Evasión de obstáculos apoyada en localización basada en simulación de instancias — Navegación autónoma en entornos cerrados.

1st Mario Bartolomé Manovel
Departamento de Ingeniería Civil
Universidad de Burgos
Burgos, España
mario.bartolome@gmail.com

2nd César I. García Osorio
Departamento de Ingeniería Civil
Universidad de Burgos
Burgos, España
cgosorio@ubu.es

3rd José-Francisco, Díez-Pastor
Departamento de Ingeniería Civil
Universidad de Burgos
Burgos, España
jfdpastor@ubu.es

4th Alejandro, Merino Gómez
Departamento de Ingeniería Electromecánica
Universidad de Burgos
Burgos, España
alejandromg@ubu.es

***Resumen*—Este artículo, detalla el uso dado a la implementación de dos algoritmos, con una gran componente estocástica, en la creación de un sistema de navegación en entornos cerrados.**

El primero, denominado Filtro de Partículas, e introducido en 1993 por N. Gordon, D. Salmond y A. Smith [5], proporciona un mecanismo de localización basado en filtros bayesianos recursivos, haciendo uso del método de Montecarlo. Se tratará de estimar la posición, realizando múltiples simulaciones del entorno en el que se desplaza el agente.

El segundo algoritmo, denominado *Vector Field Histogram*, Histograma de Campos Vectoriales, o *VFH* por su sigla en inglés, fue introducido por Borenstein y Koren en el año 1991 [2], [3], proporciona un método de evasión de obstáculos basado en el planeamiento, local, de la ruta a seguir por el agente.

Trabajando de forma conjunta, proporcionan un método de evasión de obstáculos y planeamiento de rutas tremendamente robusto.

***Index Terms*—drone, RaspberryPi, filtro de partículas, histograma de campos vectoriales, simulaciones Montecarlo**

I. Introducción

I-A. Motivación

Este artículo, es parte de un Trabajo de Final de Grado. La intención tras la implementación, y unión de los métodos a continuación descritos, es que formen parte de un sistema de control automatizado para un *drone*. El Filtro de Partículas fue elegido por su robustez y precisión, frente a sistemas incapaces de funcionar en entornos cerrados. El *VFH* fue elegido como solución a una serie de limitaciones dadas en el uso de algoritmos de Campos Potenciales[1] para la evasión de obstáculos. Lejos de quedarse *sólo* en la teoría, se ha construido un *drone* en el que instalar todo el hardware necesario.

[1]Estos no son adecuados para la navegación en corredores, o entornos altamente ocupados por obstáculos.

I-B. Estado del arte

Con el creciente uso de sistemas automatizados que ha presentado la industria en los últimos años, el nicho que los *drones* han pasado a ocupar, crece de forma rápida haciendo que se descarten métodos muy consolidados hasta el momento. Vigilancia, industria cinematográfica, ocio, deportes y mensajería son solo algunos de los ejemplos en los que los *drones* se van abriendo camino.

La industria parece moverse hacia sectores dedicados actividades en exterior, donde propuestas como la evasión de obstáculos, pueden ser algo más triviales o innecesarias:

- Sistemas de vigilancia de incendios, campos y cultivos.
- Sistemas de fumigación agrícola.
- Sistemas de inspección de zonas de difícil acceso.
- Sistemas de grabación a nivel profesional.

Sin embargo, parece estar obviándose el uso que se puede dar a un *drone* en un espacio cerrado, ya sea para acceder a zonas complejas en edificios o estructuras con gran cantidad de obstáculos, movimiento de mercancías ligeras en ciudades, en el ámbito militar, o vigilancia en general.

II. Estimación de la posición mediante simulaciones Montecarlo

El Filtro de Partículas es un algoritmo basado en simulaciones de Montecarlo, y algoritmos bioinspirados. En concreto, se trata de una ramificación de estos. Es un algoritmo de estimación de una distribución, ya que adopta la característica de remuestreo de una población, que presentan los algoritmos genéticos, pero no así otras, como la mutación o cruce.

De esta forma, estimará la posición y orientación de un agente en un entorno, haciendo uso de múltiples hipótesis

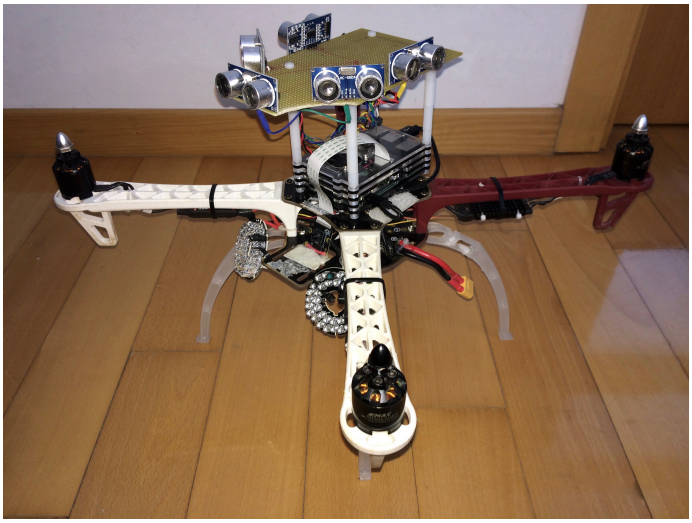

Figura 1. Detalle del *drone* construido, con los sensores de distancia instalados.

ponderadas [6], denominadas partículas[2]. Dicha ponderación, peso o probabilidad, se basa en la similitud de la hipótesis, o partícula, con la instancia que representa el agente.

Para lograr obtener información del entorno, y poder dar una serie de atributos con que poder comparar el agente y las particular que lo simulan, se debe hacer uso de sensores, como los mostrados en la Figura 1, que proporcionen la distancia del agente al obstáculo hacia el que se dirige el sensor.

Habitualmente, se comienza con una distribución de partículas aleatoriamente dispersas por el mapa y con una orientación definida, obviamente no se establecen partículas dentro de obstáculos. Esto, representa el total desconocimiento de la posición del agente, es decir, todas las hipótesis son equiprobables y por lo tanto la posición del agente es, equiprobablemente, cualquier posición del mapa.

A medida que el agente comienza a moverse, las partículas se mueven en la dirección y orientación que determina el agente. A continuación el agente y las partículas toman medidas de su entorno haciendo uso de los sensores de distancia, en el caso del agente, y de forma simulada en el caso de las partículas.

Una vez realizado el movimiento, la distribución de partículas es remuestreada en base a como de parecidas son las hipótesis al estado real del sistema. El remuestreo, o *resample*, conlleva reducir la población de partículas descartando las menos parecidas, o menos probables. De esta manera al cabo de cierto número de iteraciones la distribución converge hacia la posición real del agente.

II-A. Representación del estado

El *estado* del agente se puede representar mediante una n-tupla, donde $n = 3$ en este caso, tal que: $[x, y, \theta]$ siendo x e y las coordenadas de posición y θ la orientación. La altura del agente no es tenida en cuenta, dado que se establece esta como fija, es decir, no existen zonas en las que el agente deba navegar a menor o mayor altura para poder acceder, de manera que no es relevante para el cálculo de la posición.

El *peso* de las partículas, o estimación de la probabilidad de que la partícula sea la ubicación real del agente, es una función de densidad de probabilidad distribuida sobre el espacio de estados, descrito en [7]. En el *Filtro de Partículas* la variable de interés, la posición del agente, es representada por un conjunto de M partículas en el instante $t = k$, tal que $S^k = [e_j^k, w_j^k] : j = 1..M$ tal y como se muestra en [6]. Cada partícula tiene asignado un peso, w_j^k que define la contribución de esa partícula a la estimación de la posición.

Aquellas regiones del espacio de estados con una gran cantidad de partículas, se corresponden con zonas en las que hay una alta probabilidad de que el agente se encuentre en ellas. Aquellas zonas con una densidad de partículas baja, representan zonas en las que es improbable que el agente se encuentre.

El algoritmo asume la *propiedad de Markov*[3], de manera que solo funciona correctamente si el entorno es estático.

II-B. Actualización de movimiento

Durante la actualización de movimiento, descrita en [1], [6], el agente predice su nueva localización basándose en el movimiento, supuestamente, realizado. Sin embargo, la capacidad del agente de rotar o trasladarse no es perfecta. Si el sistema de evasión de obstáculos indica al agente que debe rotar nº para dirigirse hacia la meta por un camino seguro, el agente *tratará* de rotar nº, pero es muy probable que infra/sobrecorrija. Si el agente trata de desplazarse en línea recta, inevitablemente escorará en una dirección u otra.

El movimiento se define de la siguiente manera, para el instante de tiempo $t = k$, teniendo en cuenta el *ruido* existente en las medidas de los sensores y sistemas de movimiento:

$$\begin{bmatrix} x' \\ y' \\ \hat{\theta}' \end{bmatrix} = \begin{bmatrix} x + \rho \cdot cos(\hat{\theta}_k) \\ y + \rho \cdot sin(\hat{\theta}_k) \\ \hat{\theta}_k + f(rand|\mu, \sigma^2) \end{bmatrix} \quad (1)$$

Donde:

$[x, y, \hat{\theta}]^T$: Posición inicial del agente.
$\hat{\theta}$ = $\arctan \frac{\delta y}{\delta x}$
$[x', y', \hat{\theta}']^T$: Posición final del agente.
ρ = $\sqrt{\Delta x^2 + \Delta y^2} \cdot f(rand|\mu, \sigma^2)$: la distancia a recorrer **añadiendo ruido** dado por una distribución normal.
μ = 0,0: se desea un valor aleatorio dentro de una gaussiana con centro en 0,0.
σ: Errores de los sensores o sistemas de movimiento.

El ruido, como puede verse en la ecuación 1, se genera mediante una función de distribución normal, como la detallada en la ecuación 2, con centro en 0,0 de manera que se pueda simular el hecho de infra/sobrecorrejir el movimiento.

[2]Una *partícula* es el homólogo de un *individuo* en un algoritmo genético.

[3]En un proceso estocástico que cumple la propiedad de Markov, el estado futuro depende únicamente del estado presente, y no de los estados pasados.

II-C. Actualización de sensores

Cuando el agente toma medidas a través de sus sensores, actualiza la distribución de partículas de forma ponderada, tal y como se describe en [1], [6], de manera que aquellas que más afinidad presentan en sus mediciones, serán tenidas en cuenta más a menudo que aquellas que no se parezcan al agente.

Para ello, se calcula para cada partícula la probabilidad, p_j^k de ser la posición real del agente basándose en el estado que esta representa, y haciendo uso de una función de distribución normal, o Gaussiana:

$$p_j^{k+1} = f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

Donde:

μ = Distancias de la partícula a los diferentes obstáculos.
σ = Ruido del sistema de sensores.
x = Distancias reales obtenidas por los sensores del agente.

Cabe destacar, que la probabilidad p_j^{k+1} tiene como superíndice $k+1$, determinando este el instante de tiempo siguiente, es decir la probabilidad de la partícula j **después** de haber realizado el movimiento.

Se le asignará a cada partícula un peso, proporcional a la probabilidad de tratarse de la ubicación del agente:

$$w_j^{k+1} = w_j^k \cdot p_j^{k+1} \tag{3}$$

Dicho peso será normalizado de la forma habitual mediante:

$$w_j^{k+1} = \frac{\tilde{w}_j^k}{\sum_j^M \tilde{w}_j^k} \tag{4}$$

Donde $\tilde{w}_j^k$ es la medida de peso de una partícula j en el instante de tiempo k, antes de ser normalizado.

II-D. Remuestreo

Una vez que se han obtenido los pesos, el algoritmo deberá generar un nuevo conjunto de partículas a partir del anterior, teniendo en cuenta el peso de cada partícula. Es decir, si se deben generar N nuevas partículas, el algoritmo deberá escoger de entre el conjunto M de partículas, dando mayor prioridad a aquellas que tengan mayor peso. Cabe destacar, que el proceso de selección ha de ser con reemplazo, de manera que puede seleccionarse múltiples veces la misma partícula.

De esta forma se genera un nuevo conjunto de partículas M_{k+1}, en el que la mayor parte de las partículas se encontrará en aquellas posiciones que proporcionen medidas parecidas a las obtenidas por el agente real.

III. Evasión de obstáculos mediante planeamiento local de rutas

El *VFH* permite la detección de obstáculos y su evasión mediante el control de la dirección y velocidad del agente en tiempo real.

Para ello realiza una representación del entorno del agente en forma de histograma. Dicho entorno, llamado *Región Activa* $C*$, se representa como una *ventana* que se desplaza con el agente en el centro. Se trata un algoritmo de búsqueda local, sin embargo se ha mostrado que suele producir rutas cercanas al óptimo, aunque en el caso de este proyecto no es relevante, dado que la intención es que se exploren todas las zonas del entorno.

Se compone de tres partes:

1. Red Cartesiana de Histogramas: Derivado del concepto de *malla de certidumbre* propuesto por Moravec y Elfes en 1985, ver [4], en la universidad de Carnegie Mellon. Se construye un plano cartesiano representando el entorno cercano del agente. Por *cercano*, se entiende una distancia que los sensores de distancia puedan abarcar. Dicho plano será representado como un histograma, y de ahí su nombre habitual *Cartesian Histogram Grid*. Ver subsección III-A.
2. Histograma Polar: Una representación unidimensional, obtenida a partir de una reducción de la Red Cartesiana definida en el punto anterior. Se compone de n sectores angulares k, de anchura α, de forma que $n \cdot \alpha = 360; n \in \mathbb{Z}$. Dichos sectores k contienen un valor h_k que representa la densidad de obstáculos, *Polar Obstacle Density* o *POD*, contenida en él. Ver subsección III-B.
3. Capa de salida: Representa el resultado del algoritmo. A partir del POD, se obtienen los posibles *valles*[4] candidatos existentes entre obstáculos, y se elige el que más se aproxime a la dirección de la meta establecida. Así, devolverá la orientación necesaria a seguir por el agente. Ver subsección III-C.

III-A. Cartesian Histogram Grid

Se modela el espacio cercano al agente en forma de malla. En cada celda (i, j) de esa malla se encuentra un valor $c_{i,j} \in \mathbb{Z}$ que establece la certeza de la existencia de un obstáculo. La cantidad de celdas existentes en la malla se establece en función de los límites del sensor, y teniendo en cuenta que es necesario establecer una precisión c, por ejemplo: Si se dispone de un sensor con un rango de medidas de hasta 400 cm se puede crear una malla de 81×81, suponiendo que cada celda tenga un tamaño de $c = 5$ cm $\times$ 5 cm, en cuyo centro se encuentra el agente, dejando 40 filas y columnas a cada lado. Estableciendo así, un perímetro de medidas de 200 cm en cada dirección, con una precisión de $\pm$5cm.

Dicho valor se obtiene de la lectura proporcionada por el sensor de distancia, y se establece en el centro del ángulo de barrido sensor. La elevada eficiencia de este algoritmo se basa precisamente en incrementar el valor $c_{i,j}$ de únicamente una celda con cada medida. Dicha celda (i, j) se encuentra a una distancia de $R = \frac{\delta_{obsq}}{c}$ celdas y en la bisectriz del ángulo barrido por el sensor. Si bien está solución puede parecer una simplificación del problema, se llega a generar una distribución probabilística tomando múltiples medidas de forma continua mientras el agente se desplaza por el entorno. Por tanto, la misma celda, cercana a un obstáculo, y sus vecinas serán

[4] Así denominados por su representación en forma de histograma

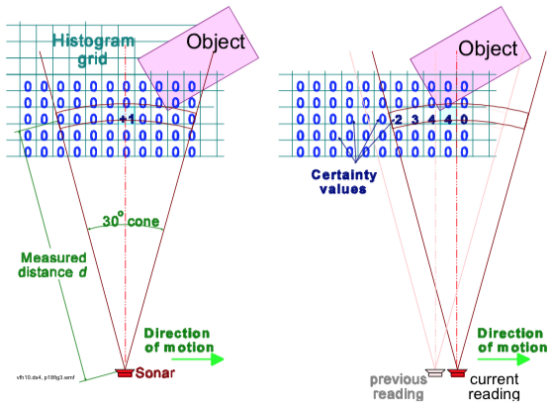

Figura 2. Distribución Histográmica de Probabilidades. Extraida de [3]

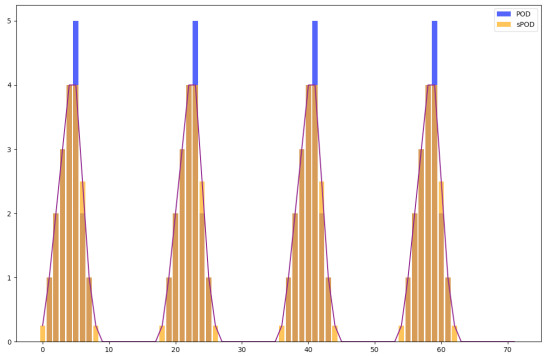

Figura 3. En azul Polar Obstacle Density (POD). En naranja el suavizado de este (sPOD).

incrementadas repetidas veces, tal y como se muestra en la Figura 2.

III-B. Polar Histogram

A partir de las medidas obtenidas en el Histogram Grid, se realiza la primera reducción de información para construir un Histograma Polar. Para ello, los contenidos de la región activa $C*$ son tratados como un *vector de obstáculos* cuya dirección está definida por la dirección β de la celda en cuestión al centro del agente *VCP*[5], y que viene definida por la función:

$$\beta_{i,j} = \tan^{-1} \frac{y_i - y_0}{x_i - x_0} \tag{5}$$

La ecuación 5 devuelve un valor $\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$rad.

Nótese, que para llevar a cabo la obtención de estos ángulos, se debería utilizar una función que tenga en cuenta el signo de las componentes del vector dirección $[y_i - y_0, x_i - x_0]$ resultante, de forma que devuelva el ángulo de giro, sea en sentido horario (valores positivos) o antihorario (valores negativos), más corto posible[6].

La magnitud de cada zona es entonces calculada de la siguiente forma:

$$m_{i,j} = c *_{i,j}^2 \cdot (a - bd_{i,j}) \tag{6}$$

Donde a representa la fuerza con que un obstáculo afecta al agente, y b representa la distancia desde la que un obstáculo afecta al agente, ambos positivos.

El plano es convertido en una secuencia de sectores que cubren los 360º. Se definen, por tanto, $n \in \mathbb{Z}$ sectores k de amplitud α, por ejemplo $n = 72; \alpha = 5$. Y se distribuyen las medidas tomadas anteriormente en los diferentes ángulos relativos al VCP del agente, en los sectores k_i correspondientes.

$$k = \frac{\beta_{i,j}}{\alpha} \tag{7}$$

[5]El VCP o *Vehicle Center Point* se define como el centro geométrico del vehículo. En nuestro caso, al tratarse de un *drone*, se considera el centro del mismo como VCP

[6]Obviamente es lo deseable, no sería eficiente realizar un giro de 225º en el sentido de las agujas del reloj, cuando uno de 135º en el sentido contrario a las agujas del reloj bastaría.

con un valor h_k, véase la ecuación 8, de densidad de obstáculos en ellos:

$$h_k = \sum m_{i,j} \tag{8}$$

Dada la discretización de los datos obtenidos del Polar Histogram el resultado, al realizar esta conversión y al obtener el sumatorio h_k de todos los valores pertenecientes a un sector k, puede parecer que el histograma varía de forma muy repentina entre dos sectores. Por ello se aplica una función de suavizado, según J. Borenstein y Y. Koren en [3]:

$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} + ... + lh_k + ... + 2h_{k+l-1} + h_{k+l}}{2l+1} \tag{9}$$

Donde l es una constante positiva a la que se le ha dado un valor de 5.

Sin embargo, existe una errata en el artículo publicado, y la ecuación 9 no es consistente, i.e: La función de suavizado parece poder expresarse de forma general tal que:

$$h'_k = \frac{1}{2l+1} \sum_{n=-l}^{l} (l - |n| + 1) \cdot h_{k+n} \tag{10}$$

ej: $l = 5$ $h'_k = \frac{1h_{k-5} + 2h_{k-4} + 3h_{k-3} + ... + \mathbf{6}h_k + ... + 2h_{k+l-1} + h_{k+l}}{2l+1}$

Como puede verse, el valor central $6h_k$ correspondiente a $n = 0$, no coincide con la posición dada para ese valor de n, donde el coeficiente que multiplica a h_k es igual a l en lugar de $l + 1$.

Por ello, se ha hecho uso de la función de suavizado de *Hann*, llamada así por el meteorólogo austriaco Julius van Hann. Se conoce a esta función por el nombre de Campana de Coseno, y se define por la ecuación 11.

$$w(n) = 0{,}5 - 0{,}5 \cos \frac{2\pi n}{M-1} 0 \leq n \leq M-1 \tag{11}$$

Donde M es el número de puntos en la ventana de salida. En este caso tomará el valor de l.

De esta forma, de obtener un histograma de n sectores con valores h_k, posiblemente dispares, en sectores k contiguos, se pasa a obtener un histograma suavizado, que será de mayor utilidad para el sistema de evasión de obstáculos. Véase la Figura 3.

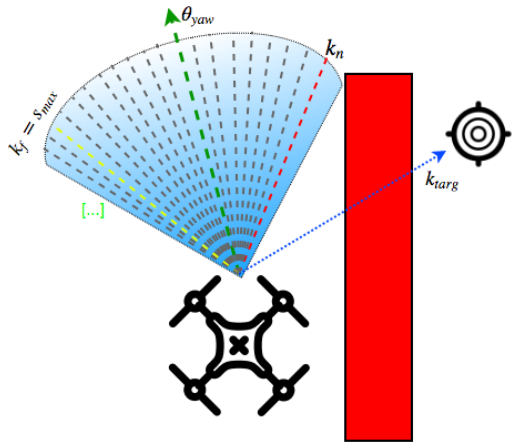

Figura 4. Valle ancho encontrado entre el obstáculo, a la derecha, y un espacio abierto, a la izquierda.

III-C. Output Layer

En el último estadio del algoritmo VFH, se computa el ángulo de giro θ que se debe aplicar al agente para dirigirlo, por una ruta segura, hacia el destino establecido. Una vez obtenido el *Polar Obstacle Density*, y aplicado el suavizado correspondiente, se puede determinar en que sectores k existe un *valle*. Se define un *valle candidato* como una zona, compuesta de varios sectores por los que es seguro navegar, es decir, están libres de obstáculos o su densidad de obstáculos está por debajo de cierto umbral.

Por lo general, se crean dos o más valles candidatos al analizar el entorno, de forma que es necesario elegir aquel que dirigirá al agente en dirección al destino k_{targ}. Una vez escogido el valle, se deberá seleccionar el sector k idóneo. Para ello se mide el tamaño del valle, es decir el número de sectores consecutivos por debajo del umbral que lo componen. De esta forma, se distinguen dos tipos de valles: amplios y estrechos. Los valles amplios son el resultado de espacios grandes entre obstáculos, o de situaciones en las que únicamente existe un obstáculo lo suficientemente cerca del agente. Por *grande* se define una constante s_{max} que determina el número de sectores k que componen un valle amplio.

El sector más cercano a k_{targ} y por debajo del umbral se denomina k_n, y representa el *borde cercano* del valle. El más lejano se denomina k_f, y en el caso de los valles amplios, coincide con el valor de $s_{max} + k_n$. Véase la Figura 4.

El sector que debe seguir el agente para dirigirse hacia k_{targ} sin peligro, se denomina θ_{yaw}, y su valor es la media de los valores de k_n y k_f, como puede verse en la ecuación 12.

$$\theta_{yaw} = \frac{k_n + k_f}{2} \tag{12}$$

Sin embargo, la ecuación 12 presenta un comportamiento errático cuando el destino se encuentra dentro de un valle, dado que se establece θ_{yaw} como la media entre los sectores cercano y lejano, al encontrarse la meta en un sector dentro del valle, el agente no se aproxima a ella, sino que realiza

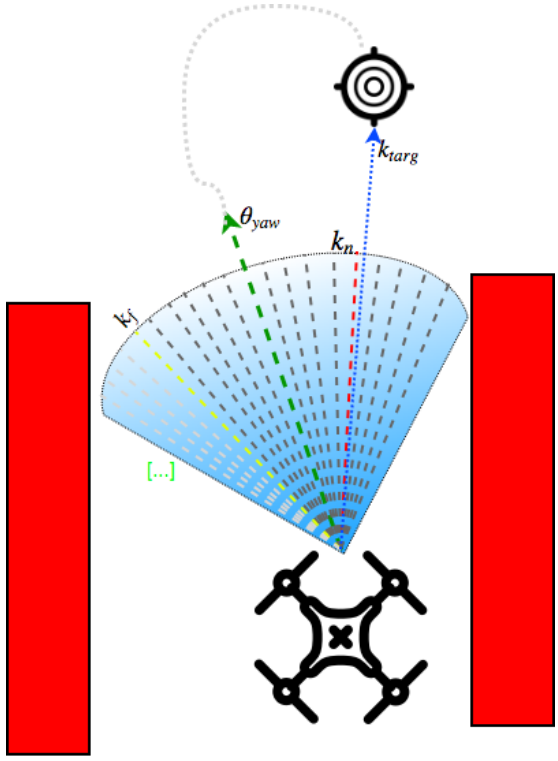

Figura 5. Movimiento circular al encontrarse k_{targ} en un sector seguro por ecuación 12

círculos cada vez más cerrados hasta llegar al destino, como puede verse en la Figura 5

La intención tras la ecuación 12 es la navegación segura en entornos con obstáculos cercanos, ya que aporta la cualidad de escoger la zona central entre dos obstáculos. Además en el momento en que la posición del agente sea intermedia entre dos obstáculos, la ecuación 12 dará como resultado el mismo sector, con lo que se consigue un movimiento rectilíneo.

Sin embargo, en el momento en el que el destino k_{targ} se encuentra en un sector k por el que es seguro navegar, no existe la necesidad escoger la zona intermedia del valle, sino que es deseable dirigirse hacia él directamente.

En el caso de *valles estrechos*, el comportamiento es prácticamente el mismo. Para este caso el valor del último sector con un POD por debajo del umbral escogido será $k_f < s_{max}$. Se usa la ecuación 12 para obtener el valor de θ_{yaw}, ilustrado en la Figura 6.

El cálculo de la velocidad, se puede derivar en el VFH, haciendo que este calcule la velocidad en base a la ocupación de un sector. Puede calcularse de la siguiente manera:

$$V = V_{\text{max}} \cdot \left(1 - \frac{h_c''}{h_m}\right) + V_{\text{min}} \tag{13}$$

Dónde:

$h_c'' = \min(h_c', h_m)$.
h_c' es la POD suavizada en el sector por el que navega el agente.
h_m es una constante establecida de forma empírica.

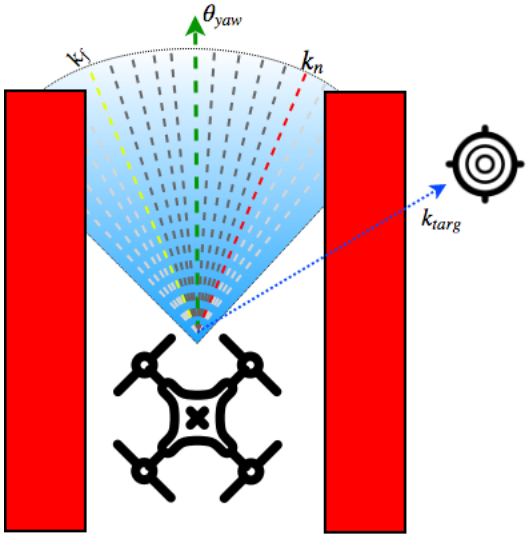

Figura 6. Valle estrecho encontrado entre dos obstáculos. La ecuación 12 proporciona un sector seguro ubicado en la zona central del valle.

V_{min} es la velocidad mínima del agente.
V_{max} es la velocidad máxima del agente.

III-C0a. Umbral: Se ha referenciado numerosas veces el *umbral* que parece definir un valle como candidato. Su definición se ha dejado para el final, debido a que el VFH presenta una gran robustez ante configuraciones probablemente erróneas. Aumentar o reducir el umbral, únicamente, afecta a la anchura de los valles candidatos cuando estos son estrechos. En el caso de los valles anchos sencillamente se aumenta o reduce la distancia existente entre el obstáculo y el agente.

Establecer un umbral muy alto, hace que el agente no sea consciente de la existencia de obstáculos y que se aproxime a ellos. Sin embargo, las repetidas medidas de un obstáculo en ese sector harán que el valor de certidumbre del *Polar Histogram* se eleve hasta superarlo, produciendo que el agente trate de variar su curso. De esta forma el agente se acercará mucho a los obstáculos, haciendo posible que llegue a colisionar, si la velocidad de aproximación es demasiado rápida.

Por el contrario, si el umbral es muy bajo, hará que los valles candidatos se vean reducidos, haciendo que el agente no pueda pasar por espacios estrechos.

IV. Resultados prácticos

El despliegue de estos algoritmos se ha realizado en una RaspberryPi 3B, haciendo uso de sensores de ultrasonidos HC-SR04, capaces de medir distancias de entre 5 cm y 400 cm con una precisión de ± 1 cm. Su tasa de adquisición de datos es, aproximadamente, 16Hz para un único sensor. Los sensores deberán obtener sus medidas de forma secuencial, para evitar la recepción de ecos de otros sensores, lo que hace que la tasa se reduzca todavía más, bajando hasta los 3Hz al utilizar 5 sensores. Esto les hace poco fiables para un entorno en producción. I.e., al proporcionar a un controlador PID el valor objetivo, sea la velocidad o la rotación del *drone*, calculado por el VFH, tan solo 3 veces por segundo, es **muy** probable que el movimiento se realice con información desactualizada. De hecho, hasta 16Hz se han mostrado insuficientes para llevar a cabo el control de la altitud en un sistema con una dinámica tan alta como es un *drone*.

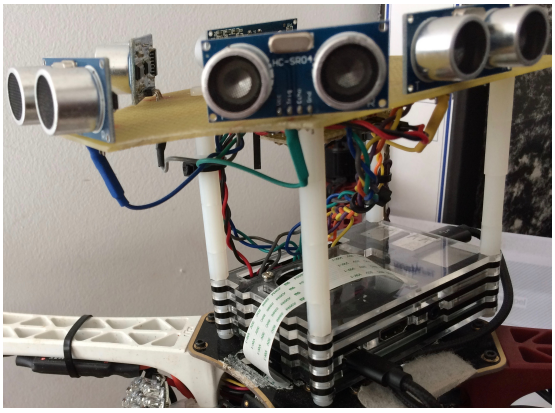

Figura 7. Detalle de los sensores y la RaspberryPi instalada en el *drone*.

V. Conclusiones y líneas futuras

El Filtro de Partículas, pese a ser computacionalmente exigente[7], provee de un medio para obtener la localización de un agente sin hacer uso de otras tecnologías menos fiables, como GPS, GLONASS o Galileo. Alimentar el *VFH* con la posición obtenida de un método como el Filtro de Partículas, puede ocasionar movimientos erráticos si se encuentran simetrías en el entorno, haciendo que la población de partículas se disperse en dichas áreas.

Se recomienda encarecidamente hacer uso de sensores con una tasa de adquisición de datos más rápida, como LIDAR o sensores LASER. De la teoría a la práctica hay, sin lugar a dudas, una ingente cantidad de parámetros que ajustar y personalizar en base a la dinámica del sistema elegido.

Referencias

[1] Domingo Gallardo López. Localización basada en filtros de partículas. *Universidad de Alicante*, 2010.

[2] Y. Koren J. Borenstein. Real-time obstacle avoidance for fast mobile robots in cluttered environments. *International Conference on Robotics and Automation*, 1990.

[3] Y. Koren J. Borenstein. The vector field histogram - fast obstacle avoidance for mobile robots. *Transactions on Robotics and Automation*, 7(3), 1991.

[4] Hans Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121, March 1985.

[5] Smith N. Gordon, J. Salmond. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2), 1993.

[6] Ioannis M. Rekleitis. A particle filter tutorial for mobile robot localization. *McGill University*, 2003.

[7] Daniel Sabinasz. Robot localization IV: The particle filter, 2017. [Internet; descargado 9-abril-2018].

[7] Si se dispone el plano como una matriz, en la que en cada posición i, j se establecen n partículas para cubrir la mayor cantidad de orientaciones posibles, y cada partícula dispone de r *rayos* que intersectan o obstáculos, dicha matriz no es tal, sino que pasa a ser un tensor tetra-dimensional, o teseracto.