INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2005-2006

Proyecto Fin de Carrera

Aplicación de seguridad basada en visión.

Tutores: José María Cañas Plaza

Pablo Barrera González

Autor: Antonio Pineda Cabello

Para mi familia y mi novia, que siempre están conmigo en lo bueno y en lo malo.

Agradecimientos.

En primer lugar quiero agradecerle a José María Cañas la extraordinaria confianza que ha depositado en mí para la elaboración de este proyecto, la paciencia que ha tenido conmigo en todo momento durante este año y sobre todo, el entusiasmo que ha despertado en mí de seguir investigando y colaborando dentro del Grupo de Robótica de la URJC.

Dentro del Grupo de Robótica quiero agraderle toda la ayuda prestada especialmente a Pablo Barrera, por toda la sabiduría que ha compartido conmigo. También quiero agradecerles todo a Carlos Agüero y a Víctor Gómez, y por supuesto a mis compañeros de batalla Víctor Hidalgo, José Alberto López, Redouane Kachack y Pedro Díaz.

A mis amigos de la facultad Jacob Alvarez, Oscar Higuera, Iván Baltasar e Israel Navas, por toda el ánimo que siempre me han dado, por la enorme amistad que nos une y que siempre perdurará.

Más especialmente quiero dedicar este proyecto de fin de carrera a mi familia. Ellos son los que han hecho posible que esté hoy aquí redactando esta líneas, por todo el apoyo, la paciencia y el amor que siempre han tenido conmigo.

Todo el esfuerzo de este trabajo también está dedicado a mi novia Katia, que cree en mí para todo lo que me propongo.

:B ?

Resumen.

En los últimos años, los grandes avances en la informática han permitido el desarrollo de nuevas disciplinas científicas como la visión artificial. Las cámaras son hoy en día sensores de bajo coste cuyas imágenes proporcionan gran cantidad de información del entorno donde se utilicen. Extraer dicha información e interpretarla posibilita la generación de aplicaciones que incorporen comportamientos inteligentes enormemente útiles en tareas complejas de nuestra vida cotidiana.

El propósito de este proyecto es crear una *aplicación orientada a la seguridad* que estime la posición tridimensional de una persona o un objeto dentro de una habitación de gran volumen. Además, detectará si esa persona localizada se aproxima demasiado a una zona de la sala declarada restringida y alertará de dicha intrusión con una alarma visual y sonora.

La solución al problema de la *localización 3D* que subyace en esta aplicación se alcanza de dos formas distintas mediante dos técnicas muestreadas. La primera es una implementación de un *filtro de partículas* basadas en técnicas probabilísticas de Monte Carlo. La segunda es un algoritmo evolutivo, diseñado e implementado por primera vez para adaptarse a la localización 3D, basado en el uso de operadores genéticos, llamado *filtro de moscas*. Los dos algoritmos utilizan la *información de color y movimiento* capturada de las imágenes que se reciben de cuatro cámaras situadas en los rincones de la habitación. El sistema aprovecha las cuatro imágenes 2D y es capaz de estimar la posición 3D de un objeto o persona.

La aplicación ha sido desarrollada apoyándose en la *plataforma software jde.c* e implementada en forma de tres hebras iterativas o *esquemas*. Este sistema es lo suficientemente *vivaz* como para seguir tridimensionalmente a una persona en movimiento, está dotado de una *precisión centimétrica* y funciona con *hardware no específico*.

Índice general

Índice de figuras

Capítulo 1
Introducción

La vista es el sentido que más utilizamos para captar información de nuestro entorno. Mediante los datos visuales que obtenemos de los ojos somos capaces de modelar un esquema de nuestro entorno que nos permite conocer detalles de los objetos de alrededor. Gracias a estos detalles somos capaces de reconocer dichos objetos por su color, por su forma, detectar movimiento en ellos o incluso estimar aproximadamente la distancia que nos separa.

La adaptación de estas acciones a sistemas informáticos y a robots es una tarea todavía en área activa de investigación. La estimación de la distancia o localización de un objeto se lleva estudiando mucho tiempo en el campo de la robótica. Gracias a estos estudios, existen a día de hoy múltiples técnicas para determinar la situación de un objeto, algunas de ellas basadas en visión artificial.

Este proyecto fin de carrera consiste en una aplicación directa de la localización visual, un sistema automático que permite seguir un objeto en el interior de una habitación y estimar su posición 3D gracias a la información de color y movimiento de las imágenes que se reciben.

En este primer capítulo se describe el contexto sobre el que se apoya el sistema: la visión artificial y la localización.

1.1. Visión artificial

Debemos entender por visión artificial o visión computacional como el área de la inteligencia artificial que se dedica a extraer información de las imágenes con el fin de comprender y asimilar lo que en ellas está sucediendo. La visión artificial tiene como objetivo por tanto interpretar las imágenes analizadas para obtener información relevante sobre elementos que en ellas aparecen.

Aunque a día de hoy el uso de la visión artificial está muy extendendido, sus comienzos se remontan a mediados del siglo XX. Ya entonces se comenzaron a desarrollar programas para la detección de tanques o sistemas de detección de obstáculos. Sin embargo, hasta comienzos de la década de 1990 no comenzaron a aparecer ordenadores con velocidad de cómputo suficiente para procesar imágenes de forma ágil y con poca demora. A partir de entonces la visión computacional comenzó a emplearse para múltiples tareas y y su desarrollo fue concentrándose en problemas de tratamiento de las imágenes como la segmentación, el reconocimiento de formas o el filtrado de bordes.

Mediante el uso de estas técnicas fue posible alcanzar numerosos propósitos tales como: *reconocimientos faciales* (figura 1.1 (a)), *reconocimiento de objetos* (figura 1.1 (b)), empleado a menudo en la industria para ayudar en el control de calidad, *seguimiento de objetos 2D* o la *detección de movimiento*, usados frecuentemente en entornos de seguridad para el control de personas.

Figura 1.1: Reconocimiento con visión artificial - Facial (a) y Objeto (b).

En la actualidad, los avances en geometría proyectiva, pares estéreo y autocalibración han abierto la puerta a la extracción de *información tridimensional* de las imágenes. El uso de estos nuevos datos permite alcanzar nuevas cotas en el área de la visión artificial, pues la información tridimensional permite conocer mejor y con una mayor precisión el entorno.

Gracias al impulso de estas nuevas técnicas es posible hoy día realizar *reconocimiento 3D* de rostros u objetos, que pueden ser utilizados en distintos ámbitos

como la representación de piezas en entornos industriales, de nuevo el reconocimiento de personas mediante el estudio biométrico en áreas de seguridad (figura 1.2), diseño infográfico para la industria del cine o de videojuegos, etc.

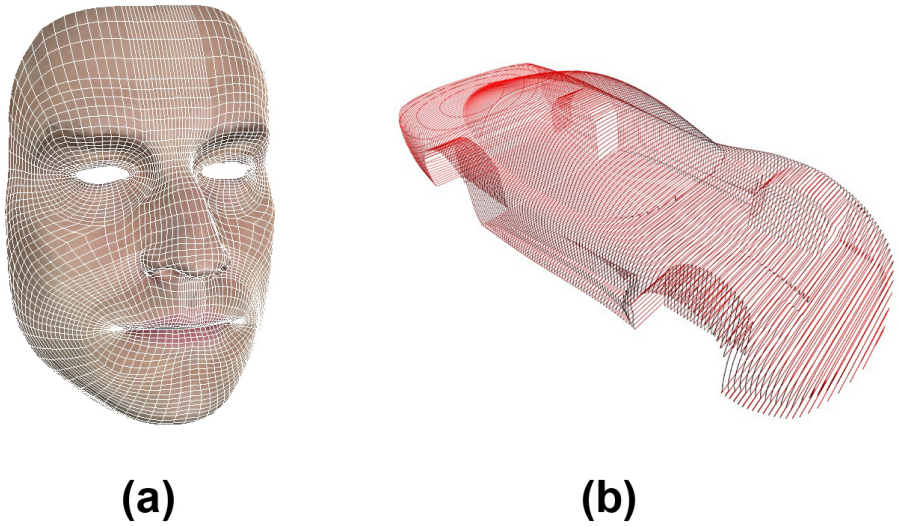


Figura 1.2: Modelado tridimensional - Rostro humano (a) y Pieza industrial (b)

Otra de las aplicaciones de la visión artificial con respecto a la extracción de información 3D es la que utiliza el sistema de *repetición tridimensional de jugadas deportivas* Eye-Vision[1]. Através de la colocación estratégica de un determinado número de cámaras alrededor de un estadio es posible obtener la realidad virtualizada de lo sucedido segundos antes. La escena tridimensional se obtiene de la composición de las imágenes 2D que consigue cada una de las cámaras.

Figura 1.3: Sistema de repetición tridimensional Eye-Vision.

[1]http://www.ri.cmu.edu/events/sb35/tksuperbowl.html

1.2. Localización

En términos generales se debe entender por localización el proceso de determinar *la posición de un objeto* dentro de un entorno. El conocimiento de esta posición viene determinado gracias a la información que se obtiene de los sensores del sistema donde se efectúa el proceso, que permiten realizar un *seguimiento* del objeto.

La localización resulta imprescindible en sistemas móviles para la toma de decisiones con respecto a la posición donde se encuentren. En sistemas de GPS (Global Positioning System) por ejemplo, el seguimiento de vehículos o personas se realiza gracias a receptores de GPS y en función de la posición global del vehículo, se pueden tomar distintas determinaciones como cambiar de ruta, parar el vehículo porque se ha llegado al destino, etc. Igualmente, en el área de la robótica móvil, la posición del robot en el mundo donde se encuentra implica tomar decisiones muy diferentes dependiendo de donde se halle. En sistemas de localización global es imprescindible el uso de de *mapas para situar el objeto* en el entorno.

Algunas técnicas para resolver la localización se describen en el apartado 1.2.2.

1.2.1. Localización 3D

La localización 3D consiste en determinar la *posición de un objeto* o elemento dentro de un mapa o *entorno tridimensional*. Esta posición habitualmente vendrá especificada como un punto tridimensional de coordenadas (x,y,z), cuya situación estará referida a un orígen o centro de referencia previamente establecido.

Para lograr la localización 3D es necesario el uso de cámaras u otros sensores que informen de la posición del objeto. En el caso de las cámaras, se necesitan al menos dos imágenes de posiciones diferentes de apuntando al mismo objeto para poder conseguir la localización 3D. Mediante las dos imágenes es posible aplicar una técnica de localización que determine la posición.

Una de las aplicaciones más famosas del seguimiento tridimensional son los sistemas Vicon[2] de captura de movimiento (figura 1.4). Estos sistemas son ampliamente utilizados en la industria cinematográfica y videojuegos para la captura de los movimientos de las personas, que serán aplicados a personajes virtuales generados

[2]http://www.vicon.com/

por ordenador. Esta técnica se basa en la grabación con múltiples cámaras de una persona vestida con un traje especial. Este traje lleva adheridos ciertos elementos que reflejan muy bien luz infrarroja emitida por gran cantidad de diodos situados en la propia cámara y que es captada por las mismas desde distintos puntos de observación. Con esta información, un ordenador puede generar una representación en movimiento de los puntos del traje, que puede ser aplicada a un modelo tridimensional para poder animarlo.

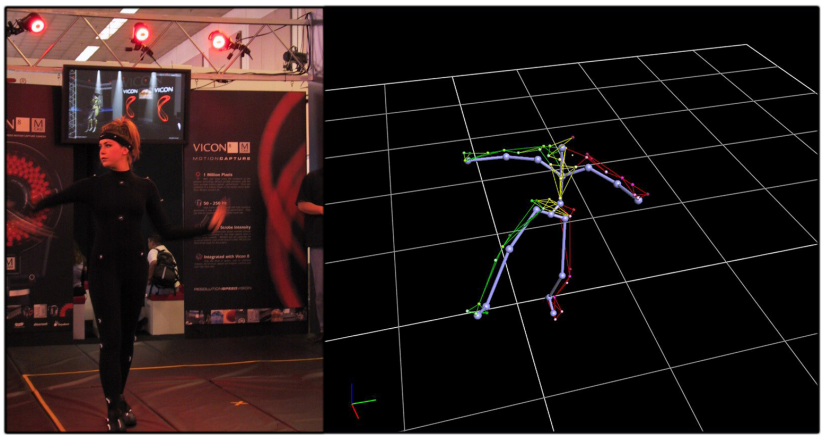


Figura 1.4: Sistema de captura de movimiento de Vicon Peak.

1.2.2. Técnicas de localización

En este apartado se contemplan algunas de las técnicas clásicas de localización:

- *Localización por mínimos cuadrados*: estiman la posición del objeto basándose en el cálculo del error mínimo de las observaciones realizadas. Necesitan de un gran número de observaciones para poder realizar una estimación precisa por lo que es difícil integrarlos en sistemas de tiempo real, aunque son muy vivaces. Guardan un historial de observaciones anteriores, que se van renovando mediante políticas de actualización. Si estas políticas no se ajustan al entorno de aplicación, las estimaciones pueden no ser fiables.

- *Localización con filtros de kalman*: tratan de estimar recursivamente la posición de mínima varianza fusionando información parcial e indirecta sobre localización [Maybeck, 1979][Bishop y Welch, 2005]. Su principal limitación es que es una técnica unimodal y exclusivamente gaussiana, por lo que no es capaz de manejar múltiples hipótesis.

- *Localización probabilística* [Thrun, 2000]: incorpora incertidumbre de acciones y observaciones que se acoplan a la incertidumbre que muestran los sensores. Este

tipo de localización consiste en determinar la probabilidad de que el objetivo se encuentre en una determinada posición a través de los datos obtenidos de los sensores. A cada posible posición se le asocia una probabilidad reflejando la verosimilitud de ser la posición actual del objetivo. Esta probabilidad se va actualizando con la incorporación de nuevas lecturas. Es posible localizar un objeto sin tener en cuenta su posición inicial permitiendo representar situaciones ambiguas que se irán desambiguando posteriormente. La eficiencia de estas técnicas generalmente depende del tamaño del entorno donde se efectúa la localización, pues en espacios muy grandes la probabilidad podría no reflejar la posición real.

- *Localización con métodos metaheurísticos*: estos filtros contemplan el problema de la localización como una búsqueda de la posición entre un número determinado de soluciones posibles. Los métodos metaheurísticos de localización realizan una búsqueda de la posición de manera algorítmica, efectuando evaluaciones puramente empíricas de los resultados posibles. Se basan en las experiencia aprendida de resultados anteriores. No aseguran una solución óptima al problema, sino más bien una aproximación que en ocasiones puede ser la mejor. Son computacionalmente muy eficientes incluso en problemas de complejidad exponencial (hasta un determinado orden).

1.3. Aplicaciones

La localización 3D puede aplicarse en múltiples escenarios en los que, según la situación, conocer la posición de un objeto o una persona puede servir y ayudar enormemente en tareas muy complejas. En este apartado se describen algunas de estas aplicaciones.

1. La localización 3D puede utilizarse para la *seguimiento de personas* en una residencia de 3ª edad. Por ejemplo, sería interesante detectar cuando algún miembro de la residencia se ha caído o desmayado, o más importante aún, cuando alguien *se encuentra tumbado en el suelo y lleva mucho tiempo sin moverse* en alguna planta de la residencia. El sistema indicaría además la situación exacta donde se ha desmayado o caído el sujeto y podría avisar a los enfermeros mediante una alarma de megafonía u algún procedimiento inalámbrico como un SMS a un teléfono móvil.

2. La vigilancia de sujetos se podría igualmente incorporar a una guardería para el *cuidado de niños pequeños*. Podría mantener prácticamente los mismos parámetros de detección que en el caso de vigilancia de ancianos y además se incorporarían otras funciones para evitar que los crios pudieran tocar enchufes o acercarse demasiado a ventanas y puertas, de forma que el sistema fuera capaz de interrumpir el paso de la corriente de dichos terminales y avisar a los cuidadores de que alguien de pequeña estatura se ha acercado demasiado a una zona peligrosa.

3. En zonas al exterior, podría *ayudar a evitar accidentes laborales*: en la construcción, en las minas, en fábricas industriales, en centrales eléctricas o en otro tipo de factorías. En todos estos ambientes de trabajo existe un claro riesgo para los trabajadores al estar en contacto directo con maquinaria industrial, por lo que un sistema de localización 3D prestaría gran ayuda a la prevención de riesgos, y no supondría una inversión muy cara.

Otra aplicación es la que se presenta en este proyecto fin de carrera: la *vigilancia y protección de objetos*. La funcionalidad consiste en localizar en 3D a cualquier objeto o persona dentro de determinada sala. La aplicación utilizará la posición estimada de dicho objeto o persona para comprobar si se encuentra cerca de una *zona de alta seguridad*, que puede ser cualquier parte de la habitación. En caso de que se encuentre a menos de un metro, el sistema *previene de la cercanía del objeto localizado* avisando con una alarma sonora y visual de dicha intrusión.

Un sistema como este podría suponer un cambio sustancial frente a las aplicaciones que a día de hoy se utilizan para la videovigilancia. Hasta la fecha casi todas las aplicaciones de este tipo se basan en la teleobservación (figura 1.5), donde un personal de seguridad se encuentra al cargo de visualizar monitores con las imágenes de todas las cámaras, o bien se realiza una grabación constante de cada cámara que es analizada al día siguiente. Algunos sistemas incorporan únicamente un sistema que permite grabar únicamente cuando se detecta movimiento en la zona.

Figura 1.5: Sistema de videovigilancia por teleobservación

Este sistema no necesitaría de ninguna persona la tarea de vigilancia y podría adaptarse a cualquier sala o zona exterior añadiendo más cámaras en los sitios estratégicos. Para mayor comodidad, la aplicación podría incorporar la posibilidad de enviar por email a una empresa de seguridad o a la policía, el instante de vídeo en el mismo momento en que alguien accede a la zona restringida.

Una vez introducido el contexto en el que se apoya este proyecto fin de carrera, se detallarán a continuación los objetivos y los requisitos que se han propuesto para la realización de esta aplicación. Posteriormente, en el capítulo 3 se describirá el entorno de desarrollo utilizado y los componentes hardware utilizados. Los capítulos 4, 5 y 6 analizan las dos técnicas de localización 3D utilizadas asi como los experimentos realizados para las dos. Por último, el capítulo 7 termina exponiendo las conclusiones obtenidas en la realización del proyecto y sugiriendo algunas líneas futuras de investigación relacionadas con la localización 3D y con esta aplicación.

Capítulo 2
Objetivos

Tras haber presentado el contexto general y particular sobre el que se asienta la aplicación, estableceremos en este capítulo los objetivos concretos que se han propuesto resolver mediante este proyecto fin de carrera al igual que los requisitos que han condicionado el desarrollo del mismo.

El objetivo del proyecto es *intentar solucionar el problema de la localización 3D de un objeto* mediante técnicas de visión. Con este propósito, se diseñará y se implementará una *aplicación de seguridad* que tratará de resolver este problema mediante dos algoritmos distintos. Esta aplicación debe ser capaz de proteger una determinada zona de una habitación y alertar sonora y visualmente si una persona se adentra en ella.

2.1. Descripción del problema

En este primer apartado, se describirá el objetivo principal de este proyecto fin de carrera. Dicho objetivo se puede dividir en varios subobjetivos más específicos:

1. *Implementación de un algoritmo probabilístico.* Estudiar el comportamiento de la aplicación mediante la implementación de un algoritmo basado en técnicas de Monte Carlo llamado *filtro de partículas*, empleando en el análisis información de color, movimiento o conjunción de ambas.

2. *Diseño e implementación de un algoritmo genético.* Estudiar el comportamiento de la aplicación mediante la utilización de un algoritmo evolutivo, diseñado e implementado para resolver el seguimiento tridimensional, llamado filtro de moscas. Se emplearan también distintas fuentes de información visual.

3. *Experimentos.* Se realizarán determinadas pruebas para probar y ajustar los algoritmos. Los resultados obtenidos serán interpretados para contrastar la eficacia de las dos técnicas.

Este trabajo pretende continuar en la misma línea del *Seguimiento tridimensional usando dos cámaras [Barrera y Cañas, 2004]* aportando como nuevas características: el aumento del espacio de localización a unos 120 m^3 aproximadamente, el uso de cuatro cámaras para realizar el seguimiento, la información no sólo de color sino también de movimiento y un primer diseño e implementación del filtro de moscas aplicado a la localización.

2.2. Requisitos

Los requisitos propuestos para la aplicación que presenta este proyecto fin de carrera son resumidos en los siguientes puntos:

1. Utilización del *lenguaje C* para programar la aplicación, apoyándose en el sistema operativo Linux.
2. Implementación del sistema apoyándose en la *plataforma software jde.c*[1].
3. Funcionamiento de la aplicación usando *hardware convencional*.
4. Aplicación del sistema a *habitación de gran volumen*, en concreto al Laboratorio de Robótica de la URJC.
5. Procesar datos con *gran vivacidad*.
6. Seguimiento de una persona o un objeto con una *precisión centimétrica*.

2.3. Metodología y plan de trabajo

El plan de trabajo utilizado en la realización de este proyecto ha consistido en el *modelo de desarrollo en espiral basado en prototipos* (figura 2.1). Este modelo de desarrollo se basa en la realización de varias subtareas sencillas que conjuntadamente compondrán el comportamiento final del sistema. Usando este modelo se aporta cierta flexibilidad en cuanto a posibles cambios de requisitos.

Este modelo de desarrollo se caracteriza por la realición de las subtareas en un número determinado de ciclos. En cada uno de estos ciclos existen cuatro etapas: *Análisis de requisitos, Diseño e implementación, Pruebas y Planificación del próximo ciclo de desarrollo.*

[1] http://gsyc.escet.urjc.es/ jmplaza/software.html

Durante todo el desarrollo del proyecto se han mantenido reuniones semanales con los tuturores para establecer y afinar los puntos que se han llevado a cabo en cada etapa, y para comentar los resultados de etapas anteriores.

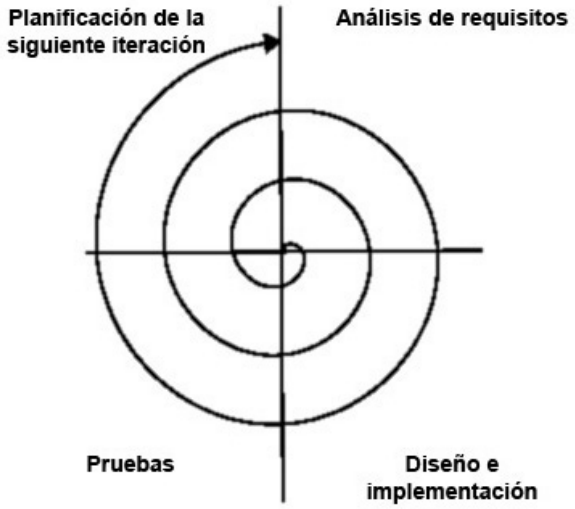


Figura 2.1: Modelo en espiral

Para el desarrollo de este proyecto se han completado los siguientes seis ciclos:

1. *Familiarización con la plataforma software jde.c* [Plaza, 2004] y estudio a fondo de la estructura de la misma. La primera iteración de la espiral consistió en la creación de varios esquemas para comprender el funcionamiento general de jde.c.

2. *Estudio de técnicas de visión artificial y otros conocimientos relativos* tales como filtros de color y movimiento, conocimientos de espacios de color, soportes de vídeo, calibración y funcionamiento de controladores de camáras USB y Firewire. Diseño y creación de nuevos esquemas en los que se aplican dichas técnicas.

3. *Diseño e implementación de la aplicación Watcher*. Esta etapa de la espiral consistió en la creación de los principales esquemas que componen la aplicación, diseñar e implementar el simulador 3D utilizando la biblioteca Progeo. Se integraron también las técnicas de visión implementadas previamente. En esta etapa se implementó también la interfaz gráfica de la que dispone el sistema.

4. *Estudio de técnicas de localización probabilísticas e implementación del filtro de partículas*. El propósito de esta etapa es el estudio de la localización 3D y de una de las técnicas utilizadas para resolverla, mediante la lectura de documentos y otros proyectos relacionados. Al termino del estudio se diseñó una implementación del filtro de partículas para resolver la localización 3D.

5. *Estudio de algoritmos genéticos y diseño e implementación del filtro de moscas.* En esta etapa se procedió exactamente igual que en el ciclo anterior. Se realizó un estudio de los algoritmos genéticos y se diseñó un filtro de moscas adaptado a resolver el problema de la localización 3D. Posteriormente se realizó una implementación del diseño realizado.

6. *Experimentos.* La última etapa de la espiral consistió en experimentos realizados con las dos técnicas implementadas, contrastando los resultados obtenidos.

Una vez descritos los objetivos, los requisitos y la metodología de este proyecto, a continuación se analizará la infraestructura software y hardware sobre la que se asienta la aplicación.

Capítulo 3
Entorno y plataforma de desarrollo

En este capítulo se describe la infraestructura software y hardware sobre la que se apoya este proyecto fin de carrera, al igual que las bibliotecas que se han utilizado para su funcionamiento.

3.1. Infraestructura hardware: Cámaras Firewire

Las cámaras son un elemento hardware indispensable en el funcionamiento de este proyecto. Son los sensores perceptivos utilizados por la aplicación de seguridad *Watcher* para poder estimar una posición 3D en la habitación. Las cuatro cámaras utilizadas son del modelo Apple iSight[1] (figura 3.1). Este modelo de cámara se caracteriza por poder capturar imágenes en color de hasta 640x480 píxeles con un ritmo de actualización de 15 fotogramas por segundo (fps), o bien de 320x240 a 30 fps tal y como se utiliza en este proyecto. Para ello se, se conecta la cámara a un ordenador através de una conexión de tipo IEEE1394 a 400 Mbps. El ritmo de actualización tan vivaz de los fotogramas se debe a la *utilización de DMA* para acceder a las imágenes capturadas. La cámara utiliza además un sistema de enfoque y apertura del iris automático.

Figura 3.1: Cámara web Apple iSight

[1] http://www.apple.com/isight

Para poder conectar las cámaras firewire y ejecutar la aplicación es necesario disponer de *al menos un ordenador* de gama media, al que estarán conectada las cuatro cámaras y que además ejecutará la aplicación de seguridad. Otra manera de conectar las cámaras es mediante una red local, en la que se necesitarían varios ordenadores funcionando como *servidores de imágenes*.

3.2. Infraestructura software: La plataforma JDE

La aplicación de seguridad *Watcher* se apoya en la plataforma robótica JDE que ha sido desarrollada en la Universidad Rey Juan Carlos. Esta plataforma permite desarrollar aplicaciones robóticas con sensores y actuadores, que tomen decisiones inteligentes de manera autónoma.

La plataforma plantea las aplicaciones como un conjunto de *esquemas* que se ejecutan simultáneamente en hilos distintos y que realizan tareas sencillas y concretas. Esta ejecución en paralelo de los esquemas dan lugar a un comportamiento. Existen esquemas de distintos tipos: *perceptivos* que se encargan de producir y almacenar información sensorial o elaborada por otros esquemas, de *actuación* son los que toman las decisiones para realizar una acción en función de la información sensorial obtenida tal como activar otros esquemas asociados., y de *servicio* que se encargan de las comunicaciones recogiendo información de sensores y enviando órdenes a los elementos actuadores. Los esquemas pueden organizarse en niveles estableciendo una jerarquía entre esquemas padre y esquemas hijo siendo el padre el que puede activar y desactivar a los esquemas hijo.

La plataforma se ha implementado en una arquitectura software "jde.c" que ofrece una interfaz de variables de percepción y de actuación (figura 3.2). Las medidas sensoriales aparecen como variables que se leen y las órdenes a los actuadores se dan escribiendo en otras variables .Si bien la arquitectura software está principalmente orientada a la programación de comportamientos en robots, ha sido adaptada al caso concreto de este proyecto para *aprovechar de ella todos los elementos dedicados a la visión artificial*. Principalmente se utiliza el acceso a las imágenes de las cámaras, que mediante la interfaz proporcionada se actualizan en las variables perceptivas *colorA, colorB, colorC y colorD*.

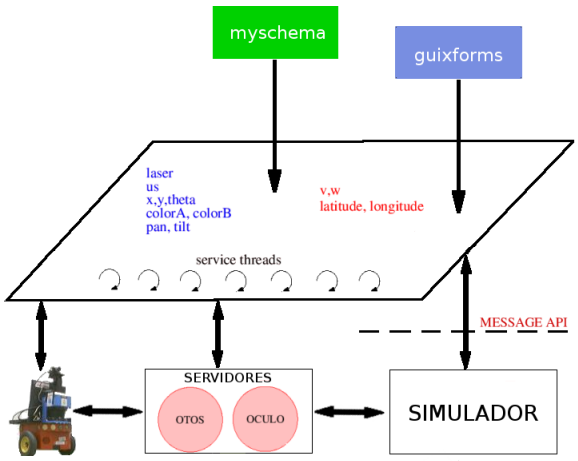


Figura 3.2: JDE básico

La plataforma cuenta además con determinados *servidores* que proporcionan acceso remoto a sensores y actuadores. Estos servidores se encargan de obtener información de distintos sensores proporcionando la funcionalidad a los clientes a través de una *API de mensajes*.

- El servidor *Otos* proporciona acceso a sensores como: láser, infrarrojos o sónar. Su uso es más específico de un robot y no se utiliza en nuestra aplicación de seguridad.

- El servidor *Oculo* se encarga de los sensores de imagen. Permite capturar imágenes de las cámaras conectadas al equipo donde se ejecute. Estas imágenes son enviadas por red en forma de mensajes de texto, indicando el tamaño de la imagen, si es en color o en escala de grises, y la información de cada canal de color RGB de cada uno de los píxeles. Este servidor es imprescindible si las cámaras no pueden conectarse localmente a un ordenador. Oculo también es capaz de obtener información sensorial de cuellos mecánicos, a los que se puede acoplar una o varias cámaras para poder cambiar la orientación de las mismas.

3.3. Bibliotecas auxiliares

En conjunción con la plataforma software "jde.c" se han utilizado también dos bibliotecas adicionales en este proyecto fin de carrera. La primera de ellas es *biblioteca de geometría proyectiva Progeo* utilizada para operar con un espacio tridimensional simulado. La otra es la *biblioteca de interfaces gráficas XForms*.

3.3.1. Progeo

La biblioteca de geometría proyectiva Progeo es utilizada en la aplicación para relacionar el mundo de las imágenes (2D) con el mundo real (3D).

Esta relación se consigue gracias a dos funciones principales:

- *Proyectar*. Esta función permite realizar la proyección geométrica de un punto 3D del espacio en el plano imagen de una de las cámaras, obteniendo así un punto 2D perteneciente a ese plano. Con ese punto 2D es posible obtener el pixel de la imagen correspondiente.
- Función *Retroproyectar*. Esta función permite obtener la recta de proyección que une el centro óptico de una cámara con el punto 3D que representa un punto 2D de su plano imagen. Ese punto 2D se correponde con un pixel en la imagene de esa cámara y mediante esta función se puede llegar a obtener el punto 3D del que es proyección en el plano. Através de la intersección de dos rectas de proyección de un mismo punto 3D es posible obtener dicho punto resolviendo el sistema de ecuaciones.

Progeo se basa en un *modelo de cámara* para realizar estas transformaciones. En concreto se utiliza el *Modelo Pinhole* [Hartley y Zisserman, 2004], cuya figura descriptiva se puede observar en la figura 3.3.

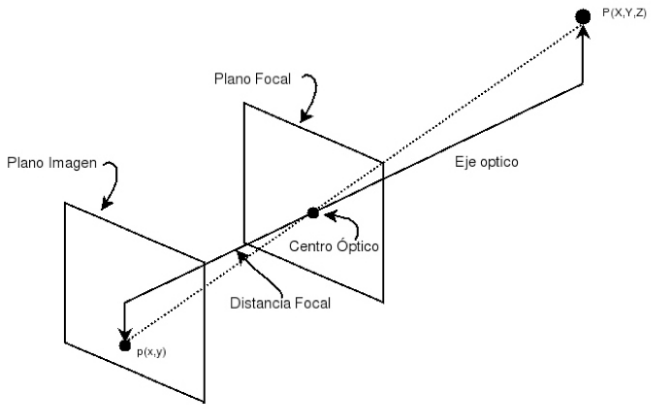


Figura 3.3: Modelo de cámara Pinhole

En este modelo se asume que cualquier punto $P(x, y, z)$ se proyecta en el plano de imagen através de otro único punto llamado *Centro óptico*. La recta que une el punto

P y el centro óptico se denomina *Línea de proyección* e intersecta al plano imagen justo en el pixel $p('x','y')$, que es la proyección de $P(x, y, z)$. El centro óptico esta situado a la *Distancia focal* del plano imagen. Este modelo lo completan el *Eje óptico*, que es una línea perpendicular al plano imagen y que atraviesa al centro óptico, y también el *Plano focal*, que es el plano perpendicular al eje óptico cuyos puntos no se proyectan en el plano imagen, incluyendo al centro óptico.

Progeo proporciona además los tipos de datos *Punto2D* y *Punto3D* para la representación de puntos en el plano y en el espacio.También proporciona los tipos *CamaraPinhole* y *CamaraStereoPinhole*. Ambos tipos se utilizan para simular el uso de cámaras y pares estéreo en el entorno tridimensional.

Para la correcta simulación del entorno virtualizado es necesario que las cámaras se encuentre calibradas. El proceso de calibración consiste en obtener los *parámetros intrínsecos* y *parámetros extrínsecos* de las mismas. En el caso de los parámetros intrínsecos es necesario conocer la *d*istancia focal en cada eje y el tamaño de del pixel central del plano imagen. Para los parámetros extrínsecos se necesita la posición 3D de la cámara y la orientación. El proceso de calibración se describe en el apartado 6.1.

3.3.2. XForms

La aplicación de seguridad *Watcher* utiliza el sistema de interfaces gráficas proporcionado por la plataforma software jde.c.

La biblioteca de interfaces gráficas XForms se apoya directamente en el servidor de imágenes X Window. La biblioteca proporciona una herramienta con la que poder crear la interfaz de botones, barras de desplazamiento, menús o cuadros de imágenes llamada *fdesign*.

Con XForms es posible crear ventanas directamente sobre el sistema X Window de todo sistema operativo Linux. No se apoya en ninguna otra bibliotecas o motores intermedios. Es por esto que una interfaz en Xforms es prácticamente compatible con cualquier sistema actual basado en unix del mercado, a diferencia de otras librerías más modernas y de diseño más atractivo. Se tiene la garantía de que la interfaz gráfica puede funcionar en todo sistema que incorpore este servidor gráfico (linux, macos, freebsd, etc).

Capítulo 4
Localización 3D con filtro de partículas

En este capítulo se expone una primera solución al problema de la localización 3D: el *filtro de partículas*. En las siguiente secciones se describen los fundamentos teóricos en los que se basa este método. Se describe también el diseño y la implementación del algorítmo y su integración en la aplicación.

4.1. Fundamentos teóricos del filtro de partículas

El *filtro de partículas* es una técnica probabilística basada en muestras que se apoya en métodos de Monte Carlo. Su eficacia se basa en mantener una representación muestreada de la distribución de probabilidad del objeto a lo largo de posibles soluciones [Fox *et al.*, 1999][Mackay, 1999][Barrera *et al.*, 2005] (figura 4.1).

El filtro de partículas es un algoritmo iterativo. Estima incrementalmente el estado multidimensional actual $X(t)$ mediante una colección de N muestras u observaciones $[obs(t), obs(t-1), obs(t-2)..., obs(t)]$. Cada una de estas muestras se corresponde con una partícula n_i que se representa por su posición (x_i) y su peso probabilístico (p_i) en cada instante, por lo que $n_i = \{x_i \mid p_i\}$. Las observaciones se relacionan con el estado multidimensional através de un modelo de observación probabilístico $p(X(t) \mid obs(t))$. Puesto que este estado puede ser dinámico se captura dicho dinamismo en un modelo de movimiento $p(X(t) \mid X(t-1))$. La población inicial se generá aleatoriamente distribuyéndose homogéneamente por todo el entorno. La evolución del conjunto dependerá por tanto de la predicción del movimiento y de la corrección de esta predicción mediante las observaciones obtenidas. En cada iteración se genera una nueva población de partículas que incorpora las nuevas posiciones de estas y sus creencias probabilísticas a raíz de las mediciones sensoriales.

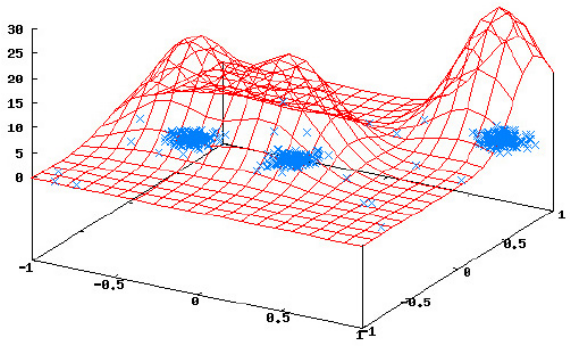


Figura 4.1: Muestreo de la función densidad de probabilidad.

Esta técnica puede desglosarse por tanto en tres etapas fundamentales:

- *Modelo de movimiento.* Se traslada la nueva población de partículas hacia nuevas posiciones. Las nuevas posiciones se calculan mediante un desplazamiento aleatorio en torno a la posiciónes medias de las partículas de la población anterior.
- *Modelo de observación.* Se calcula la probabilidad individual de cada partícula gracias a las últimas observaciones sensoriales obtenidas.
- *Remuestreo.* Se genera una nueva población de partículas a partir de las probabilidades obtenidas en el modelo de observación.

El algorítmo no guarda memoria de poblaciones anteriores ni un historial de observaciones. Sólamente actúa sobre la población que se tiene en cada momento para generar una futura y que ésta se concentre en las posiciones del espacio más probables. De este modo, se utilizan las partículas como puntos de búsqueda en el espacio de soluciones.

4.2. Diseño general y algoritmo

A continuación se expone brevemente las características de un filtro de partículas aplicado al problema de la localización 3D. También se describe el diseño de la implementación del filtro de partículas y la descripción de la interfaz de la aplicación de seguridad *Watcher*.

4.2.1. Filtro de partículas aplicado a la localización 3D

Una vez que se conocen los fundamentos del filtro de partículas se presenta en este apartado la adaptación de este método al problema de la localización 3D.

En esta implementación del filtro de partículas se pretende *estimar la posición 3D* de un objeto o una persona dentro de una habitación de gran volumen. Por tanto, cada una de las partículas representa una posible posición tridimensional de dicho objeto en el entorno. El estado x_i de una determinada partícula se corresponde con un *punto 3D de coordenadas (x,y,z)*. La verosimilitud asociada a ese estado p_i será calculada mediante la compatibilidad de esa partículas con las observaciones sensoriales. En este caso, los *sensores utilizados* son cuatro cámaras situadas en los rincones de la habitación. Una determinada partícula será compatible cuando *registre color coincidente* con el color buscado o *registre movimiento* en cada una de las cámaras utilizadas.

En esta técnica *no se utiliza triangulación explícita* para la correspondencia de objetos en las distintas imágenes de las cámaras. La naturaleza de la propia técnica ya realiza estas triangulaciones implícitamente a lo largo del desarrollo del algoritmo. Para la estimación de un estado 3D del objeto se utiliza solamente la información incompleta de las imágenes 2D, realizando las *operaciones directamente sobre la posición* de cada partícula pero no sobre posiciones derivadas.

4.2.2. Diseño de la aplicación

La aplicación *Watcher* se compone de un conjunto de hebras o *esquemas* que se asientan sobre las herramientas de la plataforma software jde.c. *Watcher* se compone principalmente de tres esquemas, dos perceptivos (*filtroparticulas* y *filtromoscas*) y un tercero de servicio y actuación *guixwatcher*.

El esquema del filtro de partículas está implementado como el esquema *filtroparticulas*, el cual forma parte de la aplicación de seguridad como un elemento perceptivo del mismo. Estos esquemas se apoyan sobre jde.c para obtener las imágenes de las cámaras conectadas al sistema además de otros elementos necesarios para la ejecución (figura 4.2). La aplicación se basa también en la biblioteca de geometría proyectiva Progeo para simular un entorno 3D con cámaras y elementos de la sala.

El esquema *guixwatcher* incorpora la interfaz de la aplicación, que proporciona la posibilidad de visualizar los resultados y ajustar el funcionamiento de los filtros de localización utilizando un simulador 3D (figura 4.3). Este simulador utiliza nuevamenta la biblioteca Progeo y permite mostrar las poblaciones de partículas y moscas en una *cámara virtual*, en la que se generan imágenes simuladas para observar dichas muestras en pleno proceso de localización de un objeto. Igualmente, los esquemas del filtro de

partículas y filtro de moscas pueden hacer uso de este simulador para auto-generarse imágenes simuladas.

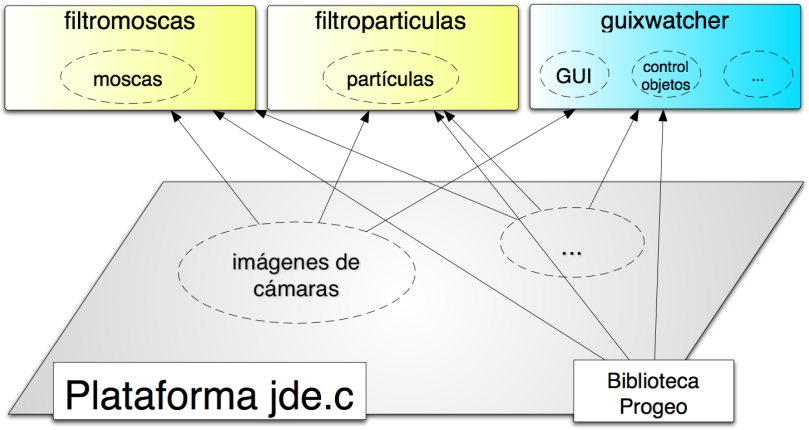


Figura 4.2: Diseño general de la aplicación de seguridad Watcher.

Es posible por tanto *probar y depurar las dos técnicas sin la necesidad de un escenario de pruebas*. Igualmente, el simulador proporcionado permite solapar objetos virtuales sobre las imágenes de las cámaras reales y también tener un *control de estos objetos 3D*. Para recrear el entorno tridimensional el sistema es capaz de *interpretar un fichero con un mapa del entorno virtual* en el que le vendrán especificadas los *parámetros intrínsecos y extrínsecos* de las cámaras y una descripción completa de la sala donde se realizará el seguimiento 3D.

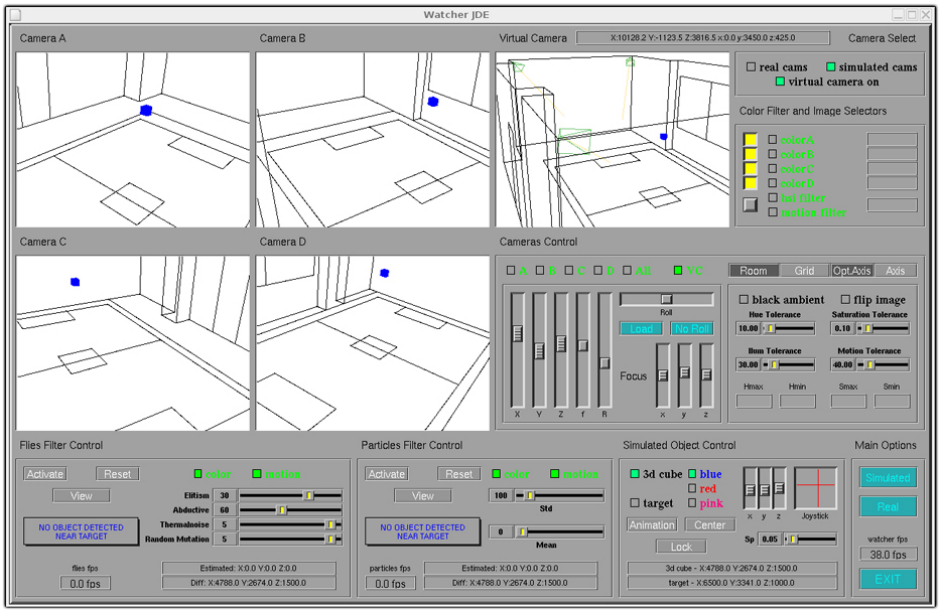


Figura 4.3: Simulador 3D de la aplicación de seguridad Watcher

Otra funcionalidad de la interfaz gráfica son los *eventos generados por ratón*. Estos eventos proporcionan gran funcionalidad a la aplicación, permitiendo *mover las cámaras del simulador virtual*, *cambiar su orientación* o *rotarlas*. En cuanto al patrón de color que seguirá en la localización 3D por color, estos eventos permiten *seleccionar con el puntero del ratón el color de seguimiento* directamente sobre las imágenes recibidas de las cámaras.

La aplicación de seguridad *Watcher* cuenta con la funcionalidad de alertar al usuario si detecta un intruso o un objeto no permitido dentro de una *zona de la habitación declarada como zona a proteger*. Esta función es posible gracias a que alguno de los dos esquemas perceptivos proporcionan al sistema el posición 3D estimada de la persona u objeto seguido. Estas alarmas pueden ser visuales si se tiene activada la interfaz gráfica, o bien sonoras utilizando el altavoz del ordenador. El sistema considera intruso u objeto no permitido al objetivo localizado que se aproxime a menos de 1 metro de la zona no permitida.

El proceso de seguimiento tridimensional requiere que las cámaras utilizadas hayan sido previamente calibradas y ajustadas al espacio virtual 3D, tal y como se detallará en el apartado 6.1. Es necesario recalcar que la interfaz gráfica de la aplicación *únicamente es necesaria para controlar y visualizar* adecuadamente la aplicación de seguridad. Las técnicas de localización son capaces de realizar un seguimiento de un objeto sin la necesidad de ningún elemento de la interfaz.

A continuación se detallan algunos de los controles que proporciona la interfaz gráfica para la visualización y ajuste de los algoritmos de localización:

- Cámaras de visualización *"Camera A, Camera B, Camera C y Camera D"*. Se encargan de mostrar en todo momento las imágenes que proporciona jde.c o las imágenes del simulador, asi como los resultados visuales que vayan apareciendo.
- Cámara de visualización virtual *"Virtual Camera"*. Se encarga de mostrar las imágenes de la cámara virtual, así como los resultados visuales que vayan surgiendo.
- Botones opcionales *"Real Cameras, Simulated Cameras, Virtual Camera On"*. Real Cameras se encuentra por defecto activado y permite elegir visualizar imágenes reales provenientes de JDE. Simulated Cameras permite activar el

simulador 3D. Virtual Camera On activará la visualización de la quinta cámara virtual.

- Barras de desplazamiento horizontales *"Hue Tolerance, Sat Tolerance, Ilum Tolerance y Mot Tolerance"*. Se encargan de ajustar manualmente los valores de las tolerancias HSI y de movimiento para el filtro de color y el filtro de movimiento.

- Barras de desplazamietno verticales *"X, Y, Z, f, R, x, y, z"*. Mediante estas barras de desplazamiento se pueden configurar los parámetros intrínsecos e extrínsecos de cada una de las cámaras cuando está activado el simulador 3D. También se puede modificar en cada momento la quinta cámara virtual.

- Controles de objetos simulados *"Cube3D, Target3D, Animation, Center, Blue, Red, Pink,..."*. Mediante estos controles se consigue introducir en el entorno la aparición de objetos 3D simulados, bien sea sobre imágenes reales o sobre el simulador. Con ellos se puede controlar la posición, el color o el movimiento automático de los mismos.

La descripción de los controles relacionados con cada algoritmo de localización se puede encontrar en la sección 4.7 y en la sección 5.6.

4.2.3. Esquema "filtroparticulas"

El *esquema filtroparticulas* implementa el algoritmo de filtro de partículas utilizado en la aplicación de seguridad. Este esquema se comporta como un elemento perceptivo dentro de la aplicación general.

En primera instancia obtiene las imágenes que le proporciona la plataforma software jde.c o recrea el entorno tridimensional necesario (si se desea realizar una simulación) para después aplicar el algoritmo de filtro de partículas de manera autónoma. Gracias a que la generación del mundo 3D y el procesado conviven en el mismo hilo de ejecución, conseguimos una *sincronización perfecta* entre los dos procedimientos y *evitamos condiciones de carrera.*

Tras haber completado una iteración completa, el filtro de partículas proporciona a la aplicación general la nueva población de partículas expresada como un conjunto de puntos 3D (x,y,z) y la posición estimada del objetivo en milímetros. Gracias a esta información, el sistema podrá tomar diversas decisiones sobre la intrusión del objeto

en la zona restringida. Podemos interpretar el esquema del filtro de partículas como una *caja negra* con el aspecto externo de la figura 4.4:

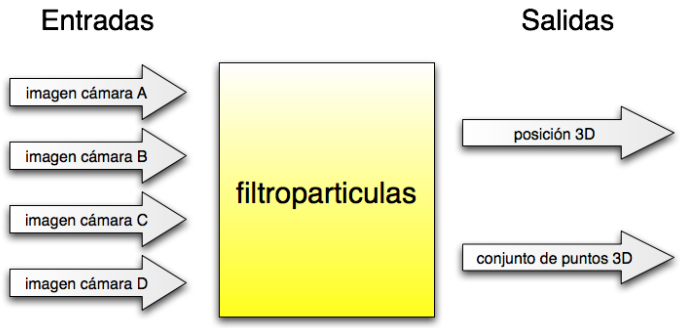


Figura 4.4: Representación del esquema filtroparticulas como caja negra.

El esquema del filtro de partículas consta de las etapas ya explicadas en el apartado 4.1: aplicación de un *modelo de movimiento*, aplicación de un *modelo de observación* y el *remuestreo*. Por último será necesario añadir una etapa adicional para el *cálculo de la estimación del objeto*. A continuación podemos observar un diagrama pseudocódigo con las operaciones realizadas en cada iteración (figura 4.5):

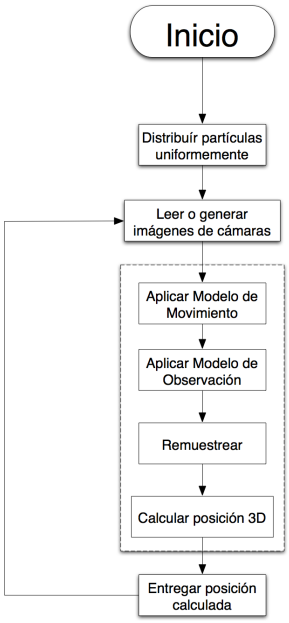


Figura 4.5: Diagrama pseudocódigo del esquema filtroparticulas.

Una descripción de la interfaz de *Watcher* relacionada con el filtro de partículas, se explica en la sección 4.7 de esta memoria.

4.3. Modelo de movimiento

La aplicación de un modelo de movimiento es la primera fase de esta implementación del filtro de partículas. En este paso se pretende aprovechar el conocimiento acerca de cómo se mueve el objeto que se está siguiendo para mejorar la estimación de su próxima posición. Por tanto, se *moverá la nube de partículas isótropamente* hacia nuevas posiciones compatibles con un *modelo de movimiento* elegido según esta intuición. Asignar este desplazamiento a las partículas antes de comprobar la información sensorial obtenida robustece al sistema frente a un posible cambio de ubicación del objetivo. La elección del correcto modelo de movimiento proveerá más previsiones favorables y menos (o ninguna) desfavorables. Por el contrario, un modelo que no se ajuste a la situación real entorpecerá el funcionamiento general del algorítmo.

En el caso concreto de esta implementación se ha optado por un *modelo gaussiano* (figura 4.6), donde la probabilidad disminuye a medida que nos alejamos del valor medio. La elección de este modelo se debe a que *desconocemos el tipo de movimiento que podrá tener el objeto o la persona*. No obstante, se toma en consideración que los movimientos pequeños son más probables que los grandes.

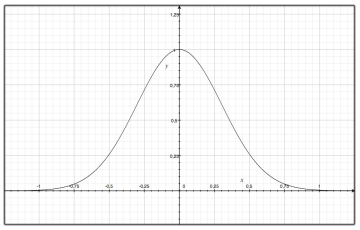


Figura 4.6: Distribución normal o gaussiana.

Las nuevas posiciones serán calculadas gracias a la suma de valores aleatorios compatibles con el modelo elegido. Las ecuaciones utilizadas para mover las partículas son las siguientes:

$$x_i(t) = x_i(t-1) + N(0, \sigma_m) \tag{4.1}$$

$$y_i(t) = y_i(y-1) + N(0, \sigma_m) \tag{4.2}$$

$$z_i(t) = z_i(z-1) + N(0, \sigma_m) \tag{4.3}$$

Los valores obtenidos serán compatibles entre los existentes en una distribución normal de media $\varepsilon_m = 0$ (centrada en la propia partícula) y desviación típica σ_m. La elección de este último parámetro supone obtener distintos comportamientos en la población de partículas y establece un *compromiso entre la velocidad de convergencia y el error residual* que se obtiene.

Utilizando *valor de σ_m pequeño* se obtendrán *movimientos pequeños y suaves* de la población. Los movimientos pequeños son útiles cuando el objetivo está localizado y relativamente quieto, sin embargo dificultan el seguimiento cuando éste se mueve velozmente. Por el contrario, un *valor de σ_m grande* provocará movimientos muy bruscos en las partículas permitiendo que exploren más espacio en menos tiempo. Estos movimientos son útiles cuando el objeto se mueve, pues es más fácil realizar el seguimiento y no perderlo. Sin embargo la estimación realizada del objeto será menos precisa debido a que las partículas están más dispersas. Estos efectos se pueden visualizar en el estudio de *Seguimiento tridimensional usando dos cámaras [Barrera y Cañas, 2004]*.

En la figura 4.7 se han representado del mismo color las poblaciones en idénticos instantes de tiempo. En ella se observa como la población (a), con σ_m pequeño, tarda demasiado tiempo en desplazarse al comportarse como una nube de puntos más densa, mientras que la otra (b), con σ_m grande, se mueve más deprisa al estar más dispersa.

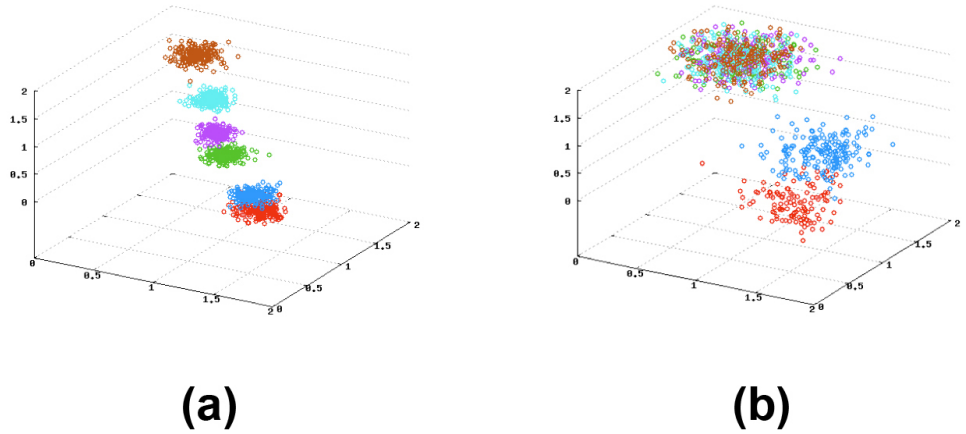


Figura 4.7: Evolución espacial de las partículas ante un movimiento.

La opción utilizada en esta implementación es la de un valor de σ_m *intermedio*, que permita tener gran capacidad de exploración y no distorsione enormemente la posición

estimada en el cálculo de la media.

4.4. Modelo de observación

En la segunda fase del filtro de partículas se calcula la probabilidad de la población de las partículas en función de las observaciones sensoriales obtenidas. Las distintas formas para realizar esta labor implican la elección de un determinado *modelo de observación*.

En esta implementación se ha optado por un *modelo de observación basado en el color y/o en el movimiento de los objetos*. De este modo, las imágenes de cada cámara serán analizadas buscando un color compatible con el buscado o movimiento entre la imagen actual y la anterior. Ambas características serán computadas de manera separada, de forma que cada partícula tendrá en cada momento una probabilidad asignada por el análisis del color y una probabilidad asignada por la comprobación del movimiento. A estas probabilidades las denominaremos *probabilidades asociadas al color o al movimiento*. En cada momento existirá también una *probabilidad total* de la partícula, que será calculada por combinación bayesiana de las dos anteriores. La probabilidad total de una partícula será muy alta cuando lo sea también en ambas observaciones.

El procedimiento genérico que realiza para procesar una determinada partícula en una de las imágenes es el siguiente:

1. Proyectar partícula 3D en el plano de la imagen.
2. Para el pixel sobre el que proyecta la partícula y los colindantes:
   a) Comprobar color y calcular probabilidad asociada.
   b) Comprobar movimiento y calcular probabilidad asociada.
3. Calcular probabilidad total de cada partícula.

La fase de actualización es, sin duda alguna, la parte más *costosa computacionalmente* de esta implementación del filtro de partículas, pues para completarla se debe repetir este bucle para todas las partículas en cada una de las imágenes de las cámaras. Finaliza cuando se ha conseguido actualizar todas las probabilidades totales de las partículas en cada imagen.

Primeramente se debe proyectar el punto 3D que representa la partícula en un pixel del plano imagen de la cámara. En este paso se utiliza la *función proyectar* de la biblioteca de geometría proyectica Progeo. Esta función se describe en el apartado 3.3.1.

Tras conocer el pixel de la imagen sobre el que proyecta la partícula se puede proceder al proceso de filtrado del mismo y al cálculo de sus probabilidades. Se analizarán también los pixeles vecinos que junto con el pixel central forman una *ventana de vecindad* (figura 4.8) de un tamaño determinado. En esta implementación del filtro de partículas se ha establecido una ventana de 5x5 pixeles.

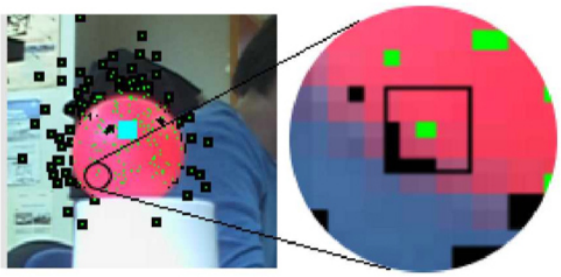


Figura 4.8: Ventana de vecindad 5x5 para el cálculo en el modelo de observación.

A partir de aquí es posible realizar *el cálculo de las probabilidades asociadas al color y movimiento* de una partícula. Para una partícula n_i, proyectada en en una determinada imagen de una cámara, sus probabilidades son las siguientes:

Para la *probabilidad asociada al color*:

$$(\text{color}_i \mid \text{img}_m) \quad = \quad \frac{\max(1,k)}{25} \tag{4.4}$$

Donde k equivale a cada uno de los píxeles de la ventana de vecindad que son compatibles con el patrón de color buscado en la imagen de la cámara c_m.

Se puede apreciar como la ecuación contempla que la proyección del punto 3D que representa la partícula se encuentre fuera del plano imagen de dicha cámara, en cuyo caso, la probabilidad de la partícula asociada al color será 1/25.

Para la *probabilidad asociada al movimiento*:

$$(\text{mov}_i \mid \text{img}_m) \quad = \quad \frac{\max(1,k)}{25} \tag{4.5}$$

Donde k equivale a cada uno de los píxeles de la ventana de vecindad que han registrado movimiento en la imagen de la cámara c_m.

En esta ecuación se tiene en cuenta nuevamente, que la proyección de la partícula no se encuentre dentro del plano imagen de dicha cámara, por lo que en este caso la probabilidad asociada al movimiento será 1/25.

La operación de extraer la información de color y movimiento, para proporcionar las imágenes filtradas, se realiza mediante un *filtro de color* y un *filtro de movimiento*. Ambos filtros se realizan *bajo demanda*, es decir, no se filtra la imagen completa sino que sólamente se procesan aquellos pixeles solicitados. De este modo se consigue un aumento en la eficiencia del proceso general. Estas dos técnicas serán descritas en los apartados 4.4.1 y 4.4.2 respectivamente.

El último punto del proceso consiste en calcular la *probabilidad total* de cada una de las partículas. Este cálculo se resume en el producto de las probabilidades totales asociadas al color y al movimiento para cada una de las cámaras. Para una determinada partícula $s_i$proyectada en M imágenes su probabilidad total se calcula mediante la siguiente ecuación:

$$p_i \quad = \quad \prod_{m=1}^{M} (\text{color}_i \mid \text{img}_m) * (\text{mov}_i \mid \text{img}_m) \tag{4.6}$$

4.4.1. Filtro de color

En los algoritmos de localización 3D implementados es posible realizar un seguimiento de un objeto basándose en el color del mismo. Estos algoritmos necesitan distinguir aquellas zonas de las imágenes que son compatibles con el patrón de color buscado. El filtro de color se encarga de realizar esta distinción, aportando a los algoritmos imagénes filtradas con el color seleccionado en la interfaz gráfica.

La plataforma software jde.c proporciona imágenes de 320x240 píxeles, y el color de cada pixel viene representado en el espacio de color RGB (rojo, verde, azul). Sin embargo, un filtro de color para el espacio RGB no es un filtro eficiente [de la Casa Puebla, 2005]. Pierde la robustez ante los posibles cambios de iluminación que se produzcan en las imágenes. Estos cambios de iluminación pueden generar variaciones muy grandes en los valores RGB, y pueden ocasionar que un pixel de color azul con mucha o muy poca iluminación no pase un filtro para el propio color azul. Por ello, en este proyecto se ha optado por un espacio de color más robusto frente a cambios de intensidad luminosa: el *espacio de color HSI* (figura 4.9). Un color en este espacio se representa mediante tres componentes:

- *Tinte* (Hue). Es el color puro propiamente dicho y se corresponde con el ángulo de la figura 4.9.
- *Saturación* (Saturation). Muestra la viveza de un color determinado y se representa en la figura 4.9 como el radio dentro del cono.
- *Intensidad* (Intensity). Es la iluminación del color y se representa como la altura dentro del cono de color.

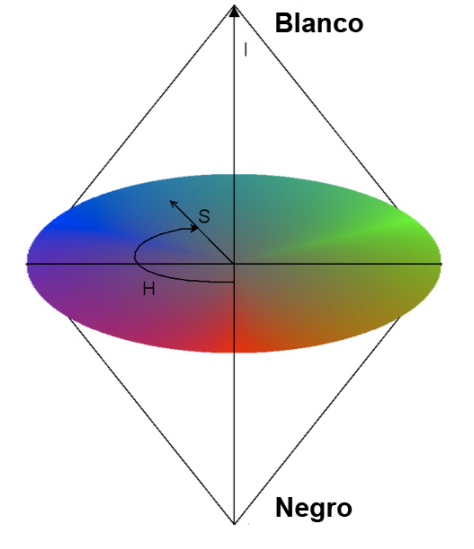


Figura 4.9: Espacio de color HSI.

Como se puede observar, el espacio HSI trabaja directamente con una magnitud relacionada con la iluminación del color, que se puede ignorar en los filtros trabajando únicamente con las componentes H y S. Este filtro es precisamente el que se ha utilizado

en este proyecto, y la relación existente entre RGB y HSI se puede observar en las siguientes ecuaciones:

$$H = cos^{-1}\left(\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{4.7}$$

$$S = 1-\frac{3}{(R+G+B)}min\,(R,G,B) \tag{4.8}$$

$$I = \frac{1}{3}(R+G+B) \tag{4.9}$$

Donde R, G, B son la cantidad de rojo, verde y azul, respectivamente, de la imágen con valores entre 0 y 1. H nos proporciona el tinte expresado como un ángulo entre 0 y 2π, S la saturación con valores entre 0 y 1, y la intensidad luminosa I entre 0 y 1.

Por último, *para realizar el filtro de color HSI*, se establecen unos umbrales máximos y mínimos por cada componente (Hmin, Hmax, Smin, Smax). Estos umbrales serán definidos a partir un color elegido para el filtro y una tolerancia establecida (*tolerancias de color*). Aquellos píxeles con componentes dentro de estos umbrales se consideran del color objetivo y computan en el cálculo de probabilidades. Los píxeles que no superen alguno de los umbrales serán redibujados en escala de grises en la interfaz gráfica de la aplicación o bien, no se dibujarán (ver figura 4.10).

Figura 4.10: Filtro de color para rojo - Imagen sin filtrar (a) Imagen filtrada (b).

4.4.2. Filtro de movimiento

El filtro de movimiento nos permite *detectar cambios de color en la imágen debido al desplazamiento* de los objetos. La información extraída de la aplicación de

este filtro permite a los algoritmos de localización 3D estimar la posición de objetos que se mueven.

La aplicación de este filtro consiste restar a cada pixel su valor anterior de RGB para cada una de las imágenes recibidas. Para ello, es necesario almacenar la imágen anterior de cada una de las cámaras que se vayan a analizar, de forma que se pueda contrastar cada pixel con él mismo en el instante anterior. De esta forma, la diferencia entre ambas imágenes dará como resultado los píxeles que han sufrido variación debida al movimiento.

Para la *implementación del filtro de movimiento*, al igual que se realiza en el filtro de color, se establece una (*tolerancia de movimiento*). Si la diferencia de un determinado pixel con su valor anterior es mayor que la tolerancia de movimiento, se considera que en ese pixel hay movimiento y por tanto computa en el cálculo de probabilidades. Si es así, en la interfaz gráfica este pixel será redibujado de un color sólido acordado para ser reconocido posteriormente. Por el contrario, los píxeles que no sobrepasen el umbral se dibujarán del mismo color que eran en la nueva imágen o bien, no se dibujarán (ver figura 4.11).

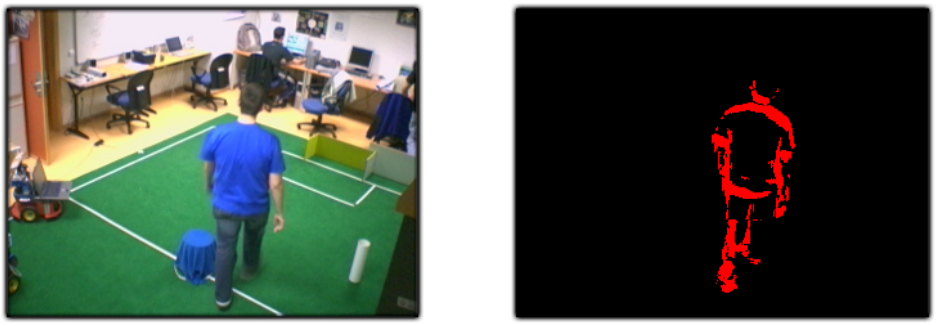


Figura 4.11: Filtro de movimiento - Imagen sin filtrar (a) Imagen filtrada (b).

4.5. Remuestreo

La tercera y última fase del filtro de partículas es la *fase del remuestro*. Su función se resume en generar una nueva población de partículas intentando que el máximo número de ellas tenga la mayor probabilidad posible. Este paso es el que incorpora la inteligencia a nuestro algorítmo y afina la búsqueda de la solución.

La nueva generación de partículas se realiza *muestreando aleatoriamente* entre las partículas de la población actual. Las partículas con más probabilidad deben proporcionar más descendientes para la nueva población. Este remuestreo se consigue gracias al *algoritmo de la ruleta*. El efecto neto es que las partículas de la nueva población se acumulen cerca de las posiciones más prometedoras identificadas en la población anterior.

El algorítmo consiste en elegir uniformemente una probabilidad entre 0 y la probabilidad total acumulada entre todas (cada punto del eje y de la figura 4.12 (b) corresponde a alguna partícula del eje x de la figura 4.12 (b)). Dicha probabilidad debe ser búscada en la lista de probabilidades acumuladas para detectar cual es la partícula que corresponde a esa acumulación (figura 4.12). Esta búsqueda se realiza mediante una *búsqueda binaria* para acelerar este paso. Esta búsqueda se ejecuta una vez por iteración. La partícula a la que corresponda esa probabilidad pasará directamente a la nueva población.

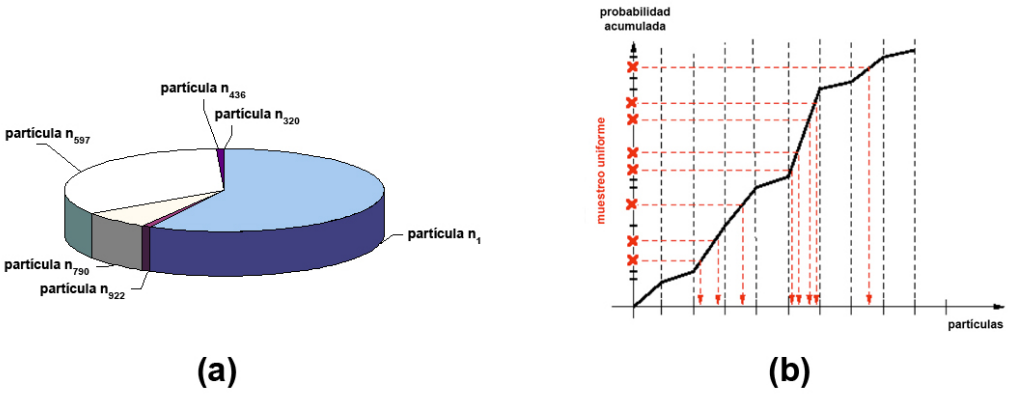


Figura 4.12: Algoritmo de la ruleta - Distribución aleatoria de partículas (a) y correspondencia probabilidad-partícula (b).

Este proceso se repite tantas veces como partículas se quieran generar para la siguiente población. El final del *proceso dará como resultado* una nueva población situada en torno a posiciones más probables, aunque *no está garantizado*. Puede ocurrir que la nueva generación descienda de una población que tenía una probabilidad general baja. En este caso, la nueva población estará posicionada en torno a posiciones de poca probabilidad, y se dice que *la población ha degenerado*.

4.6. Cálculo de la posición 3D

Por último, el filtro de partículas deberá calcular la posición 3D estimada del objeto seguido. Tras haber calculado la probabilidad total de las partículas y generado la nueva población, podemos calcular la posición estimada del objeto, realizando una media de las coordenadas de cada una de estas muestras.

Para una estimación más verosimil se utiliza una *media ponderada* en lugar de la media tradicional, pues no resultaría justo que una determinada partícula, alejada de la mayoría, pudiera deteriorar la estimación realizada. De este modo, aquellas partículas con mayor probabilidad contribuyen en mayor proporción en el cálculo de la media.

Tras haber calculado la posición estimada usando la media ponderada, el esquema filtro de partículas se encarga de entregar a la aplicación de seguridad *Watcher* la situación del objeto, así como una representación en forma de puntos 3D de la población de partículas.

4.7. Visualización de resultados

Tras recibir la posición estimada del filtro de partículas y la nube de partículas en forma de puntos 3D, la interfaz de *Watcher* podrá visualizar estos resultados. Esta interfaz ha sido diseñada de manera que permita visualizar la posición estimada en coordenadas, o también si se desea, la nube de partículas en torno al objeto en cada una de las imágenes (figura 4.13).

Las características de la interfaz con respecto al filtro de partículas son las siguientes:

- Botón principal *"Activate"*. Se encarga de iniciar y detener el filtro de partículas.
- Botón principal *"Reset"*. Se encarga de reiniciar el filtro manualmente, distribuyendo uniformemente las partículas por todo el espacio.
- Botón principal *"View"*. Se encarga de activar y desactivar la visualización de la nube de partículas en las imágenes. Este botón permitirá la visualización contínua de toda la población de partículas y su evolución.
- Botón Opcional *"Color"*. Se encarga de activar y desactivar el procesado de información de color de las imágenes. Si está desactivado, el filtro de partículas no funciona basándose en información de color.

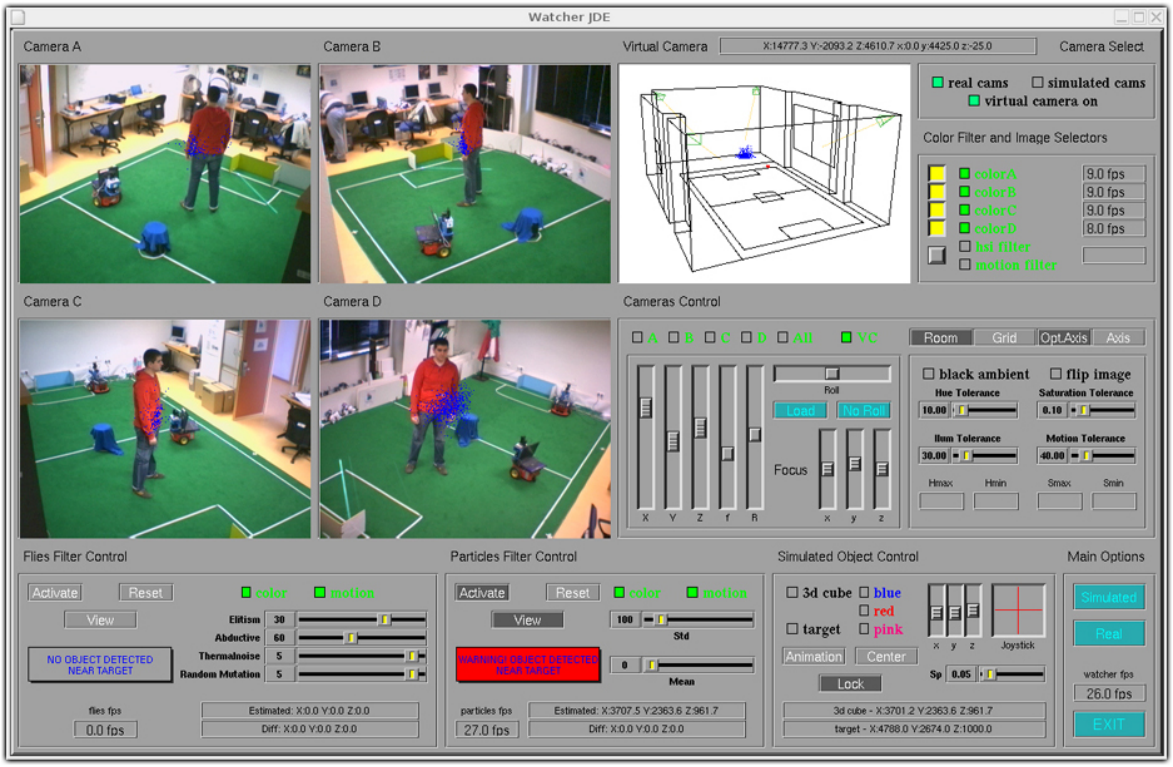


Figura 4.13: Interfaz gráfica de Watcher. Filtro de partículas.

- Botón Opcional *"Motion"*. Es análogo al anterior pero para la información de movimiento.
- Barras de desplazamiento horizontales *"Mean y Std"*. Se encargan de modificar manualmente la media y la desviación típica aplicada en el modelo de movimiento.
- Recuadro de información *"Estimated Position"*. En él se visualizará en todo momento la posición 3D expresada en coordenadas cartesianas en milímetros.
- Recuadro de información *"WARNING"*. Este recuadro aparecerá en color gris si el objetivo localizado se encuentra fuera de la zona restringida. Por el contrario, parpadeará en color rojo si se invade la zona protegida.
- Recuadro de información *"Particles fps"*. En él se muestran las iteraciones por segundo que está ejecutando el filtro de partículas en cada instante.

Capítulo 5
Localización 3D con filtro de moscas

En este capítulo se plantea otra solución al problema de la localización 3D visual mediante una técnica basada en muestras no probabilística : el *filtro de moscas*. Esta técnica, no sujeta a ningún modelo matemático de probabilidad, consiste en un *algoritmo evolutivo* que genera las nuevas poblaciones de muestras mediante la utilización de operadores genéticos.

5.1. Fundamentos teóricos del filtro de moscas

El filtro de moscas se fundamenta en los llamados *algoritmos genéticos o evolutivos*. Estos algoritmos se caracterizan por alcanzar una solución al problema planteado combinando las propiedades de los miembros del conjunto muestral para obtener nuevas generaciones más robustas. Estos algoritmos se utilizan a menudo en tareas de optimización, diseño de controladores borrosos o sistemas de aprendizaje automático.

El filtro de moscas consiste en una aplicación de estos algoritmos, y fue utilizado primeraremente para la reconstrucción 3D [Louchet, 2000]. En cada implementación del filtro se cuenta con un conjunto o población de N moscas donde cada mosca n_i tiene asociada un determinado peso o salud h_i. Cada una de las moscas es una muestra con un estado $x_i(t)$ y su salud h_i se actualiza en cada iteración del proceso. Dependiendo del propósito del algoritmo, una determinada mosca puede ser en si misma una posible solución (enfoque *Pittsburg*) o bien sólo una contribuyente a la misma (enfoque *Michigan*).

La nueva población se genera gracias a los *operadores genéticos*. Estos operadores actúan como condicionantes para la perpetuación de la población. Se encargan de generar la población en el siguiente instante utilizando en el proceso las moscas con mejor representación cromosómica. Cada nueva generación reune las mejores

propiedades de la población anterior. Estas propiedades dependerán del propósito del algoritmo, y los operadores genéticos pueden ser diseñados expresamente para captar estos rasgos de manera más eficiente.

Existen varios *operadores genéticos clásicos* que se suelen utilizar siendo adaptados al contexto específico de cada problema. Algunos de estos operadores son: *Mutación aleatoria*, *Ruido térmico*, *Cruce* o *Repulsión*. La Mutación aleatoria permite generar moscas en cualquier posición del entorno. El Ruido térmico genera moscas alrededor de otra mosca determinada. El Cruce permite combinar las características de dos moscas para generar una nueva. La Repulsión evita que dos moscas sean generadas en la misma posición del entorno.

Este algoritmo se divide fundamentalmente en las siguientes fases:

1. *Generación de la próxima población*. Se aplican los operadores genéticos para obtener la nueva población de moscas.

2. *Computación de la salud*. Se calcula la salud para cada una de las moscas en función de las observaciones realizadas.

El algoritmo solamente necesita la salud o las propiedades de las moscas de la población anterior para poder generar la próxima generación. No necesita calcular posiciones derivadas para la búsqueda de una solución sino sólo posiciones computables en un punto. Los cálculos de salud previos a la anterior población no son condicionantes para generar la nueva.

5.2. Diseño general

Una vez introducidos los fundamentos del filtro de moscas se detallará en esta sección el diseño general y la integración del filtro de moscas con el resto de los módulos que componen la aplicación de seguridad Watcher.

El objetivo del filtro de moscas en este problema consiste en *obtener la posición 3D* de un objeto y realizar un *seguimiento 3D* del mismo por todo el espacio vigilado. El filtro de moscas se comporta como un nuevo *esquema* dentro de la aplicación de seguridad, que proporcionará la situación exacta del objeto con la que el sistema podrá determinar si se encuentra dentro de una zona restringida y, en tal caso, alertar de dicha intrusión. Por tanto, se usa el algoritmo evolutivo para explorar el espacio de

posibles soluciones al problema de la localización visual 3D.

El filtro de moscas necesita en cada iteración información sensorial para calcular la posición 3D del objeto. Estas observaciones serán las *imágenes 2D provenientes de las cuatro cámaras de la habitación*, que se obtendrán gracias a la plataforma software jde.c sobre la que se apoya, o serán generadas por el propio esquema gracias al simulador 3D implementado. En cada iteración se procesará por completo cada una de las imágenes extrayendo la información de color y movimiento. Esta información será utilizada para generar la nueva población de moscas, cuyas nuevas posiciones dependerán del color de los objetos observados o de si existe movimiento en la imagen.

Tras haber finalizado cada iteración, el esquema realizará el *cálculo de la posición 3D estimada* a partir de la población de moscas y lo proporcionará junto con una representación de las moscas en forma de puntos 3D a la aplicación general. Así, la aplicación Watcher conocerá la estimación de la posición 3D del objeto, podrá mostrarla visualmente y también determinar una posible acción en función de esta posición 3D. Una vez realizado este proceso, se procederá al cálculo de una *nueva generación de moscas*.

Por tanto, desde un punto de vista de *caja negra* el esquema puede entenderse como un elemento de la aplicación presentada en el apartado 4.2.2 que presenta el aspecto exterior que muestra la figura 5.1:

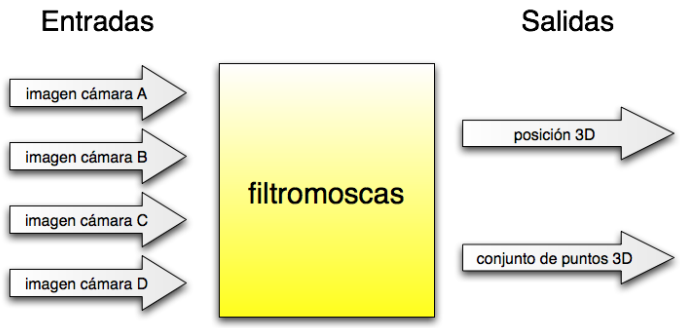


Figura 5.1: Representación del esquema filtromoscas como caja negra.

El proceso de seguimiento tridimensional requiere que las cámaras utilizadas hayan sido previamente calibradas y ajustadas al espacio virtual 3D, tal y como se detallará en el apartado 6.1. Es necesario destacar, que durante dicho proceso el filtro de moscas no realiza ninguna triangulación explícita, ni tampoco efectua ningun procedimiento

de correspondencia entre los objetos de las distintas imágenes de cada cámara. No es necesaria debido a que la vinculación de los objetos entre las distintas imágenes se realiza de forma implícita explorando de forma explícita el espacio 3D de posibles soluciones.

Para visualizar los resultados obtenidos con este filtro y ajustarlo, la aplicación de seguridad *Watcher* incorpora en la interfaz gráfica varias herramientas de visualización y control del filtro, con el que se podrá visualizar la posición estimada en forma de coordenadas e incluso las propias moscas en el simulador 3D de la cámara virtual.

5.2.1. Esquema "filtromoscas"

Una vez descrito el diseño general, se detallará en esta sección la adaptación del algoritmo de moscas al problema de la localización 3D en el *esquema filtromoscas*. El filtro de moscas fue inicialmente diseñado conforme al problema de la reconstrucción 3D [Louchet, 2000][Louchet, 2001][Louchet *et al.*, 2002]. No obstante, para agilizar la búsqueda de las posibles soluciones y aprovechar mejor los recursos disponibles, se ha realizado un nuevo diseño del algoritmo.

En este nuevo diseño, cada una de las moscas es una solución en si misma (enfoque *Pittsburg*) por lo que se corresponde con un *punto 3D* de coordenadas (x,y,z). Cada mosca cuenta además con una salud h_i que indicará cómo de buena es esa solución teniendo en cuenta las observaciones en las imágenes de las cuatro cámaras del sistema. Una mosca gozará de buena salud si es compatible con el color buscado y con el movimiento en las cuatro imágenes.

En cuanto a la generación de la siguiente población de moscas, este diseño del filtro de moscas incorpora algunos de los operadores genéticos clásicos e introduce un nuevo operador que resulta fundamental para la solución del problema.

El esquema que materializa el diseño realizado del filtro de moscas se denomina *filtromoscas*. Este esquema se divide en tres subprocedimientos principales: *Computación de la salud*, *Cálculo de la posición 3D* y *Generación de la próxima población*. Cada uno de estos apartados se ejecuta iterativamente en el orden mostrado en la figura 5.2.

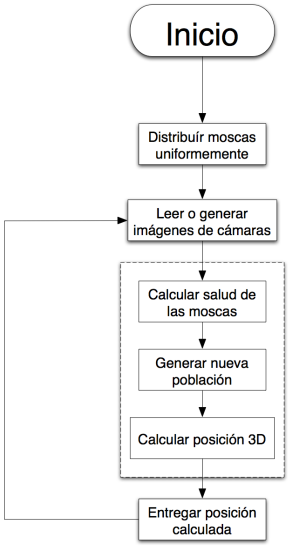


Figura 5.2: Diagrama pseudocódigo del esquema filtromoscas.

Este caracter iterativo del esquema encaja a la perfección con la propia naturaleza del filtro de moscas. El algoritmo comienza generando una *primera población totalmente aleatoria de las moscas*. De este modo, la primera población estará distribuída uniformemente por todo el espacio de la habitación.

5.3. Computación de la salud

La primera etapa del filtro de moscas consiste en la asignación de salud de cada una de las moscas de la nueva población población generada en la iteración anterior. En el caso de ser la primera iteración, el cálculo de salud se realiza sobre una población generada aleatoriamente. Al asignar salud a las moscas se podrán distinguir aquellas que se encuentran cerca de un objeto compatible con las observaciones para tenerlas en cuenta en el cálculo de la posición 3D.

A diferencia de cómo se realizaba en el filtro de partículas, en este algoritmo se realizará un *filtrado completo de color y movimiento de cada imagen*. Mediante este filtrado completo se aprovechará la potencia del *operador genético abducción*, explicado en el siguiente apartado. No obstante, para obtener un análisis más exacto de la salud de las moscas, se empleará también una *ventana de vecindad* de 5x5 píxeles, alrededor del pixel central tal y como ya se empleaba en el filtro de partículas. El funcionamiento del filtro de color y movimiento puede verse en detalle en los apartados 4.4.1 y 4.4.2, mientras que la ventana de vecindad se analiza en la sección 4.4, pertenecientes al capítulo 4.

Para el cálculo de la salud se tendrá en cuenta la información de color y movimiento registrada en las imágenes capturadas de cada una de las cámaras. El proceso para cada imagen de cada cámara es el siguiente:

1. La imagen recibida de la cámara c_m es procesada completamente por el *filtro de color* y el *filtro de movimiento*, para generar dos imágenes filtradas. Durante el proceso de filtrado de cada una de las imágenes, se almacenará en una lista aquellas posiciones que han sido compatibles con el color buscado o que han registrado movimiento.

2. A continuación, para cada una de las moscas n_i, se procederá de la siguiente manera:

   a) Se *proyectará el punto 3D* que representa la mosca sobre el plano imagen de la cámara c_m de igual forma que se realizó en el apartado 4.4. De esta forma conseguiremos las coordenadas (x,y) en el plano imagen donde se visualiza la mosca. Como ya se mencionó previamente, en este paso es fundamental haber realizado una calibración de las cámaras de la habitación.

   b) *Calcular la salud asociada al color y al movimiento para la mosca* $n_i$en cada una de las cámaras, analizando la ventana de vecindad en torno al pixel que indican las coordenadas en el plano. Para ello se observará si la zona de vecindad de esa determinada imagen es compatible con el patrón buscado o si hay movimiento, mirando para ello en las imágenes filtradas por color y movimiento previamente generadas. La salud asociada al color y al movimiento de una mosca para la imagen de la cámara c_m se calculan con las mismas ecuaciónes que se calculaban las probabilidades asociadas en el filtro de partículas, como se puede ver en la sección 4.4.

3. *Acumular salud de las moscas.* Tras haber calculado la salud de todas las moscas asociadas tanto por color como por movimiento de la cámara c_m acumularemos la salud de estas en la lista de *salud total*. Esta lista equivale al análisis de la salud de color y movimiento de cada mosca en las M cámaras de la aplicación, que se puede calcular mediante la siguiente ecuación:

$$h_i = \prod_{m=1}^{M} (\text{color}_i \mid \text{img}_m) * (\text{mov}_i \mid \text{img}_m) \tag{5.1}$$

Donde h_i es la salud total asociada a la mosca n_i.

Tras haber calculado la lista en cada iteración, el filtro de moscas se encarga de ordenar la nueva población de moscas de mayor a menor salud total.Se utiliza para ello el algoritmo de ordenación *quicksort*. De esta forma, el *operador genético elitismo* podrá seleccionar las moscas de mayor salud directamente.

5.4. Generación de la próxima población

En esta etapa del filtro de moscas se trata de *generar una población de moscas* gracias a las observaciones realizadas en las imágenes de las cámaras en la anterior iteración del filtro.

Este procedimiento se efectúa utilizando los *operadores genéticos*. Estos operadores se encargan de captar las mejores propiedades de las moscas de la población anterior de forma que se consiga una nueva que permita una localización más exacta del objeto seguido.

Como ya se mencionaba en la sección de 5.1, la correcta creación de *nuevos operadores genéticos* así como el ajuste de los *operadores genéticos clásicos* permitirán llegar a la solución del problema. La elección de los operadores genéticos de esta implementación esta basada en el nuevo diseño que se ha realizado del filtro de moscas para adaptarlo al problema de la localización 3D. Teniendo en cuenta los operadores utilizados en la reconstrucción 3D [Louchet, 2000][Louchet, 2001][Louchet *et al.*, 2002] se ha optado por conservar algunos de los operadores existentes y de introducir uno totalmente nuevo:

- *Operador de generación aleatoria*. Este operador selecciona aleatoriamente una mosca de entre todas las de la población y la posiciona aleatoriamente en otro lugar de la habitación. No mira para ello la salud de la mosca seleccionada, sino que de manera aleatoria determina una nueva posición y ubica una mosca en dicho lugar. Este operador permite *explorar uniformemente el espacio* de la habitación. Sus probabilidades de éxito son bajas pero permite evitar los máximos locales en la búsqueda de la solución.

- *Operador genético de ruido térmico*. Este operador se encarga de distribuir la mosca seleccionada en una posición aleatoria cercana a la que ocupaba en la población anterior. Permite *explotar una zona* distribuyéndose como ruido

térmico en torno a la misma. A diferencia de la generación aleatoria, este operador se basa en las posiciones de las moscas de la población anterior.

- *Operador genético de elitismo.* Este operador elige las moscas con mayor salud de la población anterior y permite que permanezcan en la nueva población sin que sufran ningún cambio. La idea del operador consiste conservar una estabilidad en la nueva población no olvidando aquellas posiciones que ya eran muy compatibles en el estado anterior.

- El operador genético creado desde la particularidad del problema de la localización 3D se trata del *operador genético de abducción.* La semántica del término abducir indica la posibilidad de dar por supuesta cierta premisa, dada otra premisa de caracter superior que nos permite derivar en esa suposición. En concreto, dada la implicación $a \rightarrow b$ podemos afirmar que *de a deducimos b* pero también que *de b abducimos a.*

Aplicado a nuestro problema, la abducción nos permitirá suponer que si en una imagen de una cámara visualizamos un objeto que es compatible con el patrón de color buscado o este se encuentra en movimiento, el objeto en sí se encontrará en algun punto de la línea de proyección que une el centro óptico con el punto donde intersecta al plano imagen. De este modo, es razonable pensar que generando nuevas moscas a lo largo de esa línea de proyección, algunas puedan estar cerca del objeto (figura 5.3).

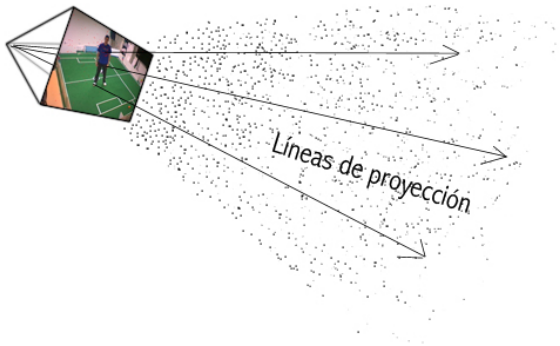


Figura 5.3: Moscas generadas a lo largo de líneas de proyección.

Basándose en este concepto, el *operador genético de abducción* genera nuevas moscas posicionándolas a lo largo de la línea de proyección de cada una de las cámaras con el objeto detectado. En cada iteración, siempre que se

detecte algún objeto compatible con el color o que se encuentre en movimiento, podrá visualizarse un haz de moscas distribuidas a lo largo de las líneas de proyección de cada cámara. De este modo, busca en zonas del espacio de soluciones compatibles con las observaciones sensoriales.

Para que el operador de abducción pueda funcionar correctamente y se pueda aprovechar al máximo de sus posibilidades, es necesario contar con una *lista de píxeles* de cada imagen que sean compatibles con el patrón de color buscado o que hayan registrado movimiento. Esta operación viene facilitada en el paso de computación de la salud, ya que se realiza un *filtro completo de las imágenes* para buscar esas características. Si no se cuenta con dicha lista de píxeles, como sucede cuando se inicia por primera vez el filtro, el operador de abducción cede el porcentaje de moscas que tenía asignado a los otros operadores, para facilitar de este modo la búsqueda más eficiente de un posible objeto.

Parte del coste computacional del filtro de moscas se concentra en realizar un filtrado completo de cada una de las imágenes. Sin embargo, su causa está justificada para favorecer a este operador genético.

Existen otros *operadores clásicos* tales como el *operador genético de cruce* o el *operador genético de repulsión*. Sin embargo, no se han utilizado en esta implementación debido a que están más orientados al problema de la reconstrucción 3D [Louchet, 2000][Louchet, 2001][Louchet *et al.*, 2002].

En esta implementación la elección de utilizar un operador u otro para generar una nueva mosca en la población se realiza mediante *porcentajes*. Se establece un porcentaje asignado a cada operador de manera que en cada iteración se asignarán ese porcentaje de moscas sobre el total de la población al operador correspondiente. Estos porcentajes son elegidos en un principio manualmente aunque pueden ser modificados dinámicamente por la aplicación dependiendo de ciertas características. Por ejemplo, en caso de que el operador de abducción no cuente con la lista de píxeles necesaria, cedera su porcentaje de moscas equitativamente al resto de operadores.

5.5. Cálculo de la posición 3D

Este paso del filtro de moscas consiste en calcular la posición 3D estimada del objeto seguido. Tras haber computado la salud de las moscas y generada una nueva

población, podemos calcular la posición estimada del objeto, realizando una media de las coordenadas de cada una de las moscas.

En esta implementación se ha utilizado una *media ponderada* en lugar de una media tradicional para el cálculo de la estimación. Así pesarán más en la media aquellas moscas que posean un mayor salud. Para aumentar aún más la exactitud de la posición se cuentan en la media ponderada unicamente aquellas moscas con una salud por encima de un umbral. De este modo, aquellas moscas que no dispongan de salud suficiente no interferiran en el cálculo.

Tras haber calculado la posición estimada usando la media ponderada, el esquema filtro de moscas se encarga de entregar a la aplicación de seguridad Watcher la situación del objeto, así como una representación en forma de puntos 3D de las moscas.

5.6. Visualización de resultados

La visualización de los resultados del filtro de moscas se efectua, al igual que sucede con el filtro de partículas, en la interfaz gráfica de Watcher. Esta interfaz ha sido diseñada de manera que permita visualizar la posición estimada en coordenadas, o también si se desea, la población de moscas en torno al objeto tanto en las imágenes reales como en la imágen de la cámara virtual (figura 5.4).

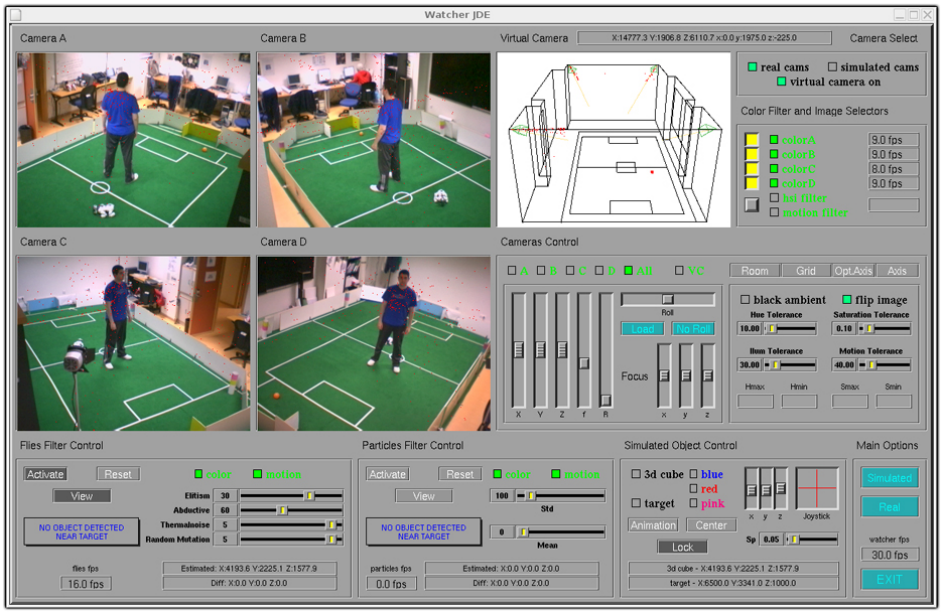


Figura 5.4: Interfaz gráfica de Watcher. Filtro de moscas.

Las características más importantes de la interfaz con respecto al filtro de moscas, las cuales permiten alterar el comportamiento del algoritmo, son las siguientes:

- Botón principal *"Activate"*. Se encarga de iniciar y detener el filtro de moscas.
- Botón principal *"Reset"*. Se encarga de reiniciar el filtro manualmente, distribuyendo uniformemente las partículas por todo el espacio.
- Botón principal *"View"*. Se encarga de activar y desactivar la visualización de la población de moscas en las imágenes. Este botón permitirá la visualización contínua de toda la población y observar su evolución.
- Botón Opcional *"Color"*. Se encarga de activar y desactivar el procesado de información de color de las imágenes. Si está desactivado, el filtro de moscas no funciona basándose en información de color.
- Botón Opcional *"Motion"*. Es análogo al anterior pero para la información de movimiento.
- Barras de desplazamiento horizontales *"Random mutation, Thermalnoise, Abductive y Elitism"*. Se encargan de modificar manualmente el porcentaje de moscas que se asignan a cada operador genético.
- Recuadro de información *"Estimated Position"*. En él se visualizará en todo momento la posición 3D expresada en coordenadas cartesianas en milímetros.
- Recuadro de información *"WARNING"*. Este recuadro aparecerá en color gris si el objetivo localizado se encuentra fuera de la zona restringida. Por el contrario, parpadeará en color rojo si se invade la zona protegida.
- Recuadro de información *"Flies fps"*. En él se muestran las iteraciones por segundo que está ejecutando el filtro de moscas en cada instante.

El resto de las opciones de la interfaz se encuentran detalladas en el apartado donde se describe la interfaz de watcher (apartado 4.2.2).

Capítulo 6
Experimentos

Una vez conocidos los fundamentos de los dos métodos estudiados e implementados, este capítulo expone los resultados obtenidos en las pruebas que se han realizado. Los experimentos son los mismos en los dos algoritmos para poder así contrastarlos.

Se ha experimentado utilizando *diferentes fuentes de información*, *diferentes velocidades del objetivo* y *un número variable de cámaras* para una población de 1000 muestras de cada algoritmo. Mediante estos experimentos se intenta analizar el comportamiento de las dos técnicas y así conocer los parámetros óptimos para su funcionamiento: manera adecuada de combinar fuentes de información, velocidad máxima a la que se puede seguir un objeto y precisión en la estimación usando un número variable de cámaras.

6.1. Escenario de pruebas

Las pruebas realizadas a los dos algoritmos han sido posibles gracias al montaje de un *escenario de pruebas* previamente instalado. Para ello se ha adaptado una habitación concreta al entorno necesario de la aplicación de vigilancia y, de esta manera, poder realizar no sólo experimentos virtuales en el simulador 3D sino también pruebas con objetos y personas reales.

El montaje físico se realizó en el Laboratorio de Robótica del Edificio Departamental II de esta universidad y consistió inicialmente en la colocación de cuatro cámaras web en los rincones superiores de la habitación. Las cuatro cámaras fueron adheridas al techo de la sala diseñando para ello unos soportes de madera específicos, cortados a medida para que encajaran adecuadamente. Las cámaras están conectadas con cables firewire a sendos ordenadores conectados en red local. La aplicación de seguridad fue ejecutada en otro equipo del laboratorio conectado a esta misma red. Las características

de este equipo son: Pentium 4 con función de Hyperthreading a 2,6 Ghz, 512 MB de RAM y sistema operativo GNU/Linux (figura 6.1 (a)).

Figura 6.1: Escenario de pruebas - Equipo principal (a) y cámara Apple iSight (b).

Para el correcto funcionamiento de la aplicación se realizó una calibración de cada una de las cámaras. Esta calibración consistió básicamente en obtener los *parámetros extrínsecos e intrínsicos* de las cámaras. Para la obtención de los parámetros intrínsecos se siguió el *modelo de cámara Pinhole*, ya detallado en el apartado 3.3.1, y la biblioteca de visión OpenCV[1]. Puesto que las cuatro cámaras son del mismo modelo (Apple iSight) (figura 6.1 (b)), sólo ha sido necesario obtener los parámetros intrínsecos de una de ellas. En el caso de estas cámaras, la matriz de calibración obtenida para una resolución de 320x240 píxeles es la siguiente:

$$\begin{pmatrix} 405{,}44 & 0{,}0 & 150{,}39 \\ 0{,}0 & 409{,}54 & 142{,}58 \\ 0{,}0 & 0{,}0 & 1{,}0 \end{pmatrix} \tag{6.1}$$

Para el *cálculo de los parámetros extrínsecos* fue necesario conocer la *posición exacta* y la *orientación* de cada cámara dentro de la habitación. Para ello fue necesario medir la situación exacta partiendo de un orígen de coordenadas común para todas las cámaras. Para la orientación de cada cámara se situó un objeto en el centro de la sala y se conseguió centrar ese objeto en todas al mismo tiempo. No obstante, gracias al simulador que proporciona Watcher, fue posible solapar las imágenes 3D con las imágenes de las cámaras reales y comprobar si la superposición coincidía (figura 6.2).

[1] http://sourceforge.net/projects/opencvlibrary

Figura 6.2: Imagen real solapada con virtual en la interfaz de Watcher.

Por último, para visualizar los resultados y ajustar los algoritmos es necesario contar con una *representación 3D de la habitación*. Esta representación se muestra en la cámara virtual junto con las partículas o las moscas en la posición 3D estimada. Por ello fue necesario medir la habitación y conocer la situación de los objetos característicos necesarios para la simulación desde el origen de coordenadas establecido.

6.2. Experimentos del filtro de partículas

En primer lugar se mostrará el funcionamiento habitual de la aplicación de seguridad *Watcher* utilizando el filtro de partículas. En ella se utilizaron las *cuatro cámaras de la habitación* y se procesaron las imágenes para obtener *información de color y movimiento*, utilizando una población de 1000 partículas. Para esta prueba se preparó el sistema para seguir un *objeto de color azul en movimiento*, en particular una camiseta (figura 6.3). La zona restringida en la habitación era el centro de la habitación, y más concretamente, la papelera situada en dicho punto.

Figura 6.3: Filtro de partículas - Alarma no activa (a) y Alarma activa (b).

En la figura (figura 6.3 (a)) se puede observar la nube de partículas superpuestas en la imágen en torno a la persona detectada. También se puede observar cómo en el momento en que el objeto localizado se aproxima a menos de 1 metro de la papelera, la alarma del sistema advierte de la presencia de alguien en la zona restringida (figura 6.3 (b)).

En esta prueba se tomaron tres medidas a diferentes distancias del orígen de coordenadas para contrastar las medidas reales y teóricas, expresadas en milímetros. Los resultados fueron los siguientes: para una posición real en P1(x,y,z) = (3200.0, 1800.0 , 1500.0) se obtuvo una posición estimada P1'(3194.7, 1804.1, 1499.2), para P2(6500.0, 1800.0, 1500.0) se obtuvo P2'(6498.1, 1794.0, 1497.5) y para P3(3500.0, 2000.0, 200.0) se obtuvo P3'(3495.4, 2003.2, 195.7). Estos resultados indican un error cometido del orden de centímetros. Se debe contemplar que siempre se introduce un cierto error de precisión al realizar una calibración manual de las cámaras. Aun con este error debido a la calibración, se confirma que *la técnica es suficientemente robusto como para detectar una única persona o un único objeto con gran precisión.* Igualmente, se observa que la precisión es la misma en para diferentes zonas de la sala.

La velocidad del sistema en la ejecución típica es de 30-32 ips, funcionando con cámaras a 12 fotogramas por segundo, sin latencia entre la obtención de cada imagen y su procesado. Esta velocidad es más que suficiente para seguir una persona dentro de la sala.

6.2.1. Ejecución con distintas fuentes de información

En este experimento se realizó una ejecución del sistema utilizando sólo *información de color azul.* Después se preparó el mismo ejercicio utilizando únicamente *información de movimiento.*

1. Utilizando sólo *información de color* para un color azul, se consigue seguir una persona con la camiseta de dicho color con éxito (figura 6.4 (a)). Sin embargo, se observa que el filtro carece de versatilidad para detectar nuevos objetos, puesto que mantiene la mirada fija en el que ya tiene localizado.

2. Usando únicamente *información de movimiento* se observa cómo es posible realizar un seguimiento de cualquier objeto o persona. Aumenta por tanto la versatilidad de detectar cualquier objeto independientemente de su color. No obstante, este seguimiento requiere que el objeto seguido se encuentre en contínuo

movimiento. Si en algún momento el objeto se detiene, la nube de partículas comienza a expandirse isótropamente en todas las direcciones para buscar nuevas fuentes en movimiento (figura 6.4 (b)).

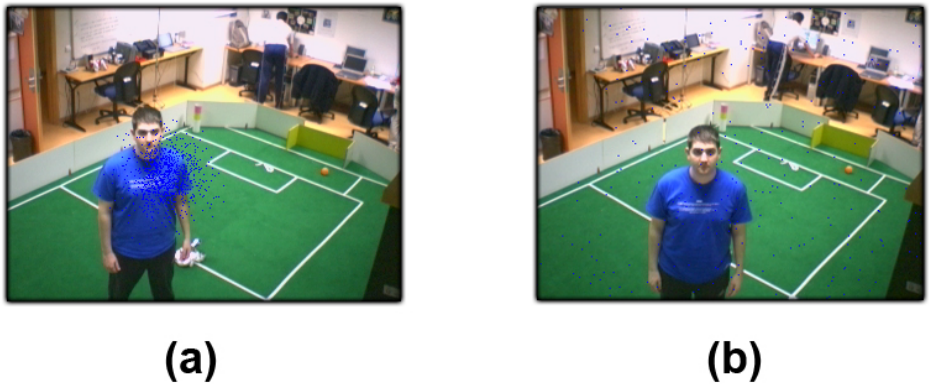


Figura 6.4: Filtro de moscas - Usando sólo color (a) y usando sólo movimiento (b).

Es necesario destacar que la utilización de una única fuente de información permite que el filtro de partículas funcione más deprisa (50-52 ips), debido a que los cálculos realizados son menos si sólo se debe procesar información de color o de movimiento.

6.2.2. Efecto del número de cámaras

En este experimento se tratará de observar el funcionamiento del filtro ante un número distinto de observaciones simultáneas. Para ello se han realizado varias ejecuciones del filtro usando color y movimiento, e incorporando progresivamente las observaciones de cada una de las cámaras.

1. Utilizando *únicamente una imagen* se observa cómo una única imagen no es suficiente para realizar una estimación de la posición. La nube de partículas se concentra en la línea de proyección de la primera cámara y comienza a desplazarse siguiendo dicha línea hacia el exterior de la cámara, debido al movimiento gaussiano aplicado.

2. Conjuntando la imagen anterior *con la imagen de la segunda cámara*, el filtro consigue estimar la posición del objeto perfectamente. La estimación realizada mantiene un error de precisión centimétrico. En este caso, la velocidad del filtro de partículas asciende a 56-60 iteraciones por segundo en conjunto. No obstante, utilizando dos cámaras de una misma zona de la sala se obtiene un efecto de *deriva sistemática* en la nube de partículas.

3. Ante la incorporación de la *imagen de la tercera cámara* observamos principalmente que la velocidad del filtro disminuye a 40-43 iteraciones por segundo. El error en la estimación sigue siendo del mismo orden que con dos cámaras. Sin embargo, se observa que debido a la incorporación de la tercera imagen, podemos realizar un seguimiento en un volumen más amplio. La imagen de la tercera cámara en conjunción con las anteriores nos proporciona más visibilidad de la sala, sacrificando para ello algo de velocidad en la estimación.

4. Al incorporar la cuarta cámara se observa cómo se tiene visibilidad en toda la sala. Las únicas zonas no cubiertas son aquellas en las que, debido a la apertura angular de las cámaras utilizadas, no es posible visualizar el objeto más que en una de las imágenes. Esta zona es justo la de debajo de cada una de las cámaras. La velocidad del filtro se mantiene constante en unas 30-32 ips, lo cual es más que suficiente para realizar el seguimiento de un objeto de forma vivaz.

Este experimiento ratifica la ituición de que el aumento del número de cámaras aumenta la percepción general del filtro al visualizar más espacio del lugar de seguimiento a costa de perder algo de velocidad. La estimación se consigue utilizando al menos dos cámaras y la precisión es centimétrica independientemente del número de imágenes procesadas en la misma iteración.

6.2.3. Velocidad de seguimiento

Mediante este experimento se trató de caracterizar la velocidad máxima de un objeto que es capaz de seguir el filtro de partículas. Dicho experimento se realizó utilizando la funcionalidad del simulador 3D de Watcher de poder incluir *objetos 3D en las imágenes reales*. Utilizando un cubo 3D de color azul como objeto a seguir y activar el movimiento contínuo de este, se realizó la localización 3D a distintas velocidades (figura 6.5).

Tras realizar algunas pruebas, se pudo observar que utilizando las cuatro cámaras y la información de color y movimiento, es posible seguir el objeto 3D hastala velocidad que tendría una persona andando. Si se desactiva alguna fuente de información o se reduce el número de observaciones simultáneas desactivando alguna de las cámaras, la velocidad de seguimiento permite seguir a una persona todavía de manera más vivaz.

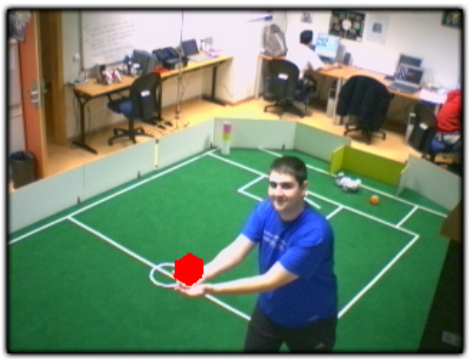


Figura 6.5: Objeto 3D sobre imagen real.

6.3. Experimentos del filtro de moscas

Tal y como ya se realizó para el filtro de moscas, en este primer experimento con el filtro de moscas se mostrarán los resultados para una ejecución típica del filtro. Se realizará un seguimiento 3D a una *persona vestida de color azul* desplazándose por la habitación. Para ello se utilizarán las *cuatro cámaras* del laboratorio y se filtrarán las imágenes por color y movimiento. Al igual que con el filtro de partículas, la zona central donde hay colocada una papelera será la zona restringida (figura 6.6 (a)).

Figura 6.6: Filtro de moscas - Alarma no activa (a) y Alarma activa (b).

Para el cálculo de la posición estimada se ha utilizado una *media ponderada* y un umbral de 0.7, elegido de forma experimental. Al igual que ocurría con el filtro de partículas, el sistema es capaz de advertir de la presencia de un objeto o una persona cerca de la papelera (figura 6.6 (b)). La distancia mínima a la que se debe de encontrar un objeto detectado es de 1 metro. Para ello se volvió a tomar las mismas tres medidas que en el filtro de partículas para contrastar la situación real con la estimada por la

aplicación. Los resultados en milímetros fueron: para una posición real en P1(x,y,z) = (3200.0, 1800.0 , 1500.0) se obtuvo una posición estimada P1'(3196.2, 1805.2, 1498.1), para P2(6500.0, 1800.0, 1500.0) se obtuvo P2'(6497.4, 1799.2, 1504.3) y para P3(3500.0, 2000.0, 200.0) se obtuvo P3'(3506.7, 2002.3, 198.1).

Estos resultados indican una precisión del orden de centímetros. De nuevo, se debe contemplar que siempre se introduce un cierto error de precisión al realizar una calibración manual de las cámaras. No obstante, se confirma que *el algoritmo es suficientemente robusto como para detectar una única persona o un único objeto con gran precisión.* Como ya sucedía con el filtro de partículas, la precisión es la misma en diferentes zonas de la sala.

La velocidad del sistema en esta ejecucuón es de 20-24 ips, funcionando con cámaras a 12 fotogramas por segundo, de nuevo sin latencia entre la obtención de cada imagen y su procesado. Esta velocidad es suficiente para seguir una persona dentro de la sala.

6.3.1. Ejecución con distintas fuentes de información

Al igual que ya se explicó con el filtro de partículas, se comentan a continuación los resultados del filtro de moscas utilizando distintas fuentes de información:

1. Usando sólo*información de color* para un color azul, se realiza perfectamente el seguimiento de una persona (figura 6.7(a)). Sin embargo, usando esta única fuente de información, este filtro resulta de nuevo poco dinámico en el seguimiento de otro objeto con otro color que pueda surgir en las imágenes.

2. Usando el filtro con sólo *información de movimiento* se observa desde el inicio cómo se puede realizar el seguimiento a cualquier objeto sin tener en cuenta su color. No obstante, y al igual que sucede con el filtro de partículas, es requisito el hecho de que el objeto localizado debe encontrarse en contínuo movimiento (figura 6.7 (b)).

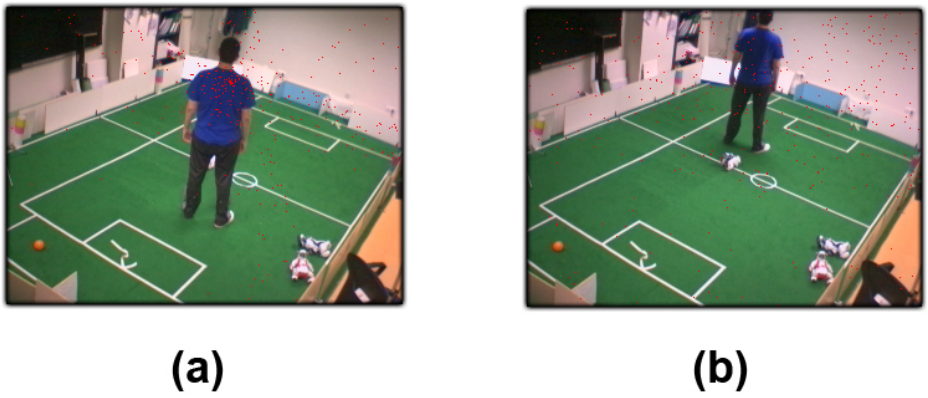


Figura 6.7: Filtro de moscas - Usando sólo color (a) y usando sólo movimiento (b).

Es necesario destacar al igual que ya se hizo en el filtro de partículas, que el uso de una única fuente de información permite que este filtro funcione más deprisa (30-34 ips).

6.3.2. Efecto del número de cámaras

Tal y como se realizó en el filtro de partículas, se observó el comportamiento del filtro de moscas ante un número distinto de imágenes de cámara, e incorporando las imágenes progresivamente. Los resultados observadas fueron los siguientes:

1. Como ya sucedía en el filtro de partículas, una *única imagen* se muestra insufienciete para estimar la posición 3D de un objeto. La población de moscas se concentra en gran parte en la línea de proyección que establece el operador abductivo.

2. Añadiendo *la imagen de la segunda cámara*, el filtro consigue estimar la posición del objeto sin ningún problema. La estimación posee un error de precisión centimétrica y responde muy vivazmente ante los cambios de posición del objeto. La velocidad del filtro de moscas se mantiene en 40 iteraciones por segundo y *no existe efecto de deriva sistemática*.

3. Ante la incorporación de la *imagen de la tercera cámara* y posteriormente de la *imagen de la cuarta cámara*, la velocidad del filtro disminuye hasta las 20-22 ips. El error cometido en la estimación permanece prácticamente igual, pero debido a la incorporación de más imágenes se cubre más volumen de la habitación y se responde aún más deprisa a un cambio de posición, debido a que el operador abductivo actua ahora en más imágenes.

Este experimiento indica, como ya sucedía en el filtro de partículas, que el aumento del número de cámaras aumenta la percepción general del filtro al visualizar más espacio del lugar de seguimiento a costa de perder algo de velocidad. La estimación se consigue utilizando al menos dos cámaras y la precisión es centimétrica independientemente del número de imágenes procesadas en la misma iteración.

6.3.3. Velocidad de seguimiento

Este experimento es similar al realizado previamente en el apartado 6.2.3. Los resultados obtenidos indican nuevamente que la velocidad del algoritmo es suficiente para seguir una persona andando. No obstante, se aprecia una mejor respuesta ante cambios en la posición del objeto debido al operador de abducción a pesar de la velocidad menor del algoritmo con respecto al filtro de partículas, como se comenta en la sección de contraste.

6.4. Análisis y comparación de algoritmos

Una vez analizados los mismos experimentos con los dos filtros, introduciremos en este apartado el contraste efectuado de los dos algoritmos de localización 3D. Ambas técnicas se han mostrado *satisfactorias* para la aplicación de seguridad. No obstante, se han detectado las siguientes particularidades:

1. En cuanto a la *utilización de distintas fuentes de información*, podemos observar como el filtro de moscas aprovecha mejor la información de color y movimiento que el filtro de partículas. Las dos técnicas precisan que el tamaño del objeto sea superior a un recuadro de 16 píxeles para ser detectado. En los casos en los que se haya localizado un objeto, ambos filtros combinan perfectamente las dos fuentes y consiguen una estimación parecida. Sin embargo, en el momento en que pierden el objeto por alguna circunstancia y acto seguido aparece un nuevo candidato en otra zona de la habitación, el filtro de moscas lo detecta prácticamente de inmediato. El filtro de partículas por el contrario tarda algunos segundos hasta que alguna partícula detecta al nuevo objeto. La ventaja del filtro de moscas se debe fundamentalmente al *operador de abducción*. Mientras que las partículas siguen buscando un objeto en las zonas cercanas a la posición de ellas mismas, el operador de abducción proporciona al filtro de moscas una detección inmediata situando las moscas en posiciones compatibles con las nuevas imágenes.

2. Ambos filtros responden bien ante *objetos con gran vivacidad*. Se ha conseguido

un rendimiento medio de 40 ips para el filtro de partículas y 25 fps para el filtro de moscas. Estas velocidades son más que suficientes para procesar las imágenes de las cámaras, que se reciben a 12 fps. Por tanto, se realizan 2 o 4 iteraciones por cada imagen recibida, lo que permite que no haya grandes desplazamientos de una imagen a otra. Ambas mediciones se han efectuado para una población de 1000 muestras y un procesado completo en las cuatro cámaras. El análisis realizado nos indica que esta diferencia de velocidad se debe principalmente a dos factores:

a) La generación de una nueva población de partículas se realiza mediante el *algoritmo de la ruleta*. Este algoritmo tiene una complejidad $n * log_n$. En el caso de esta implementación se realizan 1000 búsquedas binarias. Por el contrario, la generación de cada población de moscas es un proceso costoso. El orden de complejidad en este caso asciende a n^2. Además, el filtro de moscas tiene que ordenar de mayor a menor salud la población de 1000 moscas al finalizar cada iteración. Aunque el algoritmo utilizado es quicksort, la complejidad asciende a $n * log_n$ (en el peor de los casos n^2). Por el contrario, el filtro de partículas no necesita efectuar ninguna ordenación de las partículas.

b) El filtro de moscas necesita un procesado completo de color y movimiento para cada una de las imágenes, de forma que el operador de abducción pueda aprovechar toda la información subyacente de todas ellas. El procesado de color y movimiento utilizando ventana de atención unicamente, tal y como realiza el filtro de partículas, consiste en esta implementación de 25000 píxeles filtrados para cada imagen (1000 moscas x 25 píxeles de la ventana de vecindad). Si se realiza un filtrado completo el número de píxeles filtrados asciende a 76800 (320x240 píxeles por imagen), por lo que este cálculo contribuye negativamente a la velocidad del filtro pero positivamente a la eficiencia en la localización. En el caso de aumentar el tamaño de la imagen a una resoluciòn mayor, como 640x480, en el filtro de partículas se obtendría el mismo rendimiento mientras que el filtro de moscas funcionaría más despacio debido a que debe filtrar por completo una imagen mayor.

A pesar de la diferencia de velocidad, en nuestra aplicación 20 ips son suficientes debido a las imágenes de las cámaras llegan a la aplicación a 10-12 fps. Esta velocidad se debe a que las imágenes son capturadas por el servidor de imágenes Oculo y enviadas por red local. En el caso de conectar las cámaras localmente la

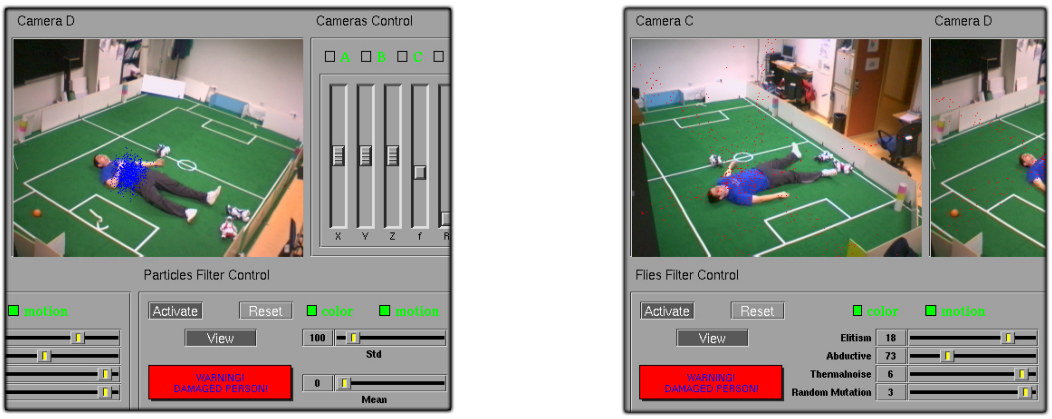
velocidad de 20 fps podría llegar a ser insuficiente.

3. Respecto al *número de cámaras* los dos filtros funcionan eficientemente con un número variable. Para dos cámaras el filtro de partículas produce un rendimiento de 60-70 ips y comete un error de 5-6 cm. El filtro de moscas supera las 35 ips y el error cometido es también de 5-6 cm. Si utilizamos cuatro cámaras, el rendimiento es inferior y el error se mantiene en el mismo orden, sin embargo, es más difícil no ser captado por la aplicación debido a que se cubre mayor superficie de la habitación.

El *seguimiento multiobjeto* no es soportado por ninguno de los dos algoritmos, no obstante los efectos observados en ambos son diferentes. A pesar de se encuentra fuera de las ambiciones de este proyecto, se ha realizado un análisis con varias fuentes compatibles cuyos resultados están descritos en el anexo de esta memoria.

6.5. Aplicación con umbral en coordenada Z

Una vez finalizados los experimentos para analizar el comportamiento general de las dos técnicas empleadas, se deseó añadir una nueva funcionalidad a la aplicación de seguridad. Esta nueva característica permite detectar cuando la persona u objeto localizado se encuentra tumbado o apoyado en el suelo (figura 6.8), debido a que ha podido caerse o desmayarse.



(a) (b)

Figura 6.8: Detección de persona tumbada suelo - Con partículas (a) y con moscas (b).

Para conseguir este propósito se detecta si la coordenada Z de la persona localizada se encuentra por encima de un determinado *umbral previamente establecido*, en nuestro

caso una distancia de 20 cm elegida experimentalmente. Para lograr esta nueva posibilidad se utiliza cualquiera de los dos *algoritmos de localización 3D* descritos en esta memoria. *Gracias a la precisión alcanzada* en las dos técnicas se puede distinguir claramente si la persona localizada se encuentra tumbada o no en el suelo. Los experimentos realizados con esta nueva característica han demostrado que cualquiera de las dos técnicas son perfectamente capaces de estimar si una persona está tumbada. En esta aplicación *no hay variación en el rendimiento* con respecto a una ejecución típica de los algoritmos.

Capítulo 7
Conclusiones y trabajos futuros

En los tres primeros capítulos se presenta una descripción del contexto sobre el que se apoya este proyecto fin de carrera, se detalla el entorno de desarrollo sobre el cual se trabaja y se presenta el problema y los objetivos de este proyecto. Los siguientes capítulos los procedimientos utilizados para resolver dicho problema propuesto: la localización 3D usando el filtro de partículas y la localización 3D usando el filtro de moscas. Después se presentan algunos experimentos realizados con las dos técnicas y los resultados obtenidos en cada una de ellas.

En este último capítulo se describirán las conclusiones obtenidas de estos experimentos y las posibles vías de continuidad para la investigación.

7.1. Conclusiones

El objetivo principal de este proyecto era *tratar de solucionar el problema de la localización 3D* de un objeto utilizando técnicas de visión. Para ello se pensaba diseñar *una aplicación de seguridad* que localizara en 3D a una persona o un objeto dentro de una habitación de gran volumen utilizando *dos algoritmos distintos*. El primero de ellos es un *filtro de partículas*, algorimo probabilístico basado en técnicas de Monte Carlo. El segundo es un algoritmo evolutivo denominado filtro de moscas. Con tal propósito se colocaron en esta habitación cuatro cámaras en cada uno de los rincones superiores. El sistema debía avisar mediante señales visuales o auditivas si la persona seguida se adentraba en una zona de la sala declarada como restringida. Para probar y ajustar los algoritmos se *realizarían diferentes experimentos* que permitirían contrastar las técnicas utilizadas.

Los requisitos propuestos para el sistema consistían en: que fuera *vivaz*, que fuera capaz de seguir a una única persona u objeto con *gran precisión*, que utilizara únicamente *hardware no específico* para su funcionamiento, que fuera capaz de aplicarse

a una *habitación de gran volumen* y que se implementara en *lenguaje C* sobre la plataforma software jde.c en forma de *esquemas*.

En primer lugar es necesario destacar que *todos* los objetivos y requisitos se han cumplido con éxito tal y como se analiza el siguiente análisis.

Para poder probar adecuadamente los algoritmos se instalado un *escenario de pruebas* que ha permitido colgar las cuatro cámaras disponibles del techo del laboratorio. Este escenario de pruebas también consistió en la calibración de las cuatro cámaras Apple iSight con las que se contaba en el sistema. La calibración realizada se detalla claramente en el apartado 6.1.

En cuanto a las técnicas de localización, se implementó en primer lugar un *filtro de partículas*, algoritmo probabilístico basado en técnicas de Monte Carlo. La implementación se recoge en el *esquema filtroparticulas* que se describe en el apartado 4.2.3. Este algoritmo utiliza la información de color y movimiento que extrae de las imágenes recibidas de la plataforma jde.c. Através de esta información, la técnica es capaz de estimar la posición 3D de un objeto de un color previamente configurado, la posición 3D de un objeto o persona en movimiento o de un objeto que reuna estas dos características. Tras calcular la estimación de la situación del objeto, el filtro de partículas entrega la estimación tridimensional calculada junto para que la aplicación general pueda visualizar el resultado en la interfaz gráfica.

El *rendimiento* obtenido para el filtro de partículas *más que satisfactorio*. Los experimientos realizados en la sección 6.2. Utilizando las cuatro cámaras de la habitación simultáneamente y un filtrado de color y movimiento en las imágenes se consigue una velocidad de entre 40-42 iteraciones por segundo. Esta velocidad supone una gran vivacidad en el sistema, más que suficiente para seguir a una persona en movimiento dentro de la sala. Además, *no existe latencia* entra las imágenes recibidas y las analizadas, debido a que es el propio filtro de partículas el que en cada iteración se encarga de obtener las imágenes que necesita. En cuanto a la precisión, se han realizado varias pruebas que han confirmado que el sistema tiene una precisión centimétrica. Por ejemplo, para una medición real en milímetros de P(x,y,z) = (3200.0, 1800.0 , 1500.0) el sistema estimó que el objeto detectado se encontraba en P'(3194.7, 1804.1, 1499.2). Además, se realizaron otras pruebas tales como un análisis multiobjeto (que se detalla en el anexo de esta memoria) y se confirmó la *gran capacidad de recuperación*

del sistema, pues ante un objeto perdido se observó cómo la nube de partículas se desplazaba isótropamente por todo el espacio en busca de una nueva fuente de color o movimiento compatible.

La segunda técnica de localización utilizada es el denominado *filtro de moscas*. Dicha técnica se implementa en el *esquema filtromoscas* de la aplicación de seguridad, y de nuevo utiliza la información de color y movimiento para estimar la posición de un objeto. Este es un algoritmo evolutivo basado en muestras de gran capacidad de búsqueda de una solución basado en la utilización de *operadores genéticos*. Estos operadores aprovecharán las mejores propiedades de cada elemento de la población de moscas para crear nuevas generaciones más robustas y cada vez más próximas a estimar la posición 3D de un objeto o una persona dentro de la sala. Para este algoritmo se ha realizado un diseño nuevo del mismo, debido a que es la primera vez que se utiliza para resover el problema de la localización 3D. La principal novedad con respecto a los anteriores diseños es la creación de un operador completamente nuevo que, sin duda, será determinante en la exploración del espacio tridimensional. Se trata del *operador genético de abducción*, que se analiza más en profundidad en la sección 5.4.

Los *experimentos* realizados con el filtro de moscas vuelven a ser *satisfactorios en cuanto al rendimiento*. Utilizando las cuatro imágenes de las cámaras y realizando filtrado por color y movimiento se ha obtenido un rendimiento de 29-30 ips. Con esta velocidad es también más que suficiente para estimar la posición de un objeto o persona en movimiento dentro de la habitación. De nuevo *no existe* latencia entre las imágenes recibidas y las procesadas puesto que es el propio filtro el que se encarga de recabar esta información cuando la necesita. En cuanto a la precisión, se han realizado algunas medidas dentro de la sala (apartado 6.3) y de todas ellas se ha obtenido una error medio en la precisión de 4-5 cm. Al igual que con el filtro de partículas, se realizaron otros experimentos adicionales, entre los que se encuentra el *análisis multiobjeto* que se puede encontrar en el anexo de esta memoria.

De forma resumida, podemos notar que la principal diferencia entre las dos técnicas es la notable diferencia de velocidad entre las dos. Tal y como se explica en el constraste de los algoritmos realizado en el punto 6.4, la diferencia radica en que el filtro de moscas debe realizar más cálculos por iteración que el filtro de partículas. Sin embargo, entre estos cálculos se encuentra el filtrado completo de las imágenes, que proporciona al operador de abducción una capacidad de exploración más grande, que permite al filtro

de moscas anticiparse con mayor velocidad a movimientos bruscos y *mayor capacidad de reacción ante posibles sorpresas*.

Ambas técnicas han sido probadas en la *aplicación de seguridad* diseñada. En ella se han probado ambos algoritmos aplicados a la detección de intrusos en la zona restringida y el resultado ha sido *muy favorable*. El sistema consigue alertar perfectamente y sin ningún retraso de tiempo real cuando una persona se aproxima andando a la zona protegida. Las dos técnicas han resultado ser completamente satisfactorias para funcionar dentro de la aplicación. Adicionalmente, y para probar una nueva posibilidad del sistema, se añadió la capacidad de detectar personas cuando estas están tumbadas en el suelo. Esta aplicación, que puede servir para detectar cuando alguien se ha caido dentro de la habitación, ha sido probada con las dos técnicas obteniendo el mismo resultado satisfactorio.

Por último cabe señalar la funcionalidad de la aplicación en cuanto a que *es capaz de simular imágenes tridimensionales* si se desea probar estos efectos con un simulador. También es capaz de generar objetos tridimensionales para la simulación de las técnicas, bien sea sobre imágenes también virtuales o sobre las imágenes de verdad. Para ello la aplicación cuenta con una *interfaz gráfica* que permite visualizar los resultados de ambas técnicas de localización gracias al uso de una cámara virtual que permite visualizar la nube de partículas o las poblaciones de moscas en cada iteración. Los controles incoporados, las señales visuales así como el sistema de eventos de ratón del sistema, proporcionan la herramiente necesaria para visualizar, depurar y ajustar cualquier resultado.

7.2. Trabajos futuros

La principal línea futura de investigación para la continuidad de este proyecto es la posibilidad de realizar *localización 3D de múltiples objetos*. En este proyecto se ha resuelto con éxito el seguimiento tridimensional de un único objeto en una sala de gran volumen, por lo que todo parece indicar que contando con un mayor número de cámaras, es posible cubrir con éxito cualquier espacio que se proponga. Los dos algoritmos empleados son escalables y permiten aumentar el número de fuentes de información y el número de muestras utilizadas para estimar la posición. Sin embargo, ambas técnicas están diseñadas para el seguimiento de un único objeto y no contemplan la lógica posibilidad de que exista más de un elemento en escena.

Por otra parte, como mejora de la aplicación de seguridad, es necesaria la incorporación de una *autodefinición de color*, de forma que en el momento en que aparezca en movimiento un nuevo objeto, la aplicación determine el color característico por el que se pueda identificale.

También sería interesante introducir en la aplicación la posiblidad de mover las cámaras para cubrir mejor la zona vigilada mediante el *acoplo de cuellos mecánicos* a cada una de las cámaras.

Anexo.

Experimentos con multiples objetos.

Adicionalmente a los experimentos ya descritos con los algoritmos de localización 3D, se realizaron otros experimentos para estudiar el comportamiento del filtro de partículas y el filtro de moscas ante la existencia de *varios objetos compatibles* en las imágenes. Aunque no es objetivo de este proyecto solucionar un seguimiento multiobjeto, se ha decidido no obstante observar el funcionamiento de las dos técnicas para esta posibilidad.

Para este experimento se han realizado tres pruebas, en las que se han utilizado las cuatro cámaras y distintas fuentes de información:

1. En la *primera prueba* se utilizó el filtro de partículas y el filtro de moscas utilizando sólo información de color, siguiendo a una persona con camiseta azul.

   - En el experimento con el *filtro de partículas*, se provocó un cruce de la persona con otro objeto del mismo color en alguna de las imágenes. Esta situación da lugar a que el filtro siga únicamente a una de las personas al mismo tiempo. Es posible lograr que la segunda persona se lleve consigo la nube de partículas, debido a que el otro objeto es igualmente compatible utilizando únicamente el color.
   - En el experimento con el *filtro de moscas*, en el momento en que aparece el segundo objeto de la misma tonalidad, a diferencia del filtro de partículas, se detecta inmediatamente la nueva fuente de color compatible pero no se pierde de vista la anterior. No obstante, cree estar siguiendo el mismo objeto que ya tenía localizado porque en nuestra aplicación se le fuerza a dar una estimación única. La posición media estimada queda distorsionada debido a este hecho.

   Este efecto es posible solucionarlo en el caso de que se desea realizar un seguimiento de un objeto en movimiento. Para ello se puede *incorporar fuente de información por movimiento* al cómputo del filtro. De esta forma, es más compatible un objeto de color azul en movimiento que un objeto de color azul estático.

2. En la *segunda prueba* se utilizó el filtro de partículas y el filtro de moscas utilizando información de color y movimiento. Este experimento muestra el comportamiento de los algoritmos cuando se intenta seguir varios objetos en movimiento de un mismo color. En este caso se experimentó qué ocurre cuando el objeto azul que se cruza con el localizado se encuentra también en movimiento.

   - En el experimento con el *filtro de partículas*, se observa que el filtro vuelve a seguir únicamente a una de las personas que se encuentran en escenas, pues en este caso ambos objetos están en igualdad de condiciones probabilísticas.
   - En el experimento con el *filtro de moscas*, ocurre una situación semejante al funcionamiento sólo con color. El operador abductivo consigue detectar distintos cambios de color y de movimiento de objetos simultáneos pero entiende que se trata del mismo. Igualmente la estimación de la posición se distorsiona debido a que la media ponderada añade los valores de la posición de los distintos objetos.

El funcionamiento de los algoritmos en el caso de un seguimiento multiobjeto ha sido incluído como primera línea de investigación futuro dentro del capítulo 7.

Bibliografía

[Barrera *et al.*, 2005] Pablo Barrera, José María Cañas, y Vicente Matellán. Visual object tracking in 3d with color based particle filter. *Int. Journal of Information Technology*, 2005.

[Barrera y Cañas, 2004] Pablo Barrera y José María Cañas. Seguimiento tridimensional usando dos cámaras. 2004.

[Bishop y Welch, 2005] Gary Bishop y Greg Welch. An introduction to the Kalman filter. page 16. 2005.

[de la Casa Puebla, 2005] Marta Martínez de la Casa Puebla. Sistema de atención visual en escena. *Proyecto Fin de Carrera, URJC*, 2005.

[Fox *et al.*, 1999] Dieter Fox, Wolfram Burgard, Frank Dellaert, y Sebastian Thrun. Monte Carlo localization: efficient position estimation for mobile robots. In *Proceedings of the 16th AAAI National Conference on Artificial Intelligence*, pages 343–349, Orlando (Florida, USA), July 1999.

[Hartley y Zisserman, 2004] Richard Hartley y Andrew Zisserman. *Multiple View Geometry in Computer Vision.* 2004.

[Louchet *et al.*, 2002] Jean Louchet, Maud Guyon, Marie-Jeanne Lesot, y Amine Boumaza. Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern recognition letters*, 2002.

[Louchet, 2000] Jean Louchet. Stereo analysis using individual evolution strategy. 2000.

[Louchet, 2001] Jean Louchet. Using an individual evolution strategy for stereovision. *Genetic Programming and Evolvable Machines*, 2001.

[Mackay, 1999] 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.

[Maybeck, 1979] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[Plaza, 2003] José María Cañas Plaza. *Jerarquía Dinámica de Esquemas para la generación de comportamiento autónomo*. PhD thesis, Universidad Politécnica de Madrid, 2003.

[Plaza, 2004] José María Cañas Plaza. Manual de programación de robots con jde. *URJC*, 2004.

[Thrun *et al.*, 1999a] Sebastian Thrun, Wolfram Burgard, y Dieter Fox. Markov localization for mobile robots in dynamics enviroments. *Journal of Artificial Intelligence Research*, 1999.

[Thrun *et al.*, 1999b] Sebastian Thrun, Wolfram Burgard, y Dieter Fox. Markov localization for reliable robot navigation and people detection. 1999.

[Thrun, 2000] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 2000.