POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Miguel Fernández Cortizas

05 TRABAJO FIN DE GRADO

INDUSTRIALES

TRABAJO FIN DE GRADO

KIT DE DESARROLLO Y VALIDACIÓN DE ALGORITMOS DE CONTROL DE ACTITUD PARA CUADRICÓPTEROS.

TRABAJO FIN DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE GRADUADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

SEPTIEMBRE **2019**

Miguel Fernández Cortizas

DIRECTOR DEL TRABAJO FIN DE GRADO:
Pascual Campoy Cervera

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
MÁSTER UNIVERSITARIO EN AUTOMÁTICA Y ROBÓTICA

POLITÉCNICA

CONTROL Y PLANIFICACIÓN PARA DRONES DE CARRERAS AUTÓNOMOS.

Miguel Fernández Cortizas

Tutor académico:
D. Pascual Campoy Cervera

Madrid - España
2020

En memoria de mi padrino Juan,
seguiré trabajando hasta alcanzar las metas
que me hubiese gustado celebrar contigo.

Agradecimientos

Resumen ejecutivo

Introducción

Los pequeños multirrotores no tripulados (comúnmente llamados drones) son vehículos cuya popularidad en el mundo de la industria es cada vez mayor siendo utilizados para realizar tareas en campos muy diversos, como la inspección industrial, la industria cinematográfica o para su uso en operaciones de búsqueda y rescate. Aunque la inmensa mayoría del uso de estas aeronaves es mediante teleoperación, el auge de estas aeronaves han llevado a la comunidad robótica hacia el desarrollo de sistemas capaces de realizar tareas de forma autónoma.

Con el ánimo de impulsar el desarrollo de las tecnologías necesarias para conseguir mejorar la autonomía de estos drones, existen diversas competiciones internacionales como el IARC (International Aerial Robotics Competition), IMAV (International Micro Air Vehicle Competition) o MBZIRC (Mohamed Bin Zayed International Robotics Challenge), en las que el objetivo es conseguir que las aeronaves sean capaces de realizar pruebas complejas de forma autónoma.

Dentro de estás competiciones internacionales, existen las carreras de drones autónomos, como AlphaPilot , en las que el objetivo es que un dron sea capaz de recorrer un circuito de forma autónoma en el menor tiempo posible.

Alcance

Los objetivos de este trabajo consisten en diseñar un sistema autonómo, para que un cuadricóptero sea capaz de recorrer un circuito de carreras de forma satisfactoria y en diseñar un controlador capaz de permitir que un cuadricóptero vuele a altas velocidades a través de un circuito, así como generar las trayectorias que el aeronave debe seguir para recorrer el circuito de forma óptima. Adicionalmente, los algoritmos están diseñados para ser ejecutados en un ordenador a bordo, por lo que se deben optimizar para obtener el mayor rendimiento. El desarrollo y validación del trabajo se realizará el simulador y las reglas empleadas en la prueba virtual clasificatoria del AlphaPilot2019.

Solución realizada

- **Controlador:** Para conseguir que el cuadricóptero recorra el circuito a altas velocidades se han implementado dos controladores, uno linealizado en torno al punto de equilibrio, en los que la orientación del cuadricóptero varía un pequeño angulo respecto al estado de hover y otro para cuando esta variación es de un gran ańgulo.

- **Generación de trayectorias:** Se ha dividido la generación de trayectorias en dos partes: Una trayectoria que abarca todo el recorrido del circuito y otra más corta dentro de un horizonte temporal próximo a la posición del cuadricóptero. Ambas son trayectorias óptimas, generadas por una *spline*.
- **Arquitectura del sistema:** Los algoritmos de control y generación de trayectorias se han integrado dentro de una arquitectura modular empleando el *framework* de robótica ROS. Adicionalmente se han desarrollado módulos de percepción y estimación de estado empleando datos provistos por el simulador.

Experimentación y resultados.

Para realizar los experimentos se ha empleado el entorno de simulación Flightgoogles [1] el cual fue el empleado para las pruebas clasificatorias virtuales del Alphapilot 2019. Durante el transcurso del trabajo se han realizado experimentos para la evaluación de los controladores implementados, en los que se prueba el rendimiento de los mismos recorriendo trayectorias sencillas a distintas velocidades. Asimismo, se han realizado experimentos recorriendo el circuito de carreras entero, en los que se prueba el rendimiento del generador de trayectorias y de la arquitectura diseñada.

En estos experimentos se puede observar como el controlador para grandes ángulos es capaz de alcanzar velocidades más altas (hasta 11 m/s) con un menor error de seguimiento que el controlador linealizado. La arquitectura presentada es capaz de recorrer el circuito entero de forma satisfactoria en 24 segundos, tiempo que se encuentra dentro de los 3 mejores tiempos obtenidos por los equipos participantes en las clasificatorias del AlphaPilot2019 [1] [1]. Un vídeo mostrando el funcionamiento del sistema en simulación se puede encontrar en `https://vimeo.com/428975020/80051cd0d8`.

Conclusiones y trabajo futuro

Se ha conseguido desarrollar sistema modular capaz de recorrer un circuito de carreras de forma autónoma de forma satisfactoria. El controlador implementado permite realizar trayectorias a velocidades muy elevadas, aunque para conseguir disminuir el error de seguimiento sería posible emplear un controlador MPC que sea capaz de mitigar los cambios bruscos en las referencias. Finalmente, la decisión de dividir la generación de trayectorias en dos partes ha permitido computar ambas trayectorias a una alta frecuencia siendo capaces de adecuarse correctamente a los cambios en las estimaciones de las posiciones de las puertas en el circuito. En versiones futuras, sería conveniente tener en cuenta restricciones espaciales en la generación de las mismas.

La arquitectura propuesta permite su extensión con nuevos módulos de forma sencilla, por lo que se podría emplear para realizar otras tareas mediante la integración de los módulos necesarios.

[1] Estos resultados fueron presentados en el Workshop *Perception and Control for Fast and Agile Super-Vehicles* de la conferencia RSS20, integrando los módulos de percepción y estimación de estados desarrollados por Carlos Redondo Plaza

Palabras clave

UAV, cuadricóptero, modelado dinámico, teoría de control, generación de trayectorias, sistemas autónomos.

Códigos UNESCO

120326 SIMULACIÓN

330104 AERONAVES

330412 DISPOSITIVOS DE CONTROL

330118 ESTABILIDAD Y CONTROL

330417 SISTEMAS EN TIEMPO REAL

Índice general

Introducción

El auge de los pequeños multirrotores no tripulados (comúnmente llamados drones) en diversos ámbitos ha crecido de forma exponencial durante estos últimos años. Se puede observar la presencia de estos drones en diversos ámbitos como: el industrial, para realizar inspecciones en lugares de difícil acceso o en lugares peligrosos, el cinematográfico, para grabar escenas aéreas, o para dispositivos de búsqueda y rescate, en los que se emplean para poder barrer grandes áreas en búsqueda de personas desaparecidas, entre otros. Además, dentro del mundo del entretenimiento las carreras de drones comienzan a hacerse un hueco en países como los Estados Unidos, donde las carreras de drones son un deporte que se retransmite en algunos canales de televisión.

Actualmente, en la mayoría de estas competiciones, los drones están pilotados por un humano, aunque recientemente, empiezan a aparecer carreras de drones autónomos, en los que las aeronaves tienen que recorrer el circuito entero de forma autónoma sin la intervención de un humano. Algunos ejemplos de estas competiciones serían: la *IROS Autonomous Drone Race*, que se celebra anualmente dentro de la conferencia de robótica internacional IROS, o más recientemente AlphaPilot, una carrera de drones autónomos organizada por la *Drone Racing League* y la empresa *Lockheed Martin* con un gran premio de 1 millón de dólares para el ganador.

Figura 1.1: Cuadricóptero empleado en el AlphaPilot 2019, mirando hacia la primera puerta del circuito [2]

Actualmente, el rendimiento obtenido por los drones de carreras autónomos aún están lejos de alcanzar el de los pilotos humanos, es por esto que el afán por superar el nivel de los

pilotos humanos es una motivación para los grupos de investigación por todo el mundo. El desarrollo de drones autónomos es un campo de estudio exigente que involucra el desarrollo de la tecnología existente en diversos campos de la robótica como: la estimación, el control, la generación de trayectorias o la percepción, entre otros.

1.1. Motivación

El desarrollo de un sistema capaz de recorrer un circuito de forma autónoma sin interacción externa con cierta incertidumbre sobre el entorno, es un problema apasionante debido a las elevadas velocidades de vuelo y la limitada capacidad computacional, que exigen tener algoritmos de percepción y de estimación de estado más precisos y rápidos, así como algoritmos de planificación y de control más rápidos y ligeros, que permitan realizar cálculos a tiempo real.

Los avances obtenidos en diversos campos de la robótica durante el desarrollo de estos drones autónomos de carreras se pueden extrapolar para mejorar el rendimiento de los drones autónomos empleados para realizar distintas tareas con una mayor relevancia como pueden ser la aplicaciones industriales o como aquellas relacionadas con la búsqueda y rescate.

Los últimos avances en las plataformas de simulación enfocadas al desarrollo de sistemas de control de drones autónomos, como el simulador fotorrealista FlightGoggles desarrollado por Guerra et al [1] empleado durante las pruebas clasificatorias del Alhpa-Pilot 2019, permiten probar el rendimiento de un sistema autónomo de forma realista y segura.

1.2. Solución propuesta

El objetivo propuesto consiste en diseñar una arquitectura de un sistema modular capaz de controlar un cuadrirrotor a través de un circuito de carreras de forma autónoma.

Debido a la gran complejidad que presenta el desarrollo completo de todos los módulos que intervienen en un dron de carreras autónomo, se han simplificado el desarrollo de los módulos de estimación de estado y de percepción del circuito, empleando datos provistos por el simulador. Es por esto que el trabajo se haya centrado en el desarrollo de la arquitectura del sistema y del desarrollo de los módulos de control y generación de trayectorias capaces de generar y seguir trayectorias agresivas a lo largo de todo el circuito un circuito de carreras simulado a altas velocidades con incertidumbre sobre el recorrido en sí.

Para el algoritmo de control se han implementado dos controladores del estado del arte, un controlador para orientaciones de la aeronave con ángulos pequeños y otro para grandes ángulos, y se ha comparado el rendimiento de ambos en el seguimiento de diversas trayectorias.

En cuanto a la generación de trayectorias, se ha optado por generar trayectorias polinómicas óptimas de tipo *spline*. Para poder mejorar la capacidad de reacción del aeronave ante cambios en el circuito se ha dividido la generación de trayectorias en dos partes: la generación de una trayectoria completa a través de todo el circuito (largo plazo) y la generación de una trajectoria corta entorno a un horizonte temporal próximo a la posición del aeronave (corto plazo).

Para desarrollar los algoritmos y comparar el rendimiento del sistema realizado, se ha empleado el simulador FlightGoogles, empleado para las pruebas clasificatorias del AlphaPilot 2019 como entorno de pruebas.

1.3. Objetivos

Para poder llevar a cabo el desarrollo del trabajo de forma adecuada es conveniente desgranar los objetivos principales en tareas de alcance más reducido:

- **Modulo de control**
  - Modelado dinámico del comportamiento de un cuadricóptero.
  - Estudio del estado del arte acerca de los algoritmos de control más relevantes en las carreras de drones.
  - Desarrollo teórico de los algoritmos de control para ángulos pequeños y para ángulos grandes.
  - Implementación de ambos algoritmos dentro del módulo de control.
  - Ajuste de los parámetros de los controladores.
  - Evaluación del rendimiento de ambos controladores en un entorno simulado.
- **Generación de trayectorias**
  - Estudio del estado del arte acerca de los métodos de generación de trayectorias más empleados.
  - Desarrollo teórico de los algoritmos de generación de trayectorias que se van a emplear.
  - División del módulo generador de trayectorias en corto y largo plazo.
  - Implementación de los algoritmos de generación de trayectorias empleados.
  - Evaluación del rendimiento de los generadores de trayectorias mediante el recorrido del circuito de carreras en simulación.
- **Arquitectura del sistema**
  - Estudio de las arquitecturas empleadas por los ganadores de las diversas competiciones de drones autónomos.
  - Descomposición del sistema en los distintos módulos separados que compondrán la arquitectura final.
  - Implementación de los módulos de estimación y percepción partiendo de las mediciones provistas por el simulador.
  - Evaluación del funcionamiento completo de la arquitectura mediante el recorrido del circuito simulado.

Estado del arte

Para contextualizar el trabajo desarrollado dentro del campo de las carreras de drones autónomos es necesario conocer los avances obtenidos por la comunidad robótica. Debido a que los contenidos del trabajo se han enfocado, principalmente, en torno al controlador y a la generación de trayectorias se ha profundizado en el estado del arte de estos campos enfocados a los drones de carreras. Adicionalmente, debido al objetivo del trabajo de generar un sistema coordinado, también se ha revisado el trabajo realizado por los finalistas de las carreras de drones autónomos más importantes.

2.1. Control y generación de trayectorias

En la primera década de los 2000, la mayoría del trabajo realizado con multirrotores empleaba controladores linealizados en torno al punto de equilibrio (*hover*), los cuales, unicamente garantizan la estabilidad de la aeronave para pequeños ángulos de *pitch* y *roll* [3]. En cuanto a las trayectorias generadas, la mayoría de ellas son trayectorias polinómicas del tipo *spline*, generadas interpolando una función entorno a los puntos de paso deseados [4][5].

En 2008 V. Raffo et al. [6] emplearon una estructura de control basada en un controlador predictivo basado en el modelo (MPC) que se encargaba de seguir la trayectoria y un controlador $\mathcal{H}_\infty$ que controlaba la rotación de la aeronave. Con esta estructura son capaces de seguir trayectorias sencillas de forma robusta ante perturbaciones.

En 2010, Guillula et al. [7] diseñaron un controlador capaz de realizar maniobras acrobáticas, como una voltereta hacia atrás, con un cuadricóptero de forma segura. Para ello emplearon un *framework* para el diseño de regiones de cambio seguras, en las que cada región presenta un modelo dinámico distinto. Sin embargo, estas maniobras se generan de forma discontinua, necesitando analizar cada parte de la trayectoria de forma independiente y generar situaciones de cambio entre estos modos de forma segura.

En 2011, Mellinger et al. [8] presentan un controlador para cuadricópteros que permite realizar maniobras agresivas en un espacio tridimensional de forma continua. Este controlador no está linealizado en torno a ningún punto de funcionamiento, por lo que permite seguir trayectorias agresivas con un bajo error de seguimiento aunque el aeronave tenga ángulos grandes de *roll* y *pitch*. Este es uno de los controladores más usados actualmente debido al rendimiento que consigue con un algoritmo sencillo y con un bajo coste computacional.

En 2012, Mallikarjunan et al. [9] diseñaron un controlador de actitud adaptativo, aplicando control $\mathcal{L}_1$, capaz de seguir trayectorias de forma precisa y robusta, con presencia de incertidumbres en el modelo de la aeronave y de las perturbaciones del entorno.

En 2016, Kamel et al. [10] comparan el rendimiento de dos MPCs, uno lineal y uno no

lineal, en el seguimiento de trayectorias agresivas con un cuadricóptero. En estos experimentos observaron que, aunque ambos controladores eran capaces de seguir las trayectorias de forma satisfactoria, el controlador no lineal, conseguía un rendimiento ligeramente superior.

En 2017, Faessler et al. [11] emplearon control LQR considerando tanto la dinámica del cuadricóptero, como la dinámica aislada de cada rotor. Además, consideran los limites de los rotores para priorizar la saturación de aquellas entradas que son relevantes para la estabilización del cuadricóptero. Asimismo, en 2018 [12], refinaron el controllador de Mellinger et al. considerando el arrastre (*drag*) de los rotores dentro del modelo dinámico del cuadricóptero, en lugar de considerarlo como una perturbación externa desconocida, consiguiendo una ligera mejora en el seguimiento de trayectorias a alta velocidad.

En 2018, Falanga et al. [13] presentan un controlador MPC consciente de la percepción, el cual unifica el control y la planificación para satisfacer objetivos de acción y percepción de forma simultánea. Las trayectorias generadas por el MPC deben tener en cuenta ambos objetivos, para conseguir realizar maniobras complicadas mientras maximizan la visibilidad de puntos de interés por la aeronave.

2.2. Carreras de drones autónomos

En 2016 se inauguró la primera competición de drones de carreras autónomos (ADR) en el marco de conferencia internacional de robótica IROS (*Intelligent Robots and Systems*). En este año Jung et al.[14] fueron capaces de completar todo el circuito obteniendo el primer puesto en la clasificación. Para ello emplearon un *framework* basado en IBVS (Image Based Visual Servoing) con el que primero estimaban las posiciones de los obstáculos empleando cámaras estéreo, para posteriormente cerrar el bucle de control emplando la posicion relativa del centro de las puertas en la imagen.

Los ganadores del ADR 2017 Moon et al. [15] emplearon un algoritmo de SLAM monocular con el cual estimaban la posición de las puertas, para posteriormente acercarse al centro de la siguiente puerta empleando un regulador PID para controlar la altura y la orientación del aeronave. La velocidad media de vuelo del ganador se encontraba alrededor de los 0.7 m/s.

En 2018, Kaufmann et al. [16] ganaron la competición IROS 2018 Autonomous Drone Race empleando técnicas de aprendizaje automático para la detección de las puertas del circuito. Para generar las trayectorias de control fijaban dos puntos de paso, uno en frente de la siguiente puerta y otro después de la siguiente puerta. Para seguir estas trayectorias emplearon un controlador MPC con el que consiguieron adquirir velocidades de vuelo próximas a los 2 m/s.

En el 2019 se celebró la primera edición de la competición AlphaPilot 2019, en la que Foehn et al. [2] consiguieron un segundo puesto empleando una arquitectura novedosa, empleando estrategias de aprendizaje automático junto con un filtrado no lineal de las mediciones de los sensores y la generacion trayectorias óptimas en tiempo. Emplearon un controlador PD con el que alcanzaron velocidades máximas de unos 8 m/s, lo que significa un aumento considerable de la velocidad de vuelo con respecto a las velocidades a las que se movían los drones en la ADR.

Además en 2019, se celebra la primera carrera de drones autónomos puramente simulada, Game of Drones en el marco de la conferencia NeurIPS 2019. Los ganadores

de la carrera Shin et al. [17] emplearon técnicas de aprendizaje por refuerzo (*reinforcement learning*) para diseñar el controlador del aeronave, con el cual consiguieron alcanzar velocidades máximas de hasta 16 m/s en un entorno simulado.

Figura 2.1: Entorno de simulación empleado en la competición Game of Drones dentro de la conferencia NeurIPS 2019

Modelado de un cuadricóptero

Un cuadricóptero es un robot aéreo con 6 grados de libertad (3 rotacionales y 3 traslacionales) y 4 motores, al tener menos motores que el número de grados de libertad, se dice que es un sistema subactuado [18][8][19].

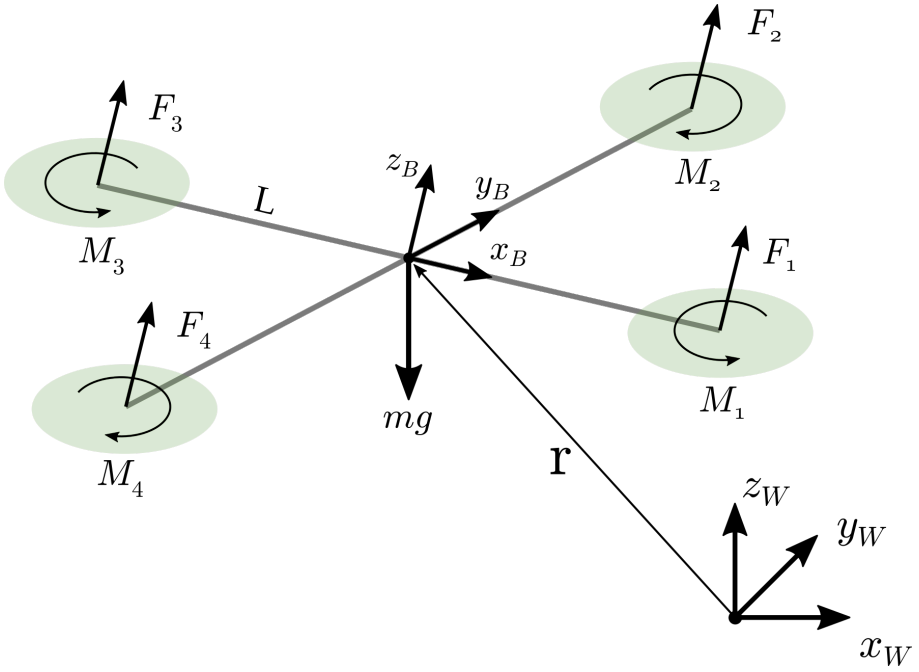

Figura 3.1: Esquema de fuerzas y momentos que actúan sobre un cuadricóptero y sus sistemas de referencia asociados.

Como puede observar en la figura 3.1, se ha empleado el subíndice W para hacer referencia al sistema de referencia del mundo, así como el subíndice B para referirse al sistema asociado al cuerpo del cuadricóptero. El *frame* B posee su origen O_B en el centro de masas de la aeronave, con el eje x_B coincidente con la dirección de avance preferente de la aeronave.

Para modelar las rotaciones del *frame* B con respecto a W se emplearán los ángulos de Euler Z-X-Y, es decir, la matriz de rotación R para transformar coordenadas desde B a W consiste en la composición de las siguientes rotaciones:

$$R = R_{\mathbf{z},\psi} R_{\mathbf{x},\phi} R_{\mathbf{y},\theta} \tag{3.1}$$

siendo $R_{\mathbf{i},\alpha}$ una rotación de un ángulo α respecto al eje i. Al desarrollar la expresión 3.1

se obtiene la matriz R resultante

$$R = \begin{bmatrix} c_\psi c_\theta - s_\phi s_\psi s_\theta & -c_\phi s_\psi & c_\psi s_\theta + c_\theta s_\phi s_\psi \\ c_\theta s_\psi + c_\psi s_\phi s_\theta & c_\phi c_\psi & s_\psi s_\theta - c_\theta s_\phi c_\psi \\ -c_\phi s_\theta & s_\phi & c_\phi c_\theta \end{bmatrix} \tag{3.2}$$

donde c_θ y s_θ denotan $cos(\theta)$ y $sen(\theta)$ respectivamente.

En el sistema de referencia B las componentes del vector velocidad angular Ω están definidas por p, q y r de la forma:

$$\Omega = p\mathbf{x_B} + q\mathbf{y_B} + r\mathbf{z_B} \tag{3.3}$$

Estas componentes están relacionadas con las derivadas de los ángulos de Euler de acuerdo a

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & -c_\phi s_\theta \\ 0 & 1 & s_\phi \\ s_\theta & 0 & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.4}$$

3.1. Análisis dinámico

3.1.1. Ecuaciones de movimiento traslacional

Como se puede observar en la figura 3.1, las fuerzas que actúan sobre el cuadricóptero son: la gravedad, en la dirección $-\mathbf{z_W}$ y la fuerza de cada uno de los motores, en la dirección $\mathbf{z_B}$. Para hallar las ecuaciones que rigen la dinámica del centro de masas del sistema C se aplican las ecuaciones de Newton sobre él. Siendo $\mathbf{r}$ el vector de posición del centro de masas C con respecto al origen de W obtenemos:

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{3.5}$$

Si denominamos $F = \sum_{i=1}^{4} F_i$, al expandir la ecuación anterior con la definición de R en 3.2 obtenemos las ecuaciones que describen el movimiento traslacional del centro de masas del cuadricóptero:

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} s_\theta c_\psi + s_\phi c_\theta s_\psi \\ s_\theta s_\psi - s_\phi c_\theta c_\psi \\ c_\phi c_\theta \end{bmatrix} F \tag{3.6}$$

3.1.2. Ecuaciones de movimiento rotacional

Como se puede observar en la expresión anterior, el movimiento del cuadricóptero depende de la rotación R, por lo que es necesario modelar el movimiento rotacional del mismo. Se define el momento angular H como:

$$H = \mathbf{I}\Omega \tag{3.7}$$

donde $\mathbf{I} \in \mathbb{R}^{3\times3}$ representa el tensor de inercia del cuadricóptero en el sistema B, y $\Omega = [p, q, r]$ representa el vector de velocidad angular de B respecto W.

Si denotamos $M_c = [\tau_x, \tau_y, \tau_z]^t$ como el momento total del cuadricóptero en el sistema B

$$\begin{aligned} M_c &= \frac{d}{dt}H \\ &= \mathbf{I}\dot{\Omega} + \Omega \times \mathbf{I}\Omega \end{aligned} \tag{3.8}$$

Se considera que el cuadricóptero presenta una distribución de masa simétrica, por lo que el tensor de inercia $\mathbf{I}$ es un tensor diagonal de la forma:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.9}$$

siendo I_{xx}, I_{yy}, I_{zz} los momentos principales del cuadrícoptero con respecto a los ejes x_B, y_B, z_B respectivamente.

Desarrollando la expresión 3.8 y reorganizando sus términos:

$$\mathbf{I}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \mathbf{I}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3.10}$$

A partir del diagrama de la figura 3.1 se puede calcular los valores de τ_x, τ_y y τ_z a partir de las fuerzas y momentos ejercidos por los motores.

$$\begin{aligned} \tau_x &= L(F_2 - F_4) \\ \tau_y &= L(F_3 - F_1) \\ \tau_z &= M_1 - M_2 + M_3 - M_4 \end{aligned} \tag{3.11}$$

Finalmente se unen las expresiones 3.10 y 3.11 para obtener la ecuación de movimiento rotacional del cuadricóptero.

$$\mathbf{I}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \mathbf{I}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3.12}$$

3.2. Planicie diferencial (*differential flatness*)

A continuación, se mostrará que la dinámica de un cuadricóptero es plana diferencialmente [19][18][8], esto implica que las entradas del sistema se pueden escribir en función de unas "salidas planas" escogidas cuidadosamente y de sus correspondientes derivadas. Esto implica que cualquier trayectoria suave generada en el espacio de las "salidas planas" puede ser seguida por el cuadricóptero de forma satisfactoria (dentro de las limitaciones físicas del sistema), lo que facilitará la generación de trayectorias posteriormente .

Las salidas planas que se han escogido vienen dadas por :

$$\sigma = [x, y, z, \psi]^t \tag{3.13}$$

siendo $\mathbf{r} = [x, y, z]^t$ las coordenadas del centro de masas del aeronave en el mundo y ψ el ángulo en yaw correspondiente. A continuación se demostrará como se puede expresar el estado del aeronave en función de estas salidas.

Posición, velocidad y aceleración

Es trivial observar como la posición, velocidad y aceleración del aeronave son funciones de las salidas escogidas:

$$[x, y, z]^t = [\sigma_1, \sigma_2, \sigma_3]^t \tag{3.14}$$
$$[\dot{x}, \dot{y}, \dot{z}]^t = [\dot{\sigma}_1, \dot{\sigma}_2, \dot{\sigma}_3]^t \tag{3.15}$$
$$[\ddot{x}, \ddot{y}, \ddot{z}]^t = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3]^t \tag{3.16}$$

Orientación

Para observar que la orientación del cuadricóptero es función de estas salidas es necesario considerar las ecuaciones de movimiento 3.5 y 3.6. Si expresamos el vector z_b en función de σ obtenemos que:

$$\mathbf{z}_B = \frac{\mathbf{t}}{||\mathbf{t}||} \quad ; \quad \mathbf{t} = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g]^t \tag{3.17}$$

Por otro lado, siendo el angulo de yaw $\psi = \sigma_4$ se puede expresar el vector $\mathbf{x}_c$, obtenido después de aplicar únicamente la rotación sobre yaw ,como:

$$\mathbf{x}_c = [cos\ \sigma_4, sen\ \sigma_4, 0]^t \tag{3.18}$$

El objetivo es conseguir expresar la matriz de rotación $R_B = [\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B]$ en función de estos dos vectores $\mathbf{x}_C$ y $\mathbf{z}_B$. Esto se obtiene directamente al calcular $\mathbf{x}_B$ y $\mathbf{y}_B$ de la siguiente forma:

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{||\mathbf{z}_B \times \mathbf{x}_C||}\ ,\ \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \tag{3.19}$$

Velocidad angular

Para expresar la velocidad angular en función de σ y sus derivadas, es necesario considerar la primera derivada de la expresión 3.5, la cual se puede expresar como:

$$m\dot{\mathbf{a}} = \dot{F}\ \mathbf{z}_B + \Omega_{BW} \times F\ \mathbf{z}_B \tag{3.20}$$

Si proyectamos ésta expresión entorno al eje $\mathbf{z}_B$, considerando que $\dot{F} = \mathbf{z}_B \cdot m\dot{\mathbf{a}}$, podemos definir la proyección $\mathbf{h}_\Omega$ de $\frac{m}{F}\dot{\mathbf{a}}$ en el plano $\mathbf{x}_B - \mathbf{y}_B$ como:

$$\mathbf{h}_\Omega = \Omega_{BW} \times \mathbf{z}_B = \frac{m}{F}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B) \tag{3.21}$$

Si descomponemos los componentes de la velocidad angular Ω_{BW} como se realiza en la expresión 3.3 se obtienen las componentes p y q de la velocidad como:

$$p = -\mathbf{h}_\Omega \cdot \mathbf{y}_B \quad ; \quad q = \mathbf{h}_\Omega \cdot \mathbf{x}_B \tag{3.22}$$

Finalmente se puede obtener la componente r de la velocidad considerando $\Omega_{BW} = \Omega_{BC} + \Omega_{CW}$, donde la tercera componente de Ω_{BC} es nula por lo que:

$$r = \Omega_{CW} \cdot \mathbf{z}_B = \dot{\sigma}_4 \mathbf{z}_W \cdot \mathbf{z}_B \tag{3.23}$$

Aceleración angular

Finalmente, para considerar que la dinámica del cuadricóptero es diferencialmente plana es necesario expresar las componentes de la aceleración angular del mismo en función de las salidas planas y sus derivadas. Para ello comenzamos derivando la expresión 3.20:

$$\begin{aligned} m\ddot{\mathbf{a}} = &\ddot{F}\ \mathbf{z}_B + 2\Omega_{BW} \times \dot{F}\ \mathbf{z}_B \\ &+ \Omega_{BW} \times \Omega_{BW} \times F\ \mathbf{z}_B + \dot{\Omega}_{BW} \times F\ \mathbf{z}_B \end{aligned} \tag{3.24}$$

A partir de esta expresión podemos encontrar las componentes de la aceleración angular $\dot{\Omega}_{BW}$ considerando $\mathbf{h}_{\dot{\Omega}} = \dot{\Omega}_{BW} \times \mathbf{z}_B$ con lo que :

$$\dot{p} = -\mathbf{h}_{\dot{\Omega}} \cdot \mathbf{y}_B \quad ; \quad \dot{q} = \mathbf{h}_{\dot{\Omega}} \cdot \mathbf{x}_B \tag{3.25}$$

Para concluir se calcula la componente $\dot{r}$ de la aceleración angular de forma análoga a como se calculaba la componente $\dot{r}$ de la velocidad angular, con lo que se obtiene que :

$$\dot{r} = \dot{\Omega}_{CW} \cdot \mathbf{z}_B = \ddot{\sigma}_4 \mathbf{z}_W \cdot \mathbf{z}_B \tag{3.26}$$

Control

En este capítulo se desarrollará la teoría general para el control de un cuadricóptero [8][20][21][19], la cual se adaptará posteriormente al caso particular de la arquitectura empleada.

El problema del control se puede expresar como: Dado el estado del sistema $\mathbf{x}(t)$, la señal de entrada al sistema $\mathbf{u}(t)$ y el modelo dinámico del sistema $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$, el objetivo del control es encontrar la función $\mathbf{u}(t)$ tal que, el estado $\mathbf{x}(t)$ sigue la trayectoria deseada $\mathbf{\Gamma}^{des}(t)$ a lo largo del tiempo.

Como se observo en el apartado 3.2, las trayectorias deseadas $\mathbf{\Gamma}^{des}(t)$ constarán de 12 componentes : las 3 coordenadas correspondientes a la posición tridimensional deseada de la aeronave y el ángulo de yaw deseado, así como la primera y la segunda derivada de las mismas. Es por esto que si consideramos $\sigma_{des}(t) = [x_{des}, y_{des}, z_{des}, \psi_{des}]^t$, entonces las trayectorias deseadas son de la forma:

$$\mathbf{\Gamma}^{des} = \begin{bmatrix} \sigma_{des}(t) \\ \dot{\sigma}_{des}(t) \\ \ddot{\sigma}_{des}(t) \end{bmatrix} \tag{4.1}$$

Si se define el error de posición $\mathbf{e_p}(t)$ y el error de velocidad $\mathbf{e_v}(t)$ del controlador como:

$$\mathbf{e_p}(t) = \sigma_{des}(t) - \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \psi(t) \end{bmatrix} \quad ; \quad \mathbf{e_v}(t) = \dot{\sigma}_{des}(t) - \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \dot{\psi}(t) \end{bmatrix} \tag{4.2}$$

el objetivo del controlador sería conseguir que los errores $\mathbf{e_p}(t)$ y $\mathbf{e_v}(t)$ converjan de forma exponencial a 0.

El estado $\mathbf{x}(t)$ de un cuadricóptero consta de 12 variables, las 6 correspondientes a su pose y sus derivadas correspondientes.

$$\mathbf{x}(t) = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^t \tag{4.3}$$

Si llamamos $\mathbf{q}$ al vector de configuración formado por las 6 primeras variables:

$$\mathbf{q}(t) = [x \;\; y \;\; z \;\; \phi \;\; \theta \;\; \psi \;]^t \tag{4.4}$$

$$\mathbf{x}(t) = [\mathbf{q} \;\; \dot{\mathbf{q}}]^t \tag{4.5}$$

Para el control de la aeronave se han definido dos señales de entrada $\mathbf{u_1}$ y $\mathbf{u_2}$ siendo

$$\mathbf{u_1} = F = F_1 + F_2 + F_3 + F_4 \tag{4.6}$$

$$\mathbf{u_2} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \tag{4.7}$$

como se puede observar la señal de control u_1 actúa sobre el empuje total de la aeronave en el eje z_B mientras la señal u_2 controla los momentos totales aplicados sobre el cuadricóptero. Los pares M_i que ejerce cada motor se relacionan con F_i de la forma $M_i = \gamma F_i$, siendo γ una constante aerodinámica dependiente de las hélices empleadas. De esta forma se puede expresar u_2 de la siguiente forma:

$$\mathbf{u_2} = \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{4.8}$$

La arquitectura de control general consiste en dos controladores anidados: un controlador de posición y un controlador de orientación, como se puede observar en la figura .incluir diagrama

4.1. Controlador para ángulos pequeños

Cuando las trayectorias que el cuadricóptero son poco agresivas se puede linealizar el modelo del cuadricóptero entorno a su punto de equilibrio. En la situación de equilibrio el cuadricóptero se encuentra en *hover*, es decir manteniendo la posición en el aire . En *hover* el estado x_0 de la aeronave es de la forma

$$\begin{aligned} x_0 &= [q_0 \;\; 0]^t \\ q_0 &= [x_0 \;\; y_0 \;\; z_0 \;\; 0 \;\; 0 \;\; \psi_0 \,]^t \end{aligned} \tag{4.9}$$

Como los ángulos ϕ y θ son pequeños se puede aproximar $cos(\phi) \approx 1, cos(\theta) \approx 1$ y $sen(\phi) \approx \phi, sen(\theta) \approx \theta$. En este estado de equilibrio la fuerza total que ejercen los motores debe ser igual al peso del cuadricóptero por lo que:

$$F_{i,0} = \frac{mg}{4} \quad , \quad \mathbf{u_{1,0}} = mg \tag{4.10}$$

Al desglosar y linealizar las ecuaciones 3.6 entorno a este estado obtenemos las ecuaciones linealizadas del movimiento traslacional:

$$\begin{aligned} \ddot{x} &= g(\Delta\theta \, cos\psi_0 + \Delta\phi \, sen\psi_0) \\ \ddot{y} &= g(\Delta\theta \, sen\psi_0 - \Delta\phi \, cos\psi_0) \\ \ddot{z} &= \frac{1}{m}\mathbf{u_1} - g \end{aligned} \tag{4.11}$$

Controlador en ángulo

Por otro lado, para obtener las ecuaciones que permiten controlar la rotación del cuadricóptero debemos desarrollar las ecuaciones 3.12. Debido a la simetría del cuadricóptero se considera que $I_{xx} \approx I_{yy}$, con lo que se obtienen las ecuaciones del movimiento rotacional:

$$\begin{aligned} I_{xx}\dot{p} &= u_2 - qr(I_{zz} - I_{yy}) \\ I_{yy}\dot{q} &= u_3 - pr(I_{xx} - I_{zz}) \\ I_{zz}\dot{r} &= u_4 \end{aligned} \tag{4.12}$$

Se puede asumir que la componente r es pequeña por lo que los productos de esta con otros términos, también son pequeños en comparación con el resto de términos. Además, en el estado de *hover* $\dot{\phi} \approx p$, $\dot{\theta} \approx q$ y $\dot{\psi} \approx r$. Es por esto que la expresión anterior se puede simplificar por:

$$\begin{aligned} \dot{p} &= \frac{u_2}{I_{xx}} \\ \dot{q} &= \frac{u_3}{I_{yy}} \\ \dot{r} &= \frac{u_4}{I_{zz}} \end{aligned} \tag{4.13}$$

Gracias a esto es posible expresar el valor de las acciones de control $\mathbf{u_2}$ como:

$$u_{2,des} = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p) \tag{4.14}$$

$$u_{3,des} = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q) \tag{4.15}$$

$$u_{4,des} = k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r) \tag{4.16}$$

En el estado de hover,las velocidades de roll y pitch son cero por lo que $p_{des} = 0$, $q_{des} = 0$.

Controlador en posición

Con el controlador anterior se consigue controlar la orientación del cuadricóptero, este controlador se debe combinar con un controlador de más alto nivel que genere las referencias angulares para el controlador angular. En cuanto al tipo de controlador en posición se ha decidido emplear un controlador PD.

En un controlador PD el objetivo de seguir la trayectoria r_T deseada es conseguir que la expresión 4.17 tienda a 0 en un tiempo exponencial:

$$(\ddot{r_T} - \ddot{r_c}) + K_d e_v + K_p e_p = 0 \tag{4.17}$$

siendo $e_v = (\dot{r}_T - \dot{r}_c)$ y $e_p = (r_T - r_c)$. Las constantes $K_d, K_p \in \mathbb{R}^+$ del controlador definen la dinámica y la estabilidad del sistema.

Desarrollando 4.17:

$$\ddot{r_c} = \ddot{r_T} + K_d e_v + K_p e_p \tag{4.18}$$

Al combinar 4.11 con 4.18:

$$\begin{bmatrix} g(\Delta\theta\, cos\psi_0 + \Delta\phi\, sen\psi_0) \\ g(\Delta\theta\, sen\psi_0 - \Delta\phi\, cos\psi_0) \\ \frac{1}{m}\mathbf{u_1} - g \end{bmatrix} = \ddot{r_T} + K_d e_v + K_p e_p \tag{4.19}$$

por lo que :

$$\mathbf{u_1} = m\left(g + \ddot{z_{des}} + K_{d,z}(\dot{z_{des}} - \dot{z}) + K_{p,z}(z_{des} - z)\right) \tag{4.20}$$

$$\phi^{des} = \frac{1}{g}\left(\ddot{x}_{des} sin\,\psi_T - \ddot{y}_{des} cos\,\psi_T\right) \tag{4.21}$$

$$\theta^{des} = \frac{1}{g}\left(\ddot{x}_{des} cos\,\psi_T + \ddot{y}_{des} sin\,\psi_T\right) \tag{4.22}$$

Como el angulo de *yaw* $\psi_T(t)$ se incluye en la trayectoria definida, entonces:

$$\psi_{des} = \psi_T(t) \tag{4.23}$$

$$\dot{\psi}_{des} = \dot{\psi}_T(t) \tag{4.24}$$

Con esto se consigue obtener la señal de control $\mathbf{u_1}$ así como las referencias del controlador angular ϕ^{des}, θ^{des} y ψ^{des}.

4.2. Controlador para ángulos grandes

A continuación se presenta un controlador que no parte de la premisa de que el estado de la aeronave es cercano al de equilibrio, sino que parte de la dinámica diferencialmente plana del cuadricóptero (apartado 3.2), lo que permite seguir trayectorias suaves de la forma $\mathbf{\Gamma}^{des}(t)$.

El objetivo de este controlador es orientar el vector z_B de la aeronave con la dirección de la fuerza necesaria $\mathbf{F}_{des}$ para poder seguir la trayectoria.

añadir diagrama

Para ello primero definiremos los errores en posición y velocidad:

$$\mathbf{e}_p = \mathbf{r} - \mathbf{r}_T \tag{4.25}$$

$$\mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_T \tag{4.26}$$

La fuerza deseada $\mathbf{F}_{des}$ en cada instante de tiempo viene dada por la expresión

$$\mathbf{F}_{des} = -K_p\mathbf{e}_p - K_v\mathbf{e}_v - mg\mathbf{z}_W + m\ddot{\mathbf{r}}_T \tag{4.27}$$

donde K_p y K_v son matrices definidas positivas. Como se puede observar, la fuerza debe tener en cuenta el peso de la aeronave, la aceleración impuesta por la trayectoria en ese instante y el error de seguimiento de la trayectoria en ese instante para poder corregirlo.

Al proyectar $\mathbf{F}_{des}$ sobre el eje x_B del cuadricóptero se puede obtener u_1, por lo que:

$$u_1, des = \mathbf{F}_{des} \cdot \mathbf{z}_B \tag{4.28}$$

Las otras tres señales de control del sistema dependen de la orientación de la aeronave. El objetivo es alinear el eje $\mathbf{z}_B$ de la aeronave con la dirección de la fuerza deseada $\mathbf{z}_{B,des}$, es decir, que la matriz de rotacion R_{des} viene dada por:

$$R_{des}\mathbf{e_3} = \mathbf{z}_{B,des}\ , \quad \mathbf{z}_{B,des} = \frac{\mathbf{F}_{des}}{||\mathbf{F}_{des}||} \tag{4.29}$$

siendo $\mathbf{e_3}$ el vector director del eje z_B de la aeronave respecto al sistema B, es decir, $\mathbf{e_3} = [0, 0, 1]^t$. Debido a que el ángulo de yaw ψ_T viene dado por la trayectoria se pueden calcular los vectores de rotación deseados de la forma:

$$\mathbf{x}_{C,des} = [cos\,\psi_T, sin\psi_T, 0]^t \tag{4.30}$$

$$\mathbf{y}_{B,des} = \frac{\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}}{||\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}||}\ ,\ \mathbf{x}_{B,des} = \mathbf{y}_{B,des} \times \mathbf{z}_{B,des} \tag{4.31}$$

por lo que la matriz $R_{des} \in SO(3)$ se expresa como:

$$R_{des} = \begin{bmatrix} | & | & | \\ \mathbf{x}_{B,des} & \mathbf{y}_{B,des} & \mathbf{z}_{B,des} \\ | & | & | \end{bmatrix} \tag{4.32}$$

Siendo R la matriz de rotación correspondiente al estado actual de la aeronave, definiremos el error en orientación $\mathbf{e}_R$ como:

$$\mathbf{e}_R = \frac{1}{2}\left(R_{des}^t R - R^t R_{des}\right)^\vee \tag{4.33}$$

donde $\vee$ denota el vee map que transforma elementos de $SO(3)$ a $\mathbb{R}$. El error de velocidad angular es simplemente la diferencia entre la velocidad angular de la aeronave y la velocidad angular deseada, ambas medidas con respecto las coordenadas del sistema de coordenadas del cuadricóptero B.

$$\mathbf{e}_\omega = {}^B[\omega] - {}^B[\omega_T] \tag{4.34}$$

Finalmente, las tres señales de salida restantes, correspondientes con los pares ejercidos sobre la aeronave, se calculan como:

$$\begin{bmatrix} u_{2,des} \\ u_{3,des} \\ u_{4,des} \end{bmatrix} = -K_R \mathbf{e}_R - K_\omega \mathbf{e}_\omega \tag{4.35}$$

Este controlador presenta dos ventajas considerables con respecto al controlador anterior:

1. El error en orientación está expresado mediante matrices de rotación y no mediante ángulos de Euler, los cuales presentan una singularidad en torno a $\theta = 90^o$ conocida como *gimbal lock*.

2. La fuerza deseada se proyecta sobre el vector z_B de la aeronave en el instante actual, en lugar de considerar que éste siempre apunta verticalmente, por lo que mejora notablemente el control en trajectorias agresivas, donde la aeronave presenta una gran inclinación.

4.3. Adaptación del controlador al entorno simulado.

En los controladores anteriores las señales de salida están directamente relacionadas con la velocidad de giro de cada motor y de la fuerza de empuje que genera cada uno, como se puede observar en las expresiones 4.6 y 4.8. Sin embargo, en la competición, no se tiene el control de cada motor de forma independientemente, si no que hay un controlador interno que controla los motores. Es por esto que, es necesario adaptar los controladores del estado del arte mostrados anteriormente al nuevo conjunto de señales de entrada disponibles.

Añadir diagrama nuevo

El controlador interno del cuadricóptero tiene como consignas a seguir las velocidades angulares del cuadrícoptero deseadas, así como el empuje total que deben proporcionar los motores. Estas son las señales que envían los pilotos de carreras de drones profesionales a través de sus mandos, debido al gran control que tienen de la aeronave. Al poder controlar el empuje total de la aeronave, la señal u_1 se mantiene igual que en los controladores anteriores, sin embargo es necesario realizar algunas pequeñas modificaciones en los controladores angulares. Las otras 3 señales de control que se van a emplear son las referencias del controlador interno p_{ref},q_{ref} y r_{ref}.

Modificaciones para ángulos pequeños

Como se puede observar en la expresión 4.14, el controlador angular es un control PD que depende de la orientación actual de la aeronave y de la orientación deseada. Dado que el objetivo es minimizar ese error de orientación, podemos expresar el controlador angular como:

$$u_2 = p_{ref} = p_{des} + k_{p,\phi}(\phi^{des} - \phi) \tag{4.36}$$

$$u_3 = q_{ref} = q_{des} + k_{p,\theta}(\theta^{des} - \theta) \tag{4.37}$$

$$u_4 = r_{ref} = r_{des} + k_{p,\psi}(\psi^{des} - \psi) \tag{4.38}$$

Modificaciones para ángulos grandes

Para el controlador para grandes ángulos se ha aplicado una estrategia similiar a la anterior. La expresión 4.35 se ha modificado para que sea de la forma:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = {}^{B}[\omega] = {}^{B}[\omega_T] - K_R \mathbf{e}_R \tag{4.39}$$

Con estas pequeñas modificaciones se ha conseguido adaptar los controladores del estado del arte al interfaz provisto por el entorno de simulación que se empleará posteriormente en los experimentos.

Generación de trayectorias

En el apartado anterior se ha diseñado un controlador cuyo objetivo es seguir una trayectoria deseada minimizando el error de seguimiento. En este apartado se tratará sobre la forma en la que se generan estas trayectorias. Este desarrollo se realiza entorno a una trayectoria unidimensional en un único eje de desplazamiento , el eje x. Este desarrollo es ampliable al espacio tridimensional realizando una trayectoria por cada eje.

5.1. Trayectorias óptimas

El objetivo general del control óptimo es encontrar la función $x^*(t)$ que minimiza la expresión

$$x^*(t) = \underset{x(t)}{argmin} \int_0^T \mathcal{L}\left(x^{(n)}, x^{(n-1)}, ..., \dot{x}, x, t\right)\, dt \tag{5.1}$$

siendo $\mathcal{L}$ el índice que se debe optimizar.

Cuando el objetivo es generar trayectorias suaves la forma de la expresión a optimizar es :

$$x^*(t) = \underset{x(t)}{argmin} \int_0^T \left(x^{(n)}\right)^2\, dt \tag{5.2}$$

siendo $x^{(n)} = u$, la magnitud de control que se desea minimizar. El valor del parámetro $n \in \mathbb{N}$, expresa el grado de la derivada de la acción de control u, es decir, para generar trayectorias de mínima distancia, se empleará un valor de $n = 1$, por lo que $u = \dot{x}$, mientras que para generar trayectorias de mínima sobreaceleración (*jerk*) se empleara un valor de $n = 3$, por lo que $u = \dddot{x}$.

De forma general la expresión a optimizar será:

$$x^*(t) = \underset{x(t)}{argmin} \int_0^T \mathcal{L}\left(x^{(n)}, x^{(n-1)}, ..., \dot{x}, x, t\right)\, dt \tag{5.3}$$

De la ecuación de Euler-Lagrange se obtiene que la la función optima $x^*(t)$ debe satisfacer:

$$\frac{\partial \mathcal{L}}{\partial x} + \sum_{i=1}^{n} (-1)^i \frac{d^i}{dt^i}\left(\frac{\partial \mathcal{L}}{\partial x^i}\right) = 0 \tag{5.4}$$

Si particularizamos la expresión anterior a expresiones donde $\mathcal{L} = \left(x^{(n)}\right)^2$, todos las

derivadas parciales $\frac{\partial \mathcal{L}}{\partial x^i} = 0 \; ; \forall i \neq n$ por lo que:

$$\frac{d^n}{dt^n}\left(\frac{\partial \mathcal{L}}{\partial x^n}\right) = 0 \tag{5.5}$$

$$\frac{d^n}{dt^n}\left(2x^{(n)}\right) = 0 \tag{5.6}$$

$$x^{(2n)} = 0 \tag{5.7}$$

Al integrar la expresión anterior se obtiene el polinomio de grado $2n-1$, con $2n$ coeficientes:

$$x(t) = c_{2n-1}t^{2n-1} + c_{2n-2}t^{2n-2} + ... + c_2t^2 + c_1t + c_0 \tag{5.8}$$

Para resolver los coeficientes, son necesarias $2n$ condiciones de contorno que corresponden con los valores de la función $x(t)$ y de sus derivadas en los instantes $t = 0$ y $t = T$. Si se considera que la trayectoria transcurre entre un estado inicial y un estado final, ambos estados de equilibrio , se puede considerar que las derivadas son nulas por lo que las condiciones de contorno serían:

$$x(0) = a; \quad \dot{x}(0) = 0; \quad ... \quad x^{(2n-1)}(0) = 0 \tag{5.9}$$

$$x(T) = b; \quad \dot{x}(T) = 0; \quad ... \quad x^{(2n-1)}(T) = 0 \tag{5.10}$$

Con estas condiciones de contorno se calcula la trayectoria óptima para pasar de una posición inicial a a una posición final b en un tiempo T minimizando la acción de control $u = x^{(n)}$.

5.2. Trayectorias continuas a trozos

Para generar trayectorias que el cuadricóptero deba seguir para completar el circuito es conveniente contar con puntos de paso (*waypoints*) por los que se quiere que pase la aeronave, por ejemplo, el centro de las puertas.

Añadir diagrama

Como se puede observar en el apartado anterior, las trayectorias que se generan entre dos puntos de paso a y b, son trayectorias polinómicas cuyo grado depende del orden de la derivada de la acción de control. Debido a esto, se ha decidido usar trayectorias de tipo *spline*, es decir, curvas diferenciables definidas en segmentos polinómicos, donde cada segmento sería una trayectoria entre dos *waypoints*. Cada trayectoria completa, $r_d(t)$ está compuesta por m segmentos de grado q, presentando la estructura

$$r_d(t) = \begin{cases} \sum_{i=0}^{q} a_{i1}t^i & , t_0 \leq x \leq t_1 \\ \sum_{i=0}^{q} a_{i2}t^i & , t_1 \leq x \leq t_2 \\ \vdots & \vdots \\ \sum_{i=0}^{q} a_{im}t^i & , t_{m-1} \leq x \leq t_m \end{cases} \tag{5.11}$$

siendo $a_{ij} \in \mathbb{R}$ los coeficientes de los polinomios. El grado q de estos polinomios dependerá del índice a minimizar durante el transcurso de la trayectoria, como se ha explicado en el apartado 5.1. Cada tramo tiene un tiempo inicial y un tiempo final, estos tiempos dependen de las limitaciones físicas de la aeronave, es decir, de su velocidad y

aceleración máximas, en el apartado 5.3 se profundiza en la forma de generar estos tramos temporales.

De cara a resolver el valor de los distintos coeficientes de cada segmento polinomial se emplearán unas condiciones de contorno que garanticen la derivabilidad de la trayectoria completa en las q primeras derivadas.

Si se considera una trayectoria completa a través de n *waypoints* P distribuidos a lo largo del tiempo, siendo t_0 el tiempo inicial del primer segmento y t_n el tiempo final de la trayectoria, la trayectoria contará con $n-1$ segmentos polinómicos S de grado q. Considerando que en el punto inicial y el punto final de la trayectoria el cuadricóptero se encuentra en un estado de equilibrio, las condiciones de contorno empleadas para calcular los coeficientes de la *spline* son:

$$\begin{array}{ll} S_i(t=t_{i-1}) = W_{i-1} & \forall i \in 1,...,n \\ S_i(t=t_i) = W_i & \forall i \in 1,...,n \\ S_1^{(k)}(t=t_0) = 0 & \forall k \in 1,...,q \\ S_n^{(k)}(t=t_n) = 0 & \forall k \in 1,...,q \\ S_{i-1}^{(k)}(t=t_{i-1}) = S_i^{(k)}(t=t_i) & \forall k \in 1,...,q\ ; \forall n \in 2,...,n \end{array} \tag{5.12}$$

siendo W_i la posición del i-ésimo *waypoint* , S_i el i-ésimo segmento polinómico y t_i el tiempo de finalización del segmento S_i.

Las dos primeras condiciones de contorno hacen referencia a las posiciones espaciales de los waypoints, es decir, que cada segmento empiece en un waypoint y termine en el siguiente. Las dos condiciones siguientes hacen referencia al valor de las derivadas en los estados iniciales y final, se ha considerado que los estado inicial y final, son estados de equilibrio por lo que el valor de sus derivadas es nulo. Finalmente la ultima condición hace referencia a la continuidad de las derivadas entre segmentos, es decir, que el valor de las derivadas del final de un segmento, sean iguales al las derivadas iniciales del segmento siguiente, de esta forma se consigue que la trayectoria completa generada cumpla la condición de derivabilidad a lo largo de todas sus q derivadas.

5.3. Estimación de tiempos

El objetivo final es generar trayectorias a través del circuito de forma que la aeronave sea capaz de recorrerlas teniendo en cuenta restricciones de posición, velocidad y aceleración. En los dos apartados anteriores se explica como generar trayectorias óptimas teniendo en cuentas restricciones de posición, fijando los *waypoints* por los que se desea que pase el cuadricóptero. Sin embargo, para poder generar trayectorias es también necesario elegir los segmentos temporales T_i entre los distintos puntos de paso, es decir, los tiempos que debe tardar la aeronave en pasar entre los distintos waypoints. Estos segmentos de tiempo deben tener en cuenta las limitaciones de velocidad y acceleración máximas de la aeronave, para poder generar trayectorias realizables. A continuación se tratará un posible método para establecer la duración de estos intervalos temporales.

Aproximación al tiempo medio

La forma más sencilla de establecer intervalos temporales es basarse en la distancia existente entre un par de waypoints y la velocidad media a la que se desea que vuele la

aeronave para estimar el tiempo necesario. Siendo $d_i = ||W_{i-1}, W_i||_2$ la distancia entre dos *waypoints* consecutivos y v_m la velocidad media de vuelo, se puede establecer la duración de un segmento S_i como

$$T_i = \frac{d_i}{v_m} \tag{5.13}$$

Para obtener los valores de tiempo absoluto t_i requeridos en el apartado anterior simplemente se deben sumar las duraciones de los segmentos anteriores.

$$t_i = \sum_{j=1}^{i} T_j \tag{5.14}$$

Para tener aumentar el control sobre la agresividad de la trayectoria se puede añadir un coeficiente α_i que permita modificar ligeramente el valor T_i siendo

$$T_i = \alpha_i \frac{d_i}{v_m} \tag{5.15}$$

Valores de α_i pequeños generan trayectorias más agresivas, así como valores más grandes producen trayectorias más seguras. Iterando el valor de este parámetro α_i se controla que las limitaciones en velocidad y aceleración máxima se cumplan para cada segmento S_i.

Esta es una forma simple y rápida de establecer los distintos segmentos temporales, sin embargo, es una aproximación que afecta a la generación de la trayectoria óptima, haciendo que el tiempo total que se tarda en recorrer el circuito sea mayor que el que se podría obtener usando otros métodos de optimización como los propuestos por Mellinger et al. [8] o por Foehn. et al. [22] para la obtención de estos segmentos temporales. A pesar de esto, debido a que las posiciones de los *waypoints* cambian rápidamente a lo largo del circuito y se deben recalcular las trayectorias continuamente, el bajo coste computacional que presenta este método hace que merezca la pena su utilización.

Metodología

En los capítulos anteriores se ha presentado los distintos algoritmos de control y generación de trayectorias que se emplearán para intentar recorrer un circuito con un cuadricóptero de carreras autónomo en un entorno simulado. En este apartado se presenta la metodología empleada para superar el circuito que fue propuesto durante las pruebas clasificatorias del Alphapilot 2019.

6.1. Sistemas de referencia

Antes de continuar con la metodología empleada, es conveniente fijar los sistemas de referencia que se van a emplear. Primeramente se empleara un sistema de coordenadas global en el mundo. El origen de este sistema de referencia global fijo viene dado por el simulador y se encuentra en el centro del circuito. A partir de este sistema estableceremos el sistema de referencias móvil del aeronave, cuyo origen se encuentra en el centro de masas del cuadricóptero y las direcciones de los ejes siguen el estándar ENU (*East North Up*). Finalmente, se generará un sistema de referencia móvil para cada puerta del circuito, cuyos orígenes corresponderán al centro de las puertas.

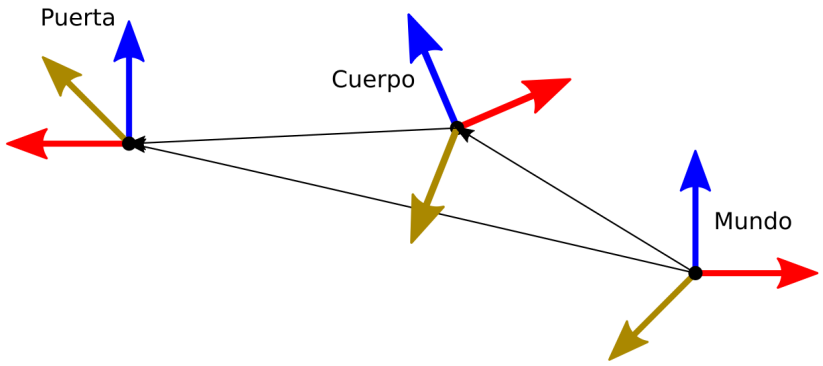

Figura 6.1: Transformaciones entre los sistemas de referencia de las puertas, el cuerpo de la aeronave y el mundo.

6.2. Generación de *waypoints*

Para recorrer el circuito de forma satisfactoria es necesario que la aeronave atraviese las distintas puertas o *gates* que componen el circuito en un orden concreto. Para conseguir

esto es necesario conocer las posiciones de las puertas en el mundo y generar los puntos de paso necesarios para que la aeronave pase a través de ellas sin colisionar.

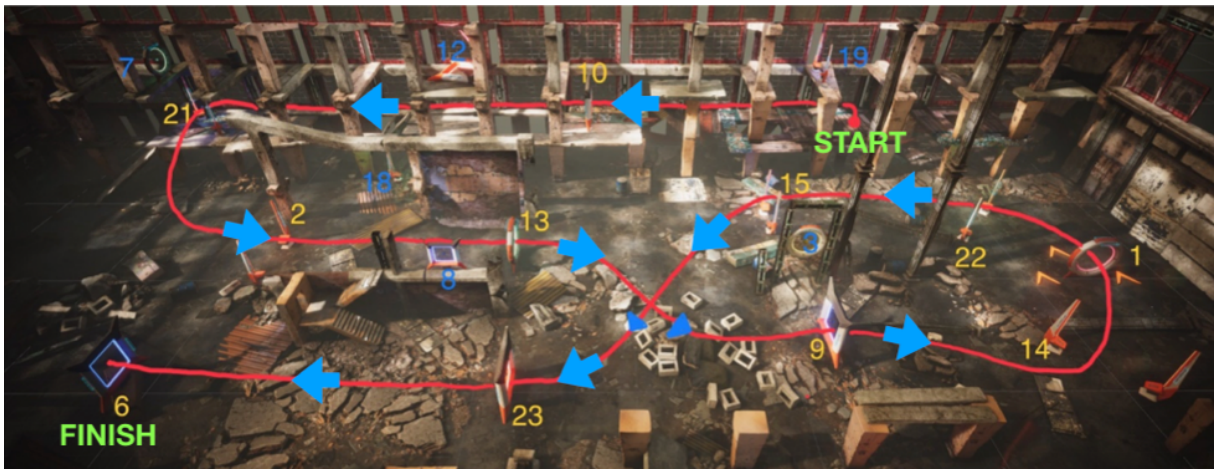

Figura 6.2: Vista aérea del circuito en el simulador FightGoggles, las puertas que se deben traspasar se simbolizan con su número en color amarillo.

Como se puede observar en la figura 6.2 la aeronave debe recorrer 11 puertas, cada una con un número de identificación, en un orden concreto. Cada una de estas puertas posee unas medidas estandarizadas de 2x2 m.

En la competición se proporciona el orden en el que se deben atravesar las puertas y una posición aproximada de las posiciones de cada una de ellas en el mundo. Esta posición aproximada posee un error significativo, por lo que es necesario corregir la estimación de la posición de estas puertas a medida que la aeronave avanza por el circuito (estimación *online*).

Para poder realizar la estimación *online* de las puertas, es necesario percibirlas, para ello se hace uso de las cámaras integradas en la aeronave simulada. Las cámaras del cuadricóptero permiten localizar las distintas puertas a lo largo del circuito. La imagen provista por el simulador, además de contener la imagen observada por la cámara, contiene también las posiciones en el plano imagen, de las esquinas de las puertas visibles, así como el identificador de puerta a la que corresponde.

Añadir imagen con los puntos rojos

Esta información extra provista por el simulador facilita enormemente la estimación de las posiciones de las distintas puertas, reduciendo también el tiempo de calculo requerido.

Dado que se cuenta con los parámetros físicos de la cámara y con las dimensiones de las puertas, es posible emplear un algoritmo PnP (*Perspective n-Points*) para calcular la posición relativa de los *frames* con respecto a la aeronave. Concretamente, se ha empleado la medida de la posición del centro de la puerta, ya que nos permite generar los waypoints de forma más sencilla.

La incertidumbre de estas medidas disminuye a medida que el aeronave se acerca a las puertas, por lo que cuando las imágenes se toman a una distancia lejana, el error que tienen es elevado. Para disminuir la influencia de las medidas erróneas cada medida se somete a un filtrado en dos pasos:

1. **Región de confianza:** Dado que siempre se posee una posición estimada de cada puerta es posible emplear esta información para descartar medidas erróneas. Siendo

$\hat{G}_i$ la estimación previa del centro de la puerta i, se establece una bola $B(\hat{G}_i, r)$ con centro en $\hat{G}_i$ y radio r como región de confianza, es decir, si una nueva medida $G_i \notin B(\hat{G}_i, r)$ entonces se desecha como una medida errónea. El valor del radio r influye en la distancia máxima que puede tener una medida respecto a la estimación original para considerarse correcta. Dado que al comienzo del circuito se tienen unas posiciones de las puertas con un error muy grande, este valor r no puede ser muy bajo, si no no se conseguiría corregir la posición de las puertas con mediciones correctas. Por el contrario, si el valor de r es muy grande, este filtrado carecería de sentido, ya que cualquier medición se consideraría válida. En la practica se han probado con distintos valores de este parámetro, obteniendo mejores resultados con valores de r de entre 4 y 6 metros.

2. **Media móvil:** Si la medida obtenida G_i se encuentra contenida en esa región, entonces, en lugar de sustituir directamente la estimación de la posición de esta puerta, se realiza una modificación de la medición anterior, de acorde a la fórmula:

$$\hat{G}_i = \alpha \hat{G}_{i,prev} + (1 - \alpha) G_i \tag{6.1}$$

siendo $\hat{G}_{i,prev}$ la estimación anterior y $\alpha \in (0, 1)$ un parámetro de filtrado. Cuando α tiene valores pequeños, la estimación cambia rápidamente con las nuevas medidas, mientras que si α posee valores altos, la estimación varía ligeramente con cada una de las nuevas medidas. Los valores que se suelen emplear son $\alpha = 0{,}9$ y $\alpha = 0{,}99$. Para este filtrado se ha empleado un valor de $\alpha = 0{,}9$ dado que presenta un buen equilibrio entre robustez y reactividad ante nuevas medidas.

Con este proceso se consiguen actualizar las posiciones estimadas de las puertas con una frecuencia aproximada de 60 Hz, la frecuencia de refresco de las cámaras. En los casos en los que se encuentran varias puertas en la imagen se realizan el filtrado anterior para cada una de ellas por separado.

6.3. Trayectorias a largo y corto plazo

Con la posición estimada de cada puerta en el mundo es posible construir la trayectoria que el dron debe seguir para completar el circuito. El problema es que estas posiciones cambian continuamente, siendo más fiables cuanto más cerca se encuentre el aeronave de la puerta, por lo que es necesario actualizar estas trayectorias de la forma mas ágil posible. Para conseguir que el sistema sea capaz de reaccionar de forma fluida a los cambios en la estimación de las puertas, se ha dividido la generación de trayectorias en dos partes : una a largo plazo que tiene en cuenta las posiciones de las puertas y una a corto plazo que genera trayectorias óptimas en un horizonte temporal finito.

Dado que el circuito es amplio y no existen obstáculos entre dos puertas sucesivas, no se tienen en cuenta los obstáculos del circuito para la generación de trayectorias. Para asegurar esto, es necesario que las trayectorias generadas no se alejen demasiado de las lineas rectas que unen las distintas puertas sucesivas. A continuación, se explicará de forma más detallada cómo se generan estas trayectorias.

6.3.1. Trayectoria a largo plazo

Para poder generar trayectorias suaves a lo largo de todo el circuito es conveniente tener en cuenta las posiciones aproximadas de todas las puertas, de esta forma a medida que el aeronave recorre la trayectoria obtiene nuevas estimaciones que permiten ir refinando la trayectoria. Si se generase la trayectoria solo teniendo en cuenta los m siguientes waypoints, la trayectoria a seguir variaría mucho después de pasar una puerta, lo que perjudicaría en el rendimiento de la aeronave. Dado que el dron obtiene nuevas mediciones a una frecuencia de unos 60 Hz esta trayectoria se tiene que ir actualizando continuamente, por lo que es conveniente minimizar el tiempo de cómputo de cada una.

Como se ha presentado en el capítulo 5 las trayectorias que se van a emplear son *splines* en las que el grado de los segmentos polinómicos depende de la acción de control a minimizar. Los dos factores que afectan principalmente al tiempo de cómputo son el grado de los segmentos polinomiales, ya que a mayor grado, mayor número de coeficientes se deben calcular, y el número de *waypoints*, ya que aumenta el número de segmentos polinomiales.

Para poder generar una ruta a través de todo el circuito que se calcule de la forma más rápida posible y que tenga en cuenta las limitaciones físicas de la aeronave, se ha empleado trayectorias de aceleración mínima ($n = 2$) con 12 *waypoints*, 11 de las puertas y 1 de la posición de la aeronave. Estas trayectorias son rápidas de calcular y permiten actualizar la trayectoria con una frecuencia superior a la de adquisición de la cámara. Estas trayectorias cambian continuamente, por lo que no es conveniente usar estas trayectorias para controlar el cuadricóptero ya que las referencias del controlador estarían cambiando continuamente, generando un movimiento muy poco fluido. Para solucionar esto se emplean las trayectorias a corto plazo.

Añadir alguna imagen en rviz

6.3.2. Trayectoria a corto plazo

Para generar la trayectoria que va a seguir el controlador, es conveniente generar trayectorias de *snap* mínimo ($n = 4$), ya que son las que generan trayectos más suaves. El grado de los polinomios que componen estas *splines* son de grado 7, lo que implica un elevado coste computacional. Además, es conveniente que los *waypoints* empleados para generar esta trayectoria estén cerca (una distancia inferior a los 8 m entre waypoints), ya que si se eligen waypoints muy separados, la trayectoria resultante puede separarse demasiado de la linea recta entre ellos, llegando a poder colisionar con el entorno. Por lo que para generar una trayectoria completa del circuito sería necesaria una gran cantidad de waypoints, por lo que el tiempo de cómputo sería muy elevado, haciendo que el sistema no sea capaz de reaccionar a cambios en las posiciones de las puertas de forma rápida.

Para poder conseguir trayectorias de control óptimas se ha decidido partir de la trayectoria a largo plazo y establecer un horizonte temporal, sobre el cual se calcule la trayectoria a corto plazo. Este proceso se genera en varias etapas:

1. Se localiza la posición del cuadricóptero dentro de la trayectoria a largo plazo. Para ello se parte de la última posición conocida de la aeronave en la trayectoria t_c y se comprueba en un pequeño intervalo, en que posición se encuentra la aeronave. Dado que el sistema tiene inercia y los algoritmos tardan un tiempo en procesarse, es necesario recalcular esta posición para obtener una mayor precisión.

2. Se muestrea la trayectoria a largo plazo para obtener los waypoints de la trayectoria a corto plazo. Partiendo de la posición obtenida previamente t_c, se fija un horizonte temporal en el que se quiere calcular la trayectoria a corto plazo t_h, también se fija la distancia a la que se quiere que se encuentren los waypoints t_d. Con estos datos se muestrea la trayectoria en el intervalo $t \in (t_c, t_c + t_h)$ con una distancia de muestreo t_d entre muestras. Las muestras obtenidas constituyen el conjunto de *waypoints* que se emplearán para generar la trayectoria a corto plazo.

3. Finalmente, se genera una trayectoria de *snap* mínimo empleando el conjunto de waypoints obtenidos previamente. La trayectoria obtenida $P(t) : [0, t_f] \rightarrow \mathbb{R}^3$ proporciona las posiciones en el espacio tridimensional en las que deberá estar el cuadricóptero para cada instante de tiempo. Para generar las consignas del controlador es conveniente conocer también las velocidades y acceleraciones para cada instante de tiempo, por lo que se derivan estos splines para obtener la trayectoria en velocidad $V(t) : [0, t_f] \rightarrow \mathbb{R}^3$ y la trayectoria en acceleración $A(t) : [0, t_f] \rightarrow \mathbb{R}^3$ siendo t_f el tiempo de finalización de la trayectoria.

Para conseguir que estas trayectorias tengan en cuenta los cambios $P(t)$, $V(t)$ y $A(t)$ en la estimación de las puertas, estas trayectorias se recalculan periódicamente a medida que se modifica la trayectoria a largo plazo.

6.3.3. Orientación de la aeronave en *yaw*

Además de las referencias de posición en el espacio tridimensional, para controlar el estado de la aeronave es necesario indicar el angulo de *yaw* deseado. De cara a obtener las mejores estimaciones en las posiciones de las puertas, se desea que la aeronave siempre este orientada de forma que las camáras miren hacia delante. Partiendo de la trayectoria a corto plazo $V(t)$ es facil obtener el valor requerido del angulo de *yaw* ψ. Siendo $v_x(t)$ y $v_y(t)$ las componentes de la trayectoria $V(t)$ para el eje x y el eje y respectivamente, en ángulo de yaw deseado es:

$$\psi_{des}(t) = -atan2(v_x, v_y) + \pi/2 \tag{6.2}$$

siendo $atan2(x, y)$ el arco tangente de dos parámetros entre x e y. Con esta simple expresión se consigue que la aeronave se oriente de forma que las cámaras miren hacia donde se va a mover la aeronave.

Arquitectura del sistema

Para conseguir que los distintos módulos que integran el sistema trabajen de forma conjunta es necesario establecer una arquitectura que permita la comunicación eficiente entre ellos así. Para coordinar el trabajo de los distintos módulos que componen el sistema se ha empleado ROS (*Robot Operating System*) [23] un *framework* orientado a el desarrollo de software para robots ampliamente extendido en la comunidad robótica. En el paradigma de ROS el código se estructura en nodos independientes que se comunican entre ellos a través de tópicos (mensajes que un nodo difunde de forma general a los demás nodos y cualquiera lo puede leer) y servicios (peticiones de un nodo particular a otro).

Esto permite desarrollar cada componente del sistema de forma independiente y comunicarlos entre ellos mediante una interfaz común. Esto permite encapsular el código, lo que aumenta la reusabilidad y la robustez de cada módulo, independiente del resto de módulos que les rodeen.

A continuación se explicará más detalladamente los distintos nodos que componen la arquitectura:

- **Simulador:** En la arquitectura propuesta el entorno de simulación empleado, Flightgoogles, se comporta como un nodo adicional. Este nodo constituye el interfaz de comunicación entre la aeronave simulada y el entorno simulado, el cual publica datos sobre el estado de la aeronave como su posición y orientación, las imágenes de las cámaras simuladas y la posición de las esquinas de las puertas en las imágenes obtenidas. Asimismo, recibe los comandos de control enviados a la aeronave.

- **Estimación de estado.** Este modulo se encarga de completar la estimación de estado de la aeronave, empleando la información temporal sobre los cambios de pose de la aeronave. Además proporciona esta información a través de mensajes estándar al resto de los módulos de la arquitectura.

- **Percepción.** El simulador provee las imágenes, tomadas por las cámaras, del entorno simulado. En estas imágenes aparecen las puertas del circuitos con un indicador del número de puerta, así como las posiciones de las esquinas de las puertas en las imágenes tomadas. Conociendo los parámetros de la cámara y las dimensiones de las puertas se puede extrapolar la posición de las puertas respecto a la cámara. El modulo de percepción se encarga de enviar las posiciones aproximadas de las distintas puertas del circuito e ir actualizando estas estimaciones a medida que la aeronave avanza a través del circuito.

- **Generador de trayectorias.** Una vez conocida la posición de la aeronave en el circuito y las posiciones aproximadas de las puertas con respecto a la aeronave se genera la trayectoria que debe seguir la aeronave para pasar a través de las puertas

de la forma más rápida posible sin colisionar con el entorno. Para poder actualizar la trayectoria de forma rápida con un coste bajo computacional se divide este trabajo en dos módulos:

- **Trayectoria a largo plazo**: Obtiene las posiciones de las puertas provistas por el módulo de percepción y genera el recorrido tridimensional completo que debería realizar el cuadricóptero.
- **Trayectoria a corto plazo**: Recibe la trayectoria a largo plazo y la evalúa en un corto horizonte temporal respecto a la posición actual de la aeronave. Dentro de este horizonte se genera la trayectoria de control óptima a seguir. Este módulo también se encarga de evaluar la trayectoria actual a lo largo del tiempo y enviar las consignas de posición, velocidad y aceleración al controlador.

- **Controlador:** El módulo del controlador recibe el estado actual de la aeronave provisto por el modulo de estimación y la referencias provistas por el módulo trayectoria a corto plazo. Con esto genera las acciones de control que se envían al módulo del simulador.

Añadir Diagrama de la arquitectura con los topics involucrados.

7.1. Detalles de implementación

Todos los algoritmos desarrollados se han implementado en C++17 haciendo uso tanto de las librerías estándar como de las librerías estándar de ROS . Además se ha hecho uso de algunas librerías específicas como OpenCV 4.0 para el tratamiento de las imágenes, así como Eigen3 para computar las operaciones matriciales requeridas por los algoritmos implementados. A continuación, se explicarán algunos detalles de implementación relevantes de los distintos módulos.

7.1.1. Módulo de control

Para facilitar el ajuste de las ganancias de los distintos controladores se ha empleado la librería de ROS *dynamic reconfigure*. Esta librería permite variar el valor de distintos parámetros en tiempo de ejecución, lo que facilita enormemente el ajuste fino de los parámetros de los controladores. Imagen Dynamic Reconfigure

7.1.2. Módulo de generación de trayectorias

Dado que las dos trayectorias generadas, la de corto y la de largo plazo, se generan empleando *splines* se ha unificado el algoritmo de generación en una clase común a ambos módulos. Aunque al comienzo del proyecto se empleo una implementación propia de estos algoritmos, finalmente se ha optado por emplear la implementación realizada por la ETHZ (disponibles para la comunidad robótica en `https://github.com/ethz-asl/mav_trajectory_generation`) debido a que alcanzaban una mayor eficacia en la generación de las mismas.

Adicionalmente, debido a que el módulo generador de trayectorias a corto plazo es el encargado también de generar las referencias del controlador ha sido necesario paralelizar la generación de las trayectorias y la evaluación de las mismas a lo largo del tiempo. Es por eso que se ha separado la computación en dos hilos diferentes: un hilo principal que se encarga de muestrear la trayectoria generada a una alta frecuencia (100 Hz) y de enviar las referencias al controlador y un hilo secundario que se encarga de generar la siguiente trayectoria a corto plazo a seguir.

Pequeña imagen de los hilos

Experimentos

Siguiendo la estructura planteada a lo largo del trabajo, los experimentos realizados se han dividido en dos partes:

1. **Comparación algoritmos de control.** Primeramente se comprobará la validez de los dos controladores propuestos evaluando su rendimiento siguiendo trayectorias simples a distintas velocidades. En estos experimentos únicamente se evalúa el rendimiento de los controladores.

2. **Recorrido del circuito de carreras**[1]**.** Después de validar el correcto funcionamiento de los controladores se emplea el mejor controlador para integrarlo junto con el resto de la arquitectura para intentar recorrer el circuito simulado propuesto. En estos experimentos se evalúa tanto el rendimiento de los controladores como el de los módulos de generación de trayectorias y de la arquitectura diseñada.

Para la elaboración de todos los experimentos se ha utilizado el entorno de simulación Flightgoogles [1] empleado para la realización de las pruebas clasificatorias virtuales de la competición Alphapilot 2019. Los mejores resultados obtenidos por los participantes de esta edición se encuentran en el artículo del mismo simulador. Se han empleado estos resultados para posicionar el rendimiento final conseguido entre los resultados obtenidos en esta competición.

Los experimentos han sido llevados a cabo en un equipo de sobremesa con Ubuntu 18.04 LTS empleando ROS Melodic. El equipo cuenta con un procesador Intel i7-9700K @ 3.60 GHz x 8, una memoria RAM con 32 GB de capacidad y una GPU Nvidia RTX2080 con 8GB de memoria dedicada.

A continuación se presentarán los resultados obtenidos en los distintos experimentos realizados.

[1]Estos resultados fueron presentados en el Workshop *Perception and Control for Fast and Agile Super-Vehicles* de la conferencia RSS20, integrando los módulos de percepción y estimación de estados desarrollados por Carlos Redondo Plaza

8.1. Comparación entre controladores

Se ha comparado el rendimiento de los dos controladores desarrollados mediante el seguimiento de una trayectoria circular de 5 metros de radio a distintas velocidades. En estos experimentos se emplea la arquitectura presentada anteriormente en el apartado (ref) empleando un generador de trayectorias simplificado, el cual genera la trayectoria con las ecuaciones parámetricas de una circunferencia en un plano bidimensional.

8.1.1. Controlador ángulos pequeños controladores

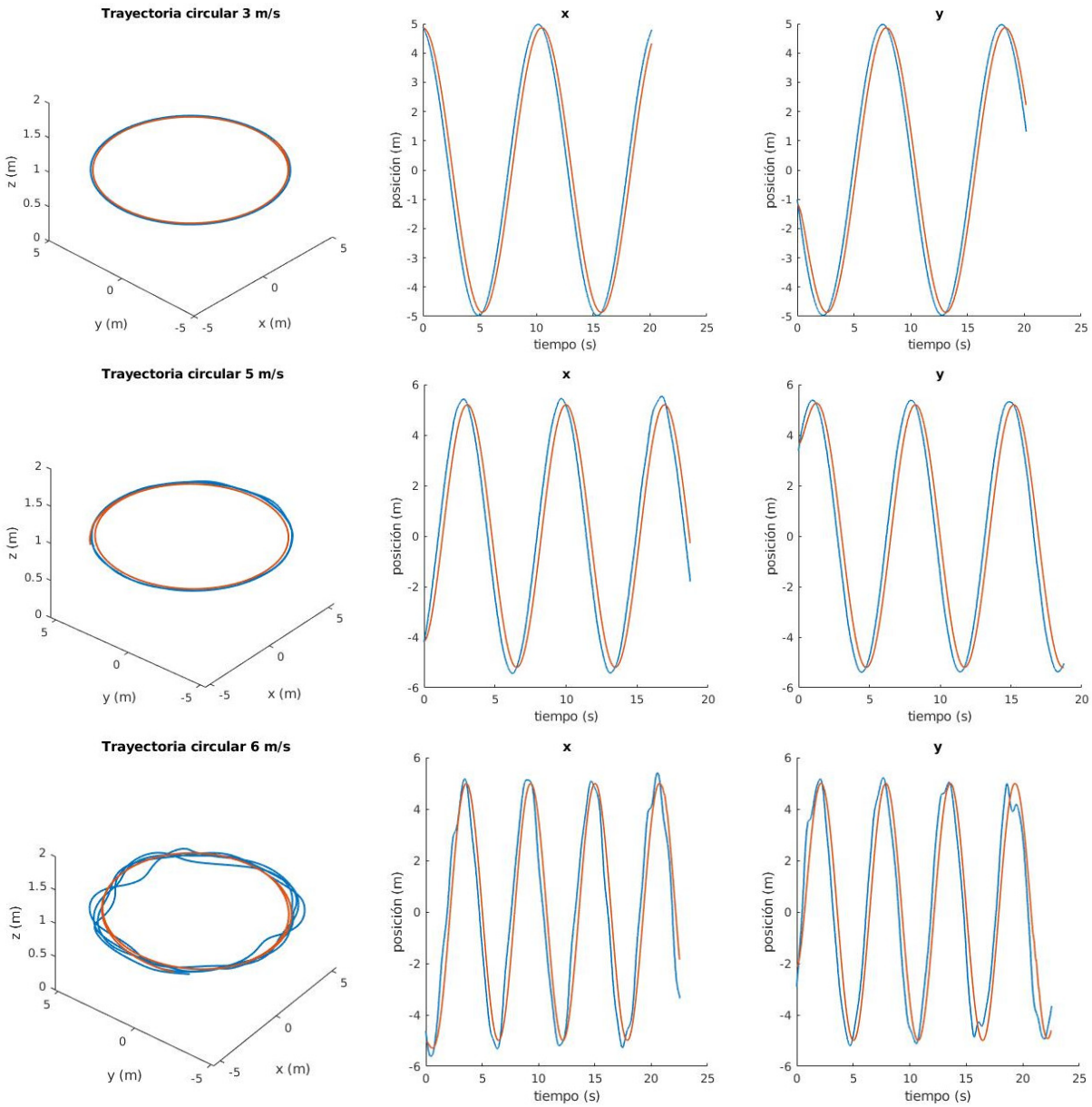

Figura 8.1: Trayectorias circulares a 3, 5 y 6 m/s empleando un controlador para ángulos pequeños. La trayectoria a seguir se muestra en rojo, mientras que la trayectoria seguida por el cuadricóptero se muestra en azul.

8.1.2. Controlador grandes ángulos

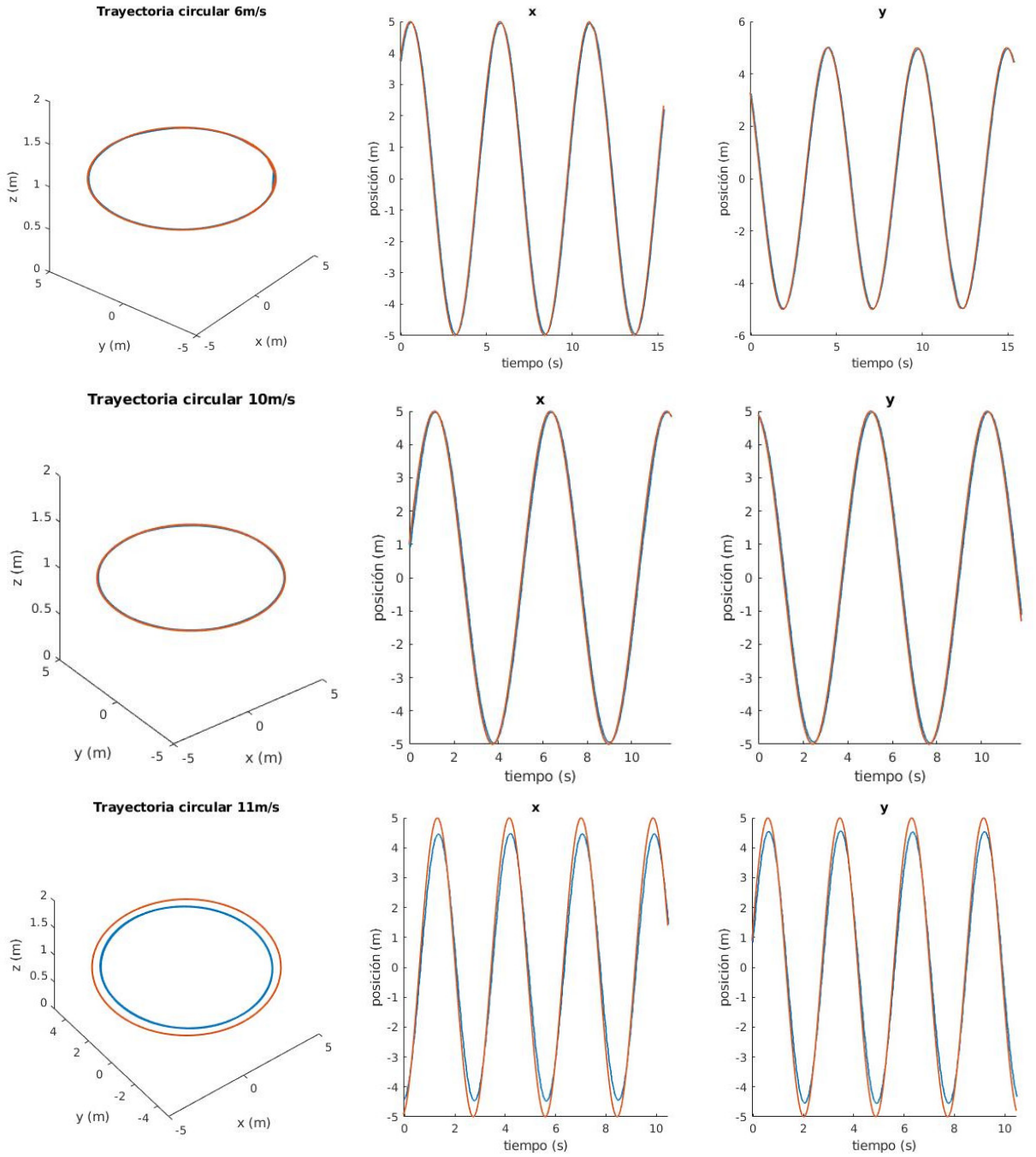

Figura 8.2: Trayectorias circulares a 6, 10 y 11 m/s empleando un controlador para ángulos grandes. La trayectoria a seguir se muestra en rojo, mientras que la trayectoria seguida por el cuadricóptero se muestra en azul.

8.2. Recorrido del circuito de carreras

Para realizar los experimentos del recorrido completo del circuito se han empleado las mismas condiciones impuestas en las pruebas clasificatorias del AlphaPilot 2019. En estas pruebas las posiciones aproximadas de las puertas son conocidas a priori por el sistema, así como los indicadores de cada puerta vista por las cámaras y las posiciones de las esquinas de las mismas en la imagen como se supuso en el apartado ref. En la arquitectura completa se ha optado por emplear el controlador para grande ángulos debido a que consigue alcanzar velocidades más altas de forma robusta.

Se ha recorrido el circuito con dos configuraciones distintas de las limitaciones dinámicas del sistema en la generación de trayectorias a corto plazo: Una configuración con valores más conservadores de velocidad y acceleración máximas y otra configuración con valores maś extremos de velocidad y aceleración máximas . A continuación se muestran los resultados obtenidos con cada configuración y las gráficas obtenidas durante el recorrido del circuito. Las trayectorias generadas mostradas en los gráficos corresponden a las referencias de posición obtenidas por el controlador a lo largo de todo recorrido, por lo que corresponden al muestreo de la trayectoria a corto plazo a lo largo del circuito.

8.2.1. Configuración de velocidad conservadora.

Los parámetros empleados durante la generación de trayectorias en este experimento son $A_{max} = 2g\ (m/s^2)$ y $V_{max} = 8\ (m/s)$. El tiempo necesario para completar el circuito completo fue de 32.93 s.

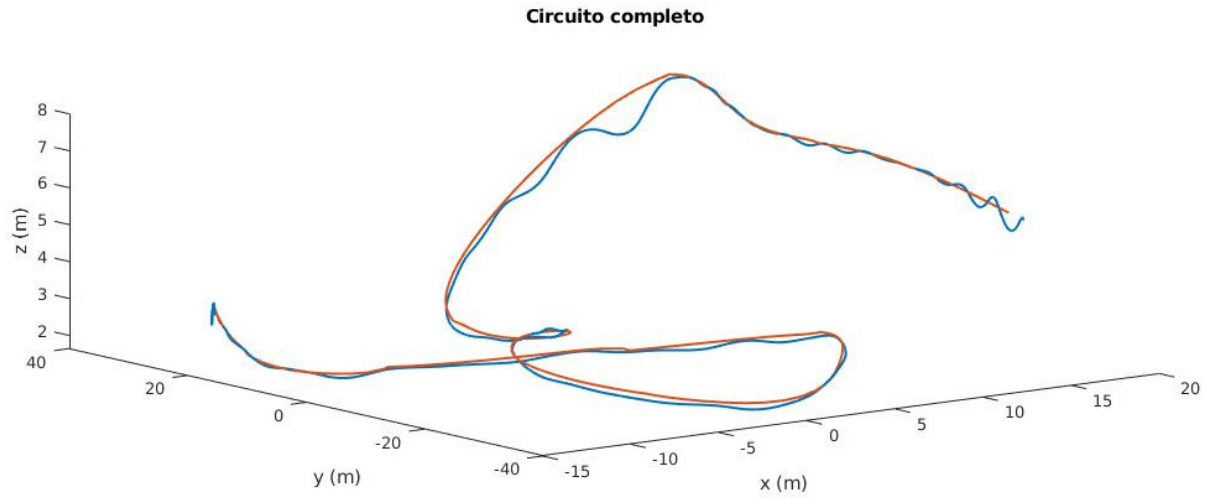

Figura 8.3: Trayectoria 3D planificada a lo largo del circuito (roja) y el recorrido realizado por el cuadricóptero (azul).

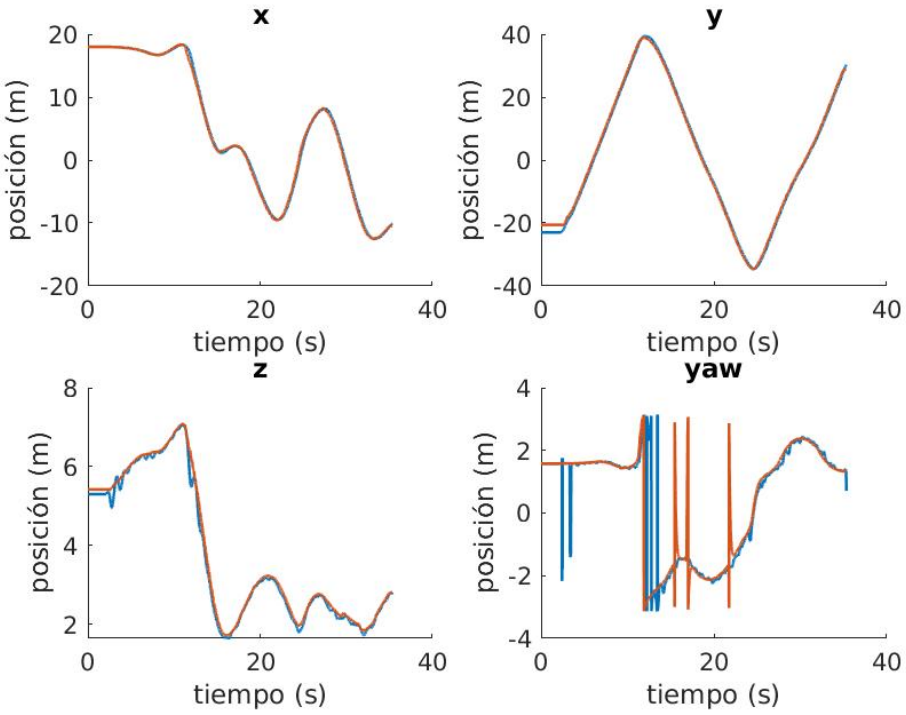

Figura 8.4: Trayectoria por coordenadas planificada a lo largo del circuito (roja) y el recorrido realizado por el cuadricóptero (azul).

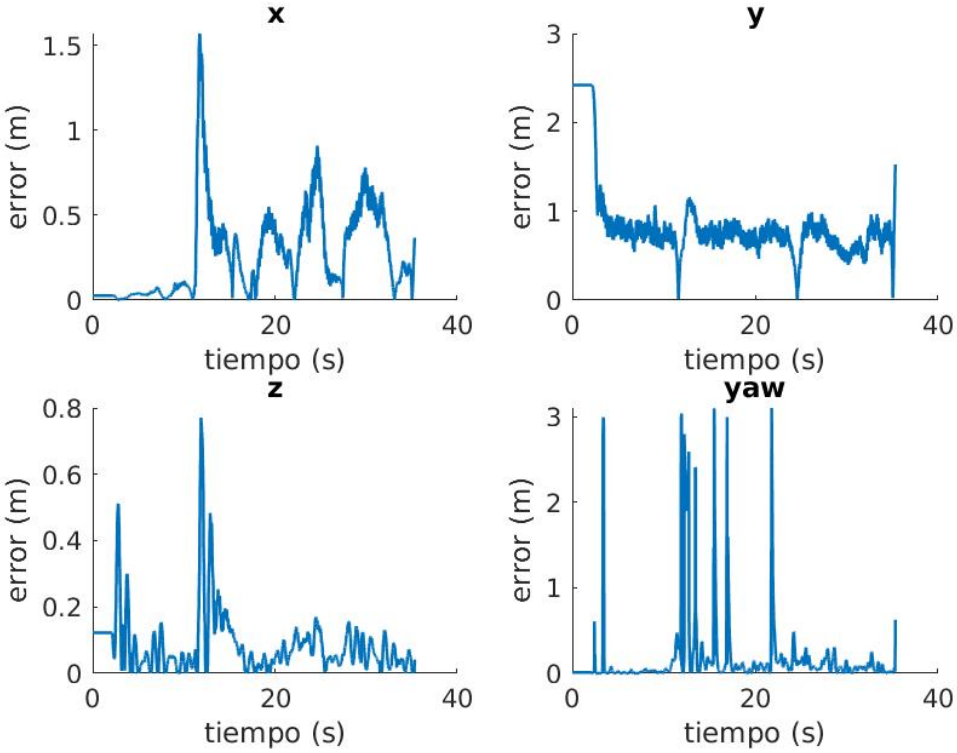

Figura 8.5: Errores de seguimiento entre la trayectoria planificada y la trayectoria seguida coordenada por coordenada.

De las gráficas obtenidas anteriormente se pueden obtener el valor medio y máximo de los errores de seguimiento en cada eje, así como las velocidades medias y máximas obtenidas durante el recorrido.

	Eje x	Eje y	Eje z
Error máximo (m)	1.5685	1.1577	0.7707
Error medio (m)	0.3143	0.7306	0.0796

Tabla 8.1: Errores de seguimiento durante el recorrido del circuito.

	Máxima	Media
Velocidad (m/s)	7.4119	5.7248

Tabla 8.2: Velocidad del cuadricóptero durante el recorrido del circuito.

8.2.2. Configuración de máxima velocidad.

Los parámetros empleados durante la generación de trayectorias en este experimento son $A_{max} = 4g\ (m/s^2)$ y $V_{max} = 12\ (m/s)$. El tiempo necesario para completar el circuito completo fue de 22.34 s.

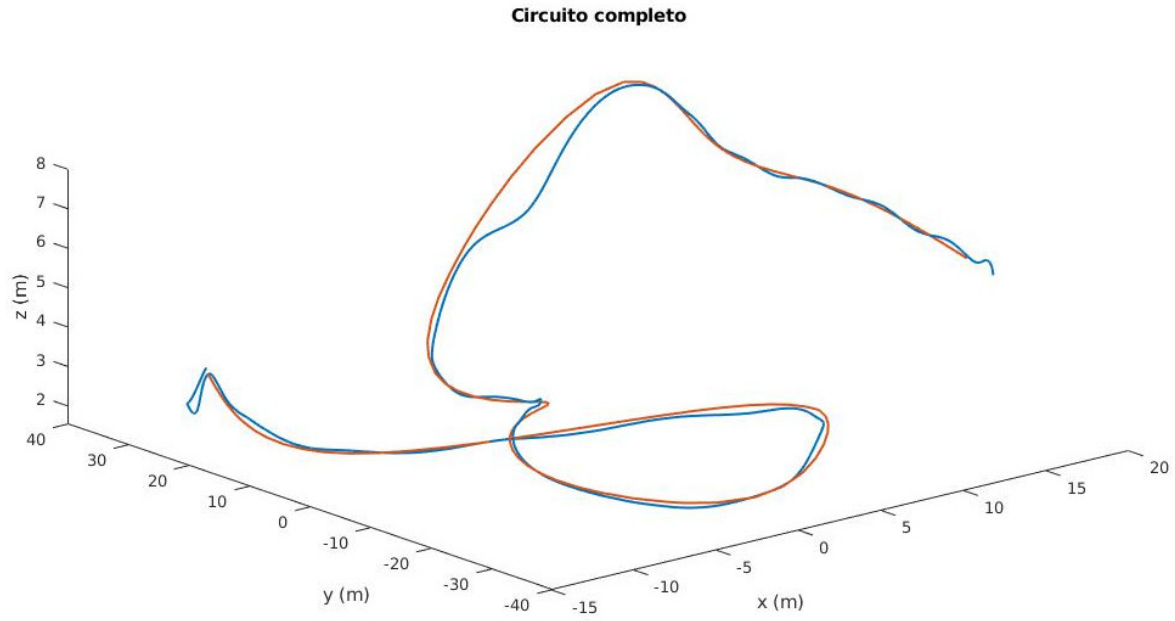

Figura 8.6: Trayectoria 3D planificada a lo largo del circuito (roja) y el recorrido realizado por el cuadricóptero (azul).

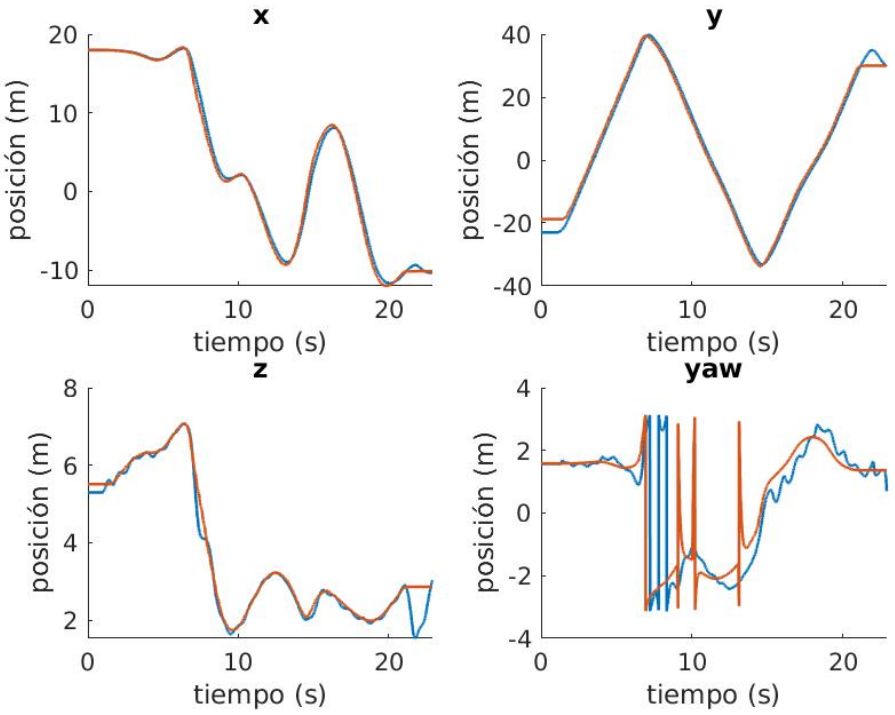

Figura 8.7: Trayectoria por coordenadas planificada a lo largo del circuito (roja) y el recorrido realizado por el cuadricóptero (azul).

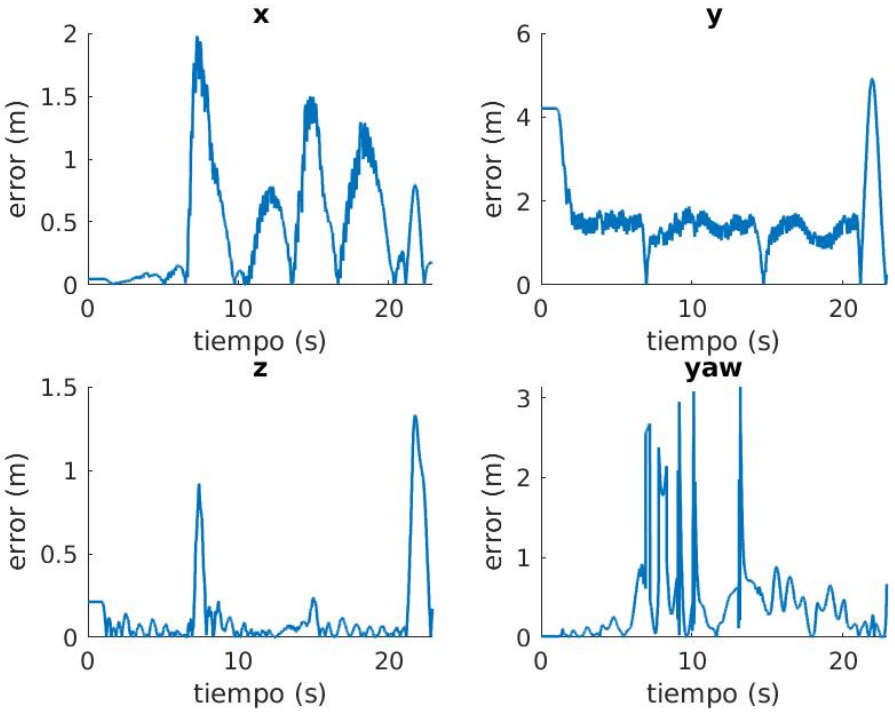

Figura 8.8: Errores de seguimiento entre la trayectoria planificada y la trayectoria seguida coordenada por coordenada.

De las gráficas obtenidas anteriormente se pueden obtener el valor medio y máximo de los errores de seguimiento en cada eje, así como las velocidades medias y máximas obtenidas durante el recorrido.

	Eje x	Eje y	Eje z
Error máximo (m)	1.9776	1.9181	1.3302
Error medio (m)	0.5973	1.4826	1.1413

Tabla 8.3: Errores de seguimiento durante el recorrido del circuito.

	Máxima	Media
Velocidad (m/s)	10.4948	9.3936

Tabla 8.4: Velocidad del cuadricóptero durante el recorrido del circuito.

Discusión de resultados.

A continuación se discutirán los resultados obtenidos en el capítulo anterior.

9.1. Comparación entre controladores

En las figuras 8.1 y 8.2 se puede observar como ambos controladores son capaces de seguir la trayectoria propuesta correctamente a bajas velocidades. Sin embargo, se puede observar como el error de seguimiento del controlador para ángulos pequeños aumenta a medida que se incrementa la velocidad, comenzando a perder la estabilidad al alcanzar velocidades de unos 6 m/s. Debido al tamaño de la circunferencia escogido, cuando el cuadricóptero se aproxima a estas velocidades los ángulos de roll ϕ y pitch θ comienzan a crecer, por lo que la linealización en torno al estado de hover comienza a fallar, provocando las oscilaciones que se pueden observar en la 3ª gráfica de la figura 8.1.

El controlador para grandes ángulos es capaz de sobrepasar esta velocidad sin problemas, llegando a velocidades de unos 10 m/s manteniendo un bajo error de seguimiento, como se puede observar en la figura 8.2. A partir de esta velocidad, se observa como el cuadricóptero comienza a alejarse de la trayectoria propuesta. Este fenómeno se puede deber a que se alcanza el empuje máximo que pueden dar los motores de la aeronave, lo que satura la señal de control de la aeronave, provocando que se llegue a una situación de equilibrio con un cierto error de seguimiento.

Con este análisis cualitativo se manifiestan los comportamientos esperados por ambos controladores: el controlador para ángulos pequeños se comporta bien a bajas velocidades, empeorando su rendimiento a medida que aumenta la inclinación de la aeronave, mientras que el controlador para grandes ángulos es capaz de volar a velocidades más elevadas de una forma más robusta. Es por esto que se decide emplear el controlador para grandes ángulos durante los experimentos del recorrido del circuito de carreras.

9.2. Recorrido del circuito de carreras

En los dos experimentos realizados con distintas configuraciones, se consigue recorrer el circuito entero de forma satisfactoria, atravesando las 11 puertas del circuito en el orden indicado sin colisionar con el entorno. Comparando las tablas **??** y 8.4, se observa que con la configuración más agresiva se consigue aumentar de forma considerable la velocidad media a la que el aeronave recorre el circuito, obteniendo un aumento de un 60 % respecto a la configuración más conservadora. Sin embargo, en las figuras 8.5 y 8.8 y en las tablas **??** y 8.3 se observa como el error de seguimiento también aumenta de forma considerable en la configuración más agresiva, llegando a duplicar el error medio en cada

eje. Este aumento hace que sea más arriesgado usar está configuración ya que se aumenta considerablemente el riesgo de colisión al atravesar una puerta, además las velocidades de vuelo más elevadas también empeoran la calidad de las estimaciones del módulo de percepción, lo que aumenta aún más el riesgo de colisión. Los errores de seguimiento más elevados se encuentran entre los cambios en las trayectorias a corto plazo, debido a la discontinuidad existente entre las referencias enviadas al controlador.

En ambos casos los módulos de generación de trayectorias son capaces de generar trayectorias a corto y largo plazo adecuadas y de actualizarlas a altas frecuencias con las nuevas mediciones obtenidas por el módulo de percepción, consiguiendo que la aeronave sea capaz de recorrer el circuito de forma satisfactoria a velocidades muy elevadas. Asimismo, el cuadricóptero tardo menos de 35 segundos en completar el circuito en ambos casos, lo que estaría dentro de los tiempos que consiguieron los 3 mejores obtenidos en las pruebas clasificatorias del AlphaPilot 2019 [1]. Sin embargo, a diferencia de las pruebas de la competición, en este trabajo se ha empleado la estimación de estado provista por el simulador, mientras que en las pruebas oficiales el módulo de percepción también debe ser implementado.

Conclusiones y trabajo futuro

10.1. Conclusiones

Durante el transcurso de este trabajo se ha desarrollado la arquitectura modular para un cuadricóptero de carreras autónomo. Esta modularidad de la arquitectura ha facilitado el desarrollo de los algoritmos de forma aislada así como el desarrollo de los experimentos de control, en los que se ha podido intercambiar los módulos de generación de trayectorias por módulos más sencillos de forma ágil. Esta modularidad también permitiría sustituir el módulo del simulador por un módulo de interfaz con una plataforma real, lo que facilitaría la realización de experimentos en real.

En cuanto al control del cuadricóptero se han conseguido implementar dos controladores del estado del arte de forma satisfactoria adaptándolos a las señales de control requeridas por el cuadricóptero. El empleo de herramientas como el *dynamic reconfigure* de ROS han permitido realizar el ajuste de las ganancias de los controladores *online*, lo que ha reducido considerablemente los tiempos necesarios para realizar estos ajustes.

Por otro lado, la decisión de separar la generación de trayectorias en dos partes, una a largo plazo y otra a largo plazo, ha permitido generar trayectorias de control óptimas en *snap* con una alta reactividad frente a los cambios en las estimaciones de las puertas, manteniendo un bajo coste computacional en la generación de ambas trayectorias.

Finalmente, se han validado los desarrollos e implementaciones realizados en un entorno de simulación fotorrealista, consiguiendo completar el circuito de carreras completo de forma satisfactoria con una velocidad máxima de vuelo de 10,5 m/s en un tiempo inferior a los 23 segundos, lo que se situaría dentro de los mejores obtenidos el año pasado durante las clasificatorias virtuales del AlphaPilot2019.

10.2. Trabajo futuro

El trabajo realizado deja una arquitectura modular que permite su ampliación con nuevos módulos para adecuarla a la realización de distintas tareas de forma sencilla. Para una utilización del sistema en un caso real sería necesario implementar los módulos de estimación y percepción del entorno de una forma integral, empleando únicamente las medidas obtenidas por los sensores de la aeronave.

Para mejorar el comportamiento del controlador sería conveniente emplear un controlador predictivo basado en el modelo (MPC), que permitiría reducir el error de seguimiento suavizando las acciones bruscas realizadas por el controlador cuando sufre cambios en la trayectoria de referencia. Junto con este controlador, se podrían emplear algoritmos de auto-identificación que permitan estimar *online* el valor de los parámetros dinámicos de

la aeronave, así como identificar posibles perturbaciones presentes en el entorno como podría ser el viento.

Finalmente, se podrían implementar métodos de optimización para calcular trayectorias que tengan en cuenta restricciones espaciales, lo que permitiría su empleo en entornos complicados con obstáculos alrededor.

Índice de figuras

Índice de tablas

Bibliografía

[1] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," *arXiv preprint arXiv:1905.11377*, 2019.

[2] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *arXiv preprint arXiv:2005.12813*, 2020.

[3] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 7410.

[4] B. Vanek, T. Peni, J. Bokor, and G. Balas, "Practical approach to real-time trajectory tracking of uav formations," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 122–127 vol. 1.

[5] A. B. Cruz, P. G. Mier, and J. C. Montaño, "Advanced uav trajectory generation planning and guidance." in *Aerial Vehicles*, T. M. Lam, Ed. In-TECH, January 2009, vol. 1, pp. 55–82. [Online]. Available: http://oa.upm.es/3492/

[6] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "Mpc with nonlinear $\mathcal{H}_\infty$ control for path tracking of a quad-rotor helicopter," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 8564 – 8569, 2008, 17th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016403277

[7] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 1649–1654.

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[9] S. Mallikarjunan, B. Nesbitt, E. Kharisov, E. Xargay, N. Hovakimyan, and C. Cao, "L1 adaptive controller for attitude control of multirotors," in *AIAA guidance, navigation, and control conference*, 2012, p. 4831.

[10] M. Kamel, M. Burri, and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," *ArXiv e-prints*, Nov. 2016.

[11] M. Faessler, D. Falanga, and D. Scaramuzza, "Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 476–482, Apr. 2017.

[12] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.

[13] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[14] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, "A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge," *Journal of Field Robotics*, vol. 35, no. 1, pp. 146–166, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21743

[15] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, no. 2, pp. 137–148, 2019.

[16] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: Optimal methods meet learning for drone racing," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 690–696.

[17] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Report for game of drones: A neurips 2019 competition."

[18] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2011, pp. 2688–2693.

[19] D. W. Mellinger, "Trajectory generation and control for quadrotors," 2012.

[20] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[21] M. Y. Amir and V. Abbass, "Modeling of quadrotor helicopter dynamics," in *2008 International Conference on Smart Manufacturing Application*. IEEE, 2008, pp. 100–105.

[22] P. Foehn and D. Scaramuzza, "Cpc: Complementary progress constraints for time-optimal quadrotor trajectories," 2020.

[23] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[24] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. de Croon, "Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone," *Journal of Field Robotics*, 2020.