UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

INGENIERO DE TELECOMUNICACIÓN

Trabajo Fin de Carrera

Sistema de detección de obstáculos y robots mediante múltiples cámaras en espacios inteligentes

Víctor Espejo Gómez

Diciembre de 2008

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

INGENIERO DE TELECOMUNICACIÓN

Trabajo Fin de Carrera

Sistema de detección de obstáculos y robots mediante múltiples cámaras en espacios inteligentes

Autor: Víctor Espejo Gómez
Director: D. Daniel Pizarro Pérez

TRIBUNAL:

Presidente: D. Ignacio Fernández Lorenzo

Vocal 1º: Dª. Marta Marrón Romera

Vocal 2º: D. Daniel Pizarro Pérez

CALIFICACIÓN: .....

FECHA: .....

ÍNDICE

I. RESUMEN

Este proyecto aborda la problemática de la detección de objetos en espacios inteligentes independientemente de la naturaleza de los mismos. Para ello, emplea la información del entorno proporcionada desde un array calibrado de cámaras que lo cubre.

La detección se realiza bajo condiciones reales y se basa en la segmentación de fondo invariante a la iluminación, y en el proyectado de la misma sobre un plano de referencia.

Así mismo se resuelve la navegación fiable de un agente robótico por el espacio inteligente al que a su a vez se hace el seguimiento basado en la odometría proporcionada

Palabras clave: Espacio inteligente, detección de objetos, visión por computador, segmentación de fondo, espacio invariante a la iluminación.

CAPÍTULO 1. INTRODUCCIÓN

1.1. Objetivos y definición del problema

El presente proyecto tiene como objetivo principal el desarrollo de un sistema de detección de objetos en un espacio tridimensional acotado empleando para ello técnicas de visión artificial. Los objetos que se pretenden detectar pueden ser de geometría rígida, como es el caso de robots móviles y obstáculos, o de geometría deformable en el caso de humanos. Dicha detección debe poder realizarse ante condiciones de iluminación reales, lo que requiere el uso de técnicas invariantes a sombras y cambios de iluminación.

Simultáneamente, se pretende realizar un control y navegación fiable de los agentes robóticos por el entorno. Éstos pueden dedicarse a tareas que el propio espacio inteligente ordene, siendo necesario asegurar la no colisión con objetos vecinos.

Una serie de objetivos secundarios son imprescindibles, éstos son:

- Interfaz gráfico para control y configuración del módulo de visión en el espacio inteligente.
- Segmentación de fondo en tiempo real.
- Compresión, envío y recepción de video en tiempo real.
- Almacenamiento sincronizado de información relevante.
- Interfaz gráfico para el control y configuración de los agentes robóticos dentro del espacio inteligente.
- Generación y visualización del mapa de ocupación del espacio inteligente.
- Sistema de representación del robot a partir de lecturas de odometría.

El diseño debe estar preparado para su utilización en espacios inteligentes, figura 1.1.

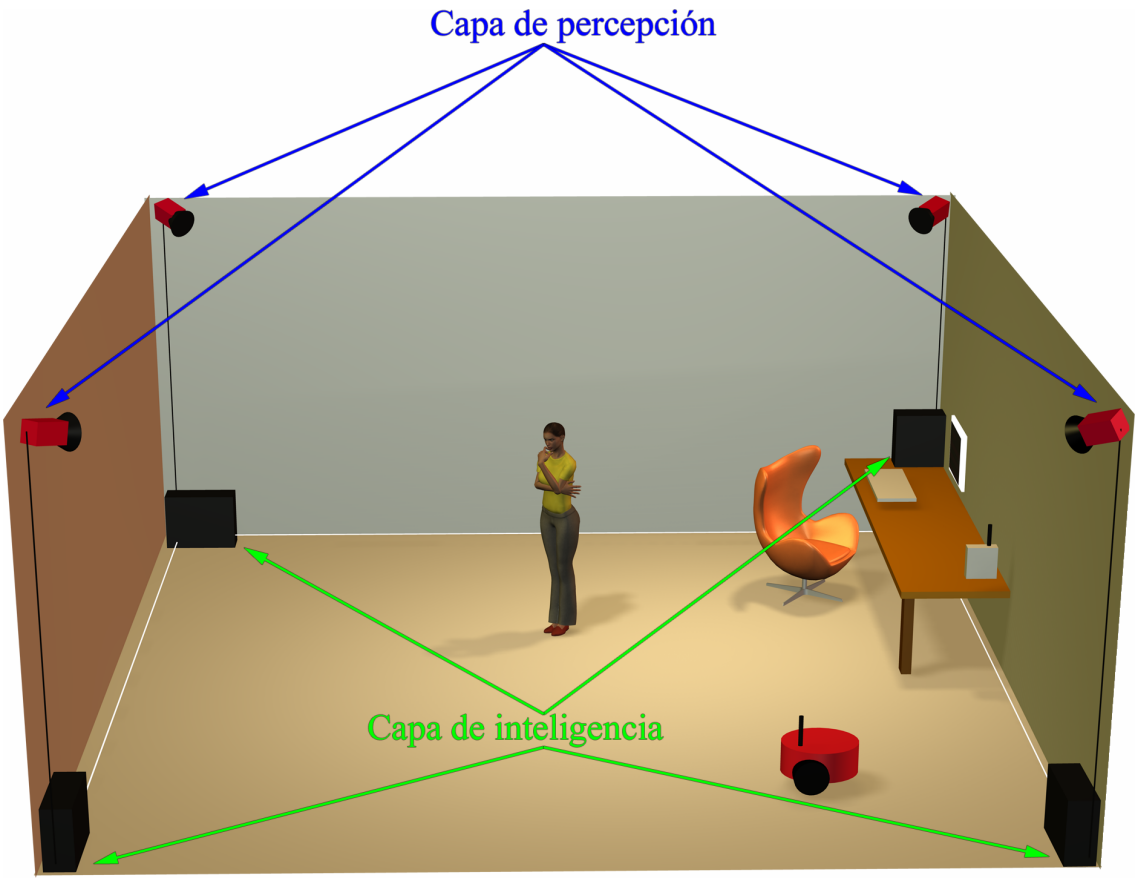


Figura 1.1 Espacio Inteligente

El sistema debe tomar datos desde la capa de percepción y enviarlos a través de la red de comunicación para la detección de objetos en la capa de inteligencia, la cual, a su vez controla el robot.

Aunque son varios los problemas a los que se enfrenta el proyecto, se va a centrar el estudio en solventar los tres de mayor importancia:

- La detección de objetos de geometría deformable imposibilita la utilización de algoritmos clásicos de reconstrucción mediante múltiples cámaras, muy extendidos en aplicaciones similares [Pizarro et al, 2005]. En su lugar se propone el desarrollo de un mapa de ocupación que realiza una partición regular de un plano de referencia, plano del suelo, a partir del cual cada objeto independiente es definido por un cúmulo de ocupación.

  Mediante el uso de múltiples cámaras es posible obtener, en cada instante de tiempo capturado, una instantánea de la ocupación del plano del suelo, por lo que la no rigidez de los objetos no influye en la detección.

- Los algoritmos que se desarrollan en el proyecto están sujetos a fuertes restricciones temporales, siendo necesario operar a una frecuencia similar a la utilizada por las cámaras para capturar. Se plantea realizar el proceso mediante computación

distribuida en los diferentes nodos, permitiendo de este modo alcanzar las restricciones de tiempo impuestas.

- La aplicación en entornos reales expone al conjunto de sensores a condiciones de sombras y cambios de iluminación. Dichos fenómenos repercuten directamente en una mala detección de objetos, pudiendo validar sombras proyectadas sobre el plano de referencia.

1.2. Punto de partida

La realización del proyecto parte de una arquitectura cliente servidor implementada secuencialmente para la configuración remota de cámaras en un espacio inteligente. Se asegura el intercambio sincronizado de datos y la correcta recepción de los mismos.

La arquitectura para la que se diseña consta de cuatro servidores y un único cliente, no siendo posible ni añadir ni quitar ninguno de sus componentes.

No queda resuelta la compresión de imágenes ni el envío de éstas, sí permitiendo el almacenamiento en bruto en los lugares de captura.

Simultáneamente, y nuevamente de forma secuencial, se conecta con un robot del que se recibe odometría y al que se mandan órdenes de movimiento mediante teclado.

1.3. Estructura del documento

El proyecto presentado está compuesto de un total de nueve capítulos. A grandes rasgos, los primeros están dedicados a la base teórica y los últimos a la implementación del sistema real. A continuación se detalla la dedicación de cada uno de ellos.

- **Capítulo 1**. *Introducción*. Es el actual capítulo el cual debe servir de base para el entendimiento del sistema que se presenta a posteriori.

- **Capítulo 2**. *Estado del arte*. Se recorren los estudios existentes de espacios inteligentes y la detección de objetos basada en grid de ocupación.

- **Capítulo 3**. *Geometría de la formación de la imagen*. Explicación teórica sobre la obtención de imágenes en las cámaras, en concreto el modelo pinhole. Se abordan aspectos como el cambio del sistema de coordenadas, la calibración, la obtención de la matriz de homografía y la generación del grid de ocupación. Todos los apartados explicados desde un punto de vista teórico sin aplicación práctica.

- **Capítulo 4**. *Espacios de color*. Se abordan algunas de las distintas formas para la representación del color en espacios. Se explican todos aquellos que son usados con posterioridad. Éstos son: RGB, YUV, HSI, HSV y el espacio invariante a la iluminación y su calibración.

- **Capítulo 5**. *Segmentación de fondo*. Concepto de segmentación de fondo, tipos, problemática y métodos propuestos.

- **Capítulo 6**. *Detección de objetos*. Se presenta la detección de objetos de forma probabilística. En concreto se hace referencia al filtro de partículas y al clasificador usados en estudios anteriores.

- **Capítulo 7**. *Implementación del sistema en tiempo real*. Explicación del sistema implementado para satisfacer los objetivos planteados. Se comienza con las conexiones necesarias en la arquitectura cliente servidor. A continuación se explica en detalle el cliente y el interfaz gráfico que lo gestiona. Finalmente el servidor.

- **Capítulo 8**. *Pruebas y resultados*. Se realizan una serie de pruebas y se muestran los resultados.

- **Capítulo 9**. *Conclusiones.* Se concluye el proyecto exponiendo las conclusiones tras su realización.

CAPÍTULO 2. ESTADO DEL ARTE

2.1. Espacios Inteligentes

El concepto de espacio dotado de inteligencia ha sido propuesto por diversos autores y ha ido evolucionando separándose en distintas implementaciones, incluyendo aquellas aplicaciones destinadas a la localización y navegación de robots.

El enfoque de "Espacio Inteligente" proviene de áreas de investigación relacionadas con el interfaz hombre-maquina (HMI). El objetivo es definir espacios donde exista una interacción no intrusiva entre usuarios y el propio entorno. La interacción tiene el objetivo de ayudar al usuario en algún tipo de tarea. En el entorno, el usuario humano recibe servicios del mismo mediante una comunicación que mantiene con él, empleando para ello un lenguaje natural entendido por ambos.

El inicio de los espacios inteligentes y la "Computación Ubicua" fueron conducidos por Weiser [Weiser, 1999] [Weiser, 1993b]. Este autor realiza una clasificación en función de cuatro elementos básicos: ubicuidad, conocimiento, inteligencia e interacción natural.

- Ubicuidad. Múltiples sistemas embebidos con total interconexión entre ellos.
- Conocimiento. Habilidad del sistema para localizar y reconocer lo que acontece en el entorno y su comportamiento.
- Inteligencia. Capacidad de adaptarse al mundo que percibe.
- Interacción Natural. Capacidad de comunicación entre el entorno y los usuarios.

Uno de los primeros sistemas ubicuos implementados con esta filosofía se propuso en el proyecto Xerox PARC de Computación Ubicua (UbiComp) [Weiser, 1993a], desarrollado a finales de los 80. La red de sensores desplegada no estaba basada en información visual debido

al coste prohibitivo para la época. Hoy en día varios grupos de investigación desarrollan y expanden el concepto de espacio inteligente y la computación ubicua. Algunos de los más notorios se indican a continuación:

- Intelligent Room [Coen, 1998]. Desarrollado por el grupo de investigación del Artificial Intelligence Laboratory en el MIT (Masachussetts Institute of Technology). Es uno de los proyectos más evolucionados en la actualidad, bajo el nombre del proyecto "Oxygen" [Look et al., 2005] [Look and Shrobe, 2007]. Se trata de una habitación que posee cámaras, micrófonos y otros sensores que realizan una actividad de interpretación con el objetivo de averiguar las intenciones de los usuarios. Se realiza interacción mediante voz, gestos y contexto.
- Smart Room [Pentland, 1996]. Desarrollado en el Media Lab del MIT bajo el grupo de investigación VisMod ("Vision and Modelling"). Mediante la utilización de cámaras, micrófonos y sensores se pretende analizar el comportamiento humano dentro del entorno. Se localiza al usuario, identificándolo mediante su voz y apariencia y se realiza un reconocimiento de gestos. La línea de investigación actual se centra en la utilización de redes de sensores para su aplicación a productividad en grandes empresas [Ara et al., 2008].
- Easy Living [Shafer et al., 1998]. Se trata de un proyecto de investigación de la empresa Microsoft con el objetivo de desarrollar "entornos activos", que ayuden a los usuarios en tareas cotidianas. Al igual que otros proyectos comentados, se intenta establecer una comunicación entre el entorno y el usuario mediante lenguaje natural. Se hace uso de visión artificial para realizar identificación de personas e interpretación de comportamientos dentro del espacio inteligente. El proyecto no posee actividad investigadora relevante desde el año 2000.

Los proyectos mencionados no incluyen robots controlados por el entorno. El uso de agentes robóticos es estudiado por un número todavía reducido de grupos de investigación.

- T.Sogo [T.Sogo et al., 1999] propone un sistema equipado con 16 cámaras fijas llamadas "Agentes de Visión", las cuales aportan la información necesaria para localizar a un grupo de robots a través de un espacio lleno de obstáculos. Los Agentes de Visión consisten en cámaras omnidireccionales no calibradas. La localización se realiza en dos pasos. En primer lugar un operador humano debe mostrar al espacio inteligente el camino que deberán seguir los robots. Una vez que el sistema ha aprendido en coordenadas de imagen el camino señalado, puede actuar sobre los robots con el objetivo de minimizar el camino que recorren en la imagen con respecto al que realizan durante el periodo supervisado.

  La técnica es similar a la utilizada en aplicaciones de servo óptico, por lo que resulta muy difícil estimar la precisión que se obtiene realmente con el sistema. Dependerá en gran medida de la técnica de comparación en el plano imagen y la posición relativa de las cámaras y los robots.
- Un trabajo más evolucionado fue propuesto en la universidad de Tokyo por Lee y Hashimoto [Hashimoto et al., 2003] [Lee et al., 2005]. En este caso se dispone de un Espacio Inteligente completo en el que se mueven robots y usuarios. Cada cámara,

unida a un sistema de procesamiento es tratado como un dispositivo inteligente conectado a una red, denominado DIND (Distributed Intelligent Network Device). Cada uno de los DIND posee capacidad de procesamiento por si solo, por lo que el espacio inteligente posee flexibilidad a la hora de incluir nuevos nodos inteligentes. Puesto que se cuenta con dispositivos calibrados, la localización se realiza en espacio métrico. Los robots están dotados de balizamiento pasivo mediante un conjunto de bandas de colores fácilmente detectables por cada DIND. En este espacio de trabajo (Ispace), se realizan experimentos de interacción entre humanos y robots y navegación con detección de obstáculos.

- Otra propuesta es el proyecto MEPHISTO ("Modular and Extensible Path Planning System using Observation") [Steinhaus et al., 1999] [Steinhaus et al., 2007] del Instituto para el Diseño de Computadores y Tolerancia a Fallos en Alemania. En este proyecto, el concepto de inteligencia distribuida se alcanza gracias a las denominadas Unidades de Procesamiento de Área Local (LAPU, "Local Area Processing Unit"), similares a los DIND de Lee y Hashimoto. Se define una unidad específica para el control del robot (RCU, "Robot Control Unit") que conecta y envía instrucciones a los diferentes robots. La localización se realiza sin marcas artificiales, usando una descripción poligonal de cada robot en coordenadas de la imagen, la cual es convertida a una posición tridimensional con un enfoque multicámara.
- En el Departamento de Electrónica de la Universidad de Alcalá existe un grupo de "Espacios Inteligentes" (grupo GEINTRA, "Grupo de Espacios Inteligentes y Transporte") [Villadangos et al., 2005] [Pizarro et al., 2005] [Fernandez et al., 2007] en el que, mediante un conjunto de sensores, se pretende realizar posicionamiento de robots móviles. Las alternativas incluyen visión artificial, ultrasonidos, infrarrojos y voz.

Los proyectos anteriormente comentados hacen hincapié en el diseño de sistemas distribuidos y flexibles, donde la inclusión de un nuevo sensor al entorno se realiza de forma dinámica sin afectar al funcionamiento del sistema. Se han desarrollado a su vez trabajos, que si bien se diferencian en el planteamiento genérico, las técnicas usadas y el objetivo se relacionan directamente con la idea propuesta. Entre dichos trabajos, se pueden citar los siguientes:

- Destaca entre otros el trabajo realizado por Hoover y Olsen [Hoover and Olsen, 2000], en el que utilizando un conjunto de cámaras con coberturas solapadas, son capaces de detectar obstáculos estáticos y dinámicos. Mediante un mapa de ocupación [Hoover and Olsen, 1999] es posible guiar un robot dentro del espacio de cobertura conjunta.

La técnica del mapa de ocupación requiere una fase previa de calibración, en la cual el espacio se encuentra libre de obstáculos. Se obtienen imágenes de referencia que serán utilizadas para la localización de obstáculos en posteriores capturas.

Mediante tablas de búsqueda se obtiene una correspondencia directa entre puntos de pantalla y las coordenadas tridimensionales de puntos pertenecientes al plano que representa el suelo del espacio. Realizando un fundido de la información de las distintas cámaras, se obtiene un mapa de aquellos puntos del suelo que tienen alta probabilidad de estar ocupados.

Uno de los principales inconvenientes de este enfoque, es la detección de objetos con bajo contraste con respecto al color del suelo en la imagen de referencia. El sistema desperdicia mucha información tridimensional mediante el mapa de ocupación que podría ser utilizada para obtener información tridimensional del entorno y de los objetos que se mueven en él.

- Un proyecto similar al anterior es el presentado en [Kruse and Wahl, 1998], denominado MONAMOVE("Monitoring and Navigation for Mobile Vehicles") y orientado a la navegación de un vehículo en un entorno industrial. En este trabajo se propone fundir la información de localización, obtenida del conjunto de cámaras distribuidas por el entorno, con sensores de ultrasonidos e infrarrojos ubicados a bordo del robot. Al igual que el trabajo de Hoover y Olsen, se utilizan tablas de búsqueda para crear un mapa de ocupación del entorno.

  Los obstáculos móviles se detectan mediante análisis diferencial en sucesivas imágenes y los estáticos son comparados con dos imágenes de referencia, obtenidas en un proceso previo de calibración. Dichas imágenes de referencia se toman bajo dos condiciones de iluminación diferentes, y se realiza una comparación ponderada con la imagen actual para obtener regiones correspondientes a objetos que existen en el entorno.

La incorporación de robots móviles al concepto de"Espacio Inteligente"es una disciplina aún emergente, en la que no se han explotado todavía las posibilidades que ofrece. Las técnicas de visión artificial aplicadas hasta la fecha, en general, requieren de balizamiento y no aprovechan toda la información de la que se dispone desde el conjunto de cámaras.

2.2. Detección de objetos basado en grid de ocupación

En el presente proyecto se propone una idea similar a la utilizada por [Hoover and Olsen, 2000], en la que, como se comentó con anterioridad, se construye un mapa de ocupación en base a la información proporcionada por un conjunto de cámaras con coberturas solapadas.

La construcción de un grid o mapa de ocupación resulta una manera eficiente de combinar la información de cámaras que se encuentran alejadas entre sí, no dependiendo de métodos que requieran complejas correspondencias entre imágenes o de modelos tridimensionales de objetos. En añadido, este tipo de técnicas admite la utilización de filtros multimodales para la detección de múltiples objetos, como pueden ser filtros de partículas, asegurando coherencia espacial y temporal.

Fuera del ámbito de los espacios inteligentes y la robótica existen numerosas publicaciones centradas en técnicas similares a la aquí propuesta. A continuación se presentan las más relevantes:

- [Berclaz et al, 2006] expone un método a través del cual un número mayor de seis personas son detectadas en base a la información obtenida en cuatro cámaras. La detección es presentada sobre un grid de ocupación.

Los autores proponen, entre otros, una optimización global de la trayectoria para resolver las ambigüedades en el seguimiento. No obstante, esto añade un retardo de cuatro segundos, lo cual hace imposible la navegación del robot.

- [Checa et al, 2003] emplea dos grid de ocupación. Uno generado a partir de un conjunto de sensores visuales, y otro a partir de un conjunto de sensores acústicos.
- [Focken and Stiefelhagen, 2002] y [Otsuka and Mukawa, 2004] trabajan con una aproximación de múltiples hipótesis probabilísticas usada para resolver ambigüedades y oclusiones.

CAPÍTULO 3. GEOMETRÍA DE LA FORMACIÓN DE LA IMAGEN

3.1. Cámara pinhole

Una cámara proporciona información del mundo 3D sobre una imagen en dos dimensiones, por lo tanto es necesario realizar una transformación de un espacio a otro. La forma en la que se procesa dicha conversión se explica mediante el modelo de la cámara que, en este caso, es el pinhole.

Lejos de tratar con lentes gruesas o finas, el modelo pinhole asume que los rayos de luz se propagan desde cada punto del espacio 3D hasta el sensor de la cámara, pasando todos ellos por un único punto.

Se empieza el estudio del modelo de una forma sencilla para más adelante añadir complejidad.

3.1.1. Idea intuitiva del modelo pinhole

Se puede tener una idea intuitiva del modelo pinhole haciendo un agujero infinitamente estrecho sobre un material infinitamente delgado tras el que se coloca una superficie sensible a la luz, siendo la superficie sensible donde se genera la imagen, figura 3.1.

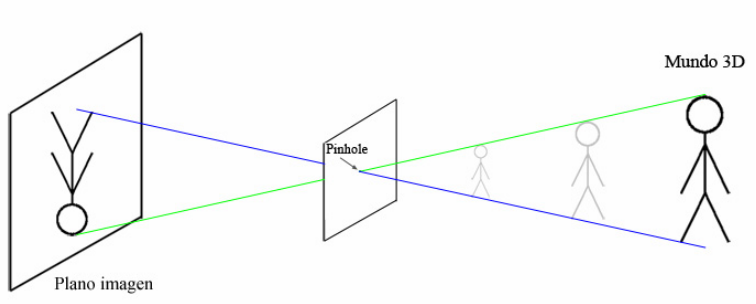


Figura 3.1 Idea intuitiva de cámara pinhole

Ésta es una forma sencilla de comprender la formación de la imagen en una cámara pinhole. No obstante, para el posterior estudio del modelo se supone que la imagen se forma delante del punto de proyección de los rayos. Así la imagen no queda invertida, figura 3.2.

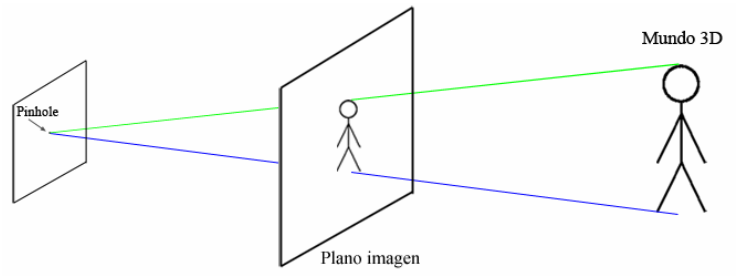


Figura 3.2 Imagen no invertida en cámara pinhole

Seguidamente se hace un estudio más detallado para obtener un modelo con el que poder trabajar.

3.1.2. Modelo de la cámara pinhole

Una vez conocida la idea intuitiva resulta sencillo ampliar el estudio. El modelo de cámara pinhole [Hartley and Zisserman, 2003] se considera como la proyección de los puntos del espacio 3D sobre un plano. El centro de proyección es el centro óptico de la cámara, *C*. El plano es el que se encuentra a una distancia *f* de dicho centro, conocido como plano imagen. Así, un punto con coordenadas $N_c = (X_c, Y_c, Z_c)^T$ se proyecta en el plano imagen en el cruce de la recta $\overline{N_c C}$ y el plano imagen. A este punto se le conoce como n_p. Se muestra en la figura 3.3.

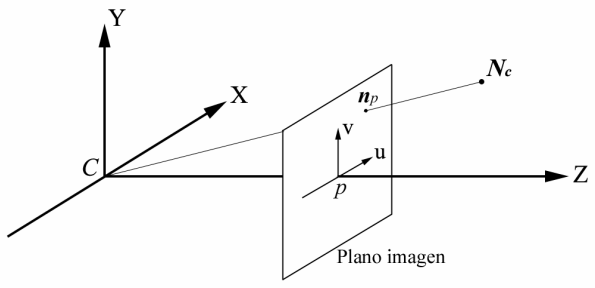


Figura 3.3 Geometría de formación de la imagen en una cámara pinhole

Donde:

C → Centro óptico o centro de proyección central. Es el centro de la cámara y fija el origen de coordenadas de ésta. Por él pasan todos los rayos de luz procedentes del espacio que van a generar la imagen en el sensor.

Z → Eje principal o rayo principal. Es la línea perpendicular al plano imagen que pasa por el centro óptico.

p → Punto principal, coincide con la intersección entre el eje principal y el plano imagen y es el centro de éste último.

N_c → Punto cualquiera en el espacio, con origen en *C*, que se proyecta sobre el plano imagen.

n_p → Punto del espacio proyectado sobre el plano imagen.

3.1.3. Matriz de proyección

Como se ha visto, un punto cualquiera N_c se proyecta sobre el plano imagen en su camino hacia el centro óptico, de tal manera que sus coordenadas se pueden deducir a la vista de la figura 3.4.

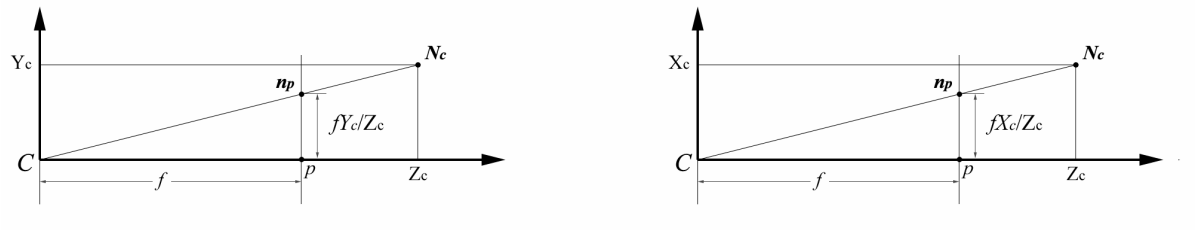


Figura 3.4 Coordenadas en el plano imagen. A la izquierda coordenada Y, a la derecha coordenada X.

Donde:

f → Distancia focal.

X_c, Y_c → Coordenada *X, Y* del punto N_c respectivamente.

Queda así demostrado que el punto $N_c = (X_c, Y_c, Z_c)$ es proyectado sobre el plano imagen en las coordenadas $n_p = (fX_c/Z_c, \ fY_c/Z_c)$. Se genera un espacio bidimensional a partir de otro tridimensional, perdiendo la información de profundidad:

$$\left(X_c, Y_c, Z_c\right)^T \rightarrow \left(\frac{fX_c}{Z_c}, \frac{fY_c}{Z_c}\right)^T \qquad (3.1)$$

Representando los puntos mediante coordenadas homogéneas resulta una expresión donde la relación entre ambos puntos es más sencilla:

$$\begin{pmatrix} fX_c \\ fY_c \\ Z_c \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{3.2}$$

O lo que es lo mismo:

$$\tilde{n}_p = P\tilde{N}_c \tag{3.3}$$

Donde la matriz *P* es conocida como la matriz homogénea de proyección de la cámara.

$$P = diag(f, f, 1)[I \mid 0] = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \tag{3.4}$$

3.1.4. Cambio de origen y escala en el plano imagen

Hasta ahora se ha asumido que el origen de coordenadas en el plano imagen coincide con el punto principal, de modo que la coordenada (0, 0) se encuentra justo en el centro. En la práctica se puede manejar un origen de coordenadas distinto. Generalmente éste queda situado en un extremo, figura 3.5.

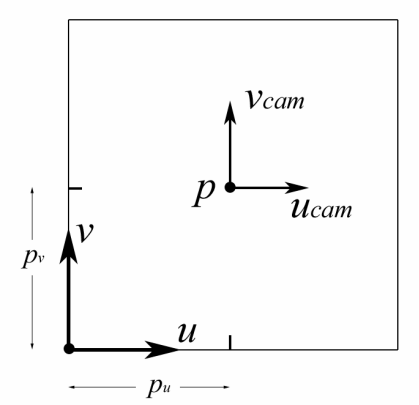


Figura 3.5 Coordenadas de imagen (u, v) y cámara (u_{cam}, v_{cam})

La transformación se modela sumando a la coordenada *(u, v)* el valor (p_u, p_v) respectivamente:

$$N_c = (X_c, Y_c, Z_c)^T \rightarrow \left(\frac{fX_c}{Z_c} + p_u, \frac{fY_c}{Z_c} + p_v \right)^T = n_K \tag{3.5}$$

Donde p_u, p_v son las coordenadas del punto principal referidas al nuevo origen, como se mostró en la figura 3.5.

Al punto obtenido se le llama n_K. Es decir, n_K es el mismo punto en el espacio que n_p pero referido al nuevo origen. En la figura 3.6 se aclara el significado de N_c, n_p, n_K.

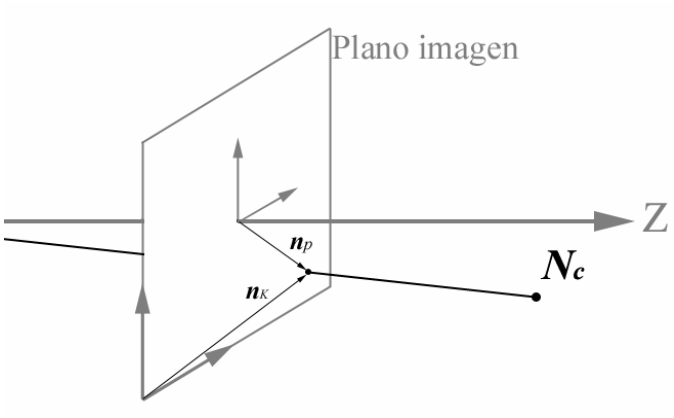


Figura 3.6 Puntos N_c, n_p, n_K

Nuevamente en coordenadas homogéneas se expresa en forma matricial:

$$\tilde{n}_K = \begin{pmatrix} fX_c + Z_c p_u \\ fY + Z_c p_v \\ Z_c \end{pmatrix} = \begin{bmatrix} f & & p_u & 0 \\ & f & p_v & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{3.6}$$

O lo que es lo mismo:

$$\tilde{n}_K = K[I \mid 0]\tilde{N}_c \tag{3.7}$$

Donde K es conocida como la matriz de calibración de la cámara.

$$K = \begin{bmatrix} f & & p_u \\ & f & p_v \\ & & 1 \end{bmatrix} \tag{3.8}$$

Por lo tanto, un punto del espacio N_c se representa en el plano imagen como n_K según la relación siguiente:

$$\tilde{n}_K = K[I \mid 0]\tilde{N}_c \tag{3.9}$$

Ahora bien, dado que se está hablando del modelo de una cámara, no se debe olvidar que ésta estará compuesta por un sensor con un número finito de puntos y que su escala no tiene por qué coincidir con las fijada. Es decir, un píxel no tiene por qué coincidir con un milímetro por ejemplo. Incluso, la anchura de dichos píxeles no tiene por qué ser igual a la altura, de modo que la escala de los ejes será distinta.

Para hacer este cambio de escala se debe tener en cuenta el número de píxeles por unidad de longitud, tanto en la dirección *u* como en la *v*, valor que será asignado a m_u, m_v. En la figura 3.7 se muestra un ejemplo de estos valores siendo $m_u = 20$, $m_v = 10$.

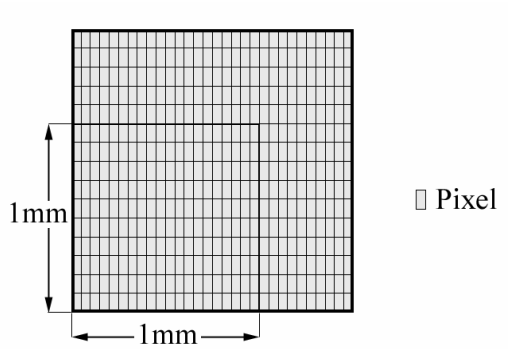


Figura 3.7 Ejemplo de CCD de una cámara donde los píxeles no son cuadrados

Una vez conocidos los valores de m_u, m_v se multiplican por la matriz de calibración de la cámara:

$$K = \begin{bmatrix} m_u & & \\ & m_v & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_u \\ & f & p_v \\ & & 1 \end{bmatrix} \tag{3.10}$$

El resultado se conoce nuevamente como matriz de calibración, donde $\alpha_u = fm_u$ y $\alpha_v = fm_v$ representan la distancia focal en píxeles y $u_0 = m_u p_u$, $v_0 = m_v p_v$ representan el punto principal.

$$K = \begin{bmatrix} \alpha_u & & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix} \tag{3.11}$$

Por último, se introduce un parámetro que modela la no perfecta perpendicularidad de los ejes. Este parámetro se representa por la letra *s* (skew).

$$K = \begin{bmatrix} \alpha_u & s & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix} \tag{3.12}$$

En cualquier caso la expresión 3.9 no se modifica. Es decir, un punto del espacio N_c se representa en el plano imagen como n_K, según la ecuación:

$$\tilde{n}_K = K[I \mid 0]\tilde{N}_c$$

3.1.5. Rotación y traslación del sistema de referencia

Hasta el momento se ha situado el origen de coordenadas de los puntos del espacio 3D en el centro óptico. En ocasiones es preferible fijar un sistema de referencia distinto siendo necesaria la conversión de uno a otro, figura 3.8.

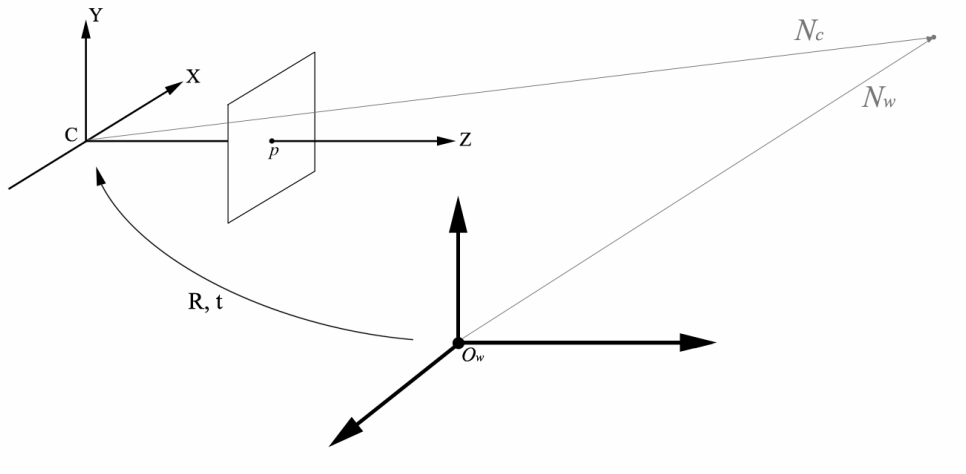


Figura 3.8 Cambio de sistema de referencia

A partir de este momento se conoce al nuevo sistema de referencia como el origen del mundo, O_w.

Si se toma N_c como un punto del espacio referido a C, y N_w como ese mismo punto referido al nuevo sistema de referencia, con origen en O_w. La relación entre ambos es:

$$N_c = R\left(N_w - C_o\right) \tag{3.13}$$

Donde:

O_w → Origen del nuevo sistema de referencia, origen del mundo, como se muestra en la figura 3.8.

R → Matriz de dimensión 3 x 3, llamada de rotación. Representa el cambio de orientación para adaptar un sistema en otro.

C_o→ Coordenadas del centro de la cámara referido al nuevo sistema de referencia.

Representando los puntos de forma homogénea:

$$\tilde{N}_c = \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{bmatrix} R & -RC_o \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \left[R \mid -RC_o\right]\tilde{N}_w \tag{3.14}$$

Donde se introduce el concepto de matriz de traslación, *t*, quedando:

$$\tilde{N}_c = [R \mid t]\tilde{N}_w \tag{3.15}$$

Para finalizar el apartado se presenta una expresión que relaciona directamente las coordenadas en el plano imagen N_K a partir del punto del espacio referido a un sistema de referencia elegido N_w. Ésta se obtiene al sustituir la expresión 3.9 en la 3.14:

$$\tilde{n}_K = K[R \mid t]\tilde{N}_w \tag{3.16}$$

Se recuerda el significado de cada término:

K → Matriz de calibración de la cámara.

R → Matriz de rotación.

t → Matriz de traslación.

$\tilde{N}_w$ → Punto del espacio referido a un sistema de coordenadas elegido, en coordenadas homogéneas.

$\tilde{n}_k$ → Punto en el plano imagen con origen en un extremo de la misma, fruto de la proyección de N_w sobre el centro óptico, en coordenadas homogéneas.

3.2. Calibración de la cámara

Se ha explicado el modelo por el cual una escena tridimensional en el mundo 3D genera una imagen bidimensional en la cámara. Se ha conseguido incluso encontrar las ecuaciones que relacionan cada punto del espacio con cada píxel en la imagen. No obstante aún queda un paso importante por dar, encontrar los valores que rigen las transformaciones. Es decir, aún falta la calibración de la cámara.

Se distinguen dos familias de parámetros a conocer en la calibración: parámetros intrínsecos y parámetros extrínsecos.

Los parámetros intrínsecos son aquellos que determinan las características internas de la cámara $(\alpha_x, \alpha_y, u_o, v_o, s)$.

Los parámetros extrínsecos están relacionados con la orientación y desplazamiento del sistema de referencia de la cámara respecto al elegido en el espacio como origen del mundo. Estos son por lo tanto externos a la cámara y son la matriz de rotación $(R_{11}, R_{12}, ..., R_{33})$ y traslación (t_x, t_y, t_z).

El método de calibración empleado en el presente proyecto es el implementado en la toolbox de calibración de Matlab [Bouguet, 2008], el cual se basa en lo expuesto en las publicaciones [Zhang, 1999] y [Heikkilä and Silvén, 1997]. El proceso es el siguiente:

Primero se estiman las homografías de un plano dado en diferentes posiciones [apéndice A de Zhang, 1999]. Dicho plano es un patrón como el mostrado en la figura 3.9, donde se conoce con exactitud las dimensiones de anchura y altura, *dx*, *dy*, así como el número de filas y columnas.

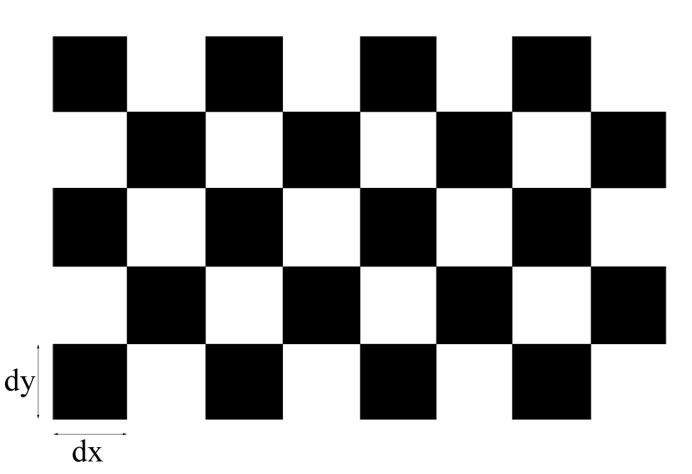


Figura 3.9 Patrón de calibración

A partir de las homografías calculadas se hace una primera estimación de los parámetros intrínsecos de la cámara, *K*, y extrínsecos a cada plano, *R*, *t*.

El siguiente paso se centra en obtener una mayor precisión en la calibración. Para ello se toman por un lado los *m* puntos de las distintas *l* imágenes, y por otro los mismos *m* puntos proyectados mediante las expresiones del modelo de la cámara. Lo que se busca es minimizar el cuadrado de la diferencia de ambos valores, es decir, se minimiza la expresión 3.16.

$$\sum_{i=1}^{l}\sum_{j=1}^{m}\left\|n_{ij}-\underline{n}_{ij}\right\|^{2} \qquad (3.17)$$

Donde:

$\underline{n}_{ij}$ → Coordenada calculada del punto *j*-ésimo de la imagen *i*-ésima a partir de las expresiones del modelo de la cámara.

n_{ij} → Coordenada del punto *j*-ésimo de la imagen *i*-ésima capturada por la cámara.

Para minimizar la expresión 3.16 se precisan unos valores iniciales de los parámetros intrínsecos y extrínsecos, éstos son los estimados en el primer paso del método. Además, la toolbox de Matlab emplea en las expresiones de modelado de la cámara unos parámetros relativos a la distorsión tangencial y radial.

En la práctica, para aplicar el método anterior se toman una serie de imágenes del patrón de la figura 3.9 en distintas posiciones. En [Zhang, 1999] se propone que el número de capturas debe ser al menos de tres, no obstante, a mayor número de imágenes mayor precisión en la calibración. Dichas imágenes son empleadas para calibrar los parámetros intrínsecos y en ellas deben aparecer perfectamente definidos los cuadros que forman el patrón. Por ejemplo, en la figura 3.10 se muestran cuatro de las treinta y cuatro capturas empleadas para calibrar una cámara real.

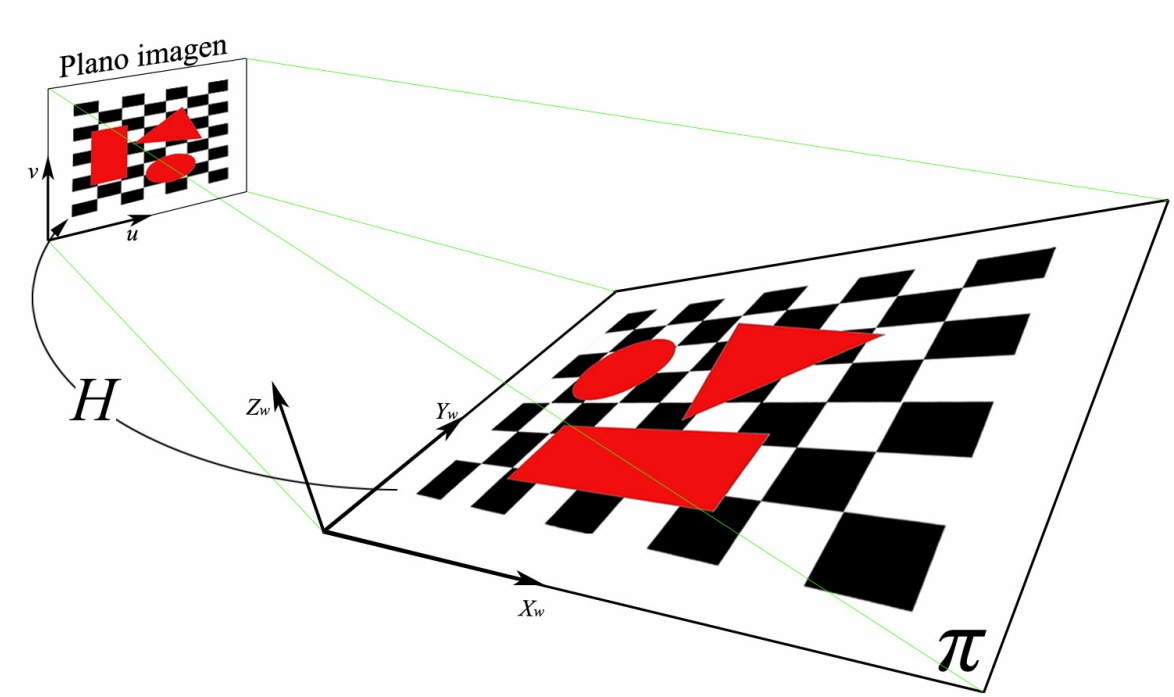
Figura 3.10 Cuatro capturas del patrón de calibración para parámetros intrínsecos

Para calibrar los parámetros extrínsecos se emplea igualmente el patrón anterior. Éste se fija sobre el sistema de coordenadas que se desee y se toma nuevamente una captura. En la figura 3.11 se muestra a la izquierda la captura tomada, y a la derecha el sistema de coordenadas superpuesto.

Figura 3.11 Imagen usada para calibrar parámetros extrínsecos y sistema de coordenadas resultante

3.3. Obtención de la homografía a partir de la calibración de la cámara

Dado un modelo de cámara pinhole con sus correspondientes matrices *K*, *R*, *t*; y dado un plano en el espacio tridimensional, π, la matriz de homografía, *H*, relaciona los puntos del plano π con los puntos en el plano imagen, figura 3.12.



3.12 Relación de la matriz de homografía

Para generar dicha matriz de homografía se sitúa el sistema de coordenadas de tal manera que el eje Z_w sea perpendicular al plano. Así los puntos contenidos en dicho plano tienen la coordenada Z_w nula.

$$N_w = (X_w, Y_w, 0)^T$$

Utilizando esta información se desarrolla la expresión 3.16:

$$\tilde{n}_K = K[R \mid t]\tilde{N}_w \quad \Rightarrow \quad \tilde{n}_K = KR\begin{pmatrix} X_w \\ Y_w \\ 0 \end{pmatrix} + Kt$$

$$\tilde{n}_K = \begin{pmatrix} \alpha_u & 0 & uo \\ 0 & \alpha_v & vo \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}\begin{pmatrix} X_w \\ Y_w \\ 0 \end{pmatrix} + \begin{pmatrix} \alpha_u & 0 & uo \\ 0 & \alpha_v & vo \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Sustituyendo la matriz *KR* por la matriz *C* y operando.

$$\tilde{n}_K = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ 0 \end{pmatrix} + \begin{pmatrix} K_1 t \\ K_2 t \\ K_3 t \end{pmatrix}$$

$$\tilde{n}_K = \begin{pmatrix} C_{11} & C_{12} & K_1 t \\ C_{21} & C_{22} & K_2 t \\ C_{31} & C_{32} & K_3 t \end{pmatrix} \overline{N}_w \tag{3.18}$$

Siendo $\overline{N}_w$ igual a $\tilde{N}_w$ tras eliminar la tercera componente. Esto se hace puesto que la coordenada Z_w ya no proporciona información, para todos los puntos es igual a cero.

Se obtiene por tanto la matriz homografía, *H*.

$$H = \begin{pmatrix} C_{11} & C_{12} & K_1 t \\ C_{21} & C_{22} & K_2 t \\ C_{31} & C_{32} & K_3 t \end{pmatrix} \tag{3.19}$$

Donde:

C→ Matriz resultante de la multiplicación de la matriz de calibración de la cámara, *K*, y la matriz de rotación, *R*.

K_i→ Fila i-ésima de la matriz de calibración.

De esta forma se demuestra que un punto en un plano dado, N_w, se proyecta en la imagen en el punto n_k según la ecuación 3.20.

$$\tilde{n}_k = H\overline{N}_w \tag{3.20}$$

Una vez conocida la relación entre los puntos del plano elegido en el espacio y el plano imagen se puede seguir ampliando el estudio. Por ejemplo, es posible que se deseen referir los puntos del plano π a un origen distinto y con un factor de escala apropiado. Dicha transformación se modela a través de la matriz *M:*

$$M = \begin{pmatrix} m_{Xw} & & D_{Xw} \\ & m_{Yw} & D_{Yw} \\ & & 1 \end{pmatrix} \tag{3.21}$$

Multiplicando dicha matriz directamente a los puntos $\overline{N}_w$:

$$\tilde{n}_k = HM\overline{N}_w \tag{3.22}$$

Donde:

m_{Xw}, m_{Yw} → Modelan el cambio de escala en las direcciones X_w, Y_w respectivamente.

D_{Xw}, D_{Yw} → Modelan el cambio de origen en las direcciones X_w, Y_w respectivamente.

Se expresa la transformación gráficamente en la figura 3.13

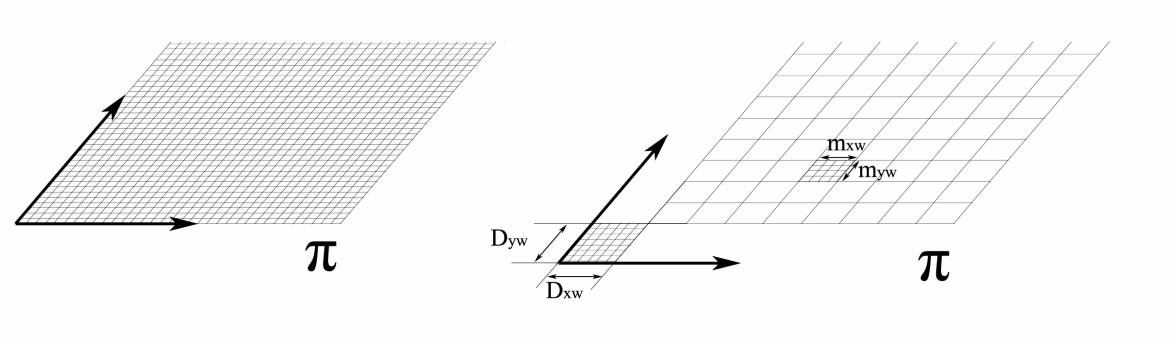


Figura 3.13 A la izquierda el plano dividido en la escala original. A la derecha origen de coordenadas y escalado tras aplicar la matriz M.

Por ejemplo si se desea referir el plano π en píxeles los parámetros de la matriz son:

m_{Xw}→ Número de milímetros por píxel en la dirección X_w

m_{Yw}→ Número de milímetros por píxel en la dirección Y_w.

D_{Xw}→ Desplazamiento del origen en la coordenada X_w, medido en milímetros.

D_{Yw}→ Desplazamiento del origen en la coordenada Y_w, medido en milímetros.

Ahora bien, en ocasiones es necesaria la relación inversa, es decir, se desea conocer el punto en el plano π a partir del punto observado en el plano imagen. Para ello no es necesario más que multiplicar por la matriz inversa.

$$\overline{N}_w = (HM)^{-1} \tilde{n}_k , \qquad (3.23)$$

donde se obtiene el punto en el plano referido al origen elegido mediante (D_{Xw},D_{Yw}) en las dimensiones impuestas por (m_{Xw},m_{Yw})

3.3.1. Puntos fuera del plano y matriz de homografía

Una vez vista la definición de homografía queda quizá una cuestión por resolver, y es que a priori no se conoce el efecto sobre los puntos externos al plano π.

Supóngase la matriz de homografía H que relaciona los puntos del plano π y el plano imagen de una cámara dada; y supóngase un punto en el espacio N_w fuera del plano π, es decir, con coordenada Z_w no nula.

Con el punto N_w y el centro óptico de la cámara C se genera una recta conocida como $\overline{N_wC}$, donde todos los puntos de dicha recta se proyectan en el mismo punto en el plano imagen, en n_k. En concreto, el punto de interés es el que pertenece tanto a la recta $\overline{N_wC}$ como al plano π, se conoce este punto como N'_w. En la figura 3.14 se apoya la explicación anterior.

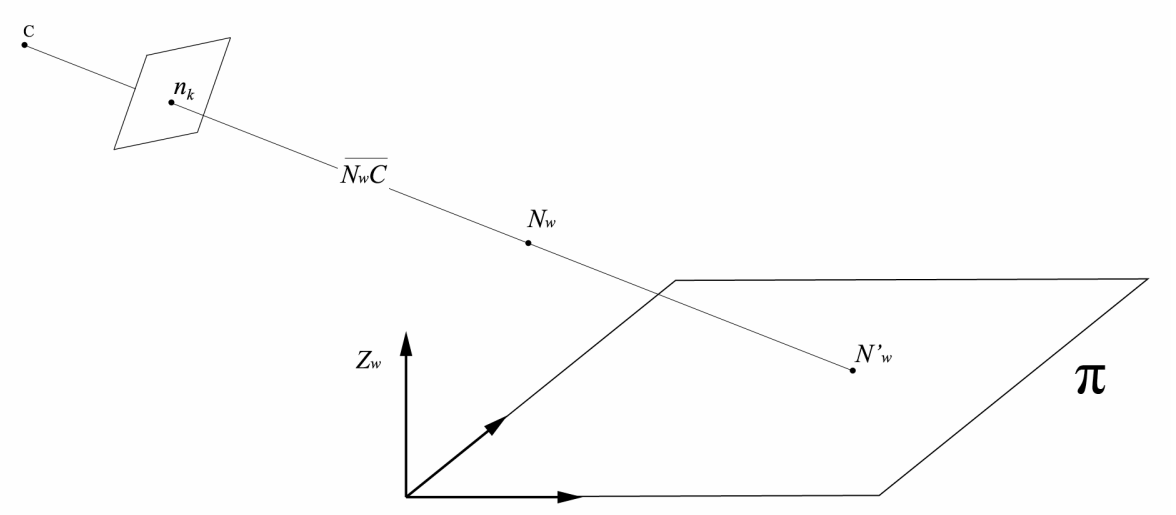


Figura 3.14 Proyección idéntica en el plano imagen para todos los puntos de la recta $\overline{N_wC}$

Si se aplica ahora la matriz homografía inversa H^{-1} al punto n_k se tiene como resultado el punto en el plano π que genere n_k, es decir, se tiene como resultado N'_w.

Por lo tanto queda demostrado que la matriz H^{-1} proyecta los puntos externos al plano π en la intersección de la recta $\overline{N_wC}$ con el plano π. Es decir, el resultado depende de la rotación y traslación de la cámara con respecto al plano.

3.4. Generación de un grid de ocupación a partir de múltiples cámaras

A partir del concepto de homografía se desprenden numerosas aplicaciones, una de éstas es la formación de un grid de ocupación.

Se entiende como grid la división de un plano en forma de cuadrícula, donde la resolución del plano se ve alterada y adaptada a la nueva subdivisión.

Se entiende como ocupación de un plano las partes del mismo en las que se encuentran situados objetos.

Se entiende como grid de ocupación la ocupación de un plano representado sobre un grid del mismo, resultando una imagen binaria de significado ocupado o no ocupado. Un ejemplo de esto se muestra en la figura 3.15.

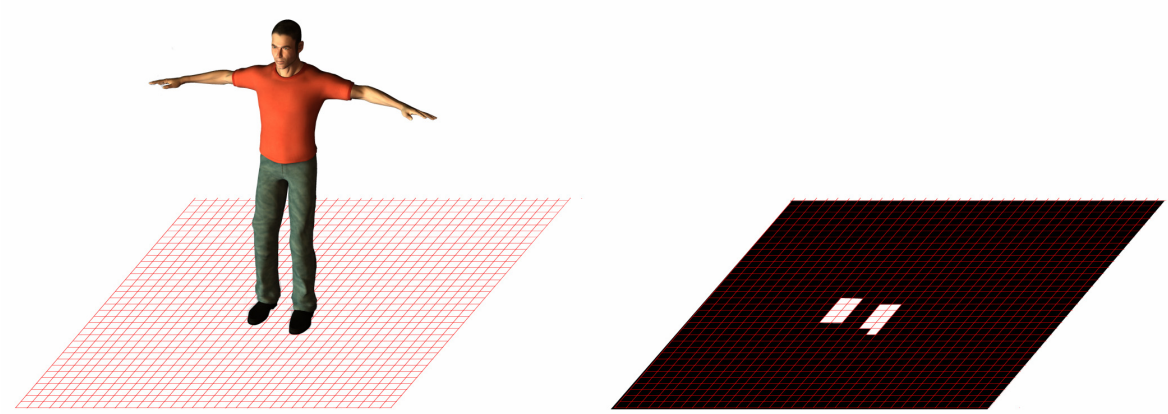


Figura 3.15 A la izquierda una persona sobre el suelo. A la derecha el grid de ocupación del plano del suelo, en blanco significa ocupado, en negro significa no ocupado

Ahora bien, para formar dicho grid hay que apoyarse en lo visto en el apartado 3.3.1, donde se demuestra cómo un punto fuera del plano π se proyecta, a través de la homografía inversa H^{-1}, en un punto del plano π que depende de la colocación de la cámara.

Supónganse ahora L cámaras calibradas, donde para todas ellas se generan las matrices de homografía referidas al mismo plano π con el mismo origen de coordenadas, H_i $i=1..L$.

De este modo, aplicando ${H^{-1}}_i$ a las L imágenes de las L cámaras se obtienen las L proyecciones sobre el plano π, donde únicamente los puntos que originariamente estén en el plano π coinciden en dichas proyecciones. Esto es debido justamente a que dichos puntos son los únicos que no dependen de la colocación de la cámara.

Una vez hechas las L transformación de las L imágenes, el grid de ocupación del plano π viene dado por la superposición de las L transformaciones; donde las zonas ocupadas son aquellas en las que las L superposiciones coincidan.

Por ejemplo, supónganse dos cámaras donde se obtienen dos imágenes de una misma escena, figura 3.16.

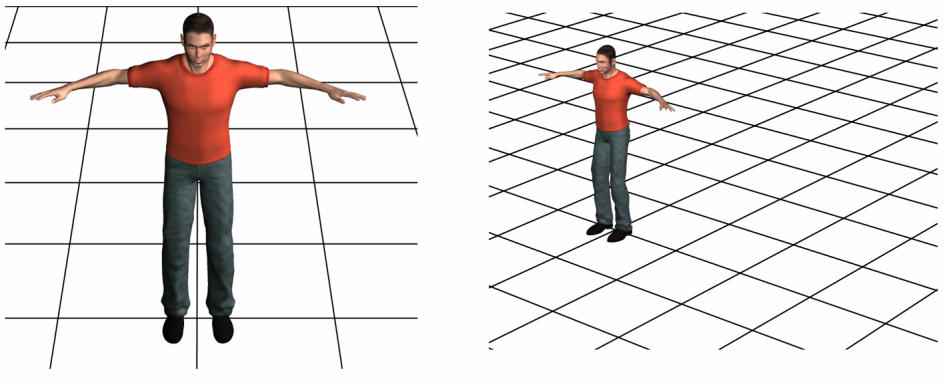


Figura 3.16 Dos imágenes de la misma escena proporcionadas por dos cámaras

Y supónganse las proyecciones de dichas imágenes sobre el plano del suelo a través de las matrices inversas de homografía, figura 3.17.

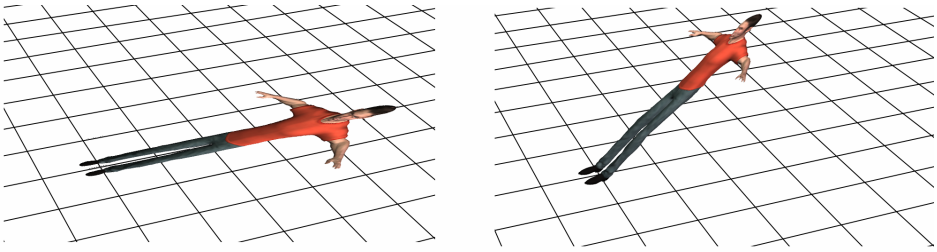


Figura 3.17 Resultado de aplicar la homografía inversa para el plano del suelo a la figura 3.16

Como la matriz homografía se ha definido para el plano del suelo, entonces únicamente los puntos que estuviesen sobre éste deben permanecer en el mismo lugar, de tal manera que si se superponen ambas imágenes solo coinciden dichos puntos.

Si se genera una nueva imagen con las coincidencias se obtiene justamente el grid de ocupación para el plano del suelo.

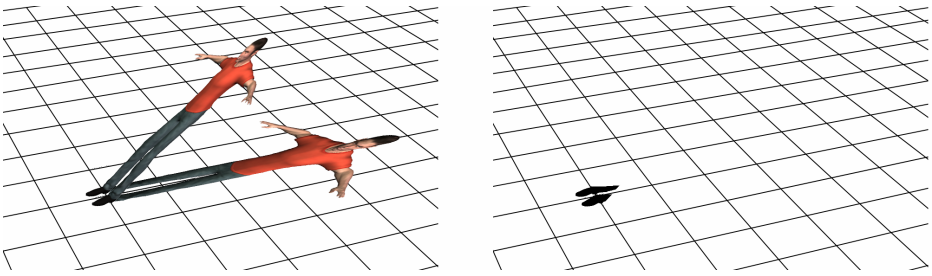


Figura 3.18 A la izquierda superposición de imágenes de figura 3.17. A la derecha las coincidencias

CAPÍTULO 4. ESPACIOS DE COLOR

4.1. El color

El color es una de las propiedades de los materiales que depende de las longitudes de onda que el cuerpo es capaz de absorber o reflejar en el espectro visible, de tal manera que, en un objeto blanco ninguna componente frecuencial es absorbida así como en un objeto negro ninguna es reflejada. Un color intermedio sería fruto de la reflexión de ciertas longitudes de onda que al combinarse lo generan. Por ejemplo, los objetos verdes reflejan la luz con longitudes de onda en el rango de 500 a 570 nm y absorben gran cantidad de energía en otras [Pajares y de la Cruz, 2001]

Figura 4.1 Espectro visible

En muchas ocasiones no es práctico manejar el color especificando las longitudes de onda reflejadas ni absorbidas con lo que se debe buscar otro método para su representación, esto es precisamente lo que se resuelve mediante los espacios de color. Se puede definir espacio de color como las diferentes bases matemáticas que pueden ser útiles para representar información luminosa [Mazo et al, 1996].Su principal función es la de facilitar la especificación de colores de una forma estandarizada.

Existen un gran número de espacios de color. A continuación se presentan los utilizados a lo largo del proyecto.

4.2. RGB

El espacio de color RGB se basa en la combinación de los tres colores primarios: rojo(R), verde(G) y azul(B), generando todos los demás a partir de éstos. Se escogen dichos colores debido a que corresponden aproximadamente con las longitudes de onda que excitan a las células sensitivas en los ojos humanos.

La forma en la que se define un color viene dado por la representación en un espacio tridimensional, donde sobre cada eje se coloca uno de los tres colores primarios. Así, cada punto en dicho espacio corresponderá a un color distinto, siendo sus coordenadas el valor correspondiente de color en cada eje. Por ejemplo, un valor nulo para un eje querrá decir que no se utiliza dicho color.

Un aspecto importante es que los niveles de grises están contenidos en la diagonal principal tal y como se ve en la figura 4.2.

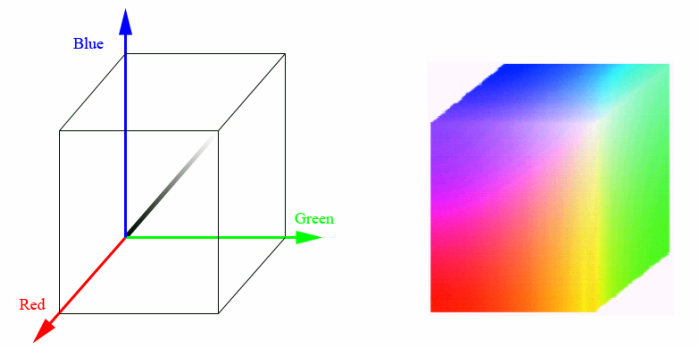


Figura 4.2 Espacio de color RGB

4.3. YUV

Este espacio de color, como se ve en la figura 4.3, no es más que un movimiento del cubo RGB.

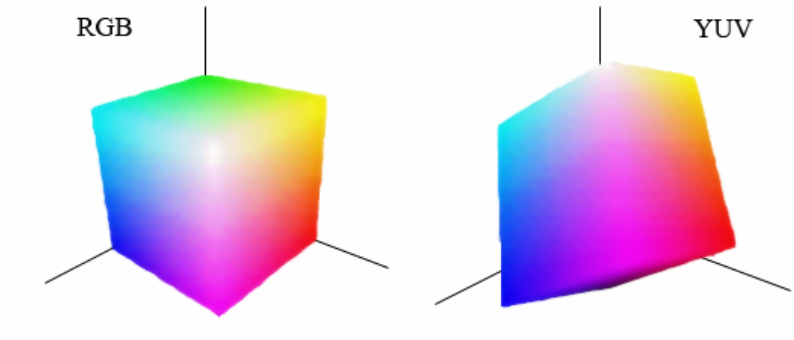


Figura 4.3 Comparación de los espacios RGB, YUV

Dicha transformación viene regida por la siguiente ecuación:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Quizá lo más interesante de YUV para este proyecto sea la colocación de los niveles de grises en el espacio, éstos ahora quedan directamente representados sobre uno de los tres ejes, el *Y*, figura 4.4.

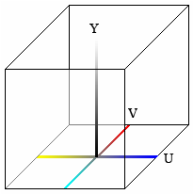


Figura 4.4 Escala de grises sobre el eje Y

4.4. HSI/HSV

El modelo HSI [Pajares y de la Cruz, 2001], matiz (H), saturación (S) e intensidad (I), se convierte en una excelente herramienta para desarrollar algoritmos de procesamiento de imágenes basados en alguna de las sensaciones de color del sistema visual humano. Esto es debido a que los humanos perciben el color mediante las componentes de matiz y saturación, que son precisamente dos de las coordenadas de este espacio de color. La tercera coordenada es la intensidad, que se encuentra separada de la información de color en la imagen. La idea fundamental es utilizar las propiedades del color del mismo modo que lo haría una persona.

Se presenta a continuación el espacio de color HSI, figura 4.5.

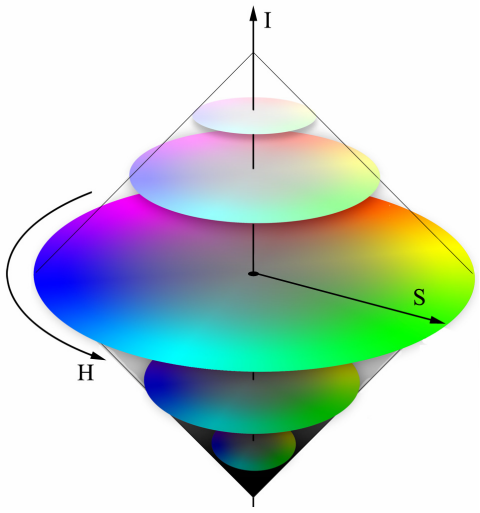


Figura 4.5 Espacio de color HSI

Donde:

H→ Matiz, se refiere a la frecuencia dominante del color dentro del espectro visible. Se mide como el ángulo formado respecto al color rojo, es decir, para H=0º el color es rojo.

S→ Saturación, se refiere al brillo del color, entre gris y blanco. Se mide como la distancia al centro de la circunferencia.

I→ Intensidad, se refiere al lugar de la escala entre blanco y negro donde se encuentra el color. Se mide con respecto a la línea perpendicular que atraviesa las circunferencias por el centro de las mismas.

El modelo HSV es similar al HSI, con la diferencia que está formado por un único cono en vez de dos, Figura 4.6.

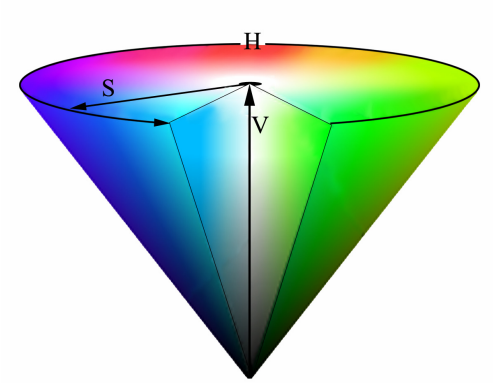


Figura 4.6 Espacio HSV

4.5. Espacio invariante a la iluminación

Antes de abordar este espacio es necesario conocer una serie de conceptos para entenderlo, estos son: superficie lambertiana, radiador planckiano y cámara de banda estrecha.

Superficie lambertiana → [Murdoch, 1985] Una superficie perfectamente difusora es aquélla que emite o refleja el flujo luminoso en forma tal que ésta presenta la misma luminancia independientemente del ángulo de visión. Tal superficie se denomina lambertiana porque responde a la ley de Lambert [Brown, 1966].

Radiador planckiano → Radiador térmico que absorbe completamente todas las radiaciones que inciden sobre él, cualquiera que sea su longitud de onda, dirección de incidencia o de polarización, también conocido como cuerpo negro. La radiación del cuerpo negro está definida por la ley de radiación de Plank. Muestra el incremento de la radiación que se convierte en espectros visible e infrarrojo en función del incremento de la temperatura, de ahí el concepto de temperatura de color. En la figura 4.7 se representan las diferentes distribuciones espectrales para diferentes temperaturas.

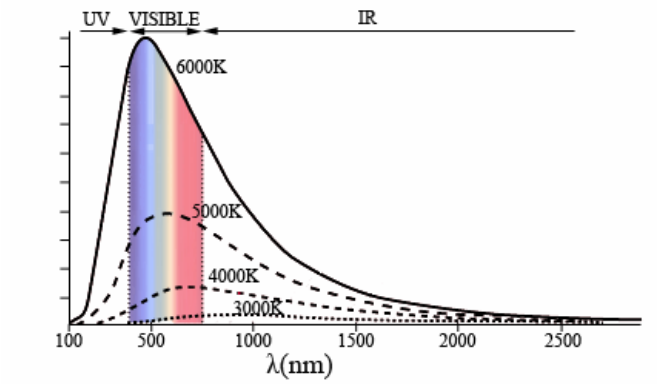


Figura 4.7 Diagrama de radiación de un cuerpo negro a distintas temperaturas

Cámara de banda estrecha → Cámara que centra la absorción en ciertas longitudes de onda con un ancho de banda infinitamente estrecho, es decir, su respuesta frecuencial se puede modelar por una serie de deltas, figura 4.8.

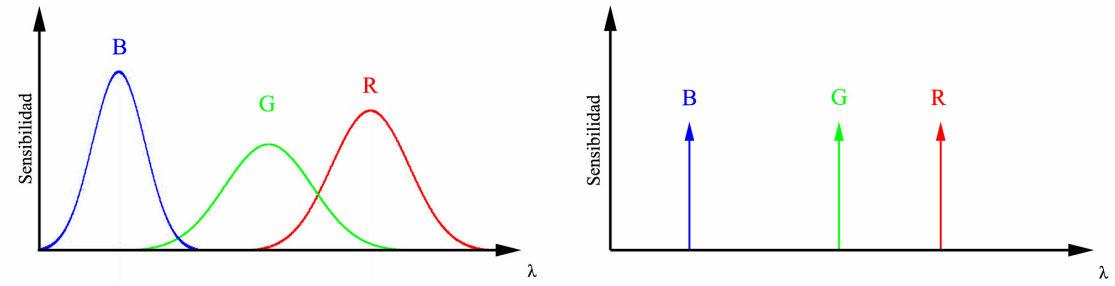


Figura 4.8 Sensibilidad de una cámara. A la izquierda una posible respuesta real, a la derecha respuesta de banda estrecha

Ahora bien, suponiendo una cámara de banda estrecha enfocando a una superficie lambertiana iluminada por un cuerpo negro a una temperatura *T*; las componentes de color de un punto en el sensor de dicha cámara vendrán dadas por la siguiente expresión:

$$\rho_k = \sigma S(\lambda_k) E(\lambda_k, T) Q_k \delta(\lambda - \lambda_k) \qquad k = 1,2,3 \qquad (4.1)$$

Donde:

$\sigma S(\lambda_k)$ → Función de reflactancia espectral de la superficie lambertiana.

$Q_k \delta(\lambda - \lambda_k)$ → Respuesta espectral en el sensor de la cámara para cada canal de color, k, centrado en la longitud de onda λ_k.

$E(\lambda_k, T)$ → Poder emisivo espectral del modelo planckiano, aproximada, para una temperatura de color entre 2500 y 10000K, por la expresión siguiente:

$$E(\lambda_k, T) = I c_1 \lambda^{-5} e^{\left(\frac{-c_2}{T\lambda}\right)}$$

I → Término que representa la intensidad de luz global.

c_1, c_2 → Constantes de valor $3.742 \cdot 10^{-16}$ Wm^2 y $1.438 \cdot 10^{-2}$ mK respectivamente.

Por lo tanto se puede desarrollar la expresión 4.1 en la siguiente:

$$\rho_k = \sigma I c_1 (\lambda_k)^{-5} e^{\left(\frac{-c_2}{T\lambda_k}\right)} S(\lambda_k) Q_k \qquad k = 1,2,3 \qquad (4.2)$$

Teniendo modelada la formación del color en un punto, se puede construir un espacio de color invariante a la iluminación [Finlayson et al, 2002].

Este espacio de color se fundamenta en la explicación que sigue:

Primeramente se componen dos coordenadas, (χ_1, χ_2):

$$\chi_1 = \log\left(\frac{\rho_1}{\rho_3}\right) = \log\left(\frac{s_1}{s_3}\right) + \frac{(e_1 - e_3)}{T} \qquad (4.3)$$

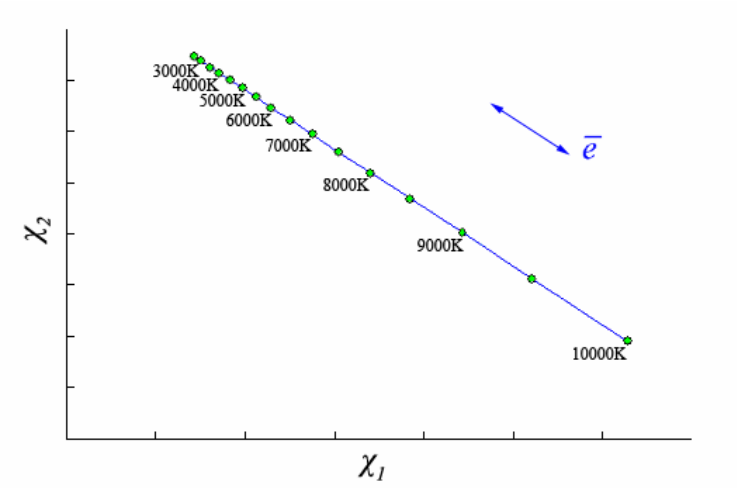
$$\chi_2 = \log\left(\frac{\rho_2}{\rho_3}\right) = \log\left(\frac{s_2}{s_3}\right) + \frac{(e_2 - e_3)}{T} \qquad (4.4)$$

Donde:

$e_k = -\frac{c_2}{\lambda_k}$ → Únicamente depende de la respuesta espectral de la cámara.

$s_k = c_1 \lambda_k^{(-5)} S(\lambda_k) Q_k$ → No depende la temperatura de color *T*.

Estas dos coordenadas forman un vector bidimensional $\chi = (\chi_1, \chi_2)$ que tiene como dirección $\bar{e} = (e_1 - e_3, e_2 - e_3)$. De este modo, una superficie iluminada a distintas temperaturas *T* genera una recta con dirección $\bar{e}$ en un espacio de dos dimensiones con ejes (χ_1, χ_2). En la figura 4.9 se presenta un ejemplo de la recta obtenida para un rango de temperatura de color desde 2500 a 10000K.



4.9 Valores de χ para temperaturas entre 2500 y 10000K

Si se toma una recta perpendicular al vector $\bar{e}$ con dirección $\bar{e}^{\perp} = (\cos(\theta), \sin(\theta))$, y se proyectan los puntos obtenidos sobre ésta se tiene lo mostrado en la figura 4.10.

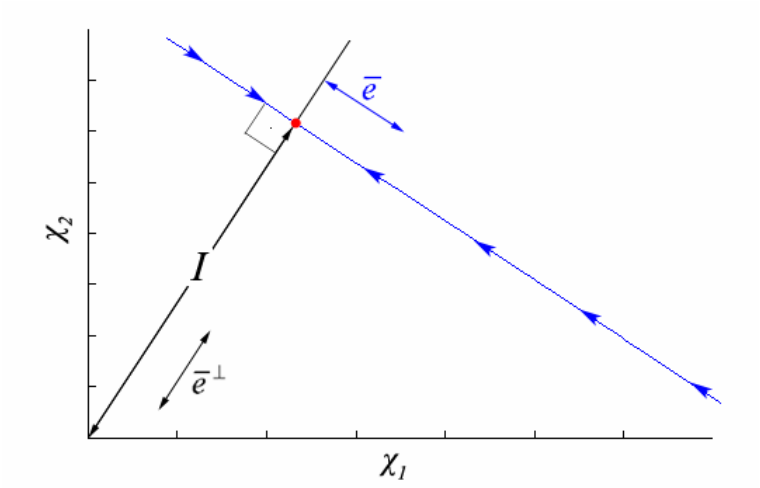


Figura 4.10 Proyección de los puntos de una superficie iluminada a distintas temperaturas sobre la recta definida por el vector $\bar{e}^{\perp}$

Donde todos los puntos de una misma superficie iluminada con distintas temperaturas se representan con el mismo valor, *I*, siendo éste el producto escalar del vector $\chi = (\chi_1, \chi_2)$ con la recta de dirección $(\cos(\theta), \sin(\theta))$, es decir, el la proyección de uno sobre otro.

$$I = \chi_1 \cos(\theta) + \chi_2 \sin(\theta) \tag{4.5}$$

En la figura 4.11 se presentan varias superficies con iluminaciones entre 2500 y 10000K y su proyección sobre la recta perpendicular de dirección $\bar{e}^{\perp}$. Se aprecia cómo para una misma superficie con distintas iluminaciones se obtiene un mismo valor en el espacio invariante *I*, y cómo al cambiar de superficie el valor varía.

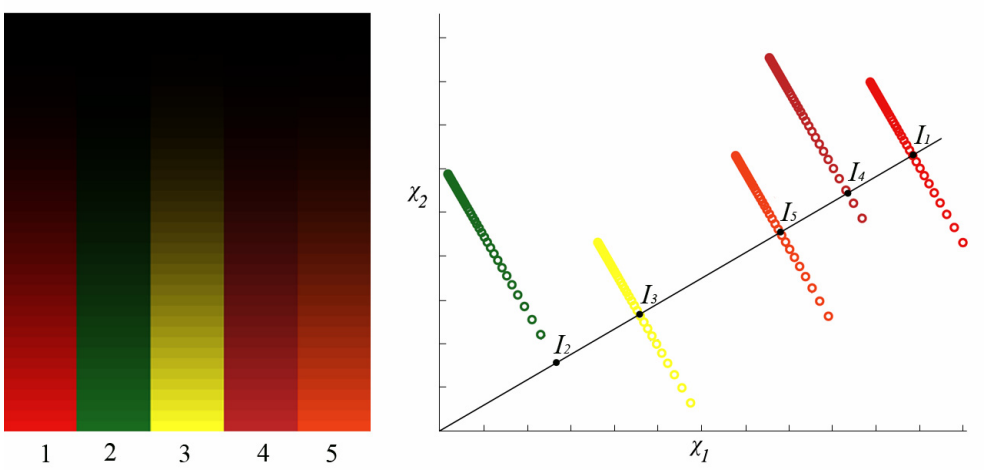


Figura 4.11 A la izquierda cinco superficies iluminadas con temperaturas entre 2500 y 10000K. A la derecha las rectas descritas por las superficies tras aplicar la transformación del espacio invariante y sus correspondientes proyecciones I_i sobre la recta perpendicular

Retomando ahora las ecuaciones 4.3 y 4.4 cabe destacar que el denominador para el cálculo de las coordenadas es un término a elegir entre los posibles canales del sensor, por ejemplo en dichas ecuaciones se ha elegido el tercer canal, ρ_3. Esta elección puede manifestarse, entre otros, en el nivel de ruido obtenido. Supóngase por ejemplo que el denominador escogido es el canal que valores más bajos toma, en este caso el ruido del numerador se hará más apreciable en el resultado final. Como solución se propone dividir los tres canales utilizando como denominador la media geométrica de los mismos:

$$\chi_1^3 = \log\left(\frac{\rho_1}{\rho_{den}}\right) \tag{4.6}$$

$$\chi_2^3 = \log\left(\frac{\rho_2}{\rho_{den}}\right) \tag{4.7}$$

$$\chi_3^3 = \log\left(\frac{\rho_2}{\rho_{den}}\right) \tag{4.8}$$

Donde:

$$\rho_{den} = \sqrt[3]{\rho_1 \rho_2 \rho_3} \tag{4.9}$$

Se obtienen por lo tanto tres coordenadas, $\left(\chi_1^3, \chi_2^3, \chi_3^3\right)$ que sumadas son:

$$\chi_1^3 + \chi_2^3 + \chi_3^3 = \log\left(\frac{\rho_1}{\sqrt[3]{\rho_1\rho_2\rho_3}}\right) + \log\left(\frac{\rho_2}{\sqrt[3]{\rho_1\rho_2\rho_3}}\right) + \log\left(\frac{\rho_3}{\sqrt[3]{\rho_1\rho_2\rho_3}}\right)$$

Aplicando que la suma de logaritmos es el logaritmo de la multiplicación.

$$\chi_1^3 + \chi_2^3 + \chi_3^3 = \log\left(\frac{\rho_1}{\sqrt[3]{\rho_1\rho_2\rho_3}} \cdot \frac{\rho_2}{\sqrt[3]{\rho_1\rho_2\rho_3}} \cdot \frac{\rho_3}{\sqrt[3]{\rho_1\rho_2\rho_3}}\right) = \log\left(\frac{\rho_1\rho_2\rho_3}{\left(\sqrt[3]{\rho_1\rho_2\rho_3}\right)^3}\right) = 0$$

Con lo que se demuestra que la suma de las tres coordenadas es nula.

$$\chi_1^3 + \chi_2^3 + \chi_3^3 = 0 \tag{4.10}$$

Esta ecuación describe un plano dado por el vector normal $v = (1,1,1)$ de tal manera que al modificar el valor de *T* se generan una serie de rectas, una para cada superficie, todas ellas ortogonales al plano como se representa en la figura 4.12.

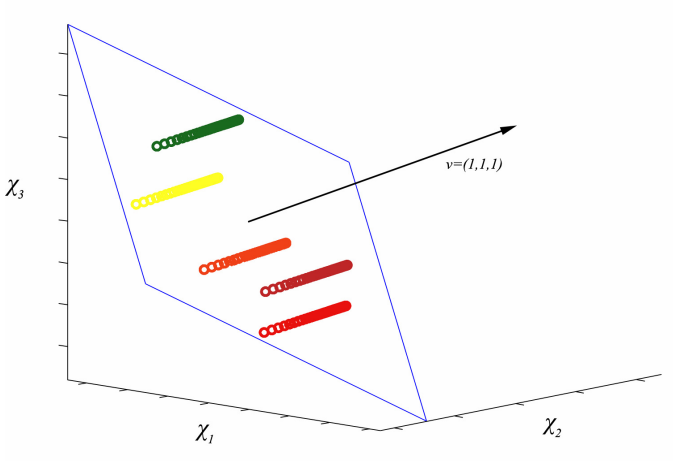


Figura 4.12 Rectas formadas al variar T, todas ellas ortogonales al plano

Esto quiere decir que la información que se puede obtener no va a dejar de ser bidimensional, y por lo tanto se puede aplicar el mismo concepto de proyección sobre la recta perpendicular como se hizo en la ecuación 4.5, generando así un espacio invariante a la iluminación.

4.5.1. Calibración del espacio invariante a la iluminación

Para poder usar adecuadamente el espacio invariante, en la práctica es necesario calibrar un parámetro. Éste es el ángulo del vector $\overline{e}^{\perp}$, es decir, el ángulo de la recta perpendicular a las descritas por las distintas iluminaciones de las superficies, véase figura 4.4.

Muchos son los métodos posibles de calibración. A continuación se expone un de ellos, el basado en mínimos cuadrados.

Partiendo de *N* imágenes de una misma escena iluminada a distintas temperaturas, figura 4.13.

Figura 4.13 Misma escena con distintas iluminaciones

Y dada una misma coordenada para las *N* imágenes, (u,v); el valor del píxel transformado al espacio invariante, $I^i(u,v)$, debe ser exactamente el mismo sea cual sea la captura, figura 4.14.

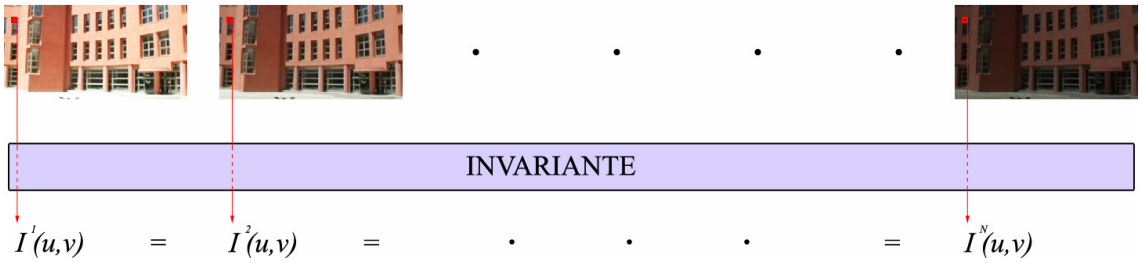


Figura 4.14 Transformación al espacio invariante del mismo píxel en todas las imágenes

Donde:

$$I^i(u,v) = \log\left(\frac{\rho^i_{1(u,v)}}{\rho^i_{3(u,v)}}\right)\cos(\theta) + \log\left(\frac{\rho^i_{2(u,v)}}{\rho^i_{3(u,v)}}\right)\sin(\theta)$$

Dado que todos los términos $I^i(u,v)$ son iguales, entonces es sencillo ver que la media de todos ellos es igual a cualquiera por separado, es decir, la media de los valores en el espacio invariante para un mismo píxel en *N* imágenes debe ser igual a cualquiera de sus valores individualmente, ecuación 4.11:

$$\frac{1}{N}\sum_{j=1}^{N} I^j(u,v) = I^i(u,v) \qquad i = 1..N \qquad (4.11)$$

De donde se despeja que:

$$\begin{pmatrix} \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\rho^{j}_{2(u,v)}}{\rho^{j}_{3(u,v)}}\right)-\log\left(\frac{\rho^{1}_{2(u,v)}}{\rho^{1}_{3(u,v)}}\right) & \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\rho^{j}_{2(u,v)}}{\rho^{j}_{3(u,v)}}\right)-\log\left(\frac{\rho^{1}_{2(u,v)}}{\rho^{1}_{3(u,v)}}\right) \\ \vdots & \vdots \\ \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\rho^{j}_{2(u,v)}}{\rho^{j}_{3(u,v)}}\right)-\log\left(\frac{\rho^{N}_{2(u,v)}}{\rho^{N}_{3(u,v)}}\right) & \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\rho^{j}_{2(u,v)}}{\rho^{j}_{3(u,v)}}\right)-\log\left(\frac{\rho^{N}_{2(u,v)}}{\rho^{N}_{3u,v}}\right) \end{pmatrix}_{Nx2} \begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix}_{2x1} = \begin{pmatrix}0\\ \vdots \\ 0\end{pmatrix}_{Nx1}$$

Se conoce a la matriz de dimensión *Nx2* como *A*. Ahora bien, si se utiliza la información de todos los píxeles contenidos en todas las imágenes se obtiene una matriz *A* de dimensión mucho mayor, ésta es ("*nº píxeles en u"* x "*nº píxeles en v"* x "*N"*) x 2. Por ejemplo, para una imagen de 640x480 dicha dimensión es: (640x480x*N*) x 2.

Una vez compuesta *A* por la máxima información se tiene la siguiente igualdad:

$$A\,\phi = 0 \tag{4.12}$$

Donde:

$$\phi = \begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix} \tag{4.13}.$$

Por efecto del ruido y no idealidad de las cámaras, la parte derecha de la ecuación 4.12 no es exactamente un array de ceros, sino un array de valores próximos a cero, ecuación 4.14.

$$A\,\phi = \begin{pmatrix}\varepsilon_1\\ \vdots \\ \varepsilon_M\end{pmatrix} = \bar{\varepsilon} \qquad ,\varepsilon_i \to 0 \tag{4.14}$$

Los términos ε_i dependen de las incógnitas buscadas, ϕ. En concreto, las incógnitas deben minimizar el sumatorio de los cuadrados de los valores ε_i, ecuación 4.15.

$$\mathrm{E} = \sum_i (\varepsilon_i)^2 = \bar{\varepsilon}^t \cdot \bar{\varepsilon} \tag{4.15}$$

Ahora bien, dado que $\bar{\varepsilon}^t \cdot \bar{\varepsilon} = \phi^t A^t A \phi$, entonces el proceso se centra en minimizar $\phi^t A^t A \phi$.

Conocido $\|\phi\| = 1$ y aplicando el método de Lagrange, se llega a la ecuación 4.16.

$$A^t A\phi = \lambda\phi \tag{4.16}$$

Donde λ es conocido como multiplicador de Lagrange.

Recordándose la definición de autovector, en la ecuación 4.16 λ corresponde al autovalor asociado al autovector ϕ de la matriz $A^{\mathrm{T}}A$. Es decir, las incógnitas buscadas son justamente el autovector de la matriz $A^{\mathrm{T}}A$.

Puesto que la matriz $A^{\mathrm{T}}A$ es de dimensión 2x2, entonces existen dos autovectores: ϕ_1, ϕ_2, así como sus dos autovalores asociados: λ_1 ,λ_2 respectivamente. Las incógnitas buscadas son el autovector asociado al menor autovalor.

En la figura 4.15 se resume el proceso de calibración.

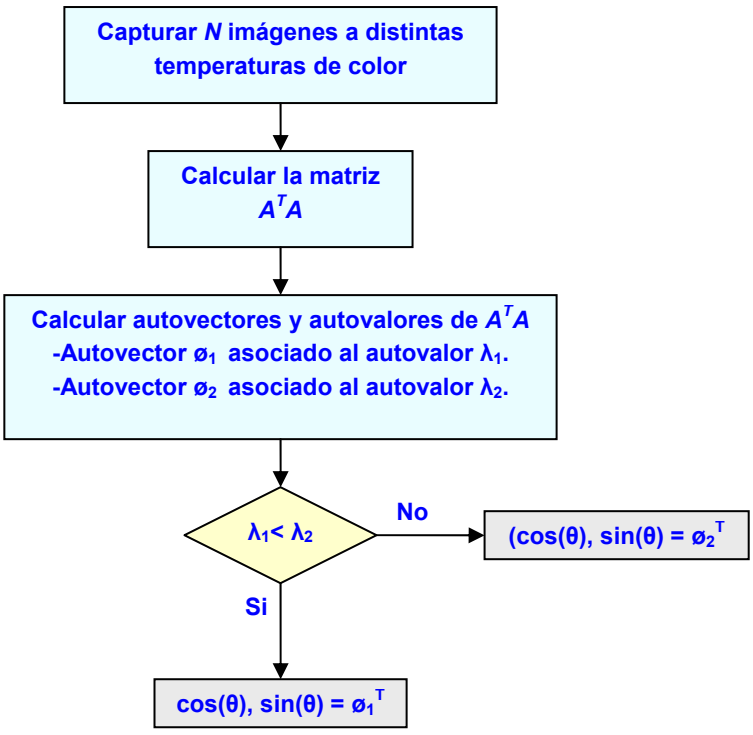


Figura 4.15 Proceso de calibración del espacio invariante a la iluminación

CAPÍTULO 5. SEGMENTACIÓN DE FONDO

5.1. Introducción

El proceso de segmentación es un paso clave en el análisis de la imagen, de éste depende en gran medida el éxito de los resultados posteriores.

Dicho proceso consiste en segmentar o dividir una imagen en los objetos que la forman, siendo los objetos buscados y el nivel de detalle algo que depende de la aplicación.

A lo largo del presente proyecto se habla de segmentación de fondo. En ésta se segmentan los objetos sobre un fondo del propio fondo, figura 5.1.

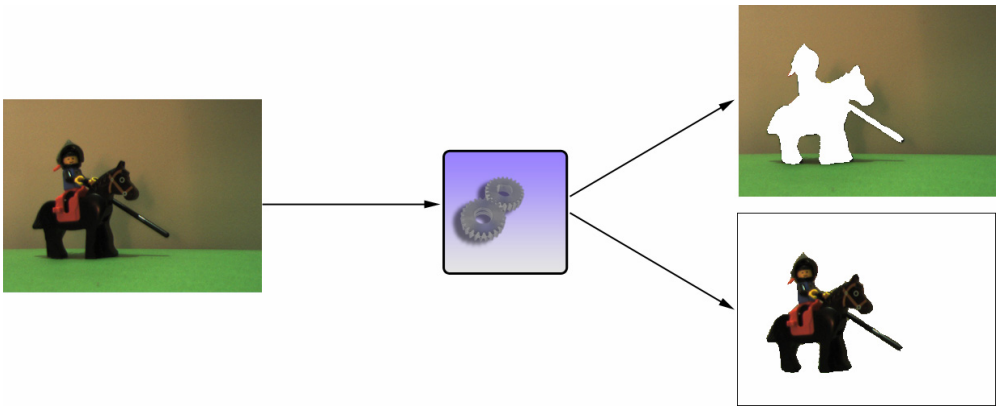


Figura 5.1 A la izquierda el fondo con objetos superpuestos. A la derecha objetos y fondo separados.

5.2. Métodos de segmentación

Básicamente se pueden distinguir tres métodos de segmentación [Jähne, 1995]: métodos basados en píxel, métodos basados en regiones y métodos basados en bordes.

El método basado en píxel segmenta la imagen únicamente con la información del valor de cada punto individualmente, es decir, este método decide si un píxel pertenece o no a un objeto operando con su valor.

El método basado en regiones centra la segmentación en una característica importante de los objetos, su conectividad. Así, mediante alguna técnica que juzgue la conectividad de los píxeles, se segmenta la imagen.

Por último, el método basado en bordes se basa en la continuidad de los píxeles en un objeto. De este modo se pasa por buscar los bordes en la imagen y se caracterizan como los límites de los objetos.

5.3. Problemática de la segmentación de fondo

A priori el proceso de segmentación de fondo puede resultar trivial ya que se asocia a la facilidad con la que el ser humano lo realiza, sólo hay que memorizar una imagen (fondo) y distinguir los objetos que pasen sobre ella. No obstante se presentan ciertos problemas que no siempre se resuelven con facilidad.

Para empezar, supóngase una misma cámara que se encuentre enfocando a una imagen fija en dos instantes de tiempo distintos; idealmente el valor de un píxel en las dos imágenes resultantes debería ser idéntico, sin embargo esto no es así. Existe un ruido en la imagen que hace oscilar el valor de cada píxel sobre su valor real. Este ruido es en la mayoría de los casos modelado como gaussiano y por lo tanto el valor de un píxel en la imagen vendrá dado por la función de densidad de probabilidad siguiente:

$$f(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-m)^2}{2\sigma^2}} \tag{5.1}$$

Siendo m el valor medio y σ la desviación típica.

En la figura 5.2 se ve cómo un mismo píxel varía su valor en las componentes RGB debido al ruido mencionado.

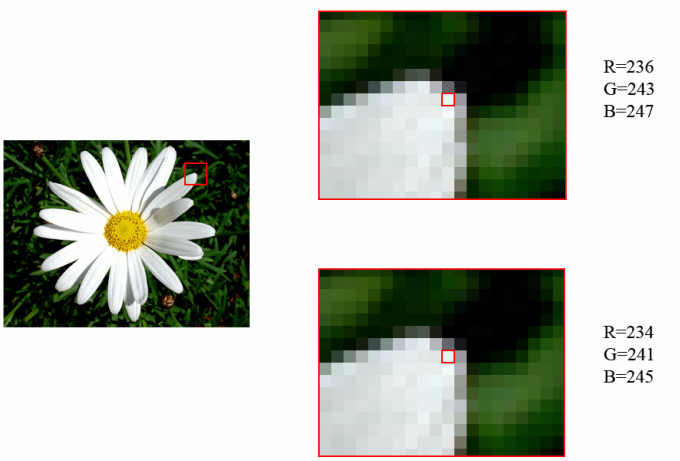


Figura 5.2 Ruido en la imagen

Por otro lado, en la imagen pueden estar presentes sombras que queriendo o no van a suponer un cambio sobre el fondo, siendo por lo tanto detectadas como nuevos objetos sin serlo.

No obstante, el problema de las sombras se engloba dentro del problema del cambio de iluminación. Una sombra no deja de ser un cambio de iluminación sobre una superficie debido a un obstáculo. De este modo, una variación de la iluminación, como por ejemplo el mal funcionamiento de un foco en un entorno interior o el paso de unas horas en un entorno exterior, puede suponer que la imagen completa cambie sin haber ningún objeto presente.

En la figura 5.3 se muestra un ejemplo del error cometido debido únicamente a la sombra producida por un cono. Siendo el color blanco el correspondiente a los objetos se aprecia cómo la sombra se ha detectado sin serlo.

Figura 5.3 Sombras en la imagen. A la izquierda la imagen de referencia. En el centro se añade un objeto. A la derecha el resultado de la segmentación

Del mismo modo que se detectan las sombras como objetos se van a detectar las reflexiones. Por ejemplo, si se trabaja en un entorno con espejos o sobre una superficie pulida o mojada se va a tener este problema. Se muestra en la figura 5.4 el resultado al trabajar con una superficie mojada y en la 5.5 por una superficie pulida.

Figura 5.4 Segmentación de fondo con reflexiones debido a superficie mojada

Figura 5.5 Segmentación de fondo con reflexiones debido a superficie pulida

5.4. Segmentación de fondo implementada

La segmentación de fondo es un proceso crucial en el correcto funcionamiento de todo el sistema. Es por ello que se realizan numerosas pruebas hasta conseguir el mejor de los resultados.

Como se ha visto, en el proceso se presentan importantes problemas. Algunos de éstos son los cambios de iluminación, las sombras y el ruido. A continuación se explica el camino seguido para la minimización de dichos inconvenientes, desde el inicio hasta el resultado final.

5.4.1. Elección del espacio de color

Escala de grises

Las primeras pruebas se realizan a partir de una imagen fija libre de objetos a la que se resta la captura en el instante actual. De esta forma sólo los objetos superpuestos sobre el fondo dan como resultado un valor relativamente grande respecto a las zonas donde no los hay. Para simplificar el proceso, esta resta se hace con un solo canal, el correspondiente a la escala de grises. A continuación se aplica un umbralizado donde se acentúan las diferencias y se atenúan los valores bajos, figura 5.6.

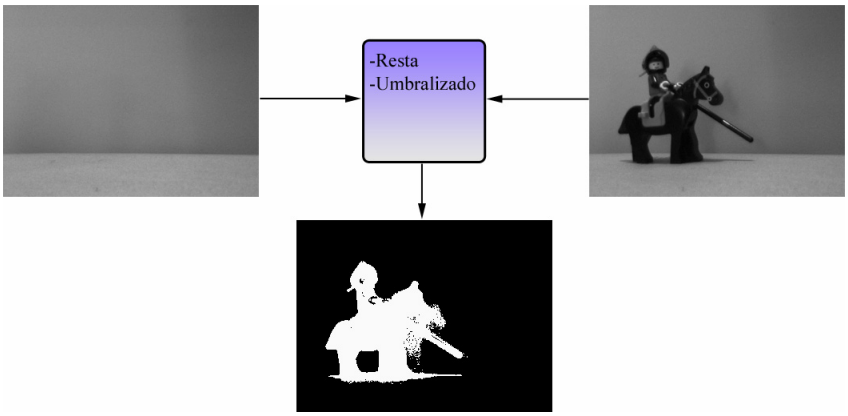


Figura 5.6 Segmentación de fondo en escala de grises

Con este método, el único parámetro que se puede modificar es el valor del umbral. Con un valor bajo la imagen se contamina de ruido y sombras, y con un valor alto desaparecen objetos.

El umbral óptimo es función de la aplicación, donde se decide la importancia de las sombras, ruido y detección de objetos entre otras.

En el caso que ocupa este proyecto es necesaria la buena detección de objetos y discriminación de sombras. Ambos requisitos son incompatibles ya que para el primero es necesario un umbral bajo y para el segundo un umbral alto.

Por otro lado, supóngase que existe un cambio de iluminación del fondo. Esto resulta catastrófico, ya que el propio fondo es detectado como objeto, figura 5.7.

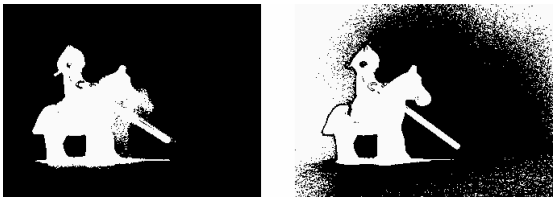


Figura 5.7 Cambio de iluminación del fondo

YUV

En el espacio de color YUV la escala de grises se encuentra proyectada directamente sobre uno de los ejes, el Y, representando la luminancia de la imagen. Suponiendo que las sombras signifiquen un aumento o disminución del valor de este eje, entonces con no tenerlo en cuenta debe ser suficiente para deshacerse de ellas.

El método implementado, al igual que el anterior, toma un fondo y lo resta a la imagen actual. En esta resta, ahora se utilizan dos de los tres canales disponibles, el U y el V. Éstos se ponderan con iguales pesos y se suman. Ambos canales contienen información de color pero no de luminancia, o lo que es lo mismo si las premisas son correctas, están libres de sombras.

Sin embargo en los resultados obtenidos, figura 5.8, las sombras siguen presentes.

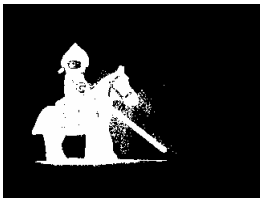


Figura 5.8 Resultado de la segmentación de fondo ignorando el canal Y

Al igual, un cambio de iluminación resulta fatal, figura 5.9.

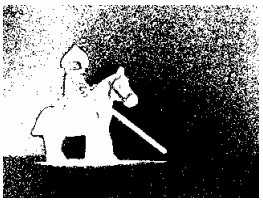


Figura 5.9 Resultado de la segmentación de fondo en YUV ante un cambio de iluminación

Si se representan los canales por separado se ve cómo efectivamente tanto el canal U como el V están afectados por las sombras, figura 5.10.

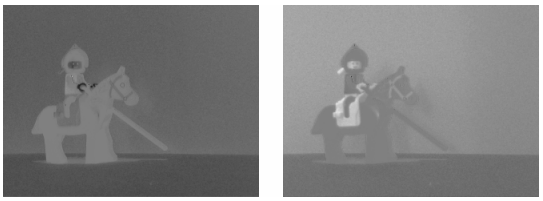


Figura 5.10 Canales U (izquierda) y V (derecha) de la imagen a segmentar. Ambos afectados por las sombras

YUV invariante

Una mejora pasa por normalizar las coordenadas U, V mediante Y.

$$U' = \frac{U}{Y} \qquad V' = \frac{V}{Y} \tag{5.2}$$

En éste nuevo espacio de color, una superficie dada a distintas iluminaciones se debe representar en la misma coordenada.

Aplicándolo a los ejemplos anteriores se obtiene una considerable mejora, tanto en las sombras como en el cambio de iluminación, figura 5.11.

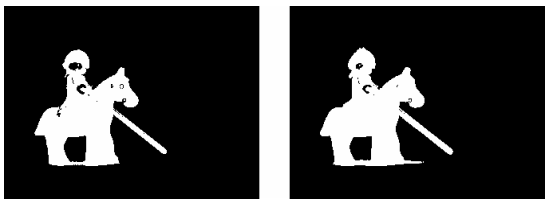


Figura 5.11 A la izquierda segmentación usando YUV normalizado. A la derecha con un cambio de iluminación

No obstante, las sombras más fuertes siguen existiendo. Por ejemplo en la figura 5.12 se muestra lo que podría ser una mesa.

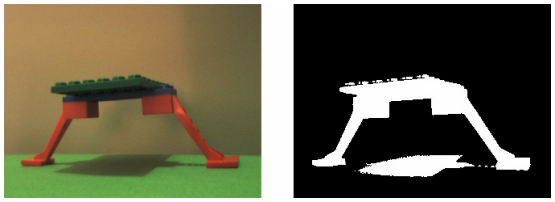


Figura 5.12 Segmentación de fondo con sombras fuertes usando YUV normalizado. Imagen a segmentar (izquierda) y resultado (derecha)

HSV

El espacio HSV está compuesto por las componentes de matiz, saturación e intensidad.

El matiz es función de la frecuencia dominante en el color sin importar saturación o intensidad. Por ello cabe pensar que considerando únicamente esta componente las sombras

deben desaparecer. Sin embargo, esto no ocurre. La iluminación de los objetos se realiza con fuentes coloreadas, lo cual implica inevitablemente el desplazamiento del matiz.

En la figura 5.13 se muestra el resultado de la segmentación de fondo.

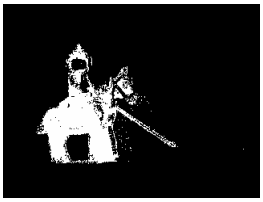


Figura 5.13 Segmentación de fondo usando HSV

La mayoría de las sombras han desaparecido, no obstante, ha sido como respuesta a elevar el umbral. Aun así, las sombras más fuertes siguen presentes.

Espacio invariante a la iluminación

Por último, se realiza la segmentación de fondo empleando el espacio invariante a la iluminación.

Éste consigue que una superficie iluminada a distintas temperaturas sea interpretada como una misma coordenada, resolviendo así los problemas de sombras y cambios de iluminación. Un ejemplo se muestra en la figura 5.14, donde se presenta un resultado simple (izquierda) y un resultado ante un cambio de iluminación (derecha).

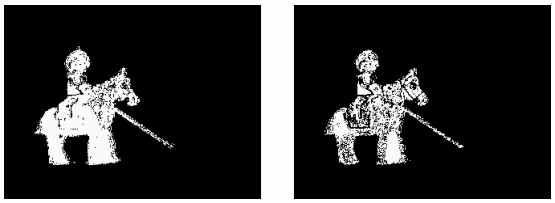


Figura 5.14 A la izquierda segmentación de fondo usando espacio invariante a la iluminación. A la derecha con un cambio de iluminación

Todas las sombras han desaparecido y el cambio de iluminación no ha supuesto variaciones significativas en la segmentación. En la figura 5.15 se muestra el ejemplo para sombras fuertes.

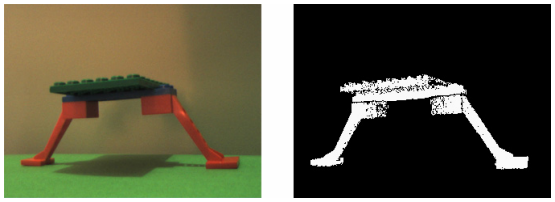


Figura 5.15 Segmentación de fondo con sombras fuertes usando espacio invariante a la iluminación. Imagen a segmentar (izquierda) y resultado (derecha)

Queda demostrada la efectividad del espacio ante los problemas de iluminación, ya sea por efecto de las sombras o por variaciones en las fuentes de luz.

5.4.2. Algoritmo de decisión en la segmentación de fondo

Para discernir los objetos del fondo hasta ahora se ha utilizado una resta entre imágenes, donde los obstáculos corresponden con los valores elevados en el resultado. Este método es muy sencillo de implementar y en algunos casos aporta buenos resultados. No obstante, ya se dijo que la segmentación de fondo es un paso decisivo para el posterior éxito del sistema y por lo tanto se siguen estudiando mejoras.

En este caso, cada canal de cada píxel es tratado como una señal independiente, la cual se encuentra contaminada por un ruido que se considera gaussiano. Un píxel en la imagen de fondo se encuentra caracterizado por su media y su varianza, figura 5.16.

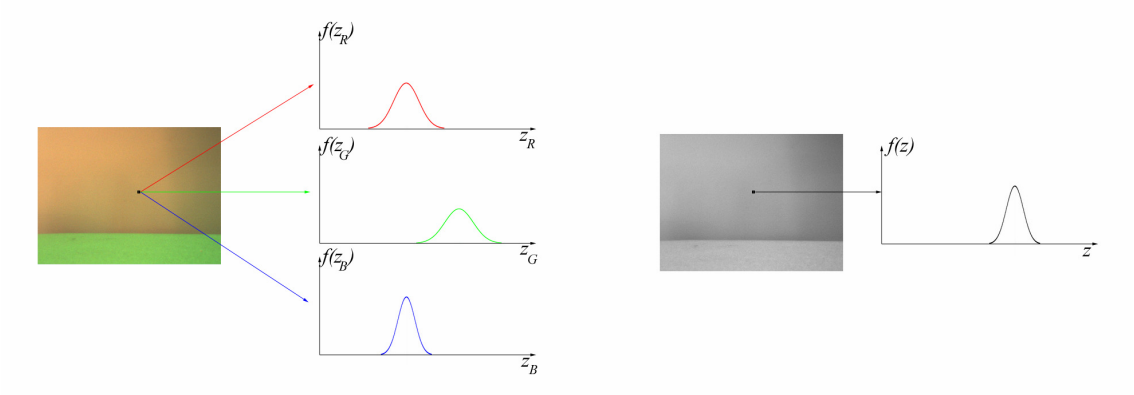


Figura 5.16 Función de densidad de probabilidad para cada canal de un píxel. A la izquierda imagen en RGB. A la derecha imagen en escala de grises

Puesto que se utiliza el espacio invariante a la iluminación, se va a trabajar con un solo canal. Es decir, para caracterizar una imagen son necesarias tantas funciones de densidad de probabilidad como píxeles.

Cualquier objeto situado sobre el fondo va a generar un cambio sobre en él. Estos nuevos píxeles se van a caracterizar nuevamente por una distribución normal, donde la media va a depender, entre otros, del color del objeto superpuesto, figura 5.17.

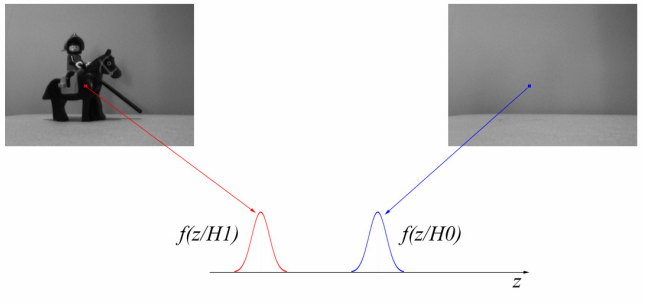


Figura 5.17 Función densidad de probabilidad para el mismo píxel con y sin objeto

Conocidas *H0*, *H1* como "*hipótesis nula*" e "*hipótesis alternativa*" respectivamente, las funciones *f(z/H1)* y *f(z/H0)* corresponden a las funciones de densidad de probabilidad en el caso de que exista o no un objeto presente.

Dichas funciones se encuentran centradas en distintos puntos. Tanto es así que a priori puede parecer que la detección de objetos es trivial; sólo hay que fijar un umbral, γ, que delimite las zonas de decisión, *D0* y *D1*, figura 5.18.

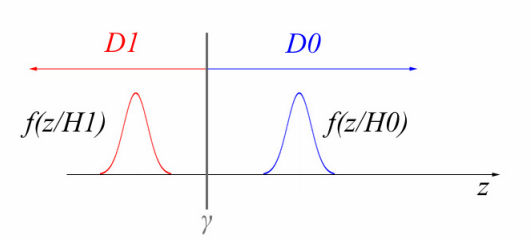


Figura 5.18 Zonas de decisión de las hipótesis

Sin embargo, *f(z/H1)* es función del tipo de objeto presente, desplazándose según su color, figura 5.19.

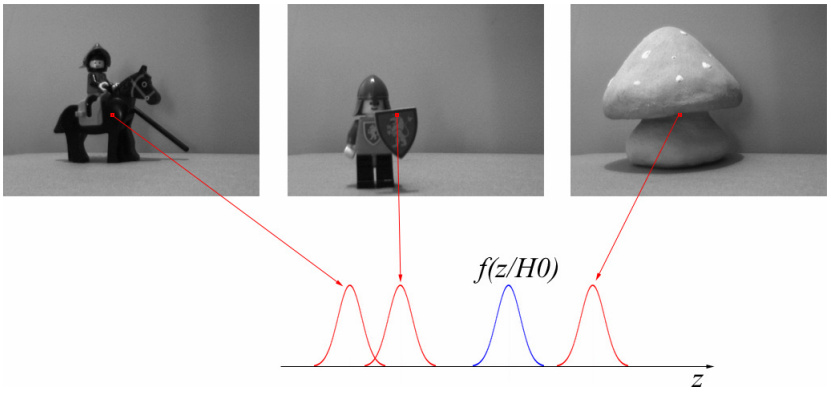


Figura 5.19 desplazamiento de f(z/H1) en función del color del objeto

Esto hace que la detección no sea tan sencilla y obliga a reducir la zona de decisión de *H0* todo lo posible.

El tamaño de *D0* es fijado según la probabilidad de falsa alarma aceptada. Entendiendo como probabilidad de falsa alarma la probabilidad de decidir *H1* cuando está ocurriendo *H0*, es decir, decidir que hay objeto cuando en realidad no lo hay.

Si *D0* se reduce mucho, entonces la probabilidad de falsa alarma crece pero a cambio muchos más objetos son detectados. Por otro lado, si *D0* aumenta, la probabilidad de falsa alarma disminuye pero también la probabilidad de detección.

Si se escoge *D0* entre ±σ sobre la media de *f(z/H0)*, entonces la probabilidad de falsa alarma ronda el 32%. Si el valor fuera de ±2σ la nueva probabilidad es aproximadamente del 5%. Quizá la mejor forma de seleccionar el tamaño de *D0* sea a partir de los resultados reales. En la figura 5.20 se muestra el ruido presente en la imagen en función del tamaño de la región de decisión.

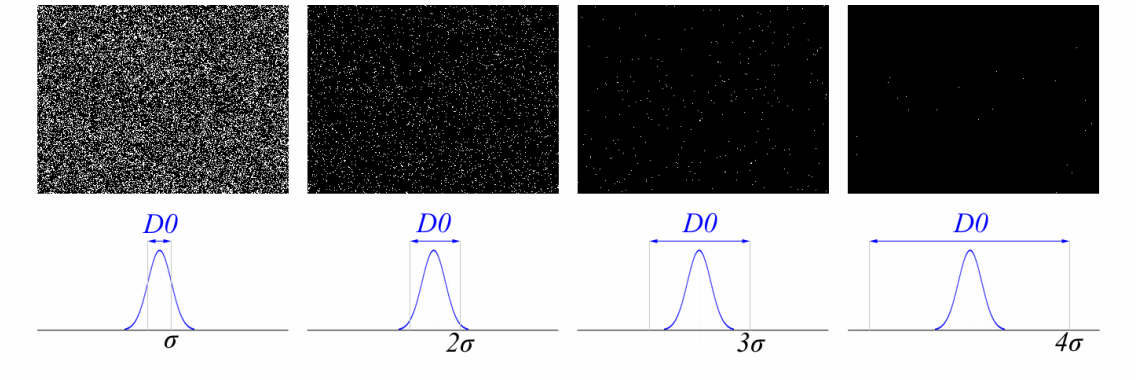


Figura 5.20 Ruido en la imagen en función del tamaño de D0

Según la tolerancia de la aplicación se fija uno u otro tamaño de *D0*.

Por último, la imagen se somete a un proceso de eliminación del ruido. Entendiendo por ruido todos aquellos píxeles que han sido detectados como objetos sin serlo así como todos aquellos píxeles que no han sido detectados y sí lo son. Se aplican dos pasos, figura 5.21.

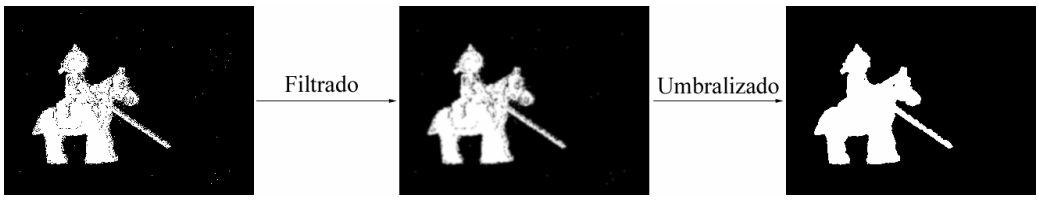


Figura 5.21 Filtrado y umbralizado en la segmentación de fondo

- Filtrado gaussiano de la imagen. Se suavizan todos aquellos píxeles que se encuentren aislados, sean blancos o negros.

- Umbralizado de la imagen. Los píxeles aislados y filtrados desaparecen y pasan a formar parte del conjunto que les rodea.

Resultados reales de la segmentación aplicada son mostrados en la figura 5.22, donde se ve la buena respuesta ante el ruido y sombras.

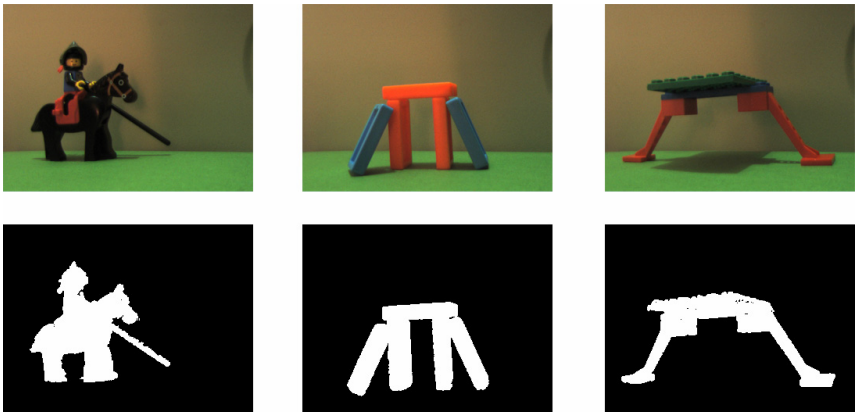


Figura 5.22 Resultados de la segmentación de fondo usada

CAPÍTULO 6. DETECCIÓN DE OBJETOS

6.1. Introducción

El ruido es un fenómeno presente en cualquier imagen obtenida de cualquier cámara, de forma que los datos que se desprenden de ésta están a su vez contaminados.

Esto hace que los procesos de detección de objetos tengan que basarse en unos datos de fiabilidad desconocida, pudiéndose sólo hablar de probabilidades. Así, la detección de objetos se convierte en un proceso de estimación que requiere el uso de algoritmos probabilísticos como el filtro de Partículas.

Varios algoritmos han sido probados en [Cabello ,2007], obteniendo los mejores resultados con: el XPFCP, "Filtro de Partículas Extendido con proceso de Clasificación", un algoritmo basado en un filtro de Partículas como estimador y un clasificador como proceso de asociación.

A continuación se explica el algoritmo.

6.2. Algoritmo "XPFCP"

El algoritmo "Filtro de Partículas Extendido con proceso de Clasificación" consta de dos partes diferenciadas. En primer lugar se encuentra el filtro de partículas extendido como estimador, y en segundo lugar un clasificador como proceso de asociación.

6.2.1. Filtro de Partículas Extendido

Se va a comenzar la explicación del filtro de partículas con las definiciones y notaciones que van a ser usadas durante el algoritmo.

- Se llaman medidas a aquellos datos obtenidos directamente de la imagen. Se denotan con el vector $\vec{y}$.

- Se llaman partículas al par $\left(\vec{x}_t^{(i)}, \widetilde{w}_t^{(i)}\right)$, siendo $\vec{x}_t^{(i)}$ el vector de estado de la partícula i-ésima en el instante de tiempo *t*, y $\widetilde{w}_t^{(i)}$ el peso asociado a la partícula i-ésima en el instante *t*.

- Se denota como $p(\vec{x}_t|\vec{y}_{1:t})$ la función de probabilidad de que el objeto bajo estudio se encuentre en el estado $\vec{x}_t$ a tenor de las medidas recogidas desde el instante de tiempo *1* hasta *t*.

La finalidad del filtro de partículas es actualizar la función de probabilidad a posteriori, $p(\vec{x}_t|\vec{y}_{1:t})$, y estimar el estado $\vec{x}_{t+1|t}$ a partir de un conjunto de *n* partículas $S_{t-1} = \left\{\vec{x}_{t-1}^{(i)}, \widetilde{w}_{t-1}^{(i)}\right\}_{i=1}^{n}$. Los tres pasos que sigue para ello corresponden al siguiente bucle:

1. **Predicción:** En este paso se parte de partículas conocidas en el instante *t-1*. Éstas se aplican en el modelo del sistema bajo estudio dando como resultado las partículas en el instante *t*, es decir, se plantea encontrar $S_{t|t-1} = \left\{\vec{x}_{t|t-1}^{(i)}, \widetilde{w}_{t-1}^{(i)}\right\}_{i=1}^{n}$. Para ello se usa el modelo de actuación del sistema (en su versión probabilística caracterizado por $p(\vec{x}_t|\vec{x}_{t-1})$), de modo que $\vec{x}_{t|t-1}^{(i)} = p(\vec{x}_t|\vec{x}_{t-1}) \cdot \vec{x}_{t-1}^{(i)}$.

2. **Corrección**: En el paso de corrección se calcula el peso de cada partícula $\left\{w_t^{(i)}\right\}_{i=1}^{n}$, comparando el vector $\vec{y}_t$ y su valor predicho basado en la estimación $\vec{x}_{t|t-1}$. Intuitivamente cada $\left\{w_t^{(i)}\right\}_{i=1}^{n}$ se asocia al grado de acierto en la estimación representada por la partícula correspondiente $\left\{\vec{x}_{t|t-1}^{(i)}\right\}_{i=1}^{n}$.

3. **Selección**: A partir de los pesos $\left\{w_t^{(i)}\right\}_{i=1}^{n}$ se genera un subgrupo de partículas, $S_t = \left\{\vec{x}_t^{(i)}, \widetilde{w}_t^{(i)}\right\}_{i=1}^{n}$, donde únicamente tienen cabida las más probables, siendo el

resto descartadas. Es este subgrupo el que se usa nuevamente en el paso de predicción en *t+1* cerrando así el bucle, y es también este subgrupo el que representa la salida final del filtro $p(\vec{x}_t|\vec{y}_{1:t})$.

El proceso anterior requiere en su primera iteración el valor del conjunto de partículas en el instante *t-1*. Es por ello que el filtro se inicia, en general, de modo que S_0 proceda del valor de las medidas en el instante *0,* $Y_0 = \{\vec{y}_0^{(j)}\}_{j=1}^{m_0}$.

Ahora bien, el algoritmo hasta ahora presentado es solamente capaz de estimar el estado de un único objeto. Para conseguir que dicha estimación sea múltiple se debe recurrir a las modificaciones propuestas en [Marrón et al, 2007], en las que se introduce un nuevo paso de reinicialización y un nuevo paso de corrección.

0. **Reinicialización:** En este paso se introduce un conjunto de n_m nuevas partículas asociado al set de datos $Y_{t-1} = \{\vec{y}_{t-1}^{(j)}\}_{j=1}^{m_{t-1}}$ obtenido del proceso de observación en el instante anterior *t-1*, de modo semejante al proceso de inicialización del filtro descrito en párrafos anteriores. De este modo se asegura que todas las hipótesis que acaban de ser sensadas tienen representación al principio del bucle de ejecución del filtro. Con objeto de que el número de partículas no se incremente a lo largo del tiempo debido a este nuevo paso, se modifica además la etapa de selección, antes descrita haciendo que genere sólo $n - n_m$ partículas a su salida.

En [Marrón, 2008] se demuestra que el proceso de reinicialización resulta mucho más efectivo si se clasifican el conjunto de medidas u observaciones $Y_t \Rightarrow G_{1:k,t}$ y se usan estas k_t clases en esta etapa de reinicialización. Además, el conjunto de medidas clasificadas se usa también en la etapa de corrección favoreciendo el proceso de asociación imprescindible en la tarea de estimación múltiple [Bar-Shalom and Fortmann, 1988].

6.2.2. El proceso clasificador

Para poder generar una salida determinística del sistema de seguimiento de múltiples objetos basado en un filtro de partículas es necesario, a parte de las modificaciones propuestas en [Marrón et al, 2007], un proceso de clasificación de salida. Su objetivo es agrupar las partículas que especifican la posición de cada objeto para generar así un conjunto de clases de salida equivalente al número de objetos seguidos.

El proceso usado con este objetivo en el algoritmo "XPFCP" [Cabello ,2007] es una versión adaptada del K-Medias para un número variable de objetos. Este clasificador es semejante al que se emplea como proceso de asociación en el paso de corrección del filtro [Marrón, 2008].

Las etapas en las que se divide el proceso son:

1. Se eligen k datos del total que son utilizados como valor inicial de los centroides de las clases a crear.

2. Cada dato es asociado a la clase cuyo centroide esté más próximo. Entendiendo como proximidad la distancia euclídea entre dicho dato y dicho centroide.
3. Realizada la asignación del total de datos en clases, se recalcula el valor de todos los centroides, siendo éste la media geométrica del conjunto de datos asignados a la clase a la que pertenece.
4. Si algún centroide se ha modificado en el tercer paso, entonces el algoritmo se repite desde el punto dos.
5. En caso contrario, o si el número de veces que se ha ejecutado el proceso excede a un limite predefinido, el algoritmo elimina las clases sin miembros asociados y finaliza.

Una serie de mejoras son propuestas en [Marrón, 2008] sobre la versión básica de este clasificador de salida. Por un lado, un proceso de validación de los datos/partículas consigue que las la información esporádica no sean procesada, dando así una mayor robustez al sistema ante el ruido. Por otro, se incluye un proceso de estimación inicial de los centroides con el fin de acortar el tiempo de cómputo del algoritmo.

La incorporación de este proceso de clasificación final a la salida del filtro de partículas da lugar al XPFCP, cuyo esquema final, extraído de [Marrón, 2008], se muestra en la figura 6.1.

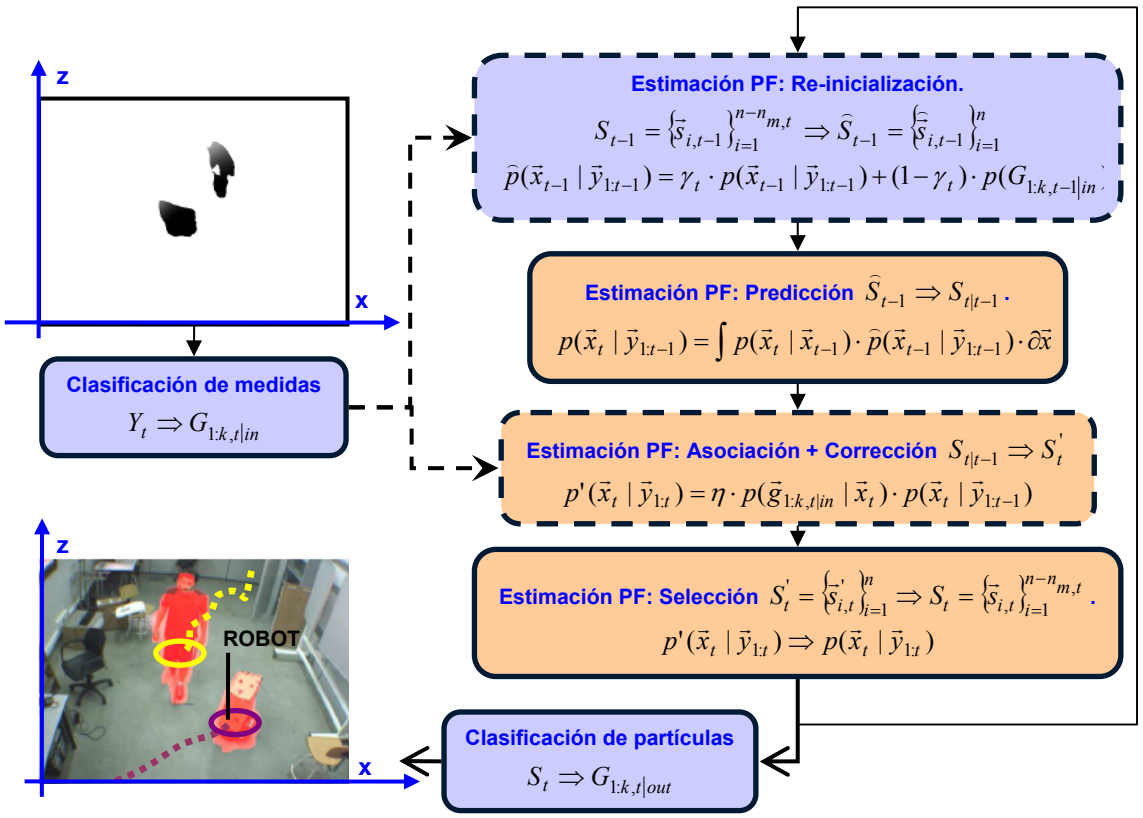


Figura 6.1 Esquema XPFCP

CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA EN TIEMPO REAL

7.1. Introducción

Basándose en la teoría vista hasta el momento se ha implementado un sistema capaz de detectar obstáculos en tiempo real utilizando como sensores únicamente las cámaras.

La metodología seguida pasa por utilizar una arquitectura cliente-servidor, donde cada cámara es tratada como un servidor que sirve imágenes al cliente, siendo este último el encargado de tratar y mostrar la información. Los obstáculos son presentados en un grid de ocupación de dimensiones y resolución prefijada.

Por otro lado, se ha implementado un sistema de manejo de robots sencillo y fiable, visualizando la trayectoria descrita a partir de los datos de odometría. Este sistema está integrado en el descrito anteriormente de manera que la recepción de datos por parte de los servidores y por parte del robot queda perfectamente sincronizada y mostrada a la par en tiempo real.

7.2. Arquitectura Cliente-Servidor y conexiones

Para el intercambio de datos entre los distintos sistemas se ha optado por la arquitectura cliente-servidor, donde es el cliente el que se conecta a los servidores y hace peticiones a éstos, figura 7.1.

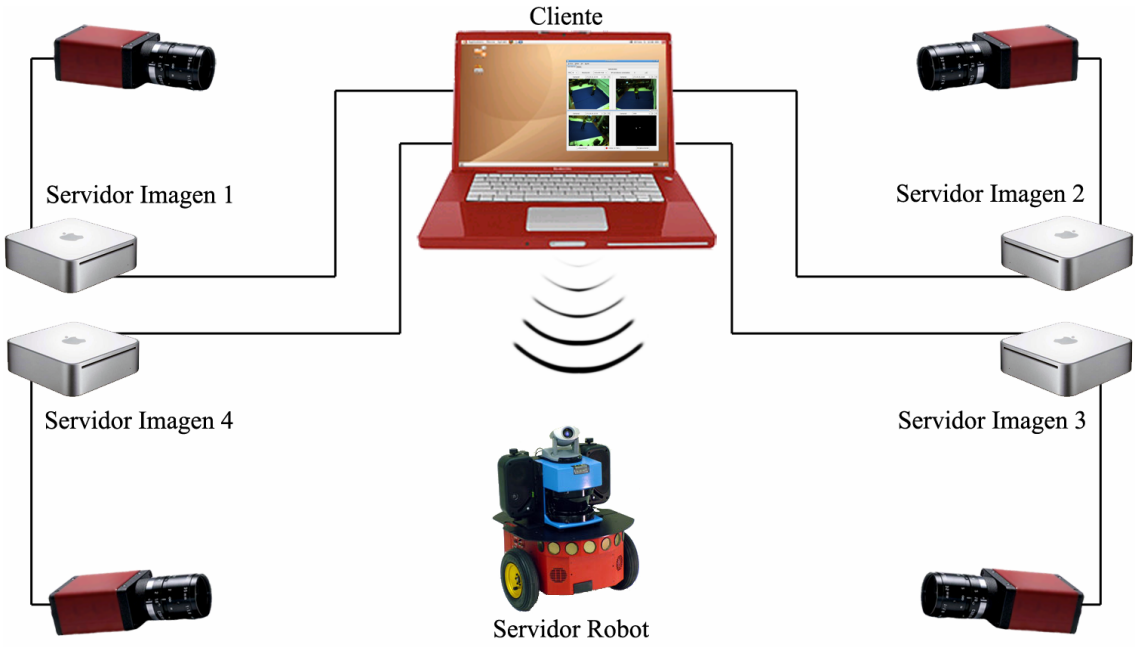


Figura 7.1 Estructura cliente-servidor

7.2.1. Servidores de imágenes

Las conexiones entre cliente y servidores son dependientes del estado en el que se encuentre el cliente, figura 7.2.

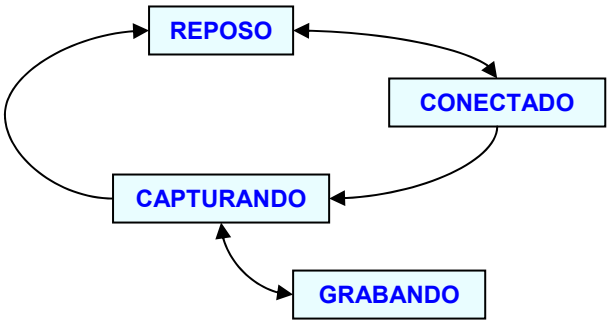


Figura 7.2 Diagrama de estados en el cliente referente a los servidores

En el estado *reposo* no hay ninguna conexión activa. El cliente se encuentra parado y los servidores están a la espera de conexiones entrantes.

Al estado *conectado* se conmuta cuando el cliente decide conectarse. Se recorre entonces un array con los servidores abriendo conexiones con todos ellos. En concreto se abren tres sockets por servidor, éstos son los responsables de transmitir información de datos (__sock_datos), de comandos (__sock_comandos) y el grid de cada servidor (__sock_grid).

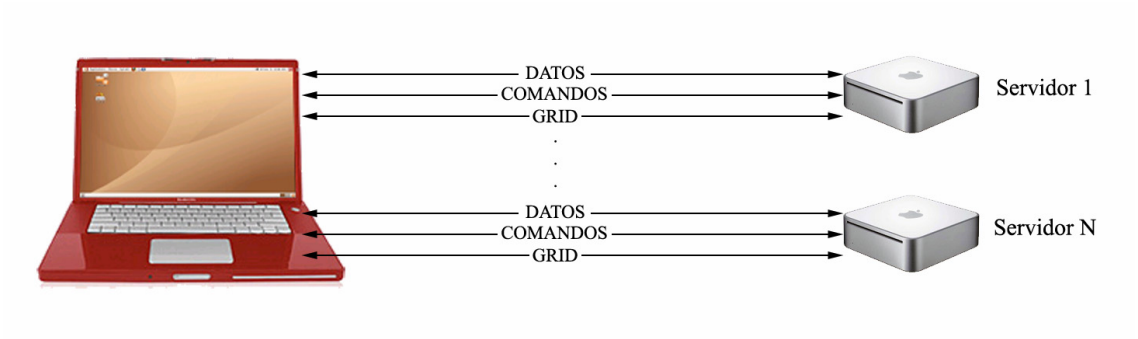


Figura 7.3 Sockets abiertos en el estado conectado

No existe un número prefijado de servidores a los que el cliente se puede conectar, y tampoco se conoce a priori si un servidor está listo para aceptar conexiones cuando el cliente lo solicita. Por ello se emplea un array con los servidores a los que se desea conectar y, en caso de que alguno no esté listo para la conexión, se elimina.

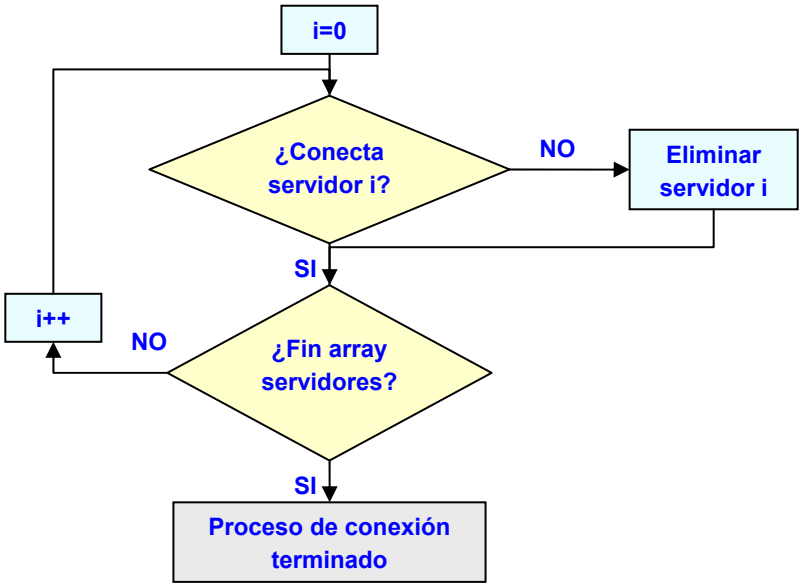


Figura 7.4 Proceso de conexión con los servidores

Una vez el cliente se ha conectado se pasa automáticamente al estado *capturando*. En éste se hacen peticiones, sincronizadamente, a los servidores para que generen el grid de ocupación y lo manden.

La única diferencia entre *capturando* y *grabando* son las peticiones enviadas a los servidores, donde en el estado *grabando* se añade la orden de guardar las imágenes al disco.

Por otro lado, fuera del diagrama de estados presentado en la figura 7.2, existen una serie de sockets no orientados a conexión, UDP, cuya finalidad es recibir las imágenes capturadas por las cámaras en los servidores. De esta forma queda separada la recepción de imágenes del proceso descrito anteriormente, figura 7.5, pudiendo así otorgar distintas prioridades a uno u otro.

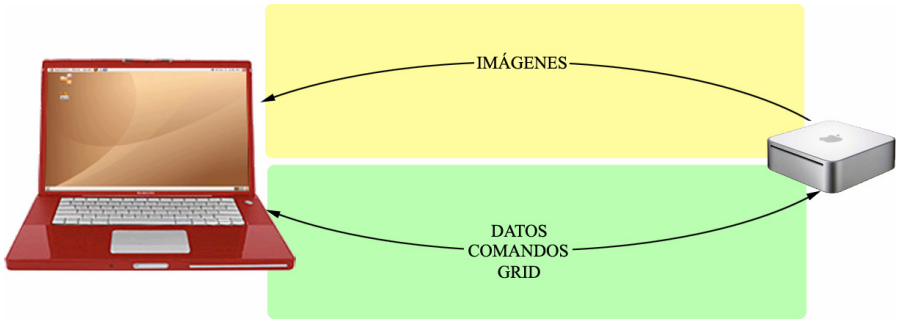


Figura 7.5 Dos procesos separados con los servidores de imágenes

7.2.2. Servidor robot

Se distinguen dos estados referentes al servidor robot: *conectado* y *no conectado*. En *no conectado* no existe conexión alguna entre cliente y servidor, siendo en el estado *conectado* donde se genera un socket TCP a través del cual el robot recibe peticiones y responde.

Dos son las peticiones que se hacen al robot. Por un lado se impone la velocidad lineal y angular que éste debe tener en cada momento. Por otro se pide la odometría, necesitando para ello una petición y una respuesta.

La petición de odometría, al igual que la respuesta, son mensajes que deben estar circulando a la mayor frecuencia posible. Sin embargo la petición asociada al cambio de velocidad se genera sólo en los momentos en el que ésta se modifica. Esto hace que sean procesos distintos los encargados de controlar una u otra orden. Ahora bien, todo esto fluye a través de un mismo socket surgiendo el consiguiente problema de colisiones entre paquetes si por ejemplo coincide una petición con una respuesta.

Es por lo tanto necesario un sistema de control de uso del socket para comunicarse con el robot. Control que se verá en la descripción del cliente.

7.3. Cliente

El cliente es la parte del sistema que mayor número de tareas controla. A continuación se exponen una a una las más importantes:

7.3.1. Servidores de imágenes

Hasta aquí se ha visto lo referente a las conexiones entre servidores de imágenes y cliente. Es momento ahora de explicar su gestión.

Gestión de los servidores

El cliente trata los servidores como una lista enlazada de estructuras, donde cada estructura corresponde a un servidor distinto. En principio, y gracias a utilizar este método, el número de servidores puede crecer indefinidamente, así como también se pueden borrar, añadir, guardar a disco, etc.

No se va a conectar a todos los servidores presentes en la lista. Cada estructura tiene asociado un campo, *activo*, que permite o no la conexión. De este modo se pueden memorizar servidores sin necesidad de conectarse a ellos. Si en algún momento interesa conectarse sólo hay que cambiar el valor de dicho campo.

Por otro lado, a la hora de gestionar el intercambio de datos con el cliente, se necesita que éste lo haga de una forma fiable y lo más rápida posible. Para dar solución a esto, a la hora de la conexión se hace una copia de los servidores a los que se va a conectar, desde la lista enlazada hasta un array estático, figura 7.6. Así se consigue: primero seguridad, ya que este nuevo array lo gestiona únicamente el cliente sin posibilidad de que el usuario lo modifique, y segundo rapidez, ya que se recorre un array continuo en memoria que corresponde sólo a los servidores con los que se intercambia información.

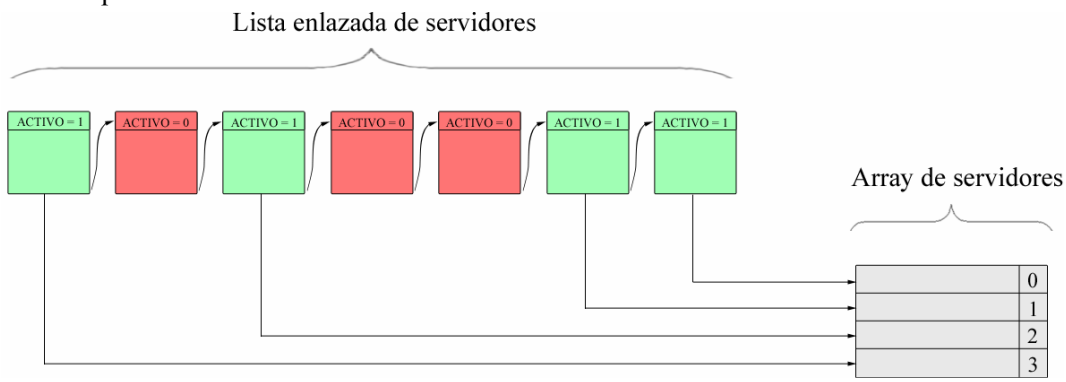


Figura 7.6 Copia desde la lista enlazada hasta el array estático

Además, lo anterior se combina con la utilización de un hilo de uso exclusivo para el intercambio de datos, figura 7.7, de manera que la rapidez del proceso aumenta considerablemente.

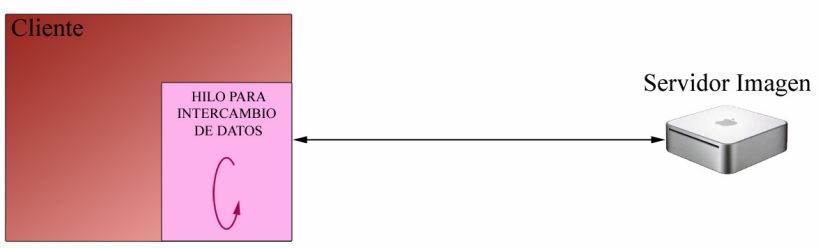


Figura 7.7 Hilo dedicado al intercambio de datos con los servidores de imágenes

Compresión del grid

Uno de los datos que el cliente solicita es el grid de ocupación. Dicho grid se puede representar en una imagen binaria donde el color negro significa libre y el color blanco ocupado.

Siguiendo con el requisito del intercambio rápido de datos se hace inviable mandar las imágenes del grid como el array que lo representa, siendo necesaria una fase previa de compresión. Dicha fase ha sufrido varias mejoras antes de adoptarse la solución final, a continuación se exponen.

En un primer momento se piensa que basta con transmitir únicamente los píxeles blancos de la imagen del grid, es decir, sólo la ocupación. Este método se descarta rápidamente puesto que ante ocupaciones grandes o abundante ruido el volumen de los datos a mandar es igualmente elevado.

Se piensa en eliminar el ruido mediante una búsqueda previa de objetos que cumpliesen, por ejemplo, ciertas condiciones de tamaño. Siendo este método igualmente desechado por la posibilidad de encontrar objetos demasiado grandes que aumenten el volumen de datos a transmitir.

Una nueva mejora pasa por mandar únicamente el contorno de los objetos. De este modo no se manda ni el ruido, ni la información de la imagen contenida ya en el contorno. Aquí el problema es la posible presencia de un número elevado de objetos de tamaño mínimo, de manera que la información a mandar es igualmente grande.

Por otro lado, recuérdese que la intención de incluir una fase previa de compresión es que la transmisión sea lo más rápida posible. Por lo tanto, no es solución implementar una compresión muy buena pero muy costosa computacionalmente ya que retarda igualmente el envío de datos. Por ello, se dejan de lado los métodos hasta aquí vistos y se empieza una nueva etapa en la fase de compresión.

Esta nueva etapa se basa en una compresión ya implementada, compresión JPEG. Se consiguen buenos resultados aplicando únicamente JPEG a las imágenes que componen el grid. No obstante, hay ciertos parámetros que se van a modificar en busca de una mayor eficiencia.

El algoritmo JPEG trabaja con la imagen en el dominio frecuencial eliminando componentes en función de su menor amplitud y de su mayor pulsación entre otras. Esto quiere decir que los cambios bruscos van a desaparecer, dando como resultado una nueva imagen de bordes

suavizados. Por ejemplo en la figura 7.8 se presentan una serie de bandas blancas y negras antes y después de aplicar JPEG. Se puede ver cómo en la imagen comprimida los bordes están modificados.

Figura 7.8 Bordes antes (izquierda) y después (derecha) de la compresión JPEG

Si la calidad de compresión aumenta los bordes quedan mejor definidos pero a cambio se genera un mayor número de datos. Por otro lado, si los colores no sufren cambios bruscos el volumen de datos generado es menor ya que predominan las componentes de baja frecuencia. Otro aspecto a tener en cuenta es el tiempo necesario para comprimir una imagen, en principio se puede pensar que a mayor compresión menor tiempo de cómputo. Se verán los resultados.

A continuación se realiza un estudio para conocer la manera más adecuada de comprimir el grid. El estudio ha sido realizado comprimiendo un número elevado de veces, 5000, una imagen de dimensiones 640x480.

En primer lugar se compara la calidad de la imagen con su tamaño final, se presenta en la figura 7.9.

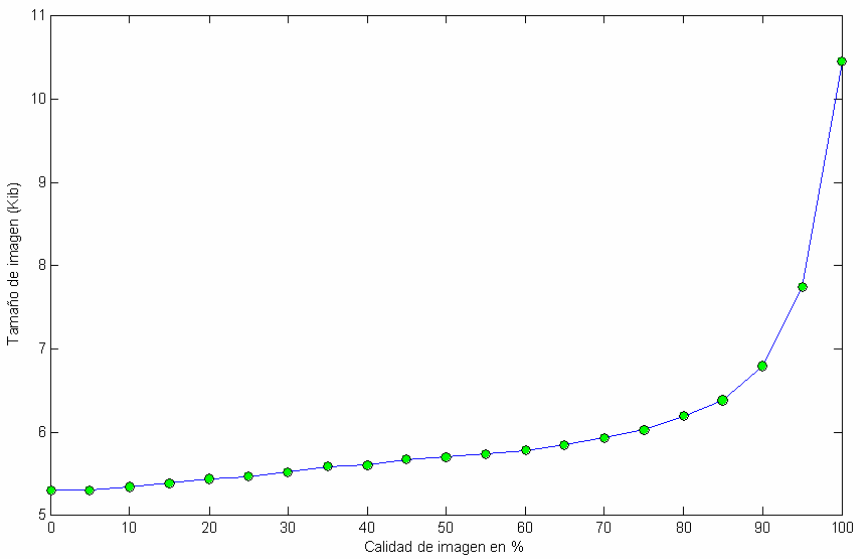


Figura 7.9 Relación Tamaño/Calidad de compresión

A medida que aumenta la calidad de la imagen su tamaño es mayor. No obstante, es a partir del 80% cuando dicho crecimiento se acentúa considerablemente.

Tiempo de cómputo respecto calidad se representa en la figura 7.10. Para esta prueba se han comprimido/descomprimido series de 5000 imágenes de 640x480.

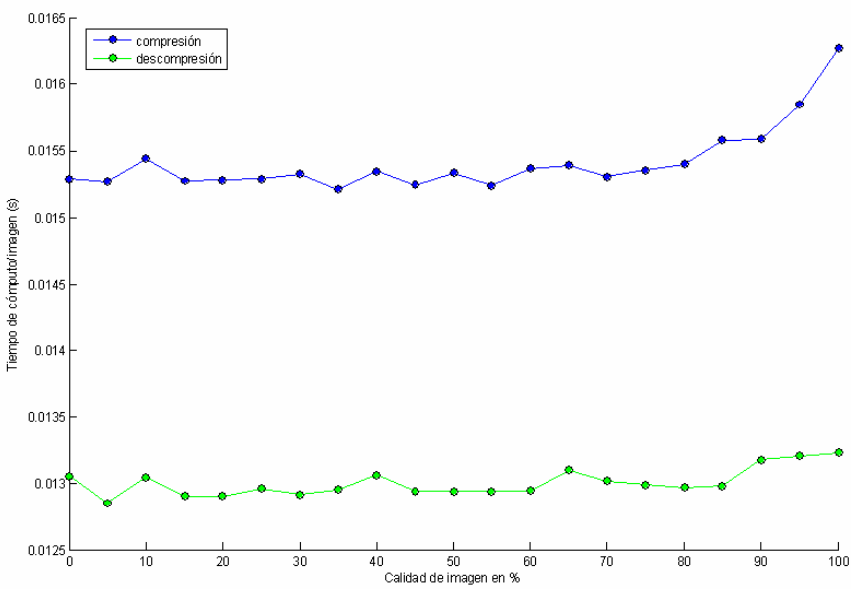


Figura 7.10 Relación T. cómputo/Calidad en imagen JPEG

Para calidades comprendidas entre 0% y 80% el tiempo de cómputo no parece ser dependiente. Es para valores superiores al 80% donde se aprecia una subida en la curva de compresión. Aún así, la máxima diferencia de tiempo es muy leve por lo que se supone un tiempo de cómputo independiente a la calidad de la imagen.

Por último cabe pensar cómo afectan los cambios de color en el tamaño final. La compresión JPEG se basa en la distribución en el dominio frecuencial de la imagen, donde esta distribución es generada a partir de dichos cambios de color.

Se muestra una gráfica donde se compara el tamaño final con el valor del cambio de color. Para realizar la prueba se han utilizado cinco imágenes donde la diferencia entre el máximo y mínimo color es de 255, 150, 50, 25 y 5, sobre el negro.

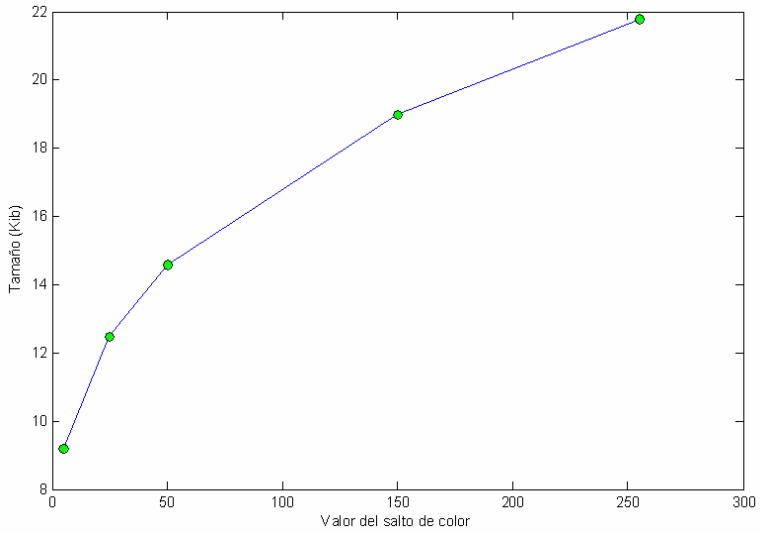


Figura 7.11 Relación Tamaño/Salto de color

El tamaño es claramente dependiente de los colores usados, es decir, las imágenes izquierda y derecha en la figura 7.12 son de tamaños distintos tras la compresión.

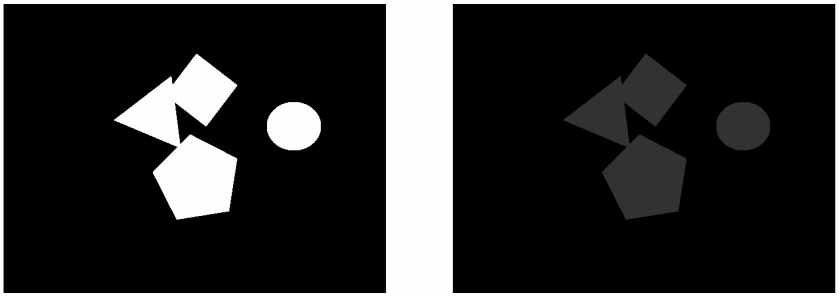


Figura 7.12 Imágenes con distintos valores en sus cambios de color

A partir de cómo afecta la calidad y cambios de color sobre el tiempo de cómputo y tamaño; se procede a fijar los parámetros en la aplicación. En concreto se compone el grid con los colores RGB (0, 0, 0) y (10, 10, 10), con una calidad del 70%. Consiguiendo imágenes de aproximadamente 6Kib en vez de 300Kib. Se reducen hasta el 2% de su tamaño original.

Debido a que JPEG es una compresión con pérdidas, se debe comprobar que la desviación entre las imágenes original y final no es excesiva. Se muestra en la figura 7.13 el resultado del ejemplo anterior. El porcentaje de píxeles iguales es del 99,9%.

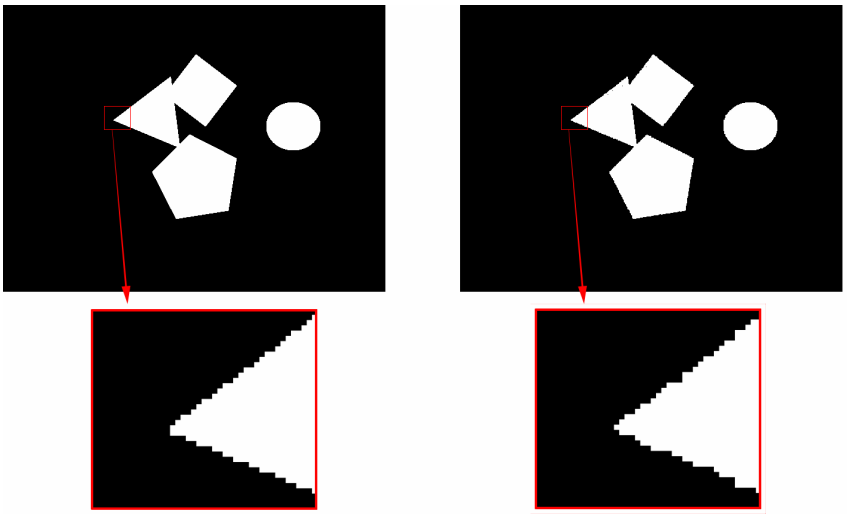


Figura 7.13 Imagen original y final tras la descompresión. Porcentaje de igualdad 99,9%

Ahora surge un nuevo inconveniente. El cliente tiene que descomprimir todo el conjunto de imágenes que mandan los servidores para generar una única imagen de grid. Esto se debe realizar lo más rápidamente posible y no puede consumir elevados recursos ya que el cliente a su vez tiene otras tareas que realizar. Por lo tanto la descompresión no debe convertirse en una tarea compleja.

La idea para solucionar esto es tratar con imágenes de menor tamaño. En concreto se minimizan las imágenes cuatro veces sobre su tamaño original. De 640x480 a 160x120.

Ésta ha sido la mejor manera encontrada para comprimir el grid en el servidor y para que el cliente lo pueda interpretar sin emplear excesivos recursos. El proceso completo de compresión del grid es el mostrado en la figura 7.14.

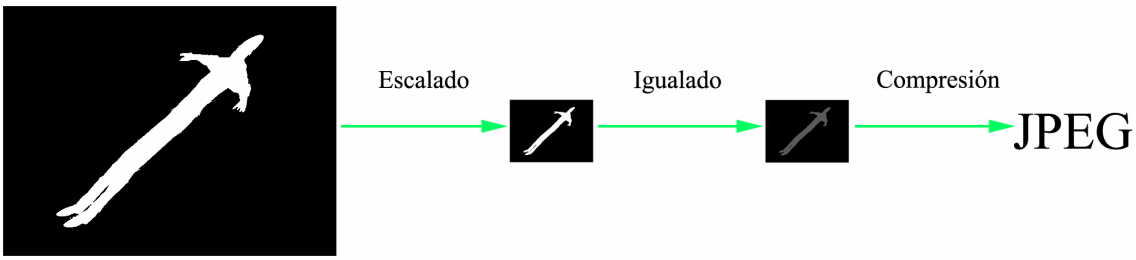


Figura 7.14 Proceso de compresión del grid

Gestión del envío directo de capturas desde el servidor

Por otro lado, los servidores también mandan las imágenes que capturan en tiempo real. La técnica utilizada para enviarlas es similar a la utilizada en el grid con la salvedad de que en este caso no se suavizan los cambios de color.

Este proceso no tiene que estar sincronizado entre servidores ni cumplir con tiempos preestablecidos, de manera que se considera de menor importancia. Para su gestión se genera un hilo independiente del control del grid, figura 7.15, posibilitando así asignar prioridades y evitar estorbos entre procesos.

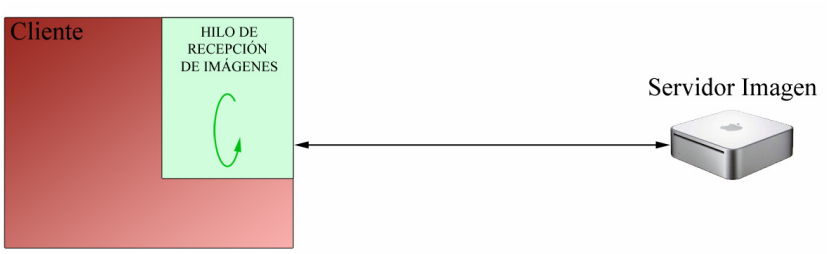


Figura 7.15 Hilo de recepción de imágenes

7.3.2. Servidor Robot

El cliente controla en todo momento el robot. Es éste el que impone a qué velocidad lineal y angular se debe desplazar y es también el que hace peticiones de odometría como se vio con anterioridad. Se puede conectar a tantos como se quiera pero sólo uno a la vez.

El control debe ser fiable y sencillo. A continuación se explica.

Gestión del robot

Al igual que en el caso de los servidores, el robot es tratado como una estructura, donde sus campos indican el estado que debe tener en cada momento.

El problema fundamental del control del robot se expuso en apartado 7.2.2. Éste es utilizar un único socket para hacer dos peticiones y una respuesta, que además se realizan con frecuencia distinta. La solución adoptada es centralizar la comunicación en un solo punto del programa. En éste se sondea si es necesario el intercambio de datos y se actúa en consecuencia.

El problema ahora se traslada a encontrar el punto adecuado de dicha comunicación.

Como se dijo, una de las peticiones a hacer es la de odometría, siendo la frecuencia con la que se genera la mayor posible. Basándose en la rapidez necesaria y en el requisito de fiabilidad, se decide lanzar un hilo exclusivo para el control y la comunicación, figura 7.16. Así incluso se puede controlar el robot aunque parte del sistema falle.

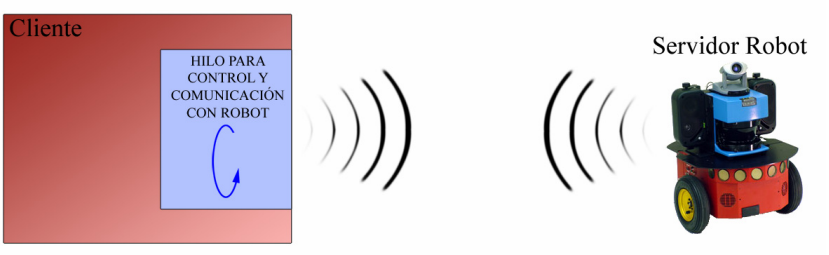


Figura 7.16 Hilo para control y comunicación con el robot

Cualquier petición relacionada con el robot es almacenada en la estructura que lo representa, siendo el hilo el encargado de mandar la orden final. La odometría es pedida y recibida cada vez que se inicia un intercambio de datos, así se refresca a la mayor rapidez.

Controles del Robot

El movimiento del robot es una tarea que realiza el usuario. Para ello se facilitan una serie de controles a través de los que se generan órdenes que el cliente recoge y gestiona. Dichos controles se muestran en la figura 7.17 y son, de izquierda a derecha: joystick, teclado, flechas y barras integradas en el propio interfaz gráfico.

Figura 7.17 Controles de movimiento del robot

Todos los controles pueden funcionar simultáneamente, teniendo preferencia la última orden sobre todas las demás. Esto puede suponer un problema ya que por ejemplo el joystick al ser tan sensible genera datos al mínimo movimiento, incluso aunque éste sea involuntario. Para evitarlo se habilita una variable que permite el funcionamiento o no del joystick, pudiendo así usarlo exclusivamente cuando se necesite.

Los controles no interactúan directamente sobre el robot sino con la estructura que lo representa. No obstante, para tener un mayor control del acceso sólo uno de los controles puede

modificar la estructura, control maestro; los demás hacen peticiones a éste último, figura 7.18. Así se consigue centralizar el acceso a la estructura y el consiguiente ahorro del uso de semáforos para proteger variables compartidas.

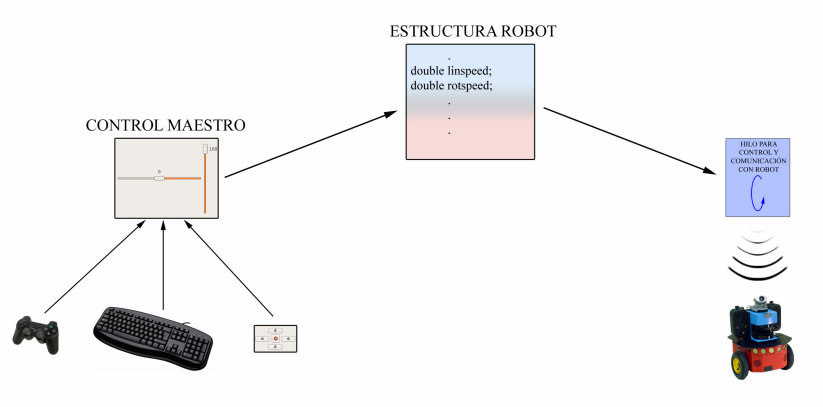


Figura 7.18 Cadena de desplazamiento del robot

Otros aspectos implementados de seguridad son los siguientes. La máxima velocidad tanto lineal como angular es prefijada de manera que nunca se alcanza una velocidad indeseada. Y si una misma velocidad es mantenida en el joystick durante un largo periodo de tiempo se asocia a un mal funcionamiento y automáticamente es puesta a cero. Recuérdese que el joystick cambia su valor ante movimientos muy leves.

7.3.3. Joystick

El joystick es uno de los dispositivos con los que se selecciona el movimiento del robot. Éste debe tener un mínimo de dos ejes a los que poder asignar la velocidad lineal y angular, de tal manera que el desplazamiento del robot es función del desplazamiento del joystick sobre cada eje.

Conexión del joystick

Lo primero que se debe hacer es conectar el joystick. Si éste tiene puerto USB no hay mayor problema, pero en el caso que ocupa este proyecto se ha utilizado un dispositivo con una precisión mucho mayor a cambio de tener que construir un adaptador para conectarlo. Sin entrar en mayor detalle se utiliza un circuito ya diseñado como el mostrado en la figura 7.19.

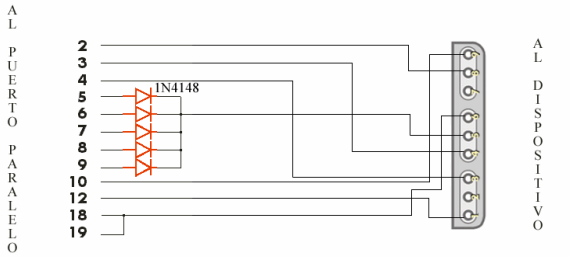


Figura 7.19 Esquema de conexión del joystick al puerto paralelo

Una vez conectado se debe insertar el correspondiente módulo en el kernel para que se pueda trabajar con él. Esto no es más que escribir las siguientes órdenes: "*modprobe joydev*" y "*modprobe gamecon map=0,7*".

Gestión del joystick

El dispositivo es gestionado por el sistema operativo como un archivo especial con las ventajas que ello reporta. El cliente lo trata como un archivo más, leyendo de él cuando considere necesario.

Para la tarea de lectura y tratamiento de datos se crea un hilo exclusivo, figura 7.20.

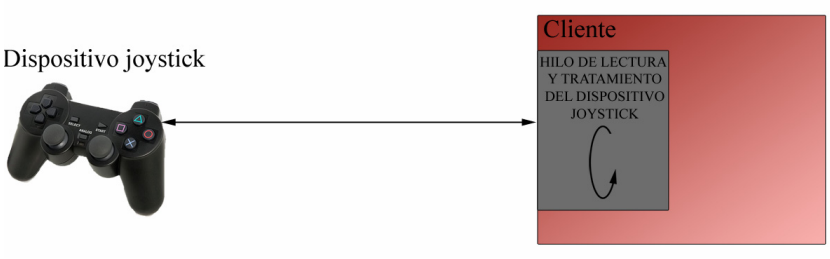


Figura 7.20 Hilo de lectura y tratamiento del dispositivo joystick

El esquema de funcionamiento de dicho hilo es el mostrado en la figura 7.21.

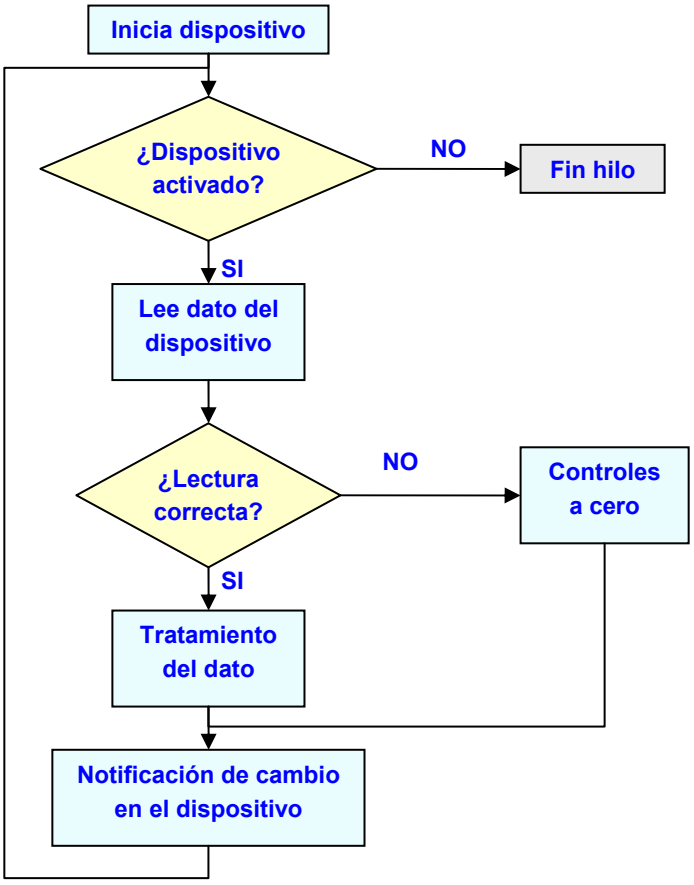


Figura 7.21 Funcionamiento del hilo de control del joystick

En la inicialización se abre el archivo especial que genera el sistema operativo y se comprueba que la apertura fue correcta. En este momento empieza el bucle que se mantendrá mientras el dispositivo esté activado, es decir, mientras el usuario decida utilizarlo.

El primer paso del bucle es leer los nuevos datos que se generan en el fichero, los cuales corresponden a movimientos en el mando. En el caso de que no se hubieran generado el proceso se queda a la espera.

Una vez leídos se comprueba que se hace de forma correcta. En caso contrario, por seguridad, se imponen los controles a la posición de seguridad, es decir, son puestos a cero.

En el caso que los datos sean correctamente leídos se pasa a la fase de tratamiento. Fase que engloba las funciones que se explican a continuación.

- En primer lugar se reconoce el control que ha sido modificado y en función de éste se aplica una u otra curva de rectificación. La curva tiene como misión adaptar los movimientos del robot a las exigencias del usuario. Por defecto se aplica la rectificación de la figura 7.22, con la que se consiguen movimientos precisos a bajas velocidades.

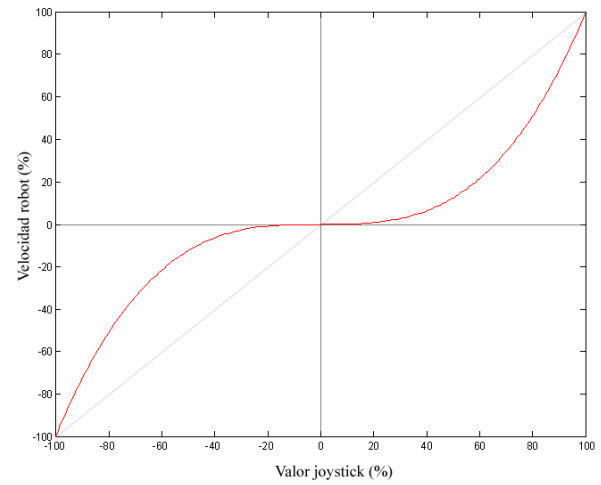


Figura 7.22 Curva de rectificación del joystick

En vista de la curva, el desplazamiento en el eje del joystick (eje horizontal de la gráfica) para imponer un tanto por ciento de velocidad en el robot (eje vertical) es distinto en valores bajos que en altos. Siendo para valores bajos necesario un movimiento mucho mayor. De este modo la navegación se convierte en sencilla y precisa.

La última función del bucle es notificar al cliente las variaciones que se han producido. Con el fin de ahorrar recursos, hasta que no se genera un cambio no se inicia la cadena que modifica la velocidad del robot. Y aun más, no todos los cambios van a ser procesados. Un timer es el encargado de recoger las notificaciones, siendo éstas sobrescritas de modo que sólo se atenderá la última antes de que el timer se ejecute.

Calibración del joystick

Cada joystick es reconocido por el sistema operativo de un modo distinto, siendo la numeración de sus ejes función del dispositivo conectado. Esto quiere decir que si el cliente no está preparado entonces sólo va a funcionar ante un único mando. No obstante, se ha implementado un sistema de calibración de ejes, abriendo la posibilidad a conectar cualquier dispositivo.

Dicho sistema de calibración se encuentra integrado en el cliente y su función es reconocer qué eje quiere ser utilizado para qué movimiento. Por ejemplo, el eje horizontal puede ser asignado a la velocidad angular y el vertical a la velocidad lineal. Su funcionamiento es el mostrado en el diagrama de flujo de la figura 7.23.

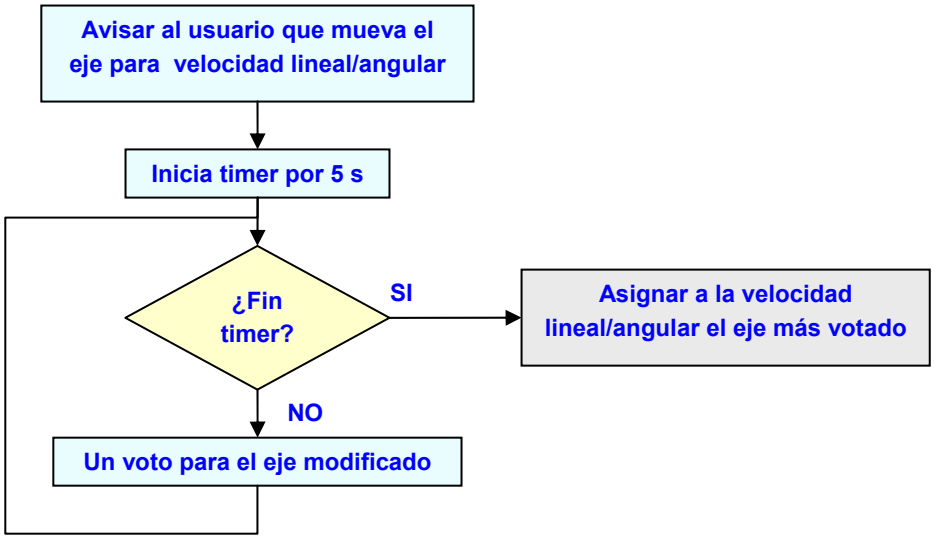


Figura 7.23 Sistema de calibración del joystick

7.3.4. Representación de imágenes

Una de las maneras en las que el cliente muestra información al usuario es a través de imágenes. En este caso se van a representar tres tipos: la odometría del robot, el grid de ocupación y las capturas en tiempo real por parte de los servidores.

La recepción de datos asociados a cada una de las imágenes es similar en los tres procesos ya que se realiza por medio de hilos. No obstante, el camino hasta su representación es distinto.

Odometría del robot

No todos los datos de odometría proporcionados por el hilo van a ser representados. Existe un timer cuya misión es componer y representar imágenes de forma constante descartando los datos entre interrupciones.

Dicho tiempo entre interrupciones debe ser lo suficientemente bajo como para generar sensación de continuidad entre imágenes y lo suficientemente alto como para no sobrecargar el sistema. Para ello se impone un tiempo por defecto, siendo posible su modificación ya que éste depende, entre otros, de en qué la máquina se ejecute el cliente.

Los datos de odometría son: posición, velocidad y ángulo. A partir de éstos se genera una imagen en la que aparece el robot y el rastro dejado por él. Esta tarea se realiza en tres pasos. Se representan en la figura 7.24.

- Primero se parte de una imagen de fondo. Ésta puede ser por ejemplo un mapa del terreno sobre el que se va a navegar así como cualquier otra. Dicha imagen se puede cambiar en cualquier momento permitiéndose la adaptación a cualquier otro entorno.

- En un segundo paso se superpone la trayectoria generada sobre el fondo, quedando así reflejado el camino seguido.

- Por último se añade una figura del robot justamente en la posición en la que se encuentra.

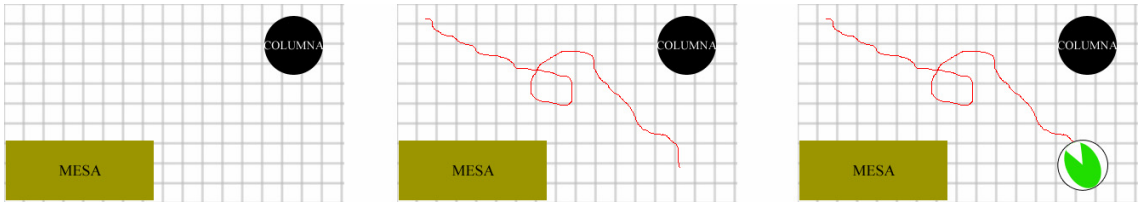


Figura 7.24 Fases de la imagen de odometría. A la izquierda el fondo elegido, en el centro superpuesto el rastro y a la derecha el robot.

La imagen de odometría está compuesta, como imagen que es, por píxeles. No obstante, está representando un movimiento por un entorno real que se mide en metros. Es por lo tanto necesaria una conversión de un espacio a otro.

Dado que la imagen de fondo es elegida por el usuario, también lo es la conversión. Para ello se requiere introducir la dimensión real del mapa, es decir, qué tamaño en metros representa la imagen elegida. De esta forma, el programa cuenta con suficiente información como para representar la odometría en las coordenadas correspondientes de la imagen.

También se permite modificar el tamaño del robot. No se debe olvidar que el cliente puede controlar varios y que éstos pueden tener medidas distintas. En la imagen se reflejan sus dimensiones, consiguiendo así una navegación más sencilla y segura puesto que además de conocer el punto en el cual se encuentra el robot, se conocen sus bordes. La intención es que resulte más complicado golpearse con objetos representados en el mapa.

Grid de ocupación

Como se ha visto, cada servidor manda su grid de ocupación al cliente, siendo éste último el encargado del tratamiento del conjunto de datos para componer el grid final.

Las imágenes llegan minimizadas, igualadas y comprimidas por lo que en algún momento se deben realizar las operaciones inversas. Como se verá a continuación, éstas se ejecutan en puntos distintos del programa con la intención de aumentar la eficiencia en el proceso. Recuérdese que éste se ejecuta varias veces por segundo y realiza varias descompresiones JPEG, lo cual es costoso computacionalmente.

La figura 7.25 muestra el proceso de generación del grid en el cliente.

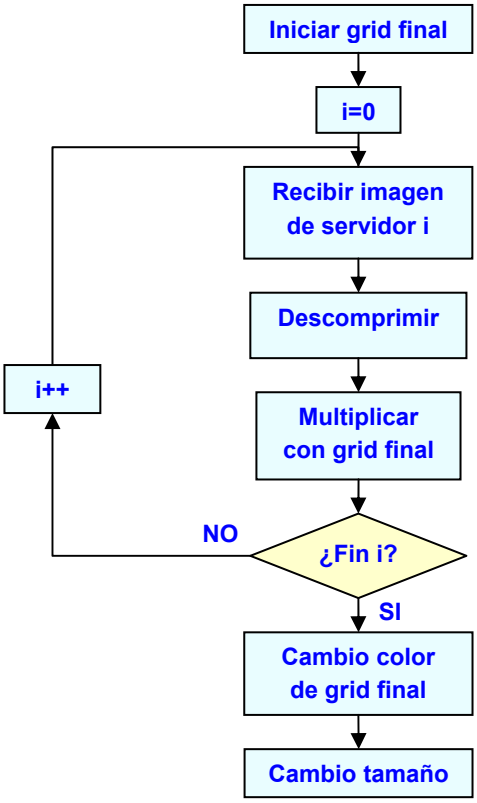


Figura 7.25 Proceso de generación del grid de ocupación final

En el primer punto del proceso se inicia el grid final. Esto no es más que componer una imagen completamente blanca para que no afecte en su posterior multiplicación con las demás. No obstante, se realiza una mejora con el fin de evitar la primera multiplicación y es iniciarlo como la primera imagen recibida.

A continuación se inicia un bucle donde se realizan las tareas de recepción, descompresión y multiplicación.

- En la recepción se genera un buffer con los datos de la imagen JPEG la cual se encuentra minimizada, igualada y comprimida.

- La descompresión genera una imagen visible quedando pendiente las tareas del cambio de color y cambio de tamaño. Con el fin del ahorrar recursos, ambas tareas se realizan una única vez por cada grid final en lugar de una vez por cada imagen recibida del servidor, es decir, ambos procesos se aplican tras acabar el bucle.

- Se pretende ahora conocer las coincidencias entre imágenes. Esto se realiza multiplicándolas, de modo que solo se mantienen aquellos píxeles cuyo valor es para todas las imágenes distinto de cero, o lo que es lo mismo, distinto del negro.

El bucle se repite tantas veces como cámaras se encuentren capturando ya que por cada imagen se tiene que hacer una multiplicación.

Una vez acabado el bucle se tiene una primera versión del grid final. Aún queda por realizar el cambio de color y el cambio de tamaño. Esto se hace con la intención de mostrar de la mejor forma posible la información al usuario. Ambas tareas se realizan en los dos últimos pasos del proceso.

Un ejemplo real de la composición del grid se muestra a continuación.

Se parte de una escena captada por dos cámaras distintas. En ellas se destaca la incorporación de un objeto cilíndrico y alto en el centro.

Figura 7.26 Capturas de dos cámaras distintas en el mismo instante de tiempo

Cada uno de los servidores manda al cliente su grid de ocupación. Éste presenta el aspecto visto en la figura 7.27, donde el objeto cilíndrico pasa a proyectarse sobre el plano del suelo.

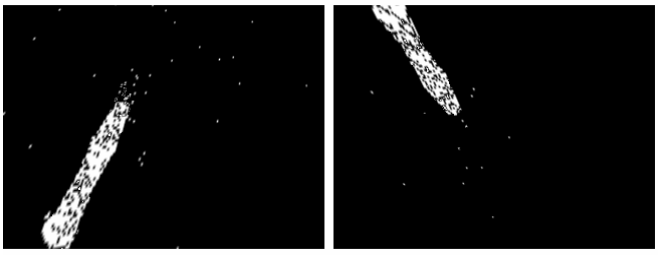


Figura 7.27 Grid de ocupación compuesto por cada uno de los dos servidores

Ahora se procede a la multiplicación, lo que equivale a superponer ambas imágenes y generar una tercera con, exclusivamente, las coincidencias. Se muestra en la figura 7.28 con colores no reales para su mejor compresión.

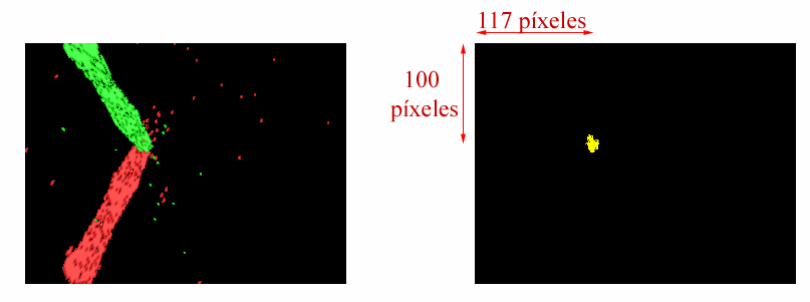


Figura 7.28 Multiplicación de ambos grid (verde y rojo) y resultado (amarillo)

Así es como se presenta la información al usuario. Una zona blanca sobre fondo negro que informa que en esa posición se encuentra situado un objeto. Se puede saber exactamente su colocación sobre el plano sin más que conocer la equivalencia entre un píxel en la imagen y un milímetro en el plano. En el caso de este proyecto se ha prefijado como 12mm/píxel. De este modo, el objeto en la imagen anterior está situado en:

X_w = 117 píxeles x 12 mm/píxel = **1.404 metros**

Y_w = 100 píxeles x 12 mm/píxel = **1.2 metros**

Capturas en tiempo real de los servidores

Las capturas obtenidas directamente de las cámaras son enviadas al cliente para que éste las presente al usuario.

En el servidor, dichas imágenes son minimizadas y comprimidas mediante JPEG. No se entra en detalles puesto que ambos procesos son idénticos al caso del grid.

En el cliente, un hilo es el encargado de la recepción, y un timer el encargado de su descompresión y presentación al usuario, figura 7.29. Gracias a dividir el proceso en un hilo y un timer, aunque se reciban imágenes a gran velocidad, sólo se descomprimen y representan cuando se considere oportuno, ya sea por motivo de recursos disponibles o por exigencias del usuario.

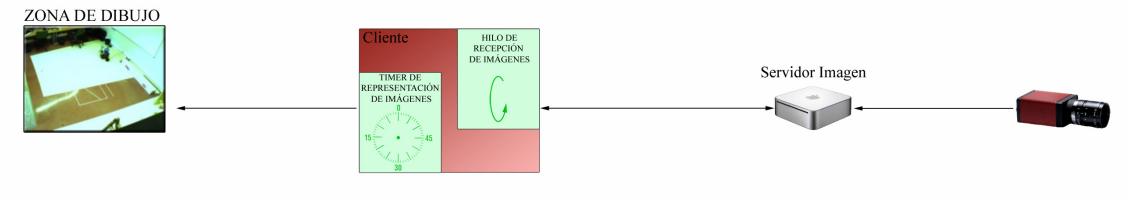


Figura 7.29 Desacoplo de la recepción y presentación de las imágenes de los servidores

El problema en este caso se encuentra en el número máximo de imágenes que se pueden representar a la vez. La limitación viene impuesta por las dimensiones de la pantalla y capacidad de cómputo del cliente.

En el programa aquí expuesto se reservan cuatro espacios de representación llamados zonas de dibujo. Es decir, el número máximo de imágenes visibles simultáneamente es cuatro. Además, se diseña un procedimiento por el cual dichos espacios pueden cambiar la fuente de datos, seleccionando qué cámara se representa en qué espacio.

El método de actuación se apoya en generar un hilo por cada zona de dibujo, donde cada hilo recibe las imágenes destinadas a un único espacio. Para diferenciar la recepción, cada uno recibe los datos por un puerto distinto, en concreto 1540, 1541, 1542 y 1543, y rellena unos buffers no compartidos asociados a cada zona, figura 7.30.

Es en la petición al servidor donde se asigna el puerto para la transmisión. De manera que si se quiere representar las imágenes de otra fuente lo único que se debe hacer es desasignar el puerto al servidor anterior y asignárselo al nuevo.

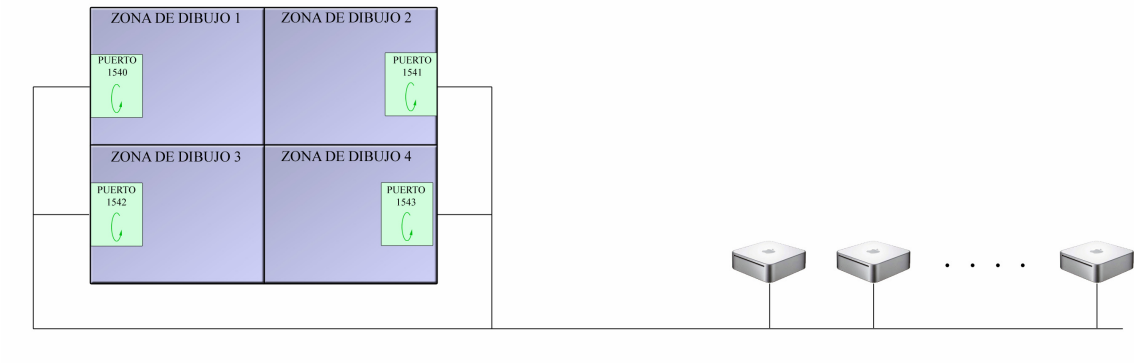


Figura 7.30 Hilos de recepción de imágenes. Uno por zona de dibujo

Otra opción que se ofrece al usuario es elegir la calidad con la que los servidores mandan las imágenes. Esto se puede seleccionar durante la conexión o incluso durante el envío de imágenes. La finalidad es liberar carga en la red evitando posibles pérdidas de datos si ésta no es lo suficientemente rápida, o visualizar mejor las imágenes si se soporta.

7.3.5. Registro de eventos

Con el fin de realizar un seguimiento en la ejecución del programa y para simplificar la depuración del mismo, se ha implementado un registro de eventos mediante el cual se puede almacenar cualquier mensaje o variable en cualquier parte del código. Éstos quedan guardados en un fichero que por defecto es *_events.txt*, pudiendo ser cambiado cuando se desee.

Para utilizarlo sólo hay que hacer una llamada a la función *registrar_evento(char * mensaje)* siendo *mensaje* el texto que se quiere almacenar.

7.3.6. Sincronización de procesos

Como ya se ha explicado, en el cliente se ejecutan simultáneamente varios procesos, figura 7.31.

- Para los servidores conectados se abre un hilo encargado de la comunicación.
- Por cada zona de dibujo existe un hilo de recepción de datos y un timer global para el tratamiento y representación de imágenes.
- El robot abre otro hilo para el intercambio de datos y utiliza un timer para la representación de la odometría.
- El joystick utiliza otro hilo para sondearlo y un timer para modificar los controles.

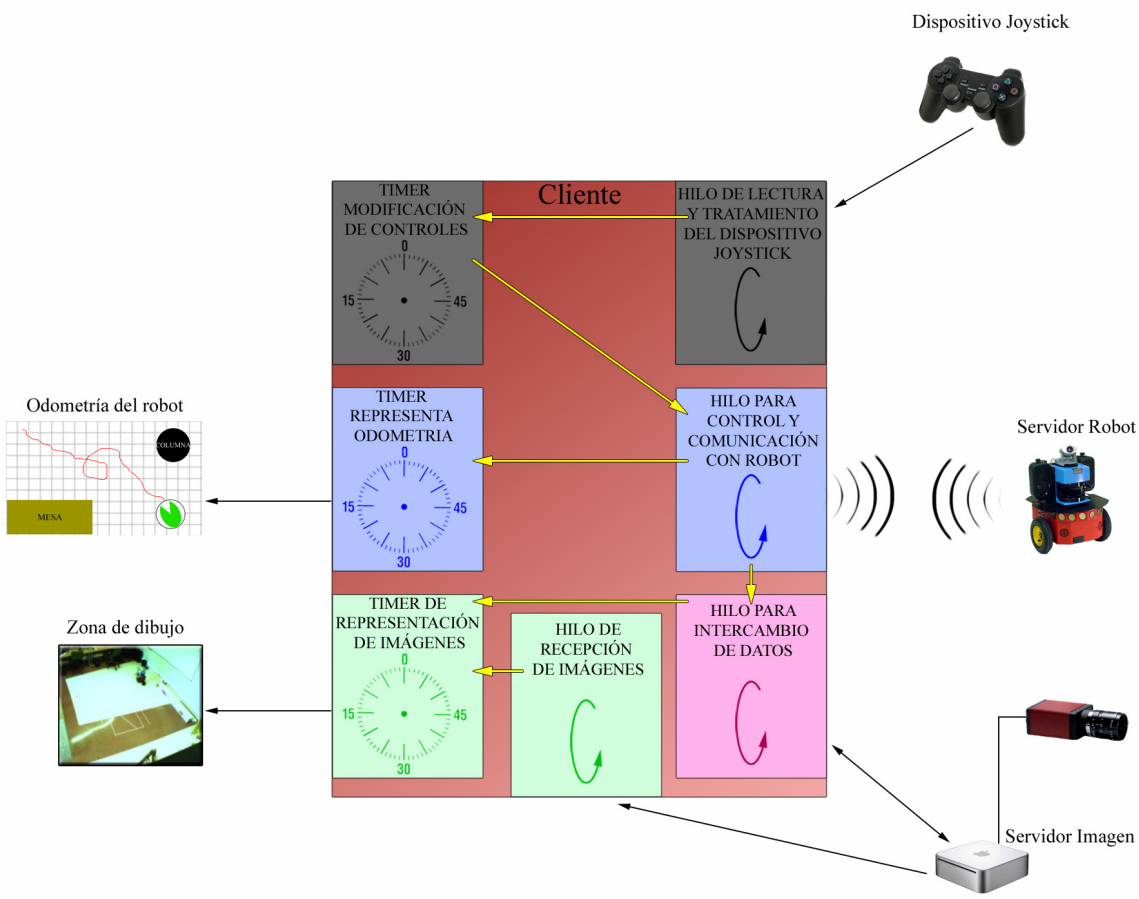


Figura 7.31 Hilos y timers en el cliente

Por ejemplo, si se conecta con cuatro servidores, se activa el robot, se maneja con el joystick y se usan cuatro zonas para representar imágenes; entonces se lanzan siete hilos y tres timer.

Existen procesos con datos compartidos. Por ejemplo, el timer no puede mostrar imágenes en la zona de dibujo sin que antes el hilo las reciba. Esto no resulta problemático ya que en la mayoría de los casos un proceso siempre escribe y otro siempre lee. Si el proceso que lee no encuentra datos simplemente se para o utiliza los últimos conocidos.

Se destaca el uso de cuatro semáforos para cuatro casos particulares. Un semáforo protege el buffer de las imágenes recibidas desde el servidor ya que el timer antes de representarlas opera con él. Otro semáforo es usado de forma idéntica que el anterior pero protegiendo el grid ya que nuevamente el timer que se encarga de representarlo lo modifica. Un semáforo controla la

escritura en el fichero que registra los eventos. Éste es necesario ya que los eventos pueden llegar de diversas partes del programa e incluso mientras el fichero se está usando. El último semáforo controla la variable que indica el estado en que se encuentra el programa ya que nuevamente las peticiones de modificación pueden llegar de diversos puntos del código.

Otro aspecto que se debe destacar es la necesidad de sincronización entre la recepción de las imágenes del grid con la recepción de la odometría en el robot. Es decir, el grid debe mostrar el robot en las mismas coordenadas que la odometría, siempre y cuando coincidan los orígenes.

Por un lado existe un hilo encargado de actualizar la odometría de una forma rápida, "*hilo para el control y comunicación con robot*". Y por otro existe un hilo encargado de recibir las imágenes del grid,"*hilo para intercambio de datos*". Ambos procesos son independientes pero se necesita asegurar la sincronización de sus datos. Esta labor se asigna al hilo encargado de la gestión del grid, y la técnica empleada es realizar una copia de la odometría justo en el momento en el que se reciben las imágenes. Así se obtiene un dato de posición del robot por cada imagen del grid, asegurando su sincronización.

7.4. Interfaz gráfico

Con toda la funcionalidad explicada hasta el momento el cliente se convierte en una herramienta completa pero difícil de gestionar. Introducir comandos por línea de órdenes o mediante menús resulta excesivamente tedioso e incómodo.

La solución pasa por implementar un interfaz gráfico donde de forma sencilla se introducen los parámetros necesarios.

7.4.1. Pantalla de servidores

La pantalla principal del programa es la mostrada en la figura 7.32. Se explican seguidamente los números marcados sobre ésta.

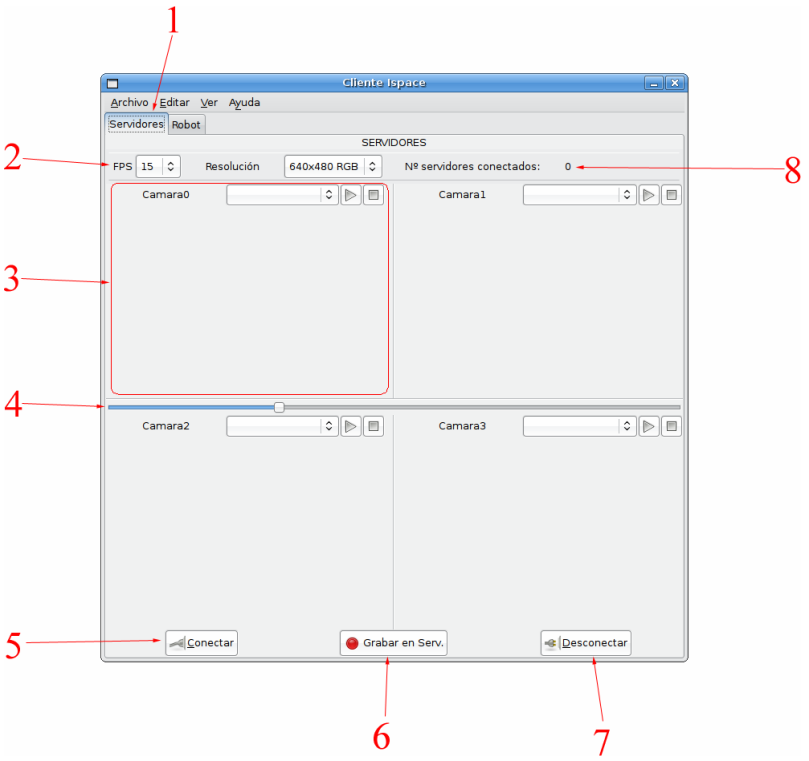


Figura 7.32 Pantalla principal del programa Cliente

1.-Existen dos pestañas principales en el interfaz, *servidores* y *robot*. Por defecto el programa comienza en la opción de *servidores*, siendo ésta la pantalla mostrada en la figura anterior.

2.-Mediante dos desplegables se configura la velocidad de captura de las cámaras y su resolución. El cliente está preparado para imponer velocidades de 7.5, 15 y 30fps y resoluciones de 640x480RGB, 320x240YUV y 640x480YUV. Siendo posible aumentar las combinaciones.

En el programa, la modificación de dichos desplegables se ve reflejada sobre dos variables: "*__fps_escogida*" y "*__resolucion_escogida*". Siendo en el momento de la conexión cuando el valor es volcado sobre el comando que se envía al servidor.

3.-En el punto tercero se hace referencia a una zona de dibujo, ésta incluye el selector de direcciones, el espacio de imagen y los botones de inicio y parada.

Mediante el selector de direcciones se elige qué servidor manda imágenes a la zona de dibujo, y mediante los botones de inicio y parada se selecciona cuándo se quieren ver dichas capturas. La dirección viene acompañada de un valor, el cual hace referencia al número de cámara dentro del mismo servidor, figura 7.33.

Figura 7.33 Zona de dibujo en funcionamiento

El funcionamiento interno afecta directamente sobre el comando enviado a los servidores. En éste existen tres campos: "*transmite_imagenes*", "*puerto_imagenes*" y "*num_camara*", los cuales controlan respectivamente: si el servidor tiene que mandar imágenes o no, el puerto asignado y el número de cámara que debe transmitir.

Las zonas de dibujo sólo se encuentran operativas una vez el cliente se conecta. De otro modo el selector de direcciones se encuentra vacío y los botones generan un mensaje de error.

4.-A través de esta barra se selecciona, en caliente, la calidad con la que los servidores envían las capturas al cliente. El lateral izquierdo indica una calidad de compresión del 0% y el lateral derecho del 100%, figura 7.34.

Figura 7.34 Diferentes calidades en las capturas enviadas por el servidor

La modificación de la barra actúa nuevamente sobre uno de los campos en el comando que se envía, "*calidad_imagen*". Éste es recibido en los servidores, los cuales comprimen las imágenes en consecuencia.

5.-Presionando el botón *Conectar* se inicia el proceso de conexión. Según lo descrito en apartados anteriores, se abren sockets y se intercambian datos con los servidores presentes en el array estático. Dicho array se genera a partir de una lista enlazada cuya gestión se verá en la explicación del menú *Editar*.

El camino seguido en la conexión pasa por los siguientes puntos:

- Construcción del array estático de servidores a partir de la lista enlazada.
- Apertura de tres sockets por servidor: "*__sock_comandos*", "*__sock_datos*", "*__sock_grid*".
- Orden a los servidores para que localicen sus cámaras. En este momento se conmuta al estado "*Conectado*".
- Se componen los comandos a enviar con la configuración de "*resolución"* y "*fps"* elegida.
- Lanzamiento del hilo de intercambio de datos con los servidores.
- Lanzamiento del hilo encargado de recibir imágenes desde los servidores y lanzamiento del timer encargado de representarlas. En este momento se conmuta al estado "*Capturando*".

6.-Una vez el cliente se encuentra capturando, se tiene la posibilidad de grabar las imágenes de las cámaras. Éstas se almacenan en el lugar donde se ejecute cada servidor, y por defecto en la carpeta "*/temp/*".

7.-Mediante el botón "*Desconectar*" el programa vuelve al mismo estado que al inicio de su ejecución. Se cierran las conexiones con los servidores, se matan los hilos y los timers dejan de interrupir.

8.-En este lugar se presenta la información del número de servidores a los que se está conectado. En la figura dicho valor es cero ya que no se ha presionado el botón *conectar*.

7.4.2. Pantalla de robot

Conmutando a la pestaña de *Robot* se gestiona lo referente a éste, figura 7.35.

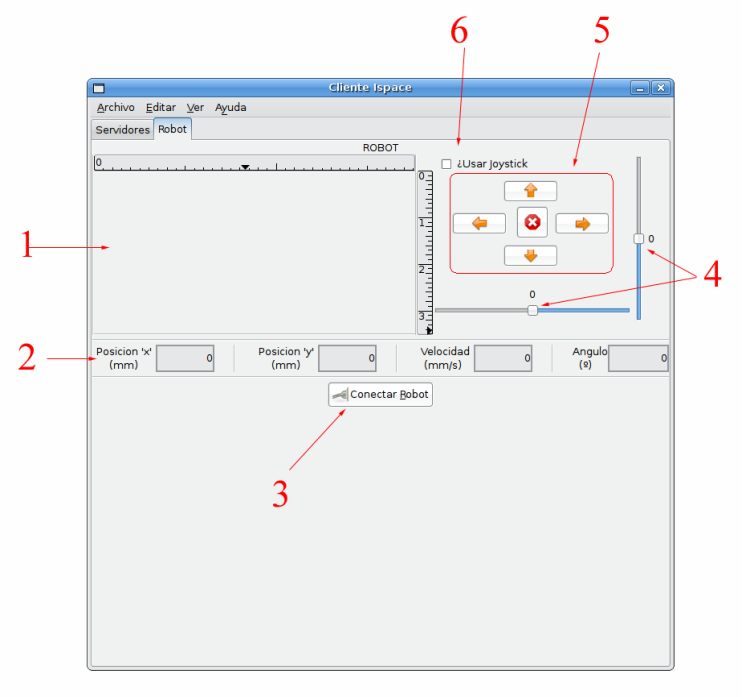


Figura 7.35 Pantalla de control del robot

1.-Éste es el espacio destinado a la representación de la odometría en forma de imagen. Por cada dato de posición se dibuja un punto, de forma que el conjunto de todos ellos forma el rastro dejado, figura 7.36.

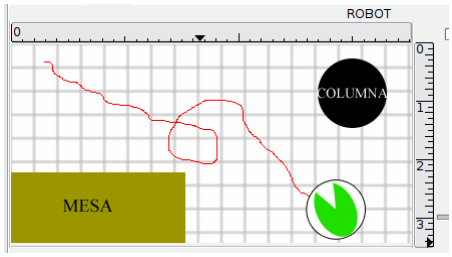


Figura 7.36 Odometría en forma de imagen

2.-Aquí se muestran numéricamente los datos de odometría. Se proporcionan valores de posición, velocidad y ángulo. Todos ellos referidos respecto a la posición inicial.

3.-Botón de conexión y desconexión. Cuando el robot se encuentra desconectado el botón ofrece conectarlo y viceversa.

A la hora de conectar, los pasos internos del programa son:

- Apertura del socket con el robot.
- Inicialización del timer encargado de la representación de la odometría.
- Lanzamiento del hilo encargado de gestionar la comunicación con el robot.
- Todos los controles son puestos a cero.

A la hora de desconectar, únicamente se modifica la estructura que representa al robot, dejando constancia de la intención. Es el hilo de comunicación el encargado de llevar a cabo la orden.

4.- Es uno de los controles del robot. Se modifica desplazando las barras con el puntero del ratón permitiendo así cambios bruscos. La barra horizontal controla el movimiento angular y la vertical la velocidad lineal. Los valores van desde el 100 negativo hasta el 100 positivo, correspondiendo al tanto por ciento del valor máximo impuesto para el movimiento.

Dichas barras son el único control con acceso a la estructura del robot. Cuando una es movida se deja constancia en la estructura, siendo el hilo de comunicación el que recoge y transmite la orden.

5.- Es otro de los controles del robot. Cada vez que una flecha es pulsada se genera un desplazamiento en las barras explicadas en el punto anterior. El valor de dicho desplazamiento es configurable, pudiendo ir desde 1 hasta 100.

Existen dos maneras de pulsarlas: Se puede realizar directamente con el ratón, o se pueden apretar mediante la tecla que las representa. Éstas son I, K, J, L, figura 7.37.

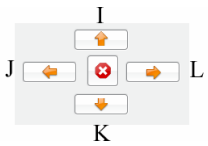


Figura 7.37 Equivalencias teclado/flechas de movimiento

6.- Aquí se hace referencia al selector del joystick. Éste tiene como función permitir o no su utilización.

A la hora de activarlo se desencadenan los siguientes procesos:

- Comprobación del funcionamiento del dispositivo. En caso negativo se muestra el mensaje de la figura 7.38 y se desactiva.

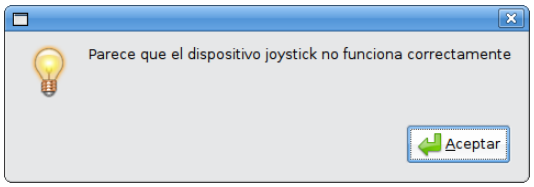


Figura 7.38 Mensaje de error al no estar bien configurado el joystick

- Lanzamiento del hilo encargado de la lectura y tratamiento de los datos del joystick.
- Inicialización del timer encargado de modificar los controles del robot.

En la desactivación se finaliza el timer y se mata el hilo.

7.4.3. Menú Archivo

El menú archivo contiene únicamente dos entradas, "*guardar odometría*" y "*salir*", figura 7.39.

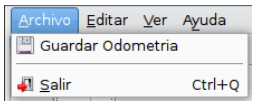


Figura 7.39 Menú Archivo

Bajo la entrada "*Guardar odometría*" se realiza la función de almacenar la imagen correspondiente a la odometría. Esto es salvarla en un archivo tal cual se visualiza en la pantalla del robot.

Para que la operación tenga éxito se debe especificar el formato junto al nombre del archivo; por ejemplo, si se desea guardar en un archivo llamado *odo* bajo un formato *.bmp,* entonces se debe introducir "*odo.bmp*".

7.4.4. Menú Editar

Este menú contiene cuatro entradas y a su vez una de ellas contiene otras tres, figura 7.40.

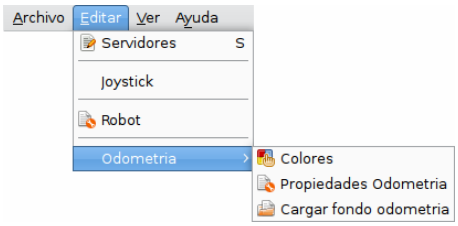


Figura 7.40 Menú Editar

Servidores

En la entrada "*servidores*" se gestiona la lista enlazada de los servidores. Se pueden realizar operaciones tales como borrar, añadir, salvar y cargar. La pantalla en la que se realiza todo esto es la mostrada en la figura 7.41.

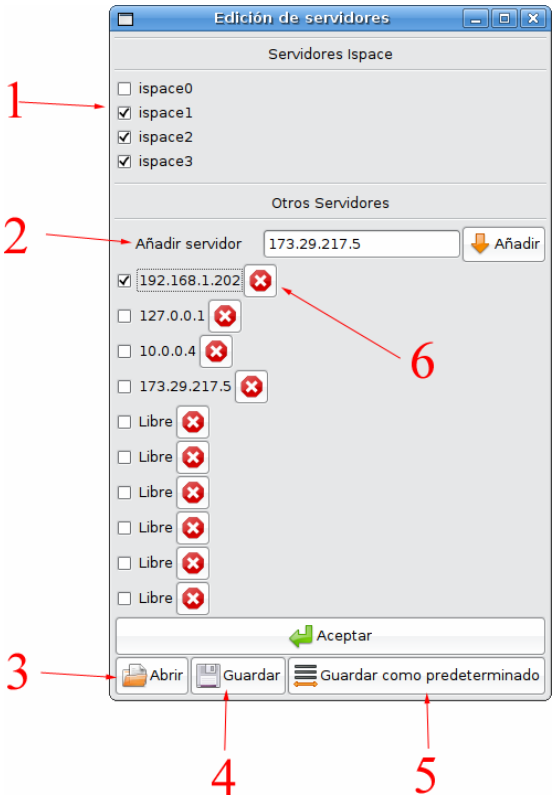


Figura 7.41 Ventana de gestión de servidores

1.-Éstos son los servidores ispace. No se pueden ni añadir ni borrar. Únicamente se puede seleccionar si conectarse a ellos o no. No obstante, dejando aparte esta peculiaridad, son tratados internamente como cualquier otro.

El botón de selección a la izquierda del nombre tiene su representación en la lista enlazada mediante el campo "*activo*". Si su valor es cero entonces el servidor no se tiene en cuenta para la conexión.

2.-Con la intención de incorporar nuevos servidores, se añade un cuadro de texto donde se teclean direcciones; seguidamente se pulsa el botón "*Añadir*" y ésta pasa automáticamente a formar parte de la parte inferior, pudiéndose así seleccionar.

Todo servidor que aparezca en la ventana tiene representación en la lista enlazada. No obstante, sólo son los marcados los que tienen el campo "*activo*" distinto de cero.

3.-Mediante el botón "*abrir*" se carga una lista de servidores previamente guardada, ahorrando así la tarea de escribirlos y seleccionarlos.

Internamente se realiza un borrado completo de la lista enlazada y a continuación se añaden los nuevos.

4.-Si se desea guardar la lista actualmente en uso sólo hay que pulsar el botón "*guardar*". De esta manera, se almacena en disco la lista tal cual en el momento actual.

5.-El botón "*guardar como predeterminado*" establece la configuración actual de servidores por defecto en la próxima ejecución del programa.

6.-Al estar acotado el número máximo de servidores, se ofrece la posibilidad de borrar los que ya no se utilicen.

Joystick

En la entrada "*joystick*" se va a configurar lo referente a este dispositivo. Esto es su dirección, calibración y comprobación del correcto funcionamiento, figura 7.42.

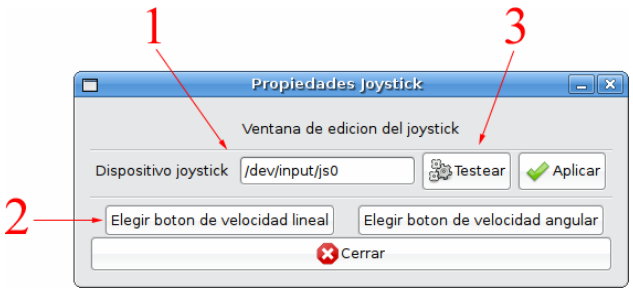


Figura 7.42 Ventana de gestión del joystick

1.-En el cuadro de texto se introduce la ruta completa del joystick tal y como lo reconoce el sistema operativo. De esta forma el cliente trata el dispositivo como un archivo más.

2.-"*Elegir boton de velocidad lineal*" y "*Elegir boton de velocidad angular*" son los dos pasos en la calibración. Como ya se dijo, si se cambia de joystick se cambia también la numeración de los ejes. Por ello se ofrece la posibilidad de elegir qué mando es utilizado para qué movimiento.

Pulsando cualquiera de los dos botones aparece un mensaje con las instrucciones a seguir, figura 7.43. Sólo hay que seguirlas.

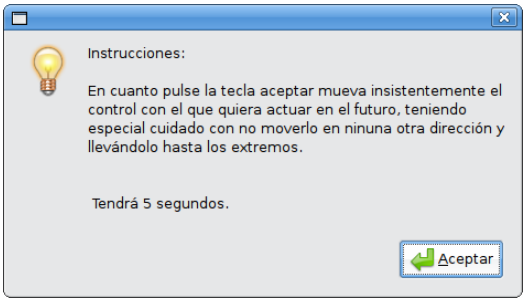


Figura 7.43 Instrucciones en la calibración del joystick

3.-Mediante el botón "*testear*" se comprueba que la dirección introducida corresponde con un dispositivo real de entrada salida como un joystick. De este modo, el usuario puede asegurarse que el sistema operativo lo ha reconocido y que puede ser utilizado.

Robot

En la entrada "*robot*" se presenta la ventana de la figura 7.44.

Figura 7.44 Ventana de gestión de robot

1.-En este punto se introduce la dirección del robot al que se va a conectar. El cliente se puede conectar a tantos robots como quiera pero tan sólo uno a la vez.

2.-Este valor hace referencia a la variación de las flechas para el control del robot. Cuando se pulsa sobre una de dichas flechas, el incremento o disminución es justamente el número introducido en este punto.

3.-Aquí se configura la máxima velocidad lineal y angular que el robot puede llegar a tomar. En todo momento se habla que los controles actúan en tanto por ciento sobre el valor máximo, siendo dicho valor máximo el asignado aquí.

Odometría

Odometría/Colores

En "*colores*" se seleccionan los colores tanto del rastro dejado por el robot como del círculo que rodea a éste. La intención es que la imagen de fondo no camufle la información que se superpone a ella, es decir, de nada sirve un rastro negro sobre un fondo negro. La solución es cambiarlo.

Se muestra en la figura 7.45 la ventana que se abre bajo la entrada "*colores*" y la ventana de selección de éstos al pulsar el botón "*seleccionar*".

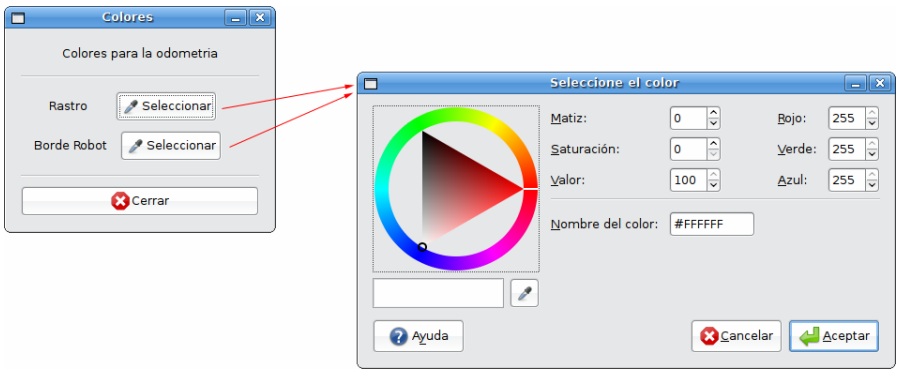


Figura 7.45 Ventanas de selección de colores

Odometría/Propiedades odometría

La siguiente entrada de odometría es "*Propiedades odometría*". En ella se configura lo referente a las medidas tanto de la imagen de fondo como del robot y su posición inicial. En la figura 7.46 se presenta dicha ventana.

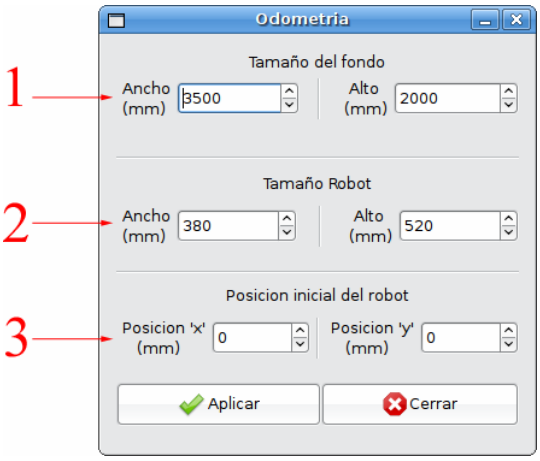


Figura 7.46 Ventana propiedades de odometría

1.-Éste es el tamaño representado por la imagen utilizada en el fondo de odometría. De esta medida se obtiene la equivalencia entre un píxel en la imagen y un milímetro en el espacio real.

Se puede dar el caso en que la equivalencia de la imagen no sea la misma en horizontal que en vertical, es decir, que un píxel a lo alto no equivalga lo mismo que a lo ancho, figura 7.47.

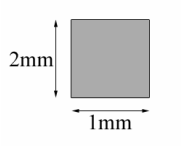


Figura 7.47 Posible equivalencia entre un píxel en la imagen y el espacio real

Esto se resuelve introduciendo las medidas de la imagen tanto en ancho como en alto. No obstante, si sólo se conoce una, entonces se deja la restante a cero y el cliente toma los píxeles como cuadrados.

2.-El tamaño del robot se introduce en milímetros. El programa automáticamente lo transforma y lo representa sobre la imagen de odometría. De esta forma se consigue una navegación donde se representan los bordes del robot con las ventajas que esto reporta.

3.- Por último se introduce la posición en la que el robot comienza su navegación. Ésta es medida desde la esquina superior izquierda de la imagen, figura 7.48.

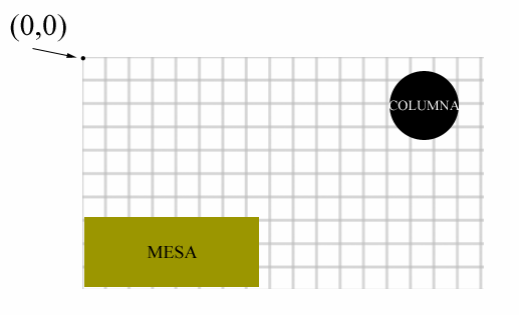


Figura 7.48 Origen de la imagen odometría. Punto respecto el que se mide el inicio de la navegación

Odometría/Cargar fondo odometría

La última entrada de odometría es "*cargar fondo odometría*". Mediante ésta se selecciona qué imagen es utiliza como fondo en la odometría. Si no se selecciona ninguna, entonces se utiliza por defecto la mostrada en la figura 7.49.

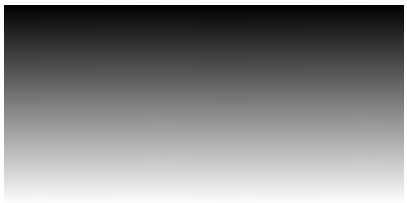


Figura 7.49 Imagen para odometría por defecto

7.4.5. Menú Ver

El menú ver no contiene entrada alguna. Se ha mantenido por si fuese necesario en actualizaciones futuras.

7.4.6. Menú Ayuda

En éste únicamente se puede consultar la entrada "*acerca de*", la cual proporciona información sobre la versión y créditos, figura 7.50.

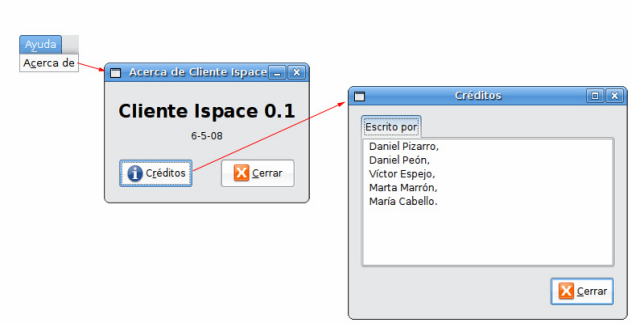


Figura 7.50 Entrada "Acerca de" del menú ayuda

7.5. Servidor

El servidor es el encargado de servir al cliente. A grandes rasgos su tarea es básicamente capturar imágenes de las cámaras y operar con ellas. A continuación se explican las funciones de mayor importancia.

7.5.1. Segmentación de fondo

La segmentación de fondo es uno de los procesos de mayor importancia en el sistema. Por ello se ha dedicado un capítulo, el quinto, donde se recorre el camino seguido hasta encontrar el algoritmo que mejor resultados aporta.

Dicha segmentación es implementada en el servidor y los pasos en los que se divide son los siguientes:

- Captura de imagen.
- Transformación de la captura al espacio invariante a la iluminación.
- Comprobación píxel a píxel, mediante su media y varianza, de si corresponden al fondo o a un objeto. Con el resultado se genera una imagen donde en blanco se representan los objetos y en negro el fondo.

Aprendizaje del sistema

Un aspecto importante que se ha obviado es la manera de conseguir la media y la varianza de cada píxel en la imagen de fondo. A continuación se explica.

Se ha implementado un algoritmo de aprendizaje donde las primeras capturas son empleadas para caracterizar cada una de las distribuciones de ruido en la imagen.

Las expresiones utilizadas son las 7.1 y 7.2 que corresponden a la media y desviación típica respectivamente.

$$\mu_{u,v} = \frac{1}{N}\sum_{i=1}^{N} I^i(u,v) \tag{7.1}$$

$$\sigma_{u,v} = \sqrt{\mu_{(u,v)^2} - \mu_{u,v}^2} \tag{7.2}$$

Siendo N el número total de imágenes usadas.

El proceso a seguir se muestra mediante un diagrama de flujo en la figura 7.51.

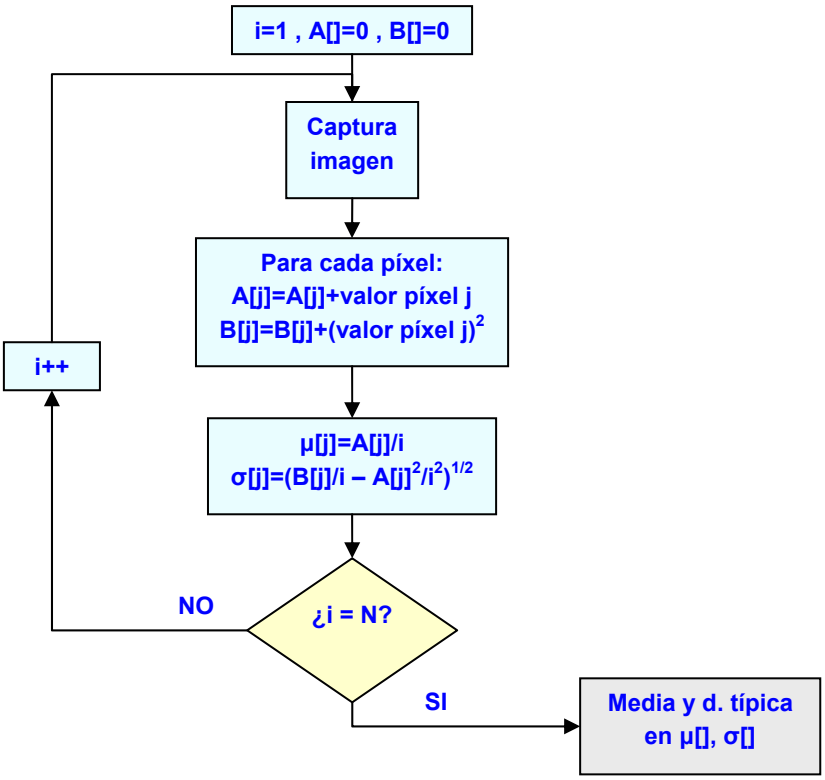


Figura 7.51 Algoritmo de aprendizaje en la segmentación de fondo

De esta forma se tiene definido cada píxel con su media y desviación típica.

7.5.2. Composición del grid

Cada servidor compone su propio grid de ocupación a partir de la segmentación de fondo y la matriz de homografía. Ambos conceptos ya han sido explicados con anterioridad por lo que no se entra en mayor detalle. El resumen del proceso es el mostrado en la figura 7.52.

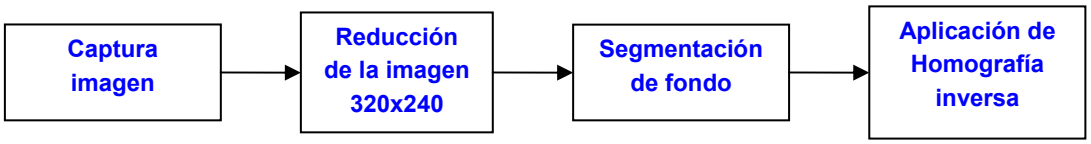


Figura 7.52 Proceso de composición del grid

Una vez que capturada la imagen, se reduce a unas dimensiones de 320x240 píxeles. La intención es acelerar el proceso de segmentación y de aplicación de homografía inversa, ya que el tiempo de éstos depende, entre otros, del tamaño de la imagen a tratar.

De la segmentación de fondo poco queda por explicar. Simplemente se aclara que el aprendizaje se realiza con las cien primeras imágenes recibidas. Durante éste, la escena debe permanecer libre de objetos.

La aplicación de la homografía inversa se realiza directamente sobre la segmentación. Tras ella se obtiene la imagen transformada al plano elegido en el espacio real, donde el origen era conocido como origen del mundo, figura 7.53.

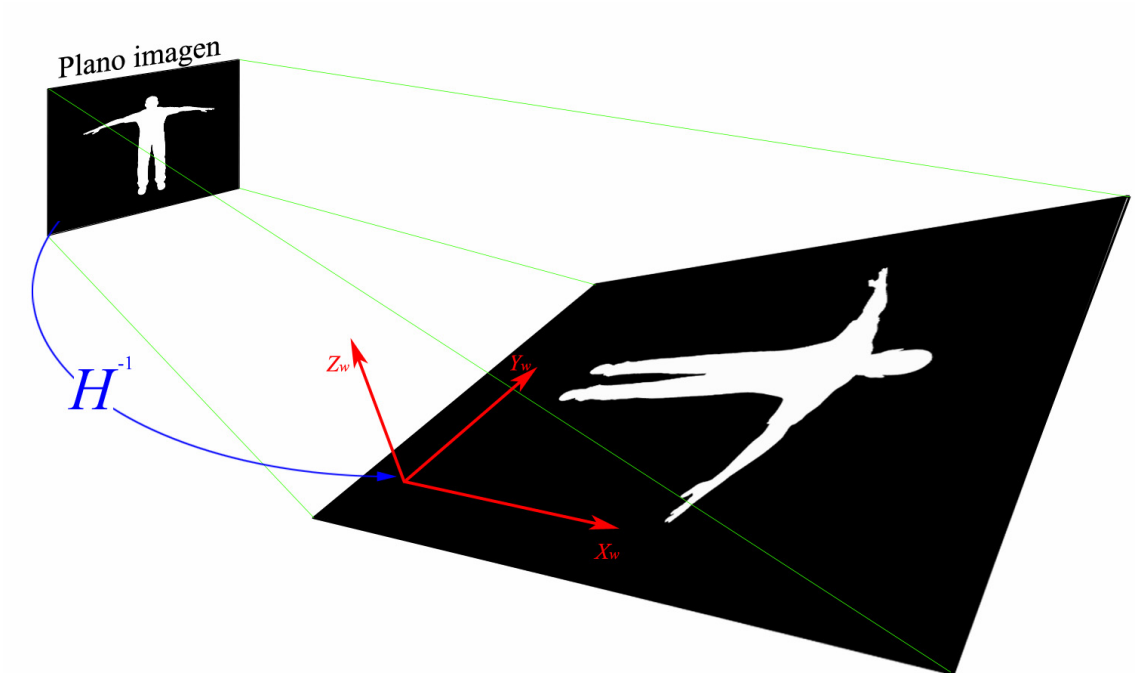


Figura 7.53 Aplicación de homografía inversa

A partir de aquí se aplica lo visto en el cliente sobre la compresión del grid en JPEG.

7.5.3. Envío directo de capturas

Otra de las tareas que realiza el servidor es el envío de capturas al cliente.

Este proceso no necesita sincronización con ningún otro y su único objetivo es informar al usuario final de lo que está sucediendo en el entorno. Es por ello que la tarea es considerada de menor importancia y es delegada a un hilo, figura 7.54. Éste envía las imágenes a una velocidad de 15 fps de forma autónoma.

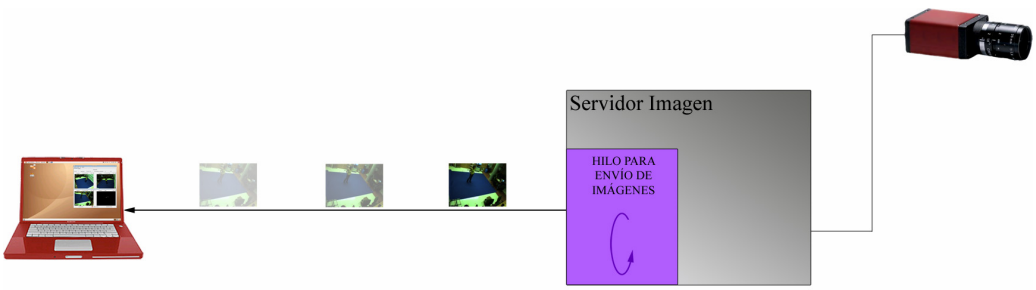


Figura 7.54 Hilo para el envío de imágenes

Con el fin de agilizar el proceso, se reducen las dimensiones de las capturas cuatro veces sobre su tamaño original, de 640x480 a 160x120. Y para no saturar la red, se comprimen mediante JPEG.

El hilo no siempre se encuentra corriendo. Es el usuario el que escoge cuándo ver las imágenes, momento en el cual el cliente ordena la activación al servidor.

Durante dicha activación se asigna un puerto por el que mandar las imágenes. Como ya se dijo, este puerto puede ser 1540, 1541, 1542 ó 1543, siendo la única diferencia el número de zona de dibujo receptora.

A continuación, figura 7.55, se presenta el diagrama de flujo del proceso de envío.

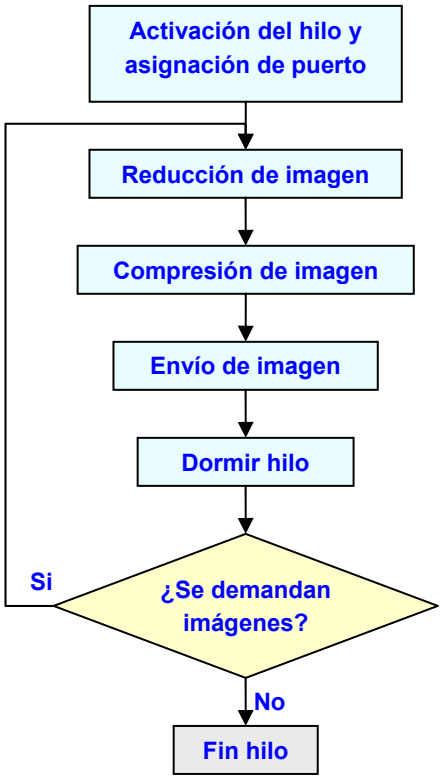


Figura 7.55 Proceso de envío de imágenes

Cabe destacar el paso en el que el hilo duerme. Esto se hace con la intención de ajustar el flujo de imágenes a 15 por segundo. Así se evita la espera activa y la correspondiente perdida de recursos en el servidor.

7.5.4. Grabación de imágenes

Otra funcionalidad ofrecida es la grabación de imágenes. Éste es un proceso extremadamente delicado pues tiene que estar sincronizado en todos los servidores y se debe realizar a la máxima velocidad y calidad posible.

Su procesado en un hilo externo no sirve de ayuda ya que el cliente lo ordena junto a otras tareas quedando a la espera de su confirmación. Es decir, de nada vale estructurar el programa de forma concurrente si éste se tiene que ejecutar secuencialmente.

En un principio, la grabación se realiza almacenando directamente el array a disco. Ahora bien, siendo éste de dimensión 640x480x3. Y siendo el tamaño máximo de 2Gib para un archivo en Linux, entonces el número de fotogramas guardados no puede superar aproximadamente los 2300, que a una velocidad de 25fps implica menos de dos minutos de grabación.

La solución adoptada utiliza nuevamente JPEG, donde ahora se comprime tal cual la imagen antes de su almacenamiento. La calidad empleada en este caso es del 80%, que es la mayor antes de la subida pronunciada en el tamaño.

Tras la compresión se obtiene un buffer de longitud indeterminada, éste depende de la imagen en concreto. Por lo tanto es necesario un procedimiento que distinga qué datos corresponden a una u otra imagen. El mecanismo implementado es muy sencillo: el primer dato en el fichero es un entero cuyo valor hace referencia al número de bytes de la primera imagen, es decir, si se leen tantos datos como indique el primer entero se tiene la imagen completa. A continuación se coloca otro entero con el número de bytes de la siguiente imagen, y así sucesivamente, figura 7.56.

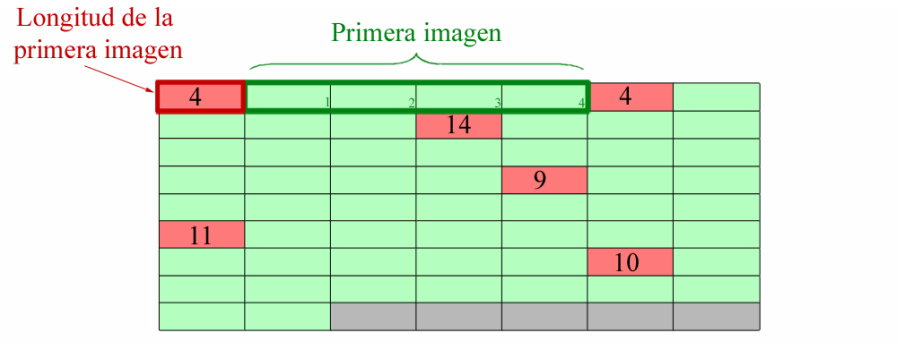


Figura 7.56 Estructura del fichero de imágenes

CAPÍTULO 8. PRUEBAS Y RESULTADOS

8.1. Condiciones iniciales

Las pruebas y resultados que se presentan a continuación han sido realizados en las siguientes condiciones:

- Para asegurar que la superficie de fondo cumple la ley de lambert, se ha superpuesto una moqueta de color azul sobre éste.
- Se han empleado tres cámaras a una altura aproximada del suelo de tres metros formando un triángulo de lados aproximados de: 7,8 4,2 y 8 metros.
- Todos los parámetros de todas las cámaras se han fijado como sigue:

Parámetro		***Control***	***Valor***
Brihtness		Manual	16
Auto exposure		Manual	125
Sharpness		Manual	0
White balance	Blue/U	Manual	97
	Red/V		34
Hue		Off	40
Saturation		Off	256
Gamma		Off	0
Shutter		Manual	2000
Gain		Manual	1

- El ángulo utilizado par el espacio invariante a la iluminación es 1.1693 radianes.
- Las cámaras comparten sus coberturas dando lugar a un plano de dimensiones 3,8 por 2,9 metros.
- Se emplean cuatro ordenadores. Tres de ellos hacen la función de servidor y uno la de cliente. Todos ellos se encuentran conectados mediante red gigabit ethernet.
- Para la conexión con el robot se emplea un punto de acceso con protocolo 802.11g
- La iluminación de la sala proviene únicamente de los fluorescentes de la misma.

8.2. Pruebas relacionadas con los servidores de imagen

8.2.1. Detección de los colores

Debido a que la única información obtenida del entorno viene dada por información luminosa, entonces la detección de objetos queda determinada al color de éstos.

A continuación se muestra la respuesta del sistema ante una serie de colores.

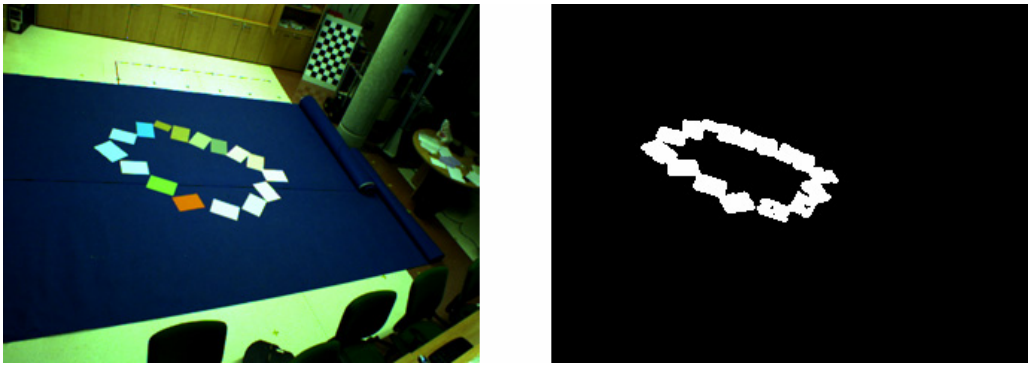


Figura 8.1 Detección según color

No parece que exista problema en la detección de cualquier color. No obstante, cabe destacar que a medida que una superficie se acerca al blanco la detección es peor; pueden existir regiones negras.

8.2.2. Detección de formas

A continuación se muestran una serie de formas. En cierta medida se quiere evaluar la precisión en la detección de los detalles de los objetos.

Figura 8.2 Precisión en la detección de detalles

Los detalles más pequeños no son correctamente detectados. Por ejemplo, la flecha en el centro de la imagen no parece tal tras la detección. Sin embargo, se debe aclarar que ésta mide

de ancho algo menos de 1,5 centímetros y la cámara que la observa se encuentra a algo más de 5 metros. Por otro lado, el paraguas guarda su forma, en la cinta blanca se aprecian los bordes.... En general, en nivel de detalle es satisfactorio.

8.2.3. Discriminación de sombras

Una prueba importante en el presente capítulo es evaluar el grado de discriminación de las sombras. Debe recordarse que dotar al sistema de una detección de objetos invariante a la iluminación ha supuesto un arduo y largo proceso.

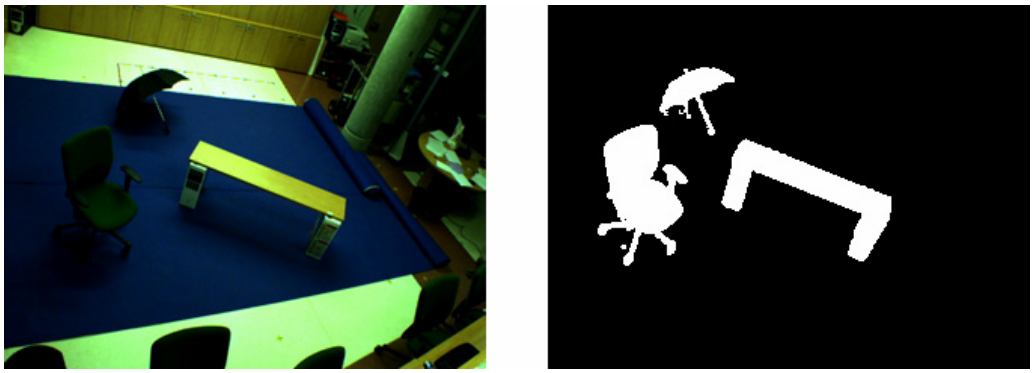


Figura 8.3 Discriminación de sombras

Obsérvese cómo la sombra bajo el paraguas desaparece. O cómo en la silla sólo es detectado su cuerpo. En vista de los resultados se entiende el proceso como exitoso.

8.2.4. Detección de tamaños

En este punto se comprueba la precisión de detección en el tamaño de los objetos. Para ello ya no se presenta una simple segmentación de fondo sino el grid de ocupación.

En la realización de esta prueba se han mostrado al sistema una serie de objetos de los cuales se conoce su tamaño. Tras la detección se han vuelto a medir sabiendo que en el grid un píxel equivale a 12 mm.

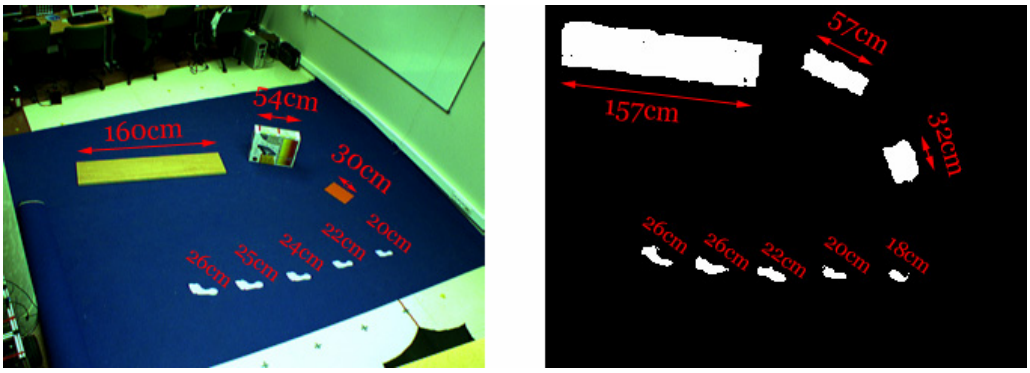


Figura 8.4 Precisión en la detección de tamaño

En la parte inferior de la imagen se han dispuesto una serie de huellas de diferentes tamaños. La intención es comprobar si es o no posible conocer el tamaño del pie. En vista de los resultados se concluye que no es posible; la precisión ronda los 3 centímetros y es insuficiente para esta labor.

8.2.5. Detección de objetos

Por último se muestra al sistema una serie de objetos tridimensionales. De éstos se detecta su corte con el plano de referencia, es decir, la parte de apoyo en el suelo.

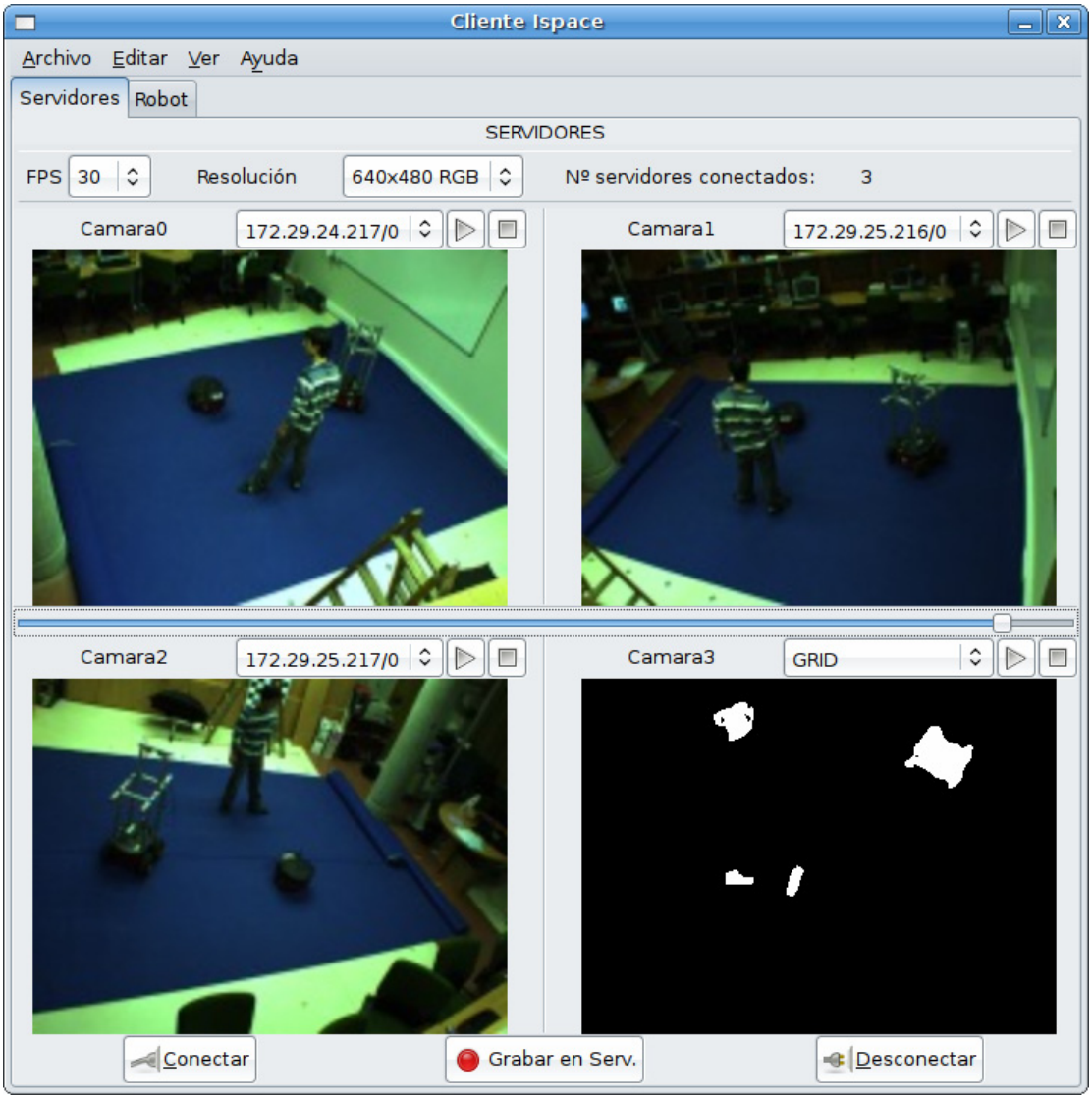


Figura 8.5 Grid de ocupación para el plano del suelo

Por ejemplo, una persona es detectada por dos cúmulos de ocupación correspondientes a los pies apoyados sobre el suelo (parte inferior en el grid de ocupación)

Un ejemplo curioso resulta de aplicar la detección a una silla. Ésta sólo tiene cinco puntos de apoyo en el suelo y esos cinco puntos son los mostrados en la detección

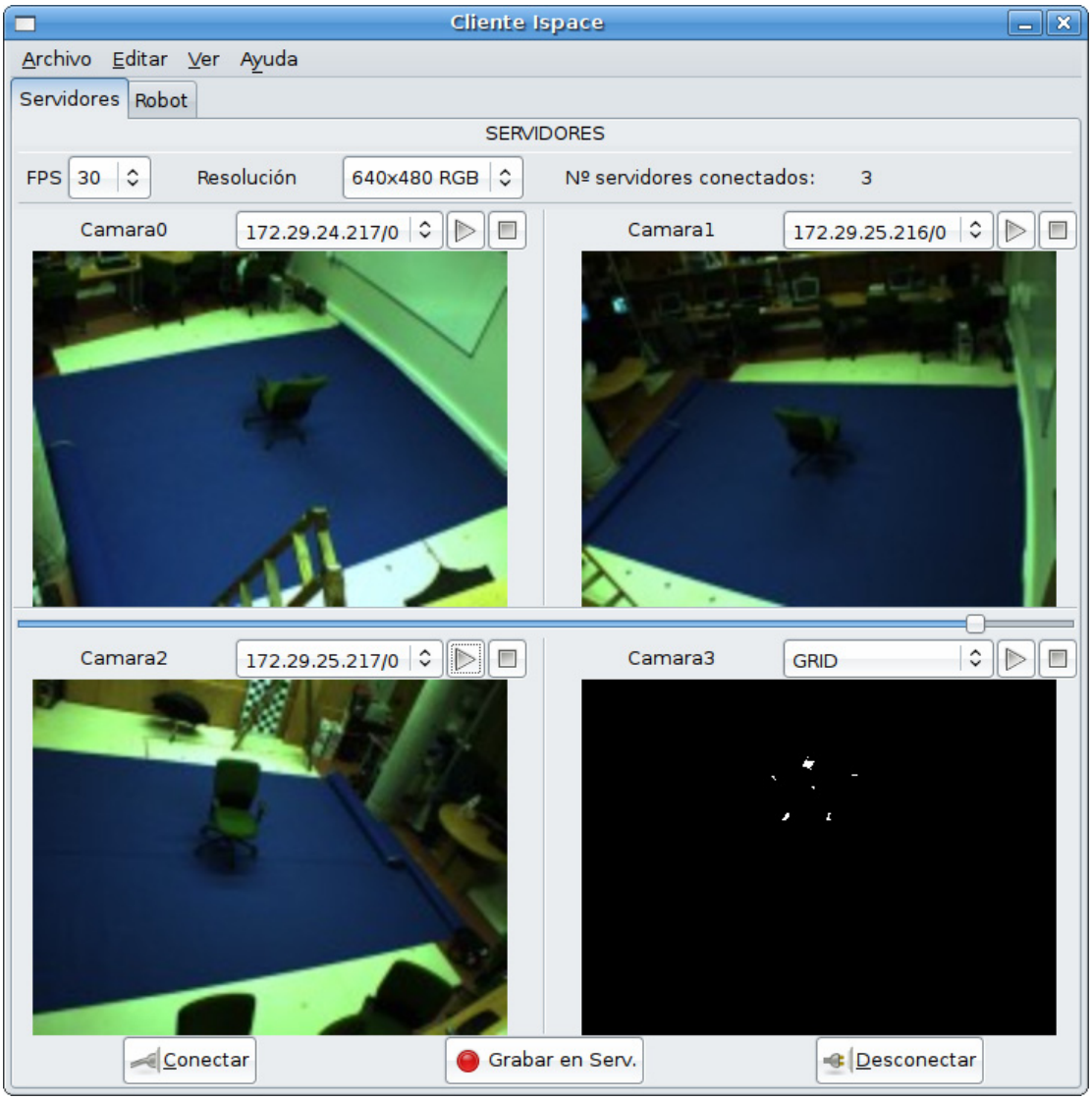


Figura 8.6 Grid de ocupación para una silla

O por ejemplo una escalera

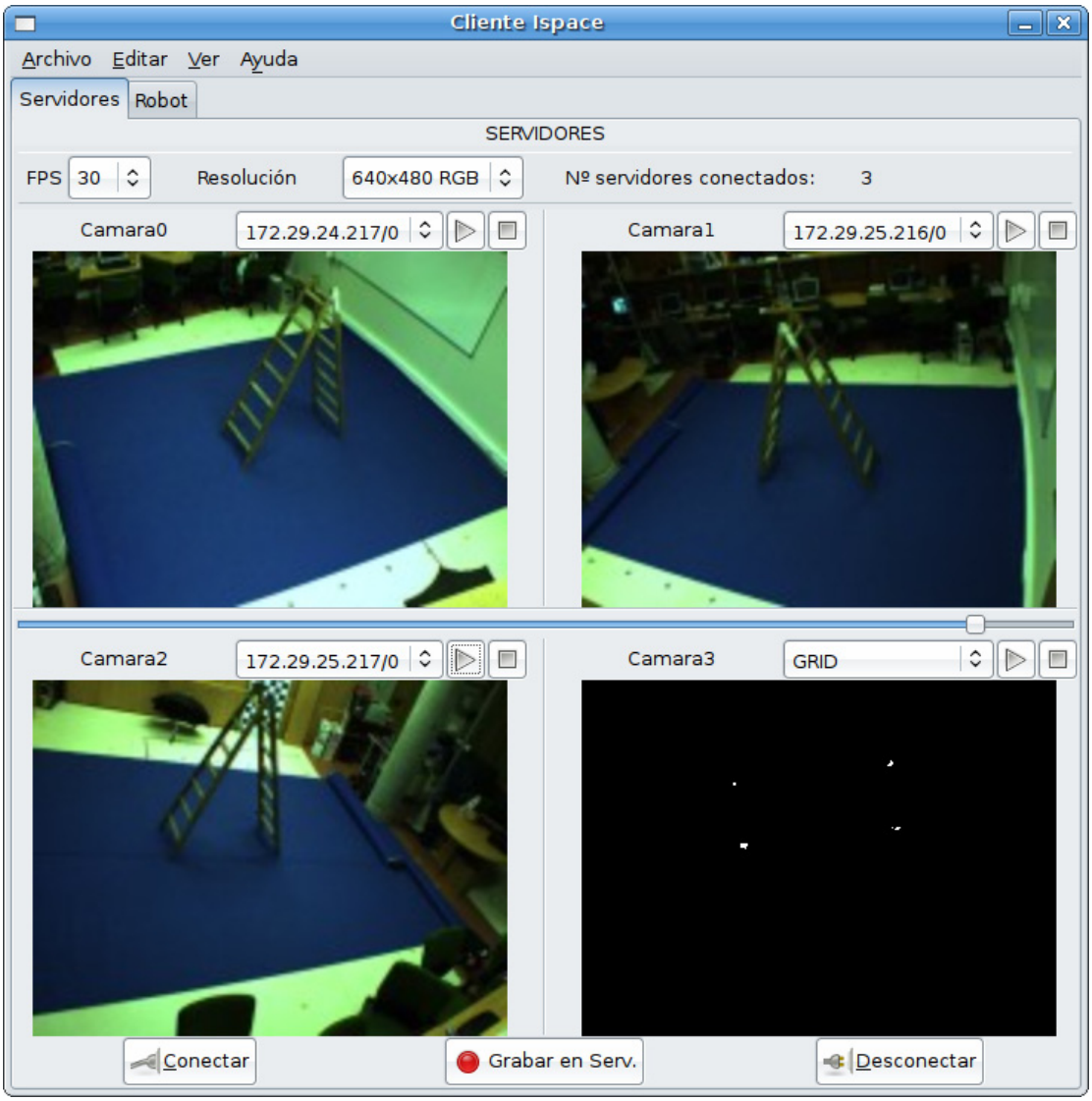


Figura 8.7 Grid de ocupación para una escalera

8.3. Pruebas relacionadas con el servidor robot

8.3.1. Representación de odometría.

La odometría del robot se representa mediante dos formas: en imagen y numéricamente, figura 8.8. En la imagen se representa un punto por cada valor de odometría recibida. Numéricamente se muestra tan solo el valor actual.

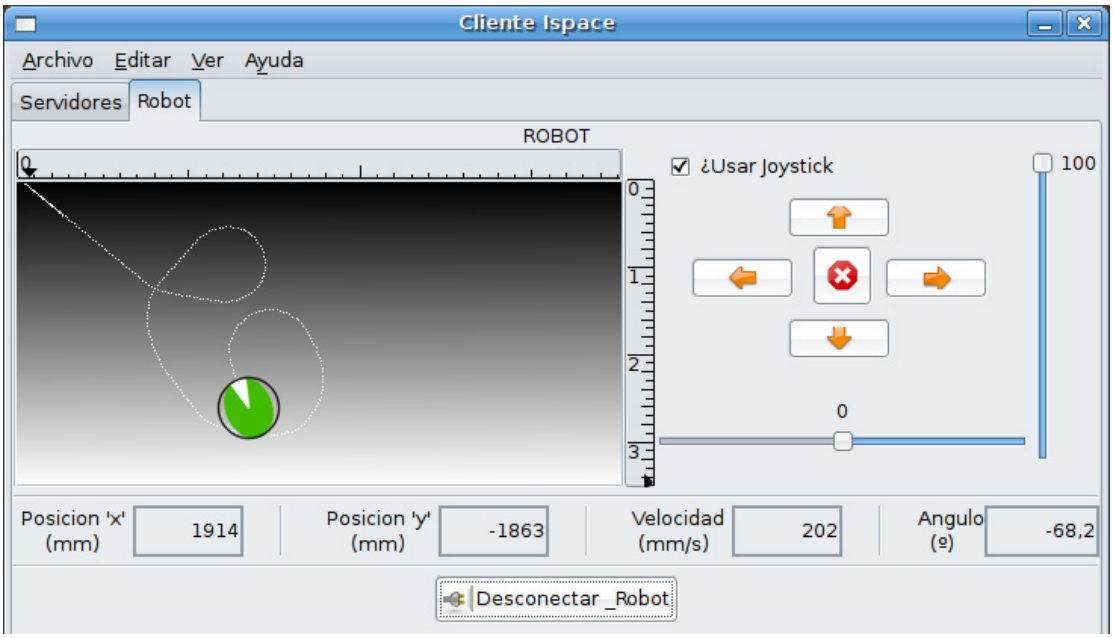


Figura 8.8 Representación de la odometría

8.3.2. Precisión en la odometría

Como es sabido, la odometría es un sistema de posicionamiento que acumula error a medida que aumenta el camino recorrido. Se muestra en la figura 8.9 cómo en un principio el posicionamiento es preciso, pero cómo tras un largo recorrido el robot no está donde cree.

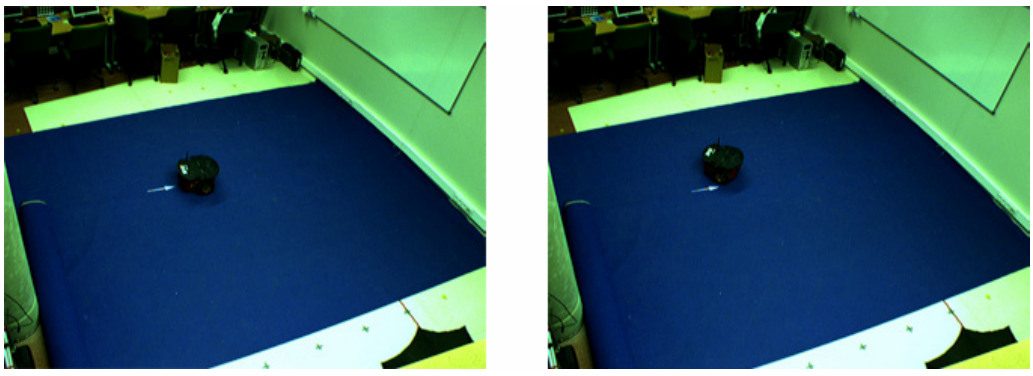


Figura 8.9 Misma posición para odometría tras la navegación del robot

8.4. Pruebas realizadas añadiendo el filtro de partículas

En este punto se añaden métodos probabilísticos para de detección y clasificación de los distintos objetos en la escena. Se trata de un código ya implementado, el cual es añadido en el sistema como una librería más.

8.4.1. Detección de varios objetos

Las pruebas se realizan introduciendo paulatinamente objetos en el entorno.

Un objeto

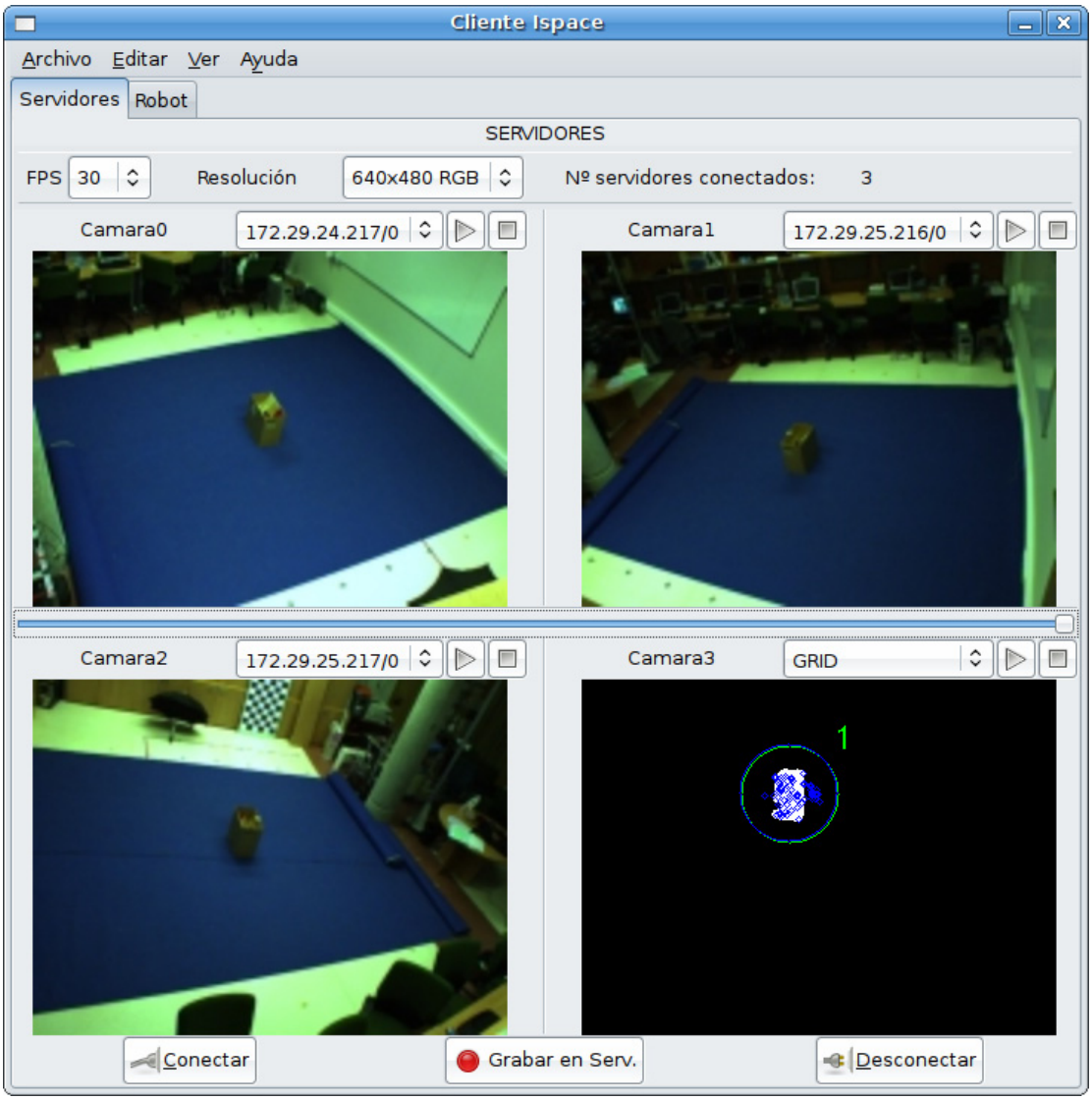


Figura 8.10 Un objeto detectado

El sistema detecta y numera el objeto en escena. Sobre éste, en azul, se representan las partículas asociadas. El número de la parte superior derecha es el número de la clase asociada.

Dos objetos

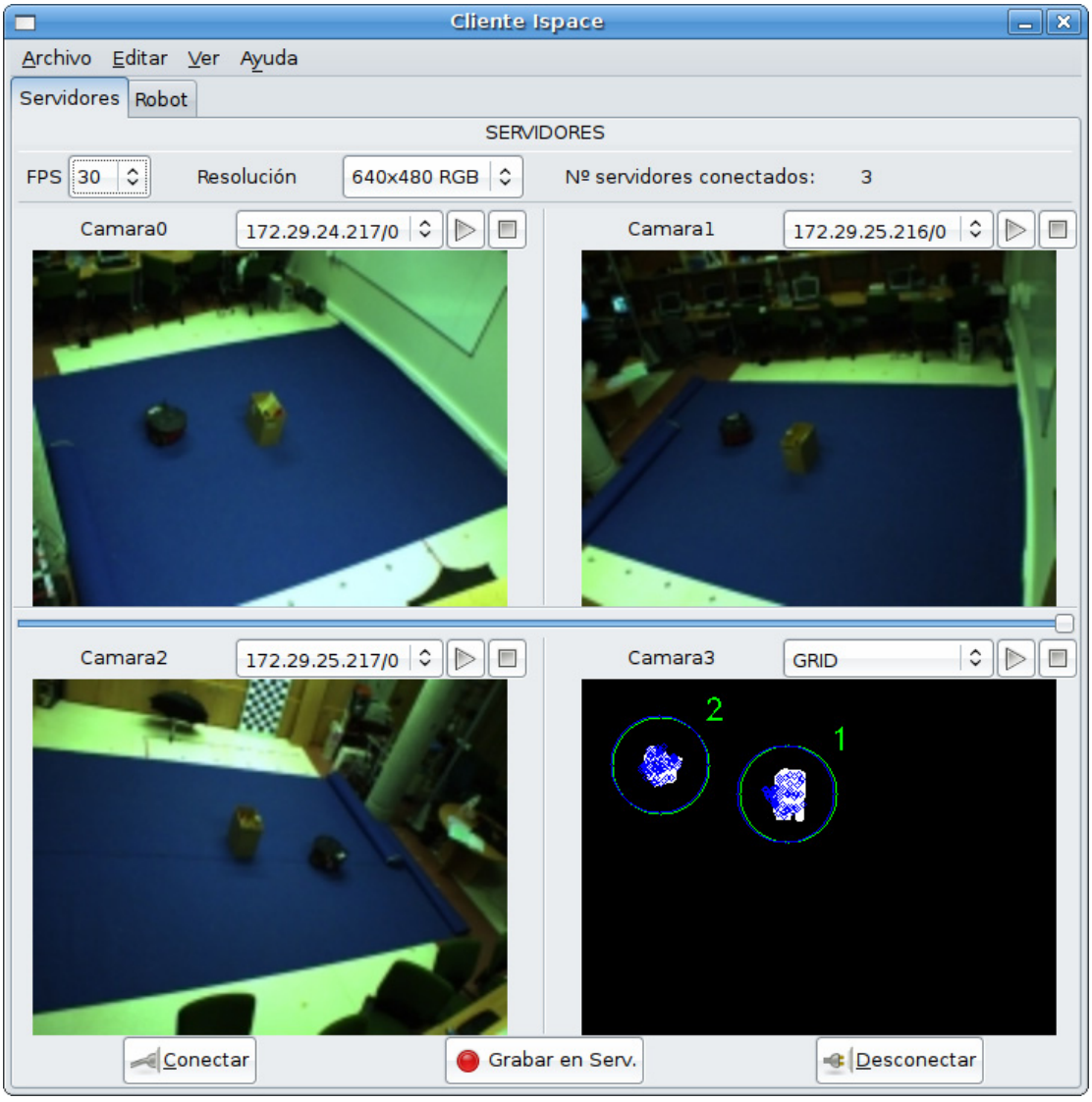


Figura 8.11 Dos objetos detectados

En esta prueba se han incluido dos objetos. Obsérvese cómo a cada uno se asocian partículas de forma independiente.

Tres objetos

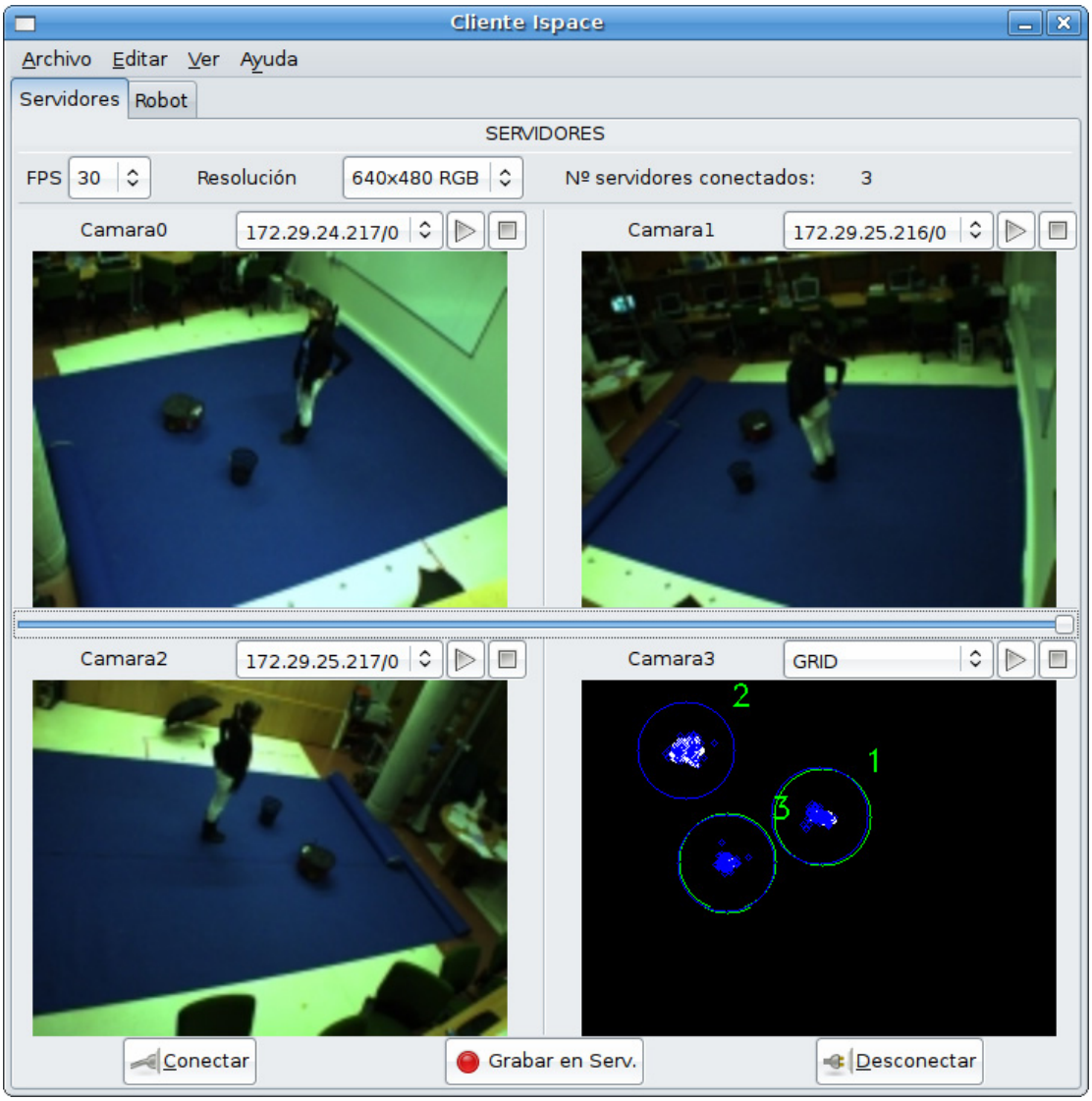


Figura 8.12 Tres objetos detectados

Cuatro objetos

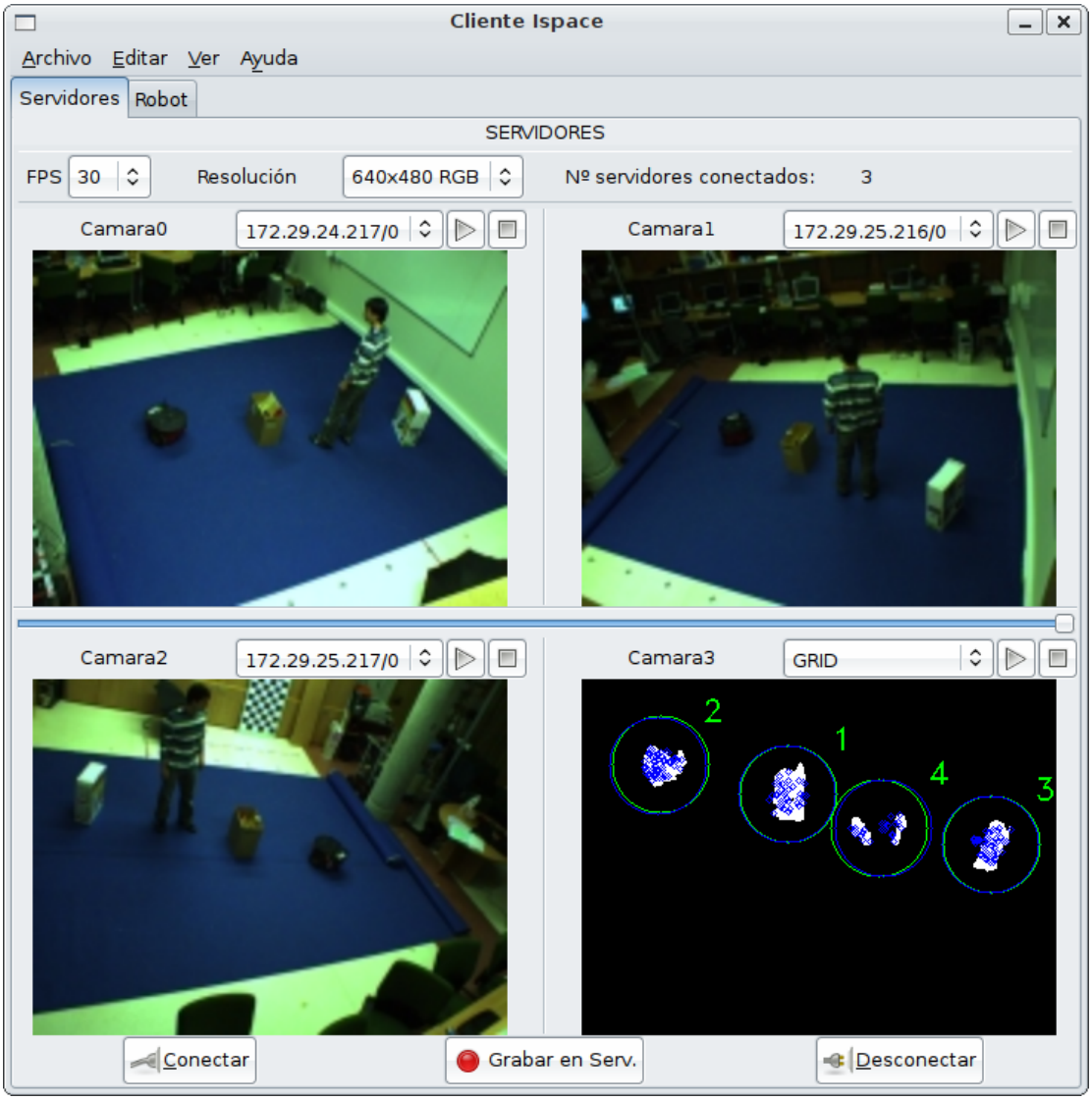


Figura 8.13 Cuatro objetos detectados

En este caso se pide especial atención en el cuarto objeto. Éste es una persona, y como tal, tiene el apoyo sobre el suelo en dos puntos, los pies, por lo tanto aparece en el grid como dos cúmulos de ocupación separados. Sin embargo, se detecta como un único objeto, demostrando la robustez del método probabilístico utilizado.

Siete objetos

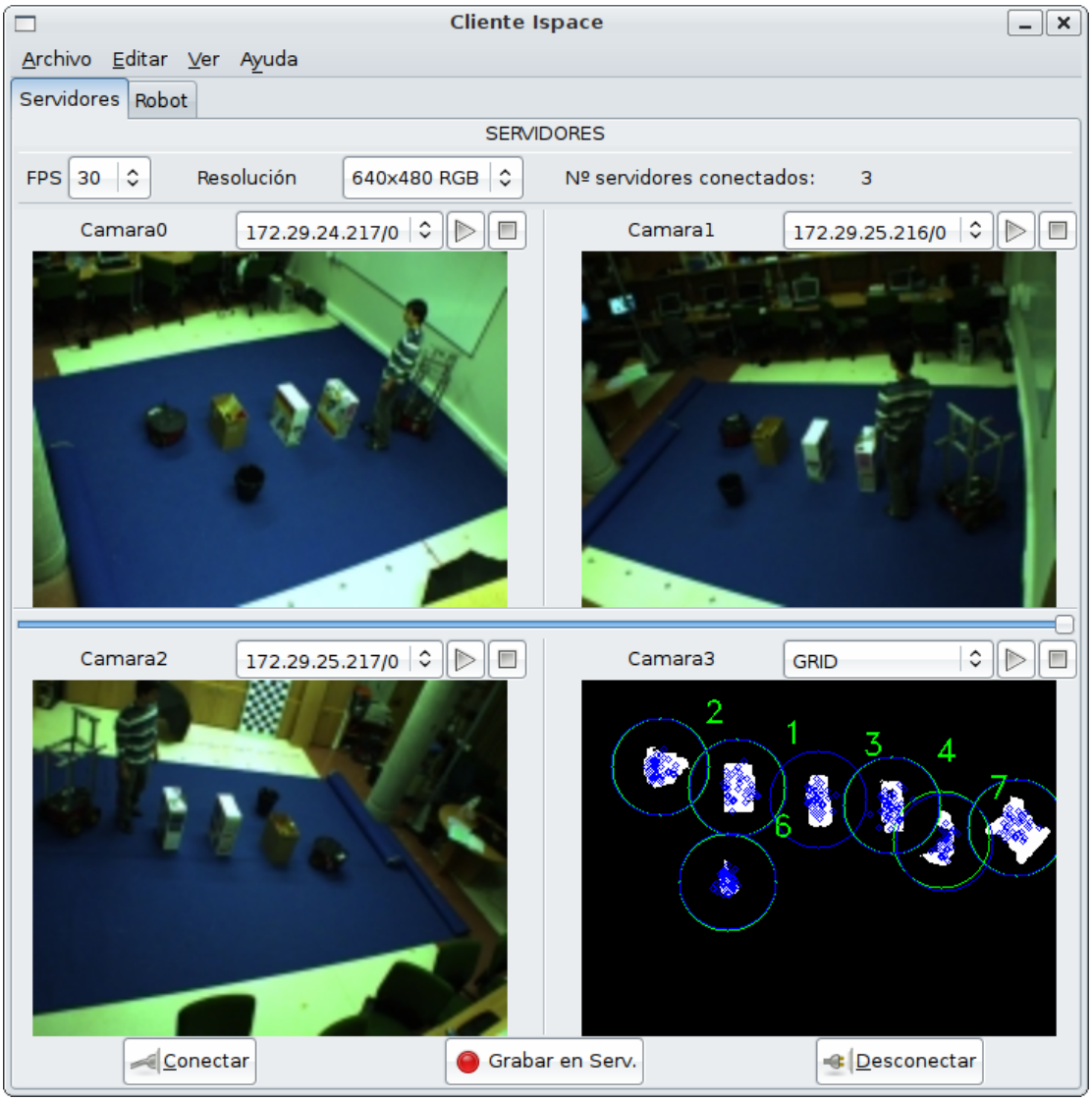


Figura 8.14 Siete objetos detectados

Para finalizar se quiere mostrar cómo el número de objetos en la escena puede llegar a ser muy elevado. En este caso se muestran siete.

CAPÍTULO 9. CONCLUSIONES

Con la realización del proyecto se han cubierto los objetivos marcados en su inicio. Ahora se dispone de un sistema capaz de detectar objetos basado en la visión por computador en espacios inteligentes. Varias conclusiones se desprenden del estudio realizado durante el proceso:

- Al estar implementado el sistema mediante un espacio de color invariante a la iluminación, la detección de objetos no se ve afectada por sus cambios. Esto abre la posibilidad de su utilización en entornos reales sin necesidad de un control sobre la luz.

  No obstante, existe un grave problema, y es que la saturación en el sensor de la cámara conlleva resultados nefastos, ya que no se cumplen las premisas para la aplicación del espacio invariante. Esto manifiesta inevitablemente las limitaciones del sistema y da pie a seguir estudiando soluciones.

  También se quiere resaltar la gran dificultad que ha supuesto "enseñar" a un ordenador a omitir las sombras; pues aunque la solución adoptada no requiere de cálculos complejos sí se fundamenta en varias publicaciones al respecto. Además, el camino seguido ha pasado por numerosas pruebas, todas ellas realizadas sin éxito.

- Gracias al método implementado para la detección de objetos se tiene un sistema capaz de detectarlos independientemente de su estructura, naturaleza y demás propiedades. Únicamente cabe destacar dos aspectos importantes:

  Por un lado, los objetos son detectados gracias a su corte con el plano de referencia, en este caso el suelo. Esto quiere decir que un obstáculo de dimensiones crecientes con la altura puede dar lugar a confusiones. Por ejemplo un cono invertido. Sin embargo, en la mayoría de los casos los obstáculos que se desean detectar son cajas,

columnas, personas, robots… para los que no existe mayor problema. No obstante, en caso que fuese necesario, siempre existe la posibilidad de cambiar el plano de referencia, o incluso fijar varios para obtener información tridimensional de la ocupación, resolviendo finalmente el problema.

Por otro lado, como el sistema sólo recoge información luminosa del entorno, entonces se puede presentar el problema de la no detección de objetos debido al color de los mismos. Esto afecta a todos aquellos obstáculos que se encuentren constituidos por el mismo material que el fondo. Por ejemplo, no se va a detectar una caja de papel blanco sobre un fondo de papel blanco. Éste es un problema que en el presente proyecto se debe asumir, ya que los sensores no proporcionan mayor información.

- La correcta calibración de las cámaras es un proceso crucial para la posterior detección. De otra manera la información obtenida en los distintos sensores no "encaja", generando como consecuencia un mapa erróneo de ocupación.

  Igualmente, de nada sirve una buena calibración si las cámaras no quedan perfectamente fijadas; punto al cual se quiere dar especial énfasis ya que la consecuencia de un leve movimiento en cualquiera de los sensores es una baja precisión de los resultados.

  Para hacerse una idea supóngase una cámara a 3 metros de altura con un ángulo de 45º respecto a la vertical, donde el punto central del plano imagen corresponde al punto situado a 3 metros de distancia sobre la horizontal, figura 9.1 izquierda. Si se calibra la cámara en dichas condiciones y accidentalmente se mueve únicamente un grado, entonces el punto central, que se piensa enfoca a 3 metros, en realidad se encuentra a 2,897 metros, figura 9.1 derecha. Es decir, se está cometiendo un error de 10,3 centímetros por haber movido un grado la inclinación.

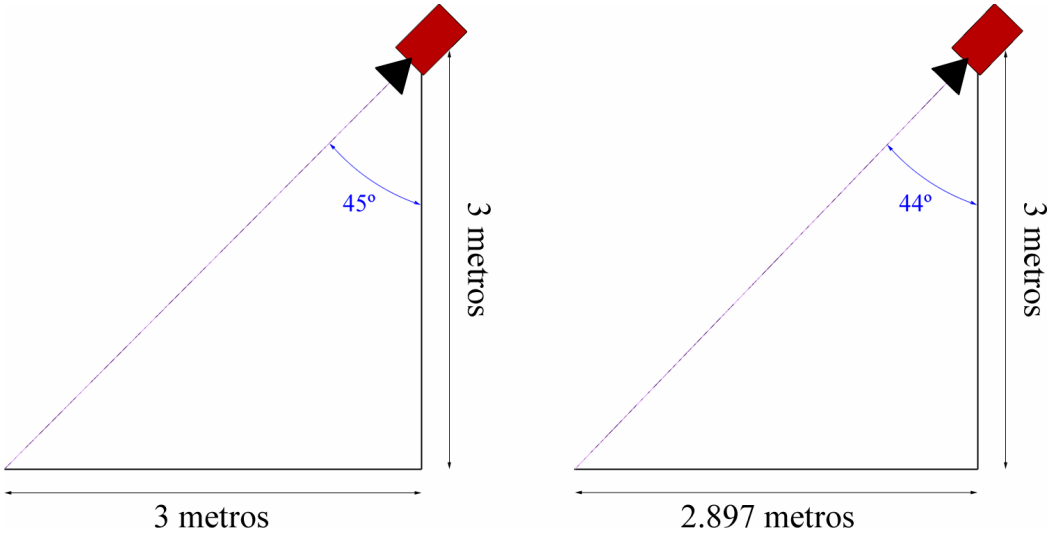


Figura 9.1 Ejemplo del error cometido al variar un grado la inclinación de la cámara

Generalmente, los movimientos sobre las cámaras son inevitables, lo cual hace necesario un método automático de calibración con el que poderse adaptar a las condiciones reales de funcionamiento; ya sea viento, temblores, etc.

- La colocación de las cámaras es algo que ha venido impuesto durante la realización del proyecto. No obstante, sí se destacan varias conclusiones.

  Para empezar, ya se ha comentado cómo el movimiento indeseado de las cámaras puede desembocar en resultados erróneos. Lo que no se ha comentado es que el porcentaje de dicho error depende de la colocación de las cámaras. No es lo mismo hacer un movimiento de un grado en un sensor colocado a 10º de inclinación que en otro a 70º; al igual ocurre con la altura a la que están colocadas las cámaras. No obstante, estos parámetros suelen venir fijados por las dimensiones de la zona a cubrir por lo que, aunque se deben de tener en cuenta, no admiten gran variación.

  Por otro lado, la colocación de las cámaras puede conseguir que la detección de objetos sea más sencilla. Por ejemplo supóngase un array de cámaras calibradas muy juntas, en este caso la información que obtiene de cada una es prácticamente igual. Sin embargo, si se colocan considerablemente separadas, entonces la información proporcionada es mucho mayor, y en la detección se consiguen mejores resultados.

- También se debe dejar constancia de cómo la odometría es un sistema de posicionamiento que acumula un elevado error a medida que aumenta el recorrido. Un método más robusto se debe poder implementar fusionando la información obtenida a través de la visión artificial y la odometría.

- Para la detección de una serie de objetos sobre un espacio acotado se han empleado, al menos, tres ordenadores operando en paralelo. Esto da una idea de la dificultad de exportar las habilidades de los humanos, y como en ocasiones se requiere de gran capacidad de cómputo para realizar tareas que en las personas no suponen esfuerzo alguno.

III. MANUAL DE USUARIO

III.1. Introducción

Este capítulo está dedicado a la puesta en marcha del sistema. Para ello se parte de una red configurada de ordenadores y de un array calibrado de cámaras. El número de éstas depende de la calidad necesaria para la aplicación; a mayor número de cámaras, en principio, mejor detección de objetos.

Se explican aspectos como la compilación de los distintos subsistemas, la configuración, la preparación del robot, el joystick, y cómo resolver los problemas más comunes.

III.2. Compilación

III.2.1. Compilación del cliente

Para la compilación del cliente son necesarios una serie de paquetes que, dependiendo de la distribución con la que se trabaje, están o no incluidos. Por ejemplo, en la versión live 8.04 de Ubuntu se deben instalar los siguientes: autoconf, automake, libglib2.0-dev, libgtk2.0-dev, libraw1394-dev, libdc1394-dev, libcv-dev y libhighgui-dev.

A continuación se localiza y copia el directorio con las fuentes a su destino final y se ejecuta "*autogen.sh*". Éste se encarga de generar los archivos necesarios para la compilación.

También se debe especificar qué archivos van a ser utilizados en la compilación. Al haberse realizado el diseño con la aplicación "*Glade*", esto se indica añadiendo el nombre de todas las fuentes en la línea "*proyecto2_SOURCES*" del archivo "*Makefile.in*" dentro de la carpeta "*/src*". El aspecto una vez añadido es el siguiente:

```
proyecto2_SOURCES = ispace.h main.c support.c support.h interface.c interface.h
    inicializacion.c inicializacion.h callbacks.c callbacks.h red.c utils.c localiza.c captura.c
    captura.h edicion_servidores.c edicion_servidores.h estructuras.h ficheros.c ficheros.h
    constantes.h grabacion.c grabacion.h robot.c robot.hrobot_alto_nivel.c robot_alto_nivel.h
    contorno.c contorno.h hilos.c hilos.h timeouts.c timeouts.h joystick.c joystick.h odometria.c
    odometria.h color.c color.h miXpfcp.h misMaths.h miXpfcp.c miCluster.c misMaths.c
    f_particulas.h f_particulas.c modifica_interfaz.c registro_eventos.c estados.c.
```

Y también se debe añadir, en este mismo archivo en la línea "p*royecto2_OBJECTS*", lo siguiente:

```
proyecto2_OBJECTS = main.o support.o interface.o callbacks.o inicializacion.o red.o utils.o
    localiza.o captura.o edicion_servidores.o ficheros.o grabacion.o robot.o robot_alto_nivel.o
    contorno.o hilos.o timeouts.o joystick.o odometria.o color.o miXpfcp.o misMaths.o
    miCluster.o f_particulas.o modifica_interfaz.o registro_eventos.o estados.o
```

El último paso previo a la compilación es incluir las liberarías que el cliente necesita. La inserción se realiza en el archivo "*config.status*" en la línea que comienza "*@PACKAGE_LIBS@*". Al final de ésta se teclea: `pkg-config --libs opencv` y `pkg-config --libs gthread-2.0`.

Ahora sí, es momento de la compilación, para ello se manda la orden "*make*".

III.2.2. Compilación del servidor de imágenes

La metodología es idéntica a la descrita en el caso del cliente. No obstante se debe tener en cuenta que las fuentes a añadir no son las mismas. En este caso:

```
gladeservi0_SOURCES = ispaced.h main.c support.c support.h interface.c interface.h
    callbacks.c callbacks.h red.c visualiza.c captura.c calibra.c utils.c menu.c localiza.c
    ispaced.c contorno.h contorno.c
```

```
gladeservi0_OBJECTS = $(am_gladeservi0_OBJECTS) ispaced.o localiza.o menu.o utils.o
    red.o captura.o calibra.o visualiza.o contorno.o
```

Además, es necesario especificar la matriz de homografía antes de la compilación. La variable que recoge la homografía es la llamada "*H_0[]*" en la función "*genera_manda_grid()*" dentro del archivo "*contorno.c*". Es decir, el valor de dicha variable se debe sustituir con la nueva matriz de homografía calculada.

Teniendo lo anterior en cuenta la compilación se realiza de forma idéntica al cliente.

III.3. Lanzar servidores de imágenes

El ejecutable generado al compilar el servidor toma el nombre de "*gladeservi0*" y se encuentra dentro de la carpeta "*/src*". El archivo se ejecuta dentro de las máquinas que se pretende actúen como servidores, con especial cuidado en tener permisos para acceder a las cámaras. Como confirmación del éxito aparecen en pantalla una serie de mensajes referentes a los sockets, en concreto seis, tres de creación y tres de enlazado, figura III.1.

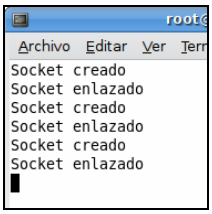


Figura III.1 Ejecución del servidor

En este punto, los servidores se encuentran a la espera de conexiones, siendo el cliente el que debe tomar la iniciativa.

III.4. Lanzar y configurar cliente

El ejecutable generado al compilar el cliente toma el nombre de "*proyecto2*" y se encuentra dentro de la carpeta "*/src*".

Configuración de cámaras

Una vez ejecutado es momento de su configuración. Por defecto existe una resolución y fps prefijados. No obstante, estos parámetros se modifican a través de los dos desplegables ofrecidos, figura III.2.

Figura III.2 Configuración de fps y resolución

Configuración de servidores de imágenes

Los servidores a los que se tiene intención de conectar se gestionan en la entrada "*Servidores*" dentro del menú "*Editar*", figura III.3.

Figura III.3 Ventana de gestión de servidores

Los servidores se añaden tecleando su dirección en el cuadro de texto y pulsando en botón "*Añadir*". Se borran pulsando en la "*X*" de su lado derecho, y se seleccionan para la conexión añadiendo la "*V*" de su lado izquierdo.

Configuración del joystick

En caso de que se utilice joystick hay que configurarlo en la entrada "*Joystick*" del menú "*Editar*", figura III.4.

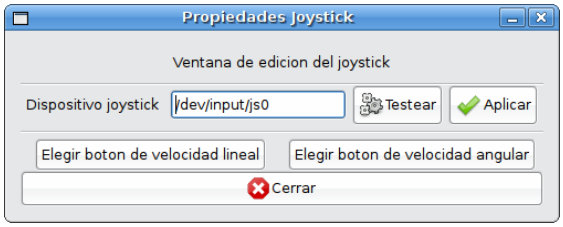


Figura III.4 Ventana de configuración del joystick

En el cuadro de texto se debe introducir la ruta absoluta del dispositivo tal y como lo gestiona el sistema operativo

Si se desean configurar los botones de velocidad lineal y angular, se debe pulsar sobre "*Elegir botón de velocidad lineal*" y/o "*Elegir botón de velocidad angular*" respectivamente, y seguir las instrucciones.

Configuración del servidor robot

Lo referente al robot es gestionado en la entrada "*Robot*" del menú "*Editar*", figura III.5.

Figura III.5 Ventana de gestión del robot

Se introduce la dirección IP del robot, su velocidad máxima y el incremento en las flechas de control. Es importante aplicar los datos una vez modificados, de otro modo no son efectivos.

III.5. Configuración del servidor robot

El robot actúa como otro servidor más, recibiendo órdenes desde el cliente y actuando en consecuencia.

Existe un programa encargado de realizar las funciones de servidor y control del robot, su nombre es "*robot_server*" y se encuentra en la carpeta "/*seigyo*".

Tras su ejecución el robot queda a la espera de conexiones, de tal manera que el cliente es el que decide cuándo conectarse a él y cómo gestionarlo.

Un aspecto de gran importancia que se debe tener siempre presente es la incapacidad de controlar el robot en caso de pérdida de señal. Esto quiere decir que si por ejemplo la conexión inalámbrica no cumple con la especificación de señal ruido mínima, entonces el robot puede no interpretar las nuevas órdenes y actúa en consecuencia a la última recibida correctamente.

III.6. Conexión

Una vez listos los distintos nodos del sistema es momento de la conexión. Todas las conexiones son gestionadas por el cliente, y por lo tanto también lo es su inicio.

La conexión referente a los servidores de imágenes comienza tras presionar el botón "*Conectar*", figura III.6.

Figura III.6 Inicio de conexión con los servidores de imágenes

Acción que se confirma mediante un mensaje que informa del número de servidores conectados, figura III.7.

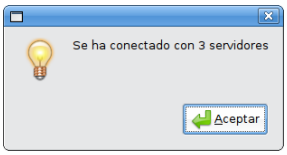


Figura III.7 Mensaje de confirmación de conexión con servidores de imágenes

Tras la conexión, se debe aguardar a que al menos sean tomadas cien capturas. En este periodo se caracteriza estadísticamente el fondo con el fin de conseguir posteriormente una mejor detección.

Una vez esperado el tiempo necesario se puede comenzar a visualizar los datos de los servidores de imágenes. Éstos son las imágenes en directo y el grid de ocupación, seleccionables desde los desplegables de las zonas de dibujo, figura III.8.

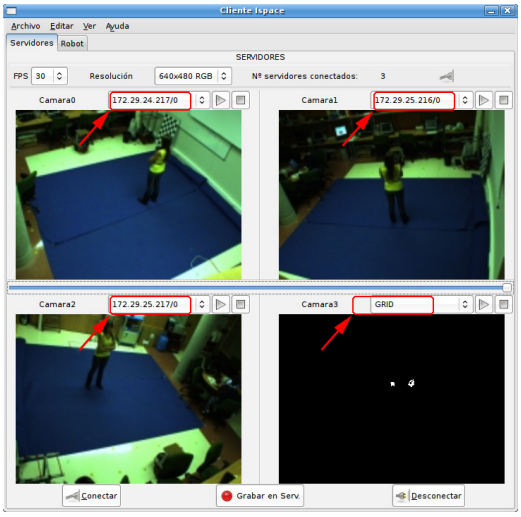


Figura III.8 Selección de la información a visualizar por parte de los servidores de imágenes

Cambiando a la pestaña del robot se puede conectar a éste y utilizar los controles para moverlo. Por ejemplo, en lo figura III.9 se ha decidido usar el joystick. No obstante, también se permite la posibilidad de emplear las flechas de movimiento, las barras y/o el teclado.

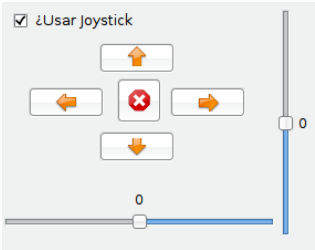


Figura III.9 Controles del robot integrados en el propio interfaz gráfico. El joystick se encuentra activado

En esta misma pestaña se muestra la odometría del robot en forma de imagen, figura III.10.

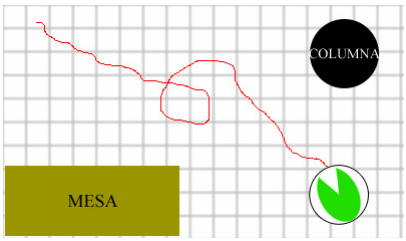


Figura III.10 Odometría del robot representada en forma de imagen

Así como también se representa de forma numérica, figura III.11.

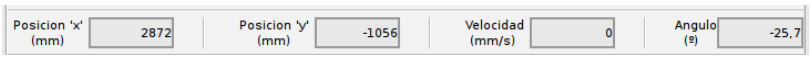


Figura III.11 Representación numérica de la odometría

III.7. Problemas más comunes

El joystick no funciona.

- Comprobar que se tienen permisos para utilizar el dispositivo (suele pertenecer al grupo plugdev)
- Comprobar que el botón de analógico esté encendido

El cliente se queda colgado al recibir las imágenes de los servidores.

- Comprobar que los servidores de imágenes los ha ejecutado un usuario con suficientes permisos.
- Comprobar que las cámaras están bien conectadas. Es recomendable también comprobar el funcionamiento de las mismas mediante un programa que las gestione, por ejemplo "*coriander*".
- Comprobar las conexiones de los cables de red.

En el grid no se muestra nada.

- Comprobar que ninguna cámara se encuentre tapada.
- Se han podido mover las cámaras. Es posible que se necesite una nueva calibración.

El grid es poco preciso o no muestra la realidad.

- Comprobar que el fondo cumple la ley de Lambert.
- Comprobar que las condiciones del entorno no saturen las cámaras
- Comprobar que los parámetros de las cámaras se encuentran en modo manual y no se han modificado durante la conexión.
- Si se ha añadido algún cambio sobre el fondo se debe volver a entrenar el sistema. Cierre el programa servidor en cada servidor de imágenes (Ctrl+C) y ejecútelo nuevamente.

Al iniciar el servidor se devuelve un mensaje de error indicando que la dirección ya está en uso.

- Esto es debido a que el servidor finalizó de forma incorrecta. La conexión se libera automáticamente al cabo de unos minutos. Intente volver a lanzarlo pasado ese tiempo.

IV. PLIEGO DE CONDICIONES

En el funcionamiento del sistema final se ven implicados tanto recursos hardware como software. En su última versión se emplean los numerados a continuación:

IV.1. Hardware

- Router gigabit ethernet D-link.
- Cuatro ordenadores de sobremesa equipados con puerto firewire y gigabit ethernet.
- Portátil Dell Latitude D610 con procesador Intel pentium M 1.8Ghz y 1Gb de RAM.
- Punto de acceso inalámbrico.
- Cuatro cámaras Sony Marlin V392 IEEE1394.
- Robot Pioneer P3-DX.
- Mando analógico de playstation y adaptador a PC.

IV.2. Software

- Sistema operativo Ubuntu en sus versiones 5.04 y 7.10.
- Librerías para control del interfaz IEEE1394.
- Librerías OpenCV.

V. PRESUPUESTO

En esta sección se estima el importe de la ejecución del proyecto. Para ello se realiza un estudio agrupando los gastos en función de su origen.

V.1. Costes de ejecución material

Los costes de ejecución incluyen tres elementos

- Costes de equipos.
- Costes del software.
- Costes de mano de obra por tiempo empleado.

V.1.1. Costes de equipos

CONCEPTO	PRECIO UNITARIO	CANTIDAD	SUBTOTAL
Ordenador portátil	650 €	1	650 €
Ordenador sobremesa	600 €	4	2.400 €
Cámara Marlin	800 €	4	3.200 €
Router gigabit	160 €	1	160 €
Punto de acceso	50 €	1	50 €
Robot Pioneer P3-DX	6.000 €	1	6.000 €
Subtotal			**12.460 €**

V.1.2. Costes de software

Los distintos equipos que han sido utilizados en este trabajo son los siguientes:

CONCEPTO	PRECIO UNITARIO	CANTIDAD	SUBTOTAL
Sistema operativo Ubuntu	0 €	5	0 €
Coriander 1.0.1	0 €	1	0 €
Matlab 7.0	2.500 €	1	2.500 €
Glade	0 €	1	0 €
Microsoft Office XP	200 €	1	200 €
Photoshop CS 3	1.000 €	1	1.000 €
Microsoft Windows XP	135 €	1	135 €
Subtotal			**3.835 €**

V.1.3. Costes por tiempo empleado

FUNCIÓN	PRECIO UNITARIO	Nº HORAS	SUBTOTAL
Ingeniería	25 €	1.440	36.000 €
Mecanografiado	15 €	240	3.600 €
Subtotal			**39.600 €**

El número de horas de ingeniería corresponde a un trabajo de seis horas diarias durante un año. El mecanografiado se ha estimado como seis horas diarias durante dos meses.

V.1.4. Coste total del presupuesto de ejecución material

CONCEPTO	SUBTOTAL
Costes de equipos	12.460 €
Costes de software	3.835 €
Costes por tiempo empleado	39.600 €
Subtotal	**55.895 €**

V.2. Gastos generales y beneficio industrial

Se incluyen los gastos derivados de la utilización de las instalaciones de trabajo y el beneficio industrial. Se estima como el 16% del coste de ejecución material

CONCEPTO	SUBTOTAL
Gastos generales y beneficio industrial	8.943,2 €
Subtotal	**8.943,2 €**

V.3. Importe total del presupuesto

CONCEPTO	SUBTOTAL
Coste total del presupuesto de ejecución material	55.895,0 €
Gastos generales y beneficio industrial	8.943,2 €
Subtotal	64.838,2 €
IVA 16%	10.374,1 €
TOTAL IVA INCLUIDO	**75.212,3 €**

El importe total del proyecto suma la cantidad de:

Setenta y cinco mil doscientos doce euros con treinta céntimos

Alcalá de Henares a 22 de Diciembre de 2008

Fdo: Víctor Espejo Gómez

Ingeniero de Telecomunicación

VI. BIBLIOGRAFÍA

[Pizarro et al, 2005] Daniel Pizarro, Enrique Santiso, and Manuel Mazo. "Simultaneous localization and structure reconstruction of mobile robots with external cameras", 2005.

[Weiser, 1999] M. Weiser. "The computer for the 21st century", 1999.

[Weiser, 1993b] M. Weiser. "Some Computer Science Problems in Ubiquitous Computing", 1993b.

[Weiser, 1993a] M. Weiser. "PARC builds a world saturated with computation", 1993a.

[Coen, 1998] M.H. Coen. "Design principles for intelligent environments", 1998.

[Look et al., 2005] G. Look, B. Kottahachchi, R. Laddaga and H. Shrobe. "A Location Representation for Generating Descriptive Walking Directions", 2005.

[Look and Shrobe, 2007] G. Look and H. Shrobe. "Towards Intelligent Mapping Applications: A Study of Elements Found in Cognitive Maps", 2007.

[Pentland, 1996] A. Pentland. "Smart Rooms", 1996.

[Ara et al., 2008] K. Ara, N. Kanehira, D. O. Olguin, B. Ñ. Waber, T. Kim, A. Mohan, P. Gloor, R. Laubacher, D. Oster, A. S. Pentland and K. Yano. "Sensible organizations: Changing our businesses and work styles through sensor data", 2008.

[T.Sogo et al., 1999] T.Sogo, H. Ishiguro and T. Ishida. "Acquisition of Qualitative Spatial Representation by Visual Observation", 1999.

[Hashimoto et al., 2003] H. Hashimoto, J.-H. Lee and N. Ando. "Self-identification of distributed intelligent networked device in intelligent space", 2003.

[Lee et al., 2005] J.-H. Lee, K. Morioka and H. Hashimoto. "Intelligent Space and Mobile Robots", 2005.

[Steinhaus et al., 1999] P. Steinhaus, M. Ehrenmann and R. Dillmann. "A Modular and Extensible Path Planning System Using Observation", 1999.

[Steinhaus et al., 2007] P. Steinhaus, M. Strand and R. Dillmann. "Autonomous robot navigation in human-centered environments based on 3d data fusion", 2007.

[Villadangos et al., 2005] J. M. Villadangos, J. Urena, M. Mazo, A. Hernandez, F. J. Alvarez, J. J. Garcia, C. D. Marziani and D. Alonso. "Improvement of Ultrasonic beacon-based Local Position System Using Multi-Access Techniques", 2005.

[Pizarro et al., 2005] D. Pizarro, E. Santiso and M. Mazo. "Simultaneous Localization and Structure Reconstruction of Mobile Robots with External Cameras", 2005.

[Fernandez et al., 2007] I. Fernandez, M. Mazo, J. Lazaro, D. Pizarro, E. Santiso, P. Martin and C. Losada. "Guidance of a mobile robot using an array of static cameras located in the environment", 2007.

[Hoover and Olsen, 2000] A. Hoover and B. D. Olsen. "Sensor Network Perception for Mobile Robotics", 2000.

[Hoover and Olsen, 1999] A. Hoover and B. D. Olsen. "A Real-Time Occupancy Map from Multiple Video Streams", 1999.

[Kruse and Wahl, 1998] E. Kruse and F. M. Wahl. "Camera-based Observation of Obstacle Motions to Derive Statistical Data for Mobile Robot Motion Planning", 1998.

[Berclaz et al, 2006] J. Berclaz, F. Fleuret, and P. Fua. "Robust people tracking with global trajectory optimization", 2006.

[Checka et al, 2003] N. Checka, K. Wilson, V. Rangarajan, and T. Darrell. "A probabilistic framework for multi-modal multi-person tracking", 2003.

[Focken and Stiefelhagen, 2002] D. Focken and R. Stiefelhagen. "Towards vision-based 3-d people tracking in a smart room", 2002.

[Otsuka and Mukawa, 2004] K. Otsuka and N. Mukawa. "Multiview occlusion analysis for tracking densely populated objects based on 2-d visual angles", 2004.

[Hartley and Zisserman, 2003] Richard Hartley and Andrew Zisserman. "Multiple View Geometry in computer vision". Ed. Cambridge, 2003.

[Zhang, 1999] Zhengyou Zhang. "Flexible camera calibration by viewing plane from unknown orientations", 1999.

[Heikkilä and Silvén, 1997] Janne Heikkilä and Olli Silvén. “A four-step camera calibration procedure with implicit image correction”, 1997.

[Bouguet, 2008] Bouguet, J.Y. (2008). Matlab calibration toolbox. http://www.vision.caltech.edu/bouguetj/calib_doc/

[Pajares y de la Cruz, 2001] Gonzalo Pajares, Jesús M. de la Cruz. “Visión por computador: Imágenes digitales y aplicaciones”. Ed. Ra-ma, 2001.

[Mazo et al, 1996] M. Mazo Quintas, L. Boquete Vázquez, R. Barea Navarro. “Visión artificial”, 1996.

[Brown, 1966] E.A. Brown. “Modern Optics”. Ed. Reinhold Pub. Corp, 2ª edic., 1966.

[Murdoch, 1985] J. B. Murdoch. “Illumination Engineering”. Ed. Macmillan, 1985.

[Finlayson et al, 2002] G.D. Finlayson, S.D. Hordley, and M.S. Drew. “Removing shadows from images”.ECCV 2002: European Conference on Computer Vision, 2002.

[Jähne, 1995] Bernd Jähne. “Digital Image Processing”. Ed Sprimger, Third Edition, 1995.

[Cabello ,2007] María Cabello Aguilar. "Comparativa teórica y empírica de métodos de estimación de la posición de múltiples objetos”, 2007.

[Marrón et al, 2007] Marta Marrón, M.A. Sotelo, J.C. García, J. Broddfelt. “Comparing improved versions of ‘K-Means’ and ‘Subtractive’ clustering in a tracking applications”, Febrero 2007.

[Marrón, 2008] M. Marrón Romera. "Seguimiento de múltiples objetos en entornos interiores muy poblados basado en la combinación de métodos probabilísticos y determinísticos". Tesis Doctoral, Universidad de Alcalá, Octubre 2008.

[Bar-Shalom and Fortmann, 1988] Y. Bar-Shalom, T. E. Fortmann. “Tracking and data association”, Mathematics in Science and Engineering, Vol. 179, Academic Press, ISBN 0-12-079760-7, 1988.