

FIGURA 2.4: Capas de la arquitectura del SDK.

Las aplicaciones que se crean para manejar el AR.Drone 2.0 se codifican en en lenguaje de alto nivel C.

2.6.1.2. La biblioteca AR.Drone

La biblioteca AR.Drone se provee actualmente como una biblioteca de código abierto.

Contiene una parte denominada *ARDroneTool*, la cual implementa de manera eficiente los cuatro servicios descritos en la sección 2.5.

Provee:

- un hilo de administración de comandos AT, que recolecta comandos enviados por todos los otros hilos, y los envía de una manera ordenada con una secuencia numérica correcta.
- un hilo *navdata* de administración que recibe automáticamente un flujo de datos de navegación, lo decodifica, y provee a la aplicación del cliente de datos de navegación listos para usarse mediante una llamada a una función.

- un hilo de administración del vídeo, el cual recibe automáticamente el flujo de vídeo y provee a la aplicación de vídeo listo para usar mediante la llamada a una función.
- un hilo para grabación de vídeo, que administra la grabación del flujo en HD y la encapsulación .mp4/.mov.
- un hilo de *control* que maneja solicitudes de otros hilos para enviar comandos seguros del dron.
- un conjunto de hilos para AR.Drone Academy[1], que automáticamente recibe las imágenes (formato .jpg) por FTP.

2.6.1.3. Administración del vídeo

El SDK incluye métodos para administrar el flujo de vídeo. El proceso completo es manejado por un *video pipeline* (conducto), construido como una secuencia de *stages* (etapas) el cual desempeña pasos básicos, como recibir los datos del vídeo de un socket, decodificar los marcos y desplegarlos visualmente.

2.6.1.4. Flujo de vídeo

El AR.Drone 2.0 utiliza códecs de vídeo estándar, con una encapsulación para administrar el flujo por la red.

Códecs de vídeo

El AR.Drone 2.0 utiliza la norma H264 (MPEG4.10) para el flujo de vídeo de alta calidad y la grabación de vídeo.

Los siguientes parámetros pueden ser ajustados para el flujo en vivo H264:

- FPS: Entre 15 y 30.
- Velocidad de transferencia: Entre 250 kbps y 4 Mbps.
- Resolución: 360p (640x360 píxeles) o 720p (1280x720 píxeles).

[1] AR.Drone Academy es un servidor que recibe los vídeos que los usuarios quieren compartir con los demás usuarios del AR.Drone en el mundo.

Mientras se realiza grabación de vídeo, el codificador por hardware H264 no está disponible para el flujo de vídeo en vivo, en consecuencia el AR.Drone 2.0 utiliza un codificador visual por software MPEG4.2 para el flujo de vídeo en vivo.

Los siguientes parámetros pueden ser ajustados para el flujo de vídeo en vivo MPEG4.2:

- FPS: Entre 15 y 30.
- Velocidad de Transferencia: Entre 250 kbps y 1 Mbps.

Encapsulamiento del vídeo en la red

Para la transmisión por la red, los frames (marcos) de vídeo son enviados por medio de encabezados de código, que contienen mucha información acerca de cada frame.

Estos encabezados contienen información valiosa y contienen en una estructura los siguientes datos:

- Tamaño del encapsulamiento
- Ancho del flujo codificado (píxeles)
- Largo del flujo codificado (píxeles)
- Posición del frame en el flujo
- Tiempo en milisegundos
- Tipo de Frame (puede ser I-Frame o P-Frame)
- Total de rebanadas de imagen que componen el frame actual
- Posición de la rebanada de imagen en el frame

Transmisión del flujo de vídeo

El flujo de vídeo del AR.Drone 2.0 es transmitido por el puerto TCP 5555. El dron comenzará enviando un frame inmediatamente cuando un cliente se conecta al socket.

Un frame puede ser enviado en varios paquetes TCP, y por lo tanto debe ser ensamblado de nuevo por la aplicación antes de llegar al decodificador de vídeo.

Grabación del flujo de vídeo

La grabación de vídeo utiliza el puerto TCP 5553 para transmitir los frames H264-720p. Este flujo de datos es deshabilitado cuando la aplicación no realiza la grabación.

Este flujo utiliza el mismo encapsulamiento para su transmisión al igual que el flujo de vídeo en vivo.

La conversión del códec H264 y un archivo .mov/.mp4 es realizada por la aplicación.

2.6.2. ROS

ROS (Robot Operative System) es un framework para escribir software para robots de código abierto. Es un conjunto de herramientas, bibliotecas, y convenios que tienen por objeto simplificar la tarea de crear comportamientos complejos y robustos a través de una amplia variedad de plataformas robóticas [24].

ROS ha permitido crear una extensa cantidad de algoritmos que ayudan a manipular diversos modelos de dispositivos robóticos que se encuentran en el mundo científico. Entre estos frameworks se encuentra el controlador *ardrone_autonomy*.

2.6.2.1. ardrone_autonomy

ardrone_autonomy es un controlador de ROS para el cuadricóptero AR.Drone 1.0 y 2.0. Éste controlador está basado en el SDK del AR.Drone [8, 25]. Es un controlador desarrollado por el grupo Autonomy Lab [26] de la Universidad Simon Fraser [27]. Cuenta con su propia documentación [28].

2.6.3. RT_ARDrone

Sebastien Druon es un investigador de la Universidad de Montpellier, Francia, que ha desarrollado entre otros controladores de código abierto para robots, el *RT_ARDrone* [29], en colaboración con otros investigadores, el cual es un framework para sistemas Linux que permite controlar el AR.Drone 2.0 de manera fácil y rápida, desarrollado en lenguaje C, el cual puede encontrarse en su cuenta de Github. Es este framework el que se ha utilizado en este trabajo de tesis, y a través del cual se ha estructurado el proyecto como se describe en el Capítulo 3 y realizado los avances mostrados en el Capítulo 4.

Capítulo 3

Operación y Administración de las Funciones del MAV

La plataforma crea una estructura llamada `ARDrone` que contiene los cuatro servicios principales mencionados en 2.5, que a su vez cada una es una estructura en lenguaje C.

```
typedef struct {

        ATStream*       at_stream;
        NavDataStream*  navdata_stream;
        VideoStream*    video_stream;
        ControlStream*  ctrl_stream;

} ARDrone;
```

3.1. Movimiento del ARDrone 2.0

Como se mencionó en 2.5, por medio de comandos AT son enviadas las instrucciones de control y de configuración al MAV para principalmente, moverlo. Estos comandos son cadenas de texto.

La estructura `ATStream` contiene una cadena de caracteres de nombre `command`, donde guarda el comando AT con nombre correspondiente a las funciones de movimiento, contenidos en la Tabla 3.1.

```
typedef struct {

        int socket;
        struct hostent* host;
        struct sockaddr_in addr;

        int seq;                // Numero de secuencia para comandos AT

        pthread_t thread;
        pthread_mutex_t mutex;          //Este mutex (exclusion mutua)
    protege el acceso a at_command y los campos at_argument

        char at_command[80];                    // Comando AT ejecutado en
    este momento
        char at_argument[80];

} ATStream;
```

Movimiento	**Comando AT**
Take off/Land	`AT*REF=%d, %d`
Move	`AT*PCMD=%d, %d, %d, %d, %d`

TABLA 3.1: Comandos AT para mover el MAV

La sintaxis para despegar y aterrizar es el comando: `AT*REF=%d, %d<CR>`, donde

- Argumento 1: es un número de secuencia contenido en la estructura `ATStream` y el segundo argumento es un número n de 32 bits, $n \in [0, 2^{32} - 1]$.
- entero de 32 bits que representa una orden de control.

Para hacer el *despegue* (Takeoff) y el *aterrizaje* (Land) del MAV, utilizamos los comandos AT que comienzan con la cadena `AT*REF=`, seguida de un número de secuencia y la cadena `290718208` ($10001010101000000001000000000_2$) para el despegue, o la cadena `290717696` ($10001010101000000000000000000_2$) para el aterrizaje, con el bit 9 en 1 y en 0 respectivamente.

Para realizar el *movimiento* del MAV utilizamos básicamente los comandos que comienzan con la cadena `AT*PCMD=`, seguida de los argumentos que ejecutan los movimientos descritos en 2.2,

los cuales hacen un cambio en la velocidad de los motores, siendo números de punto flotante que va entre 1 y -1.

Así, la sintaxis para el comando es: `AT*PCMD=%d,%d,%d,%d,%d,%d<CR>`, donde `%d` representa cada argumento y:

- Argumento 1: número de secuencia del comando AT
- Argumento 2: bandera de habilitación de envío de comandos progresivos
- Argumento 3: desplazamiento izquierda - derecha, en el rango $[-1, \ldots, 1]$
- Argumento 4: desplazamiento adelante - atrás, en el rango $[-1, \ldots, 1]$
- Argumento 5: velocidad vertical, en el rango $[-1, \ldots, 1]$
- Argumento 6: velocidad angular, en el rango $[-1, \ldots, 1]$

Para movimientos con el teclado, dependiendo la dirección a la que se necesita moverse, se asignaron valores de -1 o de 1, por lapsos de tiempo cortos ($100\ ms$) por cada vez que se presiona una tecla. Esto genera un movimiento del MAV brusco aunque por muy poco tiempo si se aprieta una sola vez la tecla, pero permite mover de manera muy rápida al MAV si la tecla se deja presionada por un momento.

Para el proyecto, se definieron las teclas de la Tabla 3.2, que generarán el movimiento a voluntad por parte del operador en un teclado latinoamericano.

Tecla	**Movimiento**	**Inclinación**
W	Pitch$+$	$\vartheta > 0$
A	Roll$-$	$\phi < 0$
S	Pitch$-$	$\vartheta < 0$
D	Roll$+$	$\phi > 0$
Flecha Arriba	Throttle$+$	$\zeta > 0$
Flecha Abajo	Throttle$-$	$\zeta < 0$
Flecha Izquierda	Yaw$+$	$\psi > 0$
Flecha Derecha	Yaw$-$	$\psi < 0$
Inicio	Take off	
Fin	Land	

TABLA 3.2: Teclas para mover el MAV

Para efectos de la realización del proceso explicado en el Capítulo 4, la **Barra espaciadora** activa o desactiva al reconocimiento y seguimiento, de modo que, al encontrarse desactivados,

el MAV queda solamente en espera de las acciones contenidas en la Tabla 3.2 que el controlador desde la estación de trabajo le mande hacer.

3.2. Captura de navegación

Como se menciona en la sección 2.5, existe un canal de comunicación llamado *navdata*, a través del cual se recibe la información referente al estado del AR Drone 2.0.

La estructura `NavDataStream` contiene a su vez otra estructura llamada `NavData`, que contiene los valores de inclinación que se encuentran en la tabla 3.2 (`thetha`: θ o ϑ, `phi`: ϕ y `psi`: ψ).

```
typedef struct {

        int socket;
        struct sockaddr_in      addr;
        struct hostent* host;

        pthread_t thread;
        pthread_mutex_t mutex;

        NavData current_navdata;

} NavDataStream;
```

`NavData` también contiene los valores de velocidad de los motores en `vx`, `vy` y `vz`, la altitud y el nivel de batería del MAV (`bat`).

```
typedef struct {

        uint32_t state;
        uint32_t bat;

        float theta;
        float phi;
        float psi;

        int32_t altitude;

```

```
12          float vx;
13          float vy;
14          float vz;
15
16  } NavData;
```

La información importante que se recibe del MAV en este trabajo son:

La cantidad de energía de la batería, que cuando llega a un determinado nivel bajo, el MAV no acciona los motores aunque le sea enviado el comando correcto.
El ángulo de inclinación del MAV, que determina la dirección del movimiento del MAV.
El valor del magnetómetro, que permite saber hacia qué dirección se encuentra posicionado respecto del Norte geográfico.
La altura con respecto del piso, que solo está activado cuando los motores están en funcionamiento.
La velocidad de los motores, que determina la inclinación y por consiguiente el movimiento del MAV.

3.3. Captura de video

Se obtiene el stream de video descrito en 2.6.1.4, el cual es convertido en una imagen RGB de 24 bits, es decir, 8 bits por cada canal de los colores Rojo (Red), Verde (Green) y Azul (Blue).

La estructura `VideoStream` contiene la estructura `AVCodec` correspondiente a la norma que define el códec MPEG4.10 mencionado en 2.6.1.4 y la estructura `AVFrame` que contiene la imagen que se envía por el puerto 5555 a la estación de trabajo.

```
1   typedef struct {
2
3           // Propiedades del socket
4
5           int socket;
6           struct hostent* host;
7           struct sockaddr_in      addr;
8
9           // Propiedades del hilo de ejecucion
10
```

```
11          pthread_t thread;
12          pthread_mutex_t mutex;
13
14          // Decodificador H264
15
16          AVCodec* pCodecH264;
17          AVCodecContext* pCodecCtxH264;
18          AVDictionary* opts;
19          AVPacket packet;
20
21          AVFrame* picture;               // last picture captured. protected
        by mutex
22
23          int bool_new_picture;   // True if one picture was captured since
        last read
24
25          // Conversion a RGB24
26
27          struct SwsContext* sws_ctx;
28          AVFrame* RGBPicture;
29
30  } VideoStream;
```

Cada una de estas imágenes o frames de video contenidas en `AVFrame` son procesadas por la computadora para realizar las tareas descritas en el Capítulo 4.

Capítulo 4

Visión por computadora con el MAV

La visión por computadora o visión artificial es el mecanismo por el cual el MAV puede ejecutar tareas complejas, sobre todo cuando un cuadricóptero tiene que desarrollar actividades de manera autónoma.

Como se ha mencionado en 2.4, el MAV cuenta con una cámara que le permite captar las imágenes del exterior, después enviarlas a la computadora y en ella realizar el procesamiento del vídeo. En este capítulo se presenta la manera en la que el MAV observa el mundo para poder ejecutar sus tareas.

4.1. Umbralización

El proceso de Umbralización es una técnica de Segmentación basado en los valores de píxel, uno de los tipos de segmentación mencionados en 1.2.2.1, que consiste en separar en una imagen a los objetos que forman parte de nuestro interés o que se pretenden distinguir de entre los demás objetos en una escena basándose en el color de cada píxel. Es un tipo de reconocimiento de objetos basado en color. El proceso separa por medio de los valores de color de los canales de cada píxel a aquéllos que tienen un valor entre un intervalo adecuado de sus canales.

Para nuestro caso, cada canal (R, G y B) contiene un rango de 0 a 255. Dependiendo del color de los objetos, se determinará si forman o no parte del intervalo y serán resaltados y se conocerá su ubicación exacta en la matriz de píxeles, los que no forman parte del intervalo, ya sea que su valor esté por debajo o por arriba de este, se discriminarán (que comúnmente los transforman en

píxeles negros). En la Figura 4.1, se puede apreciar que un objeto de color rojo es detectado por la cámara del MAV.

FIGURA 4.1: Detección de objeto con una segmentación basada en color

4.2. Histograma de Gradientes Orientados

Para el reconocimiento de personas con el ARDrone 2.0, se ha utilizado con éxito el Histograma de Gradientes Orientados (Histogram of Oriented Gradients, HOG).

Este método, como se menciona en 1.2.2.2, se basa en evaluar histogramas locales normalizados de las orientaciones de gradiente de una imagen.

Para calcular el Histograma de Gradientes Orientados se ha utilizado el Algoritmo 1.

Algoritmo 1 Calcular el Histograma de Gradientes Orientados de la siguiente forma

1: (Opcional) Normalización de la imagen
2: Calcular el gradiente de la imagen en x y y
3: Calcular los histogramas de los gradientes
4: Normalización a través de los bloques
5: Recolección de datos para procesar en el clasificador

Primero se aplica una normalización global para reducir la influencia de los efectos de ilumina-ción. Por lo general se utiliza corrección gamma

$$r' = q\left(\frac{r}{q}\right)^{\gamma}$$

Donde γ es un real positivo, $\gamma \in (0, 1]$, $r \in [0, q]$ y q son los niveles de cuantización, generalmente $q = 256$.

Ésta corrección ayuda a reducir los efectos de las sombras y la variación de la iluminación.

La segunda etapa calcula los gradientes de las imágenes, los cuales se pueden apreciar en la Figura 4.2. Los gradientes capturan el contorno, las siluetas e información referente a las texturas, aunque proveen resistencia adicional a las variaciones de iluminación. El canal de color dominante localmente es el que se utiliza, el cual provee invariación de color en una gran extensión.

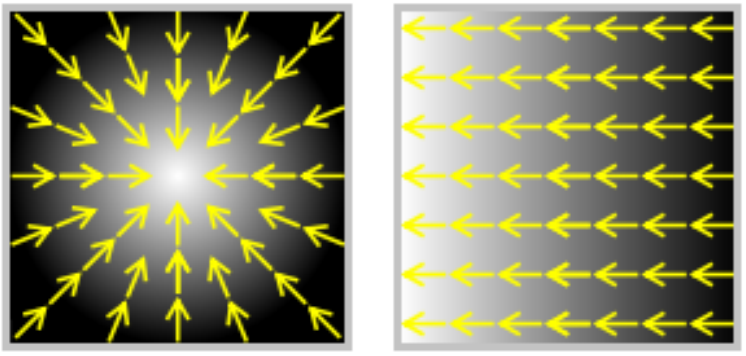

FIGURA 4.2: Gradientes en una imagen en escala de grises

La tercera etapa tiene como objetivo producir una codificación que es sensible al contenido de los cuadros locales sin dejar de ser resistente a pequeños cambios en la postura o la apariencia. El método adoptado junta la información de orientación de las gradientes de la misma forma que SIFT [1].

La imagen está dividida en regiones pequeñas, llamadas "celdas" como las que se muestran en la Figura 4.4. Por cada celda se obtiene un histograma del gradiente o también llamados *orientación de borde* de todos los píxeles de la celda. Un histograma de los canales de color de una imagen son los que se pueden observar en la Figura 4.3 (b). Este histograma combinado a nivel de celda forma el "histograma de orientación".

Cada histograma de orientación divide el rango de ángulo del gradiente en un predeterminado número de partes, como en la Figura 4.5. Las magnitudes en el gradiente de los píxeles en la celda son utilizados para conformar el histograma de orientación.

[1] SIFT es un algoritmo de Visión por Computadora para detectar y describir características locales en imágenes

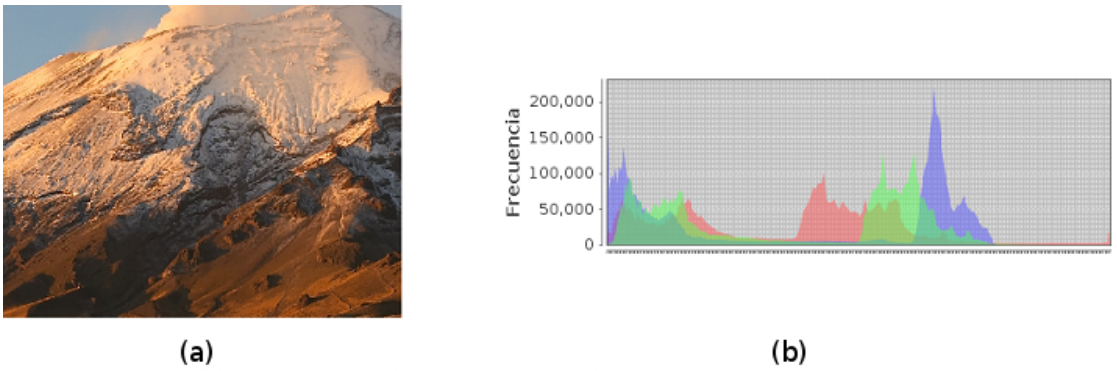

FIGURA 4.3: **(a)** Imagen de la cual se obtiene su histograma en *(b)* **(b)** Histograma de la imagen *(a)*

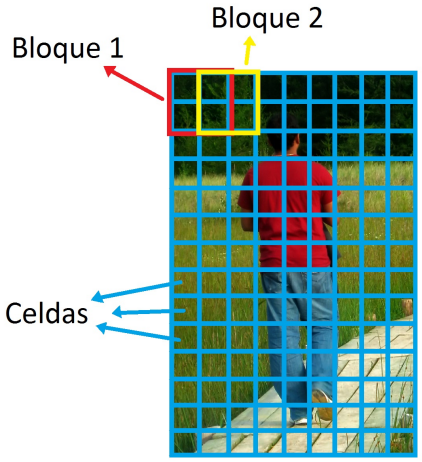

FIGURA 4.4: Bloques y Celdas de la imagen

La cuarta etapa ejecuta una normalización, que toma grupos locales de celdas. La normalización introduce una mejor invarianza a la iluminación, sombras y contraste de los bordes. Se realiza mediante la acumulación de una medida del histograma local de "energía" sobre los grupos locales de celdas llamados "bloques" (Figura 4.4). El resultado es utilizado para normalizar cada celda en el bloque.

Típicamente cada celda individual es compartida por varios bloques, pero su normalización es dependiente del bloque y por lo cual es distinta. La celda entonces aparece varias veces en el vector de salida con diferentes normalizaciones. Esto podría parecer redundante pero mejora el rendimiento. El descriptor de bloque normalizado es el conocido como *descriptor HOG*.

La última etapa recoge los descriptores HOG de todos los bloques de una densa red de superposición de bloques que están en la detección de la imagen en un vector de características

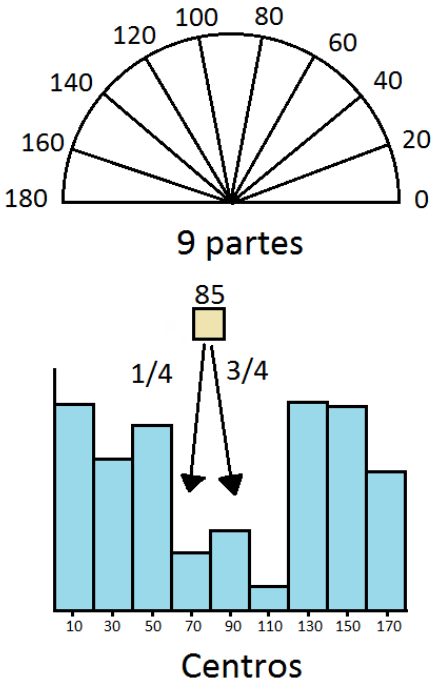

FIGURA 4.5: Rangos de división de los ángulos de Orientación

combinado para introducirlo en una Máquina de Soporte Vectorial, en la cual se realiza un entrenamiento para obtener un promedio de gradiente que será utilizado para reconocer al objeto en cuestión, como se puede ver en la Figura 4.6 (a), donde se puede notar que la silueta tiene una forma muy simétrica.

Es a partir de ésta etapa como la presente Tesis lleva por nombre *Seguimiento de objetos*, debido a que puede ser una persona u otro tipo de figuras con contorno definidos parecidos entre esos objetos los que se pueden ingresar a la SVM, para poder reconocer otras siluetas y obtener un promedio de gradiente diferente al de la Figura 4.6 (a).

Para la detección de personas con el MAV, se realiza una división con celdas de 8×8 píxeles, ya que entre más pequeña sea la celda mejor será su eficiencia (no necesariamente siempre sucede así), aunque su tiempo de procesamiento será grande. A pesar de esto, no puede existir una celda tan pequeña debido a que puede resultar en demasiados falsos positivos.

Una vez aplicado, el Histograma de Gradientes Orientados produce buenos resultados inclusive cuando hay muchas personas en una imagen, como se observa en la Figura 4.8

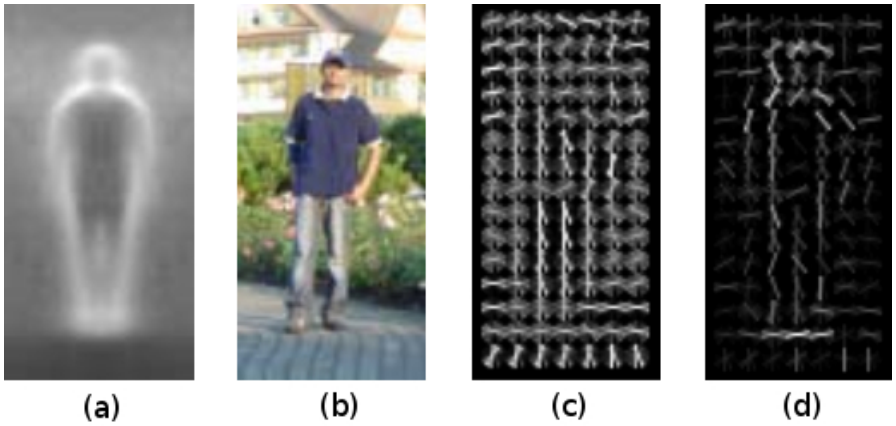

FIGURA 4.6: **(a)** Promedio de gradiente de las imágenes de entrenamiento. **(b)** Imagen de prueba. **(c)** Resultado del HOG ejecutado. **(d)** Gradientes una vez que se obtienen los pesos de la SVM

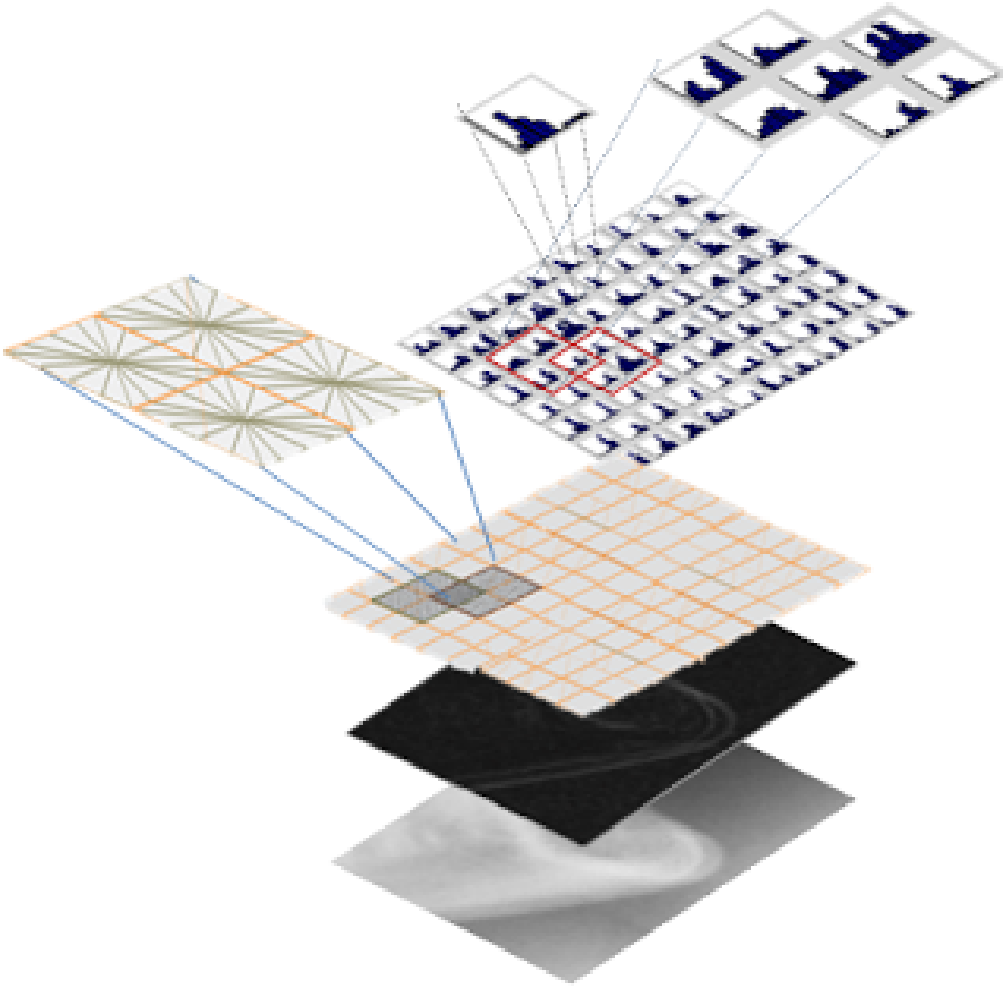

FIGURA 4.7: Etapas del Histograma de Gradientes Orientados

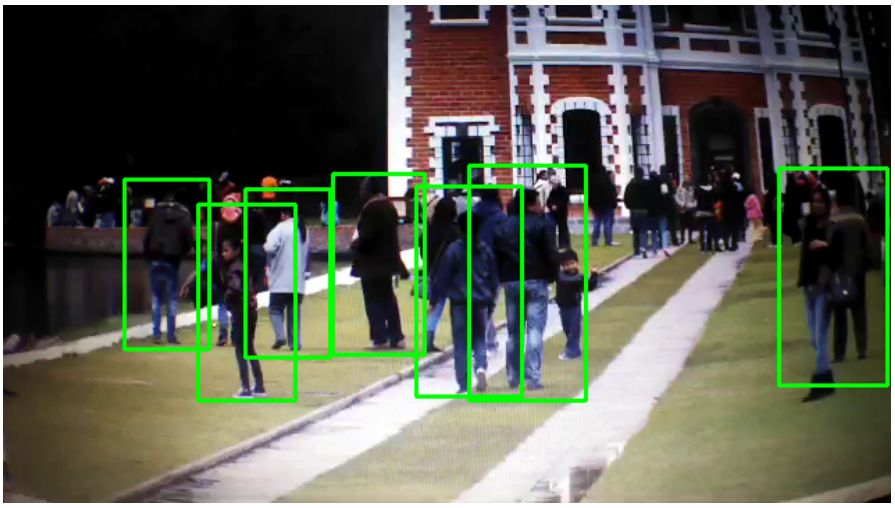

FIGURA 4.8: Histograma de Gradientes Orientados aplicados desde la cámara del MAV

4.3. Filtro de partículas

Una vez que se realiza la detección de personas, se ha aplicado el Filtro de Partículas para hacer el seguimiento de la persona.

Basado en la posición x_t que tiene en el instante actual la persona, se hace una predicción de la siguiente posición x_{t+1}.

A continuación se realiza el proceso de actualización de los pesos de las partículas utilizando como componente la función gaussiana descrita en la ecuación 4.1, para obtener los datos de la posición de la persona en cada una de las imágenes.

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right] \tag{4.1}$$

Si x_R es el ruido de la medición y es igual a la varianza σ^2, entonces $\sigma^2 = x_R$ y la covarianza $\sigma = \sqrt{x_R}$.

t es la observación actual z en el instante t, por lo cual $t = z$.

Una vez que la posición de las partículas son actualizadas, se actualizan las observaciones z_i^{update} para cada partícula i, por lo cual $\mu = z_i^{update}$, $i = 1 \ldots 30$

De esta forma, la ecuación 4.1 queda como la ecuación 4.2

$$\frac{1}{\sqrt{x_R}\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{z - z_i^{update}}{\sqrt{x_R}}\right)^2\right] \tag{4.2}$$

Que al simplificar la ecuación 4.2 obtenemos la ecuación 4.3.

$$\frac{1}{\sqrt{2\pi x_R}} \exp\left[\frac{-\left(z - z_i^{update}\right)^2}{2x_R}\right] \tag{4.3}$$

La ecuación 4.3 es la que se utiliza en cada iteración durante todo el proceso de seguimiento.

FIGURA 4.9: Filtro de partículas aplicado al seguimiento de una persona con el MAV AR.Drone 2.0

En la Figura 4.9, se puede observar el Filtro de partículas cuando el MAV va siguiendo a una persona que es reconocida con el HOG. En color Fucsia se ven las partículas, en color Verde se plasma una línea que recopila los estados que han sucedido en el tiempo que se realiza el seguimiento, con una cruz Roja se observa el estado actual de la persona y con una cruz Blanca se observa el estado actual que el Filtro de Partículas predice.

Capítulo 5

Resultados

Este Capítulo tiene el objetivo de dar a conocer los logros alcanzados en cuanto al seguimiento de personas por el ARDrone 2.0, con las técnicas aplicadas sobre el MAV descritas en el Capítulo 4.

5.1. Pruebas

Para la umbralización se tienen buenos resultados siempre y cuando el objeto a seguir tenga el color que fue definido dentro del proceso en algún lugar o momento, como puede ser, en el vestuario de una persona, sin embargo, la umbralización no es buena cuando el objeto presenta cambios en su iluminación, con lo cual puede parecer tener un color distinto del que originalmente es o del que se veía en otro momento. Otro factor importante es que puede reconocer otros objetos que tienen el mismo color que el que se definió como el objeto a seguir y eso provoca que pierda constantemente el objetivo. Como único medio de reconocimiento no es bueno, siempre será mejor combinarlo con alguna otra técnica de reconocimiento de objetos.

Se ha aplicado el proceso de *Histograma de Gradientes Orientados* para el reconocimiento de personas, obteniendo buenos resultados de reconocimiento, como se observa en la Figura 5.1, inclusive cuando se encuentran varias siluetas en la imagen que captura el Vehículo Aéreo No Tripulado (Figura 5.2).

Dado que la tarea de reconocimiento de personas es pesada computacionalmente, el reconocimiento no se hace en el mismo proceso de ejecución del proceso que recibe el stream del vídeo

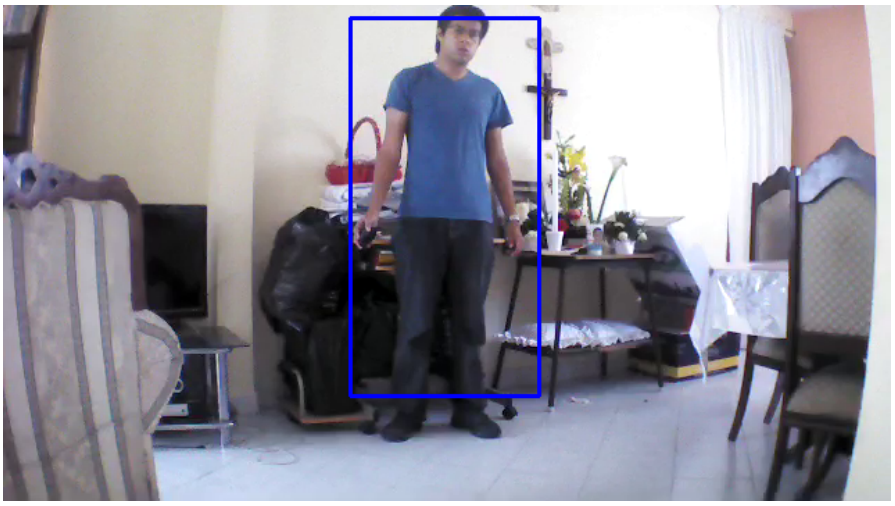

FIGURA 5.1: Histograma de Gradientes Orientados (HOG) que reconoce la silueta humana

del MAV, ya que entre mayor es la imagen que se recibe, más tiempo tarda en obtenerse resultados adecuados, y puede producir que se obtengan menos FPS, debido a que estaría retrasando el stream y se puede volver muy perceptible para el ojo humano.

FIGURA 5.2: Histograma de Gradientes Orientados (HOG) que reconoce varias personas

Para imágenes muy pequeñas, del orden de 64×128 píxeles, el reconocimiento de una persona tarda en promedio 40 ms. Para la imagen que se procesa desde el MAV, el tiempo que tarda en hacer el reconocimiento es de 200 ms. en promedio, siendo este valor muy perceptible y si fuera aplicado en el mismo proceso de ejecución, el vídeo se vería pausado.

FIGURA 5.3: Seguimiento de una persona por el MAV AR.Drone 2.0

En la Figura 5.3 se observa el momento en el que el MAV sigue a una persona y en la Figura 5.4 se puede apreciar el funcionamiento del Filtro de Partículas y el Histograma de Gradientes Orientados para seguir y reconocer a la persona respectivamente.

FIGURA 5.4: Aplicación del Filtro de Partículas y el Histograma de Gradientes Orientados por el MAV AR.Drone 2.0

5.2. Restricciones del problema

El MAV puede ejecutar sus acciones de manera exitosa siempre y cuando cumpla con los siguientes puntos:

- Debe seguir a su objetivo en condiciones de iluminación natural del día.
- Deben existir pocas siluetas de seres humanos en la imagen, para evitar redundancia en la búsqueda de la persona central para seguirla.

Cuando no se cumple alguno de los puntos anteriores, los resultados pueden no ser muy buenos y el comportamiento del MAV es impredecible.

Capítulo 6

Conclusiones

Se ha demostrado que el Histograma de Gradientes Orientados hace un reconocimiento muy bueno de varias siluetas de personas en la escena, sin embargo, aún falta por acoplar las órdenes que se envían al MAV con el seguimiento del objeto central de la escena, que si bien con el Filtro de Partículas fácilmente podemos saber cuál objeto se debe seguir, las órdenes de movimiento aún son un tanto complejas debido a los factores externos muy variables que inherentemente se encuentran con los Vehículos Aéreos No Tripulados, como lo es el viento.

Sin embargo, la estrategia seguida en este ámbito ha sido exitosa debido a que se ha logrado implantar un comportamiento inteligente en el vehículo que le permite valerse por sí mismo en la tarea de seguimiento de una persona. Dicha estrategia puede ser aplicada en vehículos de diferente escala que cuenten con las características de sensores similares a las del AR.Drone 2.0 y con ello poner en marcha aplicaciones en diversos campos, como el científico o el del entretenimiento.

Los vehículos aéreos no tripulados de no tener mucho renombre hace dos años, tienen un auge impresionante en este 2015 y seguirán avanzando con tecnologías cada vez más sofisticadas y de tener un uso más versátil. La comunidad científica de todo el mundo está preparando bases sólidas para resolver los más diversos problemas que aquejan a la humanidad. Es solo cuestión de imaginar y aplicar nuestros conocimientos de manera ética.

6.1. Trabajo futuro

Uno de los aspectos más significativos que saltan a la vista es la dificultad de estabilidad en los movimientos del MAV, sobre todo en condiciones de viento fuerte. La plataforma permite al vehículo permanecer en su posición cuando no tiene órdenes de moverse, sin embargo, es necesario implementar un ciclo de control que al presentar cambios en su posición en la tercera dimensión por agentes externos, logre regresar a la posición que se encuentra, tomando provecho de los sensores con los que se cuentan como la cámara que apunta al piso y el sensor ultrasónico. Es por eso que para el seguimiento en condiciones más extremas, es necesaria la intervención sobre el proyecto para lograr mejores resultados y la continuidad del proyecto.

Uno de las tareas en la que conviene trabajar fuertemente es en el seguimiento del objeto cuando se encuentran muchas personas (siluetas) u objetos del mismo tipo en la escena, sin perder de vista el objetivo principal.

Referencias

[1] K. P. Valavanis. *Advances in Unmanned Aerial Vehicles - State of the Art and the Road to Autonomy*. Dordrecht, The Netherlands, 2007.

[2] William Thielicke. Shrediquette - the award-winning multirotor mav by william thielicke, 2008. URL http://shrediquette.blogspot.de/p/shrediquette-bolt.html. Último acceso: junio 2014.

[3] K. P. Valavanis. *Advances in Unmanned Aerial Vehicles - State of the Art and the Road to Autonomy*. Springer, Dordrecht, The Netherlands, 2007.

[4] P. M. Lambermont. *Helicopters and autogiros of the world.* Casell, Londres, 1970.

[5] Syma x3 quadcopter - out in house review, . URL http://syma107.com/syma-x3-quadcopter/. Último acceso: junio 2014.

[6] Massachussetts Institute of Technology. Aerospace controls laboratory | projects, . URL http://acl.mit.edu/projects/. Último acceso: junio 2014.

[7] T. Krajník, V. Vojtech, D. Fiser, and J. Faigl. Ar-drone as a platform for robotic research and education. 2011.

[8] Parrot®. Ardrone open api platform, . URL https://projects.ardrone.org/. Último acceso: junio 2014.

[9] Parrot®. Parrot for developers, . URL developer.parrot.com/. Último acceso: octubre 2015.

[10] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. 2011.

[11] Onboard software - eth pixhawk: Mav computer vision, . URL https://pixhawk.ethz.ch/software/start. Último acceso: junio 2014.

[12] B. University. ardrone brown - brown-ros-pkg - brown university repository for ros packages - google project hosting. URL https://code.google.com/p/brown-ros-pkg/wiki/ardrone_brown. Último acceso: junio 2014.

[13] Massachussetts Institute of Technology. Portal sct:.: Regula la sct el uso de aeronaves no tripuladas (drones), . URL http://www.sct.gob.mx/despliega-noticias/article/regula-la-sct-el-uso-de-aeronaves-no-tripuladas-drones/. Último acceso: octubre 2015.

[14] ExpoDrone MX. Expodrone mx. URL http://expodrone.mx. Último acceso: octubre 2015.

[15] FayerWayer. Se anuncia expodrone mx 2015, la primera exhibición de drones en méxico - fayerwayer. URL https://www.fayerwayer.com/2015/02/expodrone-mx-exposicion-drones-mexico/. Último acceso: octubre 2015.

[16] MacCormick and A. Blake. Probabilistic exclusion and partitioned sampling for multiple object tracking. *International Journal of Computer Vision*, 39(1):57–71, 2000.

[17] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/nongaussian bayesian state estimation. *IEE Proc.-F*, 140(2), 1993.

[18] J.J. Báez Rojas, M.L. Guerrero, J. Conde Acevedo, A. Padilla Vivanco, and G. Urcid Serrano. Segmentación de imágenes de color. *Revista Mexicana de Física*, 50(6):579–587, Diciembre 2004.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005. URL http://lear.inrialpes.fr/pubs/2005/DT05.

[20] AlwaysInnovating. Always innovating: Introducing the mecam, . URL https://www.alwaysinnovating.com/home/. Último acceso: octubre 2015.

[21] AlwaysInnovating. Always innovating: Introducing the mecam, . URL https://www.alwaysinnovating.com/products/mecam.htm. Último acceso: octubre 2015.

[22] AirMind. Airmind. URL http://www.airmind.co/. Último acceso: diciembre 2014.

[23] CNN. ¿es este el primer 'drone acosador' del mundo? URL http://cnnespanol.cnn.com/2014/11/30/es-este-el-primer-drone-acosador-del-mundo/. Último acceso: diciembre 2014.

[24] ROS.org. Ros.org | powering the world's robots. URL http://www.ros.org/about-ros/. Último acceso: junio 2015.

[25] Parrot. ar drone - parrot for developers. URL http://developer.parrot.com/ar-drone.html. Último acceso: octubre 2015.

[26] AutonomyLab. Autonomy lab, . URL http://autonomylab.org. Último acceso: junio 2015.

[27] SFU. Sfu.ca - simon fraser university. URL http://www.sfu.ca/. Último acceso: junio 2015.

[28] AutonomyLab. ardrone_autonomy ardrone_autonomy indigo-devel documentation, . URL http://ardrone-autonomy.readthedocs.org/. Último acceso: junio 2015.

[29] GitHub Sebastien Druon sebastien.druon@univ-montp2.fr. Iut-beziers/rt_ardrone. URL https://github.com/IUT-Beziers/RT_ARDrone. Último acceso: junio 2015.