Instituto de Computación, Facultad de Ingeniería

Universidad de la República

Montevideo, Uruguay

Planificación e instrumentación de vuelo de una flotilla de drones

Autores:
Santiago Díaz
Bruno Garate

Supervisores:
Santiago Iturriaga
Sergio Nesmachnow

Diciembre 2018

Resumen

La presencia en aumento de drones en el mercado ha permitido la existencia de vehículos con cada vez mayor capacidad de vuelo y procesamiento a bordo a un bajo precio, haciendo que su uso sea de creciente interés en el ámbito comercial e industrial. Este proyecto aborda la utilización de una flotilla de drones autónomos aplicada a la vigilancia de un predio. La solución presentada hace foco en la coordinación de la flotilla, el procesamiento de imágenes y la navegación por marcadores así como la capacidad de recarga de forma completamente autónoma. El análisis experimental demuestra la viabilidad de la solución y propone líneas de mejora, evidenciando la complejidad que implica crear una solución aplicable a un escenario real.

Agradecimientos

En primer lugar, damos las gracias a nuestros supervisores de proyecto por el soporte y buenos consejos brindados a lo largo del trabajo. Sin lugar a dudas, también a nuestros familiares y amigos por el eterno apoyo y confianza, no sólo a lo largo de nuestra carrera académica sino de nuestra vida. Además, agradecemos a Geocom por haber provisto los materiales para la realización del proyecto.

Índice

Índice de figuras

1. Introducción

Cada vez existen más y mejores vehículos aéreos no tripulados (también llamados drones) en el mercado (Zeng et al. 2016; UAV Coach 2018). Este tipo de aeronaves tienen múltiples usos además del uso recreativo, como la fotografía y filmación aérea, la seguridad, búsqueda y rescate en desastres naturales, topografía, fumigación de cultivos, detección de incendios forestales, seguimiento del tráfico, entre otros (Villasenor 2014). Por otra lado a pesar de que el concepto de vehículos autónomos no es nuevo, su uso va en aumento debido a que el avance tecnológico a aparejado la existencia de cada vez mejores sensores y mayor capacidad de cómputo, lo que permite crear vehículos autónomos más sofisticados (IET 2014).

La gran oferta de drones y la demanda en aumento en el área de seguridad hace que una solución de vigilancia automatizada sea de gran interés (FTI Consulting 2014; J. Cruz 2009). Buena parte las tareas de vigilancia constan de recorridos rutinarios en búsqueda de anomalías. Los avances en cuanto a navegación autónoma de vehículos aéreos hacen viable en que estos recorridos pueden ser realizados por parte de drones autónomos (Yang et al. 2016). Sin embargo no solo la navegación puede ser realizada de forma autónoma. Con los sensores existentes hoy en día y los avances en el área del procesamiento de imágenes también es viable la detección de anomalías de forma autónoma (Haritaoglu 1998). Además la existencia de redes inalámbricas hace posible la comunicación entre vehículos cercanos, permitiendo intercambiar información con fines de coordinación (Sahingoz 2014). Por lo que una solución completa y escalable de vigilancia que no necesite supervisión humana utilizando drones es potencialmente posible.

El objetivo de este proyecto fue investigar e implementar una solución de vigilancia de un predio por parte de una flotilla de drones autónomos. Para poder abordar el objetivo planteado, fueron definidos un número de problemas a resolver. El primero de estos es la navegación, es decir que los drones de la flotilla tengan la capacidad de navegar el espacio. El segundo problema es la recarga de las baterías, donde los drones deben ser capaces de recargar su batería de forma autónoma. En tercer lugar se encuentra la detección de intrusos, donde los drones deben ser capaces de detectar, tomar imágenes y notificar la presencia de intrusiones en la zona vigilada. Por último, los drones deben poder comunicarse para hacer uso de información compartida y actuar de manera coordinada.

La solución diseñada se basa fuertemente en el procesamiento de imagen. Para resolver el problema de la navegación, se eligió una solución basada en marcadores. Los marcadores son ubicados en la zona de vuelo y reconocidos por los drones mediante una cámara abordo a fin de estimar su posición y rotación. En base a estos datos, los drones de la flotilla son capaces de seguir rutas configuradas fuera de línea para patrullar la zona. Para la recarga de las baterías se utilizó una plataforma de carga (Junaid et al. 2017) y se diseñó un sistema de aterrizaje utilizando marcadores similares a los de navegación, que permiten a los drones alinearse y aterrizar sobre ella. En lo referente a la detección de intrusos, se desarrolló un algoritmo que detecta la presencia de figuras humanas en las imágenes capturadas por los drones, además de un sistema que permite estimar su posición y perseguirlas (Li y Yeung 2017). Para permitir la coordinación se implementó un mecanismo de comunicación que permite a los drones de la flotilla intercambiar su estado y mantener un comportamiento colaborativo. Esta coordinación implica cubrir las rutas de vuelo de los drones que se ausentan durante la carga, evitar que dos drones se dirijan a una misma plataforma de carga y notificar al resto de la flotilla en caso de detectar intrusos en la zona objetivo.

La solución propuesta se evaluó ensayando cada uno de los componentes individualmente. La solución basada en marcadores resultó capaz de seguir correctamente un recorrido predefinido. Varias configuraciones de los algoritmos de detección y seguimiento de intrusos fueron ensayadas para balancear las tasas de detección, resultando en una implementación dependiente de una buena elección del algoritmo de detección. El mecanismo de alineación y aterrizaje resultó efectivo, el proceso de aterrizaje tomó varios minutos en realizarse. Para evaluar la coordinación se experimentó con un entorno simulado. Este demostró un comportamiento correcto en el uso de una lógica simple para alcanzar los objetivos solicitados.

Este trabajó permitió un investigación inicial y el diseño de una prueba de concepto enfocada en los problemas planteados. En términos generales, las pruebas resultantes fueron prometedoras, considerando así la investigación como exitosa y demostrando la potencial viabilidad del enfoque propuesto. Además gracias a la utilización de tecnologías y soluciones ya desarrolladas, se logró crear una solución más general y se logró demostrar la viabilidad de proyectos mayor porte.

Los capítulos 1, 2, 3 y 4 de este documento presentan el marco teórico, introducen el tema a tratar, presentan el problema y analizan los trabajos relacionados. Los capítulos 5, 6 y 7 desarrollan la solución propuesta. El

capítulo 5 hace una presentación del diseño y arquitectura de la solución, mientras que los detalles de la implementación son descritos por los capítulos 6 y 7. Los capítulos 8 y 9 ofrecen una evaluación de la solución, nuestras conclusiones generales sobre el trabajo realizado y proponen posibles puntos de mejora sobre lo alcanzado al finalizar el proyecto. Finalmente, al lector interesado, los anexos A y B ofrecen información adicional sobre los drones utilizados y presentan detalles extras de la solución implementada. Salvo en los casos donde se incluye específicamente la fuente, todas las imágenes contenidas en este documento son originales de los autores.

2. Drones

Este capítulo presenta la clasificación, la composición y el funcionamiento de los drones. Las secciones 2.2 y 2.3 describen los aspectos básicos del vuelo. La sección 2.4 presenta los componentes básicos de un dron. Finalmente la sección 2.5 explica el mecanismo básico de estabilización más utilizado en este tipo de aeronaves.

2.1. Clasificación

La palabra dron es empleada para referirse de forma genérica a los vehículos aéreos no tripulados. En este grupo se distinguen los multicópteros que son drones impulsados por aspas accionadas por rotores. Esto los diferencia de los vehículos de ala fija o aeroplanos que emplean la sustentación generada por sus alas y el impulso hacia adelante para mantenerse en vuelo.

La principal clasificación de los multicópteros se da por el número de rotores: tricópteros, cuadricópteros, hexacópteros, octacópteros, etc.

2.2. Vuelo

Esta sección describe el mecanismo de vuelo utilizado por los cuadricópteros, sin embargo esta explicación se puede extender a multicópteros de mayor número de rotores como hexacópteros y octacópteros.

En un cuadricópetro cada uno de los rotores generan dos fuerzas: una fuerza de sustentación, en sentido de la vertical del dron y sentido opuesto al de la fuerza de gravedad, y un torque ortogonal a la fuerza de sustentación y en el sentido opuesto al del giro de las aspas. Si todos los rotores girasen en el mismo sentido, el torque producido sobre los rotores haría que el dron rotase de forma continua sobre su eje vertical, por lo que dos de los rotores giran en sentido horario y dos en sentido antihorario. Distintas configuraciones de velocidades en lo rotores permiten la rotación del dron en tres dimensiones. La rotación del dron sobre el eje vertical es denominada guiñada (*yaw*), la rotación sobre su eje longitudinal es conocida como alabeo (*roll*) y la rotación sobre su eje transversal como cabeceo (*pitch*) (figura 1).

En los multicópteros con cuatro o menos rotores, la pérdida de uno de los

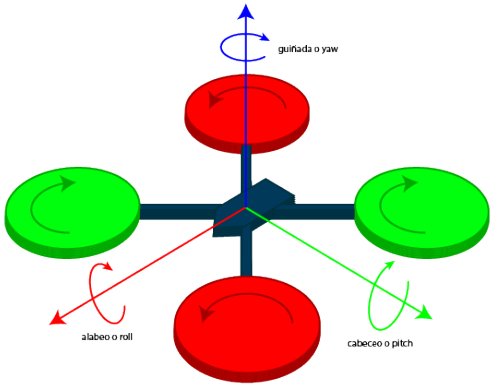


Figura 1: Ejes de movimiento de un cuadricóptero.

rotores es catastrófica y hace que sean imposibles de maniobrar. El trabajo publicado por Mueller et al. (2014) propone un mecanismo de seguridad para el vuelo en caso de la pérdida de uno, dos o tres rotores en un cuadricóptero.

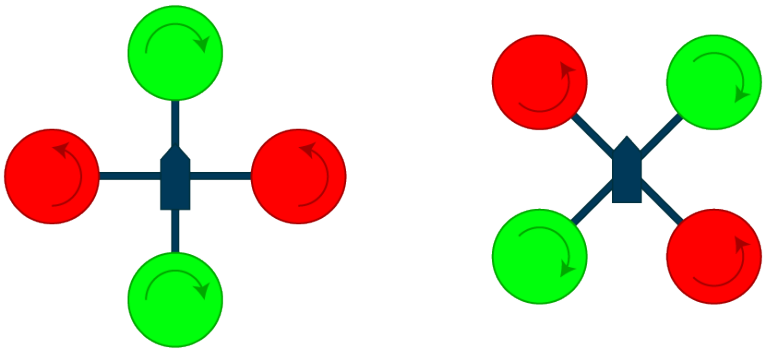


Figura 2: Configuración + vs configuración X.

Existen dos configuraciones ampliamente difundidas entre los cuadricópteros: X y + (figura 2). La primera sitúa a dos brazos del dron a 45 grados del eje longitudinal mientras que en la configuración + uno de los brazos se encuentra a lo largo de este eje. La configuración X es muchas veces preferida ya que ofrece un mejor ángulo de visión para una cámara frontal. En la figura 3 se ilustra la velocidad de rotores utilizada para realizar los movimientos de guiñada, alabeo y cabeceo según su configuración.

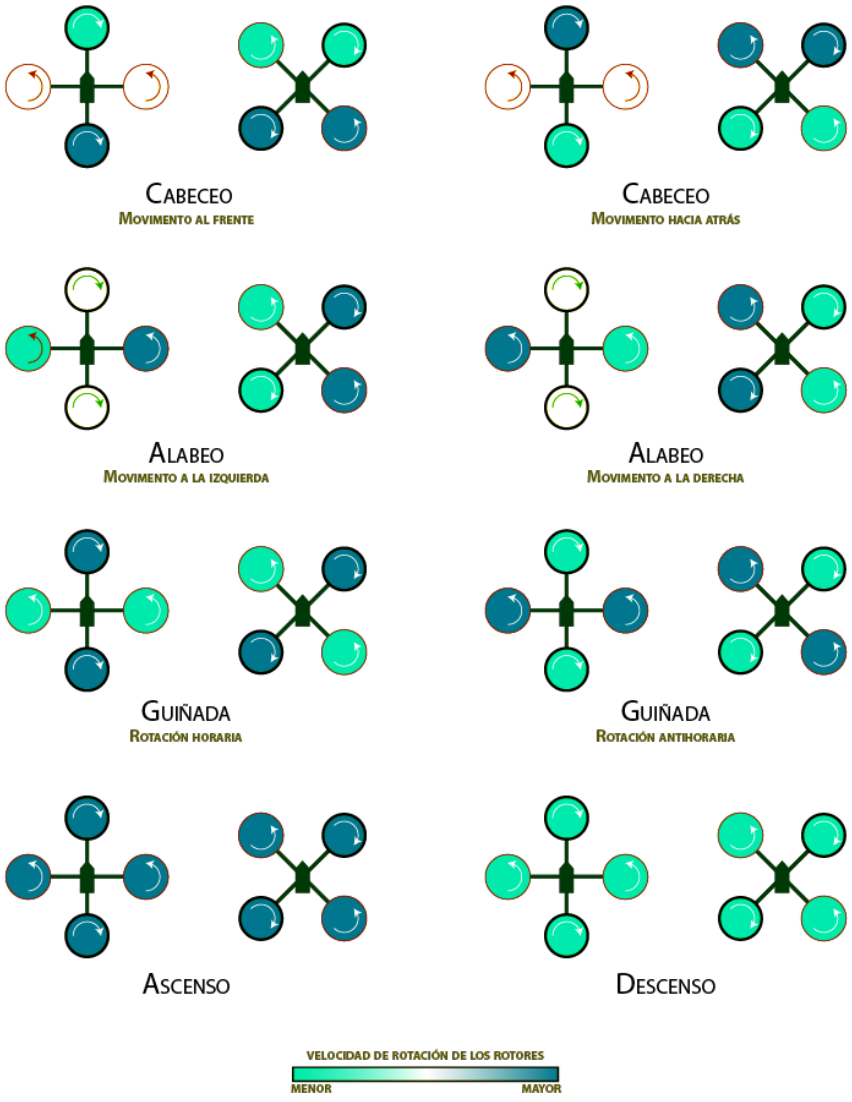


Figura 3: Movimientos del dron en sus configuraciones + y X.

2.3. Modos de vuelo

Tanto los drones comerciales como las soluciones a medida ofrecen diferentes modos de vuelo. Algunas, como ArduPilot, incluso ofrecen hasta 20 diferentes configuraciones (ArduPilot 2018). La configuración de vuelo afecta como son interpretados los comandos recibidos y en que forma actúa el controlador de vuelo para mover la aeronave. A modo de ejemplo, se presentan algunos de los más usuales:

- **Stabilize:** uno de los modos de control más sencillos para pilotos sin experiencia. La entrada de los movimientos de cabeceo y alabeo es interpretada como el ángulo de inclinación deseado. Cuando los controles son liberados, el dron se estabiliza. La entrada del movimiento de guiñada rota el dron sobre su eje vertical. Cuando la entrada vuelve a 0 el dron deja de rotar. La entrada de *gaz* determina la velocidad promedio de los rotores, por lo que para mantener la altura el piloto debe mantener control continuo sobre esta velocidad. La salida de *gaz* es ajustada por el controlador a fin disminuir los ajustes que debe realizar el piloto al afectar la actitud del dron.
- **Altitude Hold:** funciona de forma similar a *stabilize* con la salvedad que la falta de entrada de *gaz* mantiene la altura actual mientras que las entradas positivas y negativas la modifican.
- **Loiter:** similar al modo anterior pero donde el dron busca mantener la posición del dron cuando no hay entradas. El uso de los sensores de a bordo permite mantener el dron en una posición estable incluso en presencia de viento y otras perturbaciones externas e internas.
- **RTH o RTL:** *return to home* o *return to launch* emplea el GPS para dirigir el dron al punto de donde despegó. Esto puede implicar mantener una altitud sobre el punto de despegue o aterrizar en él.
- **Waypoints:** muchos drones permiten programar un recorrido en base a puntos de referencia (*waypoints*) que incluyen información como la posición GPS, la orientación y el tiempo a pasar en cada uno.
- **Acro:** es uno de los modos más complejos pero más flexibles para maniobras y acrobacias. Las entradas de *yaw*, *pitch* y *roll* controlan la velocidad angular del dron. Cuando la entrada es 0 el dron mantiene su actitud y no se vuelve a estabilizar automáticamente.

- **Follow:** utilizando información de GPS del dron y de un dispositivo conectado, como un celular y con la ayuda de seguimiento visual el dron es capaz de seguir un objetivo.

2.4. Hardware

Los drones en general, constan de varios componentes comunes que son presentados a continuación (PaparazziUAV 2017; Parrot 2018a).

- **Controlador de vuelo:** es el encargado del procesamiento de la información recolectada por los sensores, las órdenes recibidas por los comandos y la generación de instrucciones y señales a enviar a los actuadores.

- **Batería:** al no poseer la capacidad de planear, los drones dependen de la fuerza de sustentación producida por sus rotores para mantener un vuelo controlado. Esto hace que las baterías empleadas requieran no sólo gran capacidad de almacenamiento de carga sino que una gran capacidad de descarga. Por esto son utilizadas generalmente baterías con química LiPo (polímero de litio).

- **Rotores:** debido a la alta velocidad a la que giran los rotores de un dron, este emplea motores sin escobillas (*brushless*) (figura 4). Estos motores, también denominados *brushless direct current motors* (BLDC motors), a diferencia de los motores tradicionales de corriente continua, no utilizan escobillas para realizar el cambio de polaridad requerido para hacerlos rotar. Esto reduce el desgaste y pérdida de la potencia por fricción, chispas y ruido eléctrico, así como la caída en voltaje por la resistencia de las escobillas. En cambio se utilizan imanes de polaridad fija que giran alrededor de una armadura fija con bobinados que cambian de polaridad a partir de la señal de entrada del motor.

- **Electronic Speed Control:** con los motores sin escobillas la dificultad se traslada a la generación de la señal de control de los motores que debe generar la polaridad requerida en las bobinas de los motores. Para esto es empleado un controlador electrónico de velocidad (Electronic Speed Control, ESC) (figura 5). Por lo general, este tipo de dispositivos tienen tres cables de salida, sobre los que se genera una señal trifásica

Figura 4: Motor *brushless* para aviación a escala (imagen de Unknownlfcg CC BY-SA 4.0, de Wikimedia Commons).

de corriente alterna. Mediante la variación de voltaje y frecuencia (modulación por ancho de pulsos o PWM) se controla la velocidad, sentido y frenado del motor.

- **Hélices:** las hélices de un cuadricóptero son las que permiten producir empuje y sustentación a partir de su movimiento rotacional. Hay tres propiedades importantes a considerar en las hélices: el paso, el diámetro y el material. El paso es el ángulo que forma la hélice con respecto al vector de movimiento que sigue el aire al atravesarla. Una hélice de un determinado paso debe girar al doble de velocidad para recorrer la misma distancia que una del doble de paso. A mayor paso se obtiene mayor velocidad pero al costo de eficiencia, estabilidad y consumo eléctrico. Las hélices con mayor diámetro producen más empuje a expensas de un mayor requerimiento de torque al rotor y pérdida de maniobrabilidad al responder el rotor más lentamente a los cambios de velocidad por el mayor momento de inercia pero provee un dron más estable. El diámetro de la hélice debe estar de acuerdo con las revoluciones del rotor, donde a mayor diámetro menos revoluciones por minuto. Los materiales mas comunes utilizados en las hélices son la fibra de carbono y el plástico. Las hélices de fibra de carbono, en comparación con las de plástico requieren menos balanceo manual, son excelente para motores con altas revoluciones, son más livianas, por lo que hacen al dron más responsivo y más rígidas por lo que vibran menos, pero a su vez son menos flexibles, pudiendo comprometer el motor en un impacto además de ser más caras y producir menos empuje.

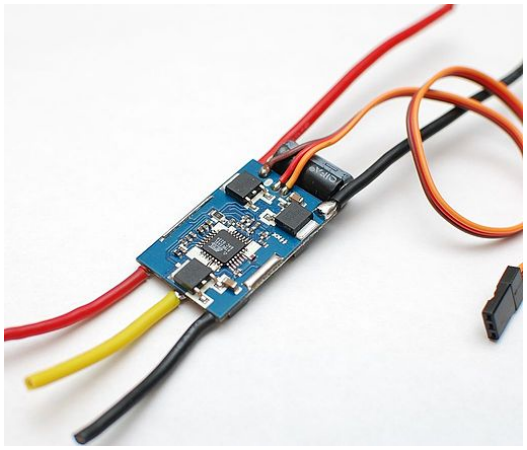


Figura 5: ESC de 35 A (imagen de Avsar Aras CC BY-SA 3.0, de Wikimedia Commons).

- **Sensores:** para mantener y mejorar su estabilidad, así como planear su ruta, determinar su orientación, posición y altitud, los drones suelen contar con múltiples tipos de sensores que envían información a la controladora de vuelo para su procesamiento.

- **Cámara frontal:** aunque también se puede considerar como un sensor, el principal uso de la cámara frontal es la visión en primera persona (*First Person View*, FPV) para el operador del drone.

- **Comunicación:** existen múltiples alternativas empleadas para la comunicación con un dron. Pueden implicar comunicación unidireccional o bidireccional con telemetría sobre varios canales de radio, comunicación analógica o digital, sobre protocolos como la transmisión de señales PWM o MAVLink.

2.5. Estabilización

Un cudricóptero es naturalmente inestable. Cualquier diferencia en la velocidad de rotación de las aspas, su superficie, viento o cualquier otra perturbación en las fuerzas producidas por los rotores es capaz de desestabilizarlo. Debido a esta inestabilidad, este tipo de aeronaves requieren un mecanismo de estabilización activo provisto por un microcontrolador a bordo. Estos controladores se basan en tres componentes: un giroscopio que permite detectar la variación en la rotación, un acelerómetro que permite medir la aceleración

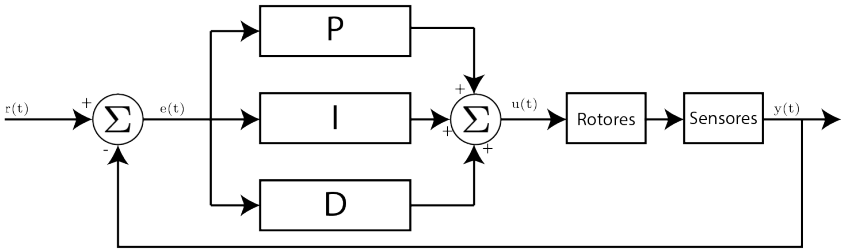


Figura 6: Flujo de procesamiento de un controlador PID

lineal del dron y un magnetómetro que brinda información sobre la orientación. Para determinar los ajustes necesarios a realizar a partir de las lecturas de estos sensores, el dron emplea un mecanismo denominado proporcional, integral y derivativo (PID). Aunque existen múltiples variantes a este modelo de controlador, a continuación se describe su forma básica a efectos instructivos.

El sistema en donde opera un controlador PID consta de tres partes: sensores, actuadores y el controlador (figura 6). En este caso, el controlador PID es el encargado de mantener la orientación y altura $r(t)$ deseadas del drone. A través de sus sensores el controlador recibe el estado actual $y(t)$. La diferencia entre estos dos valores se considera el error $e(t)$. Para intentar disminuir el error, el controlador envía una señal de control a los actuadores y vuelve a obtener el valor sensado para ajustar su señal de control. A este ciclo de lectura y ajuste se le llama ciclo de control (*control loop*). La señal de salida $u(t)$ del controlador consta de tres términos: el término proporcional, el integral y el derivativo.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

donde K_p, K_i y K_d son los parámetros de ajuste de los términos proporcional, integral y derivativos respectivamente (figuras 7, 8 y 9).

El control proporcional produce una salida que es proporcional al error. Un valor muy alto de K_p puede hacer que el sistema se vuelva inestable mientras que un valor muy pequeño puede llevar a un sistema poco responsivo. Como este control requiere un error para operar y su salida es proporcional al mismo, existe un valor para el error de estado estacionario, cuando el estado deseado se mantiene estable.

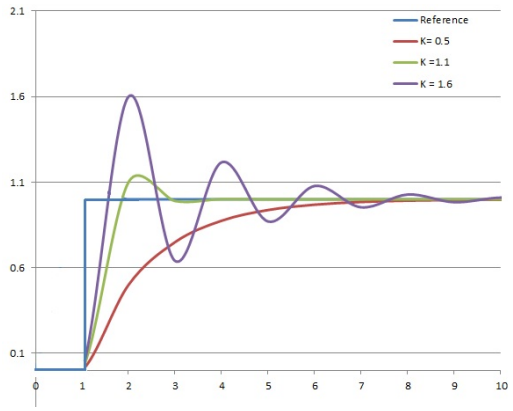


Figura 7: Respuesta de $y(t)$ para tres valores de K_p (TimmmyK CC0, de Wikimedia Commons).

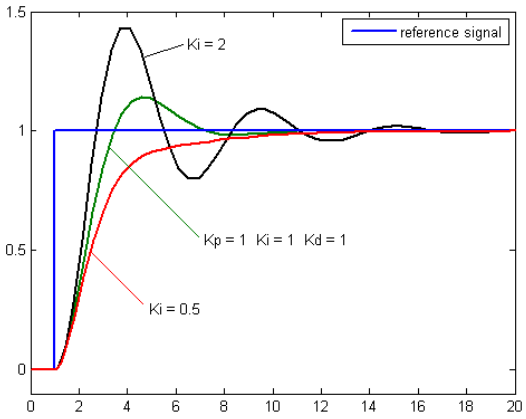


Figura 8: Respuesta de $y(t)$ para tres valores de K_i (imagen de Skorkmaz. Public domain, de Wikimedia Commons).

El control integral depende del error y su duración. Este control acelera el desplazamiento hacia el estado deseado así como elimina el error de estado estacionario.

El control derivativo no depende del valor del error, pero depende de su variación en el tiempo. Este control predice el valor futuro del sistema mejorando la estabilidad y tiempo de estabilización. En general se incorpora un filtro de paso bajo a la señal del término derivativo para evitar que el ruido y señales de alta frecuencia afecte su rendimiento.

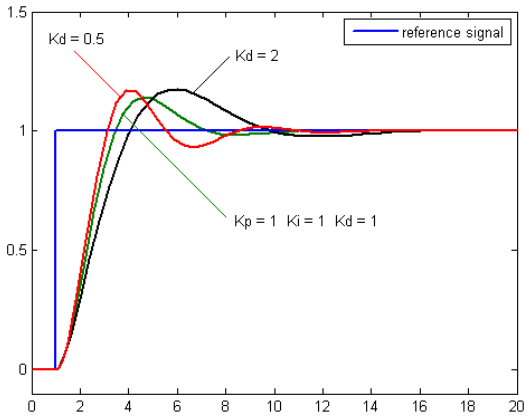


Figura 9: Respuesta de $y(t)$ para tres valores de K_d (imagen de Skorkmaz. Public domain, de Wikimedia Commons).

3. Navegación y procesamiento de imágenes

Este capítulo brinda el marco teórico necesario para abordar dos de los pilares fundamentales de este proyecto: la navegación autónoma y el procesamiento de imágenes. La sección 3.1 aborda la navegación autónoma mientras que la sección 3.2 aborda el procesamiento de imágenes.

3.1. Navegación autónoma

El problema de navegación autónoma es uno de los aspectos más complejos de resolver a la hora de investigar y diseñar cualquier solución con vehículos autónomos. Por su complejidad un sistema de navegación puede constar de uno o múltiples componentes, que ejecutan distintas tareas de sensado, planificación y actuación (Murphy 2000). Estos componentes se describen a continuación.

- **Planificación de recorridos:** la planificación de recorridos busca determinar hacia dónde debe ir el vehículo en cada momento, de modo de cumplir de manera óptima con un determinado objetivo. Por ejemplo en el caso de vigilancia esta optimización implicaría que cada vehículo debe cubrir la mayor área posible con su recorrido y cubrir la mayor área posible con su campo de visión, reduciendo lo más posible el uso de la batería.
- **Construcción de mapas:** la construcción de mapas es utilizada para construir un modelo del entorno del vehículo a medida que navega el espacio. A través de esta construcción de mapas, el vehículo puede detectar cambios en el entorno y contar con más información para la planificación de recorridos.
- **Localización:** la localización permite saber donde se encuentra el vehículo en cada momento. Este es el componente fundamental de todo mecanismo de navegación sofisticado, permitiendo seguir ruta definidas y realizar construcción de mapas.
- **Prevención de colisiones:** la prevención de colisiones es un componente altamente deseable en sistemas de navegación y sobre todo en vehículos aéreos donde una colisión resulta altamente riesgosa. La prevención de colisiones se puede categorizar según el tipo de obstáculo.

Por un lado la evasión de obstáculos estáticos, previene la colisión con obstáculos que forman parte permanente del entorno, como por ejemplo paredes. Este tipo de evasión por lo general forma parte de la planificación de recorridos. Por otro lado la evasión de obstáculos dinámica, previene la colisión con obstáculos que no forman parte fija del entorno y que varían en su ubicación o forma. Esta es una tarea compleja y que se ejecuta de forma continua apoyando el sistema de navegación estático

3.2. Procesamiento de imágenes con visión monocular

Esta sección brinda una introducción a el modelo de cámara estenopeica y su utilización en el procesamiento de imágenes con visión monocular. La subsección 3.2.1 brinda una introducción al modelo de cámara estenopeica. La subsección 3.2.2 presenta la técnica de histogramas de gradiente orientados. Finalmente la subsección 3.2.3 presenta la técnica de flujo óptico.

3.2.1. Modelo de cámara estenopeica

En visión por computadora, la matriz de cámara o matriz de proyección, es una matriz de 3×4 que describe la proyección de los puntos en el espacio homogéneo 3D al espacio homogéneo 2D en una cámara estenopeica (*pinhole*) (figura 10).

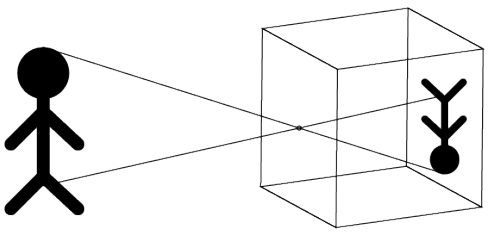


Figura 10: Modelo de cámara estenopeica.

Podemos considerar la matriz de proyección en la notación de bloque:

$$P = [M| - MC]$$

Donde M es una matriz invertible 3×3 y C es el vector columna representando la posición de la cámara en el sistema de coordenadas del espacio. Ahora, podemos descomponer la matriz de cámara en el producto de dos matrices: una matriz intrínseca K y una matriz extrínseca $R| - RC$.

$$P = K[R| - RC]$$

Donde K es una matriz triangular superior 3×3 que describe los parámetros internos de la cámara como la longitud focal y R es una matriz rotación 3×3 cuyas columnas son los ejes del sistema de coordenadas del entorno en el marco de referencia de la cámara. El vector $t = -RC$ representa la posición del origen de las coordenadas del entorno en las coordenadas de la cámara.

La matriz extrínseca describe la posición y orientación de la cámara, parámetros que no dependen de la cámara en sí mismo, de ahí su nombre.

$$K = [R|t] = \left(\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right)$$

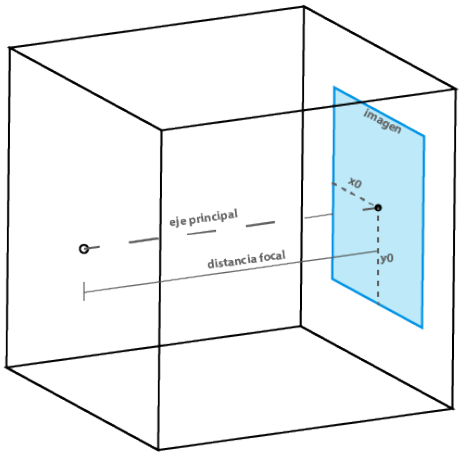


Figura 11: Propiedades de la matriz intrínseca representados en la cámara estenopeica.

La matriz intrínseca describe parámetros propios de la cámara, de donde se deduce su nombre (figura 11). La matriz intrínseca modela una cámara estenopeica, donde la luz pasa a través de un pequeño orificio en el frente

de la cámara y se proyecta invertida sobre la superficie en la parte posterior. Entonces:

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

donde f_x y f_y representan la distancia focal y por lo general tienen el mismo valor f. Esta distancia se mide en píxeles. $x_0\, y_0$ es la ubicación del *pinhole* respecto a la superficie donde se proyecta. Por su parte, s representa la distorsión *skew* de la imagen.

Podemos considerar una imagen virtual, no invertida, que se forma sobre un plano imaginario ubicado delante de la cámara (figura 12).

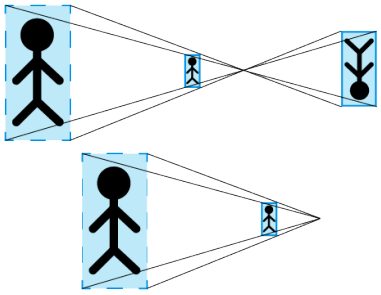


Figura 12: Construcción de la imagen virtual.

Todas las cámaras presentan un nivel de distorsión respecto a la cámara ideal. Esta distorsión puede ser causada por distintos factores, como por ejemplo el lente que utiliza la cámara. La calibración permite volver a mapear los píxeles provenientes de la cámara para obtener una imagen con la menor distorsión posible. Utilizamos la calibración para realizar mediciones apropiadas de distancias en unidades absolutas. La distorsión radial se puede ver en valores positivos como distorsión de barril (*barrel*) y en negativos alfiletero (*pincushion*) (figura 13):

$$x_{corregido} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corregido} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Figura 13: Distorsión radial positiva y negativa.

Por su parte la distorsión tangencial ocurre porque el sensor de la cámara no se encuentra perfectamente paralelo al lente (figura 14):

$$x_{corregido} = x + \left(2p_1xy + p_2(r^2 + 2x^2)\right)$$
$$y_{corregido} = y + p_1(r^2 + 2y^2) + 2p_2xy$$

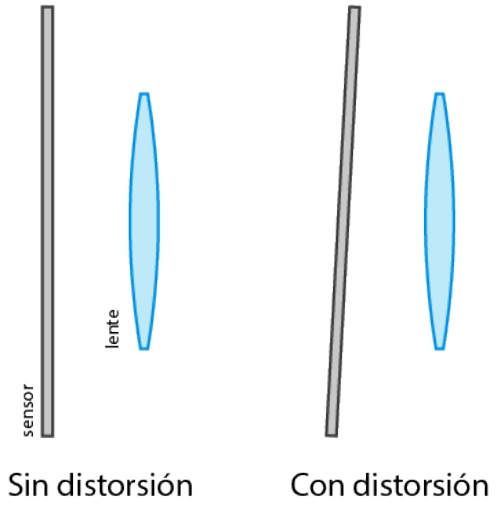


Figura 14: Distorsión tangencial.

La librería de procesamiento de imágenes OpenCV posee un método (`calibrateCamera`) que nos permite obtener la matriz de cámara y los coeficientes de distorsión

de los datos obtenidos a partir de diferentes tomas de un patrón de calibración. En OpenCV los coeficientes de distorsión se representan como 5-tupla

$$D_{coefficients} = (k_1, k_2, p_1, p_2, k_3)$$

3.2.2. Histograma de gradientes orientados

La técnica histogramas de gradiente orientados (histrogram of oriented gradients, HOG) es muy utilizada para la detección de peatones y de la figura humana en general (Dalal y Triggs 2005). Esta técnica se basa en analizar la imagen por celdas. Las celdas utilizadas pueden ser diferentes tipos: circulares, rectangulares o cuadradas. Supongamos una celda cuadrada de $n \times n$ píxeles. Para los n^2 píxeles en la celda se calcula el vector gradiente y se construye un histograma de m canales donde cada vector vota sobre la orientación. La orientación puede ser en el rango $[0, 2\pi]$ o $[0, \pi]$ así como el peso del voto de cada píxel es una función de la magnitud del vector. Además el aporte de cada vector se distribuye entre los dos canales más cercanos en el histograma de forma proporcional a su cercanía al centro de los canales (McCormick 2013).

Para la detección de figuras humanas, la técnica HOG hace uso de un histograma de nueve canales, un rango $[0, \pi]$ y la función magnitud del vector como función de voto daban los mejores resultados. Para el resto de esta explicación se asumirán estos valores.

Para hacer el algoritmo resistente a cambios en contraste e iluminación, las celdas son agrupadas en bloques y normalizadas localmente por bloque. Aunque los bloques pueden tomar versiones radiales o rectangulares, asumiremos bloques cuadrados de 2×2 celdas. El llamado descriptor HOG es la concatenación de los componentes de todos los histogramas de las celdas de todos los bloques. Los bloques pueden superponerse por lo que las celdas pueden concatenarse más de una vez al descriptor.

Dalal et al. propusieron cuatro funciones de normalización. Siendo v el vector no normalizado conteniendo todos los canales de todos los histogramas de un bloque, $\|v\|_k$ su k-norma con $k = 1, 2$ y e una pequeña constante:

$$L2 - norm = \frac{v}{\sqrt{\|v\|_k^2 + e^2}}$$
$$L2 - hys = \text{L2-norm acotado y luego normalizado}$$
$$L1 - norm = \frac{v}{\|v\|_k + e}$$
$$L1 - sqrt = \sqrt{\frac{v}{\|v\|_k + e}}$$

El ejemplo, en una imagen de 64x128 píxeles con bloques de 2×2 celdas de 8×8 y una superposición de bloques del 50 % nos deja con 7×15 bloques: 105 bloques en total. Cada bloque contiene 4 celdas con un histograma de 9 canales en un vector de 36 valores. Los 36 valores se repiten en los 105 bloques totalizando un descriptor HOG de 3780 valores.

Aunque el algoritmo está desarrollado para la obtención de los descriptores y es agnóstico al clasificador utilizado, sus autores, Dalal et al. lo utilizaron como entrada de una máquina de vectores soporte (*support vector machine*, SVM). Además estudiaron diversas variaciones del algoritmo como el uso de una ventana gaussiana para cambiar los pesos de los píxeles de los histogramas de las celdas, disminuyendo el valor de los píxeles cercanos a los bordes de los bloques.

3.2.3. Flujo óptico

El flujo óptico se define como el patrón del movimiento aparente de los elementos de una escena (Fleet y Weiss 2006; OpenCV 2018b). Consideremos una partícula en el espacio tridimensional que se mueve siguiendo una trayectoria $P(t)$. Proyectando esta trayectoria sobre el plano bidimensional, obtenemos una trayectoria $p(t) = (x(t), y(t))$. Un punto de partida para obtener este flujo óptico sería asumir:

- La intensidad I de los píxeles de la proyección de un objeto no varían (significativamente) de un cuadro a otro.
- Los píxeles cercanos tienen un movimiento similar.

Partiendo de un punto (x, y) en el momento inicial, que se mueve una distancia $(\Delta x, \Delta y)$ en un tiempo Δt y partiendo de las asunciones anteriores:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Si consideramos una variación pequeña, podemos tomar la serie de Taylor:

$$I(x + dx, y + dx, t + dt) = I(x, y, t) + \frac{\partial I}{\partial y}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + ...$$

Descartando los términos de mayor orden de las series obtenemos la restricción del gradiente y de las ecuaciones anteriores se sigue:

$$\frac{\partial I}{\partial y}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0$$

Dividiendo entre Δt

$$\frac{\partial I}{\partial y}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0$$

Que nos da:

$$\frac{\partial I}{\partial y}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

Si tomamos V_x y V_y como los componentes x e y de las velocidades del flujo óptico $I(x, y, t)$ y I_x, I_y e I_t como las derivadas parciales de la intensidad en (x, y, t):

$$I_x V_x + I_y V_y = -I_t$$

Esta ecuación posee dos incógnitas por lo que no se puede resolver. El método utilizado en OpenCV para la resolución del flujo óptico es el conocido cómo método de Lucas-Kanade (Bouguet 2000). En este caso, el algoritmo asume que no sólo el desplazamiento es pequeño, sino que dentro de una ventana

alrededor del punto, el flujo es constante. Dentro de la ventana, podemos decir que:

$$\begin{gathered} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \vdots \\ I_x(q_{n-1})V_x + I_y(q_{n-1})V_y = -I_t(q_{n-1}) \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{gathered}$$

En notación matricial:

$$A = \begin{pmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_{n_1}) & I_y(q_{n_1}) \\ I_x(q_n) & I_y(q_n) \end{pmatrix} \quad v = \begin{pmatrix} V_x \\ V_y \end{pmatrix} b = \begin{pmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_{n-1}) \\ -I_t(q_n) \end{pmatrix}$$

$$Av = b$$

Como este sistema puede estar sobre determinado y no poseer solución, L-K utiliza mínimos cuadrados para hallar la solución que minimiza el error cuadrático. Esto implica la resolución del siguiente sistema de 2×2:

$$A^T Av = A^T b$$

La librería OpenCV implementa el método Lucas Kanade. Antes de invocar este método, se invoca un método auxiliar para encontrar buenas esquinas (*corners*) sobre los cuales aplicar el método de L-K (J. Shi y Tomasi 1994).

4. Vigilancia de un predio utilizando drones autónomos

Este capítulo presenta la utilización de vehículos aéreos autónomos aplicada a la vigilancia de un predio, presenta los objetivos de este proyecto y analiza los trabajos relacionados. La sección 4.1 presenta los usos de este tipo de vehículos en el área de seguridad y vigilancia. Las secciones 4.2 y 4.3 presentan el material a utilizar en este proyecto de grado. La sección 4.4 presenta los problemas y objetivos planteados. Finalmente la sección 4.5 analiza los trabajos relacionados.

4.1. Usos de los vehículos aéreos autónomos y seguridad

El diccionario de la lengua española define seguridad como *"ausencia de peligro o riesgo"* (RAE 2018). Además este mismo diccionario define vigilar como *"observar atentamente a una persona o cosa y estar pendiente de ella para evitar que sufra o cause algún daño o peligro"*.

En el contexto de este proyecto de grado definimos seguridad como un conjunto de técnicas que permiten mantener fuera de peligro ya sean lugares, objetos o personas. La vigilancia es una de la técnicas que pueden ser aplicadas para mantener fuera de peligro un lugar o zona determinada.

El objetivo de la vigilancia es detectar amenazas, a fin de prevenir cualquier daño que ellas puedan provocar sobre la zona vigilada así como a objetos o personas presentes en ella. Los daños que quieren ser prevenidos pueden ser de múltiples tipos: el hurto, violencia sobre personas, daño de objetos materiales, etc.

En la actualidad existe una gran variedad de vehículos aéreos no tripulados en el mercado. La variedad incluye pequeñas y sencillas aeronaves para uso recreativo, de mayor porte y gran calidad para uso profesional así como robustas y confiables para uso industrial (Zeng et al. 2016; UAV Coach 2018). Estos drones tienen en la actualidad un uso extensivo en distintas áreas incluyendo la seguridad (Microdrones 2018). Sin embargo a pesar de este uso ya instalado, en la mayoría de los casos los vehículos utilizados son controlados por un operador humano (Yang et al. 2016). Estos operadores están limitados a controlar unos pocos drones a la vez. Por lo que para aumentar la capacidad de trabajo, se necesitan aumentar proporcionalmente tanto la cantidad de vehículos como de operadores.

La utilización de una flotilla de vehículos aéreos autónomos en la vigilancia de un predio es un área de estudio interesante. Mediante navegación autónoma y colaboración entre drones se lograría evitar la supervisión humana y se podrían esperar una reducción de los costos de operación de este tipo de soluciones.

4.2. Parrot Bebop 2

Los drones empleados en esta tesis son cuadricópteros, una variedad particular de los multicópteros. A pesar de su limitada autonomía, su portabilidad, sus costos en descenso, su capacidad de cómputo creciente y su gran maniobrabilidad hace de los cuadricópteros una herramienta de trabajo y estudio ideal.

Figura 15: Bebop 2 en vuelo.

En este trabajo, se utilizó el modelo de cuadricóptero Parrot Bebop 2, ilustrado en la figura 15 (Parrot 2018a). Estos cuadricópteros cuentan con una cámara frontal con estabilización de imagen por software, autonomía de 25 minutos y un peso con batería de 500g. El fabricante provee un SDK, empleado en este trabajo, para la interacción con el dron (Parrot 2015). A continuación se detallan sus componentes y características.

- **Controlador de vuelo:** posee un controlador que además del vuelo realiza otras tareas que incluyen el mantenimiento de una red WiFi y el procesamiento, almacenamiento y transmisión de la imagen capturada por la cámara frontal.

- **Batería:** cuenta con una batería de 3 celdas (3 × 3.7v, 11.1v) y una capacidad de 2700 mAh con una descarga máxima de 21.5A (8,0 C = 21.5 A ÷ 2700 Ah) que brinda hasta 25 minutos de vuelo.
- **Rotores:** los motores del dron utilizado tienen una velocidad de rotación que va desde los 7500 a 12000 revoluciones por minuto (RPM).
- **Hélices:** sus hélices de plástico son capaces de alcanzar una velocidad horizontal de 16m/s y vertical de 6m/s.
- **Sensores:** cuenta con un acelerómetro de 3 ejes MPU 6050 para la determinación de la aceleración sobre el dron y un giroscopio de 3 ejes en el mismo chip MPU 6050 para la detección de la velocidad y rotación. También cuenta con un magnetómetro de 3 ejes AKM 6983 para la detección de la orientación absoluta del dron respecto al campo magnético terrestre. Además de una cámara vertical que provee información adicional del movimiento respecto al piso (Bristeau et al. 2011). Un sensor de ultrasonido ubicado debajo del dron permite determinar la altitud con precisión hasta los primeros 8m de altura respecto al piso y un barómetro MS5607 que brinda información de altitud para un mayor rango. La ubicación GPS del dron es provista por un GNSS o *Global Navigation Satellite System* NEO-M8 que aprovecha las señales de los sistemas de navegación satelital GPS, QZSS, GONASS, BeiDou y Galileo.
- **Cámara frontal:** consta de una cámara ojo de pez de 180° con sensor CMOS de 14 Mega píxeles. Existe la posibilidad de obtener el stream de la cámara en su formato de 180° aunque se puede optar obtener una versión recortada de la imagen que se encuentra estabilizada digitalmente.
- **Comunicación:** la comunicación se realiza a través de una red WiFi con transmisores MIMO de dos bandas, con antenas duales dipolo de 2.4 GHz y 5 GHz con una potencia de hasta 21 dBm y un alcance de hasta 300m. Se emplea un protocolo establecido por el ARSDK3 (Parrot 2015).

En lo que refiere a modos de vuelo, de fábrica el bebop 2 incorpora los modos Loiter, RTH, Waypoints y Follow con la capacidad de realizar algunas maniobras adicionales programadas (flips, parábolas, velas, espirales, etc). El SDK empleado provee acceso a todos estos modos menos el Follow que puede ser implementado mediante la información disponible.

4.3. Plataforma de carga

Para que los drones puedan cargar de forma autónoma se contó con una plataforma de carga de la empresa Skysense (Skysense 2018). Esta plataforma consta de un panel cuadrado de cobre dividido en celdas. El panel está encastrado en una base de plástico resistente y conectado a una caja estanca que contiene el controlador. La caja se conecta a una fuente de electricidad AC 100∼230V. En la figura 16 se puede observar el dron sobre la plataforma de carga.

Figura 16: Bebop 2 cargando sobre la plataforma.

Por su parte, a los drones se les incorporó un microcargador conectado a su batería, especialmente diseñado para funcionar con la plataforma. Este microcargador esta conectado a dos contactos de cobre, colocados en uno de los brazos del vehículo (figura 17). Estos contactos están dispuestos de forma tal de garantizar que en contacto con el panel de la plataforma, el dron se cargue independientemente de la posición y orientación de aterrizaje.

La plataforma cuenta además con conexión Ethernet. A través ella y mediante la utilización de un software provisto por el fabricante, es posible obtener y visualizar información de monitoreo. Esta información incluye entre otras monitor de carga, monitor de salud de la batería y alertas de aterrizaje y despegue.

4.4. Objetivos de este trabajo

El objetivo este proyecto es investigar e implementar una prueba de concepto para la vigilancia de un predio por parte de una flotilla de drones autóno-

Figura 17: Contactos de carga en uno de los brazos del dron.

mos. Además es deseable que los drones utilizados sean de venta comercial y la solución utilice únicamente el hardware a bordo. Para poder abordar el objetivo planteado, fueron definidas un número de problemáticas a estudiar cuya resolución haría factible una solución acorde.

El primer problema a resolver es la navegación. Los drones deben ser capaces de navegar el espacio de forma autónoma. Esto implica que cada dron tenga referencia de su ubicación en el espacio y un mecanismo que determine hacia donde se debe mover en cada momento. Además, es deseable que los drones consideren la presencia de obstáculos e implementen mecanismos para evitarlos.

Para lograr la autonomía también se debe resolver la recarga de baterías de los drones. Se propone entonces emplear una plataforma de carga, donde los drones puedan aterrizar y recargar su batería. Con el fin de lograrlo, deberán conocer su ubicación, volar hasta su posición y poseer control fino de vuelo para aterrizar sobre ella.

Otro problema a resolver es la detección de intrusos. Los drones deberán ser capaces de detectar, tomar imágenes y notificar la presencia de intrusiones en la zona vigilada. También es deseable que logren determinar la posición de las amenazas en tiempo real mientras duren las intrusiones.

Por último, los drones deben hacer uso de información compartida y actuar en consecuencia así como coordinar su accionar para desarrollar un comportamiento sinérgico. Para esto es necesario que sean capaces de comunicarse entre sí de modo de intercambiar información de su estado y los eventos que

perciben.

La colaboración entre drones deberá garantizar que la vigilancia de la zona objetivo se lleve acabo eficientemente, maximizando el área cubierta con el numero disponible de drones y minimizando el tiempo de cada recorrida. Por este motivo esta colaboración permitirá determinar los recorridos que deben realizar los drones en cada momento, cuándo deben utilizar la plataforma para recargar su batería, qué acción se debe realizar ante la presencia de intrusos y cualquier otro accionar que contribuya a vigilar mejor la zona objetivo.

También se definió un número de restricciones al alcance del problema. Todo el proyecto se realizó sobre el material suministrado: hasta tres drones disponibles con sus accesorios y una plataforma de carga. Además se asumen condiciones apropiadas de iluminación en la zona de trabajo.

Aunque se asumió como deseable que la mayor parte del procesamiento se realizase sobre el dron, se permitió ejecutar sobre unidades de procesamiento externas.

4.5. Trabajos relacionados

Para enfrentar el objetivo planteado en este proyecto y realizar la investigación fue necesario descomponerlo en sus partes constituyentes y analizar las soluciones existentes de forma individual.

El problema del vuelo estable de los cuadricópteros es un problema muy analizado y con soluciones ya implantadas a nivel comercial. Aunque existen algunos estudios con métodos como lógica difusa (Sureshkumar y Cohen 2014) o redes neurales (Nicol et al. 2008), el control utilizando controladores PID es predominante por su performance y su simplicidad. Estos controladores calculan constantemente la desviación del vehículo utilizando los sensores y modifican la velocidad de sus rotores para corregirla. En 1992, la Universidad de Michigan ya empleaba controladores PID en su HoverBot (Borenstein 1992). El HoverBot fue una de las primeras plataformas eléctricas capaces de realizar vuelos estacionarios y despegar horizontalmente de forma autónoma.

Las soluciones al problema de la navegación, en cambio, se encuentra aún en plena investigación. Estas soluciones muchas veces operan bajo restricciones y en escenarios particulares. La mayoría de los drones comerciales emplean

GPS como método por defecto para la navegación. Sin embargo debido a su relativamente baja precisión, en el orden de varios metros y el hecho de no estar siempre disponible, por ejemplo en áreas cerradas, existen otras alternativas investigadas.

Dependiendo de los mecanismos de sensado del ambiente a utilizar, existen distintos métodos de navegación sin GPS. Shen et al. (2013) emplean escáneres láser denominados LIDAR (Laser Imaging Detection and Ranging) que consisten en haces de luz pulsados. Las distancias son calculadas midiendo el tiempo entre la emisión del pulso y la detección del haz reflejado que encuentra en su camino. También es ampliamente difundida la navegación visual (Balamurugan et al. 2016) que emplea técnicas de procesamiento de imágenes para determinar el desplazamiento en el entorno (odometría), en algunos casos generando mapas de entornos desconocidos y logrando posicionarse en estos mapas.

Pese a que algunas soluciones de navegación visual emplean información de cámaras estéreo (Seitz et al. 2006), su uso restringe el espectro de drones compatibles, ya que pocos cuentan con dos sensores visuales en disposición estereoscópica, o sea, cámaras dispuestas en el espacio de forma de poder inferir información tridimensional a partir de la diferencia entre las imágenes capturadas por cada una de ellas. Matsumoto et al. (1999) publicaron una propuesta de navegación visual que hacía uso de de cámaras monoculares, es decir una única cámara.

Las soluciones de navegación visual monoculares carecen de información de medidas absolutas por lo que para obtener información de odometría en unidades físicas se fusiona la información visual con información inercial (García et al. 2012; Mourikis y Roumeliotis 2007). Ya en 1991 era estudiada la utilización de filtros extendidos de Kalman (*Kalman Extended Filter*, KEF) para fusionar la información inercial (Grewal et al. 1991). Nombrados a partir de Rudolf E. Kálmán, uno de sus principales desarrolladores, los KEF utilizan un conjunto de mediciones en el tiempo que contienen ruido y otras imprecisiones para producir una medida más exacta que la generada por una única medición. Otra técnica empleada, se basa en una segunda cámara de visión vertical para el cálculo de flujo óptico respecto al piso, técnica de la que hacen uso los drones Parrot empleados en este proyecto (Bristeau et al. 2011).

Los métodos de navegación visual se pueden clasificar en general en aquellos sin mapa, con mapa estático y con mapa construido dinámicamente. Horn et al. (1981) utilizan el desplazamiento de los elementos en el campo visual

conocido como flujo óptico, en un ejemplo de método sin mapa. Otro método sin mapa es la detección de características (*features*), que representan elementos de interés en la imágenes (Cho et al. 2013). Otros métodos basados en características son los métodos SURF (Bay et al. 2006), ORB (Rublee et al. 2011) y SIFT (Lowe 2004). Tantos aquellos basados en características o aquellos basados en flujo óptico emplean información obtenida del sensor de la cámara para calcular el movimiento relativo del vehículo.

Uno de los primeros intentos de construir un mapa mientras se navega utilizando una sola cámara fue presentado en 1983 empleando un robot construido por la universidad de Standford (Moravec 1983). En la actualidad destaca el método de localización y modelado simultáneos (SLAM), basado en detección de características que permiten la construcción de un mapa en tiempo real al mismo tiempo que el vehículo se posiciona dentro de este mapa (Leonard y Durrant 1991; Davison 2003; Esrafilian y Taghirad 2016). Entre los métodos SLAM modernos se destacan ORB-Slam (Mur-Artal et al. 2015; Mur-Artal et al. 2017) y SVO (Forster et al. 2014). cómo métodos visuales monoculares dispersos. También existen soluciones que utilizan SLAM en combinación con filtros extendidos de Kalman para fusionar la información visual con información inercial para obtener medidas precisas sobre unidades físicas (Engel 2011).

Otro mecanismo empleado en navegación es la utilización de marcadores (*markers* o *landmarks*), es decir elementos físicos conocidos que pueden ser reconocidos a fin de determinar el estado del vehículo respecto a estos. Garrido-Jurado et al. (2014) desarrollaron la librería de marcadores fiduciarios ArUco que es empleado en este proyecto. Otra propuesta de navegación por marcadores es la desarrollada por Nitschke (2014). Estos mecanismos también son capaces de aprovechar la fusión con información inercial o con otros sensores utilizando los filtros de kalman mencionado anteriormente como lo demuestran Sánchez et al. (2017).

Los mecanismos de evasión de obstáculos, al igual que los mecanismos de navegación visual están generalmente distribuidos en dos tipos: basados en flujo óptico y basados en detección de características. Por una parte, Gosiewski (2011) y Al-Kaff (2016) desarrollaron mecanismos visuales de evasión de obstáculos empleando flujo óptico. Por otra parte, la tesis de Sagar (2014) empleó algoritmos ya desarrollados para la detección de características a fin de determinar obstáculos durante el vuelo. En los tres casos, las publicaciones se desarrollaron para el vuelo de pequeñas aeronaves no tripuladas como las empelados en este proyecto.

El aterrizaje autónomo también es un tema investigado. El principal articulo tomado como referencia, por Barták et al, plantea la utilización de reconocimiento de un patrón visual para ubicar el área de aterrizaje utilizando la cámara y aterrizar sobre ella (2014).

La detección de objetos y personas utilizando algoritmos de reconocimiento de imagen es un área ampliamente investigada. Uno de los algoritmos más utilizados para la detección de la figura humana es el histrograma de gradientes orientados (*histograms of oriented gradients*, HOG) (Dalal y Triggs 2005; Blondel et al. 2014), que desarrolla el uso de descriptores particulares para el entrenamiento de algoritmos de aprendizaje como pueden ser las máquinas de vectores de soporte (Support Vector Machine, SVM). Sin embargo también son utilizados otros métodos como los basados en redes neuronales. Un ejemplo de un método de este tipo es YOLO (Redmon, Divvala et al. 2015; Redmon y Farhadi 2016; Redmon y Farhadi 2018), un método con gran capacidad de detección con más de nueve mil clasificaciones diferentes, pero que requiere de gran capacidad de computo para ejecutarlo en tiempo real.

Los algoritmos de detección de objetos también pueden ser combinados con sistemas de seguimiento (*tracking*). Esta combinación implica que luego de detectar un objeto se procede a seguirlo en la imagen, una operación mucho más eficiente que la detección cuadro a cuadro, permitiendo aumentar el tiempo de detección y lograr un seguimiento más preciso (Henriques et al. 2015). Existen múltiples algoritmos de seguimiento que pueden utilizados: *trackers* SURF (Bay et al. 2006), SIFT (Lowe 2004), ORB (Rublee et al. 2011), KCF (Henriques et al. 2015), Medianflow (Kalal et al. 2010), Mil (W. Shi et al. 2014), Boosting (Grabner et al. 2006) y TLD (Kalal 2011).

5. Diseño y arquitectura de la solución propuesta

Este capítulo presenta la arquitectura de la solución desarrollada. La sección 5.1 describe la arquitectura, sus componentes y detalla aspectos relevantes de la implementación. Adicionalmente la sección 5.2 presenta aspectos de configuración y depuración utilizados.

5.1. Arquitectura

La arquitectura de la solución de software desarrollada para resolver las problemáticas planteadas consta de cuatro grandes componentes que denominamos Cerebro, Intercomunicación, Cuerpo y Hardware Abstraction Layer (HAL). Cada componente trabaja sobre un nivel de abstracción diferente del problema a resolver. El cerebro se encarga de la toma de decisiones a alto nivel, la intercomunicación se encarga de la comunicación con los drones de la flotilla, el cuerpo se encarga de llevar a cabo las acciones a partir de las decisiones del cerebro y HAL se encarga de la comunicación de bajo nivel con el dron. En la figura 18 se pueden ver estos componentes y su interacción entre ellos (representado con flechas dobles), junto con el dron (parte inferior derecha) y el resto de la flotilla (parte superior izquierda).

La principal función del cerebro es la toma de decisiones y la planificación de las acciones a tomar por parte del dron. Estas decisiones deben ser traducidas en órdenes para que ejecute el cuerpo y en mensajes para informar el estado del dron, además de información relevante al resto de la flotilla. El cerebro funciona en base a dos fuentes de información: los mensajes de actualización del estado del resto de la flotilla recibidos a través del módulo de intercomunicación y la información de estado relevante al dron proveniente del cuerpo. La lógica del cerebro funciona como una máquina de estados. Cada estado corresponde a una acción que realiza el dron, por ejemplo, patrullar la zona. Las transiciones responden a eventos internos como cuando el dron tiene poca batería o externos como cuando un dron recibe información de un intruso detectado por otro dron. Los detalles sobre la implementación de máquina de estados se desarrollan en la subsección 7.2.2.

La intercomunicación se encarga de resolver la comunicación con el resto de la flotilla. Este componente es utilizado a demanda por el cerebro que envía su estado para que sea distribuido entre los integrantes de la flotilla. La intercomunicación se encarga entonces de codificar y decodificar los mensajes

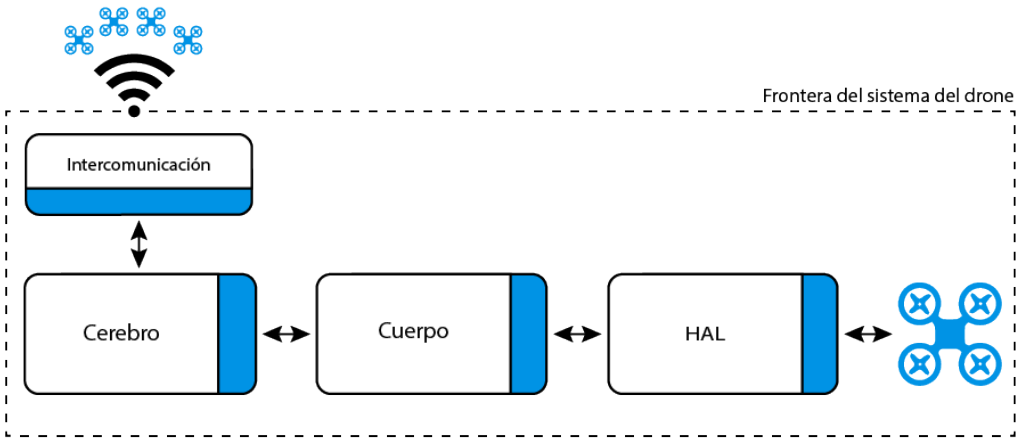


Figura 18: Arquitectura de la solución propuesta.

de actualización de estado, además de implementar un protocolo de difusión a través del cual son distribuidos. En la subsección 7.2.1 se detallan los medios y protocolos de comunicación utilizados.

El cuerpo es el encargado de ejecutar las órdenes que recibe de desde el cerebro y de comunicar a éste de eventos percibidos desde el entorno. El cuerpo también implementa una máquina de estados para responder a los comandos determinados por el cerebro, además de implementar sistemas que se encargan de brindar información relevante para ser utilizada por la máquina de estados. Cada sistema puede habilitarse o deshabilitarse dependiendo del estado por el que se esté transitando. Los sistemas implementados corresponden a:

- `BatterySystem`: obtiene el valor de carga de la batería a través del HAL.
- `FollowerSystem`: se encarga de la detección, seguimiento y persecución de intrusos.
- `MarkerTrackerSystem`: se encarga de detectar marcadores y estimar la posición del dron en base a estos marcadores.

- `OpticalFlowSystem`: determina posibles obstáculos basado en el flujo óptico de la entrada de la cámara.
- `PadLandingSystem`: se encarga de determinar los comandos de movimiento necesarios para alinearse y aterrizar sobre la plataforma.

La máquina de estados, con ayuda de los sistemas, es la encargada de ejecutar las las órdenes provenientes del cerebro. Además facilita a este último información de estado del dron. Los estados implementados corresponden a:

- `AlertState`: se encarga del seguimiento de una intrusión reportada por otro dron.
- `FollowingState`: realiza el seguimiento de una intrusión detectada por el propio dron.
- `GoToPadState`: dirige al dron hacia la plataforma de carga.
- `GoToPathState`: dirige al dron de vuelta a patrullar desde la plataforma de carga.
- `PatrollingState`: realiza el seguimiento del camino asignado.
- `ShutdownState`: aterriza inmediatamente.
- `TakingOffState`: despega antes de pasar al patrullaje.
- `VirtualBodyState`: realiza una simulación del comportamiento del cuerpo del dron.

El cuerpo se abstrae de la implementación de la operación con el dron físico mediante una capa de abstracción denominada HAL. Esta abstracción permite añadir drones a la flotilla, ya sean reales o simulados, sin impactar en la lógica del resto de los componentes. También facilita el desarrollo mediante la utilización de un HAL simulado, permitiendo manipular completamente la salida de este componente con fines de depuración. Los tres tipos de HAL implementados corresponden a:

- ***Dummy***: implementa la interfaz común de los *HAL* pero no efectúa acción alguna: los comandos son descartados y se retornan valores por defecto en los métodos. La única salvedad es la salida de la cámara que devuelve los cuadros de una salida definida en la configuración: un archivo de video o el primer dispositivo de cámara encontrado.

- ***VRep***: es empleado para una evaluación en un ambiente simulado. Implementa una fachada con V-Rep, un simulador robótico desarrollado por Coppelia Robotics (Coppelia Robotics 2018). V-Rep ejecuta una simulación del dron y provee información de las capturas realizadas por la cámara.

- ***Pb2***: implementa la interfaz con el dron físico. Este *HAL* hace uso del SDK que provee Parrot para la integración y desarrollo con sus productos.

Para la implementación de la arquitectura se decidió proceder con una separación a nivel de hilos (threads) del cerebro y el cuerpo. En las implementaciones iniciales se separó el cerebro y el cuerpo en procesos independientes y la comunicación entre ellos se realizó mediante *sockets*. Esta solución permitía ejecutar el cerebro y el cuerpo en máquinas diferentes, logrando total independencia. Finalmente se descartó la solución por resultar muy compleja dificultando las pruebas y no reportar beneficios apreciables siendo que en los escenarios de prueba concebidos todos los componentes se ejecutaban en un mismo sistema. La nueva solución descartó los procesos pesados del sistema operativo y optó por una solución liviana basada en *pthreads* (*POSIX threads*). El mecanismo de comunicación basado en *sockets* se remplazó por un área compartida de memoria. Además existe una clase principal (*main*), que es encargada de crear el área de memoria compartida y los hilos correspondientes para a cuerpo y cerebro. Luego de creados, estos hilos se ejecutan en bucle resolviendo en cada iteración su lógica interna.

5.2. Configuración, depuración y registro

Para poder gestionar de mejor manera los parámetros de ejecución de la solución construida, se creó una solución para el manejo de la configuración basada en archivos YAML (YAML Ain't Markup Language, originalmente Yet Another Markup Language) (YAML 2018).

```
// Obtener la ubicación del modelo entrenado del CascadeDetector
std::string path = config->Get(ConfigKeys::Body::CascadeDetector)
```

Listado 1: Ejemplo de obtención de un parámetro de configuración.

Al ejecutar el programa se busca el archivo de configuración en la ubicación estipulada por el primer argumento de línea de comandos o en su defecto el archivo *config.yaml* en el directorio de trabajo. De no existir, un archivo de configuración nuevo es generado con todos los parámetros asignados con sus valores por defecto. Esta solución desarrollada para el manejo de la configuración hace uso de plantillas (*templates*) de C++ para hacerla fuertemente tipada, reduciendo los errores en tiempo de ejecución y mejorando la legibilidad del archivo de configuración. En el listado 1 se puede ver un ejemplo de uso de un parámetro de configuración.

```
// Se imprime la ubicación del archivo de configuración
Logger::logDebug("Saving configuration file at %s") << path;
```

Listado 2: Ejemplo de *logging* con formateo.

Por otra parte, tener mecanismos de depuración (*debbugging*) y registro (*logging*) es fundamental a la hora de facilitar el desarrollo y realizar pruebas. En el contexto de la solución propuesta, estos mecanismos son aún más importantes debido a la existencia de información espacial sobre el plano bi o tridimensional como posiciones, rotaciones o vectores. Esta información es recibida en tiempo real y muchas veces debe procesarse a medida que llega, haciendo muy difícil insertar puntos de interrupción y reproducir situaciones mediante la técnica de depuración paso a paso. Los mecanismos diseñados e implementados fueron los siguientes:

- **Registro**: La primera y más sencilla estrategia consistió en un motor de *logging* liviano que envía a consola los mensajes solicitados. Se implementó de forma de permitir una interfaz con formato similar a la *Format Boost Library* (Boost 2018). En el listado 2 muestra un ejemplo de su uso.
- **Visores**: Se incorporaron visores en tiempo real que permitiesen visualizar la salida de la cámara con indicadores gráficos sobreimpresos en la imagen como marcadores, recuadros y textos para ayudar a determinar el estado del cuerpo y el cerebro. En la figura 19 se pueden observar dos capturas de visores en funcionamiento.
- **Almacenado de imagen y video**: Las salidas de los visores puede ser almacenadas en un archivo en formato de video o generar capturas para su estudio posterior. Para el formato video existen además dos

(a) Seguimiento de una detección (b) Estimación de posición de marcador

Figura 19: Captura un visor en diferentes escenarios.

opciones: una de las salidas incluye gráficos impresos sobre la entrada de cámara mientras la otra ofrece la salida sin procesar.

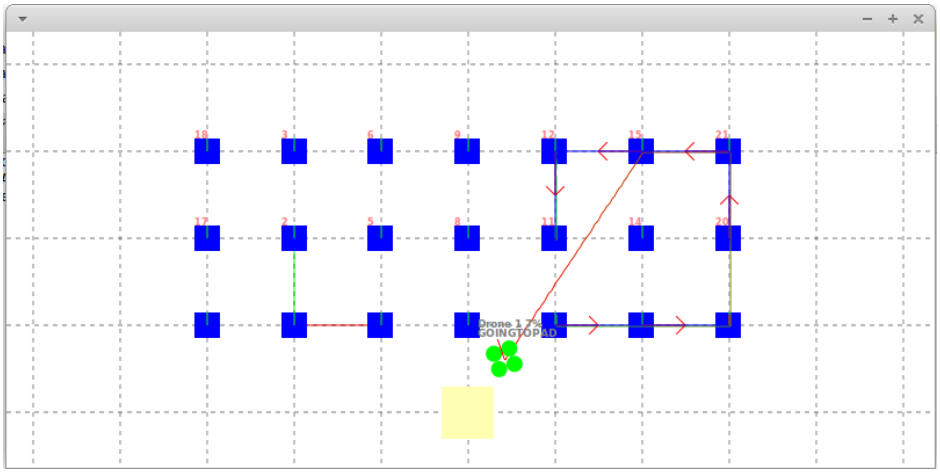


Figura 20: Ejemplo de mapa que muestra el dron y los distintos elementos del ambiente.

- **Mapas**: Además de los visores, se implementó una vista 2D para ubicar los marcadores, plataformas y las estimaciones de la posición y orientación de los drones en el espacio. En la figura 5.2 se puede observar una captura del mapa durante un vuelo simulado. En el mapa se puede observar los marcadores en color azul, la plataforma en color amarillo y el dron en color verde.

- **Comandos**: Los visores permiten también capturar y procesar la en-

trada del teclado para ejecutar comandos que permitan modificar el estado, acceder al modo de control manual o incluso a abortar la misión durante el vuelo. Algunos parámetros también pueden ser modificados en vuelo como el valor de inclinación de la cámara, la saturación, exposición y balance de blancos de la imagen.

6. Navegación y detección de intrusos

Este capítulo presenta los algoritmos implementados para resolver la navegación y la detección de intrusos. La sección 6.1 aborda la navegación, mientras que la sección 6.2 aborda la solución propuesta para la detección de intrusos.

6.1. Navegación

Esta sección presenta la solución de navegación implementada en base a marcadores. Por su parte las subsecciones 6.1.2 y 6.1.3 presentan soluciones investigadas para abordar los problemas de evasión de obstáculos dinámica y localización y mapeo simultaneo respectivamente.

6.1.1. Navegación por marcadores

El sistema de navegación desarrollado se basa en la estimación de la localización de los drones por medio de la utilización de marcadores. Los marcadores son objetos que se colocan en la zona de vuelo y que pueden ser reconocidos por el dron utilizando su cámara. Los marcadores utilizados consisten en una grilla cuadrada de 6×6 (figura 21). El borde exterior, de una casilla de ancho, es negro por lo que la información del marcador se codifica en la grilla interior de 4×4. Tanto para generarlos como para reconocerlos se utiliza la librería *OpenCV* basada en *ArUco Markers* (Garrido et al. 2014).

Si la codificamos numéricamente, la grilla de un marcador se puede leer como una 16-tupla:

$$m_i(w_0, w_1, w_2, ..., w_{15})$$

El diccionario de marcadores elegidos consta de 50 elementos, donde a cada tupla se le asigna un identificador. Los diccionarios son generados de forma de maximizar la distancia de *Hamming* mínima entre los marcadores, de forma de maximizar la capacidad de detección y corrección de errores al identificar los valores de las tuplas.

Cuando se detecta un marcador, se calcula el valor de la tupla sin rotar $\tau 0$. Además se calcula el valor para las 3 rotaciones diferentes $\tau_{90}, \tau_{180}, \tau_{270}$. Y se calcula la distancia de Hamming a cada uno de los elementos del diccionario D_{50} para cada una de las posibles rotaciones.

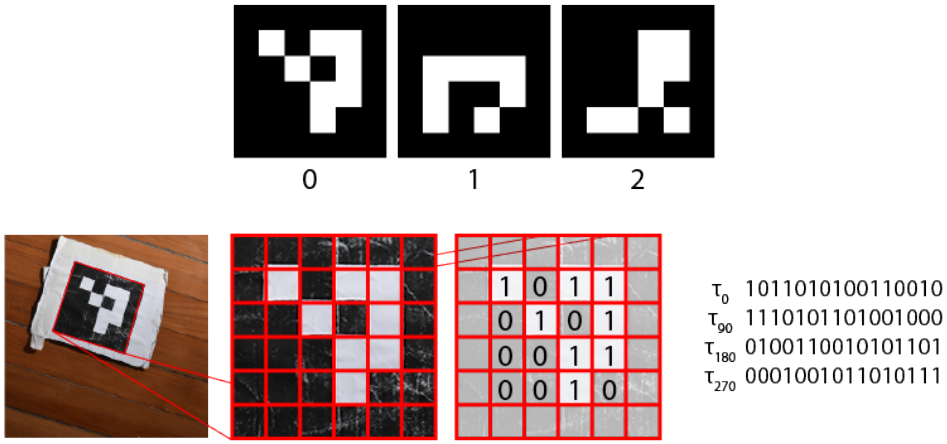


Figura 21: Marcadores utilizados y su codificación.

Siendo el marcador detectado k, R_γ la rotación de la tupla γ en grados y $H(m_i, m_j)$ la distancia de Hamming entre los valores de las tuplas m_i y m_j:

$$\mathcal{D}(m_i, m_j) = \min_{\theta \in \{0\,90\,180\,270\}} \{H(m_i, R_\theta(m_j))\}$$

$$\mathcal{S}(k) = i : \min_{m_i \in D_{50}} \{\mathcal{D}(k, m_i)\}$$

Donde $\mathcal{D}(m_i, m_j)$ es la distancia mínima de Hamming entre la tupla m_i y las rotaciones de la tupla m_j y $\mathcal{S}(k)$ es el identificador de la tupla con menor distancia a cualquiera de las rotaciones de las tuplas en el diccionario D_{50}.

Luego de identificado el marcador, se debe hacer una estimación de la posición del marcador respecto a la cámara. La salida del módulo de detección de los marcadores es una lista de tuplas $(x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, i)$, una tupla por cada marcador identificado. x_j e y_j con $j \in 0, \ldots, 3$ son las posiciones de las esquinas del marcador en la imagen, en sentido horario, empezando por la esquina superior izquierda e i es el identificador del marcador detectado.

Suponiendo que la cámara ya se encuentra calibrada (ver 3.2.1), se puede estimar la pose de los marcadores, es decir su posición y orientación, respecto a la cámara. Como conocemos las dimensiones de los mismos, la estimación

puede hacerse en unidades físicas. Esta estimación se realiza mediante el método *solvePnP* de la librería de procesamiento de imágenes *OpenCV* (2018a). Este método permite calcular la posición y orientación de un objeto, a partir de correspondencias entre puntos 3D y 2D e información de de la cámara.

La salida de este algoritmo es un conjunto de rotaciones y traslaciones en el espacio 3D. Estas representan la posición y pose de la cámara respecto a un sistema de coordenadas de mano derecha ubicado en el centro del marcador, con el eje Z saliendo del frente del marcador perpendicularmente. Estos son los llamados parámetros extrínsecos que conforman la matriz extrínseca (*view matrix*). La rotación se encuentra en una representación ángulo-eje que se puede pasar a una representación matricial mediante el método *rodrigues*.

Si tomamos t como el vector traslación, r el vector rotación en representación eje-ángulo y $\mathcal{R}$ la transformación que pasa de la representación eje-ángulo a matricial.

$$R = \mathcal{R}(r)$$

La rotación R y la traslación t son la rotación y traslación del eje de coordenadas local del objeto con respecto al de la cámara. Por lo tanto,

$$x' = Rx + t$$

x' es el punto en coordenadas de la cámara y x es el punto en coordenadas locales del objeto. Sabiendo que la inversa de una matriz de rotación es su traspuesta:

$$\begin{gathered} x' - t = Rx \Rightarrow \\ R^T(x' - t) = R^T Rx \Rightarrow \\ R^T x' - R^T t = x \end{gathered}$$

Obtenemos así $R^T x' - R^T t = x$ que representa el pasaje de coordenadas locales de la cámara a coordenadas del objeto. Por lo tanto, el centro $(0, 0, 0)$ de la cámara se encuentra en el punto $-R^T t$ y los ejes se encuentran rotados por la matriz R^T.

Descomponemos la rotación R^T con ayuda del método *decomposeProjection-Matrix* en *ángulos Euler (Slabaugh 1999)*: (ϕ_x, ϕ_y, ϕ_z). Como asumimos que los marcadores se encuentran planos sobre el piso, asumimos $(0, 0, \phi_z)$. Considerando la posición del marcador en coordenadas *world* (coordenadas absolutas dentro del escenario de vuelo) cómo (M_x, M_y, M_z) y su rotación sobre el eje vertical M_γ, la posición $P = (C_x, C_y, C_z)$ y rotación C_ρ de la cámara en coordenadas *world*:

$$
\begin{gathered}
(c_x, c_y, c_z) = -R^T t \\
(\phi_z, \phi_y, \phi_z) = decompose(R^T) \\
(C_x, C_y, C_z) = (M_x + c_x, M_y + c_y, altitud) \\
C_\rho = M_\gamma - \phi_z
\end{gathered}
$$

Esta operación es realizada para los n marcadores detectados y se toma el promedio de las posiciones y las orientaciones de los marcadores como:

$$
\begin{gathered}
P = \frac{1}{n} \sum_{i=1}^{n} P_n \\
C_\rho = atan2 \left(\sum_{i=1}^{n} \sin C_{\rho,i}, \sum_{i=1}^{n} \cos C_{\rho,i} \right)
\end{gathered}
$$

Se utiliza la media aritmética de las posiciones y la media de las medidas circulares para las orientaciones.

Para evitar valores atípicos (*outliers*) en la estimación de cada posición, particularmente cuando son detectados pocos marcadores, se almacenan las posiciones estimadas $H_m, H_{m-1}, \ldots, H_2, H_1$ de los últimos m cuadros y se calcula la media ponderada tomando como el peso $w_i = i$

$$P_{smooth} = \frac{\sum_{i=m}^{1} w_i H_i}{\sum_{i=m}^{1} w_i} = \frac{2\sum_{i=m}^{1} iH_i}{w_i(w_i+1)}$$

Esto nos determina una posición final P_{smooth} y una orientación C_ρ. Para poder estimar los movimientos a partir de estos valores y recorrer el camino especificado se calculan una serie de variables de interés basadas en *steering behaviours* (Reynolds 1999; Shiffman 2012): próxima posición, próxima posición proyectada en el camino y objetivo a obtener, ilustradas en la figura 22.

Considerando a D_i como el intervalo de tiempo transcurrido en μs entre el cuadro $i-1$ y el cuadro i en los últimos m cuadros y utilizando nuevamente una media ponderada de los últimos m desplazamientos, se calcula la posición esperada para el D_i.

$$P_{next} = H_i + \frac{2\sum_{i=m}^{1} i(H_i - H_{i-1})}{w_i(w_i+1)\sum_{i=m}^{1} iD_i}$$

Notando el segmento del camino que recorre el dron en este momento como el segmento orientado $\overrightarrow{AB}$, podemos tomar el punto proyectado como:

$$\overrightarrow{v} = B \hat{-} A$$
$$\overrightarrow{x} = P_{next} - A$$
$$P_{proj} = A + \overrightarrow{v}(\overrightarrow{x} \cdot \overrightarrow{v})$$

El punto proyectado se desplaza una cantidad constante c sobre el segmento $\overrightarrow{AB}$ y se obtiene el objetivo a seguir:

$$P_{displaced} = P_{proj} + c\,\overrightarrow{v}$$

$$P_{target} = \begin{cases} A & \overrightarrow{x} \cdot \overrightarrow{v} + c < 0 \\ B & \overrightarrow{x} \cdot \overrightarrow{v} + c > \|\overrightarrow{AB}\| \\ P_{displaced} & \end{cases}$$

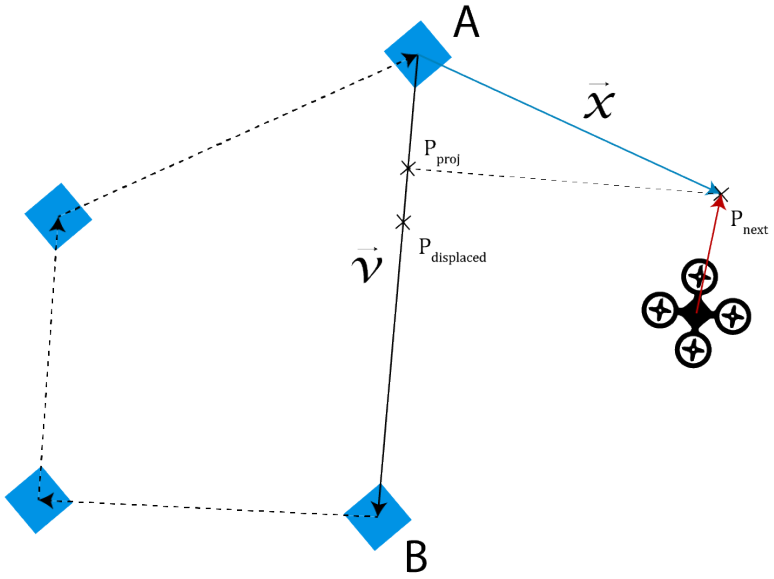


Figura 22: Proyección sobre el camino de la posición actual.

6.1.2. Evasión de obstáculos dinámica

Como ya mencionamos, los drones utilizados poseen una sola cámara monocular. El problema con las cámaras monoculares es la dificultad en extraer información de profundidad. La información de profundidad permite determinar la proximidad de los objetos en el entorno así como la velocidad con la que nos acercamos a ellos.

El enfoque que se tomó en la implementación desarrollada toma un enfoque indirecto para la detección de proximidad. Este se basa en el propuesto en el trabajo publicado por Sagar (2014). Este método consiste en cuatro etapas: sustracción de fondo, flujo óptico, agrupamiento (*clustering*) y segmentación

de profundidad o estimación de proximidad. La última etapa ofrece dos posibles métodos ensayados.

En lo que refiere a la sustracción de fondo, muchos métodos emplean la información de la variación en los píxeles de un fotograma a otro para separar un fondo estático de un frente dinámico. El problema con este enfoque es que no se puede emplear en el caso de un dron en movimiento. Por eso se emplea la técnica conocida como mixtura de gaussianos (mixture of gaussians, MOC) para generar la máscara para remover el fondo (Zivkovic 2004). Mediante la aplicación de esta técnica, se elimina gran parte del fondo de la imagen tomada por el dron. Luego se genera una máscara con áreas de un tamaño mínimo mediante el dibujo de contornos. A continuación se filtran los puntos característicos (*feature points*) relevantes de modo de realizar el cálculo de los vectores de flujo solo en puntos dónde resulten eficientes. Finalmente se utiliza la técnica de Lucas-Kanade para calcular los vectores de flujo (Bouguet 2000).

Los puntos encontrados son agrupados en clusters por su proximidad y filtrados por el tamaño mínimo del cluster a partir del algoritmo DBSCAN (Ester et al. 1996).

Finalmente la segmentación de profundidad es una de las opciones para generar la información necesaria para realizar una posible maniobra de evasión. Para esta opción se asume que un objeto a mayor distancia cambia su posición a menor velocidad respecto a aquellos más cercanos. Los vectores de flujo óptico producidos por el algoritmo Lucas-Kanade son segmentados en cercanos y lejanos a partir de su norma. Para realizar la segmentación se toma la media de la mayor y menor norma:

$$l_{min} = \min_{p \in features} \|p_{fin} - p_{inicio}\|$$
$$l_{max} = \max_{p \in features} \|p_{fin} - p_{inicio}\|$$
$$l_{threshold} = \frac{l_{min} + l_{max}}{2}$$

La estimación de proximidad es otro método para determinar la proximidad de un objeto es comparando que tan rápido aumenta de tamaño en el campo visual al acercarse. Cuanto más velozmente lo haga, más cerca consideramos que se encuentra. Para obtener la estimación de proximidad se reutilizan

los grupos obtenidos por el algoritmo DBSCAN. De estos grupos se obtiene los centros así como su cuadro delimitador (*bounding box*). Luego se obtiene la porción de la imagen que corresponde a este cuadro delimitador tanto en el cuadro anterior como el actual. La porción de la imagen anterior es comparada en varias escalas con la porción de la imagen actual. Se considera la escala que genera la mejor coincidencia con la imagen actual. A mayor la escala de la versión con mayor coincidencia, más cerca se considera el objeto frente a otros con menor valor de escala.

El prototipo desarrollado que implementa este sistema es capaz de detectar potenciales obstáculos pero no actúa en consecuencia. Es decir que toma los cuadros de entrada, los procesa y muestra los resultados en un visor. Como se mencionó por motivos de tiempo y complejidad no se alcanzó a implementar un mecanismo que generase comandos de movimiento que permitiesen al dron evitar los obstáculos detectados.

6.1.3. Localización y Mapeo Simultáneo

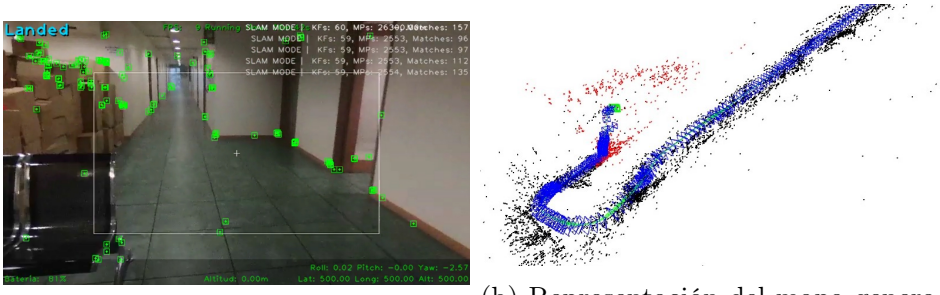
La localización y modelado simultáneo (simultaneous localization and mapping, SLAM), se define como el problema de construir o actualizar un mapa del entorno, al mismo tiempo que se determina la posición de un agente dentro de él.

Existen múltiples soluciones al problema de construcción de mapas que emplean métodos estadísticos e imponen diferentes restricciones sobre las condiciones de trabajo. Los mecanismos de sensado del entorno son una de estas restricciones y muchas veces hacen uso de cámaras binoculares o monoculares, telémetros láser de barrido o en casos raros, sensores táctiles. Además la información de estos sensores suele ser combinada con información de odometría del robot para mejorar las estimaciones del modelo.

Para este proyecto de grado, las restricciones de la plataforma implicaron el uso de una cámara RGB monocular sin información odemétrica ya que el SDK de los drones utilizados no permite acceder a ella. Además, el procesamiento debía ser en línea con un tiempo de respuesta razonable. El uso de una cámara monocular sin otra información hace que las mediciones del entorno sean relativas, por lo que aún siendo correcto el mapeo, la unidad de escala del mapa sería, al menos en principio, desconocida. Es con odometría que muchas veces se compensa esto ya que las mediciones vienen acompañanadas de dimensiones conocidas.

Habiendo investigado múltiples alternativas, se decidió utilizar la librería ORB-SLAM2 (Mur-Artal et al. 2015; Mur-Artal et al. 2017). Esta librería presenta múltiples ventajas dado el escenario y las restricciones planteadas:

- **Open Source:** su código se encuentra disponible para su estudio y modificación bajo licencia GPLv3.
- **Actualizado y activo:** durante la realización de este trabajo, se encontraba en desarrollo activo y con soporte.
- **Stack tecnológico:** el stack de tecnologías que utiliza es compatible y similar a el utilizado en este proyecto.
- **Monocular:** soporta nativamente el escenario de SLAM visual monocular además de RGB-D y Stereo.
- **Otras características:** el algoritmo en el que se basa esta librería fue diseñado para procesamiento en tiempo real. Además soporta cierres de bucle (*loop closing*) para detectar los bucles en el recorrido del agente, recalculando la posición y ajustando el mapa en consecuencia. También ofrece dos modalidades de trabajo: SLAM y localización. La primera realiza seguimiento de la posición, mapeo y cierres de bucle mientras que la segunda solo intenta determinar la posición del agente dado un mapa ya generado, resultando su ejecución mucho más liviana.



(a) Identificación de los *features*

(b) Representación del mapa generado

Figura 23: ORB SLAM integrado al dron.

La librería ORB-SLAM2 basa su procesamiento en tres hilos de ejecución. El primer hilo se encarga de la localización buscando coincidencias de características (*features*) con el mapa generado. El segundo hilo se encarga de la generación y optimización del mapa. El tercer hilo se encarga de la detección

de bucles y corrección de la desviación. Para poder utilizarla es necesario el envío cuadro a cuadro de las imágenes tomadas por la cámara asociados a una marca temporal. Luego del procesamiento del un cuadro la salida ofrece un vector de coordenadas de ubicación, en base a un sistema de coordenadas tridimensional, que representan las características mapeadas. Además ofrece un vector de coordenadas de ubicación y uno de rotación que representan la localización de la cámara. Esta información puede ser representada para reconstruir el mapa y el recorrido de la cámara como muestra la figura 23.

En los inicios de nuestra investigación se apuntó a utilizar un sistema de SLAM como base de la navegación. Luego de un primer reconocimiento del lugar, se pretendía tomar como base la localización calculada a través de este método para definir los recorridos a realizar por los drones de la flotilla. De esta forma se podría tener en cuenta perfectamente las características del lugar y evitar obstáculos, mediante el análisis del mapa generado. Sin embargo por la capacidad de cómputo necesaria y la falta de robustez general de las soluciones investigadas se optó por la navegación con marcadores como solución final.

6.2. Detección, seguimiento y persecución

Para la detección de intrusos se utilizaron técnicas de reconocimiento de imagen. Cada dron ejecuta un algoritmo que procesa la información proveniente de su cámara, en busca de personas. Como se mencionó anteriormente, los drones empleados cuentan con una única cámara RGB, por lo que los algoritmos empleados debieron tener en cuenta esta restricción.

El proceso de detección de intrusos desarrollado consta de tres partes: detección, seguimiento (*tracking*) y persecución.

- **Detección:** durante esta etapa se analizan los cuadros de imagen recibidos a través de la cámara en busca de figuras humanas. Esta es la etapa computacionalmente más exigente.
- **Seguimiento:** luego de una detección el objetivo es actualizar la ubicación de la detección en las capturas subsiguientes de la cámara (figura 24).
- **Persecución:** conjuntamente con la parte anterior se ejecuta la persecución. Esta etapa implica estimar la posición del intruso con respecto

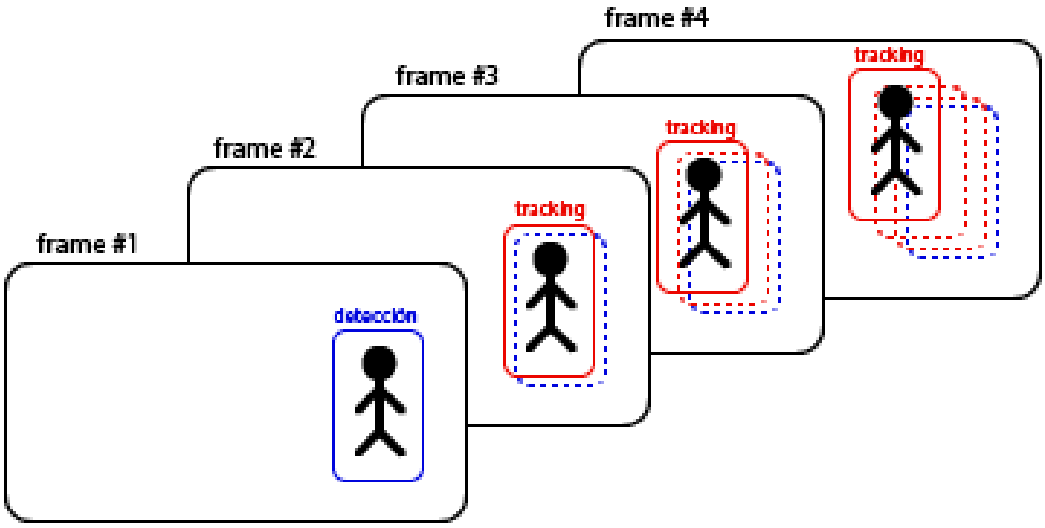


Figura 24: Combinación de detección y seguimiento.

al dron y calcular los movimientos necesarios para mantenerlo dentro del campo visual y lo suficientemente cerca.

El algoritmo de detección se ejecuta cada N frames y se mantiene ejecutando durante M frames (ambos valores configurables) para obtener candidatos (figura 25). Durante todos los frames de detección se acumulan las posiciones de las detecciones.

Figura 25: Secuencia de detecciones y tracking.

El algoritmo empleado para la detección es el conocido como histograma de gradientes orientados (histogram of oriented gradients , HOG) (Dalal y Triggs 2005). Se utiliza la implementación de la librería *OpenCV* y el detector entrenado por el proyecto *vision-ary* (2018). Antes de alimentar el algoritmo el cuadro obtenido de la cámara es escalado para mejorar su eficiencia.

Antes de pasar a la etapa de seguimiento se toman tanto los objetos de la iteración anterior del algoritmo así como las detecciones acumuladas en la nueva iteración. Los candidatos pasan por un filtro para determinar cuales

se consideran objetivos válidos. Para esto se aplican los siguientes filtros (figura 26):

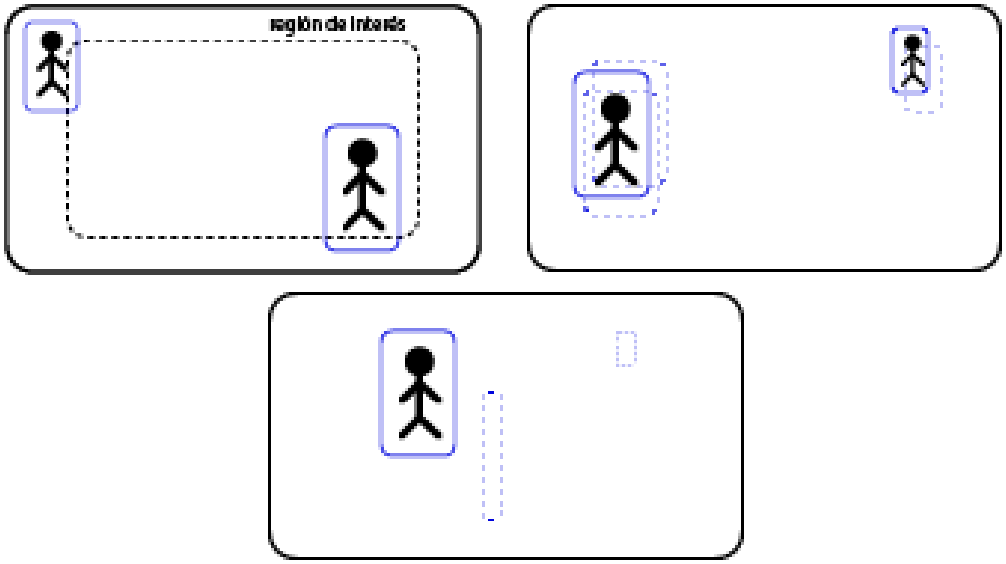


Figura 26: Filtro de detecciones por región de interés, área y tamaño y non maxima supression.

- **Región de interés:** las detecciones se filtran contra una región de interés. Si el centro de la detección se encuentra fuera de esta región de interés se descarta la detección.
- **Área y tamaño:** las detecciones que no cumplen con un criterio de dimensiones y área mínimo son descartadas.
- **Non maxima supression:** luego de los filtros anteriores, cuando múltiples detecciones se solapan más de un cierto porcentaje, se toma la primera de las detecciones solapadas ordenadas por el eje x, luego el y (Hosang et al. 2017).

Obtenidos y filtrados los candidatos a seguir se les asigna un número de identificación. Aquellas detecciones que provienen de una iteración anterior conservan su identificador. El algoritmo de seguimiento utilizado también es implementado por la librería *OpenCV* y es una abstracción sencilla que permite realizar el seguimiento a múltiples objetivos ejecutando el algoritmo sobre cada objetivo.

Para poder llevar a cabo la persecución, se elige la detección de la lista de detecciones cuyo centro se encuentre más cerca del centro del cuadro. El objetivo de la persecución es mantener al intruso lo más centrado horizontal y verticalmente con el fin de mantenerlo en el campo de visión (figura 27).

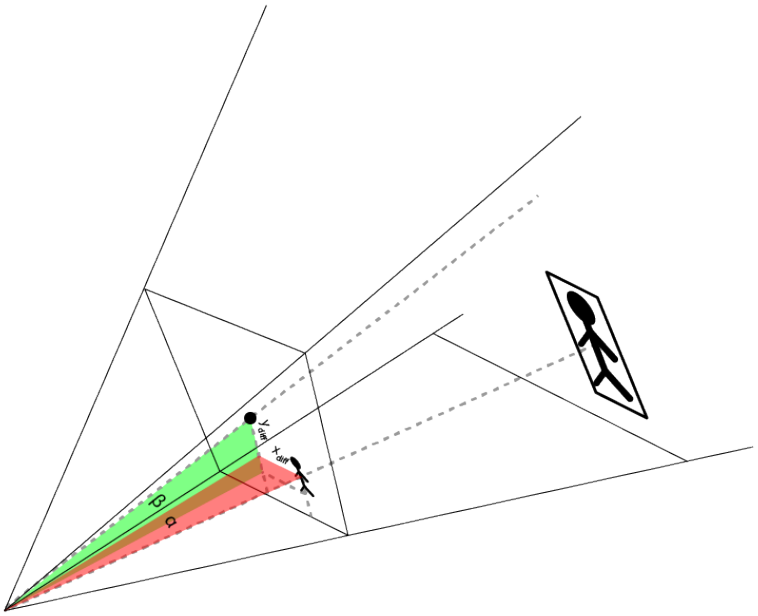


Figura 27: Cálculo del desplazamiento horizontal y vertical respecto al centro del cuadro.

Para centrar al intruso en cada cuadro de imagen, se estima en el eje horizontal el ángulo que tiene el objetivo con el rayo que pasa por el centro de la línea de visión de la cámara. Para esto se toma la distancia $x_{diff} = x_{centro} - x_{track}$ siendo x_{centro} el valor en el eje x del centro del cuadro y x_{track} el valor en el eje x del centro del objetivo. El ángulo α queda dado por:

$$\alpha = \tan^{-1}\left(\frac{2x_{diff}}{ancho_{frame}}\tan\left(\frac{FOV}{2}\right)\right)$$

Donde FOV es el campo de visión horizontal de la cámara y $ancho_{frame}$ el ancho del cuadro de la cámara en píxeles. Al ángulo *alpha* también le podemos llamar *pan* (del inglés *panning*). A partir de este valor se calcula la velocidad de rotación a ejercer. Para valores absolutos mayores a un umbral *YawAproximationAngle* predefinido, el valor absoluto de la velocidad se topea a un valor máximo configurable. Para valores menores el valor se interpola linealmente entre la velocidad máxima y 0.

El cálculo de y_{diff} es similar al anterior con la diferencia que se considera el valor y de la base del objetivo seguido: $y_{diff} = y_{centro} - y_{abajo}$. En este caso el objetivo es mantenerse a una distancia objetivo del objeto detectado (figura 28). Para obtener esta distancia se asume: que el objeto se encuentra siempre parado verticalmente sobre el piso, que se conoce el ángulo *tilt* de inclinación de la cámara con respecto al piso sobre el que se encuentra parado el objeto y la altitud h del dron coincide con la diferencia de altura respecto a los pies del sujeto seguido.

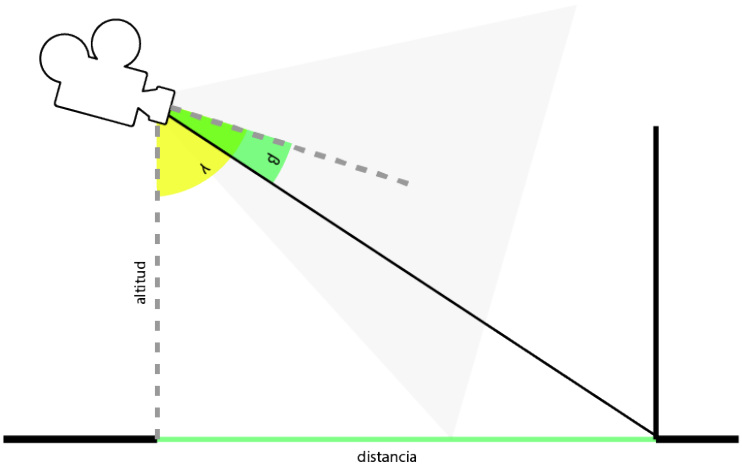


Figura 28: Cálculo de distancia al objetivo de lado.

Se considera FOV_v como el campo de visión vertical que se puede deducir del FOV horizontal y las dimensiones del cuadro $(ancho_{frame}, alto_{frame})$: $FOV_v = \frac{FOV \cdot alto_{frame}}{ancho_{frame}}$.

La distancia d al objeto está dada entonces por:

$$\gamma = 90 - tilt - \beta$$

$$d = h \tan^{-1}(\gamma)$$

En la figura 29 se puede ver la distancia d estimada según la posición y_{track} de la base del seguimiento en el eje y. Los valores negativos se dan cuando la base se encuentra por sobre el horizonte, situación que se descarta dado que asumimos que los objetos se encuentra apoyados sobre el plano horizontal $h = 0$. Es importante notar como los objetos cercanos al dron gozan de

mucho mayor precisión en su estimación que aquellos cercanos en el eje y al horizonte.

La posición del horizonte en el eje y queda determinada por el valor del $\beta = -tilt$, es decir:

$$y_{horizonte} = \left(1 - \frac{\tan(-tilt)}{\tan\left(\frac{FOV_v}{2}\right)}\right) \frac{ancho_{frame}}{2}$$

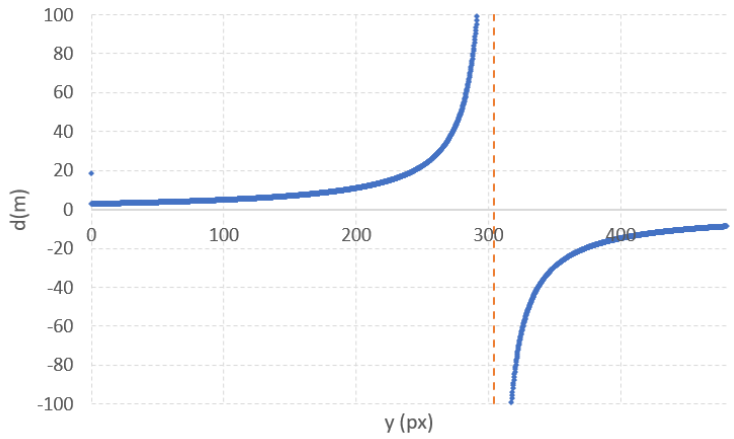


Figura 29: Distancia horizontal en metros al objetivo dada la posición en el eje y de su base para $tilt = \frac{\pi}{12}$, $FOV = \frac{2\pi}{3}$, $alto_{frame} = 640$, $ancho_{frame} = 480$ y $h = 5$. $h_{horizonte} \approx 304,3078$

Para calcular la velocidad lineal a ejercer se toman en cuenta dos constantes: la distancia objetivo a la que mantenerse del objeto seguido (*FollowerTargetDistance*) y el margen de reducción de velocidad (*TargetSlowdownRadius*). Cuando la distancia al objetivo es mayor a *FollowerTargetDistance* + *TargetSlowdownRadius* la velocidad toma un valor máximo predefinido. Cuando el objetivo se encuentra a menor distancia, la velocidad se interpola linealmente entre el máximo y 0 (figura 30). Se hace notar que en caso que la distancia al objeto es menor a la distancia objetivo el dron no retrocede para evitar colisiones traseras.

(a) Velocidad angular (b) Velocidad lineal

Figura 30: Curvas de velocidades durante el seguimiento de intrusos.

7. Carga autónoma y coordinación

Este capítulo presenta las soluciones propuestas para la carga autónoma y la coordinación de la flotilla. La sección 7.1 aborda la carga autónoma, mientras que la sección 7.2 aborda la coordinación de la flotilla.

7.1. Carga autónoma

Para lograr que la flotilla de drones sea totalmente autónoma se debe prever la recarga de baterías. Para esto se utilizó la plataforma de carga descrita en la sección 4.3. El mecanismo diseñado permite que cuando el dron requiera recargar su batería, pueda aterrizar sobre la plataforma y así mantener su tiempo de vuelo de forma indefinida. Combinando este recurso con un mecanismo de coordinación y la cantidad suficiente de drones, es posible mantener siempre al menos un dron en vuelo de forma totalmente autónoma.

Para que el dron sea capaz de aterrizar sobre la plataforma de carga de forma autónoma fue necesario resolver dos problemas: navegar hasta la plataforma y aterrizar sobre ella. Para poder navegar hacia la plataforma y sobrevolarla se utiliza el sistema de navegación por marcadores descrito en la sección 6.1. La ubicación de la plataforma es conocida y se obtiene a partir del archivo de configuración de cada dron de la flotilla. Luego de que el dron se encuentra sobrevolando la plataforma, la ubicación obtenida por los marcadores de navegación no es lo suficientemente precisa para aterrizar sobre ella. Por este motivo se diseñó e implementó un mecanismo de control de aterrizaje, que también se basa en marcadores. Sin embargo, los marcadores de aterrizaje son reconocidos y procesados de forma diferente a los marcadores de navegación, lo que permite mayor precisión en el cálculo de la ubicación de la plataforma con respecto al dron.

Los marcadores de aterrizaje constan de cuadrados ubicados de forma concéntrica. Los cuatro marcadores empleados son exactamente iguales y se ubican en los extremos de la plataforma de carga (figura 31). Para procesar estos marcadores solo se toma en cuenta su ubicación dentro del cuadro de la imagen tomada por la cámara, que es enfocada hacia abajo para poder reconocerlos. El cuadro de imagen tomado por la cámara del dron se interpreta como un sistema espacial donde el píxel es la unidad de medida y el punto $(0, 0)$ se ubica en el extremo inferior izquierdo de la imagen. El eje x es el borde inferior y el eje y el borde izquierdo de la imagen. La ubicación

de cada marcador se codifica entonces como una coordenada (x, y).

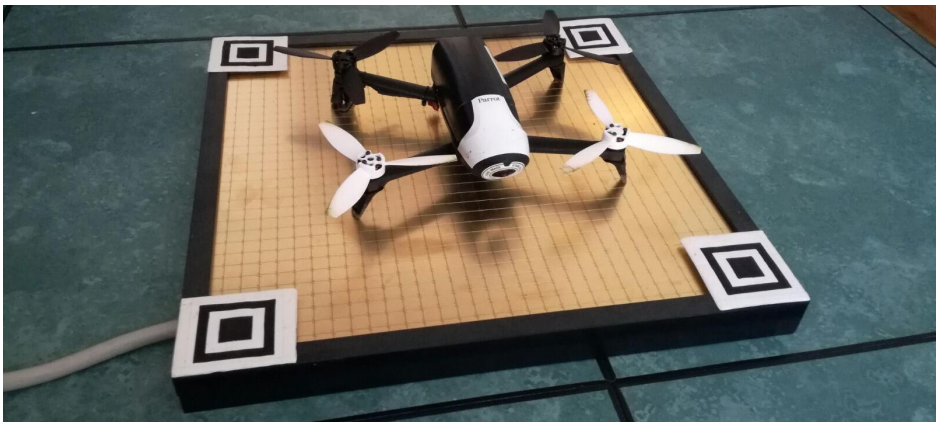


Figura 31: Ubicación de marcadores de aterrizaje.

El sistema de alineación se basa en el cálculo del centro de la plataforma utilizando la ubicación de los marcadores detectados. Una vez calculado el centro, la alineación consiste en hacer coincidir el centro del cuadro de la imagen tomada por la cámara, con el centro de la plataforma de carga. Una vez el dron está alineado, este desciende ella.

Para calcular el centro de la plataforma una vez que el dron se encuentra sobre ella, se elige la posición de los primeros tres marcadores detectados:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3)$$

Luego se toma la posición del primer marcador y calcula la de su opuesto, es decir el que está a mayor distancia euclidiana:

$$(x_o, y_o) = \underset{(x,y)}{\text{argmax}} \quad \mathcal{D}((x_1, y_1), (x, y))$$

El centro de la plataforma se estima entonces como, el punto medio del vector resultante entre los dos marcadores tomados como referencia:

$$(x_{pc}, y_{pc}) = ((x_1 + x_o)/2, (y_1 + x_o)/2)$$

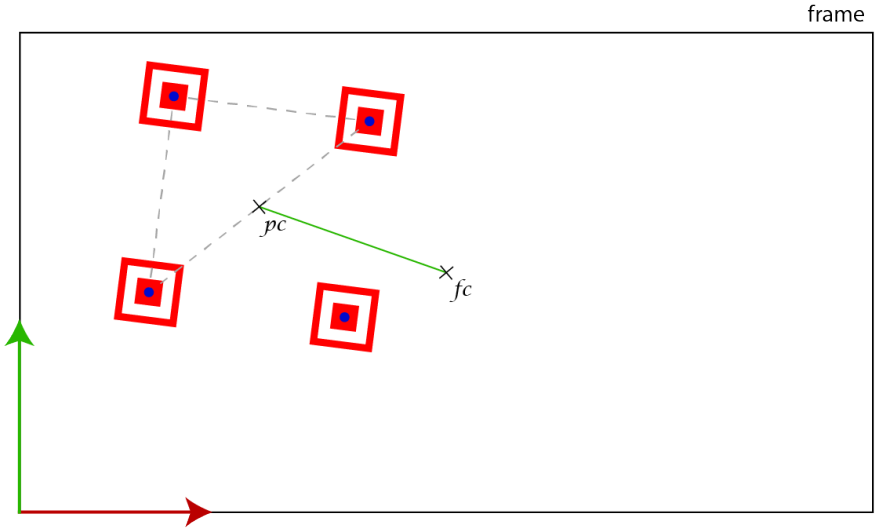


Figura 32: Estimación del centro de la plataforma.

Una vez estimado el centro de la plataforma, se calcula la velocidad y orientación de los movimientos que debe hacer el dron para corregir su posición y alinearse con la plataforma. Estas velocidades se calculan en base a la distancia relativa entre centro de la plataforma (fp) y el centro de la imagen (fc) (figura 32).

Los movimientos que debe hacer el dron en cada momento para corregir su posición se descomponen en un componente de cabeceo ($pitch$) y uno de alabeo ($roll$). El cabeceo permite reducir la distancia en el eje y y el alabeo permite reducir la distancia en el eje x.

Para calcular ambos componentes de la velocidad primero se calcula la distancia entre el centro del cuadro y el centro de la plataforma. Luego esta se normaliza dividiendo sobre la máxima distancia posible (mitad del cuadro en el eje correspondiente). Finalmente para controlar con que velocidad el dron se acerca a la plataforma, el resultado se multiplica por un factor configurable:

$$pitch = ((y_{pc} - y_{fc})/y_{fc}) * pfactor$$
$$roll = ((x_{pc} - x_{fc})/x_{fc}) * rfactor$$

Este cálculo contempla el caso de que sean reconocidos por lo menos tres marcadores. Sin embargo en el caso de que solamente se reconozcan dos marcadores, se omite el cálculo del opuesto y se toma el centro de la plataforma como el punto medio entre los marcadores detectados. A su vez en el caso donde solo se vea un solo marcador, se toma el centro de la plataforma como su posición. Luego en los dos casos se utiliza la misma fórmula para calcular los movimientos de cabeceo y alabeo.

Es importante tener en cuenta que la estimación del centro de la plataforma, cuando se cuenta con menos de tres marcadores como referencia generalmente difiere de la posición real. Sin embargo cuando esto sucede, mediante la alineación al centro estimado, el dron se acercará al centro real. Esto permite que más marcadores entren en foco y sean reconocidos, logrando así un cálculo más exacto.

Para determinar cuando el dron está suficientemente alineado con la plataforma, se define un rango de tolerancia para cada eje. Cuando la distancia entre el centro del cuadro y el centro de la plataforma está dentro del rango, el dron está listo para descender sobre la plataforma. Sin embargo este margen solo es válido mientras el centro se estime con tres marcadores, evitando aterrizar en otro caso por falta de certeza en la estimación.

Al comenzar el descenso, el dron realiza un movimiento de ajuste. Este ajuste permite centrar mejor el brazo del dron donde se encuentran los contactos de carga. Los drones utilizados poseen estos contactos en el brazo trasero derecho. Por lo que el movimiento lleva el dron hacia adelante y hacia la izquierda. El valor de ajuste en cada eje se definió de forma heurística y puede ser ajustado desde el archivo de configuración de cada dron.

7.2. Coordinación de la flotilla

Para lograr la coordinación entre los drones de la flotilla, fue necesario desarrollar dos componentes claves, descritos en las próximas subsecciones:

- **Comunicación:** permite distribuir en todo momento la información sobre el estado interno de cada dron y del entorno que este percibe. La información de estado de los drones a distribuir incluye, la tarea que está realizado el dron, su posición, nivel de batería, etc. Por su parte del entorno se distribuye información sobre la presencia de intrusos en la zona vigilada y su ubicación.

- **Toma de decisiones:** permite utilizar toda la información recolectada directamente por el dron y la información recolectada mediante la comunicación con el resto de la flotilla para la toma de decisiones. Estas decisiones permiten a la flotilla trabajar como una unidad, colaborando para dar seguimiento a los intrusos y coordinar el uso de la plataforma de carga.

7.2.1. Comunicación

Para definir cualquier mecanismo de comunicación se necesita: un medio a través del cual efectuar la comunicación y un protocolo que establezca cuándo y qué información se intercambia.

Como medio de comunicación se eligió una red inalámbrica local (*WiFi*). Además se consideraron dos alternativas para la topología de la red (figura 33). Una alternativa es una red centralizada en modo infraestructura (*infraestructure*) mediante la utilización de uno o varios puntos de acceso (AP) externos que tengan cobertura en la totalidad de la zona de vuelo. Este tipo de red tiene como ventaja ser más simple de implementar pero necesita uno o varios puntos de acceso con potencia suficiente. La otra alternativa es una red de malla distribuida (*mesh*) donde cada dron funciona como nodo de la red, recibiendo y distribuyendo información propia o del resto de la flotilla para permitir la comunicación sin necesidad de hardware adicional.

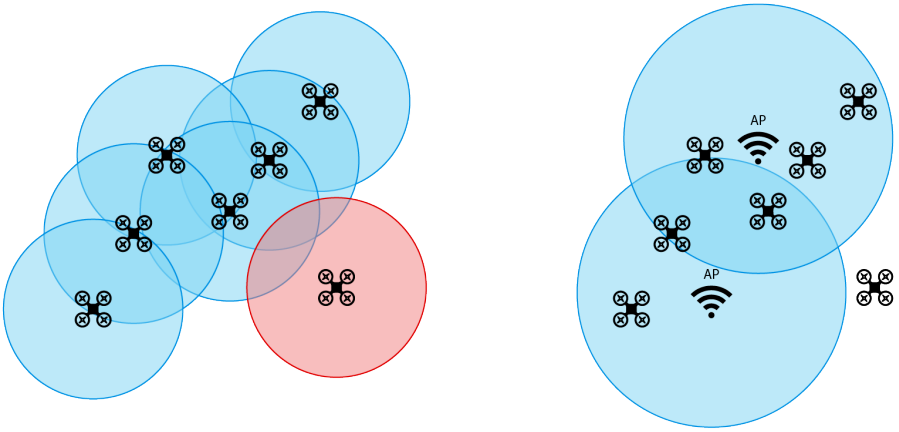


Figura 33: Red *mesh* vs centralizada.

A pesar de que una red de malla no necesita hardware externo, este tipo de red es más compleja de implementar ya que requiere un manejo a nivel del

sistema operativo de la red o un manejo a nivel de aplicación pero alcanzando una cobertura dependiente de la distribución de los drones en el espacio. BATMAN adv (2018) es un módulo del núcleo de linux que provee enrutamiento a nivel de capa 2 para *mesh*. Por otro lado, una red centralizada es más simple de implementar y la cobertura de la red depende de los dispositivos utilizados, permitiendo ajustar el nivel de cobertura a las necesidades del caso. Sin embargo es menos portable que una red de malla y requiere un mayor costo de implantación.

En su configuración de fábrica, los drones ofrecen una red *adhoc* proporcionando un punto de acceso *WiFi*. Para poder comunicar los drones se debe proveer una forma de ruteo (*routing*) entre los drones, ya sea mediante una red de malla o de infraestructura. Implementar cualquiera de las soluciones requiere la modificación de la configuración por defecto de los drones, pero puede hacerse de forma transparente a la capa de aplicación.

Como protocolo para realizar la comunicación se eligió el llamado difusión amplia (*broadcast*). A la hora de comunicar su estado a los demás, cada dron envía un mensaje utilizando este protocolo. Este mensaje es la única vía de comunicación entre drones por lo que incluye toda la información que es necesaria para la toma de decisiones. La difusión amplia puede ser adoptada fácilmente en tanto en en una red centralizada, como en una red de malla debido a que no es necesario el establecimiento de una conexión punto a punto.

Cada dron de la flotilla distribuye su estado a los demás utilizando un intervalo de tiempo configurable t_1. Para evitar la información desactualizada se utiliza un tiempo de expiración t_2, también configurable, a partir del cual se asume el último mensaje de un dron cualquiera de la flotilla como expirado y se asume la pérdida de comunicación.

La información de este mensaje incluye los siguientes campos:

- **ip**: dirección de red IPv4 desde donde se envía el mensaje.
- **port**: puerto UDP desde donde se envía el mensaje.
- **drone id**: identificador del dron.
- **seq num**: número de secuencia utilizado por el receptor para ordenar los mensajes.
- **name**: nombre del dron.

- **position**: coordenadas de ubicación del dron con respecto al sistema de marcadores.
- **rotation**: vector de rotación del dron con respecto al sistema de marcadores.
- **current task**: nombre de la tarea actual que el dron está realizando.
- **battery level**: nivel de batería actual
- **pad in use**: identificador del pad de carga en el cual el dron se dispone a cargar.
- **covered drone id**: identificador del dron al que se está realizando cobertura
- **followed position**: coordenadas de ubicación del intruso al que se está siguiendo.

7.2.2. Toma de decisiones

Existen dos modelos básicos para construir un sistema de toma de decisiones de forma colectiva. El modelo jerárquico, donde uno de los componentes es el encargado de tomar las decisiones y los demás obedecen sus órdenes. Y el distribuido, donde la responsabilidad se distribuye entre los participantes. En este último no hay un único tomador de decisiones, si no que las decisiones son tomadas en conjunto por todos los integrantes del sistema. Las decisiones provienen entonces del comportamiento emergente del sistema a partir del actuar individual de sus participantes.

El método elegido se construyó en base al modelo de toma de decisiones distribuido. En la solución planteada el sistema de toma de decisiones esta integrado por un conjunto de drones, los cuales en base a su estado y el último estado conocido de los demás, determina las acciones a realizar para vigilar la zona, realizar coberturas, recargar sus baterías y colaborar en la persecución de intrusos. Todos los drones operan con el mismo algoritmo por lo que su funcionalidad no se diferencia.

El componente de toma de decisiones de cada dron fue modelado mediante dos componentes: una planificación fuera de línea y una máquina de estados ejecutada en línea. La planificación fuera de línea define cual es el camino (*path*) que debe seguir cada dron durante el patrullaje. Además se definen

rutas de cobertura a fin de mantener el área vigilada cuando algún dron de la flotilla está recargando su batería. Todas estas rutas son especificadas mediante el archivo de configuración.

El componente que toma las decisiones en línea, la máquina de estados, determina que tarea debe realizar el dron en cada momento. El cerebro, descrito en la sección 5.1, ejecuta esta maquina de estados en un bucle infinito donde obtiene y procesa la entrada, modifica su estado actual y determina las acciones a realizar. Las transiciones entre estados son provocados por eventos percibidos por el propio dron o por eventos y cambios de estados del resto de drones de la flotilla.

El flujo general de la máquina de estados comienza con el dron inactivo. Luego este pasa a patrullar la zona objetivo. Cuando un intruso invade en la zona y uno de los drones de la flotilla detecta su presencia este comienza a seguirlo y los demás drones de la flotilla se mantienen alerta en caso de que necesiten asistirlo. Luego de que el intruso abandona la zona los drones vuelven a patrullar. En caso de que un dron necesite recargar su batería, este aterriza sobre la plataforma de carga, espera que su batería este lo suficientemente cargada y retoma el vuelo para patrullar.

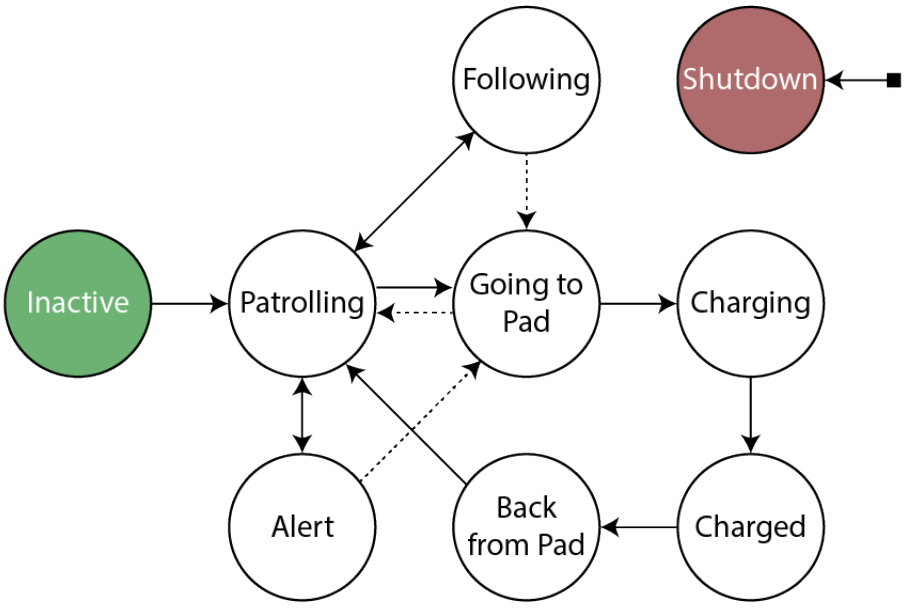


Figura 34: Representación de la máquina de estados implementada en *cerebro*.

En la figura 34 se representan los estados que conforman la maquina de

estados. Las flechas en líneas sólidas representan las transiciones normales mientras que las flechas en líneas punteadas transiciones excepcionales, al alcanzar un nivel crítico de batería o desistir de cargar para evitar una posible colisión cuando dos drones van a cargar al mismo tiempo. A continuación se detallan los estados con una breve descripción de cada uno:

- `INNACTIVE`: estado inicial en el cual inicia la ejecución de la máquina de estados.
- `PATROLING`: indica que el dron está patrullando la zona objetivo en busca de intrusos.
- `FOLLOWING`: indica que el dron ha encontrado un intruso y está realizando su persecución.
- `ALERT`: indica que el dron está alerta ante la presencia de intrusos detectados por otro dron de la flotilla.
- `GOINGTOPAD`: indica que el dron está en camino a la plataforma de carga.
- `CHARGING`: indica que el dron está aterrizado sobre la plataforma de carga recargando su batería.
- `CHARGED`: indica que el dron está aterrizado sobre la plataforma con carga completa, esperando que se libere su ruta para volver a patrullar.
- `BACKFROMPAD`: indica que el dron está en camino a su ruta con el objetivo de patrullar luego de haber cargado su batería.
- `SHUTDOWN`: indica que el dron está en proceso de aterrizar y terminar su ejecución.

Como se mencionó las transiciones representadas como líneas sólidas en la figura 34 son las que integran el flujo normal ya descrito. A continuación se detallan las condiciones para cada transición:

- `INNACTIVE` ⇒ `PATROLING`: se produce una vez el dron comienza su ejecución.
- `PATROLING` ⇒ `FOLLOWING`: se produce cuando un dron está patrullando y detecta un intruso.

- `FOLLOWING` ⇒ `PATROLING`: se produce cuando el intruso abandona la zona objetivo y el dron deja de seguirlo.
- `PATROLING` ⇒ `ALERT`: se produce cuando se detecta la presencia de un intruso, el cual está siendo seguido por otro dron.
- `ALERT` ⇒ `PATROLING`: se produce cuando el intruso abandona la zona objetivo y el dron encargado de seguirlo deja de hacerlo.
- `PATROLING` ⇒ `GOINGTOPAD`: se produce cuando el nivel de batería es bajo y hay una plataforma de carga libre.
- `GOINGTOPAD` ⇒ `CHARGING`: se produce una vez que el dron aterriza con éxito en la plataforma de carga.
- `CHARGING` ⇒ `CHARGED`: se produce cuando la batería está completamente cargada y el dron está listo para volver a patrullar.
- `CHARGED` ⇒ `BACKFROMPAD`: se produce cuando el dron detecta que ningún otro dron está cubriendo su ruta de patrullaje.
- `BACKFROMPAD` ⇒ `PATROLING`: se produce cuando el dron llega a su ruta y vuelve a patrullar.
- ⇒ `SHUTDOWN`: se produce en cualquier momento cuando se ejecuta el comando *shutdown*. Actualmente este comando es disparado presionando la tecla *Esc*.

Existe un umbral configurable debajo del cual el nivel de batería se considera bajo. Pero existe además un segundo umbral, también configurable, debajo del cual el nivel de batería se considera crítico. Este último umbral es utilizado en las siguientes transiciones:

- `FOLLOWING` ⇒ `GOINGTOPAD`: cuando el nivel de batería se considera crítico y existe una plataforma libre. Para evitar que el dron se quede sin batería en mitad de la persecución se cambia de estado a modo de recargar su batería.
- `ALERT` ⇒ `GOINGTOPAD`: esta transición se provoca por el mismo motivo que el anterior, cuando se tiene un nivel de batería crítico se previene quedarse sin batería mientras se está alerta.

Además existe una transición que evita que dos drones vayan a cargar a la misma plataforma de carga para prevenir una colisión:

- `GOINGTOPAD` ⇒ `PATROLING`: esta transición se produce cuando existen dos drones yendo a cargar, pero además lo están haciendo ambos a la misma plataforma. Para evitar una posible colisión, el dron que tenga más nivel de batería deja de hacerlo. En el caso excepcional de que ambos tengan igual nivel de batería, el que tiene el id mayor deja de hacerlo.

Por último, existe un único evento que no se refleja en la máquina de estados. Este evento consiste en la cobertura de la ruta de un dron por parte de otro, mientras el primero recarga su batería. Cuando un dron está patrullando y recibe la información de que otro dron está recargando su batería (`CHARGING`), verifica en su lista de rutas la existencia de un ruta que le permita cubrirlo. Si se dispone de una ruta para realizar la cobertura se cambia la ruta por defecto a la ruta de cobertura. Una vez el dron que estaba cargando pasa al estado `CHARGED` el dron que lo cubría retoma su ruta habitual.

8. Análisis experimental

Este capítulo presenta un análisis de la solución implementada. La sección 8.1 presenta las evaluaciones realizadas al sistema de navegación. La sección 8.2 presenta las evaluaciones al sistema de aterrizaje. La sección 8.3 presenta las evaluaciones realizadas al sistema de detección de intrusos. La sección 8.4 presenta las evaluaciones realizadas al comportamiento colectivo. Finalmente la sección 8.5 evalúa el comportamiento de un dron en escenario de vigilancia real.

8.1. Evaluación de la navegación por marcadores

Esta sección presenta las evaluaciones realizadas para el sistema de navegación con marcadores. La subsección 8.1.1 presenta una evaluación de la estimación de la posición en un escenario simulado. Por su parte la subsección 8.1.2 presenta una evaluación de la estimación de la posición en un vuelo real.

8.1.1. Estimación de distancia en escenario simulado

La primera prueba realizada a la utilización de marcadores como base de la navegación, tuvo como objetivo medir el error de calculo en la posición en un escenario simulado. Este escenario consistió en ubicar el dron en una posición fija, a un metro de altura y ubicar marcadores delante con distintas configuraciones. Para el cálculo del error en la posición estimada, se utilizó solo la se utilizó la distancia al marcador medida sobre el plano horizontal.

Las distancias elegidas para ubicar los marcadores fueron uno, dos y tres metros. Para cada una de estas distancias se tomaron medidas con uno y dos marcadores. En los casos donde se utilizó un solo marcador, este fue alineado con el eje longitudinal del dron. En los casos donde se utilizaron dos marcadores, uno de ellos fue ubicado en linea con el eje longitudinal del dron y el restante un metro a la izquierda. Los tres valores tenidos en cuenta para la evaluación fueron la distancia estimada al marcador, el error absoluto y el porcentaje de error respecto a la distancia real. Los resultados se presentan en el cuadro 1.

En los resultados obtenidos se pude observar que a medida que la distan-

Distancia	Nro. marcadores	Estimación	Error absoluto	Error %
1m	1	1.5	0.5	50 %
1m	2	1.3	0.3	30 %
2m	1	3.1	1.1	55 %
2m	2	3.0	1.0	50 %
3m	1	4.8	1.8	60 %
3m	2	4.6	1.6	53 %
Promedio	-	-	1.1	50 %

Cuadro 1: Resultados de las pruebas de estimación de distancia en escenario simulado.

cia real a los marcadores aumenta, también aumenta el error absoluto en el cálculo de la estimación. Por otra parte el cálculo de error porcentual muestra valores similares al promedio, lo que hace presumir que la curva de crecimiento del error con respecto a la distancia tiene tendencia lineal en el rango ensayado.

8.1.2. Estimación de posición en un escenario real

Para la segunda prueba utilizada para evaluar el comportamiento de la navegación por marcadores se creó un escenario base de ejemplo. El escenario consistió en un cuadrilátero de 10 por 10 metros. Los marcadores de navegación fueron ubicados en toda la superficie espaciados cada dos metros en toda la superficie del escenario. Por su parte los cuatro marcadores de aterrizaje fueron ubicados en los vértices de la plataforma de carga, la cual se ubicó en el centro del cuadrilátero. En la figura 35 se ilustra el escenario donde los cuadrados azules representan marcadores, mientras que el cuadrado amarillo la plataforma de carga.

A fin de mapear y reflejar el escenario en el archivo de configuración y poder realizar las pruebas, la ubicación de los elementos se midió en base a un sistema de coordenadas ortogonal. Se tomó como referencia uno de los lados del cuadrilátero como eje x (lado inferior), el lado perpendicular más a la izquierda se tomo como eje y, y el tercer eje z se ubicó perpendicular a los otros dos. El vértice inferior izquierdo del cuadrilátero se consideró entonces como el punto (0,0,0) y el sistema de medida utilizado fue el métrico. En la figura 35 se puede observar el eje x en color rojo y el eje y en verde en la parte

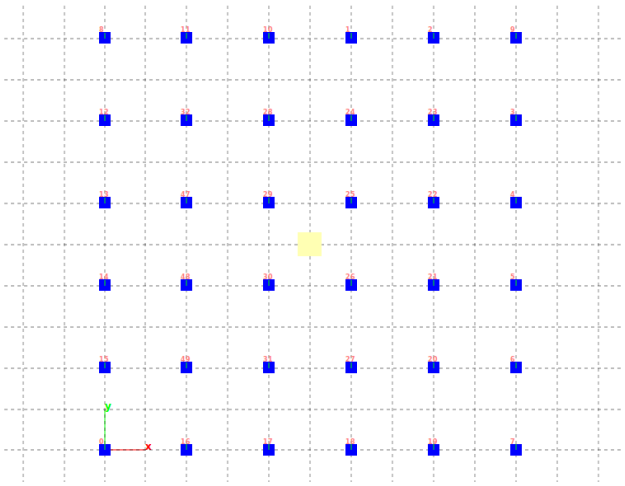


Figura 35: Esquema del escenario base.

inferior izquierda de la imagen. Además a modo de simplificar, se consideró solo la rotación sobre el eje z. Esta rotación se mide en grados sexagesimales y se consideró el eje y como cero grados, creciendo en sentido horario.

Tomando el escenario base se definió una ruta a seguir para llevar a cabo un vuelo real. La ruta a seguir constó de cuatro puntos con sus respectivas rotaciones:

1. posición = (2, 8, 0) y rotación = 180
2. posición = (2, 2, 0) y rotación = 45
3. posición = (8, 8, 0) y rotación = 180
4. posición = (8, 2, 0) y rotación = -45

Por su parte el dron se ubicó en la posición (-1, 8, 0) con rotación 90 cerca del primer punto de la ruta. En la figura 36 se ilustra la ruta a seguir por el dron y su posición inicial.

El objetivo de la prueba fue evaluar el desvio del dron con respecto a la trayectoria perfecta definida por la ruta de patrullaje. Para esto se filmó con cámara fija la trayectoria del dron recorriéndola. Luego se procesó la imagen generada para estabilizarla cuadro a cuadro y obtener una perspectiva superior. Finalmente se trazó la ruta recorrida por el dron junto a la ruta objetivo

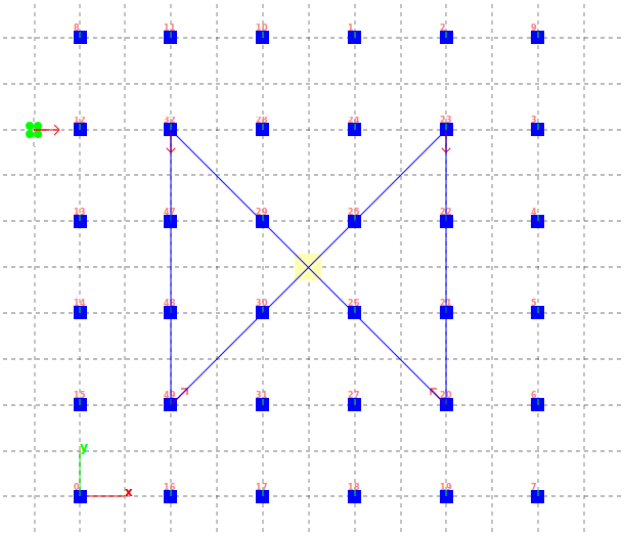


Figura 36: Ruta definida para las pruebas.

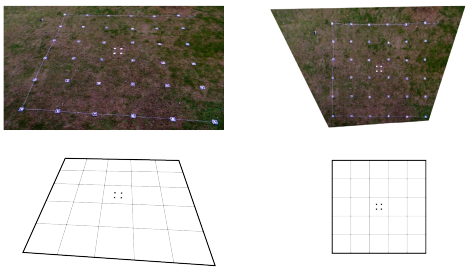
para poder comparar. La ruta recorrida fue trazada utilizando un algoritmo de seguimiento (*tracking*) sobre el dron en la imagen. En la figura 37 se puede observar la estabilización de imagen cuadro a cuadro, la trasformación de la perspectiva y el trazado de la trayectoria del drone.

La prueba fue repetida para múltiples valores de suavizado (*smooting*, representado en este contexto como α). Tal como explica la sección 6.1 el suavizado permite disminuir el error en el calculo de la posición, promediando su valor con cálculos anteriores. Sin embargo demasiado suavizado puede generar retraso (*delay*), aumentando el error en vez de reducirlo. A continuación se presentan los valores de α ensayados y sus respectivos resultados:

- $\alpha = 1$ cuyos resultados son ilustrados en la figura 38a
- $\alpha = 6$ cuyos resultados son ilustrados en la figura 38b
- $\alpha = 12$ cuyos resultados son ilustrados en la figura 38c
- $\alpha = 18$ cuyos resultados son ilustrados en la figura 38d

Para poder poder comparar el desfazaje de la ruta recorrida con la ruta ideal, se tomaron 5 puntos como referencia. Los puntos tomados fueron los cuatro puntos que definen el path (nombrados en orden como p_1, p_2, p_2 y p_4 respectivamente) y el punto central donde se cruzan las trayectorias del path

(a) Cuadro sin estabilización (b) Cuadro con estabilización



(c) Transformación de perspectiva.

(d) Trazado de trayectoria.

Figura 37: Procesamiento de imágenes y trazado de trayectoria.

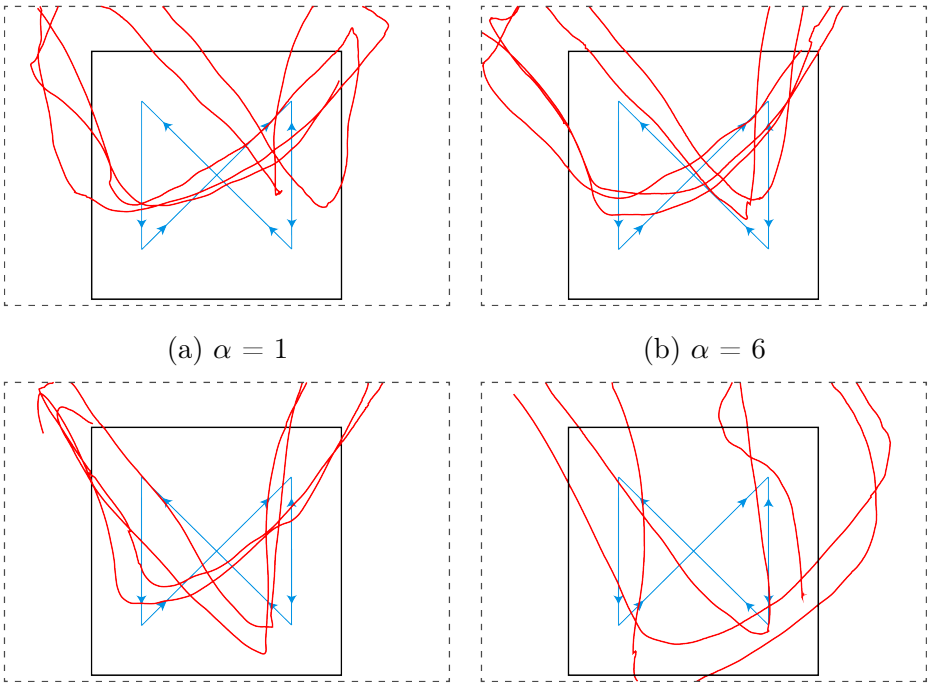
(nombrado como p_c). El error de desfazaje se calculó en base a la mínima distancia de cada punto de referencia al recorrido del drone. Las mediciones fueron tomadas en la imagen y convertidas a la escala real en metros. Se tomó el promedio de dos mediciones para cada punto de referencia, se calculó el promedio total y la suma de cuadrados. Los resultados se presentan en el cuadro 2.

α	p_1	p_2	p_3	p_4	p_c	$\bar{p}$	$\sum p^2$
1	3.5m	2.3m	0.9m	2.8m	1.7m	2.2m	77
6	1.3m	2.5m	0.5m	2.5m	0.6m	1.4m	34
12	1.1m	1.8m	1.3m	1.3m	1.6m	1.5m	34
18	0.6m	0.9m	0.6m	1m	3.8m	1.8m	93

Cuadro 2: Resultados de las pruebas de posición en escenario real.

Si consideramos el promedio total como medida de error, el menor error se produce con $\alpha = 6$. Le sigue $\alpha = 12$ con un error 9 % mayor, $\alpha = 18$ con un error 34 % mayor y por último $\alpha = 1$ con un error 60 % mayor.

Si consideramos la suma de cuadrados como medida de error, nos encontra-



(a) $\alpha = 1$ (b) $\alpha = 6$

(c) $\alpha = 12$ (d) $\alpha = 18$

Figura 38: Trayectoria del dron según el valor de α.

mos que el menor error se da con $\alpha = 12$, con $\alpha = 6$ se produce un error 2 % mayor, con $\alpha = 1$ se observa un error 130 % mayor y con $\alpha = 18$ un error 180 % mayor.

8.2. Evaluación de el aterrizaje autónomo

Para evaluar el aterrizaje sobre la plataforma se utilizó el mismo escenario base utilizado en la evaluación de la navegación con marcadores (subsección 8.1.2). Se definió como ubicación inicial para el dron el punto (1, 8, 0) con rotación 90. La tarea que debía realizar el dron consistió en cuatro etapas. Primero debía despegar, luego volar hacia la plataforma, alinearse sobre ella y aterrizar. El objetivo de esta prueba fue evaluar si el dron era capaz de aterrizar con éxito sobre la plataforma y en cuanto tiempo. La prueba consistió entonces en tomar los tiempos de múltiples aterrizajes. Los tiempos considerados fueron:

- t_{pos}*:* tiempo que el dron tarda en llegar a la plataforma.

- t_{alin}*:* tiempo de alineamiento sobre la plataforma.
- t_{tot}*:* tiempo total.

Los tiempos de aterrizaje y despegue no fueron considerados ya que son constantes y no dependían de ningún paramétro controlado.

En todos los ensayos realizados el dron fue capaz de aterrizar sobre la plataforma. Los tiempos se presentan en el cuadro 3:

Nro de prueba	t_{pos}	t_{alin}	t_{tot}	Atterrizó
Prueba 1	42 s	100 s	142 s	✓
Prueba 2	43 s	23 s	66 s	✓
Prueba 3	29 s	34 s	63 s	✓
Prueba 4	23 s	80 s	103 s	✓
Promedio	34 s	59 s	94 s	100 %

Cuadro 3: Resultados de las pruebas de aterrizaje.

8.3. Evaluación de la detección de intrusos

Esta sección presenta las evaluaciones realizadas sobre el sistema de detección de intrusos. La subsección 8.3.1 presenta una evaluación de la detección y seguimiento de intrusos en base a videos de intrusiones. Por su parte la subsección 8.3.2 presenta una evaluación del la estimación de la posición de un intruso en un escenario simulado.

8.3.1. Evaluación de detección y seguimiento

Para evaluar el funcionamiento de la detección y seguimiento de intrusos se filmó un conjunto de videos que simulan escenarios de intrusión. De los seis videos filmados, cinco de ellos fueron utilizados principalmente enfocados a evaluar las detecciones acertadas y uno fue utilizado para evaluar los falsos positivos. En la figura 39 se muestran capturas de los seis videos.

Todos los videos se filmaron con la cámara del drone. En los cinco primeros se ubicó el drone sobre un muro y una persona camino en frente de él. En

Figura 39: Captura de vídeos tomados para su evaluación de la detección de intrusos.

la figura 40 se puede observar los patrones de movimientos ejecutados: de adelante a atrás (a), atrás a adelante (b), izquierda a derecha al frente (c), derecha a izquierda (d), zigzag de atrás a adelante (e). El video restante se filmó simulando un vuelo de forma manual, dando una vuelta de trecientos sesenta grados para filmar todo el entorno.

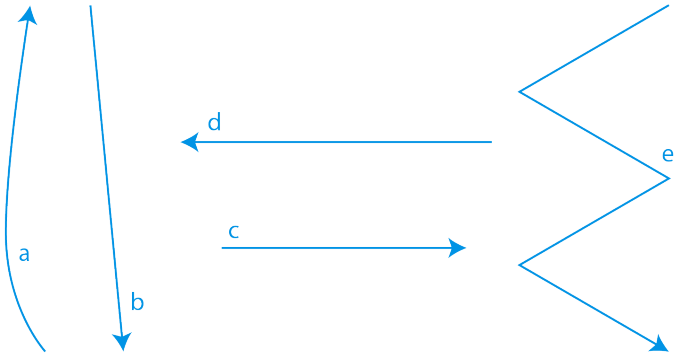


Figura 40: Desplazamientos utilizados en los videos de evaluación de la detección de intrusos.

Para las pruebas se tomaron en cuenta los tres parámetros de los que depende el algorítmo de detección y seguimiento: sensibilidad de detección, cantidad de detecciones e intervalo de seguimiento. La sensibilidad de detección depende de la configuración del algoritmo de detección. Mientras más sensibilidad de detección son más probables tanto las detecciones correctas como las erróneas. La cantidad de detecciones es la cantidad de cuadros en los cuales se utiliza el algoritmo de detección y se acumulan sus resultados. El intervalo de seguimiento es la cantidad de cuadros durante los cuales se mantiene el

algoritmo de seguimiento antes de volver a detectar. Para realizar las pruebas, se tomaron tres valores para cada parámetro, los cuales se presentan en el cuadro 4:

Parámetro	$Valor_1$	$Valor_2$	$Valor_3$
Sensibilidad	Sensible	Normal	Estricta
Cantidad detenciones	1	3	6
Intervalo tracking	10	30	60

Cuadro 4: Parámetros para las pruebas de detección y seguimiento.

En la primera fase de las pruebas se evaluaron los parámetros individualmente sobre el video de zig zag. Para cada parámetro se probaron sus tres valores posibles fijando los restantes dos con $Valor_2$. Las medidas utilizadas para evaluar el rendimiento fueron las conocidas *precision*, *recall* y *f-measure*. Los resultados se presentan en los cuadros 5, 6 y 7.

Sensibilidad	TP_{frames}	FP_{frames}	Precision	Recall	f
Sensible	156	66	70 %	96 %	81 %
Normal	150	30	83 %	93 %	88 %
Estricta	112	0	100 %	69 %	82 %

Cuadro 5: Prueba detección variando la sensibilidad de detección (total de frames = 162).

Cant. detec.	TP_{frames}	FP_{frames}	Precision	Recall	f
1	150	30	83 %	93 %	88 %
2	150	30	83 %	93 %	88 %
3	150	60	71 %	93 %	81 %

Cuadro 6: Prueba detección variando la cantidad de detecciones (total de frames = 162).

En la segunda fase de las pruebas se tomó como referencia los resultados obtenidos en la primera. En base a estos se escogieron dos configuraciones con las cuales realizar más pruebas. Para el parámetro sensibilidad de detección, se escogió el valor "Normal". El valor "Sensible" demostró ser poco preciso debido a una cantidad alta de falsos positivos. Por su parte la el valor "Estricta" demostró precisión perfecta pero muy poca exhaustividad.

Inter. tracking	TP_{frames}	FP_{frames}	Precision	Recall	f
10	130	20	87 %	80 %	83 %
30	150	30	83 %	93 %	88 %
60	154	34	82 %	95 %	88 %

Cuadro 7: Prueba detección variando el intervalo de tracking (total de frames = 162).

La cantidad de detecciónes no demostró una mejora al aumentar su valor con respecto al valor por defecto (una detección). Incluso la configuración 6 mostró peores resultados acumulando más falsos positivos. Por lo que para la segunda fase se escogió el valor 1. El intervalo de tracking demostró una mejora en sus dos valores más altos (30 y 60) con respecto al bajo (10). Por este motivo se eligió utilizar los dos valores más altos para las pruebas. Esto da como resultado dos configuraciones, presentadas en el cuadro 8.

Configuración	Sensibilidad	Cant. detec.	Inter. tracking
C_1	Normal	1	30
C_2	Normal	1	60

Cuadro 8: Configuraciones para la segunda etapa de las pruebas de detección.

Estas configuraciones fueron probadas para todos los videos filmados. Para presentar los resultados a cada video se le asigno una letra:

- **a**: de adelante a atrás.
- **b**: atrás a adelante.
- **c**: izquierda a derecha al frente.
- **d**: derecha a izquierda.
- **e**: zigzag de atrás a adelante.
- **f**: video del entrono.

Las medidas utilizadas para evaluar el rendimiento fueron las mismas que en la fase uno (*precision*, *recall* y *f-mesure*). En los cuadros 9 y 10 se presentan

Video	Tot_{frames}	TP_{frames}	FP_{frames}	Precision	Recall	f
a	102	85	30	74 %	83 %	78 %
b	102	98	38	72 %	96 %	82 %
c	52	20	31	39 %	38 %	39 %
d	62	48	0	100 %	77 %	87 %
e	162	150	30	83 %	93 %	88 %
f	422	0	387	0 %	0 %	-
total	902	401	516	44 %	44 %	44 %
total sin f	480	401	129	76 %	84 %	79 %

Cuadro 9: Resultados de la configuración C_1, en la segunda etapa de las pruebas de detección.

Video	Tot_{frames}	TP_{frames}	FP_{frames}	Precision	Recall	f
a	102	85	60	59 %	83 %	69 %
b	102	98	60	62 %	96 %	75 %
c	52	10	38	21 %	19 %	20 %
d	62	52	0	100 %	84 %	91 %
e	162	154	0	100 %	95 %	97 %
f	422	0	387	0 %	0 %	-
total	902	933	545	42 %	44 %	43 %
total sin f	480	399	158	72 %	83 %	77 %

Cuadro 10: Resultados de la configuración C_2, en la segunda etapa de las pruebas de detección.

los resultados de cada video para cada configuración, el total y el total sin considerar el video f.

Es de observar que el mal rendimiento de las pruebas sobre el video c, para ambas configuraciones, fue provocado por el algoritmo de seguimiento. En este video el algoritmo de detección fue capaz de detectar perfectamente a la persona frente al dron, pero fue el algoritmo de seguimiento que no fue capaz de seguir al objetivo. Esto puede haber ocurrido debido a que el objetivo estaba muy cerca de la cámara y se movía a una velocidad considerable: estos algoritmos se basan en buscar los desplazamientos de los píxeles objetivos dentro de un entorno local pero si desplazan fuera de la ventana de búsqueda la detección falla.

Los resultados obtenidos para ambas configuraciones fueron muy similares, siendo C_1 ligeramente superior . Además ambas pruebas arrojaron una gran cantidad de falsos positivos. Esto demuestra que el rendimiento de la detección y seguimiento esta determinado principalmente por el rendimiento del algoritmo de detección.

8.3.2. Evaluación del cálculo de distancia

Otro aspecto del sistema de detección de intrusos evaluado fue el cálculo de distancia. Como se explica en la sección 6.2, para poder llevar a cabo la persecución se calcula un estimación de la distancia al intruso con respecto al dron. Para evaluar su precisión se utilizaron dos escenarios simulados, en donde se ubicó un intruso delante del dron con el objetivo de medir el error de la distancia estimada con respecto a la distancia real.

En el primer escenario se ubicó el dron sobre un apoyo a 1.5 metros del piso. Para poder ajustar el parámetro de inclinación de la cámara respecto a la horizontal se ubicó un sujeto a 7 metros por delante del dron y se eligió un valor donde la distancia estimada y la real coincidieran. Luego de esto el sujeto se posicionó a distintas distancias y se tomaron los valores estimados en cada una de ellas. Los tres valores utilizados en la evaluación fueron la distancia estimada al sujeto, el error absoluto y el porcentaje de error respecto a la distancia real. En el cuadro 11 se presentan los resultados.

Distancia	Distancia estimada	Error absoluto	Error %
4m	3.5	0.5	13 %
8m	8	0.0	0 %
10m	10.4	0.4	4 %
15m	16	1	7 %
Promedio	-	0.5	8 %

Cuadro 11: Resultados de las pruebas del cálculo de distancia al intruso con altura fija.

En el segundo escenario también se ubicó el dron a 1.5 metros de altura, sin embargo este caso de forma manual simulando un vuelo real, de modo de dejar libre el sensor de altura del dron. La altura de horizonte utilizada en esta prueba fue la misma utilizada en el primer escenario. Como en este caso el valor la altura no era constante, debido a el error de medición del sensor

del dron, se tomó el promedio de 4 valores de distancia para cada medición. Estos valores fueron tomados con una separación de 2 segundos entre sí. Los tres valores utilizados en la evaluación fueron iguales al escenario anterior: distancia estimada al sujeto, el error absoluto y el porcentaje de error. En el cuadro 12 se presentan los resultados.

Distancia	Distancia estimada promedio	Error absoluto	Error %
5m	7.8	2.8	53 %
7m	10.5	3.5	35 %
10m	14.4	4.4	40 %
14m	17.4	3.4	17 %
Promedio	-	3.5	36 %

Cuadro 12: Resultados de las pruebas del cálculo de distancia al intruso con altura real.

En el primer escenario con altura ideal, se muestra un error relativamente bajo. Por su parte el segundo escenario muestra un error mucho mayor, aunque con valores de estimación consistentes entre mediciones. Como se ve en la figura 41, la medida se encuentra siempre cerca de dos unidades por encima del valor real. Como no se analizó la precisión de la estimación de altura del dron, se carece de información sobre el comportamiento de esta diferencia para otras alturas.

8.4. Evaluación del comportamiento colectivo

Debido a la simplicidad del sistema de comportamiento colectivo no se realizó una evaluación utilizando métricas. Para verificar su funcionamiento se creo y ejecutó un ambiente simulado capturado a través de un video. En este video se puede ver el comportamiento de una flotilla conformada por dos drones vigilando un escenario simple. La simulación fue construida manualmente sin utilizar modelado físico porque no se hizo incapié en la reproducción fidedigna de los sistemas físicos. Además se ejecutó independientemente a nivel del cuerpo de cada dron de la flotilla, permitiendo evaluar el comportamiento real de los componentes intercomunicación y cerebro que ejecutan los sistemas de comunicación y toma de decisiones.

La simulación de movimiento fue construida en base a la ruta que el dron debía seguir y una velocidad definida a la cual este debía moverse. Se definió

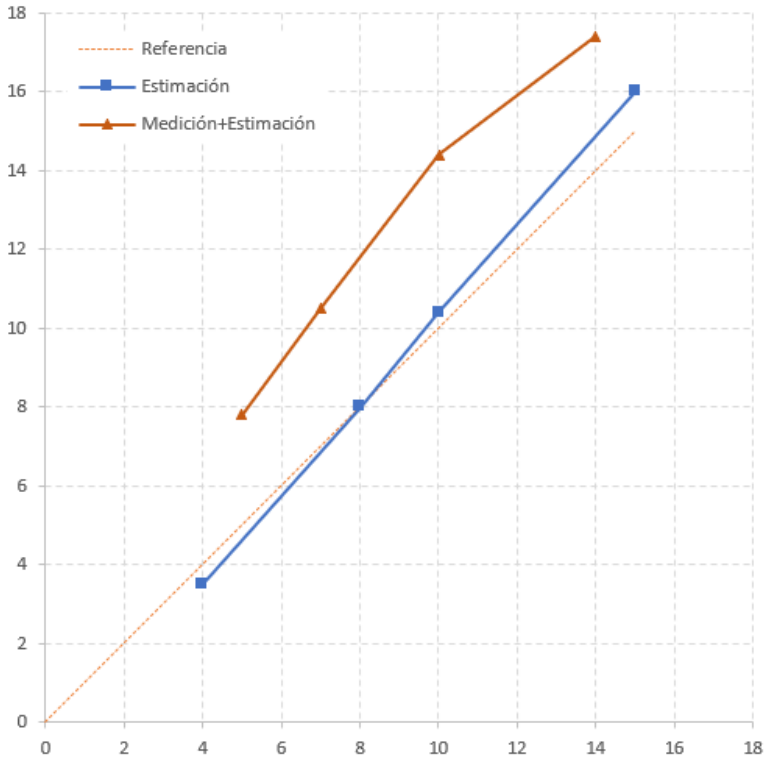


Figura 41: Estimación de distancias considerando la estimación de altura provista por el dron

además un intervalo de tiempo que permitió establecer la frecuencia de actualización de la posición y la rotación. Estos valores fueron calculados como la interpolación lineal entre el punto anterior y el próximo punto de la ruta a seguir y luego multiplicado por el factor de velocidad mencionado.

Para simular el ciclo de carga y descarga de la batería de cada dron se definieron dos parámetros de simulación: la duración de descarga de la batería y la duración de carga. Los drones comenzaron la simulación con carga completa en su batería. Mientras estaban en vuelo el porcentaje de carga de la batería se interpolaba linealmente entre 100 % y 0 % tomando en cuenta la duración de descarga y el tiempo de inicio del vuelo. Mientras el dron estaba cargando el porcentaje de la batería se interpolaba linealmente entre 0 % y 100 % esta vez tomando como referencia el tiempo de carga y el tiempo de inicio de la carga. Luego de cargada la batería y cuando en dron volvía a volar se comenzaba nuevamente con el ciclo de descarga.

Para simular las intrusiones se definió un probabilidad de que en un momento

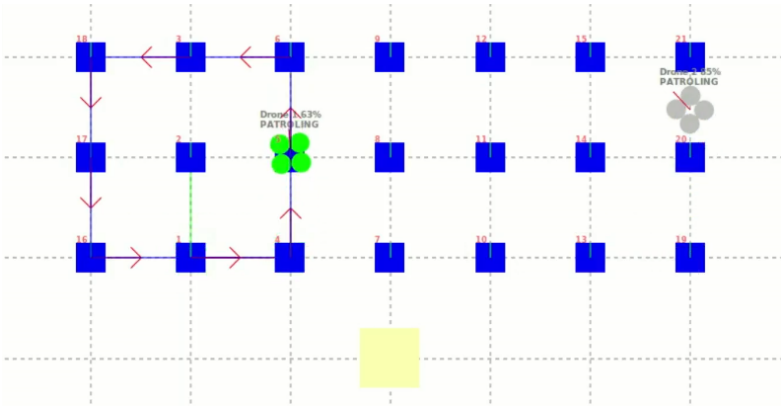


Figura 42: Simulación de comportamiento colectivo: ambos drones patrullando.

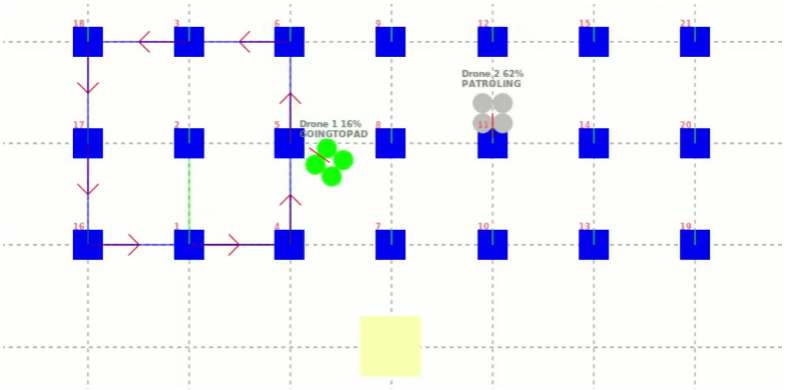


Figura 43: Simulación de comportamiento colectivo: batería baja.

dado, un intruso irrumpiera la zona y una duración. Además para simplificar la simulación, mientras el dron se encuentra persiguiendo a un intruso simplemente sigue su ruta de patrullaje y reporta al resto de la flotilla la posición del intruso como su misma posición.

El escenario elegido para la simulación fue un rectángulo de seis metros por dos metros. Los marcadores fueron ubicados en toda su superficie con 1 metro de separación. Si tomamos como referencia uno de los dos lados de mayor longitud del rectángulo como inferior, la plataforma fue ubicada en el eje central, un metro por debajo de este. La ruta del primer dron fue definida como el perímetro del cuadrado formado por los 8 marcadores más a la izquierda y la ruta del segundo dron fue definida de igual manera con en base a los 8 marcadores más a la derecha.

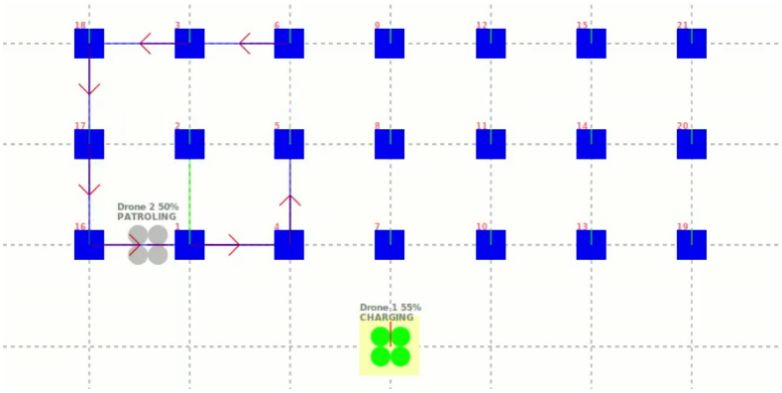


Figura 44: Simulación de comportamiento colectivo: cobertura de ruta de patrullaje.

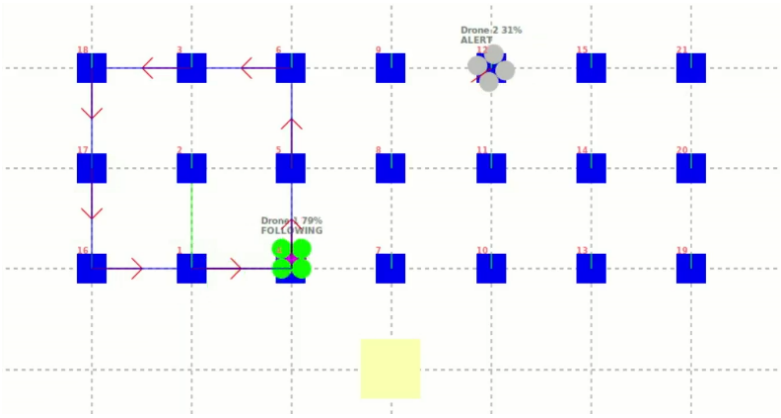


Figura 45: Simulación de comportamiento colectivo: intrusión.

Para que la simulación de cada dron de la flotilla fuera independiente una de ellas se ejecutó desde el sistema operativo anfitrión del equipo de trabajo y la segunda simulación se ejecutó en una máquina virtual dentro del mismo equipo utilizando un interfaz de red en modo puente. La simulación fue observada desde el mapa generado por uno de los drones. En la figura 42 se puede observar a ambos drones patrullando, en verde se puede ver el dron que genera el mapa, en azul con flechas rojas la ruta a seguir por este, en gris el dron restante, en azul los marcadores de aterrizaje y en amarillo la plataforma de carga.

Durante la simulación se pude observar el comportamiento colectivo (descrito en la subsección 7.2.2). Se observó como los drones fueron a cargar cuando su nivel de batería era bajo (menos de 20 % en la simulación)(figura 43).

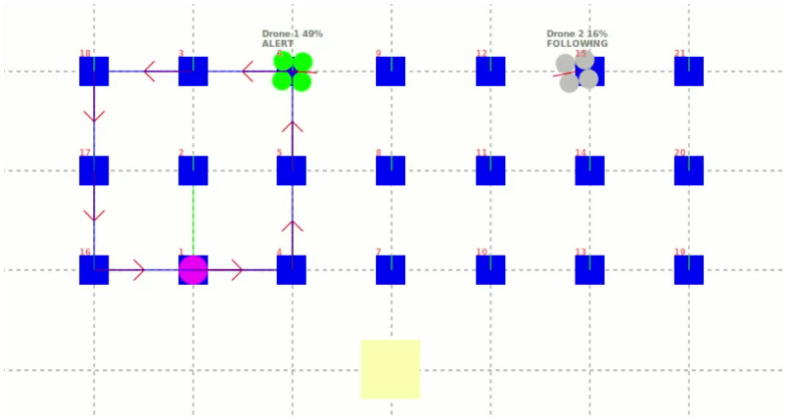


Figura 46: Simulación de comportamiento colectivo: batería crítica.

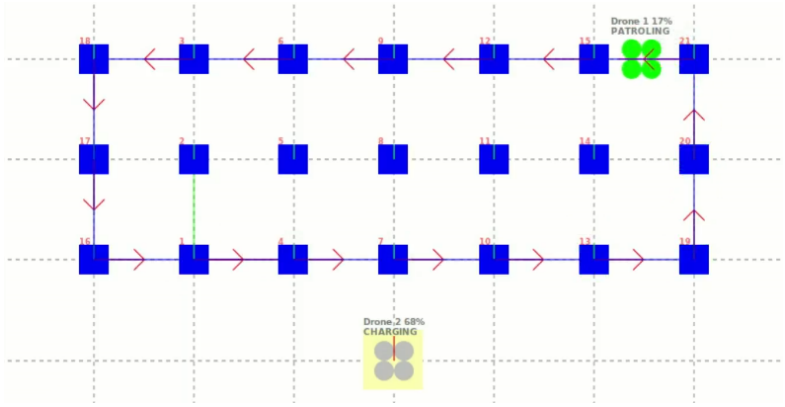


Figura 47: Simulación de comportamiento colectivo: plataforma ocupada.

También se observó como mientras uno de los drones recargaba su batería el restante cubría su ruta de patrullaje (figura 44). Se observó como mientras uno de los drones estaba persiguiendo a un intruso el restante estaba en estado de alerta mirando hacia su posición (figura 45). Se observó como a pesar de estar con batería baja (pero no con batería crítica, menos de 10 % para esta simulación), uno de los drones no fue a cargar por estar persiguiendo a un intruso (figura 46). También se observó como uno de los drones a pesar de tener batería baja, no fue a cargar debido a que la plataforma estaba ocupada por el dron restante (figura 46).

8.5. Evaluación en escenario real sobre un dron

En la última evaluación realizada se buscó simular un escenario real de vigilancia donde un dron debió realizar un flujo de vigilancia básico. El objetivo fue evaluar si era capaz de realizar las tareas de de patrullaje, seguimiento de intrusos y recarga en la plataforma de forma encadenada, realizando las transiciones correctamente.

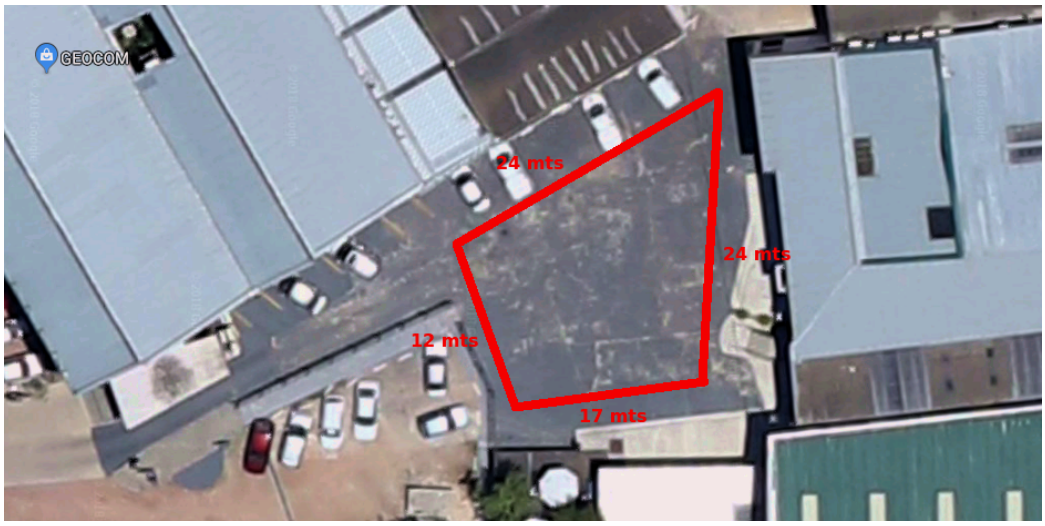


Figura 48: Escenario utilizado para la prueba en un escenario real.

El escenario de vuelo constó de un cuadrilátero escaleno de 12 x 17 x 24 x 24 metros, representado por el recuadro rojo en la figura 48. Los marcadores fueron ubicados toda en la superficie central del escenario y sobre ellos se definió una ruta de patrullaje. Por su parte los marcadores indicando la ubicación de la plataforma de carga fueron ubicados aproximadamente en el centro del cuadrilátero. En la figura 49 se pueden ver la distribución de los marcadores (cuadrados azules), la plataforma de carga (cuadrado amarillo) y la ruta definida (línea azul).

A efectos de reducir la duración de la prueba, no se trabajó con el ciclo real de carga y descarga de la batería del dron. Los eventos de batería baja y carga completa fueron disparados manualmente a través de un comando. Esto permitió decidir el momento adecuado para enviar al dron a aterrizar sobre la plataforma de carga, y que este despegase inmediatamente una vez haya aterrizado para volver a patrullar.

Para evitar que el dron detectara y persiguiera intrusos mientras se estaba

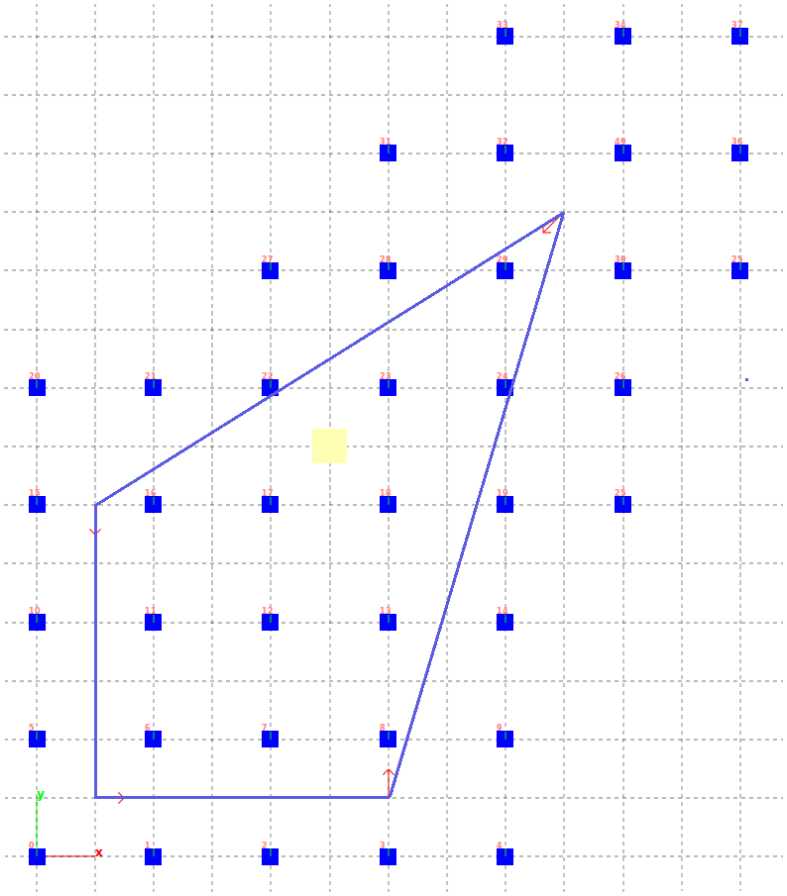


Figura 49: Marcadores y ruta de patrullaje en escenario real.

evaluando el patrullaje, esta funcionalidad se configuró como desactivada al inicio. Para poder activarla en el momento oportuno se utilizó otro comando. De esta forma se evitó que el dron persiguiera personas que se encontraban en el entorno del escenario y que interrumpiera el patrullaje ante la detección de falsos positivos.

La prueba comenzó con el dron aterrizado próximo al punto (0,0) en el sistema de coordenadas definido por los marcadores, representado por la intersección de los ejes x e y en la figura 49 (esquina inferior izquierda de la imagen). Una vez iniciada la ejecución el dron despegó y comenzó a seguir la ruta definida. Luego de que este volviera al punto inicial y comenzara de nuevo el recorrido el evento de batería baja fue activado. Esto provocó que el dron aterrizara sobre la plataforma, una vez aterrizado se ejecutó el comando indicando al dron que debía volver a su ruta. Una vez el dron volvió a patrullar, un individuo se ubicó en su campo de visión y la detección de

Figura 50: Patrullaje y persecución de intruso en escenario real.

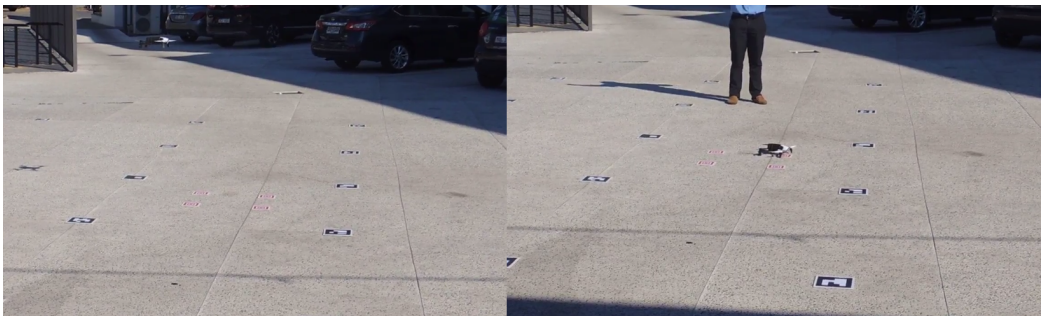


Figura 51: Alineamiento y aterrizaje en escenario real.

intrusos fue activada.

En la figura 50 se puede observar a la izquierda al dron patrullando la zona y a la derecha al dron persiguiendo un intruso. Por otra parte en la figura 51 se puede observar a la derecha al dron durante el alineamiento sobre la plataforma y a la izquierda al dron aterrizado sobre ella. Durante las pruebas se observó como dron cumplió con las tareas de patrullaje, recarga en la plataforma y seguimiento de intrusos, realizando las transiciones exitosamente.

9. Conclusiones y trabajo futuro

Este capítulo presenta las conclusiones y el potencial trabajo futuro. La sección 9.1 presenta las conclusiones de los autores y la sección 9.2 analiza mejoras en la solución planteada y nuevas líneas de investigación.

9.1. Conclusiones

Durante la realización de este proyecto quedó probado que es posible diseñar e implementar una solución basada en una flotilla de drones autónoma, particularmente aplicada al área de la seguridad aunque resulta una tarea compleja. Nuestro trabajo permitió una investigación inicial y el diseño de una prueba de concepto enfocada en proponer una solución al conjunto de problemas planteados: la navegación, detección de intrusos, carga autónoma y coordinación de una flotilla de drones en un escenario de vigilancia de un predio. La bibliografía presenta soluciones de bajo nivel como son la estabilización del cuadricóptero, los algoritmos de detección o la implementación de los protocolos de comunicación. Sin embargo, este proyecto hizo uso de tecnologías ya desarrolladas con el objetivo de demostrar la viabilidad de proyectos de mayor envergadura. Se optó por una solución más abarcativa y no por mayor funcionalidad y robustez.

En lo que refiere a la navegación por marcadores, creemos que el sistema diseñado presenta una solución viable. A pesar que en las pruebas de distancia en un escenario simulado presentaron un error en estimación de la posición nada despreciable, las pruebas de vuelo mostraron una capacidad apropiada para el seguimiento de un plan de vuelo. En cuanto a las desventajas de este sistema, cabe destacar que es necesario contar con una cantidad de marcadores importante, además de que se debe configurar de antemano su ubicación, lo que quita portabilidad a la solución. También es posible que un intruso ocluya uno o más marcadores de la zona de vuelo reduciendo la capacidad de navegación.

El sistema de detección de intrusos utilizando reconocimiento de imagen mostró clara dependencia al algoritmo empleado. Los principales problemas constatados durante las pruebas realizadas fueron las altas tasas falsos positivos y negativos en la detección. En una solución autónoma, mantener bajas ambas tasas resulta crucial. Con demasiados falsos positivos el sistema pierde credibilidad, mientras que con demasiados falsos negativos el sistema no resulta

confiable. Por su parte, los algoritmos de seguimiento de la detección y estimación de la distancia mostraron buen comportamiento. El seguimiento logró capturar apropiadamente el desplazamiento del intruso por el campo visual luego de ser detectado. En cuanto a la distancia estimada, aunque su valor era altamente variable debido a la inestabilidad de la salida de los sensores, el promedio se encontró en correlación con la distancia real al objetivo. Por último, cabe desatacar que la solución hace foco en intrusiones humanas y no se consideran otro tipo de amenazas como desastres naturales e intrusiones animales. Esto hace que no sea una solución completa y lo suficientemente robusta para su aplicación en un escenario real.

El sistema utilizado para permitir la carga autónoma por parte de los drones demostró efectividad. Basándose en un algoritmo simple, permitió un aterrizaje exitoso sobre la plataforma en el cien por ciento de los casos. Creemos que debido a la reducida autonomía de los drones, seria interesante hacer foco en reducir los tiempos de alineación de modo de reducir lo más posible el gasto de batería en el proceso. Otro aspecto a tener en cuenta en el uso de plataformas de carga en escenarios reales es su ubicación. Si se ubican en un lugar accesible, un intruso podría dañarlas o dañar sus marcadores impidiendo que los drones puedan cargar en ellas, o incluso dañar el dron mientras se encuentra cargando.

El sistema de colaboración entre drones fue implementado en base a dos componentes: la comunicación entre drones y la toma de decisiones. Su evaluación fue realizada mediante drones virtuales simulando una lógica similar a la implementada por el dron físico. A pesar de que no fue probada en profundidad, la comunicación entre drones demostró ser una solución efectiva para distribuir el estado de cada dron al resto de la flotilla. Por su parte, el mecanismo de toma de decisiones brindó solamente una aproximación al problema de la coordinación de la flotilla. Este permitió un comportamiento colectivo básico para los escenarios de cobertura y manejo de intrusiones. Por lo que este aspecto es el que tiene más potencial de desarrollo y que más debe ser trabajado si se desea que la flotilla de drones trabaje de forma realmente colaborativa en un escenario real.

Como se mencionó anteriormente el objetivo de este proyecto fue desarrollar una prueba de concepto. Creemos que la solución desarrollada aporta un enfoque valioso al tema y contribuye al validación experimental de este tipo de soluciones, ya sea en el escenario de la vigilancia como en cualquier otro escenario donde se utilicen vehículos aéreos no tripulados. Este trabajo partió de un escenario de vigilancia, que a través de un análisis detallado se

descompuso en un número de problemas a estudiar. Para cada uno de ellos se presentó al menos una solución viable que consideramos exitosa. Sostenemos que son representativos de los desafíos que una solución profesional se encontraría a la hora de embarcarse en una propuesta de la misma índole.

9.2. Trabajo futuro

Existen múltiples aspectos sobre los que se puede profundizar en la investigación, especialmente considerando una posible aplicación comercial fuera del ámbito académico. A continuación se detallan algunos de esos aspectos.

Uno de los desafíos a la hora de detectar intrusos fue la elección del algoritmo de detección. Si se contara con mayor capacidad de cómputo, se podrían utilizar algoritmos más potentes y obtener mejores resultados de detección. Otro aspecto a destacar es que el algoritmo de detección empleado utiliza la figura humana para la detección lo que presenta escenarios que deben ser considerados: intrusos en poses extrañas o que forman contornos disímiles al humano, intrusos no humanos, amenazas variadas como incendios o inundaciones, entre otros. Además sería importante poder diferenciar intrusos de otras personas que no lo son para permitir utilizar el sistema en un ambiente poblado.

El sistema de navegación de desarrollado incluye un mecanismo de navegación básico. Por lo que sería interesante profundizar en otros mecanismos de navegación que como lo son SLAM, odometría visual e inercial y GPS, entre otros. También la utilización de mecanismos de control más complejos para el vuelo, como por ejemplo un controlador PID permitiría generar mejor respuesta a las propiedades físicas y obtener control durante el vuelo. Por otra parte sería muy relevante incorporar un mecanismo de evasión de obstáculos totalmente autónomo y más sofisticado que permita la evasión de obstáculos estáticos y dinámicos.

En cuanto a la carga autónoma, cabe observar que la recarga de este tipo de baterías tiene velocidad decreciente. En nuestro prototipo cada dron de la flotilla decide que es hora recargar luego de que su nivel de batería está bajo un cierto umbral, para luego retornar a vuelo con carga completa. Este mecanismo no toma en cuenta los tiempos de carga, por lo que este aspecto puede ser investigado en búsqueda de minimizar el tiempo que cada dron está sobre la plataforma.

El sistema utilizado para la coordinación de la flotilla se caracteriza por su simplicidad, por lo que creemos que este es uno de los aspectos que más potencial de desarrollo posee. Lo primero interesante a añadir a la solución es un mecanismo de planificación de recorridos (*path planning*) que permita optimizar los recorridos a nivel global y en tiempo real. También sería interesante investigar otros enfoques utilizando paradigmas más complejos como el orientado a objetivos (*goal driven*), modelado distribuido del mundo (Lima y Custodio 2004), aprendizaje automático (*machine learning*), entre otros.

Otro de los desafíos que fue apenas investigado en este trabajo es la ejecución a bordo de los drones de todos los algoritmos de coordinación y control. El inconveniente se encuentra en la capacidad de cómputo de los drones disponibles en el mercado. Potencialmente la mejor opción para aumentar la capacidad de cómputo puede ser la incorporación de hardware externo, como por ejemplo utilizar placas Nvidia Jetson (2018).

Por último otro aspecto que resulta interesante es la integración de los drones con sistemas externos. Por un lado, sería relevante la capacidad del sistema de reportar alertas por correo electrónico o mensajes de texto por ejemplo. Por otro lado, resultaría interesante que los drones reciban información de sensores externos como cámaras y detectores de movimiento que agreguen información al modelo del mundo exterior.

A. Acceso al hardware

Este anexo describe el procedimiento utilizado para acceder al sistema operativo que corre en el dron.

Cuando se trabaja con vehículos aéreos autónomos es deseable que la solución a desarrollar total autonomía y que no utilicen nodos de procesamiento externos, por lo que todos los programas que controlen y provean inteligencia al dron deberían ejecutar a bordo. El Parrot Bebop Drone 2 corre una versión del kernel de linux. En su firmware versión 4.0.6 ejecuta la versión de kernel 3.4.11 con busybox 1.20.2. En el repositorio github se puede encontrar el software open source utilizado en el firmware (Parrot 2018b).

Para tener acceso de lectura y escritura sobre algunos directorios del sistema operativo del Parrot Bebop Drone 2 se puede utilizar el servicio FTP, que viene activado por defecto. Otra alternativa para acceder al sistema es utilizar telnet. Este servicio puede ser activado en el puerto 23 presionando cuatro veces el botón de encendido del drone (nicknack70 2018). Desde telnet se puede modificar la configuración, como por ejemplo acceder al sistema de archivos y ejecutar comandos.

Para ejecutar programas a bordo en el Bebop 2, estos deben compilarse externamente utilizando una compilación cruzada (*cross compiling*) desde x86 o x86-64 a ARMv7 (por ejemplo utilizando *GCC Cross Compile Toolchain*). Luego de esto de esto el programa debe ser transferido al dron, utilizando por ejemplo FTP. Finalmente el programa debe ser ejecutado vía telnet o añadiendo un script en el archivo /etc/init.d para que este se ejecute al inicio del sistema.

B. Detalles de la implementación

B.1. Librerías de terceros

A continuación se describen algunas de las librerías utilizadas durante el desarrollo de este proyecto.

- **OpenCV:** originalmente desarrollado por Intel, OpenCV es una librería enfocada en visión por computadora. Es liberada bajo licencia BSD. Es muy utilizada para desarrollo con fines tanto comerciales como no comerciales. Se encuentra desarrollada en C/C++, es multiplataforma y puede aprovechar la capacidad multinúcleo de los procesadores modernos, así como la aceleración por hardware mediante OpenCL, en caso de estar activada. Esta librería implementa miles de algoritmos publicados, algunos de los cuales fueron empleados en este proyecto. Son aprovechadas también las bibliotecas matemáticas que implementan vectores y matrices así como las que permiten desplegar imágenes y capturar los eventos del ratón y teclado.

```
1  syntax = "proto3";
2
3  message DroneState {
4      fixed32 ip = 1;
5      uint32 port = 2;
6      uint32 drone_id = 3;
7      int64 seq_num = 4;
8      string name = 5;
9
10     Point position = 6;
11     Point rotation = 7;
12 }
```

Listado 3: Extracto del archivo de especificación del mensaje `DroneState`.

- **Protocol buffers:** desarrollado por Google es un mecanismo de serialización de datos estructurados. Los datos son serializados en formato binario, fue diseñado para ser compacto, rápido y compatible hacia adelante como hacia atrás con otras versiones del sistema. Una de las

desventajas de este mecanismo es que sus mensajes no son auto descriptivos por lo que se requiere una especificación externa para comprenderlos. El formato de los mensajes es especificado mediante archivos de texto (con extensión .proto) y un generador de código produce el código fuente necesario para crear, serializar, deserializar y manejar estos mensajes en el lenguaje objetivo: C++, Java, C#, Objetive-C, Python, Go, Ruby, etc. En el listado 3 se puede ver un fragmento del archivo de especificación del mensaje de difusión de estado utilizado para coordinar la flotilla.

- **YAML:** presentado en 5.2, YAML es un formato de serialización de datos estructurados. El contenido de los archivos YAML consta de caracteres imprimibles UTF-8 y su estructura se basa en la indentación por espacios. Las principales estructuras son listas, diccionarios y datos escalares. En el listado 4 se puede observar un fragmento de un archivo YAML. Para la serialización y deserialización de los datos fue utilizada la librería `yaml-cpp` desarrollada por Jesse Beder.

```
1  Drone:
2    Name: drone-01
3    Id: 1
4    CameraTilt: 15
5    FOV: 80
6    FrameSize:
7      - 640
8      - 480
9    CameraMatrix:
10     rows: 3
```

Listado 4: Extracto del archivo de configuración YAML.

- **SDK3:** parrot, el desarrollador del Bebop 2, provee una librería para el control de varios de sus productos (Rolling Spider, Cargos, Mambo, Swing, Jumping Sumo, Jumping Sumo Evos, Bebop Drone, Bebop 2, Bebop 2 Power, Disco, Bluegrass, SkyController y SkyController 2). La librería corre sobre sistemas UNIX-like, iOS y Android. Sobre el final del desarrollo de esta tesis fue liberado además un simulador de drones: Sphinx. La ejecución del SDK3 inicia un hilo de ejecución (*thread*) que mantiene una comunicación con el Bebop 2. Esta es llevada a cabo mediante una conexión TCP inicial, que permite realizar el descubrimiento

del dron y la negociación inicia. Luego se utilizan mensajes UDP, a través de los cuales se intercambia información como ser comandos, datos de los sensores, cuadros de imagen, entre otros.

B.2. Parámetros de configuración

A continuación se detallan los parámetros configurables presentes en archivo `config.yaml`.

Drone		
Name	string	Nombre del dron
Id	int	Identificador del dron
FOV	double	Ángulo del campo de visión horizontal del drone
FrameSize	size	Tamaño del fotograma recibido desde el drone
CameraTilt	double	Inclinación del centro de la cámara respecto al plano horizontal cuando se encuentra en posición centrada.
CameraMatrix	matrix	Camera Matrix resultado de la calibración de la cámera. Ver subsección 3.2.1.
DistortionCoefficients	matrix	Coeficientes de distorsión resultado de la calibración de la cámera. Ver subsección 3.2.1
FrameSizeRatio	double	Cociente entre el ancho y el alto del fotograma. Valor calculado. No se pueden asignar directamente.
VerticalFOV	double	Ángulo del campo de visión vertical del dron. Valor calculado. No se pueden asignar directamente.

Communication		
BroadcastPort	int	Puerto a donde se envían los mensajes de broadcast
StateSendLapse	int	Tiempo en ms para el envío del estado del dron
StateExpireLapse	int	Tiempo en ms para considerar invalidado el último estado conocido de un dron

Debugging		
VisualDebugEnabled	bool	Indica si se encuentra habilitado el `VisualDebugger`
OutputHudVideoEnabled	bool	Indica si se encuentra habilitada la salida de video a archivo con el HUD sobreimpreso
RealTimeVideoOutputEnabled	bool	Indica si se repiten frames en la salida para alcanzar los FPS de reproducción del video. Si hay más FPS de los requeridos no se quitan frames
OutputRawVideoEnabled	bool	Indica si se encuentra habilitada la salida de video de la entrada de la cámara sin HUD
OutputPath	string	Directorio donde se almacenan las salidas de video y capturas
MapDebuggerScale	int	Escala (px por metro) del visor `MapDebugger`
MapDebuggerEnabled	bool	Indica si se encuentra habilitado el `MapDebugger`

Body (Parte 1)		
SleepDelay	int	Tiempo de espera en μs mínimo del thread del cuerpo en cada iteración del loop
Hal	HalType	HAL a ejecutar. Puede tomar los valores `Dummy`, `Vrep` o `Pb2`.
Start	bool	Indica si se inicia el thread del cuerpo
DummyCameraVideoSource	string	Ubicación del archivo a usar como entrada de video en el HAL `Dummy`. Si se indica -1 se utiliza el primer dispositivo de captura de video encontrado
CascadeDetector	string	Ubicación del modelo a utilizar para el `CascadeDetector`.
TrackingSmoothingSamples	int	Cantidad de muestras a tomar al calcular el *moving average* de posición y orientación del dron por el `MarkerTracker`.
WhiteBalance	WB	Modo de balance de blancos de `Pb2`. Puede tomar los valores `Auto`, `Tungsten`, `Daylight`, `Cloudy` y `CoolWhite`.
Saturation	float	Saturación de color de la imagen devuelta por el HAL `Pb2`. Toma valores en el rango $[-100, 100]$
Exposure	float	Exposición de la imagen devuelta por el HAL `Pb2`. Toma valores en el rango $[-100, 100]$

Body (Parte 2)		
TargetSlowdownRadius	float	Distancia en metros a la que empieza a disminuir velocidad el dron al acercarse a un objetivo
AlignmentAngleThreshold	float	Ángulo en grados máximo que se considera lo suficientemente cerca del ángulo objetivo
DisplacementMaxVelocity	float	Velocidad máxima de desplazamiento horizontal del dron. Toma valores en el rango $[0, 1]$.
YawMaxVelocity	float	Velocidad angular máxima de guiñada. Toma valores en el rango $[0, 1]$.
AltitudeSlowDownRadius	float	Diferencia de altura en metros a las que se empieza a disminuir la velocidad al acercarse a un objetivo
FollowerTargetDistance	float	Distancia objetivo en metros a mantener entre un intruso y el drone
YawAproximationAngle	float	Diferencia de ángulo en grados a los que empieza a disminuir la velocidad angular de guiñada el dron al acercarse a una orientación objetivo.
TargetReachedRadius	float	Distancia en metros a la que se considera que un objetivo se encuentra lo suficientemente cerca
StartAlignementDistance	float	Distancia en metros a la que se comienza a alinear el dron con la orientación objetivo

Brain		
Start	bool	Indica si se inicia el *thread* del cerebro
LowBatteryLevel	int	Nivel de batería que se considera bajo para requerir carga. Toma valores en el rango $[0, 100]$
CriticalBatteryLevel	int	Nivel de batería que se considera bajo para requerir carga inmediata. Toma valores en el rango $[0, 100]$

Land (Parte 1)		
MarkerDetectTolerance	float	Nivel de tolerancia al detectar marcadores de aterrizaje. Toma valores en el rango $[0, 1]$
PitchVelFactor	float	Factor por el que se multiplica el componente pitch de la velocidad durante la alineación previa al aterrizaje sobre la plataforma de carga.
RollVelFactor	float	Factor por el que se multiplica el componente roll de la velocidad durante la alineación previa al aterrizaje sobre la plataforma de carga.
GazVelFactor	float	Factor por el que se multiplica el componente gaz de la velocidad durante la alineación previa al aterrizaje sobre la plataforma de carga.
LandAltitude	float	Distancia en metros a la que se mantiene el dron durante la alineación previa al aterrizaje sobre la plataforma de carga.
PitchTolerance	float	Límite del componente pitch de velocidad debajo del cual el dron se considera alineado.
RollTolerance	float	Límite del componente roll de velocidad debajo del cual el dron se considera alineado.
GazAdjustment	float	Valor del componente gaz de la velocidad que el dron aplica luego de alinearse y previo a aterrizar.
PitchAdjustment	float	Valor del componente pitch de la velocidad que el dron aplica luego de alinearse y previo a aterrizar.
RollAdjustment	float	Valor del componente roll de la velocidad que el dron aplica luego de alinearse y previo a aterrizar.

Land (Parte 2)		
NoReferenceGazVel	float	Valor del componente gaz de la velocidad que el dron aplica para reubicar la plataforma.
MoveWithoutRefereceTime	int	Tiempo durante el cual el dron sigue el último comando de alineación cuando no tiene marcadores como referencia, antes de considerase perdido.
StabilizationTime	int	Tiempo que durante el cual el dron tiene que estar alienado para que la alineación se considere lo suficientemente estable como para aterrizar.

VirtualBody		
VirtualBodyEnabled	bool	Indica si el body virtual está habilitado
VirtualBodySpeed	float	Velocidad a la que se mueve el dron durante la simulación.
VirtualBatteryDuration	int	Tiempo de duración de la batería simulada.
VirtualBatteryChargeTime	int	Tiempo de carga de la batería simulada.

World			
Objects	id	int	Identificador del objeto dentro de los del mismo tipo
	type	ObjectType	Tipo del objeto.
	position	Vec3d	Posición del objeto en coordenadas *World*
	rotation	Vec3	Rotación (Euler) del objeto en coordenadas *World*

Paths			
droneId		id	Identificador del drone
path	position	Vec3d	Posición del objeto en coordenadas *World*
	rotation	Vec3	Rotación (Euler) del objeto en coordenadas *World*

Glosario

actitud Vector de rotaciones de un vehículo aéreo: *yaw*, *pitch* y *roll*. IV

ángulos de euler Vector de tres coordenadas angulares utilizadas para especificar la orientación de un sistema de ejes, respecto a otro. IV

alabeo Movimiento de rotación de un vehículo aéreo sobre su eje longitudinal. IV

ArUco Librería de reconocimiento visual para la generación y detección de marcadores. IV

autonomía Capacidad de vuelo de una aeronave. Puede estar dado por el tiempo que puede permanecer en vuelo o la distancia recorrida en este tiempo. IV

balance de blancos Ajuste realizado sobre una imagen con el objetivo de mejorar la reproducción de colores. IV

bounding box Mínima caja que contiene a una serie de puntos. IV

cabeceo Movimiento de rotación de un vehículo aéreo sobre su eje transversal. IV

cámara estenopeica Cámara sin lente constituida por una pequeña apertura o estenopo en una caja estanca. IV

cámara estereoscópica Cámara capaz de capturar imágenes en tres dimensiones. IV

cámara pinhole Ver **cámara estenopeica**. IV

cuadricóptero Multicóptero de cuatro rotores. IV

dron TVer **UAV**. IV

drone Término en inglés. Ver **UAV**. IV

empuje Se utiliza para referirse a la fuerza producida al desplazar una masa de aire en sentido opuesto al del movimiento en aeronaves. IV

exposición Cantidad de luz que deja pasar el lente de una cámara o presente un área de la imagen. IV

feature Término en inglés utilizado en el procesamiento de imágenes para referirse a una característica de la imagen que aporta información relevante. IV

frame Término en inglés para referirse a una imagen instantánea tomada de un video. IV

gaz Término en francés. Ver *throttle*. IV

guiñada Movimiento de rotación de un vehículo aéreo sobre su eje vertical. IV

HUD Sigla en inglés *head-up display* de la superposición de información sobre una imagen para permitir a los usuarios visualizar información sin cambiar su punto de vista. IV

LIDAR Acrónimo del inglés *Laser Detection and Ranging*, dispositivo que permite determinar la distancia a un objeto utilizando un pulso láser. IV

marcador Objeto que puede ser fácilmente reconocido por un algoritmo de procesamiento de imágenes. IV

matriz extrínseca En el modelo de cámara ***pinhole***, es la matriz que describe la transformación de puntos del espacio en coordenadas *world* a coordenadas de la cámara. IV

matriz intrínseca En el modelo de cámara ***pinhole***, es la matriz de los parámetros propios de la cámara: longitud focal, formato del sensor y punto principal.. IV

multicóptero Vehículo aéreo que utiliza múltiples rotores como propulsores. IV

odometría Técnica utilizada en robótica para estimar la posición de un vehículo mediante sensores. IV

pan Término en inglés para describir a la rotación de una cámara sobre su eje vertical. IV

PID Por sus siglas en inglés Proportional Integral Derivative. Mecanismo de control por retroalimentación utilizado comúnmente para la estabilización de drones y otros sistemas de control. IV

pitch Término en inglés. Ver **cabeceo**. IV

red adhoc Red descentralizada que no requiere de infraestructura pre existente donde los nodos se encargan del *routing* y *forwarding*. IV

red mesh Topología de red en la que los nodos se conectan entre ellos de forma no dinámica y no jerárquica para el *routing* y distribución de datos. IV

roll Término en inglés. Ver **alabeo**. IV

saturación Intensidad de un color. IV

sensor CMOS Referidos a sensores basados en la tecnología acuñada por sus siglas en inglés *Complementary Metal-Oxide-Semiconductor* técnica utilizada en la fabricación de circuitos integrados. IV

SLAM Por sus siglas en inglés *Simultaneous Localization And Mapping* refiere al problema de construir un mapa del entorno al tiempo de localizar un agente dentro de él. IV

sustentación Fuerza generada sobre un cuerpo que se desplaza sobre un fluido. En aeronáutica, es la principal fuerza responsable de mantener a las aeronaves en vuelo. IV

telemetría Técnica que permite a un observador lejano medir magnitudes físicas. Se utiliza para referirse a la obtención por parte del operador o de una estación de monitoreo de los datos de los sensores de la aeronave. IV

throttle Del inglés, aceleración. En multicópteros, se utiliza para referirse al control de velocidad de los rotores. En particular, puede referirse al control del movimiento vertical del dron. IV

tilt Término en ingles para describir a la rotación de una cámara sobre su eje transversal. IV

UAV Sigla en inglés de *unmanned aerial vehicle* (vehículo aéreo no tripulado) para designar aeronaves que operan sin tripulación. Es posible que el vehículo sea controlado remotamente u opere con autónoma. IV

yaw Término en inglés. Ver **guiñada**. IV

Referencias

[1] ArduPilot. *ArduPilot*. `http://ardupilot.org/`. 2018.

[2] G. Balamurugan, J. Valarmathi y V. Naidu. "Survey on UAV navigation in GPS denied environments". En: *Signal processing, communication, power and embedded system*. 2016.

[3] R. Barták, A. Hrasko y D. Obdrzalek. "A controller for autonomous landing of AR drone". En: *Control and Decision Conference*. 2014.

[4] batman-adv. *Open mesh: batman-adv*. `https://www.open-mesh.org/projects/batman-adv/wiki/Wiki`. 2018.

[5] H. Bay, Tinne Tuytelaars y L. Van Gool. "SURF: speeded up robust features". En: *Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[6] P. Blondel, A. Potelle, C. Pégard y R. Lozano. "Human detection in uncluttered environments: from ground to UAV view". En: *Control Automation Robotics Vision*. 2014.

[7] Boost. *Boost format library*. `https://www.boost.org/doc/libs/1_64_0/libs/format/index.html`. 2018.

[8] J. Borenstein. "The hoverBot - an electrically powered flying robot". University of Michigan. 1992.

[9] J. Bouguet. *Pyramidal implementation of the lucas kanade feature tracker description of the algorithm*. Intel Corporation, Microprocessor Research Labs. 2000.

[10] P. Bristeau, F. Callou, D. Vissière y N. Petit. "The navigation and control technology inside the ar. drone micro uav". En: *International federation of automatic control* 44.1 (2011), págs. 1477-1484.

[11] D. Cho, P. Tsiotras, G. Zhang y M. Holzinger. "Robust feature detection, acquisition and tracking for relative navigation in space with a known target". En: *AIAA guidance, navigation, and control conference*. 2013.

[12] Coppelia Robotics. *Coppelia robotics V-REP: Create. Compose. Simulate. Any Robot*. `http://www.coppeliarobotics.com/`. 2018.

[13] N. Dalal y B. Triggs. "Histograms of oriented gradients for human detection". En: *Computer vision and pattern recognition*. Washington, DC, USA: IEEE Computer Society, 2005.

[14] A. Davison. "Real-time simultaneous localisation and mapping with a single camera". En: *Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2003.

[15] J. Engel. "Autonomous camera-based navigation of a quadrocopter". Tesis de mtría. The Entrepreneurial University, 2011.

[16] O. Esrafilian y H. Taghirad. "Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM". En: *Robotics and Mechatronics*. 2016.

[17] M. Ester, H. Kriegel, J. Sander y X. Xu. "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise". En: *Knowledge discovery and data mining*. AAAI Press, 1996.

[18] D. Fleet e Y. Weiss. "Optical flow estimation". En: *Mathematical models in computer vision*. Boston, MA: Springer US, 2006, págs. 237-257.

[19] C. Forster, M. Pizzoli y D. Scaramuzza. "SVO: fast semi-direct monocular visual odometry". En: *Robotics and Automation*. 2014.

[20] FTI Consulting. "Public insecurity in Latin America". 2014.

[21] L. García, A. Dzul, R. Lozano y C. Pégard. "Combining stereo vision and inertial navigation system for a quad-rotor UAV". En: *Intelligent & robotic systems* 65.1-4 (2012), págs. 373-387.

[22] S. Garrido, R. Muñoz y F. Madrid M. Marín. "Automatic generation and detection of highly reliable fiducial markers under occlusion". En: *Pattern recognition* 47.6 (2014), págs. 2280-2292.

[23] Z. Gosiewski, J. Ciesluk y L. Ambroziak. "Vision-based obstacle avoidance for unmanned aerial vehicles". En: *Image and Signal Processing*. 2011.

[24] H. Grabner, M. Grabner y H. Bischof. "Real-time tracking via on-line boosting". En: *The british machine vision conference*. British Machine Vision Association, 2006.

[25] M. Grewal, V. Henderson y R. Miyasako. "Application of Kalman filtering to the calibration and alignment of inertial navigation systems". En: *Transactions on Automatic Control* 36.1 (1991), págs. 3-13.

[26] I. Haritaoglu. "A real time system for detection and tracking of people and recognizing their activities". En: *Computer Vision*. 1998.

[27] J. Henriques, R. Caseiro, P. Martins y J. Batista. "High-speed tracking with kernelized correlation filters". En: *Transactions on pattern analysis and machine intelligence* 37.3 (2015), págs. 583-596.

[28] B. Horn y B. Schunck. "Determining optical flow". En: *Artificial intelligence* 17.1-3 (1981), págs. 185-203.

[29] J. Hosang, R. Benenson y B. Schiele. "Learning non-maximum suppression". En: *Conference on computer vision and pattern recognition* abs/1705.02950 (2017), págs. 6469-6477.

[30] IET. "Autonomous vehicles: a thought leadership review of how the UK can achieve a fully autonomous future". 2014.

[31] J. Cruz. *Public insecurity in Central America and Mexico*. Inf. téc. 28. Vanderbilt University, 2009.

[32] A. Junaid, A. Konoiko, Y. Zweiri, M. Sahinkaya y L. Seneviratne. "Autonomous wireless self-charging for multi-rotor unmanned aerial vehicles". En: *Energies* 10.6 (2017), pág. 803.

[33] A. Al-Kaff, Q. Meng, D. Martín, A. de la Escalera y J. Armingol. "Monocular vision-based obstacle detection/avoidance for unmanned aerial vehicles". En: *Intelligent Vehicles Symposium*. 2016.

[34] Z. Kalal. "Tracking learning detection". British Library, EThOS. Tesis doct. University of Surrey, Guildford, UK, 2011.

[35] Z. Kalal, K. Mikolajczyk y J. Matas. "Forward-backward error: automatic detection of tracking failures". En: *International conference on pattern recognition*. IEEE Computer Society, 2010.

[36] J. Leonard y H. Durrant. "Simultaneous map building and localization for an autonomous mobile robot". En: *Intelligent robots and systems*. 1991.

[37] S. Li y D. Yeung. "Visual object tracking for unmanned aerial vehicles: a benchmark and new motion models". En: *Association for the advancement of artificial intelligence*. 2017.

[38] P. Lima y L. Custodio. "Artificial Intelligence and Systems Theory: Applied to Cooperative Robots". En: *Advanced Robotic Systems* 1.3 (2004), pág. 15.

[39] D. Lowe. "Distinctive image features from scale-invariant keypoints". En: *International journal of computer vision* 60.2 (2004), págs. 91-110.

[40] Y. Matsumoto, k. Ikeda, M. Inaba y H. Inoue. "Visual navigation using omnidirectional view sequence". En: *Intelligent robots and systems*. 1999.

[41] C. McCormick. *HOG person detector tutorial*. `http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/`. 2013.

[42] Microdrones. *Take security to a higher level.* `https://www.microdrones.com/en/industry-experts/security`. 2018.

[43] H. Moravec. "The Stanford cart and the CMU rover". En: *Proceedings of the IEEE* 71.7 (1983), págs. 872-884.

[44] A. Mourikis y S. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation". En: *Robotics and Automation*. 2007.

[45] M. Mueller y R. D'Andrea. "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers". En: *Robotics and Automation* 1109/ICRA.10 (2014), págs. 45-52.

[46] R. Mur-Artal, J. Montiel y J. Tardós. "ORB-SLAM: a versatile and accurate monocular SLAM System". En: *Transactions on robotics* 31.5 (2015), págs. 1147-1163.

[47] R. Mur-Artal, J. Montiel y J. Tardós. "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras". En: *Transactions on robotics* 33.5 (2017), págs. 1255-1262.

[48] R. Murphy. *Introduction to AI robotics*. Massachusetts Institute of Technology, 2000.

[49] nicknack70. "An unofficial Bebop drone hacking guide 1.7.3". 2018.

[50] C. Nicol, C. Macnab y A. Ramirez. "Robust neural network control of a quadrotor helicopter". En: *Electrical and computer engineering*. 2008.

[51] C. Nitschke. "Marker-based tracking with unmanned aerial vehicles". En: *Robotics and Biomimetics*. 2014.

[52] Nvidia. *Nvidia Jetson*. `http://www.nvidia.es/object/jetson-tk1-embedded-dev-kit-es.html`. 2018.

[53] OpenCV. *OpenCV: camera calibration and 3D reconstruction*. `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`. 2018.

[54] OpenCV. *OpenCV: optical flow*. `https://docs.opencv.org/3.3.1/db/d7f/tutorial_js_lucas_kanade.html`. 2018.

[55] PaparazziUAV. *Bebop - PaparazziUAV*. `https://wiki.paparazziuav.org/wiki/Bebop`. 2017.

[56] Parrot. *ARSDK protocols*. `https://developer.parrot.com/docs/bebop/ARSDK_Protocols.pdf`. 2015.

[57] Parrot. *Bebop drone*. `https://www.parrot.com/global/drones/parrot-bebop-2#technicals`. 2018.

[58] Parrot. *Open source software used in Parrot Bebop*. `https://github.com/Parrot-Developers/bebop-opensource`. 2018.

[59] RAE. *Diccionario de la lengua española*. `http://dle.rae.es/?w=diccionario`. 2018.

[60] J. Redmon, S. Divvala, R. Girshick y A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". En: *Computing research repository* abs/1506.02640.10 (2015), págs. 779-788.

[61] J. Redmon y A. Farhadi. "YOLO9000: Better, Faster, Stronger". En: *Computing research repository* abs/1612.08242 (2016), págs. 6517-6525.

[62] J. Redmon y A. Farhadi. "YOLOv3: an incremental improvement". En: *Computing research repository* abs/1804.02767 (2018).

[63] C. Reynolds. "Steering behaviors for autonomous characters". En: *Game developers conference*. 1999.

[64] E. Rublee, V. Rabaud, K. Konolige y G. Bradski. "ORB: an efficient alternative to SIFT or SURF". En: *International conference on computer vision*. 2011.

[65] J. Sagar. "Obstacle avoidance using monocular vision on micro aerial vehicles". En: *Aircraft engineering and aerospace technology: an international Journal*. 2014.

[66] O. Sahingoz. "Networking models in flying ad-hoc networks (FANETs): concepts and challenges". En: *Intelligent & robotic systems* 74.1 (2014), págs. 513-527.

[67] J. Sanchez, V. Arellano, M. Tognon, P. Campoy y A. Franchi. "Visual marker based multi-sensor fusion state estimation". En: *Automatic Control World Congress*. Toulouse, France, 2017. URL: `https://hal.laas.fr/hal-01501980`.

[68] S. Seitz, B. Curless, J. Diebel, D. Scharstein y R. Szeliski. "A comparison and evaluation of multi-view stereo reconstruction algorithms". En: *Computer vision and pattern recognition*. 2006.

[69] S. Shen, N. Michael y V. Kumar. "Obtaining liftoff indoors: autonomous navigation in confined indoor environments". En: *Robotics Automation Magazine* 20.4 (2013), págs. 40-48.

[70] J. Shi y Tomasi. "Good features to track". En: 1994.

[71] W. Shi, W. Wang, D. Li, Z. Wu y L. Mei. "Visual tracking with online multiple instance learning based on background classification". En: *Future information technology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[72] D. Shiffman. *The nature of code*. Daniel Shiffman, 2012.

[73] Skysense. *Drone charging infraestructure*. `http://www.skysense.co/`. 2018.

[74] G. Slabaugh. "Computing Euler angles from a rotation matrix". 1999.

[75] V. Sureshkumar y K. Cohen. "Autonomous control of a quadrotor UAV using fuzzy logic". En: *Journal of unmanned system technology* 2 (2014), pág. 2014.

[76] UAV Coach. *The top professional drones for serious commercial UAV pilots*. `https://uavcoach.com/professional-drones/`. 2018.

[77] J. Villasenor. "Drones and the future of domestic aviation". En: *Proceedings of the IEEE* 102.3 (2014), págs. 235-238.

[78] Vision-ary. *Boost the world: pedestrian detection*. `http://www.vision-ary.net/2015/03/boost-the-world-pedestrian/`. 2018.

[79] YAML. *YAML ain't markup language*. `http://yaml.org/`. 2018.

[80] L. Yang et al. "L. Yang and B. Xiao and Y. Zhou and Y. He and H. Zhang and J. Han". En: *Cyber technology in automation, control, and intelligent systems*. 2016.

[81] Y. Zeng, R. Zhang y T. Lim. "Wireless communications with unmanned aerial vehicles: opportunities and challenges". En: *Communications Magazine* 54.5 (2016), págs. 36-42.

[82] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". En: *Pattern Recognition*. 2004.