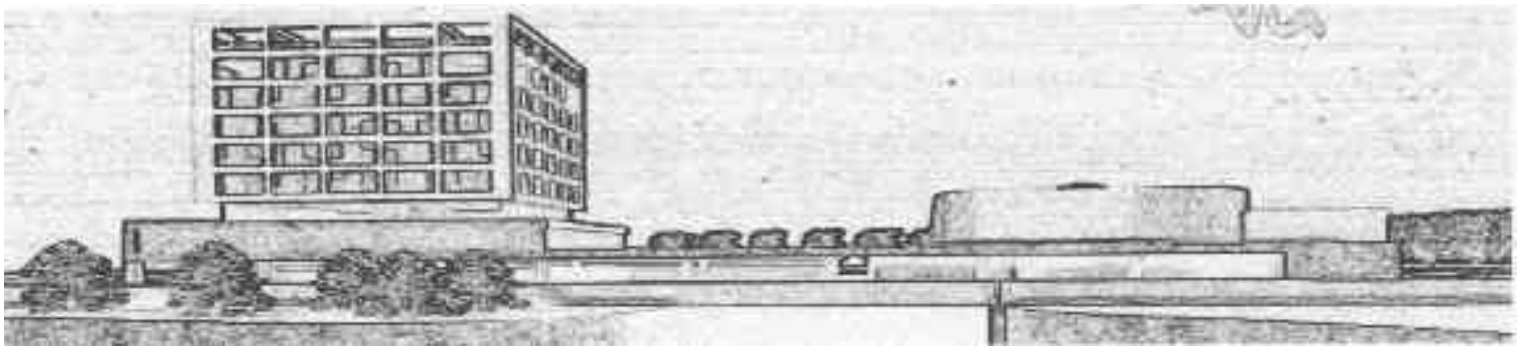
VOLUME IV, NUMBER 7



REPORTS ON SYSTEMS AND COMMUNICATIONS

SEGUIMIENTO TRIDIMENSIONAL USANDO DOS CÁMARAS

PABLO BARRERA GONZÁLEZ Y JOSÉ MARÍA CAÑAS

Móstoles (Madrid), Spain, Dic. 2004
Depósito Legal: 50653-2004
ISSN: 1698-7489

Índice

1. Introducción

Uno de los sentidos más importantes de los animales superiores y de las personas es la visión. Los ojos proporcionan muchísima información sobre el entorno. En contraste, el uso de cámaras en los robots móviles aún no se ha extendido masivamente. Esto se debe en parte a la complejidad conceptual de extraer la información visual del flujo de imágenes y en parte al alto coste computacional que conlleva el tratamiento de imágenes.

Tradicionalmente el campo de la visión artificial ha evolucionado aparte de la robótica móvil y la generación de comportamiento, concentrándose en problemas como la segmentación, el reconocimiento de formas, el filtrado de bordes, distintas transformadas y filtros, etc. No obstante, siempre ha recibido atención por parte de la comunidad robótica. Dentro de los sistemas deliberativos clásicos, la misión de la visión es la reconstrucción "objetiva" tridimensional y etiquetada del entorno [18]. Dentro de los sistemas basados en comportamientos, la visión se inserta como un sensor más [13] sobre el que se hacen cómputos específicos, en el marco de una percepción subjetiva y orientada a tarea.

Algunos usos típicos de la visión en los robots móviles incluyen el seguimiento visual [10], el reconocimiento de caras [1], la autolocalización [8] y la navegación por flujo óptico [9]. La mayoría de ellos utilizan un análisis bidimensional de imágenes. Los avances en geometría proyectiva [6, 4], pares estéreo, autocalibración, etc. han abierto la puerta a la extracción de información tridimensional de las imágenes. En este sentido, la localización y el seguimiento tridimensionales eficientes se muestran como tareas útiles, por ejemplo, en sistemas de seguridad, en la interacción hombre-máquina, en la percepción espacial para robots móviles, en domótica, etc.

Nuestro objetivo consiste en desarrollar un sistema de seguimiento tridimensional en tiempo real, empleando varias cámaras, con una cota de error del orden de centímetros. Con este trabajo no se persigue crear una herramienta comercial ajustada, sino hacer una primera aproximación a la visión tridimensional en un robot móvil, dejando sentadas las bases para un sistema de seguimiento más versátil y robusto. El algoritmo debe escalar a un número creciente de cámaras, funcionar con hardware no específico, y mantener un bajo coste computacional. Ese bajo consumo de cómputo permite mantener el tiempo real y resulta muy útil, por ejemplo, si se va a embarcar en un robot móvil, que tiene que atender a muchas otras tareas.

En esta línea, para poder abordar el problema hemos realizado varias asunciones iniciales sobre el sistema que reducen su complejidad. Primero, el seguimiento se realizará únicamente sobre *un objeto* en el espacio, de esta manera obviamos el problema de la correspondencia entre imágenes. Segundo, el objeto se *identifica por su color*, en vez de por movimiento o por su forma. Tercero, se emplearán *dos cámaras*, no necesariamente situadas como par estéreo y sin requerir la sincronización de las imágenes de ambas cámaras. Cuarto, supondremos que las cámaras están *fijas y con orientación constante*.

El resto de este artículo se ha organizado como sigue. En la segunda sección des-

cribimos las técnicas de retroproyección y triangulación utilizadas para obtener una estimación tridimensional de la posición instantánea del objeto. En la tercera parte extendemos el sistema con un filtro de partículas que fusiona las observaciones instantáneas y aprovecha la continuidad temporal para entregar estimaciones más fiables. En la cuarta sección se detalla la plataforma real donde se ha implementado el sistema y los resultados experimentales conseguidos. Dedicamos el quinto apartado a analizar los trabajos relacionados. Finalmente resumimos las conclusiones obtenidas y las líneas futuras por las que se puede mejorar el sistema.

2. Observación tridimensional por triangulación

Los sistemas de visión basados en una única cámara adolecen de indeterminación en cuanto a la distancia a la que se encuentran los objetos detectados. Dicha ambigüedad de profundidad se puede reducir añadiendo información al sistema sobre el tamaño de los objetos. Con ello se puede realizar una estimación de la distancia a la que se encuentran partiendo de su tamaño en la imagen. Otra posible forma de obtener la profundidad es emplear más de una cámara para observar la misma escena. En este caso no es necesario introducir información externa al sistema acerca del tamaño de los objetos. Al tener dos cámaras podemos realizar un proceso de triangulación detectando en ambas imágenes algún punto de interés de los objetos buscados.

Este planteamiento es el que sigue nuestro sistema, que se divide en dos partes separadas. En primer lugar se realiza una búsqueda por color del objeto en las imágenes, por ejemplo en los experimentos se ha buscado una pelota rosa. En segundo lugar, se proyectan las líneas de visión para cada cámara según un modelo geométrico. Tal y como muestra la figura 1, nuestro sistema estima que el objeto se encuentra en la posición tridimensional en la que intersectan estas rectas de proyección, o en su defecto su punto de cruce más cercano.

Aunque en las asunciones del problema a resolver hemos supuesto que sólo emplearíamos dos cámaras, el método usado es general, y escala sin problemas para emplear más cámaras. El sistema también requiere de cámaras calibradas, y conocer su posición y orientación, lo cual resulta imprescindible para deshacer correctamente la proyección.

2.1. Detección de objetos en cada imagen

Dado que vamos a emplear un sistema de triangulación, es necesario localizar un único punto representativo del objeto buscado. Para simplificar buscaremos un objeto de un color dado realizando un filtrado de la imagen en dicho color. Consideramos como punto representativo del objeto el centro de masas de todos los píxeles que pasan dicho filtro de color.

Al utilizar el color como única distinción es necesario hacer especial hincapié en las condiciones de iluminación de la escena. La mayor parte de las cámaras de bajo coste, como las que pretendemos utilizar, trabajan en los espacios de color RBG o

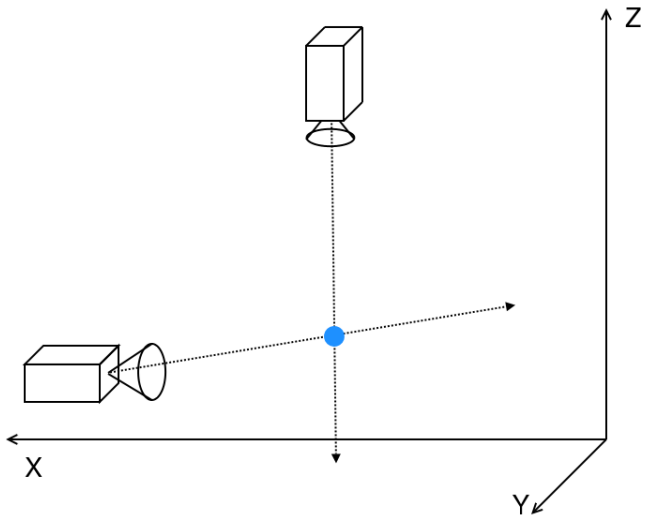


Figura 1: Cruce de las líneas de visión en el espacio tridimensional.

YUV. Ambos espacios utilizan componentes primarias de color pero no trabajan de manera directa con magnitudes relacionadas explícitamente con la iluminación. Por dicho motivo realizar un filtrado de color robusto frente a cambios de iluminación se torna complicado.

A diferencia de los modelos de color RGB o YUV, el espacio HSI (ver figura 3) separa la iluminación de la información de color. Las tres componentes que utiliza este modelo son:

1. el *tinte* (*Hue*), o color puro propiamente dicho, se corresponde con el ángulo del círculo de color de la figura 3;

2. la *saturación* (*Saturation*), muestra la viveza de un color determinado y se representa en la figura 3 como el radio dentro del cono; y

3. la *intensidad* (*Intensity*), que es la iluminación del color y se representa como la altura en el cono de color.

En este espacio HSI podemos realizar un filtrado más robusto ignorando la componente I, que sólo depende de la iluminación de la escena.

La relación existente entre RGB y HSI está resumida en las siguientes expresiones, donde R, G, B son la cantidad de componente roja, verde y azul, respectivamente, medidas entre 0 y 1. I nos da la intensidad luminosa, entre 0 y 1, H el ángulo de color y S, la saturación entre 0 y 1.

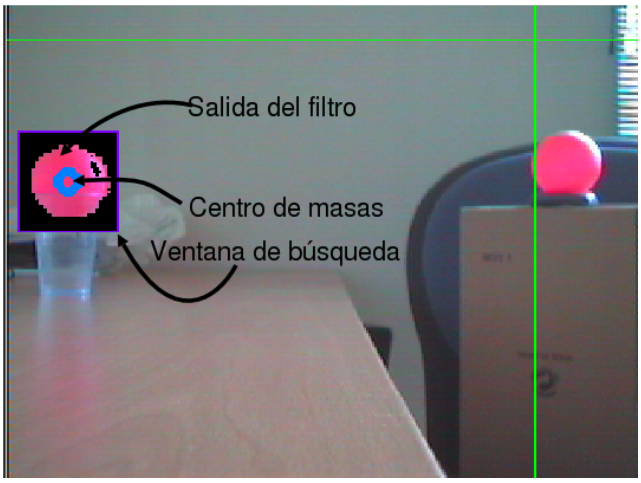


Figura 2: Filtro de color del objeto de interés y ventana de búsqueda.

$$H = cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \quad (1)$$

$$S = 1 - \frac{3}{R+G+B}min(R,G,B) \quad (2)$$

$$I = \frac{1}{3}(R+G+B) \quad (3)$$

Como ya se ha indicado las cámaras de bajo coste suelen trabajar en formato RGB o YUV por lo que será necesaria una conversión entre estos espacios de color y HSI para poder disfrutar de las ventajas de éste. Como veremos en la sección 4, esta conversión por software se muestra como una de las partes computacionalmente más costosas de todo el proceso. Además, las componentes de color H y S no son del todo independientes de la iluminación. Cuando los puntos están próximos al blanco o al negro (valores extremos de I), el rango de variación de la saturación baja, y la precisión es cada vez más pobre en cuanto al color (nótese el estrechamiento del círculo de color en la figura 3). Como vemos, la utilización de HSI no está exenta de limitaciones. Por este motivo se ha tomado en los experimentos una pelota de color rosa fuerte, para facilitar el proceso de detección.

El cálculo del centro del objeto se hace buscando el centro de masas de los píxeles que pasan el filtro de color. Esto se puede hacer obteniendo la media aritmética de los puntos de los puntos que pasan el filtro tanto para la coordenada x como la y. Una forma sencilla de obtener dicha media es empleando los *momentos de orden cero y orden uno* para calcularlo. Estos momentos pueden calcularse de la siguiente manera:

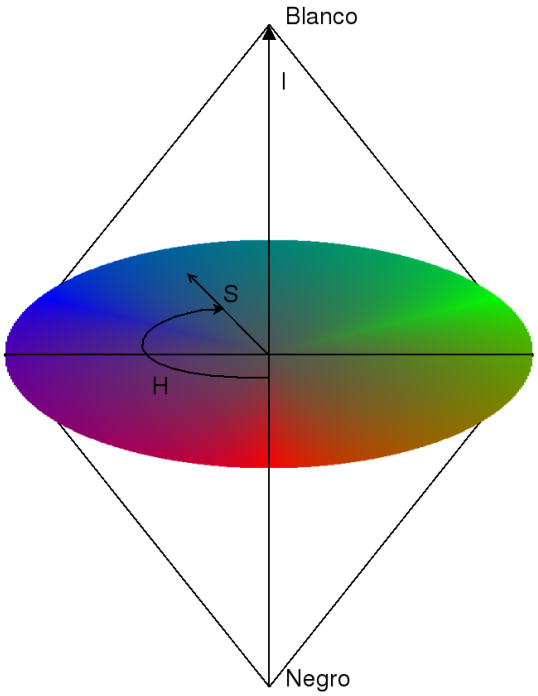


Figura 3: Espacio de color HSI.

$$M_{00} = \sum_u \sum_v I(u,v) \tag{4}$$

$$M_{10} = \sum_u \sum_v uI(u,v) \tag{5}$$

$$M_{01} = \sum_u \sum_v vI(u,v) \tag{6}$$

donde

$$I(u,v) = \begin{cases} 0, & \text{pixel } (u,v) \text{ no pasa el filtro} \\ 1, & \text{pixel } (u,v) \text{ pasa el filtro} \end{cases} \tag{7}$$

Las coordenadas del centro podemos obtenerlas dividiendo los momentos:

$$u_c = \frac{M_{10}}{M_{00}} \tag{8}$$

$$v_c = \frac{M_{01}}{M_{00}} \tag{9}$$

El problema del coste computacional de la conversión puede aliviarse ciñendo el filtrado de la imagen a una zona de interés. Aplicando un mecanismo de atención simple podemos limitar, y mucho, el número de píxeles sobre los que se realiza el filtro de color y por lo tanto conseguir trabajar a mayor número de fotogramas por segundo. Dicha atención puede centrarse en una *ventana de búsqueda* alrededor de

la última posición conocida del objeto en cada una de las imágenes. Cuanto menor sea el tamaño de dicha ventana mayor será el número de fotogramas procesados por segundo, al no analizar tal cantidad de píxeles por imagen. Para fijar el tamaño de la ventana debemos considerar la velocidad de seguimiento máxima que necesitamos. La ventana debe ser tal que abarque el recorrido máximo del objeto entre fotogramas. Obviamente la velocidad de proceso de imágenes influirá en el tamaño necesario de la ventana.

En caso de que el sistema de búsqueda pierda, en un momento dado, el objeto, tanto por un movimiento brusco imprevisto, un bloqueo en el campo de visión o cualquier otra circunstancia, realizará un filtrado de toda la imagen hasta que recupere la visión del objeto. Esto reducirá el número de fotogramas por segundo que pueden tratarse en ese momento, pero al encontrarnos en modo búsqueda dicha disminución no plantea un problema. Cuando se localice de nuevo el objeto, se volverá a fijar la *atención* en él.

2.2. Modelo de proyección

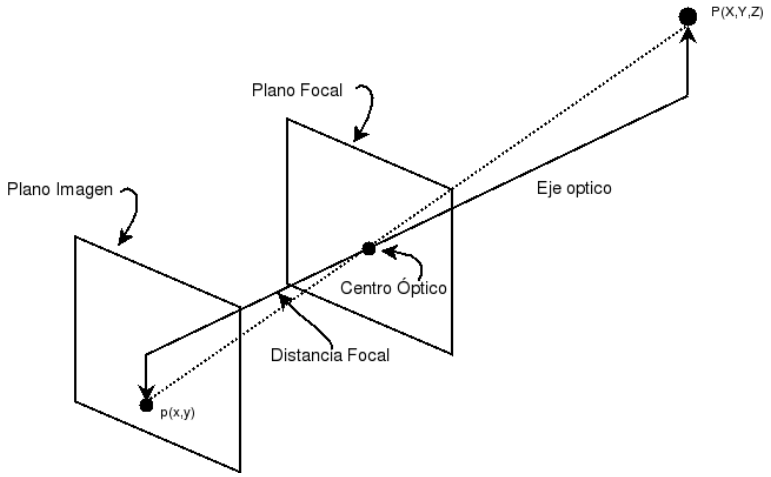
Una vez conseguidas las coordenadas del objeto en cada imagen, para obtener su posición tridimensional deshacemos la proyección de cada una de las cámaras, reconstruyendo la línea de proyección. Esta línea contiene todos los puntos tridimensionales que se proyectan en esas coordenadas en la imagen. Idealmente el objeto se encontrará en la intersección de las rectas de proyección de las dos cámaras (ver figura 1). Para obtener esta recta es necesario conocer tanto el modelo de proyección como la posición y orientación de la cámara en el espacio. Estos dos grupos son los llamados parámetros intrínsecos y extrínsecos, respectivamente, de la cámara. El modelo de cámara con el que trabajaremos es el modelo *pin-hole*. En él se asume que un punto tridimensional $P(x, y, z)$ se proyecta en el plano de imagen pasando a través de un único punto llamado *Centro óptico*. La recta que une el punto P y el centro óptico se llama línea de proyección e intersecta al *plano imagen* justo en el píxel $p(x', y')$ que es la proyección de $P(x, y, z)$. El centro óptico está situado a la *distancia focal* del plano imagen. Este modelo lo completan el *eje óptico*, que es una línea perpendicular al plano de imagen que atraviesa el centro óptico, y el *plano focal*, que es el plano perpendicular al eje óptico cuyos puntos no se proyectan en el plano de la imagen, incluyendo al centro óptico.

A partir de este modelo, aplicando el teorema de Tales, podemos obtener las ecuaciones de proyección fácilmente. Tomemos, sin perder generalidad, que el eje óptico se encuentra situado sobre el eje **Z**. En dicho caso la coordenadas del plano de imagen, que llamaremos x, y, se calculan en función de la posición tridimensional del objeto (X, Y, Z) y de las distancias focales en x e y, f_x, f_y, de la siguiente forma:

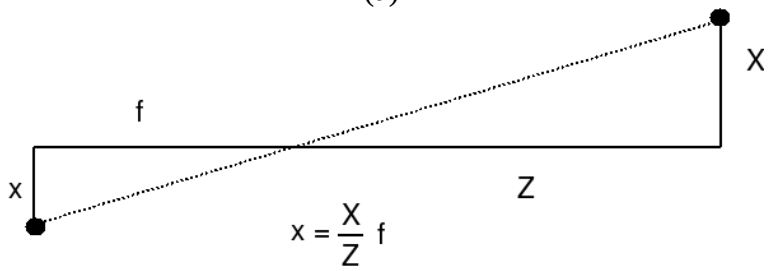
$$x = \frac{X}{Z} f_x \tag{10}$$

$$y = \frac{Y}{Z} f_y \tag{11}$$

Al trabajar con cámaras digitales estaremos atados a la utilización de unidades



(a)



(b)

Figura 4: Modelo de proyección *pin-hole*.

discretas, como píxeles a la hora de trabajar. Dichos píxeles tendrán un determinado tamaño, s_x y s_y, que en principio puede ser diferente para cada eje de la imagen. También debemos considerar que la posición de la cámara no está justo en el centro del plano de la imagen, por lo que se centrará en función de este punto (u_0, v_0). Con esto la relación entre las coordenadas de la imagen, medidas en píxeles, y las espaciales queda resumida a la expresión:

$$\begin{aligned} u &= \frac{X}{Z} f_x s_x + u_0 \\ v &= \frac{Y}{Z} f_y s_y + v_0 \end{aligned} \tag{12}$$

Expresando esta proyección en forma de matriz obtenemos la relación siguiente:

$$\frac{1}{Z}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x s_x & 0 & u_0 \\ 0 & f_y s_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{13}$$

Para aplicar estas expresiones es necesario conocer los parámetros intrínsecos de las cámaras o poseer un sistema de calibración que los proporcione. Los parámetros intrínsecos son los siguientes:

- $s_x f_x$, $s_y f_y$: los productos distancia focal por tamaño de pixel por cada eje, y
- u_0, v_0: posición del centro de la imagen en píxeles.

Si incluyeramos la distorsión el modelo sería más complejo y necesitaría algún parámetro adicional. Hemos eliminado deliberadamente la componente de distorsión, para ver qué resolución consigue el sistema utilizando un modelo de cámara sencillo. Como veremos en la sección 4, aún con esta suposición simplista y empleando cámaras no muy sofisticadas los resultados obtenidos son satisfactorios.

Esta función de proyección asume que la cámara está situada en el origen y orientada hacia el eje **Z**. En caso de querer tener otra orientación y posición es necesario aplicar una rotación y una traslación desde un sistema de referencia externo (X_w) a uno solidario a la cámara (X_c). La ecuación (14) expresa el cambio de base necesario en dos componentes, siendo T la posición de la cámara en coordenadas del mundo y R la matriz de cambio de base, entre coordenadas del mundo y coordenadas de la cámara.

$$X_c = R(X_w - T) \tag{14}$$

Reconstruir la línea de visión buscada consistirá en invertir el proceso de proyección y la rotación/translación. Para reconstruir la línea de visión es necesario conocer dos puntos por los que pase. El primero será el centro óptico de la cámara, que es igual al vector T. Para el otro punto podemos tomar cualquiera que cumpla las restricciones de las expresión (12), para las coordenadas u, v de nuestro objeto. Para simplificar puede tomarse $z = 1$, de este modo se obtienen las expresiones (15).

$$\begin{aligned} x &= \frac{(u - u_0)}{f_x s_x} \\ y &= \frac{(v - v_0)}{f_y s_y} \\ z &= 1 \end{aligned} \tag{15}$$

Una vez calculado el centro de masas del objeto en cada imagen, p_1 y p_2, se aplican las ecuaciones (15) para obtener las rectas de proyección r_1 y la recta r_2. El corte de las dos líneas de visión, o en su defecto, el punto más cercano de cruce, marcará la estimación del sistema de la posición del objeto en tres dimensiones.

3. Seguimiento temporal con filtro de partículas

Una estimación instantánea, como la empleada en la anterior sección, adolece de una serie de carencias al desechar la información que aporta la continuidad temporal. Por ejemplo, si en un determinado instante una de las cámaras deja de ver correctamente el objeto, la combinación de las medidas será complemente errónea, aunque el objeto permanezca exáctamente en la misma posición.

Para solucionar este problema podemos optar por distintos métodos de combinación de hipótesis, desde un sencillo filtro paso bajo, que tarde cierto tiempo en olvidar las últimas muestras, hasta un más elaborado filtro de Kalman[2], o un filtro de partículas, que intentarán mantener una representación interna de la distribución de la probabilidad de la posición del objeto en el mundo. En este trabajo hemos elegido un filtro de partículas, por su mayor generalidad y su buen funcionamiento con sistema complejos.

Los filtros de partículas [2, 14] se enmarcan dentro de las técnicas de estimación, en concreto de los filtros bayesianos, para problemas de seguimiento. En ese contexto, una de las técnicas más usadas son los Procesos de Decisión de Markov Parcialmente Observables (POMDP). En esos entornos el estado del sistema no es extraíble directamente, sino parcialmente a través de ciertas observaciones. Adicionalmente, el estado tiene su propia dinámica temporal. La estimación se expresa como una función densidad de probabilidad sobre el espacio de posibles estados. Los POMDP se han usado con éxito, por ejemplo, en problemas de autolocalización de robots, donde el estado del sistema es la posición del robot [12]. Sin embargo los POMDP no escalan cuando el espacio de posibles estados es desorbitadamente grande, porque entonces el coste computacional de mantener una estimacion de probabilidad para todos los estados se hace inmanejable. Recientemente se han explorado técnicas de *MonteCarlo* para muestrear esa distribución de probabilidad manteniendo la representatividad y disminuyendo significativamente el coste computacional [16]. Los filtros de partículas son métodos secuenciales de *MonteCarlo* y también se pueden ver como una generalización de los tradicionales filtros de Kalman para problemas no gaussianos y multimodales.

Los filtros de partículas mantienen una representación muestreada de la función de densidad de probabilidad de los estados. Cada muestra es una *partícula*, un punto en el espacio de estados, con una determinada masa en función de su significatividad. La estimación del filtro, y por lo tanto su colección de partículas va cambiando con el paso del tiempo (materializando el modelo de evolución del sistema) y con la incorporación de nuevas observaciones parciales del estado (modelo de observaciones).

En nuestra aplicación el estado es la posición (y velocidad) del objeto que está siendo seguido, y lo llamaremos vector x_t. La dinámica propia del sistema se captura en un modelo de evolución, que se expresa como la probabilidad de cambio de estado $p(x_t|x_{t-1})$. La información que las observaciones parciales z_t aportan a la estimación del estado se puede representar también probabilísticamente como $p(z_t|x_t)$, que se puede interpretar como verosimilitud del estado a la luz de la observación sensorial.

El modelo de evolución del sistema coincide en nuestro caso con el modelo de movimiento propio del objeto que está siendo seguido. Entre las asunciones no incluimos ninguna limitación a los movimientos que el objeto a seguir puede realizar. Por este motivo el modelo tiene simetría esférica: el movimiento no tiene ninguna dirección privilegiada, es isótropo. Además, dado un intervalo temporal (el que hay entre dos fotogramas, por ejemplo) al considerar una velocidad variable has-

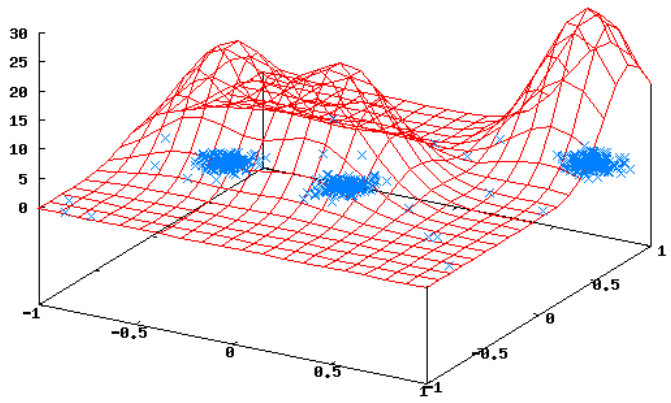


Figura 5: Muestro de una función de densidad de probabilidad empleando partículas.

ta un cierto máximo, los movimientos largos son menos frecuentes que los cortos. Los movimientos se toman siempre relativos a la posición actual de la partícula, siguiendo un esquema incremental.

Hemos modelado esa evolución como una gaussiana tridimensional, simétrica y de media nula, que se suma a la posición en cada estado (ecuación (16)). Los ejes son totalmente independientes y con la misma probabilidad de movimiento en cada uno de ellos. Uno de los parámetros modulables del filtro de partículas es σ_e (ecuación 17) que representa la amplitud de la gaussiana y que representa los tipos de movimientos esperables del objeto.

$$p(x_t|x_{t-1}) = \frac{1}{(2\pi)^{d/2}|\Sigma_e|^{1/2}} \exp\left\{-\frac{1}{2}(x_t - x_{t-1})^T \Sigma_e^{-1}(x_t - x_{t-1})\right\} \tag{16}$$

donde d es la dimensión del espacio de estados y $|\Sigma_E|$ es el determinante de la matriz de covarianza, que también puede expresarse de la siguiente forma:

$$\Sigma_e = \begin{pmatrix} \sigma_e & 0 & 0 \\ 0 & \sigma_e & 0 \\ 0 & 0 & \sigma_e \end{pmatrix} \tag{17}$$

El *modelo de observación* especifica la probabilidad de una observación z_t cuando el sistema se encuentra en el estado x_t. En nuestro caso la observación será directamente la estimación instantánea tridimensional de la posición del objeto descrita en la sección 2.1.

Asumimos que las observaciones tridimensionales conseguidas con el anterior proceso de localización (sección 2.2) son correctas salvo un cierto ruido gaussiano aditivo, causado por las pequeñas imprecisiones. Tal y como expresa la ecuación (18), modelamos la probabilidad de observación como otra gaussiana tridimensional, de media nula. El ruido nuevamente es isótropo, por lo que la matriz de covarianza Σ_m será diagonal y los elementos de la diagonal principal serán iguales a σ_m, la desviación típica del error aditivo, el segundo parámetro de nuestro filtro de partículas.

$$p(z_t|x_t) = \frac{1}{(2\pi)^{d/2}|\Sigma_m|^{1/2}} \exp\left\{-\frac{1}{2}(z_t - x_t)^T \Sigma_m^{-1}(z_t - x_t)\right\} \tag{18}$$

El último de los parámetros del filtro es el número de partículas empleado, N_s. Es de esperar que valore mayores permitirán una mayor precisión, pues la función verosimilitud del estado al tener un mayor número de muestras. El contrapunto será un mayor coste computacional.

El filtro de partículas es un algoritmo iterativo y comienza con una población de partículas aleatorias, todas ellas equiprobables, distribuidas uniformemente sobre el espacio de búsqueda. En cada iteración hay dos pasos, uno de evolución, materializando la dinámica propia de las partículas, y otro paso de observación ajustando los pesos de las partículas en función de la observación.

La generación de la población de partículas para la iteración siguiente se realiza muestreando aleatoriamente entre las actuales, de manera que las que acumulan más probabilidad tengan proporcionalmente más opciones de propagarse a la siguiente población. Esto se ha implementado muestreando uniformemente en el eje Y de la figura 6, que tiene la probabilidad acumulada de 0 a 1, y seleccionando como partícula para la siguiente población la correspondiente en el eje X.

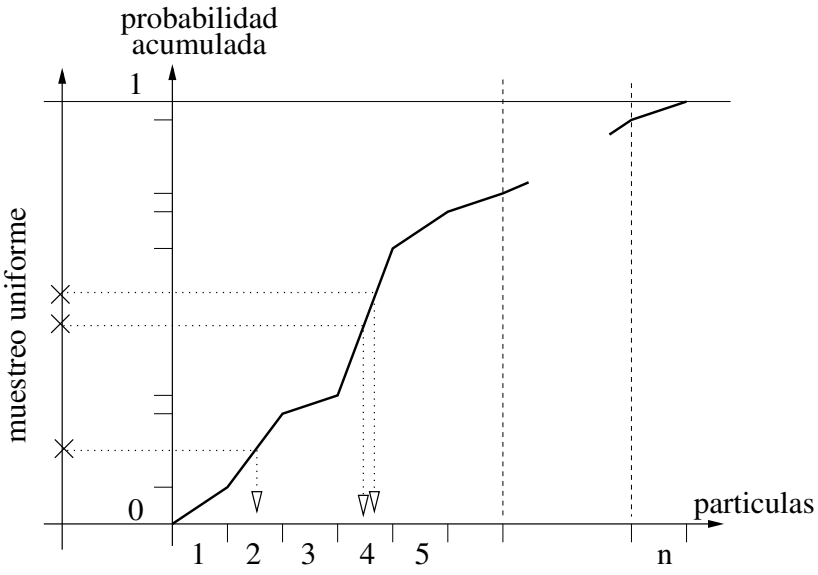


Figura 6: Remuestreo entre las partículas para generar la siguiente población.

Para obtener la mejor estimación en cada instante puede usarse la media ponderada de las partículas. De esta forma se consigue una aproximación a la media de la función de probabilidad que estamos aproximando, $p(x_t|z_t)$. Las partículas con bajo peso (por lo tanto menos relacionadas con la medida) influirán menos que las que acumulan más peso al usar una media ponderada.

- Inicialización del sistema. Para $n = 1, \ldots, N_s$
  - s_{old}: Partículas del instante anterior: $s^n_{old} = rand()$
  - s_{new}: Partículas del estado actual: $s^n_{new} = s_{old}$
  - π^n: Peso inicial de las partículas: $\pi^n = \frac{1}{N_s}$
- Se itera para cada instante de tiempo

1. La población nueva pasa a antigua $s_{old} \leftarrow s_{new}$
2. Se generan n nuevas partículas remuestreando
   a) Muestreo proporcional: $r = rand(),\ r \in [0, 1]$
   b) Muestreo proporcional: se busca, por subdivisión binaria, el menor j que cumpla $c^j_{old} \geq r$
   c) Se selecciona la partícula j para añadirla a la siguiente generación, $s' = s^j_{old}$.
   d) Añade, a la partícula seleccionada, innovación según el modelo de evolución gaussiano $s^n_{new} = s' + randn(0, \sigma_e)$
   e) Se calcula la probabilidad de la medida en este instante, z, en función de la nueva partícula:

$$\pi^n = p(z|x = s^n_{new}) = \frac{1}{\sqrt{(2\pi\sigma^2_m)}} \exp\left\{-\frac{(z - s^n_{new})^T(z - s^n_{new})}{2\sigma^2_m}\right\}$$

3. Se normaliza los pesos de las muestras para que $\sum_n \pi^n = 1$ y se calculan los pesos acumulados: $c^n_{new} = c^{n-1}_{new} + \pi^n$
4. La estimación de $\hat{x}$ se obtiene con la media de la densidad de probabilidad muestreada por las partículas: $\hat{x}_t = \sum_{n=1}^{N_s} \pi^n s^n_{new}$

4. Resultados experimentales

Para validar experimentalmente el método de seguimiento se ha realizado un montaje de prueba y se ha programado una implementación del *software* necesario.

Sobre ese prototipo se han hecho tres conjuntos de pruebas para comprobar el funcionamiento del sistema. En primer lugar, se ha medido el error absoluto cometido en la estimación de la posición del objeto, estando éste fijo en una serie de puntos. De esta manera hemos caracterizado la precisión conseguida. En segundo

lugar, se ha analizado la eficiencia temporal del programa, identificando las partes de mayor coste computacional. Por último, se ha estudiado la dinámica de la población de partículas para comprender el funcionamiento del filtro y ajustar sus parámetros internos.

A continuación se describen tanto el montaje como los resultados obtenidos.

4.1. Montaje de pruebas

El prototipo construido consta de un par de cámaras y el software necesario para procesar las imágenes obtenidas por ellas. Las cámaras empleadas son de tecnología *CMOS* de bajo coste, de las que normalmente se utilizan en vídeoconferencia. En particular hemos empleado dos cámaras de Sun[1] conectadas a la entrada *S-Video* del PC. El PC incluido en el prototipo tiene un procesador Intel(R) Pentium(R) IV CPU y 1.70 GHz, y ejecuta un sistema operativo *Debian GNU/Linux*.

En cuanto a la infraestructura software, hemos utilizado un servidor de imágenes ya existente en nuestro laboratorio que se encarga de capturar las imágenes utilizando *video4linux* y enviárselas a sus clientes por la red empleando el protocolo TCP/IP. Este servidor independiza las aplicaciones de las particularidades del proceso de captura de imágenes y permite mayor facilidad de posicionamiento de las cámaras, ya que el ordenador de captura y el de procesamiento pueden ser diferentes. Dicho servidor, llamado *Óculo*, ya ha sido empleado con anterioridad en otros desarrollos dentro del mismo laboratorio [19]. Ofrece imágenes de 320x240 píxeles al ritmo que le vaya pidiendo la aplicación del cliente, y con un límite experimental de unos 12 fps.

Para que el sistema funcione es necesario adoptar un sistema de referencia para expresar en él la posición tridimensional de las cámaras y los objetos a seguir. Según muestra la figura 7, se ha tomado como origen de coordenadas el rincón de la habitación, que es el punto formado por la unión de las dos paredes que se aprecian en la imagen y el suelo. Los ejes de coordenadas los marcan las aristas de las paredes, según se observa en esa figura.

Este sistema de referencia tomado es un convenio totalmente arbitrario y sólo se emplea para anclar el sistema al mundo real y así realizar medidas contrastadas.

Para establecer la línea de retroproyección descrita en la sección 2.2 fue necesario realizar una calibración de las cámaras, obtiendo sus parámetros intrínsecos. Dicha calibración se realizó empleando software basado en la biblioteca de visión artificial OpenCV[2]. El proceso realizado adolece de ciertas simplificaciones por lo que los resultados obtenidos son aproximados. La matriz de proyección obtenida, siguiendo el formato de la ecuación (13), es la siguiente:

$$\begin{pmatrix} 415{,}93 & 0 & 162{,}45 \\ 0 & 410{,}76 & 111{,}44 \\ 0 & 0 & 1 \end{pmatrix} \tag{19}$$

[1] Sun Camera II, modelo IK-M28SE

[2] http://www.intel.com/research/mrl/research/opencv

Figura 7: Montaje experimental en el laboratorio.

Tal y como muestra la figura 7, las cámaras se han situado ortogonales entre sí, separadas unos 2,5 metros, y cada una aproximadamente perpendicular a uno de los planos de proyección. La cámara cenital se encuentra orientada hacia el plano XY, y la cámara lateral (a la izquierda de la imagen 7), está orientada hacia el plano YZ. El volumen cubierto por el campo de visión de ambas cámaras viene a ser de unos 3 metros cúbicos.

La posición (de su centro óptico, en metros) y orientación de la cámara cenital vienen dadas por el vector (21) y la matriz de rotación (20), respectivamente (ambas definidas según la expresión (14)). Del mismo modo la posición y orientación de la cámara lateral vienen dadas por (23) y (22).

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{20}$$

$$T = (0{,}71, 0{,}74, 2{,}87)^T \tag{21}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix} \tag{22}$$

$$T = (1{,}45, 0{,}68, 0{,}77)^T \tag{23}$$

Las imágenes tomadas por dos cámaras cercanas, como en el caso de un par estéreo clásico, son muy parecidas, tienen pequeñas desviaciones una de otra. Por ello requieren de mayor precisión en sus parámetros de calibración para calcular la profundidad de los objetos con una exactitud aceptable. A cambio es más fácil encontrar similitudes entre las dos imágenes y así localizar objetos por su forma, precisamente por tener pocas diferencias. En contraste, la colocación perpendicular de las cámaras en el prototipo facilita la compensación de errores al tomar mayor información del entorno. Las cámaras lejanas perpendiculares dan mayor información al completar en una cámara lo que la otra no ve. Aunque en aplicaciones como seguridad en edificios esta disposición sea viable, en otras como los robots móviles puede no serlo. Una contrapartida de esa ubicación es un posible aumento de la dificultad de encontrar correspondencias entre las imágenes. En nuestro sistema esta dificultad no se presenta aún, puesto que se persigue a un único objeto, fácilmente distinguible mediante un filtro de color.

4.2. Diseño software

Todo el sistema ha sido implementado en C++ para ser probado en las máquinas del laboratorio. El desarrollo ha empleado las bibliotecas OpenCV como herramienta auxiliar y XForms[3] como sistema de ventanas. Nuestro programa se llama *Sauron*[4].

La interfaz gráfica de la aplicación desarrollada puede verse en la figura 8. Muestra las dos imágenes capturadas, el filtro de color realizado y los centros de masas. Además permite al usuario configurar el rango de color por el que quiere filtrar al objeto.

Internamente el programa se conecta a dos servidores de *Óculo* que corren en las máquinas que tengan instaladas las cámaras. La comunicación se realiza utilizando TCP por lo que no hay problema a la hora de mover las cámaras a otros emplazamientos, siempre y cuando la red de comunicación sea lo suficientemente ágil. La clase *OculoClient* es la encargada de realizar la tarea de proporcionar las imágenes al resto de las clases.

[3] http://www.go.dlr.de/pdinfo_dv/xforms.html

[4] http://gsyc.escet.urjc.es/~barrera/software/sauron

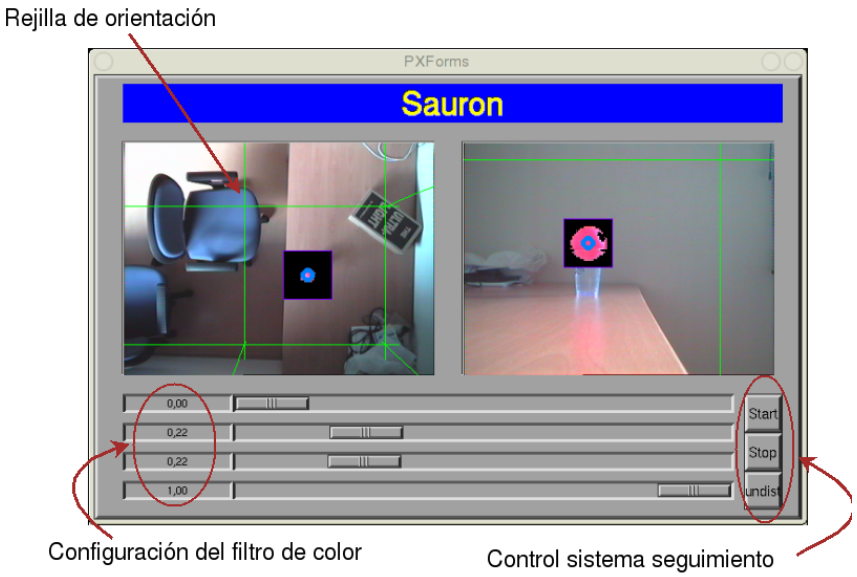


Figura 8: Captura del programa en funcionamiento.

Por encima de la parte de red del programa, éste se organiza empleando tres clases principales. Justo por encima del nivel de red, del cliente de Óculo, la clase *Eye* se encarga de realizar la búsqueda en la imagen obtenida para localizar la pelota. Una vez localizada obtiene la línea de visión aplicando el modelo de proyección descrito en la sección 2.2. Tanto de esta clase como de *OculoClient* tendremos tantas instancias como a cámaras queramos conectarnos. Será la clase *Sauron* la encargada de mezclar la información de las líneas de visión y realizar el seguimiento temporal. Esta arquitectura permite incrementar fácilmente el número de cámaras del sistema, que es una de nuestras líneas futuras de trabajo. Aparte existe un clase encargada de la interfaz gráfica del programa (figura 8). En la figura 9 se puede ver un esquema de la relaciones entre los objetos de nuestro programa.

Durante la realización de los experimentos se encontraron dificultades a la hora estimar la orientación correcta de las cámaras. Por dicho motivo se añadió a las imágenes la proyección de una rejilla tridimensional de referencia, como la mostrada en la figura 10, con las líneas separadas un metro entre sí. Estas líneas sirven para alinear correctamente las cámaras a marcas del laboratorio (paredes, suelo, etc.).

4.3. Medidas de precisión

Para caracterizar la precisión del sistema hemos realizado una serie de medidas colocando la pelota en distintas posiciones conocidas y anotando la estimación de posición dada por el sistema. Estas medidas aparecen resumidas en el cuadro 1, en centímetros, siendo X_m, Y_m, Z_m la posición medida y X_r, Y_r, Z_r la posición real.

La diferencia entre la posición real y la estimada se encuentra aproximadamente entre 3 y 5 centímetros. Más allá de los números exactos, la principal conclusión de estas pruebas es que en el prototipo experimental el sistema entrega un error

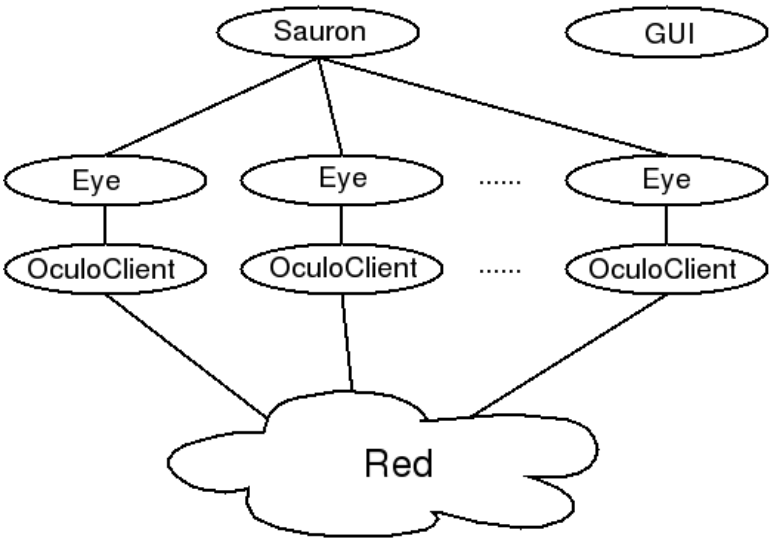


Figura 9: Captura del programa en funcionamiento.

absoluto del orden de centímetros. Sobre este margen de error habría que considerar la posible existencia de fallos centimétricos en las medidas reales al realizarse con un metro convencional. También hay que matizar esos resultados pensando en que el volumen de solape entre los campos visuales de ambas cámaras es de unos $3m^3$. Habría que estudiar qué ocurre cuando el volumen cubierto es mayor y las cámaras están más lejanas.

Un factor crítico encontrado a la hora de acotar los errores son las variaciones de orientación de ambas cámaras. Incluso usando sistemas de calibración groseros, los errores de orientación eran mucho más influyentes en el error final del sistema que los errores en los parámetros intrínsecos. Variaciones de menos de 5 grados en

Medido (cm)			Real (cm)			Error (cm)
X_m	Y_m	Z_m	X_r	Y_r	Z_r	
58	55	82.5	60	55	83	2.1
103	55	81.7	107	49.5	83	6.9
50	46	75	52	46	74	2.2
26	59	109	27	58	111	2.4
80	46	76	82	47	78	3.0
69	67	76	71	68	78	3.0

Cuadro 1: Posición medida, posición real y error entre ambas, en centímetros.

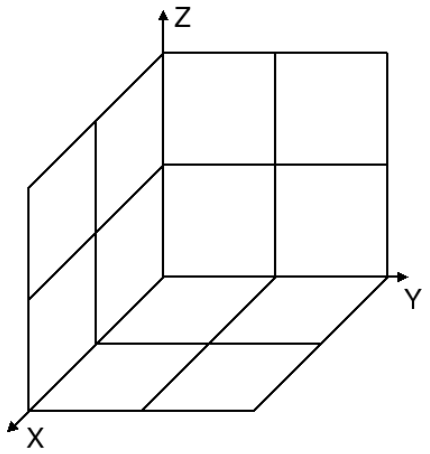


Figura 10: Rejilla de referencia para orientar las cámaras.

una de las cámaras respecto a la matriz de rotación usada desembocan en errores en la medida en torno a 30 cm, a una distancia de medio metro de la cámara.

4.4. Rendimiento temporal

Uno de los requisitos del sistema es la vivacidad, pensando que un coste computacional ligero permite incorporarlo sin problema, por ejemplo, al sistema perceptivo de un robot móvil. Una medida sencilla de esta vivacidad es el número de imágenes que el sistema es capaz de analizar por segundo. En el montaje experimental obtuvimos una velocidad de 6 fotogramas por segundo en cada cámara, filtrando el color en las imágenes completas (resolución de 320x240).

Como se comentó en la sección 2.1, para aumentar la velocidad del sistema podemos emplear una *ventana de búsqueda*, según se aprecia en la figura 2. Filtrando el color exclusivamente en ella la velocidad del sistema aumenta hasta los 20 fotogramas por segundo y cámara, dando un total de 40 imágenes procesadas al segundo. Esta es la velocidad a la que nuestro sistema procesa imágenes, que no es la misma que la velocidad a la que las captura. En el montaje experimental el sistema solo es capaz de proporcionar 12 imágenes nuevas por segundo para ser procesadas, por lo que 8 serán repetidas.

Otra de las medidas del rendimiento temporal que hemos realizado es ver qué partes del programa conllevan mayor coste computacional. Esto permite localizar las partes donde es más conveniente optimizar. Hemos empleado la utilidad del kernel de Linux `Oprofile`[5]. Esta herramienta muestrea periódicamente en el tiempo y comprueba qué instrucción está ejecutando el programa en esos instantes. Como salida indica, agrupando por funcionalidad, los fragmentos del proceso que más tiempo consumen, medido en muestras obtenidas sobre el total tomadas. Con ello podemos ver cuáles son las partes más costosas del proceso. Para nuestra aplicación

[5]http://oprofile.sourceforge.net/

el consumo de CPU se reparte de la siguiente manera, expresado como el porcentaje sobre el total de muestras tomadas:

60 %	Traer la imagen por la red
28 %	Realizar el filtrado de color
3.5 %	Filtro de partículas
2.5 %	Visualización
2.0 %	Proyecciones de las líneas de visión

Resulta interesante comprobar que la parte más costosa de todas, y que se lleva más de la mitad del tiempo, es la obtención de las imágenes. Aún así trabajamos en tiempo real (con las dos cámaras), lo que nos indica el bajo coste que tiene el sistema en sí. Además, tanto para la parte de filtro de partículas como para el sistema de proyección existen optimizaciones a nivel de programación que pueden realizarse para mejorar el rendimiento.

4.5. Dinámica del filtro de partículas

Tal y como lo hemos presentado en la sección 3, el filtro de partículas presenta tres parámetros fundamentales que permiten variar su comportamiento: N_s (número de muestras), σ_m (desviación típica del modelo de Medida sensorial) y σ_e (desviación típica del modelo de Evolución). Para conocer mejor su dinámica y poder ajustar esos parámetros a unos valores óptimos se realizaron una serie de experimentos con el mismo. En particular utilizamos un número suficiente de partículas, $N_s = 200$, y hemos estudiado el efecto de sendas desviaciones típicas.

Los experimentos consistieron en entregar al filtro de una serie de observaciones simuladas. De esta manera que enfocamos el estudio exclusivamente en la evolución de la población de muestras, separándolo de la calidad de las observaciones 3D puntuales conseguidas en el prototipo. En concreto, para comprender la dinámica del filtro, estudiamos la respuesta de la población de partículas a la *función escalón*, que simula el movimiento instantáneo del objeto a seguir desde un punto a otro. Inicialmente colocamos el objeto en una posición determinada, desplegamos las partículas en un cubo de 2x2x2 metros y esperamos a que el sistema converja. Una vez ha convergido realizamos un movimiento instantaneo a dos metros de distancia.

Los criterios utilizados para evaluar la bondad de una u otra configuración son dos: la *velocidad de convergencia* y el *error residual* una vez que la población de partículas ha convergido. Ambos los hemos medido desde la evolución temporal del error en la estimación. Por ejemplo, la gráfica de la figura 11 muestra en el eje Y el error (entendido como distancia entre la posición real y la estimada por el filtro de partículas) durante el experimento. En el eje X aparece el número de iteraciones del algoritmo, es decir, el paso del tiempo. En la figura puede verse la convergencia inicial y el incremento del error al desplazar el objeto bruscamente en la iteración 30, con dos poblaciones de σ_m diferente. Puede apreciarse que una converge antes que la otra.

Para comprender mejor el efecto de σ_e y σ_m en la dinámica hemos barrido el

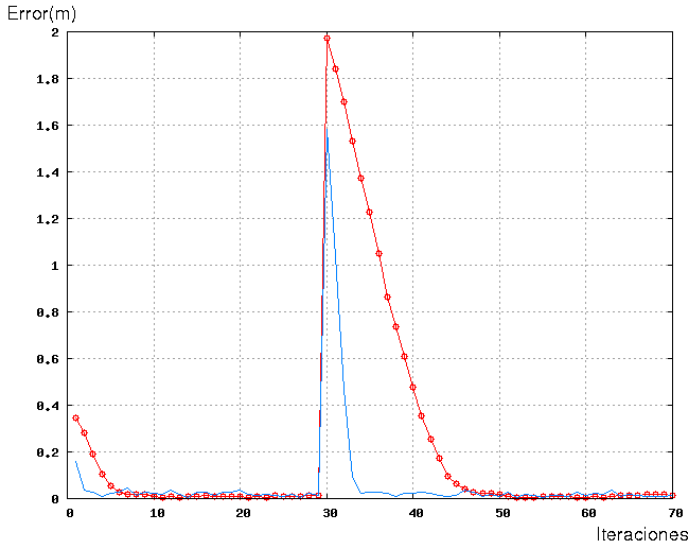
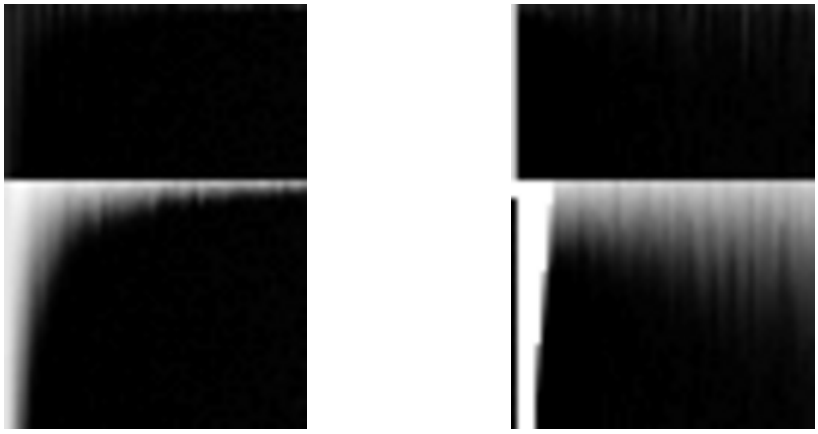


Figura 11: Error de localización (en metros) ante un movimiento escalón para dos filtros de partículas.

espacio de estos parámetros y analizado la evolución del error en la estimación de la figura 12.



(a) (b)

Figura 12: Evolución espacial de las partículas ante un movimiento

En la figura 12(a) se tiene la evolución temporal del error para 50 valores distintos de σ_e manteniendo constante $\sigma_m = 0{,}1$. En el eje Y se representa el paso del tiempo (70 iteraciones), y en el eje X valores crecientes de σ_e (desde 0.001 a 0.3, en incrementos lineales). El color de cada pixel indica el error para la población de partículas en ese instante de tiempo, más pequeño cuanto más oscuro. De esta manera una columna en esta figura muestra toda la evolución temporal para unas σ_e y σ_m determinadas.

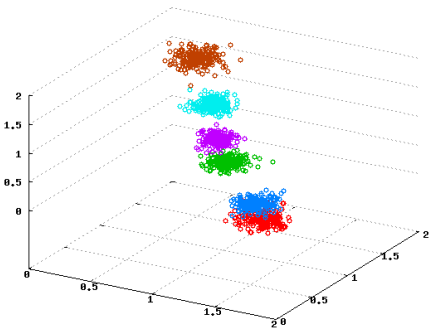
σ_e indica el grado de movimiento introducido en las partículas de una iteración a la siguiente. Si dicha innovación es alta el seguimiento se hará rápidamente. Esto se nota en la parte derecha de la figura 12(a), donde el error cae en apenas 3 o 4 iteraciones tras el cambio de posición del objeto. A medida que σ_e disminuye la convergencia se ralentiza, el error de estimación tarda más iteraciones en reducirse. En el caso límite de valores muy pequeños de σ_e (columnas de la izquierda en la figura 12(a)), la población no le da tiempo a converger tras las 40 iteraciones a la nueva posición. Es llamativo este comportamiento, cuando con los mismos valores la población inicial sí converge a la primera posición. La diferencia radica en que la población inicial es completamente aleatoria, mientras que para localizar la segunda posición se parte de una población muy concentrada alrededor de la primera posición del objeto.

Análogamente, en la figura 12(b) se aprecia el comportamiento del error cuando σ_m varía desde 0.001 a 0.4 (valores crecientes de izquierda a derecha) y se mantiene fija $\sigma_e = 0{,}06$. σ_m permite discriminar entre las muestras más y las menos compatibles con la medida sensorial, bajando la probabilidad de las muestras poco compatibles.

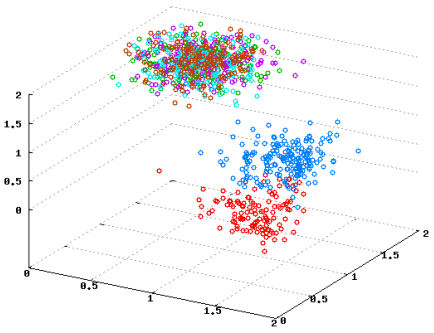
Con σ_m pequeñas sólo las muestras muy cercanas a la observación se propagarán a la siguiente generación, la riqueza del muestreo disminuye, y se ralentiza el seguimiento. Este efecto se observa claramente en las columnas blancas en parte izquierda de la figura 12(b), donde el error se mantiene alto despues de 70 iteraciones, es decir que la población no converge, no encuentra la nueva ubicación del objeto. A medida que σ_m crece acepta como muestras probables a las que caen en la vecindad, manteniendo cierta heterogeneidad que acelera la convergencia a la posición real. Si σ_m crece demasiado, parte derecha de la figura 12(b), entonces el sistema apenas discrimina entre muestras cercanas y menos cercanas, todas son igualmente válidas, lo que lleva a una disminución del error muy lenta.

En la elección conjunta de los parámetros hemos apreciado un compromiso entre la velocidad de convergencia y el error residual que obtenemos. En la figura 13 podemos apreciar la evolución durante el movimiento en escalón. Tanto en la imagen de la izquierda como en la de la derecha se han representado del mismo color las poblaciones en los mismos instantes de tiempo. Como se aprecia en 13(a) (con σ_m menor), la evolución es mucho más lenta, el conjunto de las partículas es más compacto y cuesta más realizar la innovación necesaria para llegar a la nueva posición. Sin embargo en el caso de la figura 13(b), la velocidad de convergencia es mucho más rápida debido a la dispersión de las partículas. Es justo esta dispersión la que se traduce en un error residual más alto una vez que el sistema ha convergido.

Una opción posible consistiría en ajustar los modelos de evolución y medida según estimemos la velocidad del objeto. Por ejemplo, en caso de quedarse quieto un tiempo podemos rebajar las desviaciones típicas de las gaussianas a fin de realizar una estimación más precisa de su posición; y cuando el objeto está moviéndose continuamente mantener las gaussianas más amplias para acelerar la convergencia y no perderlo.



(a)



(b)

Figura 13: Evolución espacial de las partículas ante un movimiento.

5. Trabajos relacionados

La utilización de dos o más cámaras para reconstruir escenas y/o localizar objetos en el mundo son problemas que se han abordado tanto en el campo de la visión artificial como en el de la robótica. Mientras en visión computacional se permiten enfoques basados en procesamiento por lotes y el tiempo no es una restricción primordial, en robótica la vivacidad es un requisito de partida, y el procesamiento de las imágenes ha de ser forzosamente secuencial, a medida que éstas se van obteniendo.

Los sistemas geométricos han sido muy utilizados, principalmente con pares estéreo de cámaras. La base de éstos reside en buscar correspondencia de homólogos de una imagen en la otra empleando la restricción epipolar. La mayor limitación de este sistema es que emplea una imagen como principal y la otra como auxiliar, presentando en la primera un punto único de fallo. Se puede incrementar el número de cámaras, pero sigue existiendo esta limitación en cuanto a una cámara principal.

En cuanto a localización de objetos utilizando visión, Margaritis [17] emplea rejillas espaciales para almacenar la información probabilística del sistema. En este

caso emplea una única cámara que va moviendo por una estancia a diferentes posiciones desde la que observa un mismo objeto. Reconstruyendo la línea de visión para cada una de ellas va ajustando los valores de toda la rejilla a la vista de la nueva situación. El procesamiento se realiza de forma secuencial, sin añadir nueva información hasta que no se ha procesado la anterior. Este punto puede presentar problemas si una imagen errónea elimina valores de la rejilla que luego resultan ser importantes.

Utilizando el mismo enfoque bayesiano, otra aproximación para la localización visual de objetos en 3D, en este caso puertas, es [20]. Este sistema emplea visión activa para situar las cámaras en puntos de interés donde previsiblemente se puede recabar más información y reducir más la incertidumbre en la localización de las puertas. Resalta también el sistema desarrollado por Gorodnichy para localizar la nariz de una persona, pensado para facilitar la interacción hombre-máquina [11], y el desarrollado por Jennings para localizar un dedo [15].

Al problema de la localización de objetos móviles desde cámaras de posición conocida se le puede dar la vuelta y utilizar las mismas técnicas para localizar una cámara móvil partiendo de unas balizas visuales de posición conocida [7].

En cuanto a la reconstrucción visual de escenas, algunos trabajos no necesitan el emparejamiento típico de los pares estéreo y utilizan geometría proyectiva combinada con rejillas de probabilidad tridimensionales. Por ejemplo, Collins [5] emplea una rejilla tridimensional en cuyas casillas almacena la probabilidad de que el objeto se encuentre en esta posición. Desde cada cámara reconstruye la linea de visión e incrementa el valor de las celdas que atraviesa el rayo. Cuando las líneas de visión de diferentes cámaras se aproximan aparecen zonas con valores altos. Realizando un estudio bayesiano de los resultados obtiene la posición del objeto. Este método trata a todas las cámaras por igual y puede escalar para utilizar cualquier número de cámaras. Entre sus limitaciones está que la precisión máxima viene dada por el tamaño de la rejilla, tamaño que afecta directamente al coste computacional.

Otro sistema de reconstrucción de escenas empleando proyección geométrica es el de Bustos [4]. Este sistema reconstruye en tiempo real el entorno cercano de un robot a partir de imágenes tomadas por sus dos cámaras. Para ello busca puntos relevantes (zonas de alto contraste, esquinas) en cada una de las imágenes, los empareja y combina a partir de un modelo proyectivo. El objetivo de este trabajo consiste en obtener una representación unificada, con información de profundidad, de las imágenes proporcionadas por las dos cámaras.

Un trabajo interesante de reconstrucción que no emplea emparejamiento entre imágenes es el de Louchet et al [3]. En su sistema se tiene un conjunto de puntos tridimensionales hipotéticos, llamados *moscas*, que van evolucionando, de manera similar a la población empleada en nuestro filtro de partículas. Utilizan operadores genéticos para generar nuevas poblaciones y la similitud entre las proyecciones de las moscas en ambas imágenes para promocionar a las mejor situadas. El resultado final es que las moscas acaban posandose en las posiciones tridimensionales de los bordes de los objetos del mundo.

6. Conclusiones y líneas futuras

En este trabajo hemos presentado un sistema de seguimiento tridimensional basado en visión. La técnica utilizada combina la triangulación tridimensional desde varias cámaras con un filtro de partículas para explotar la continuidad temporal.

Hemos implementado y probado experimentalmente el sistema en un prototipo, obteniendo varias conclusiones relevantes. Primero, en cuanto a precisión hemos obtenido errores de posición centimétricos en un volumen de búsqueda de varios metros cúbicos. Segundo, hemos comprobado que el algoritmo tiene un coste computacional acotado, lo que se traduce en un funcionamiento vivaz en tiempo real. El hecho de emplear líneas de proyección geométrica es más eficiente y rápido que la utilización de sistemas bayesianos sobre rejillas 3D para la existencia de un único objeto. Tercero, el factor más crítico a la hora de obtener un buen funcionamiento en el sistema reside en dar correctamente las posiciones y orientaciones de las cámaras (parámetros extrínsecos). Incluso un error en la calibración de los parámetros intrínsecos no afecta tanto a la hora de obtener la línea de visión de un objeto desde la cámara como lo es la orientación de ésta. La calibración que hemos usado a pesar de ser grosera, se ha demostrado totalmente satisfactoria.

También hemos podido estudiar la dinámica del filtro de partículas para un caso real. La dinámica debe mantener siempre un compromiso entre la velocidad de convergencia y el ruido residual.

Quizá la extensión natural más relevante al sistema actual sea la incorporación de observaciones puramente visuales al filtro de partículas. Por ejemplo, en vez de tomar como observaciones las estimaciones 3D de triangulación, tomar directamente las imágenes de las cámaras (o alguna característica bidimensional suya, por ejemplo el centro de masas de un objeto rosa). De esta manera se evita el paso intermedio de la triangulación y es el filtro de partículas el que concluye cierta posición a partir de datos de las imágenes, incompletos en cuanto a profundidad. Esto facilitaría la extensión del sistema a un número arbitrario de cámaras y proporcionaría robustez frente a oclusiones y fallos en alguna de ellas.

Otras posibles mejoras se centrarían en eliminar las asunciones realizadas para simplificar el problema planteado. En esta línea: extender el sistema para que sea capaz de seguir a más de un objeto, detectar los objetos no sólo por color, sino por movimiento también, generalizar el algoritmo para emplear más de dos cámaras, etc.

El prototipo desarrollado se ha probado con un volumen relativamente pequeño. Convendría ampliar el volumen en el que se puede seguir el objeto, y ver como se degrada entonces el error obtenido. Previsiblemente para mantener la precisión en ese escenario sea necesaria más exactitud en los parámetros de calibración. Otro paso para ampliar el volumen de trabajo consistiría en utilizar cámaras móviles, por ejemplo encima de cuellos mecánicos, de manera que puedan cubrir más espacio que el simple cono estático de visión. Se podrían probar entonces estrategias de seguimiento activo.

En vista de las limitaciones encontradas, también resultaría muy útil añadir al

sistema un módulo de autocalibración de los parámetros extrínsecos. De esta forma se podrá aislar en mayor medida los resultados de los problemas encontrados en la calibración manual.

Referencias

[1] Ronald C. Arkin, Masahiro Fujita, Tsuyoshi Takagi, and Rika Hasegawa. An ethological and emotional basis human-robot interaction. *Robotics and Autonomous Systems*, 42:191–201, 2003.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[3] A.M. Boumaza and J. Louchet. Mobile robot sensor fusion using flies. In Günther R. Raidl, Jean-Arcady Meyer, Martin Middendorf, Stefano Cagnoni, Juan J. Romero Cardalda, David Corne, Jens Gottlieb, Agnès Guillot, Emma Hart, Colin G. Johnson, and Elena Marchiori, editors, *Applications of Evolutionary Computing, EvoWorkshops 2003*, volume 2611 of *Lecture Notes in Computer Science*, pages 357–367. Springer, 2003.

[4] P. Bustos, P. Bachiller, J. Vicente, M. Broncano, and C. Fernández. Murphy: hacia un robot con visión estereoscópica. In *Actas del 2do Workshop de Agentes Físicos, WAF'2001*, pages 91–104, Móstoles, Madrid, March 2001.

[5] R. T. Collins. A space-sweep approach to true multi-image matching. Technical Report UM-CS-1995-101, , 1995.

[6] José Vicente Crespo. *Métodos Visuales de Reconstrucción 3D en Robots Móviles*. PhD thesis, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 2002.

[7] Andrew J. Davison. SLAM with a single camera. In *Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots, in conjunction with Int. Conf. on Robotics and Automation ICRA-2002*, 2002.

[8] F. Dellaert, W. Burgard, D. Fox, and Thrun S. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Ft. Collins, CO (USA), June 1999. IEEE Computer Society.

[9] Andrew P. Duchon, William H. Harren, and Leslie Pack Kaelbling. Ecological robotics. *Adaptive Behavior*, 6(3/4):473–507, 1998. Special Issue on Biologically Inspired Models of Spatial Navigation.

[10] M.C. García-Alegre, P. Bustos, and D. Guinea. Complex behaviour generation on autonomous robots: a case study. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1729–1724, Vancouver (Canada), October 1995.

[11] D. Gorodnichy, S. Malik, and G. Roth. Affordable 3d face tracking using projective vision. In *Proc. Intern. Conf. on Vision Interface*, pages 383–390, Alberta, Canada, May 2002.

[12] María Elena López Guillén. *Sistema de navegacion global basado en procesos de decisión de Markov Parcialmente Observables. Aplicación a un robot de asistencia personal.* PhD thesis, Escuela Politécnica. Universidad de Alcalá, 2004.

[13] Ian Horswill. Visual architecture and cognitive architecture. *Journal of Experimental and Theoretical AI*, 9(2-3):277–292, April-September 1997.

[14] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking, 1998.

[15] Cullen Jennings. Robust finger tracking with multiple cameras, 1999.

[16] D.J.C. Mackay. Introduction to Monte Carlo methods. In M. Jordan, editor, *Learning in Graphical Models*, pages 175–204. MIT Press, Cambridge (MA, USA), 1999.

[17] Dimitris Margaritis and Sebastian Thrun. Learning to locate an object in 3D space from a sequence of camera images. In *Proc. 15th International Conf. on Machine Learning*, pages 332–340. Morgan Kaufmann, San Francisco, CA, 1998.

[18] N.J. Nilsson. A mobile automaton: an application of artificial intelligence techniques. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence IJCAI*, pages 509–520, Washington, (USA), 1969.

[19] José María Cañas Plaza. *Jerarquía Dinámica de Esquemas para la Generación de Comportamiento Autónomo*. PhD thesis, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 2004.

[20] José María Cañas Plaza, Reid Simmons, and María C. García-Alegre. Detección probabilística de puertas con visión monocular activa, Febrero 2001.