Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática

Curso académico 2007-2008

Proyecto Fin de Carrera

RECONSTRUCCIÓN 3D MEDIANTE UN SISTEMA DE ATENCIÓN VISUAL

Autor: Roberto Calvo Palomino
Tutor: José María Cañas Plaza

Madrid 2008

Una copia de este proyecto, las fuentes del
programa y vídeos de los experimentos están
disponibles en la siguiente dirección:

http://jde.gsyc.es/index.php/Rocapal_visual_attention_3D

A mis padres y hermano,
que siempre han estado a mi lado,
y siempre lo estarán.

A María, que siempre
está a mi lado apoyándome.

Y a mis amigos, por ser
verdaderamente mis amigos.

Gracias.

Agradecimientos

Quiero dar las gracias a todos los miembros del Grupo de Robótica de la Universidad Rey Juan Carlos por su apoyo y colaboración. También quiero dar las gracias a los miembros que componen la comunidad de JDE, que poco a poco va asentándose y ofreciendo resultados interesantes al mundo de investigadores.

De igual modo, quiero dar las gracias a todos los miembros que componen el grupo de investigación LibreSoft, que gracias a ellos cada día se convierte en una experiencia nueva.

Como no podía ser de otra forma, quería agradecer de una manera especial el trabajo, apoyo y dedicación que José María me ha dedicado durante todo este proyecto. Ya son más de cuatro años trabajando juntos en diferentes proyectos y por ello aprovecho esta oportunidad para alabar su trabajo, consideración y excelente trato durante todo este tiempo.

A todos, muchas gracias!

Resumen

La visión computacional y la atención visual son hoy en día uno de los campos más atractivos e interesantes en los que la comunidad de investigación dedica tiempo, esfuerzo y dinero para avanzar. Las cámaras, cual ojos en el ser humano pueden proveen de información sensorial importantísima sobre el entorno a los robots. El problema es que extraer información de una imagen no es algo trivial, ya que hay que distinguir las partes importantes e interesantes de la imagen. Para ello se utilizan sistemas de atención visual que nos ayudan a filtrar lo relevante de lo desapercibido, analizando el flujo de datos de las imágenes lo más rápido posible.

El presente proyecto aborda el desarrollo para un robot móvil de un sistema de atención visual unido a una representación en 3D del entorno. Para ello se realiza un control atentivo, basado en un mapa de saliencia, sobre los objetos relevantes de la imagen. Por medio de un par de cámaras, y analizando las imágenes reales, localizamos esos objetos relevantes en el mundo 3D. El sistema atentivo programado analiza la imagen de una forma similar a como lo hace un ojo humano, fijándose en las cosas que suelen llamar la atención: colores llamativos, bordes, movimiento o conocimiento previo de los objetos en la realidad. Ciertas teorías, argumentan que el sistema visual humano tiene una capacidad de procesamiento limitado y que la atención actúa como un filtro neuronal que selecciona la información debe ser procesada en cada instante.

Se ha puesto énfasis en la velocidad de proceso del sistema. Es necesario que se realice la atención visual sobre las imágenes reales y la representación 3D en el menor tiempo posible, para que el sistema tenga un comportamiento parecido al de un ser humano. A lo largo de la memoria se comentarán los numerosos ajustes que se han llevado a cabo para obtener un rendimiento aceptable.

Para la implementación de este proyecto, se ha utilizado la arquitectura y plataforma software de JDE, programando una serie de esquemas en C y C++. Además se han integrado y mejorado varios esquemas y librerías de esta infraestructura, obteniendo un correcto y eficiente funcionamiento.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En esta capítulo vamos a introducir el contexto del proyecto y la motivación que nos ha llevado a abordarlo. Explicaremos en un contexto general la visión, los robots y la visión en los robots. Y comentaremos en un contexto más concreto los sistemas de atención visual y la obtención de representación 3D.

1.1. Visión artificial

La visión artificial es un campo de la inteligencia artificial cuyo propósito es programar un computador para que "entienda" una escena o ciertas características de una imagen. Esta ciencia posee numerosas aplicaciones en el mundo real, como puede ser en el ámbito de la medicina, la industria, la robótica, etc.

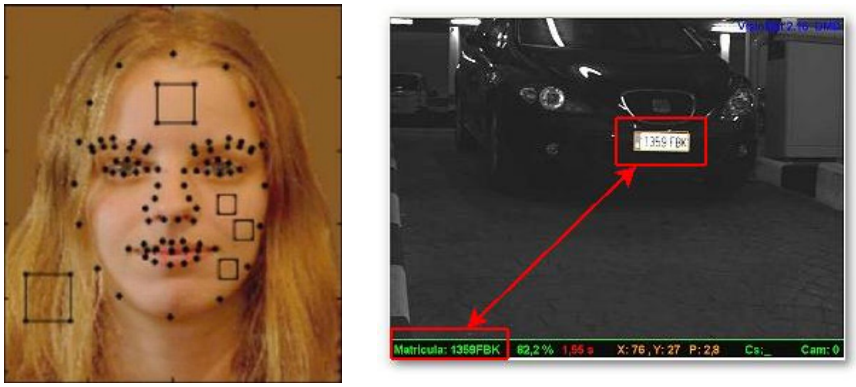
La visión artificial surge en la década de los 60 con la idea básica de conectar una cámara de vídeo a una computadora. Esto implicó no sólo la captura de imágenes a través de la cámara, sino también la compresión de la información de dichas imágenes. En 1961 Larry Roberts, creador de *ARPAnet*, desarrolló un programa en el que el robot podía *ver* una estructura de bloques sobre una mesa, analizar su contenido y reproducirlo desde otra perspectiva, demostrando así que esa información enviada por la cámara había sido procesada por el ordenador.

La visión es, sin duda, nuestro principal sentido. Gracias a ella somos capaces de recibir información acerca del mundo que nos rodea, información que empleamos para desarrollar nuestra inteligencia y para interactuar con el mundo. La visión artificial se ha visto como una forma de aproximarnos al entendimiento de nuestra propia visión, y de ahí a nuestra

propia inteligencia. Conseguir desarrollar sistemas de visión tan buenos como los del ser humano, o más, implica ser capaces de reproducir nuestras capacidades de manera artificial.

Tareas que los humanos realizamos de manera automática, sin dificultad aparente, se han mostrado como problemas gigantescos para nuestras máquinas. Reconocer la cara de una persona es algo automático para nosotros, pero los ordenadores han tardado años en poder hacer esto de manera limitada y menos eficaz que la nuestra.

Hasta comienzos de la década de los 90 no comenzaron a aparecer ordenadores con velocidad de cómputo suficiente para procesar imágenes de forma ágil. A partir de entonces, la visión computacional comenzó a emplearse para múltiples tareas y su desarrollo fue centrándose en problemas concretos como reconocimiento de objetos, seguimiento visual y reconstrucción 3D. Existen numerosas aplicaciones en el reconocimiento de objetos, como puede ser reconocer una cara (1.1(a)) o reconocer una matrícula (1.1(b)).



(a) (b)

Figura 1.1: Reconocimiento de una cara (a) y de una matrícula (b).

La localización y el seguimiento visual se utilizan cada vez más en acontecimientos deportivos. Un par de ejemplos los tenemos tanto en el fútbol americano como en el tenis. El *ojo de halcón*[1] es una técnica que se utiliza desde hace varios años en el tenis profesional. Por medio de 8 cámaras colocadas a lo largo de la pista, y cubriendo todos los ángulos, es capaz de representar en 3D cualquier bote de la pelota dentro del campo. Un ejemplo de la representación que ofrece este sistema lo podemos observar en la figura 1.2(a).

La representación de objetos en 3D siempre ha llamado la atención por la posibilidad de poder ver el objeto reconstruido desde cualquier ángulo de la escena. Por esta razón hemos incluido una representación 3D de la escena en nuestro proyecto. La reconstrucción

[1] http://www.hawkeyeinnovations.co.uk/

(a) (b)

Figura 1.2: Simulación del sistema *ojo de halcón* (a) y reconstrucción 3D de una cara (b)

en 3D es un problema clásico dentro de la visión artificial y que se resuelve de un modo conocido. Mediante un par de cámaras se obtienen los puntos relevantes de la escena, se calculan sus puntos homólogos y se triangula para representar en el espacio 3D. De esta forma se obtienen representaciones como las que podemos observar en la figura 1.2(b).

La cámara digital poco a poco a ido incorporándose en multitud de aparatos tecnológicos: consolas, teléfonos móviles, ordenadores portátiles, etc. Esto es debido a que es un sensor barato y que además ofrece un flujo de información muy alto. Aún así, hoy en día es bastante complicado extraer información valiosa de una imagen ya que requiere un coste computacional grande.

1.2. Robótica

La palabra robot proviene de la palabra eslava *robota* que se refiere al trabajo realizado de manera forzosa. Tenemos conocimiento de los primeros robots móviles en los años 70, donde el hombre tenía la intención de crear objetos artificiales con un elevado grado de autonomía. Tuvieron una gran utilización en los años 80, donde se implantaron las primeras máquinas en las fábricas con un objetivo sencillo y para facilitar las tareas repetitivas. A partir de la década de los 80, es familiar ver a robots y autómatas en el sector de la industria.

La industria fue pionera en los llamados robots manipuladores. Los primeros manipuladores eran controlados por humanos, de tal forma que el humano le comandaba las órdenes al manipulador. Los manipuladores más comunes que se pueden encontrar en

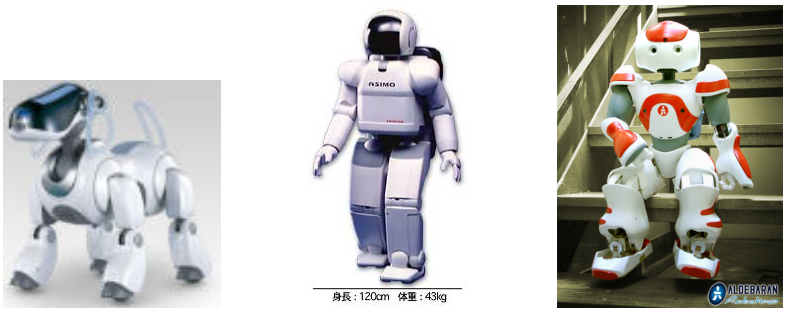
(a) (b)

Figura 1.3: Robots soldadores en una factoría (a) y brazo transportando materiales (b).

una planta industrial se dedican a mover herramientas, piezas, cortar, soldar, etc. Estos manipuladores evolucionaron de tal forma que no necesitaban el control humano, ellos mismos eran capaces de tomar decisiones sobre sus movimientos. A partir de este momento se podía hablar de autonomía aunque ésta fuese muy reducida. El primer tipo de brazo articulado que existió fue el PUMA (*Programmable Universal Manipulator for Assembly*). Dos de las principales ventajas que ofrece el brazo robotizado frente el brazo humano son la precisión y la rapidez.

Uno de los campos de investigación que está captando interés creciente es la robótica móvil. Típicamente un robot móvil consta de un conjunto de sensores, más un conjunto de actuadores más, un ordenador. Dentro de la robótica móvil tenemos dos campos bien diferenciados. Los robots móviles teleoperados son aquellos que no poseen autonomía y todos sus movimientos son comandados desde la distancia por el ser humano. Desde el nacimiento del robot teleoperado, éstos se han venido usando para desempeñar tareas peligrosas como desactivación de bombas, rastreo de minas, exploraciones submarinas o exploraciones en otros planetas como Marte (*Spirit y Opportunity*). Estos últimos cada vez tienen mayor autonomía pero en esencia son teleoperados. Por otro lado tenemos a los robots totalmente autónomos, que no necesitan la ayuda de ningún ser humano para funcionar y moverse por un entorno.

Cabe destacar que el campo de la inteligencia artificial siempre ha estado altamente relacionado con el de la robótica. Esta última se apoya en el razonamiento artificial para crear comportamientos autónomos y a la vez que sean capaces de ir aprendiendo. Como ejemplo de esto último tenemos a los robots humanoides de demostración que están dotados de cierta inteligencia y capacidad de aprendizaje, como el robot ASIMO que podemos ver en la figura 1.4(b).



(a) (b) (c)

Figura 1.4: Mascota AIBO (a), el famoso androide ASIMO (b) y el moderno robot NAO(c)

La robótica es una disciplina que cada vez tiene más auge en la sociedad debido a la gran autonomía que pueden llegar a tener los robots. Los robots serán más útiles en el futuro y más populares cuanta mayor autonomía posean. El interés actual es dotar a los robots de la mayor autonomía posible para desempeñar labores en el hogar como: planchar, barrer, lavar o cualquier otra tarea doméstica. De hecho robots como Aibo (figura 1.4(a)), lego[2] y roomba[3] llevan años adentrándose en los hogares para realizar tareas de limpieza y entretenimiento.

1.2.1. Visión en robots

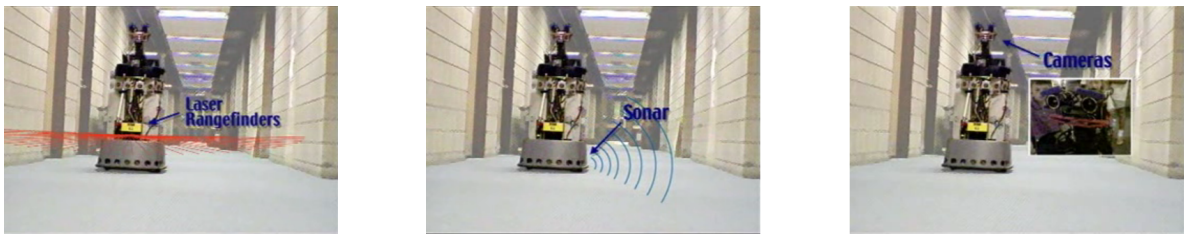
Los robots autónomos usan sensores para percibir la información sobre el entorno que les rodea y así tomar decisiones al respecto. Hasta hace bien poco era normal ver sensores de contacto, sónares o láseres en los robots. Algunos de estos sensores como los láser pueden llegar a ser realmente caros y la información que se obtiene de ellos dista mucho de la que se puede conseguir mediante cámaras.

Por ello en los robots modernos, a parte de incorporar los sensores típicos, incorporan un cámara digital que ofrece mucha más información y además con un precio reducido (el precio de una cámara digital es similar al de un sonar). Robots relativamente modernos como el AIBO(1.4(a)), ASIMO(1.4(b)) o NAO(1.4(c)) incorporan como sensor principal en sus sistemas una o varias cámaras. Las cámaras pueden ser utilizadas para detectar e interactuar con las personas, realizar navegación visual, percibir el entorno de una manera más rica o auto-localizarse en un entorno.

Se han realizado estudios sobre la navegación visual monocular utilizando el robot

[2]http://mindstorms.lego.com/eng/New_York_dest/Default.aspx
[3]http://es.wikipedia.org/wiki/Roomba

AIBO dónde es capaz de estimar distancias analizando imágenes con una sola cámara [Lenser and Veloso, 2003]. También se ha utilizado la visión monocular para percibir el entorno del robot, evitando de esta manera los obstáculos y detectando los objetivos a seguir [Ortiz, 2004], [Gómez, 2002].



(a) (b) (c)

Figura 1.5: Sonar y laser del robot Minerva utilizados para evitar obstáculos(a y b) y 2 cámaras para poder interactuar con la gente y una cámara apuntando al techo para localizarse(c)

Otro ejemplo de robot que usa la visión es *Minerva*. *Minerva* es un robot móvil (figura 1.5) que posee sensores sonares, sensor de láser y varios sensores visuales (cámaras) y cuyo fin es servir de guía a los visitantes de un museo, siguiendo un *tour* e interactuando con las personas. Mediante la visión obtenida por la cámara apuntado hacia arriba y utilizando como balizas las luces colocadas en el techo, es capaz de guiar y localizarse en todo momento, [Sebastian Thrun, 2004].

Figura 1.6: Robot que utiliza la visión para colocar bombones.

En el mundo de la industria también podemos observar ejemplos de robots o brazos mecánicos que toman sus decisiones a raíz de las imágenes que analizan. Uno de estos

ejemplos lo podemos ver en la imagen de la figura 1.6, donde un robot basándose en la información obtenida por la cámara es capaz de detectar y diferenciar un bombón y situarlo dentro de las cajas en su hueco correspondiente.

En nuestro anterior proyecto fin de carrera diseñamos e implementamos un sistema de seguimiento de personas sobre un robot móvil que también usaba una cámara. Mediante la visión éramos capaces de detectar una persona, seguirla, estimar a qué distancia se encontraba y evitar los posibles obstáculos [Calvo Palomino, 2004].

1.3. Atención visual

Uno de los campos más activos de investigación dentro de los robots guiados por visión es la *atención visual*. La atención visual humana representa el mundo en una serie de fotogramas de un séptimo de segundo de duración, de una manera similar a como lo hacen las cámaras de vídeo o como un estroboscopio[4], que se enciende y se apaga periódicamente. Se ha descubierto así que la atención visual funciona como un haz luminoso que ilumina uno o varios objetos, pero no de manera continua, sino que lo hace siete veces cada segundo. Esta captación periódica de información se produce incluso cuando al observador se le presenta un solo estímulo de visión. El descubrimiento resulta esencial para la comprensión de los fenómenos de la atención, que podrían estar relacionados con las oscilaciones que se conocen de la actividad eléctrica cortical.

Los ojos humanos perciben la imagen mediante una distribución irregular de foto-receptores, teniendo en la parte central alta densidad de información (*fóvea*) y en la parte exterior poca densidad de información (*periferia*). La naturaleza, y en concreto los mamíferos, mediante la evolución ha llegado a obtener un sistema de atención compuesto de dos sensores (los ojos) que son capaces de obtener una nitidez extrema en su centro óptico (*fóvea*). Gracias a que prestamos atención en la *fóvea* y la periferia queda desatendida debido a su distorsión, podemos en una *mirada* filtrar un flujo enorme de datos e interesarnos únicamente en la zona central. Por ello los ojos humanos saltan continuamente mediante los movimientos sacádicos para obtener la mayor densidad de información de la escena.

En la imagen de la figura 1.7 podemos ver el recorrido que hace un sistema de atención humano cuando analiza una famosa imagen de *Yarbus*[5]. El sistema de atención humano

[4] http://es.wikipedia.org/wiki/Estroboscopio

[5] http://en.wikipedia.org/wiki/Alfred_L._Yarbus

se basa en movimientos sacádicos (movimientos rápidos, precisos y de corto recorrido) que permiten orientar el punto central de máxima resolución a conveniencia. De este modo podemos ver el recorrido que hacen los ojos humanos sobre las zonas de interés. Es importante observar que el sistema de atención humano presta atención a elementos muy concretos como los ojos, nariz, boca y contorno. El resto de información nunca es interpretada por la zona central de atención (*fóvea*).

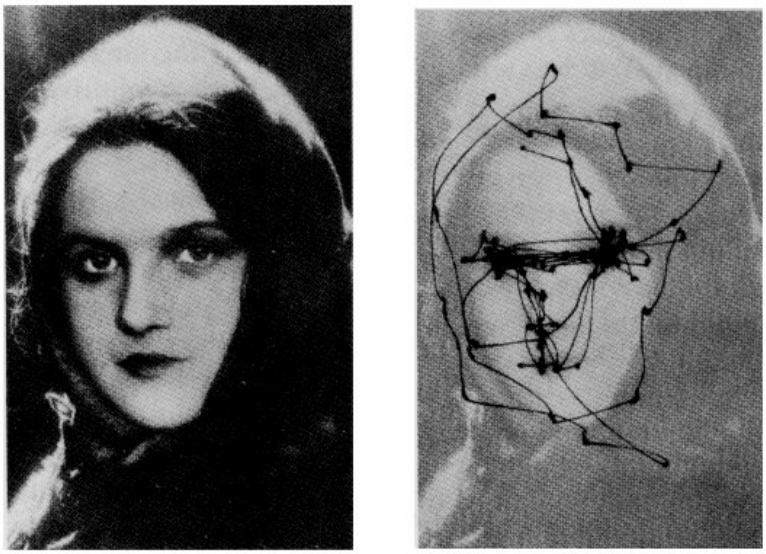


Figura 1.7: Atención del ojo humano sobre la famosa imagen de Yarbus.

Existe un experimento curioso de atención visual realizado por la Universidad de Illinois[6] para constatar que los fenómenos atentivos juegan un papel importante en el sistema visual humano. Antes de mostrar un vídeo se pone énfasis al lector en que preste atención al número de veces que se pasan la pelota un grupo de personas. El sistema humano es capaz de prestar tanta atención a este hecho de pasarse la pelota, que no advierte la presencia de un gorila entre dicho grupo de personas. Esto nos hace reflexionar sobre el grado de atención que es capaz de generar el sistema de atención del ser humano para abstraerse de cualquier situación no relevante para la tarea que sucede en la escena.

Típicamente, la atención visual artificial se puede modular para que llamen la atención ciertos objetos o representaciones. De esta manera podemos realizar una modulación en color, bordes o movimiento para que únicamente estas características nos llamen la atención. Así podemos ver cómo en la figura 1.8 [Itti and Koch, 2004],una imagen se analiza mediante tres características de atención (luminosidad, contraste y orientación) y

[6]http://ibasque.com/experimento-psicologico-de-atencion-visual/

podemos observar cómo el sistema va mirando y saltando sobre aquellos objetos que le suscitan especial interés, es decir, aquellos objetos que su saliencia es mayor. El sistema mira y salta a todos los objetos de interés debido a la *inhibición de retorno*, que evita que la atención se pose únicamente en un objeto, ya que cada vez que *miramos* a un objeto deja de ser interesante para el sistema.

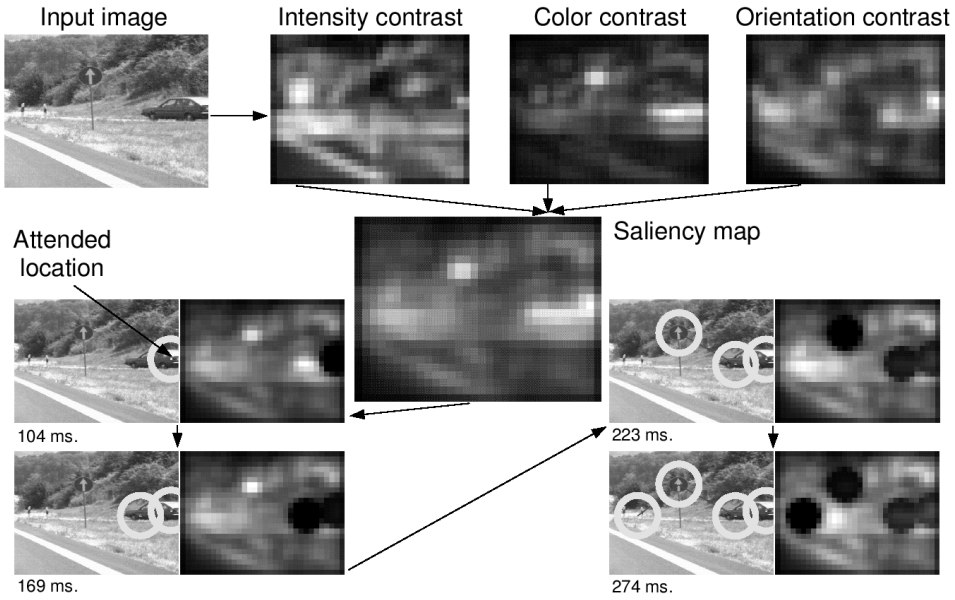


Figura 1.8: Ejemplo de un sistema de atención en funcionamiento.

Dentro de la robótica autónoma es importante realizar un control de atención visual. Las cámaras de los robots proveen de un amplio flujo del que hay que seleccionar lo que es interesante e ignorar lo que no, en esto consiste la atención selectiva. Existen dos vertientes de atención visual, *global* (overt attention) y *local* (covert attention). La atención *local* consiste en seleccionar de una sola imagen aquellos datos que nos interesan (técnica que emplearemos en nuestro proyecto). Y la atención *global* consiste en seleccionar del entorno que rodea al robot aquellos objetos que interesan, y a los que dirigiremos la mirada ([Cañas Plaza, 2005]).

1.4. Reconstrucción 3D mediante un sistema de atención

La motivación de este proyecto es generar un sistema de atención visual que represente en 3D los objetos interesantes de una escena, que sirva de entrada perceptiva a un robot

móvil.

En el Grupo de Robótica de la Universidad Rey Juan Carlos se han abordado problemas de visión artificial, localización visual y sistemas de atención desde diferentes puntos de vista y con diferentes resultados. Se estudió inicialmente un mecanismo de atención 3D sencillo con una única cámara y mediante movimientos sacádicos se iba teniendo una representación general de la escena([Martínez De la Casa Puebla, 2005]). También se estudió la navegación de robots a través de sistemas basados en la atención 3D ([León Cadahía, 2006]).

Se siguió invirtiendo esfuerzos en algoritmos de percepción en 3D que usaban triangulación y segmentación sobre imágenes simuladas ([Pacios and Cañas Plaza, 2007]) y sobre imágenes reales ([Mendoza Baños, 2008]), no llegando a obtener la suficiente vivacidad como para llegar a una representación en tiempo real. Además existen también líneas de investigación que reconstruyen en 3D realizando algoritmos evolutivos([García Martínez, 2007]).

Siguiendo la línea de atención visual y la representación 3D planteamos el presente proyecto. Este proyecto está enfocado a diseñar e implementar un sistema atentivo visual que genere una representación en 3D de los objetos relevantes de la escena. Pretendemos con ello poder representar en 3D cualquier objeto interesante, pudiendo configurar el sistema para prestar atención a objetos móviles, de un color determinado, bordes o incluso a zonas de interés que son calculadas mediante una atención cognitiva del sistema.

La presente memoria detalla todos los aspectos relevantes del desarrollo de este sistema de atención. Esta memoria se compone de 6 capítulos. El capítulo 2 trata los objetivos y requisitos planteados para el proyecto. El capítulo 3 ofrece una descripción de la infraestructura sobre la que ha sido realizado. La descripción informática, la implementación y detalles del comportamiento se exponen en el capítulo 4. En el capítulo 5 detallamos varios experimentos realizados para validar la solución desarrollada y para afinar el comportamiento del sistema. Por último, en el capítulo 6 se reflejan las conclusiones extraídas sobre la realización de este proyecto y las posibles líneas futuras de mejora que se pueden aportar.

Capítulo 2

Objetivos y Metodología

Una vez presentado el contexto general sobre el que se asienta este trabajo, vamos a pasar a explicar una descripción detallada de los objetivos que pretendemos resolver con la realización de este trabajo y de los requisitos que han condicionado la solución desarrollada.

2.1. Descripción del problema

El objetivo general del proyecto consiste en mantener una representación en 3D de los objetos interesantes que se encuentran en una escena determinada, mediante un algoritmo de atención visual. La aplicación debe ser capaz de explorar las imágenes reales, extraer los objetos que más le llamen la atención (color, bordes, movimiento ...) y representarlos en un entorno 3D. Además, el sistema debe ser dinámico y atentivo constantemente, encontrando de esta manera cualquier nuevo objeto que entre en la escena y que llame su atención. Para ello se contará con dos cámaras digitales, a modo de sensores principales, que utilizaremos para obtener las imágenes y generar la representación en 3D.

Este objetivo final lo hemos divido en dos sub-objetivos:

1. *Sistema de atención:* El sistema de atención debe ofrecer información sobre qué objetos llaman la atención en la escena. Realizará una atención selectiva sobre la imagen, de esta forma el sistema de atención irá saltando de unas zonas a otras de la imagen mediante movimientos cortos y rápidos (movimientos sacádicos similares a los del ojo humano).

   El sistema de atención se podrá regular y configurar mediante características visuales *ascendentes* que utilizan como fuente las propias imágenes para determinar

qué objetos producen atención. El sistema de atención deberá poseer varias características visuales de atención (color, bordes y movimiento). Además, el sistema de atención se podrá regular igualmente mediante características cognitivas (*descendentes*) que utilizan una capa de alto nivel, y distinta a la de las imágenes, para calcular, decidir y razonar sobre nuevas zonas de atención.

Estudiaremos la posibilidad de incorporar al sistema una distribución no regular de foto-receptores de la imagen parecida a la que posee el ser humano, teniendo en la fóvea mayor calidad y nitidez que en la periferia.

2. *Reconstrucción 3D:* El sistema de reconstrucción 3D será capaz de posicionar correctamente en el espacio 3D la secuencia de puntos de interés que obtiene del sistema de atención. Para ello, será necesario utilizar técnicas adecuadas para conseguir una buena triangulación de los puntos en tiempo real. La problemática de este sistema se puede resumir en: obtención de puntos (sistema de atención), emparejamiento de puntos entre las dos cámaras y triangulación en 3D.

   En el emparejamiento de puntos se estudiarán diferentes técnicas para, dado un punto de interés de una imagen, obtener el punto homólogo en la segunda imagen. La correcta representación final en 3D depende en gran medida de una técnica adecuada de correlación entre puntos.

Además, realizaremos numerosas pruebas y experimentos para comprobar el correcto funcionamiento de los algoritmos empleados. Se realizarán mediciones de cómputo para intentar llegar a un comportamiento en tiempo real.

Por último, como objetivo personal tenemos el hecho de aprender y conocer más a fondo cómo se realiza un proyecto de investigación de estas características.

2.2. Requisitos

El desarrollo del proyecto estará guiado por los objetivos comentados anteriormente y deberá ajustarse a los requisitos que comentaremos a continuación para asegurar un buen comportamiento del sistema.

Estos requisitos son:

1. Las cámaras deberán ser calibradas con *precisión*, para poder realizar los cálculos geométricos necesarios correctamente y con cierta exactitud.

2. El sistema debe ser *vivaz, rápido y eficiente* para conseguir un análisis continuo de imágenes reales en tiempo real.

3. El sistema debe ser *robusto*, especialmente a los cambios de luminosidad que se producen entre las imágenes de las dos cámaras.

4. En cuanto al software, vamos a desarrollar el comportamiento del sistema en la plataforma denominada *JDE* que se explicará detalladamente en el siguiente capítulo. La utilización de esta plataforma facilita el desarrollo del comportamiento.

5. Ejecutará sobre el sistema operativo *GNU/Linux*, concretamente sobre la distribución Debian.

6. El código resultante de este proyecto será liberado bajo la licencia **GPLv3**[1]. La documentación generada a lo largo de estos meses de investigación la liberaremos bajo la licencia Creative Commons Reconocimiento-Compartir.

2.3. Metodología empleada

En esta sección veremos la metodología utilizada para la realización de este proyecto. Básicamente se basa en realizar iteraciones que se componen de: diseño, implementación y experimentos de cada una de las partes del sistema.

Para la realización de un proyecto se deben establecer unas tareas a realizar desde la idea del proyecto hasta la realización del mismo. Este modelo de desarrollo establece unos requisitos de entrada para producir salidas satisfactorias.

El desarrollo de este proyecto se seguirá el modelo de desarrollo en espiral basado en prototipos. La elección de este modelo de desarrollo se basa en la necesidad de separar el comportamiento final en varias subtareas más sencillas para luego fusionarlas. Cada tarea finalizada aporta los requisitos e información necesaria para abordar la siguiente iteración del modelo de desarrollo.

La gran ventaja de este modelo de desarrollo es la existencia de puntos de control al finalizar cada iteración. Además es altamente flexible en cuanto al cambio de requisitos, hecho muy común en este tipo de proyectos de investigación.

[1] http://www.gnu.org/licenses/gpl-3.0-standalone.html

2.3.1. Desarrollo en espiral

En este tipo de desarrollos, los productos son creados gracias al número de iteraciones que se da en el proceso de vida de software.

- Determinar los objetivos. Los objetivos de un ciclo de desarrollo deben ser identificados y especificados.
- Valorar y reducir los riesgos. Los riesgos son valorados y ciertas actividades son acotadas para reducir los riesgos claves.
- Desarrollar y validar. El sistema se desarrolla y es validado usando pruebas que verifican el cumplimiento de los requisitos fijados.
- Planificar. El proyecto es repasado y la próxima fase de la espiral es planificada.

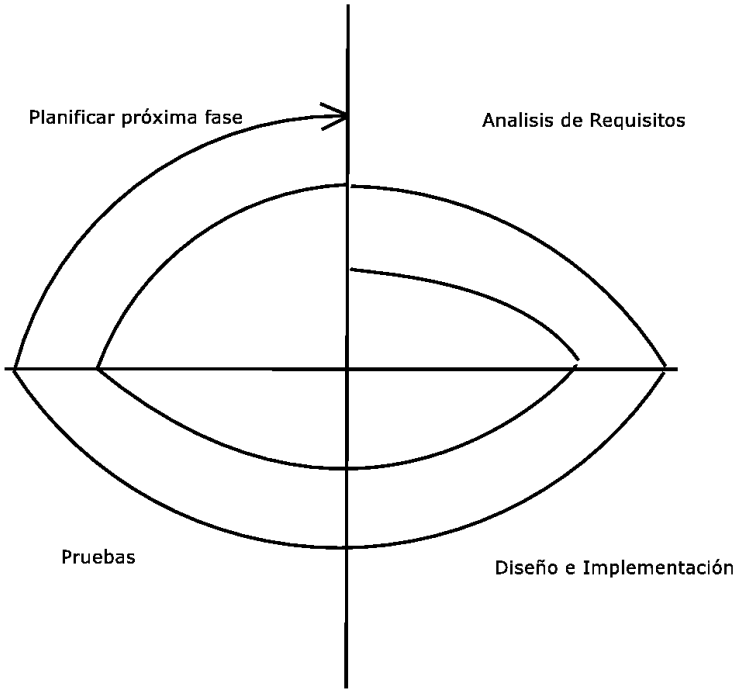


Figura 2.1: Modelo en espiral

En la figura 2.1 se pueden observar las fases que forman cada ciclo en este modelo de desarrollo: Análisis de requisitos, diseño e implementación, pruebas y planificación del próximo ciclo de desarrollo.

2.3.2. Plan de trabajo

En esta sección vamos a dejar reflejado el plan de trabajo que hemos seguido, repartiéndolo en diferentes fases cuyos resultados son prototipos o conocimiento adquirido.

Un prototipo es una versión preliminar de un sistema con fines de demostración o evaluación de ciertos requisitos. Se suele estimar la finalización de una iteración en el modelo de desarrollo como la obtención de un nuevo prototipo. De hecho, el modelo que hemos seleccionado se basa justamente en esto, en la finalización de cada iteración se obtiene un nuevo prototipo. Cierto es que la obtención de un prototipo no implica que se utilice para el producto final, si se cree necesario se puede desechar dicho prototipo.

- **Fase 1:** Estudio del estado del arte. En esta fase estudiaremos y analizaremos bibliografía sobre sistemas de atención y de reconstrucción 3D. Para ello leeremos diferentes artículos de investigación y proyectos fin de carrera para obtener de esta forma un contexto técnico. Esta fase tiene como resultado: *adquisición de conocimiento*.

- **Fase 2:** Familiarización con plataforma software JDE. Es necesario un primer contacto con la arquitectura software que vamos a utilizar para la realización del proyecto. Esta fase tiene como resultado: *adquisición de conocimiento*.

- **Fase 3:** LogPolar. En este prototipo implementamos la representación logpolar de una imagen. Gracias a esta implementación podemos observar los movimientos sacádicos que se realizan, similares a los de un ojo humano. Esta fase tiene como resultado: *prototipo-1*.

- **Fase 4:** Atención Visual. En esta fase estudiaremos el mecanismo de atención visual basado en mapas de saliencia. Además se añadirán diferentes características de atención como movimiento, bordes, o filtrado por color. Esta fase tiene como resultado: *prototipo-2*.

- **Fase 5:** Calibración de Cámaras. En esta fase se modificará y utilizará un calibrador de cámaras para asentar bien los parámetros intrínsecos y extrínsecos de la cámara. Este prototipo es muy importante ya que una mala calibración de las cámaras implica un resultado poco fiable en la representación 3D. Esta fase tiene como resultado: *prototipo-3*.

- **Fase 6:** Cálculo de homólogos y triangulación. En esta fase se realizarán multitud de pruebas para dar con la fórmula apropiada para el cálculo de puntos homólogos en

2D, y posteriormente obtener los puntos 3D correspondientes. Esta fase tiene como resultado: *prototipo-4*.

- **Fase 7:** Representación en 3D. En esta fase se llevarán a cabo multitud de cálculos geométricos y ajustes para conseguir una representación en 3D a partir de las dos imágenes. Esta fase tiene como resultado: *prototipo-5*, que será el prototipo que se convierta en la aplicación final.

2.4. Esfuerzo temporal del proyecto

Hemos querido mostrar el esfuerzo que hemos empleado a lo largo de este proyecto, para la realización de los prototipos. No todos los prototipos nos han requerido el mismo esfuerzo.

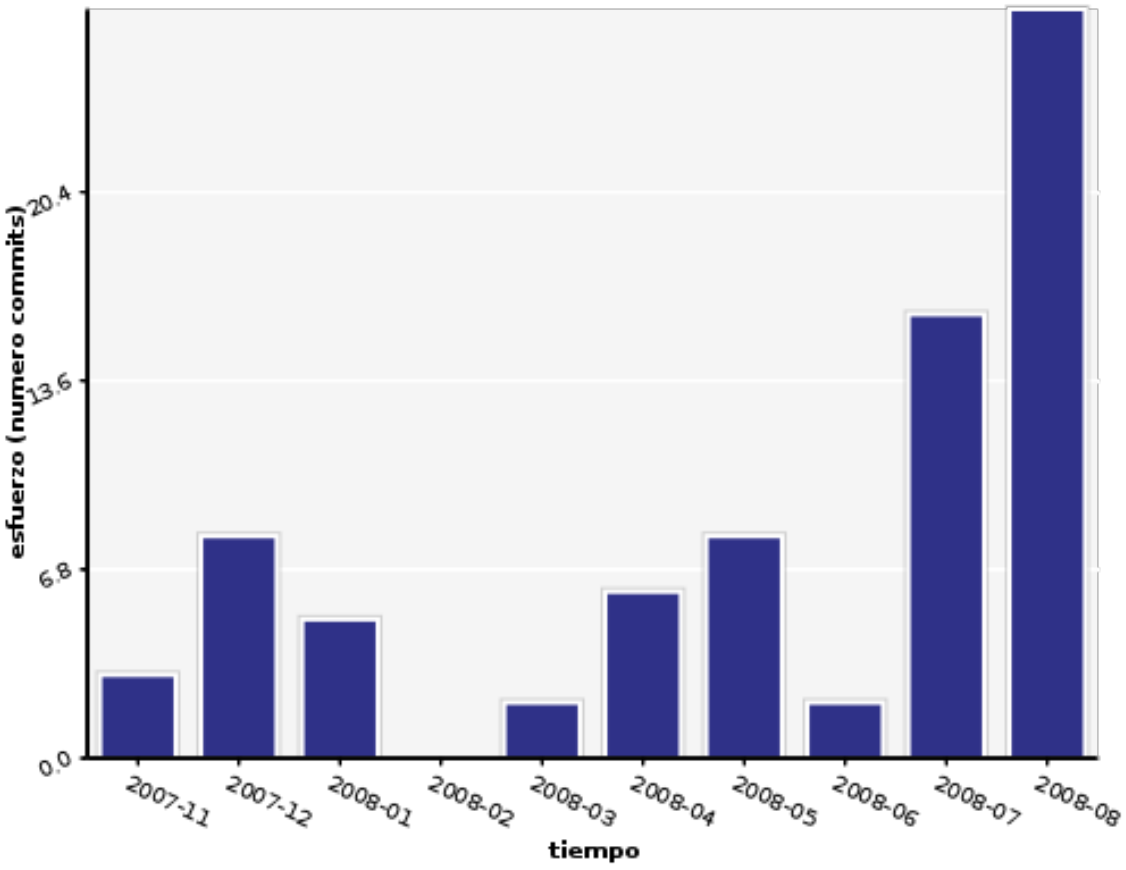


Figura 2.2: Gráfica de esfuerzo

Si observamos la figura 2.2 podemos ver cómo iniciamos el proyecto suavemente adquiriendo conocimientos sobre el estado del arte y sobre la plataforma software. Un primer pico de trabajo lo tuvimos en diciembre del 2007 ya que estábamos implementando y afinando el prototipo 2 (atención visual). Más tarde, en abril y mayo del 2008 también observamos dos picos, correspondientes a los prototipos 3 y 4. El prototipo 3 costó afinarlo y obtener un comportamiento estable y robusto. Igualmente el prototipo 4 es uno de los puntos importantes del proyecto, donde se buscó la eficiencia y robustez en el cálculo del píxel homólogo. Finalmente, durante los meses de julio y agosto, se dedicó exclusivamente a engrasar las piezas del proyecto y conseguir un comportamiento correcto. Además, la realización de la memoria ha constituido un esfuerzo considerable, representado en la gráfica.

Capítulo 3

Plataforma de Desarrollo y Mejoras

Una vez que hemos detallado los requisitos y objetivos de este proyecto, vamos a describir en este capítulo la plataforma de desarrollo empleada, tanto hardware como software. Además, describiremos el repertorio de bibliotecas en las que nos hemos apoyado para la realización del proyecto.

3.1. Plataforma Hardware

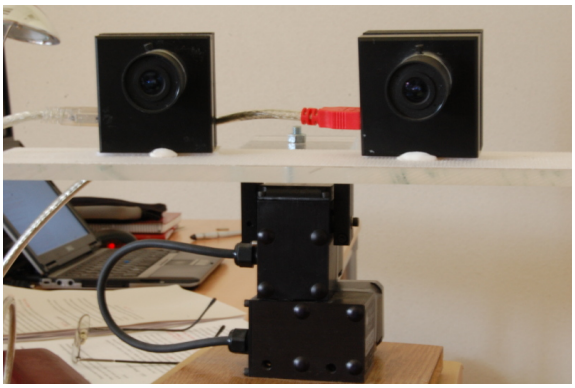
Presentamos ahora de una manera más detallada los componentes hardware que hemos utilizado para este proyecto: un par estéreo para obtener las imágenes y un portátil para la realización de cálculos.

3.1.1. Cámaras firewire

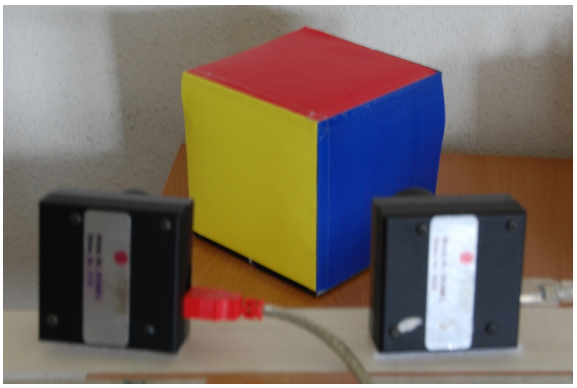
Las cámaras son el elemento fundamental para el funcionamiento de este proyecto. Son los sensores perceptivos utilizados por la aplicación, y se usan para poder triangular y estimar la posición 3D de los objetos. Las dos cámaras utilizadas son del modelo *Videre Cam-L*. Este modelo de cámara, que posee auto-iris, permite capturar imágenes en color de hasta 640x480 píxeles con un refresco de 15 fotogramas por segundo (fps) o bien de 320x240 a 30 fps, que es el modo que hemos utilizado en este proyecto. Dichas cámaras se conectan al bus firewire del portátil utilizando un *hub*. La decisión de utilizar estas cámaras firewire en vez de USB tradicionales, es que los datos de las imágenes llegan por DMA, liberando así de esta tarea a la CPU y quedando más desahogada para el resto de cálculos. Las cámaras poseen un sistema de enfoque manual, lo que nos permite ajustar la imagen en la escena de la mejor forma posible.

El par de cámaras lo hemos montado sobre un cuello mecánico, como se puede ver en la figura 3.1(a). Con esta disposición podríamos mover, en caso de necesitarlo, el par de cámaras hacia cualquier otra dirección.

Para obtener las imágenes de las cámaras hemos utilizado la plataforma software JDE, descrita en la sección 3.2.2



(a)



(b)

Figura 3.1: Par estéreo de cámaras montado sobre el cuello mecánico (a) y visión aproximada que tienen las cámaras sobre la escena (b).

3.1.2. Portátil

Para la captura de imágenes y cálculos de algoritmos hemos utilizado un portátil DELL D430. Es un ordenador con procesador Intel(R) Core(TM)2 Duo CPU @ 1.33GHz, bus firewire, bus USB, y aceleración gráfica, lo que nos permitirá un alto rendimiento en la representación en 3D.

3.2. Plataforma Software

La implementación de este proyecto consiste en un conjunto de componentes software que se apoyan unos en otros, ofreciendo así el comportamiento deseado. Presentamos en esta sección el software y librerías auxiliares que hemos utilizado para la realización de este proyecto.

3.2.1. Sistema operativo y lenguaje

Debian[1] es un sistema operativo basado en GNU/Linux, de libre distribución, multitarea y multiusuario. Concretamente hemos utilizado *Debian Sid* sobre la versión de kernel *2.6.18-5-686*. Es un sistema operativo más que probado en arquitecturas de 32 bits, robusto, ágil y con un número extenso de aplicaciones y librerías instalables. Es accesible gratuitamente por medio de internet.

En cuanto al lenguaje de programación, se ha implementado en C y C++. La arquitectura de JDE permite utilizar C o C++, a gusto del programador, dependiendo de la tarea a realizar. Además, al existir una gran comunidad alrededor de estos lenguajes en el grupo de investigación, la reutilización o integración de partes de código se vuelve más factible.

3.2.2. JDE suite

La plataforma que hemos elegido para la realización de este proyecto es *JDE*[2] (Jerarquía Dinámica de Esquemas). Este entorno, diseñado para crear comportamientos en robots móviles, ha sido creado y diseñado por el Grupo de Robótica de la URJC a lo largo de más de 5 años de trabajo. JDE facilita la creación de aplicaciones para que el robot se comporte de una determinada manera y exhiba conductas autónomas. Para ello resuelve los aspectos más generales de la programación de robots, como son el acceso a los sensores y actuadores, la multitarea, interfaces gráficas y las comunicaciones entre componentes.

La arquitectura de JDE se basa en pequeñas unidades de comportamientos denominadas esquemas. Estos esquemas son un flujo de ejecución independiente (modulable, iterativo, y que puede ser activado o desactivado a voluntad) con un determinado objetivo. En cada uno de estos esquemas se encapsula diferente funcionalidad que podrá ser reutilizada en posteriores ocasiones. Siguiendo la línea de comportamientos basados en esquemas, nuestro software de este proyecto se compondrá de un esquema.

Existen dos tipos de esquemas en JDE, los esquemas perceptivos y los esquemas motores o de actuación. Los esquemas perceptivos son los encargados de recoger los datos sensoriales, procesarlos y poner esa información a disposición de cualquier esquema. Los esquemas motores utilizan la información proporcionada por los esquemas perceptivos y generarán una orden de movimiento sobre el robot. Los esquemas se materializan en una hebra de kernel, independiente de las demás.

[1] http://www.debian.org/
[2] http://jde.gsyc.es/

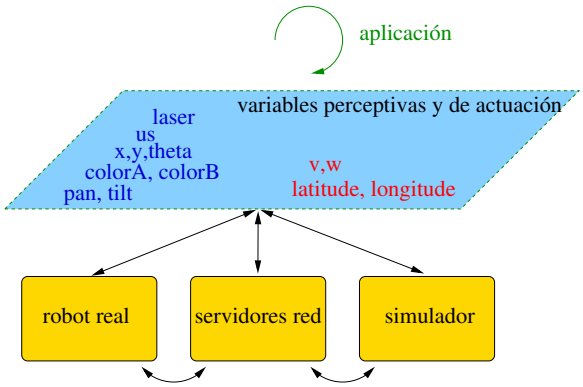


Figura 3.2: Comportamiento de la arquitectura con un esquema en ejecución.

La manera en la que cada esquema se comunica con otros componentes es mediante el uso de variables compartidas. De este modo se simplifica la comunicación entre ellos y agiliza enormemente en comparación al uso de paso de mensajes. Un esquema perceptivo consigue una determinada lectura del sensor físico y modifica la variable común respectiva, que está a disposición de todos los esquemas.

Además de soportar el acceso a a sensores y actuadores reales, *JDE* incluye el soporte a simuladores como *Stage*[3] o *Gazebo*[4]. Para este proyecto hemos optado por no utilizar simuladores y analizar directamente las imágenes reales de las cámaras, lo que implica una mayor problemática, pero supone una funcionalidad más útil y fiable.

Actualmente *JDE* se encuentra en su versión 4.2.1, que cuenta con un número bastante alto de componentes: 18 esquemas y 12 drivers. Gracias a los drivers *firewire* y *mplayer* tenemos resuelto el acceso a las cámaras firewire y USB, ofreciendo las imágenes *RGB* en arrays de 320x240 píxeles. Además, ofrece drivers como *graphics-gtk* o *graphics-xforms* que nos dan la posibilidad de realizar la interfaz en diferentes gestores de ventanas. En nuestro proyecto hemos optado por utilizar el driver *graphics-xforms*.

La arquitectura de *JDE* va creciendo año a año y permite que cada vez los proyectos se asienten sobre una base más estable y rica en comportamientos.

3.2.3. Biblioteca Progeo

La arquitectura de *JDE* se compone, además de los esquemas y núcleo, de un conjunto de librerías que dan solución a los problemas típicos dentro de visión. Así, tenemos la librería

[3]http://playerstage.sourceforge.net/
[4]http://playerstage.sourceforge.net/gazebo/gazebo.html

de geometría proyectiva *Progeo*, utilizada para operar con un espacio tridimensional. Esta librería es utilizada por nuestra aplicación para relacionar el mundo de las imágenes (2D) con el mundo real (3D).

Esta relación se consigue gracias a dos funciones principales:

- *Proyectar.* Esta función permite realizar la proyección geométrica de un punto 3D en el espacio, en el plano imagen de la cámara, obteniendo así un punto 2D perteneciente a ese plano. Ese punto 2D nos da información del píxel de la imagen en el que proyecta el punto 3D correspondiente.
- *Retro-proyectar.* Dado un píxel perteneciente al mundo 2D, está función permite obtener el rayo óptico 3D del conjunto de puntos que 3D proyectan en ese píxel.

La librería Progeo se basa en un *modelo de cámara* para realizar estas transformaciones. En concreto se utiliza el modelo *Pinhole*, cuya figura descriptiva se puede observar en la figura 3.3

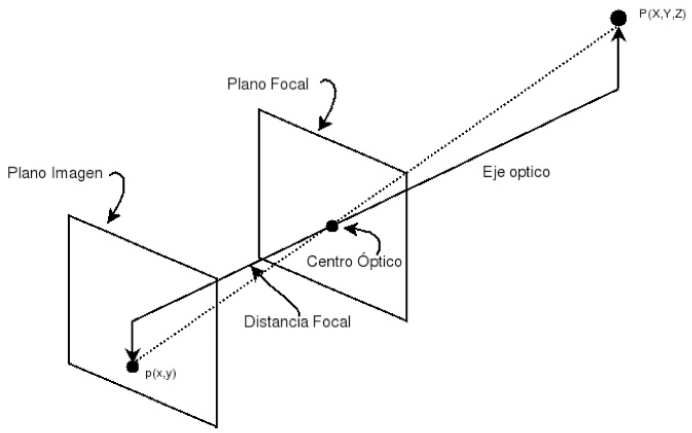


Figura 3.3: Modelo de cámara Pin-Hole.

En este modelo se asume que cualquier punto *P(x,y,z)* se proyecta en el plano imagen a través de otro único punto llamado *centro óptico*. La recta que une el punto P y el centro óptico se denomina *línea de proyección* e intersecta al plano imagen justo en el píxel *p(x.y)*, que es la proyección de *P(x,y,z)*. El centro óptico está situado a la *distancia focal* del plano imagen. Este modelo lo completan el *eje óptico*, que es una línea perpendicular al plano imagen y que atraviesa al centro óptico, y también el *plano focal*, que es el plano

perpendicular el eje óptico cuyos puntos no se proyectan en el plano imagen, incluyendo al centro óptico.

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

3.2.4. Calibrador de cámaras

Para la correcta simulación del entorno 3D es necesario que las cámaras se encuentren calibradas correctamente. En este proyecto hemos utilizado un par estéreo de cámaras para obtener las imágenes de la escena a analizar. Estas imágenes nos proveen de información en 2D y además nos apoyaremos en ellas para realizar la triangulación y posicionamiento en 3D. Para que toda la geometría funcione bien, necesitamos que las cámaras estén correctamente calibradas, esto es, obtener sus parámetros extrínsecos (posición, ángulo, ...) e intrínsecos (foco, distancia focal, ...).

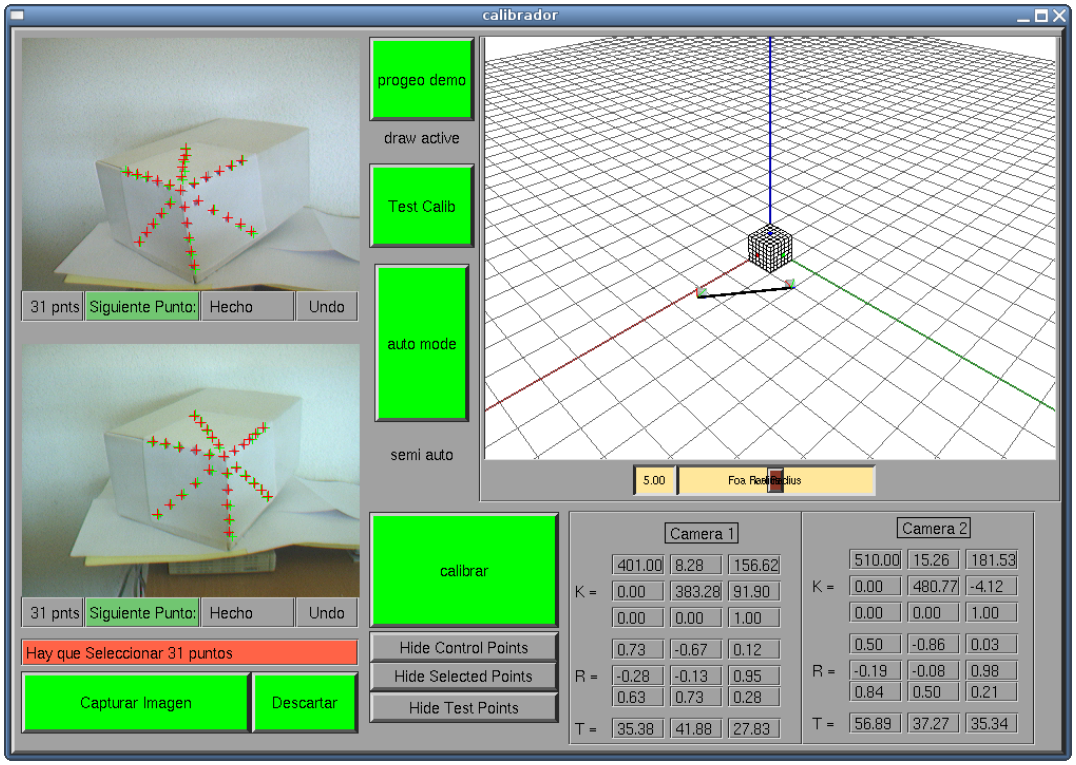


Figura 3.4: Esquema calibrador en JDE en el momento de calibrar el par estéreo usado en este proyecto.

Típicamente en el Grupo de Robótica se venía usando la aplicación *ARToolkit*, que era capaz de calibrar las cámaras, pero el proceso de calibración era bastante costoso. En este proyecto hemos optado por utilizar el esquema ya integrado en JDE que calibra cámaras, y que es el resultado del proyecto fin de carrera [Kachach, 2008]. Gracias a este esquema y utilizando la arquitectura de *JDE* podemos obtener de una manera fácil y eficaz los parámetros extrínsecos e intrínsecos de las cámaras. Estos conjuntos de parámetros que se obtienen de la cámara se representan en un modelo compatible con el modelo de cámaras de la librería *progeo*.

Como podemos observar en la figura 3.4, el esquema que calibra las cámaras posee un interfaz amigable y es posible comprobar al momento si la calibración es correcta, mediante diversas demos y comprobaciones que posee.

3.2.5. OpenGL

Esta biblioteca la hemos utilizado para la representación en 3D de la escena, añadiendo una cámara virtual para poder observar dicha escena desde cualquier posición.

OpenGL[5] ofrece un API multi-lenguaje y multi-plataforma para escribir aplicaciones que produzcan gráficos eficientes en 2D y 3D. Esta especificación fue desarrollada por Silicons Graphics y a partir de ella, los fabricantes han creado hardware específico que implementa y procesa instrucciones de OpenGL de una manera eficiente y rápida. Las operaciones necesarias de proceso se delegan en la *GPU* (Graphical Process Unit). De este modo se libera a la CPU de todas las tareas que tienen que ver con la generación de gráficos, quedando así disponible para los restantes cálculos necesarios.

Las operaciones básicas de OpenGL operan sobre elementos básicos, puntos, líneas, polígonos. El proceso de convertir estos elementos en píxeles se traduce mediante un *pipeline* (tubería) denominada la máquina de estados de OpenGL (ver figura 3.5).

3.2.6. OpenCV

OpenCV es una librería libre de visión artificial desarrollada por Intel. Su licencia BSD permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. La librería es multi-plataforma, y puede ser usada en Mac OS X, Windows y Linux. Está orientada al tratamiento de imágenes en tiempo real, lo

[5]http://www.opengl.org/

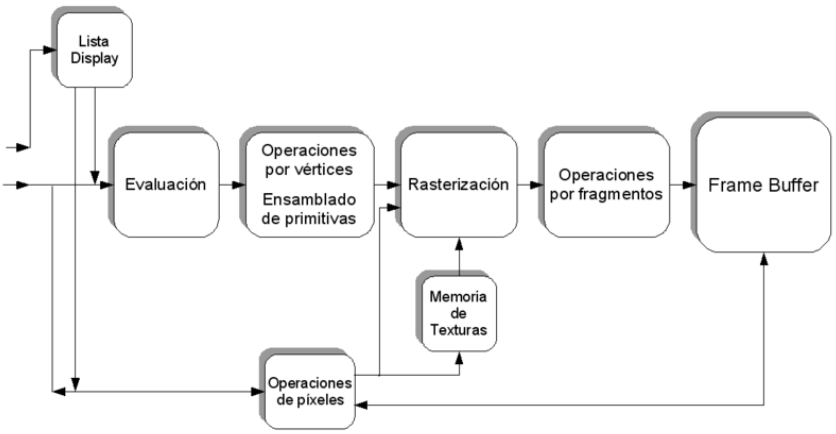


Figura 3.5: Pipeline de OpenGL

que es posible si encuentra en el sistema las Primitivas de Rendimiento Integradas de Intel (IPP).

OpenCV es muy utilizado, como por ejemplo en el sistema de visión del vehículo no tripulado de Stanley, el ganador del año 2005 del *Gran desafío DARPA*[6]. Además se usa también en numerosos sistemas de vigilancia de vídeo.

Nosotros hemos utilizado esta librería para analizar las imágenes y obtener un filtrado de bordes (*cvCanny*) y para obtener un filtrado por esquinas (*cvCornerHarris*).

3.2.7. GSL

GSL (*GNU Scientific Library* es una librería de uso libre que ofrece un conjunto muy amplio de operaciones matemáticas. La biblioteca ha sido escrita de forma entera en el lenguaje C y ofrece un API muy fácil de usar facilitando al programador el uso de las distintas funcionalidades que ofrece. Entre todas estas funcionalidades podemos encontrar rutinas de: álgebra lineal, números complejos, polinomios, vectores y matrices, permutaciones y muchas más operaciones.

En este proyecto hemos utilizado GSL para la realización de un recubrimiento para la salida de datos del esquema *calibrador*. Este esquema ofrece una salida de matrices correspondientes a los datos intrínsecos y extrínsecos de la cámara. Gracias a GSL, todas las operaciones matriciales las hemos realizado sin problemas.

[6]http://en.wikipedia.org/wiki/DARPA_Grand_Challenge

3.2.8. XForms

Por último, hemos utilizado la biblioteca de interfaces gráficas *XForms* para programar la interfaz del componente software desarrollado en este proyecto. Ésta ofrece un amplio repertorio de objetos gráficos con los que confeccionar la interfaz, cada uno con una serie de atributos y operaciones. Estos objetos no sólo pueden recibir eventos provenientes del usuario mediante el ratón o el teclado, sino que también pueden ser accesibles desde código.

Las aplicaciones sobre cualquier sistema de visión y robótica suelen incluir una interfaz gráfica, y ésta biblioteca posibilita crear una propia. A través de ésta interfaz podremos visualizar los elementos y variables característicos de nuestro proyecto, como pueden ser las imágenes, filtros, contadores, etc. Además, podremos interactuar y depurar la aplicación con facilidad y rapidez.

JDE incluye soporte para que los esquemas puedan programar su interfaz ayudándose en esta librería. Este soporte se materializa en el driver *graphics-xforms*.

La biblioteca *Xforms* permite a la aplicación muestrear de manera no bloqueante el estado de la interfaz y mantener el control del flujo de ejecución. De esta manera la interfaz gráfica se inserta en los programas del sistema como dos tareas: muestrear si el usuario humano ha producido algún evento y refrescar las imágenes y datos que se están mostrando.

3.3. Mejoras realizadas a la plataforma

Durante el desarrollo del proyecto nos hemos encontrado con diferentes dificultades que hemos tenido que solucionar para poder seguir adelante. En algunos casos simplemente ha sido retocar cierto código, y en otros casos ha sido necesario implementar librerías de servicio.

3.3.1. Driver firewire

Como hemos comentado anteriormente, la captura de imágenes la hemos realizado a través del bus firewire utilizando dos cámaras. Al utilizar un kernel moderno hemos observado que el driver firewire para *JDE* no funcionaba correctamente, congelándose la imagen de las cámaras y produciendo efectos no deseados en la aplicación. El antiguo driver firewire para *jde* estaba compilado contra la librería *libdc1394-13* y su funcionamiento sólo era aceptable en kernels menores al *2.6.15*.

Por todo ello, hemos implementado un nuevo driver firewire para *JDE* utilizando la librería *libdc1394-22*, que siendo una actualización de la anterior librería cambia completamente el API de acceso al bus firewire y por lo tanto el acceso a las estructuras de imágenes de la cámara. De ahí la necesidad de reescribirlo. Con este aporte, la arquitectura software de *JDE* es más rica y versátil en drivers firewire.

3.3.2. Librería colorspaces

En multitud de ocasiones hemos necesitado realizar un filtro de color en el espacio HSV. Típicamente esta conversión es costosa debido a que se ha de realizar una invocación por cada píxel de la imagen, para pasar del espacio de color *RGB* al espacio de color *HSV*. Esta tarea de conversión ralentiza notablemente el funcionamiento del comportamiento.

Por ello, hemos implementado una librería *eficiente* de conversión de espacios de color. Hemos desarrollado la librería de tal manera que se genera inicialmente una tabla de transformación entre *RGB* y *HSV*. En el espacio *RGB* cada una de las tres características de color puede tomar valores entre 0 y 255 (8 bits). Eso implicaría que la tabla generada tendría *16777216* entradas, que como podemos suponer es bastante espacio en memoria y bastante costosa de crear. Debido a esto, hemos optado por reducir el número de entradas de la tabla a *262144*, utilizando únicamente los 6 bits más significativos de los valores *RGB*.

Una vez creada la tabla, cada consulta de conversión entre espacios se resuelve como un acceso directo a la tabla (*array*) de conversiones. De esta manera, por cada invocación para el cambio de espacio de color realizamos una indexación directa en un array, en lugar de invocar la ejecución de un *arco-coseno* y una *raíz cuadrada*, que son las operaciones necesarias en la conversión y lo que verdaderamente ralentiza a los algoritmos típicos de conversión entre espacios de color. Introduciendo esta nueva librería, hemos observado una mejora notable, consiguiéndose un rendimiento del *300* %.

La librería *colorspaces* está incluida en la suite de *JDE* desde la versión *4.2.1*, y ha sido utilizada para el desarrollo de varias aplicaciones y proyectos fin de carrera, además de éste.

3.3.3. Mejoras en el calibrador de cámaras

El calibrador de cámaras que hemos utilizado funciona correctamente, pero hemos tenido que añadir cierta funcionalidad para poder trabajar cómodamente en el proyecto.

- *Aplanado de datos.* Hemos realizado un aplanado de los datos de la calibración a texto plano para poder leerlos desde otra aplicación o esquema sin demasiados problemas.
- *Abstracción de cámaras:* Hemos realizado una clase en C++ que partiendo de la información del fichero generado por el calibrador, es capaz de crear una estructura de cámaras de progeo totalmente configurada.
- *Captura de imágenes:* Hemos añadido funcionalidad sobre el calibrador para que guarde en fichero las imágenes propias sobre las que se realiza la calibración.

Capítulo 4

Descripción Informática

Una vez que hemos explicado en los capítulos anteriores los requisitos y las herramientas necesarias para la elaboración del proyecto, vamos a explicar en este capítulo el software desarrollado del proyecto y cómo hemos abordado y resuelto los objetivos principales. Para ello, vamos a comenzar dando una visión general del diseño y posteriormente se describirán sus detalles y la implementación que se ha llevado a cabo en cada una de sus partes.

4.1. Diseño global

Como comentamos en el capítulo 2 el objetivo global del proyecto es representar en 3D una escena, partiendo de las características salientes del sistema que se obtienen mediante el análisis de las imágenes ofrecidas por las cámaras. La aplicación desarrollada es capaz de explorar las imágenes, obtener las zonas de atención correctamente, y representar esos valores en un entorno 3D. Se ha definido que las zonas de la imagen con alto contraste (bordes), colores llamativos o movimiento llamen la atención (mecanismos de atención ascendente). Estas características pueden mezclarse o utilizarse individualmente para obtener las zonas y puntos de interés en el sistema. Además, gracias a las características de atención descendente o cognitivas (que no basan sus decisiones en características de la imagen) podemos fijar la atención en ciertas zonas de la imagen interesantes por alguna razón o conocimiento no visual.

Siguiendo la línea marcada por la plataforma *JDE*, se ha resuelto el objetivo de este proyecto mediante la creación de un esquema y utilización de drivers de obtención de imágenes como son el driver firewire y el driver imagefile. El diseño global de la aplicación se observa en la figura 4.1. En ella podemos observar el esquema creado, denominado

attention, que es el encargado de procesar la información sensorial de las cámaras y dar como salida una lista de puntos en 3D, que son el resultado del procesamiento visual atentivo y del emparejamiento.

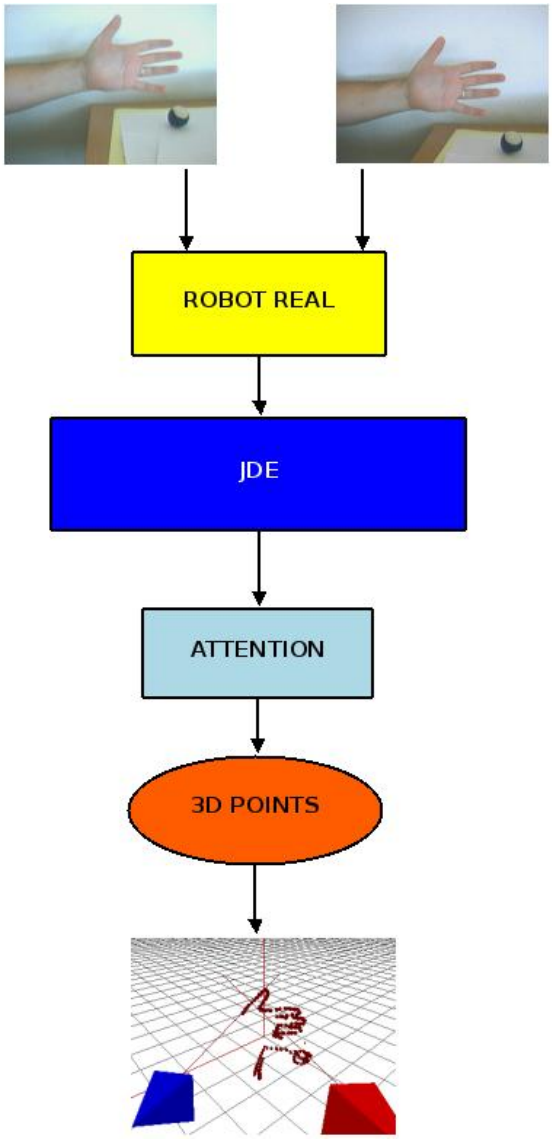


Figura 4.1: Esquema global del comportamiento en JDE.

El funcionamiento final del comportamiento queda reflejado en la figura 4.2. El sistema captura las imágenes de las cámaras. La imagen de la cámara maestra (*imagen-A*) es tratada por el sistema de saliencia para generar los mapas de saliencia instantáneo y acumulado para después poder obtener los puntos o zonas interesantes de la imagen. Estas zonas interesantes vienen moduladas dependiendo del filtro de atención que haya sido seleccionado. Como veremos, podemos mantener la atención en colores concretos, bordes, movimiento o incluso estas tres características combinadas entre sí. Además,

podemos incorporar zonas de interés adicionales mediante comportamientos cognitivos que hipotetizan dónde mirar en casos determinados. Una vez que hemos obtenido los puntos de interés de la imagen, calculamos sus puntos homólogos mediante el cálculo de la recta epipolar y el cálculo de los puntos candidatos (aquellos que están en la recta epipolar y poseen saliencia) en la *imagen-B*. El cálculo del punto homólogo no es tarea sencilla, y de hecho hemos implementado diferentes métodos para intentar obtener la técnica más eficiente y robusta. El cálculo correcto del punto homólogo en la *imagen-B* nos permite, mediante geometría, poder triangular y representar ese punto en la escena 3D. Las diversas restricciones de dónde buscar el punto homólogo nos ha dado la posibilidad de ofrecer un resultado eficaz y vivaz.

Una vez que tenemos la colección de puntos 3D correspondiente a los puntos que ha obtenido el sistema de atención, podemos pasar a utilizarlos para diferentes situaciones: una de ellas es representar esos puntos en una escena en 3D mediante algún gestor gráfico que lo permita, como OpenGL. Otra utilidad es aprovechar esos puntos 3D para analizar, navegar o localizarse en un entorno determinado. Como aporte interesante, también podemos utilizar estos puntos 3D para estimar, razonar y tomar decisiones de una manera mucho más precisa y muy similar a como lo hace el sistema humano en 3D.

Una vez dada la visión general que posee el sistema y sus características globales, pasamos a detallar el diseño y la implementación del esquema realizado y sus partes. En las siguientes secciones se tratará primero lo relacionado con visión en dos dimensiones y el sistema de atención (sección 4.3) y en tres dimensiones (sección 4.4), para después acabar hablando del aspecto y manejo global de la aplicación y la interfaz gráfica (sección 4.5).

4.2. Atención estática con imágenes Log-Polar

Típicamente la representación cartesiana de imágenes se ha utilizado para casi cualquier análisis de una escena, lo que implica tener una distribución uniforme de la imagen. Ya que buscamos con este proyecto intentar acercarnos al sistema de atención humano, vamos a estudiar el resultado que se obtiene al utilizar representación de las imágenes en log-polar, que nos ofrece una distribución no uniforme de los foto-receptores de las imágenes y nos permite obtener focos de atención.

Como ya mencionamos, La representación visual que tenemos los seres humanos se caracteriza por tener una visión nítida, muy nítida, en el punto central hacia donde dirigimos la mirada (fóvea), y cuanto más nos alejamos de esa zona central, la imagen se

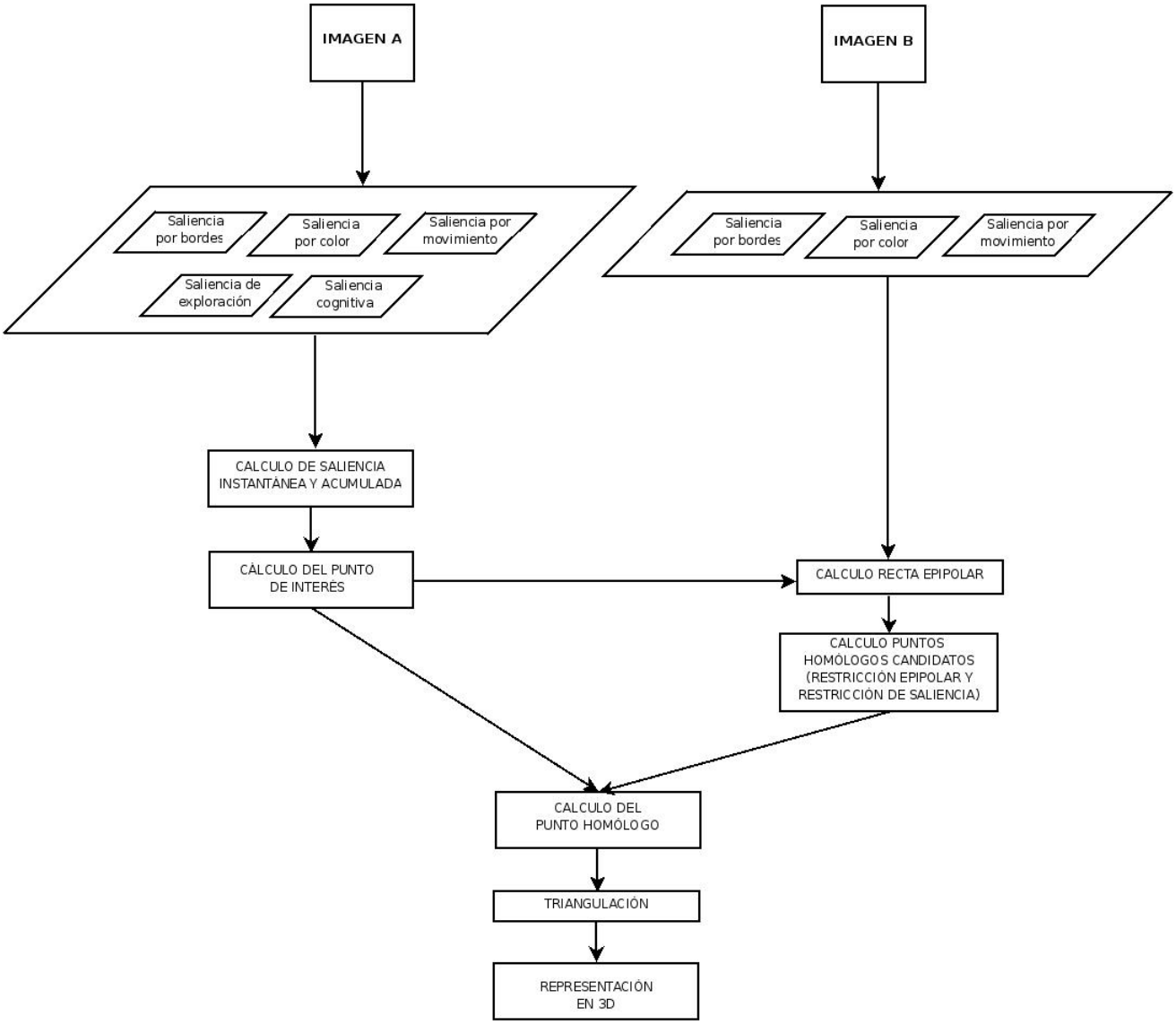


Figura 4.2: Flujo del comportamiento global.

va distorsionando y desenfocando (periferia). Este mismo mecanismo es el que intentamos imitar mediante una imagen en coordenadas *log-polar*.

La idea que motivaba la creación de imágenes en coordenadas logpolar es su semejanza con la estructura de la retina de algunos sistemas de visión biológica y sus cualidades en cuanto a la compresión de datos [Javier Traver, 2005]. En comparación con las habituales imágenes cartesianas, se ha comprobado que el sistema de imágenes en logpolar permite velocidades de muestreo más rápidas en sistemas de visión artificial, sin reducir el tamaño del campo de visión y aumentando la resolución sobre la parte central de la retina (*fóvea*).

En la imagen 4.3 se puede observar un mapa esquemático de la imagen cartesiana (dividida en porciones) y la imagen en logpolar (dividida en píxeles). La conversión de una a otra viene dada por la siguiente fórmula:

$$\begin{cases} \xi = & \log\sqrt{x^2+y^2} \\ \eta = & \arctan(\frac{y}{x}) \end{cases}$$

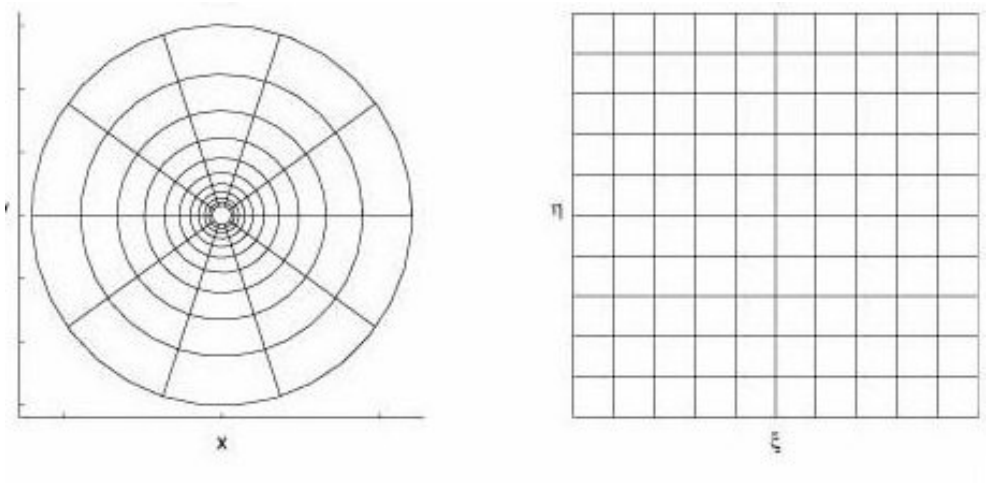
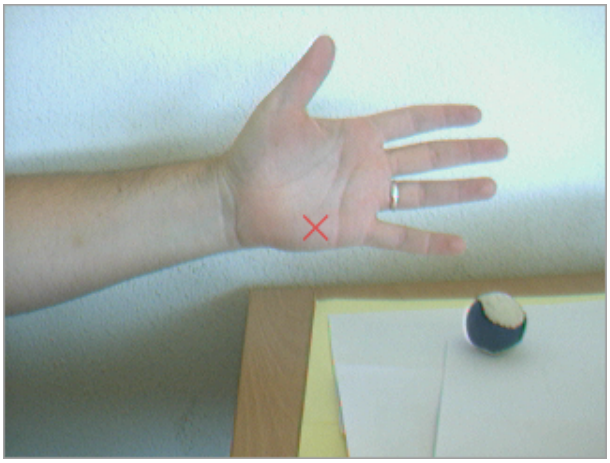


Figura 4.3: Conversión de cartesianas a logpolar.

Una de las ventajas de utilizar este modo de representación es que se reduce mucho el muestreo de datos. Ésto lo obtenemos mediante la reducción de la resolución en la imagen periférica, y aumento de la resolución en la *fóvea*, mejorando así el procesamiento.

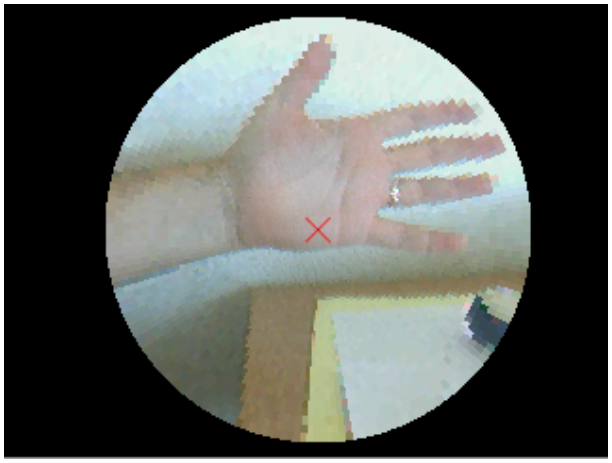
En nuestro sistema hemos utilizado imágenes cartesianas de 320x240 píxeles, como la que podemos observar en la figura 4.4(a). Es importante destacar que dada una imagen cartesiana se pueden obtener multitud de imágenes en formato log-polar, ya que depende explícitamente de dónde situemos el punto de atención máximo (*fóvea*). En la imagen

cartesiana que hemos utilizado para el ejemplo de la figura 4.4, el punto de atención máxima viene reflejado mediante un aspa en color rojo.



(a)

(b)



(c)

Figura 4.4: Representación en coordenadas cartesianas (a), representación en coordenadas log-polar(b) representación cartesiana reconstruida desde la log-polar (vista de retina) (c)

Partiendo de la imagen cartesiana original y sabiendo que el centro de atención lo tenemos ajustado en el aspa rojo, la hemos convertido a coordenadas logpolar para poder representar la imagen como si de una retina biométrica se tratase. Como podemos observar, en la figura 4.4(b) tenemos una imagen de un tamaño de 110x200 píxeles que es el resultado de hacer la transformación a coordenadas *log-polar*, donde el eje horizontal representa el ángulo, y el eje vertical representa el radio. Si nos fijamos veremos que más de la mitad de la imagen en logpolar contiene información de los píxeles de la mano que rodean el aspa rojo. Por el contrario, podemos observar que tanto la pelota como los papeles, que

aparecen lejos en la imagen cartesiana, ocupan un tamaño reducido en la representación log-polar. Cuanto más lejos se encuentre un objeto del punto de atención máxima, menos información tendremos de él en la representación log-polar, y por el contrario, cuanto más cerca se encuentre un objeto del punto de atención máxima, más información tendremos de él en la representación log-polar.

Por último, podemos observar en la figura 4.4(c) la representación en coordenadas cartesianas reconstruida de la imagen en logpolar. En esta imagen se ve con más claridad cómo en la periferia del *ojo virtual*, la imagen aparece muy pixelada (poca información), y en la parte central, donde se encuentra la *fóvea*, la imagen es muy nítida (mucha información).

Con esta nueva representación de la imagen, en log-polar, podemos analizar las imágenes de un modo similar al que lo hacen algunos sistemas biológicos. Podemos realizar un análisis más rápido y eficiente que en coordenadas cartesianas. Y lo más importante, es que esta representación de la imagen nos simula, en cuanto que la fóvea se puede desplazar a voluntad, un sistema de atención estático que se materializa en focos de interés donde prestar la atención en secuencia.

4.3. Control de atención

En la sección anterior hemos contado la manera en que vamos a analizar los puntos de interés simulando la visión humana. En esta sección vamos a explicar qué mecanismos hemos seguido para situar los puntos interesantes dentro de una escena.

Nuestro único *sentido* que tenemos en el sistema para captar la atención, son las imágenes (la *vista*). En la adquisición de una imagen, tenemos *320x240* píxeles que potencialmente pueden llamarnos la atención. Por ello debemos tener un algoritmo que nos indique a qué zonas debemos mirar para procesar esa porción de imagen, y no toda ella. Para calcular la posición del punto al que prestará atención en cada instante hemos desarrollado un algoritmo de atención que usa una dinámica de saliencia. A continuación, en las siguientes secciones pasaremos a comentar cómo la *dinámica de saliencia* permite alterar entre los distintos puntos de atención y saber a qué punto hay que dirigirse en todo momento.

4.3.1. Mapa de saliencia

Saliencia es todo aquello que llama la atención o que sobresale en una situación determinada. Cada punto/píxel de la imagen a tratar tiene asociada una saliencia

determinada y el sistema fija como punto de atención en cada momento el de mayor saliencia, por lo que el sistema decide dónde mirar [Itti and Koch, 2004].

En nuestro sistema de atención, la saliencia indicará qué puntos de atención deben ser visitados y en qué momento. Si tuviéramos un punto de atención con una saliencia muy alta, éste sería visitado muy próximamente. Sin embargo, si la saliencia del punto fuera baja, dicho punto no sería visitado próximamente. A diferencia de otras implementaciones que optaron por utilizar saliencia de objetos concretos ([León Cadahía, 2006], [Martínez De la Casa Puebla, 2005]), en este proyecto hemos optado por utilizar una saliencia general de la escena.

Para poder analizar la saliencia general de una imagen es necesario tener un mapa de saliencia acumulada y que se vaya refrescando en cada iteración. Sobre sus datos tomaremos las decisiones de qué puntos visitar. Para imágenes de 320x240 píxeles tendremos un mapa de saliencia de 320x240 píxeles, donde cada posición en ese mapa nos indica la saliencia acumulada que tiene dicho punto. Es decir, cada píxel de la imagen puede ser el centro de atención en algún momento. Es importante destacar que hemos optado por tener una cámara maestra para el cálculo de la atención. De esta forma, la imagen A de nuestro sistema es sobre la que se calculará el mapa de saliencia instantáneo que será volcado al mapa de saliencia acumulado y sobre este mapa se tomarán las decisiones de a dónde mirar (sobre la imagen B también calcularemos un mapa de saliencia instantáneo que utilizaremos para poder triangular, como veremos en la sección 4.4). El mapa de saliencia acumulado se actualiza acorde a la fórmula siguiente:

$$saliencia_{acc}(objeto(t)) = saliencia_{acc}(objeto(t-1)) * \alpha + (1-\alpha) * saliencia_{inst}(objeto(t)) \tag{4.1}$$

$$saliencia_{acc}(objeto_visitado(t)) = -30{,}0 \tag{4.2}$$

La fórmula (4.1) muestra cómo se refresca la saliencia acumulada de un objeto cuando no es visitado, donde saliencia instantánea indica la saliencia instantánea generada por diferentes criterios de atención (bordes, color, movimiento, cognitivo). La fórmula (4.2) muestra a qué valor se establece la saliencia de un objeto cuándo es visitado. La inhibición de retorno, que explicaremos en la sección 4.3.2, se basa en está fórmula.

En la fórmula (4.1) podemos observar cómo dependiendo del valor que establezcamos a la constante α podremos dar más peso a la información de saliencia instantánea, o

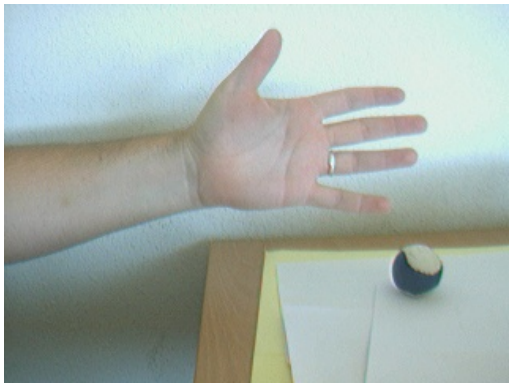
a la información de saliencia acumulada. Dependiendo de este ajuste podemos obtener diferentes comportamientos. Si el valor de α es grande, daremos más peso a la información de la saliencia acumulada, y por tanto la saliencia de ese objeto crecerá más despacio, aunque en un rango de tiempo determinado su saliencia instantánea sea máxima. Por el contrario, si optamos por establecer a α un valor pequeño, daremos peso a la saliencia instantánea y por ello la saliencia acumulada de un objeto crecerá deprisa. Ajustando esta constante α podemos variar la latencia en visitar con la mirada objetos que irrumpen en la escena.

Es importante destacar que aunque en la fórmula (4.1) no exista ninguna *resta*, la saliencia acumulada va disminuyendo hasta llegar a cero debido a que si la saliencia instantánea de una región es 0, la segunda parte de la fórmula siempre es 0. Y además la saliencia acumulada se multiplica por un valor menor que uno, lo que conlleva iteración a iteración a disminuir su valor.

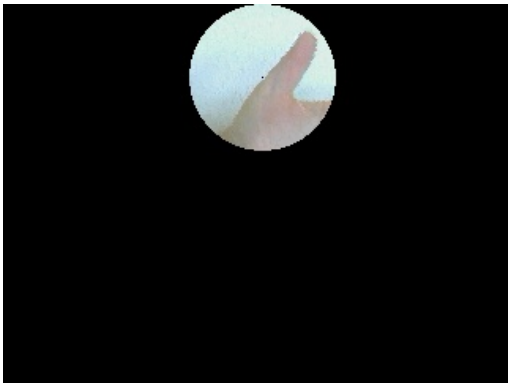
En nuestra implementación hemos optado por utilizar un valor alto *(α=0.9)*. Con ello conseguimos que un objeto sea saliente debido a que lleva varias iteraciones mostrándose como interesante. Además, de este modo, un objeto que aparece y desaparece rápidamente de la escena, no nos llamará la atención notablemente. En nuestra implementación hemos utilizado un rango de saliencia de *0-255*, donde los valores cercanos a 0 no muestran interés, y los valores cercanos a 255 indican gran interés.

Una vez relleno el mapa de saliencia acumulado podremos tomar decisiones de hacia qué punto mirar. Para ello barremos el mapa de saliencia acumulado de arriba hacia debajo, y de izquierda a derecha y elegimos como punto saliente aquel que tenga la saliencia máxima y esté por encima de cierto umbral. Este umbral lo utilizamos para evitar ruidos del análisis de la escena, así nos aseguramos que los puntos cuya saliencia está por encima del umbral son puntos que realmente nos llaman la atención. En el caso que varios puntos empataran a saliencia máxima no es relevante cuál de ello se visita primero. Por convenio elegiremos el primer punto que encontramos siguiendo el barrido de la imagen explicado anteriormente.

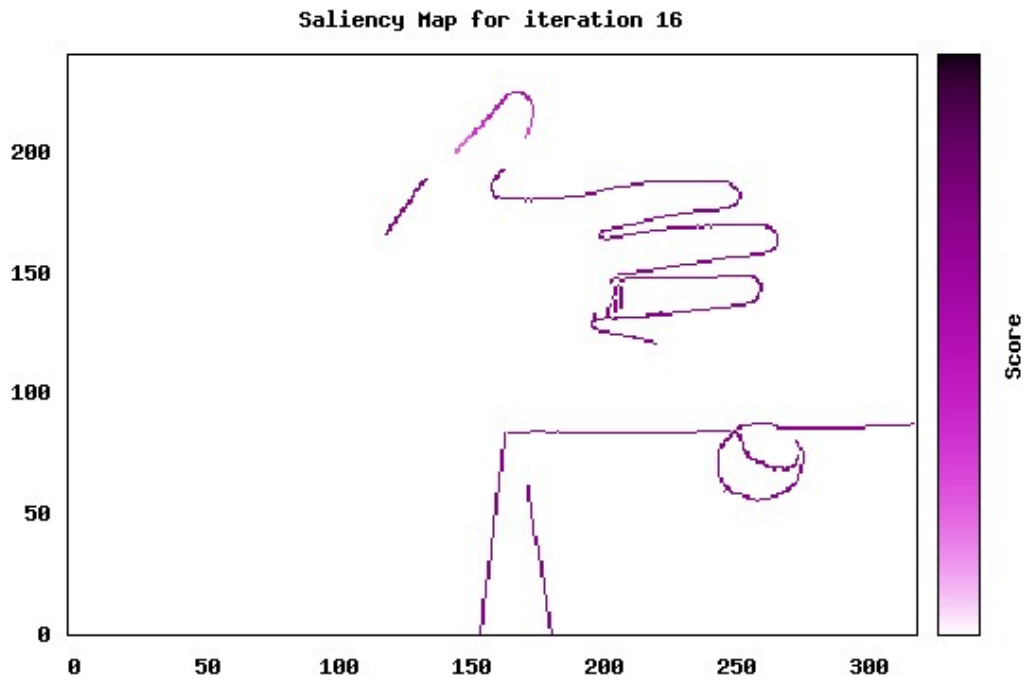
En el conjunto de imágenes de la figura 4.5 podemos observar el estado de la saliencia y atención en una iteración en concreto. Partimos de la imagen original (4.5(a)) tomada por la cámara para analizar y obtener los puntos de atención en esta iteración. El sistema de atención está configurado para que le llamen la atención únicamente los bordes. En esta iteración en concreto, el sistema decide que tiene que mirar cerca del dedo más alto, como se puede observar en la imagen retinal (4.5(b)). Esta acción desencadena la actualización del mapa de saliencia acumulado, que establecerá la mínima saliencia en la zona donde acaba de mirar. El mapa de saliencia resultante de esta iteración lo podemos ver en la



(a)



(b)



(c)

Figura 4.5: Tres imágenes en la misma iteración: Imagen original (a), imagen en vista de retina sobre el punto de atención (b) y mapa de saliencia por bordes.(c)

imagen 4.5(c), que lo hemos representado como un *mapa de temperatura*, donde los valores con mucha atención tendrán un color cercano al negro, y los valores con poca o nada de atención tendrá un color cercano al blanco.

Como podemos observar también en la imagen 4.5(c) la zona que queda por debajo del punto de atención en esta iteración es más oscuro que la zona que queda por encima. Esto es debido a que la zona superior ya ha sido visitada y su saliencia se estableció a 0 y las demás zonas atentivas han aumentado su saliencia. Iteración a iteración esta zona visitada irá aumentando su saliencia ya que los bordes llaman la atención al sistema.

4.3.2. Inhibición de retorno (IOR)

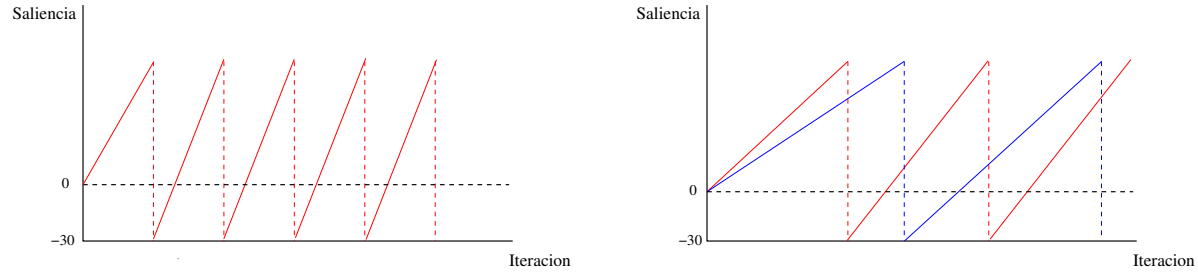
Es conocido por estudios de biología [Itti and Koch, 2005b], que cuando el ojo humano responde a un estímulo que aparece en una posición que ha sido previamente atendida, el tiempo de reacción suele ser mayor que cuando el estímulo aparece en una posición nueva; este efecto se conoce como *inhibición de retorno* (al lugar).

Traducido a nuestro proyecto, viene a ser que un punto que provoca atención y que anteriormente no ha sido visitado, crecerá su saliencia de una manera más rápida que un punto que provoca atención y que anteriormente sí ha sido visitado.

En la figura 4.6(a) podemos observar cómo va variando la saliencia en un escenario donde únicamente tenemos un punto de atención. En el estado inicial su saliencia es *0* e iteración a iteración va aumentando su saliencia acumulada hasta que llega a un máximo establecido y por tanto es visitado. Como consecuencia de ser visitado, su saliencia baja a *-30.0* inmediatamente (como ya mostramos en la fórmula 4.2) y deja de tener interés para el sistema. La razón de establecer la saliencia a un valor negativo se debe a que si en ese preciso instante aparece un objeto interesante en la escena, el sistema de atención prestará interés antes por el objeto nuevo que por el objeto recién visitado.

La gráfica que interpreta la saliencia de dos puntos la podemos observar en la figura 4.6(b). Aquí podemos observar una característica muy importante, y es que la saliencia no crece igual de deprisa para todos los puntos, depende directamente de la saliencia instantánea de los puntos. Como vemos, el punto representado por la línea roja requiere más atención del sistema, y por ello lo visita antes que el punto representado por la línea azul.

En este proyecto, hemos optado por introducir la inhibición de retorno ya que nos permite repartir la mirada entre varios objetos interesantes y además nos evita tener



(a) (b)

Figura 4.6: Comportamiento de saliencia e IOR con: un solo objeto (a), y con dos objetos (b).

hambruna. Cuando se visita el punto y establecemos su saliencia a *-30.0*, también establecemos la saliencia a *-30.0* de los píxeles próximos. Concretamente hemos establecido que los *píxeles próximos* son una ventada de 22x22 píxeles cuyo centro es el punto de atención que hemos visitado. La razón de realizar este paso adicional, es que no tiene sentido volver a visitar la zona de vecinos.

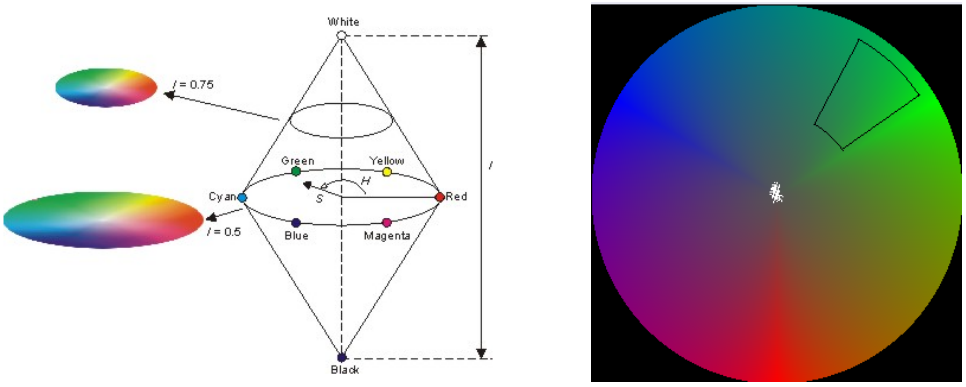
4.3.3. Características ascendentes de atención

Como hemos comentado a lo largo de la sección anterior, el sistema de atención se puede modular y configurar para determinar qué características le llaman la atención a partir del análisis directo de las imágenes (características ascendente o *bottom-up*). Incluso pueden estar varias características activadas al mismo tiempo. A continuación explicaremos las tres características ascendentes que hemos implementado y que utilizan las características de la imagen como fuente de información para generar las zonas de atención interesantes: color, bordes y movimiento.

Característica de color: HSV

El color siempre ha sido una característica de atención para el ser humano. De hecho siempre atendemos más la parte de imágenes con colores llamativos y de alto contraste, que a las partes de bajo contraste. Por ello, hemos decidido implementar esta característica en nuestro sistema de atención. Partiendo de las imágenes que nos ofrecen las cámaras en el espacio de color RGB vamos a convertirlas al espacio de color HSV usando la librería *libcolorspaces* que ya comentamos en la sección 3.3.2.

El espacio RGB es muy sensible a los cambios de iluminación del entorno. Esto es un problema para el análisis ya que el hecho de tener dos cámaras implica que cada una recibe la luz de una manera diferente, y por tanto objetos del mismo color en la realidad aparecen de distinto color en las imágenes. Las pruebas que se realizaron confirmaron que utilizar el espacio RGB para realizar un filtro no es una buena solución puesto que es bastante sensible en los cambios de iluminación y contraste.



(a) (b)

Figura 4.7: Modelo cónico (a) y disco de color (b) del espacio de color HSV.

El modelo de color HSV fue diseñado teniendo en mente el modo en que artistas y diseñadores gráficos usan los términos de saturación (pureza del color), tono (el color en sí mismo) e intensidad (brillo del color). Este modelo se representa geométricamente en un cono, como se muestra en la figura 4.7(a). En el modelo HSV, las componentes HS se mantienen aproximadamente invariables frente a los cambios de iluminación que se producen en el entorno. Estos cambios de iluminación quedan absorbidos por la componente *V*. La gran desventaja de este espacio de color es que la mayoría de las tarjetas digitalizadoras actuales no entregan las imágenes en este formato y esto conlleva cierta carga computacional para convertir las imágenes de la cámara a este espacio de color.

En este caso, la imagen nos viene en formato RGB directamente desde las cámaras firewire por lo tanto estamos obligados a realizar el cambio de espacio de colores al HSV. Este cambio lo realizamos teniendo en cuenta las fórmulas siguientes:

$$H = cos^{-1}\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)*(G-B))}}$$

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

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

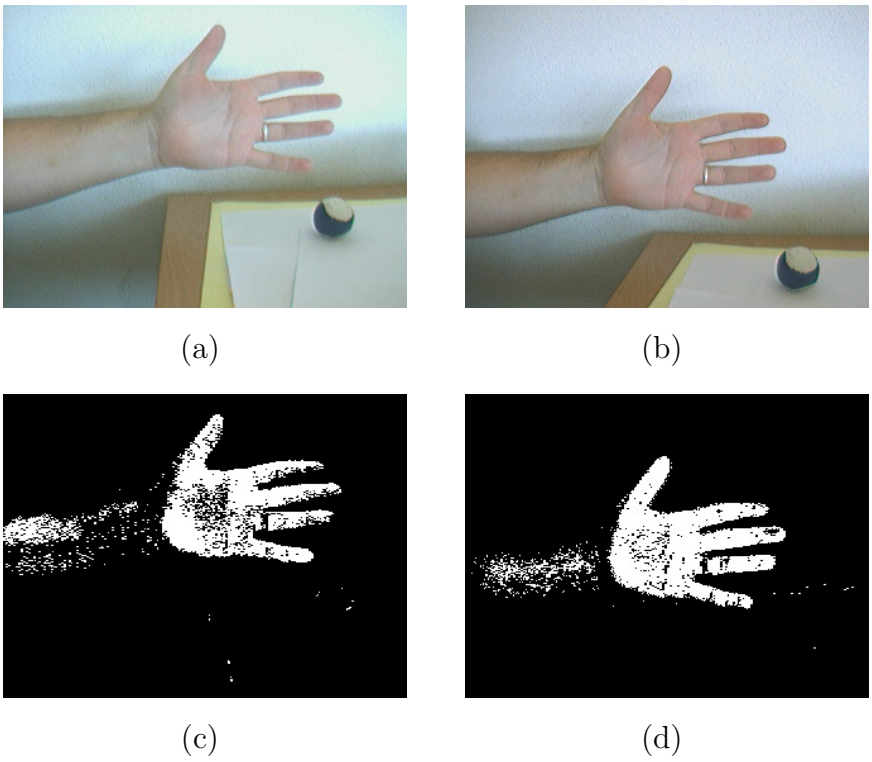
La intensidad (V) y saturación (S) están normalizadas (entre cero y uno) y el tono (H) está entre 0 y 360 grados. La función que aparece en el calculo de la S, *min(R,G,B)* selecciona el menor valor de entre los tres.

Hay que tener en cuenta que en ciertas ocasiones, dependiendo de los valores R,G y B, la fórmula no se aplica tal cual. Hay casos que debemos contemplar como por ejemplo cuando los tres valores son iguales a cero. El siguiente código representa dicho cambio en un píxel:

```
if (((R-G)*(R-G)+(R-B)*(G-B))<=0)
  H = -1;
else
  H = acos ((0.5*((R-G)+(R-B))) / sqrt((R-G)*(R-G)+(R-B)*(G-B)));
if ((R+G+B) == 0.0)
  S=1.0;
else{
  S = 1.0 - (3.0/(R+G+B)) * min(R,G,B);
  V = (1/3)*(R+G+B);
}
H = H*RADTODEG;
```

Podemos observar lo explicado anteriormente en la figura 4.8, que muestra las imágenes que capturan las cámaras (imágenes 4.8(a) y 4.8(b)) y como son filtradas por color en el espacio HSV con la intención de filtrar únicamente la mano. El resultado del filtro lo podemos observar en las imágenes 4.8(c) y 4.8(d)). Las zonas blancas de estas imágenes son aquellas que han pasado el filtro y dichas regiones son las que potencialmente llamarán la atención del sistema por su color.

Para definir el "color objetivo" al que el sistema de atención presta interés es necesario picar sobre un píxel de la imagen. Una vez capturado los valores H y S de color de ese píxel, se le aplica una tolerancia por encima y por debajo, creando un umbral para la detección de color.



(a) (b)

(c) (d)

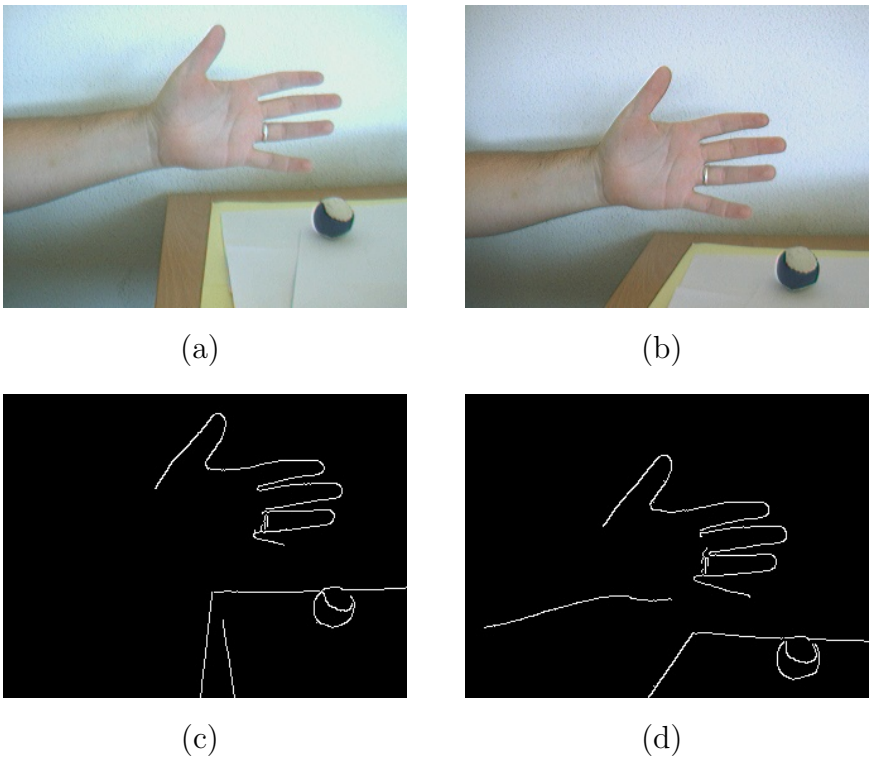
Figura 4.8: Imágenes obtenidas del par estéreo (a) y (b). Filtro de color HSV para el color de la mano (c) y (d)

Característica de bordes

Los bordes o esquinas siempre han tenido una gran importancia a la hora de realizar filtrado de imágenes, ya que nos dan una aproximación bastante acertada del entorno representado mediante segmentos. Los bordes se puede interpretar como *puntos de alta derivada espacial en luminancia* y contienen mucha información de la imagen. Los bordes cuentan donde están los objetos, su forma, su tamaño, y también sobre su textura. Los ejes o bordes se encuentran en zonas de una imagen donde el nivel de intensidad fluctúa, cuanto más bruscamente se produce el cambio de intensidad, el eje o borde es más fuerte.

Un buen proceso de detección de bordes facilita la elaboración de las fronteras de objetos con lo que, el proceso de reconocimiento de objetos se simplifica. Para poder detectar los bordes de los objetos, debemos detectar aquellos *puntos borde* que los forman.

Para la obtención de bordes hemos optado por utilizar la librería de visión artificial *opencv* que ya comentamos en la sección 3.2.6. Entre las diferentes funciones que ofrece este API para la detección de bordes, hemos utilizado la función `cvCanny` por su simplicidad y buen funcionamiento. A continuación se detalla dicha función:



(a) (b)

(c) (d)

Figura 4.9: Imágenes obtenidas del par estéreo (a) y (b). Filtro de bordes (c) y (d)

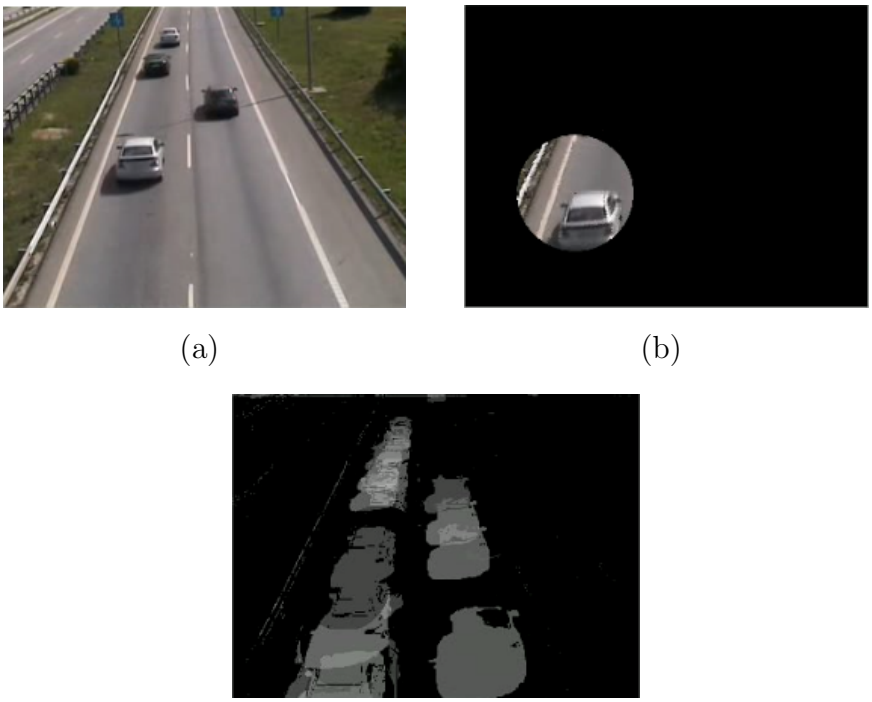
```
CVAPI(void)  cvCanny( const CvArr* image, CvArr* edges, double threshold1,
                      double threshold2, int  aperture_size CV_DEFAULT(3) );
```

Al igual que vimos con la característica de color, en la figura 4.9 podemos observar el resultado de realizar el filtro de bordes sobre el par estéreo de las cámaras. Como es de esperar, los puntos que pasan el filtro son los que tomará en cuenta el sistema de atención para decidir a dónde mirar con más atención.

Característica de movimiento

El movimiento siempre ha tenido un papel importante en sistema de visión, sobre todo en sistemas de vídeo-vigilancia. Siempre es útil saber qué se está moviendo en una escena. El movimiento se puede interpretar como *puntos de alta derivada temporal en luminancia.*

Para la obtención del movimiento hemos utilizado una técnica simple y eficaz computacionalmente que es la resta de píxeles. Realizando la resta de píxeles entre imágenes consecutivas podemos saber qué ha cambiado entre dichas imágenes y por tanto estimar que en esa zona hay movimiento.



(a) (b)

(c)

Figura 4.10: Imagen de un vídeo (a), su representación en vista de retina (b) y su mapa de saliencia (c)

En la figura 4.10 se muestra un vídeo (4.10(a)) en el que pasan los coches continuamente. En una iteración dada, vemos cómo el mapa de saliencia va adoptando valores altos para los objetos que se mueven (figura 4.10(c)), y cómo el sistema presta atención y *mira* al objeto que estima oportuno (figura 4.10(b)) según su movimiento.

4.3.4. Características descendentes de atención

Las características descendentes o *top-down* son aquellas características de atención que no basan directamente en la imagen el cálculo de las zonas de interés. En este proyecto hemos incluido dos tipos de características *top-down*, una de ellas se encarga de volcar al mapa de saliencia acumulado puntos de interés exploratorios con el fin de visitar zonas de la imagen que no se visiten mediante las características ascendentes. La segunda característica descendente que hemos incluido es una característica cognitiva que es capaz de realizar hipótesis de dónde dirigir la mirada en una situación determinada.

Característica exploratoria

Gracias al mapa de saliencia unido a un mapa de "zonas menos visitadas", podemos determinar fácilmente qué zonas o regiones son las menos visitadas, o las que nunca se han visitado, y mediante un sencillo algoritmo realizar exploraciones de puntos aleatorios a dichas zonas.

Esta característica es realmente útil ya que por ejemplo, si en una escena aparece un nuevo objeto que provoca atención, es posible que gracias a la "exploración de puntos aleatorios menos visitados" podamos observar otras zonas de la imagen.

Como detalle de implementación, cada 10 iteraciones el sistema de atención mira en el mapa de dinámica de "zonas menos visitadas", calcula todos los puntos menos visitados y realiza un elección aleatoria. El punto resultante pasará a ser el punto de atención con mayor saliencia en esa iteración, y por tanto será el siguiente punto donde el sistema posará la atención.

Característica cognitiva

La característica cognitiva supone un gran paso adelante en este proyecto. Todas las características ascendentes se basan en información que posee la imagen. Nosotros hemos querido darle un matiz de inteligencia al proyecto añadiendo un caso concreto en el cuál el sistema de atención pueda hipotetizar sobre dónde mirar, basado en conocimiento de "alto nivel", y posteriormente validar la hipótesis.

De esta manera intentamos simular un comportamiento humano, ya que muchos de los movimientos de atención que realizar el ser humano provienen de una capa de alto nivel y no de las imágenes. Dirigir la mirada por la ventana para ver el tiempo que hace, o mirar la parte inferior de un semáforo esperando que éste se ponga en verde, son ejemplos de atención cognitiva del ser humano.

Como veremos en el capítulo de experimentos, hemos realizado un experimento (5.3.2) que se basa en la saliencia cognitiva del sistema. El experimento analiza una imagen que contiene un cuadrado incompleto, faltándole una de sus cuatro esquinas. La característica de saliencia cognitiva evaluará las posiciones de las otras tres esquinas, y realizará una hipótesis de dónde puede encontrarse la cuarta. Esta hipótesis de dónde se encuentra la cuarta esquina la materializará volcando en el mapa de saliencia una región con alto interés para el sistema, que es la zona en la que se espera la cuarta esquina.

4.3.5. Implementación

Hemos querido dedicar una sección a la parte de implementación del sistema de saliencia en *C++*, ya que hemos empleado un tiempo importante en ajustar su diseño y rendimiento.

En la figura 4.11 podemos observar el diagrama de clases del sistema de atención. Hemos optado por crear un *manager* que se encargue de todo lo relacionado con la atención. Así tenemos la clase *saliencyManager* que la hemos implementado como un *singleton*[1], para mayor comodidad a la hora de usarlo. Esta clase es la encargada de calcular los mapas de saliencia instantáneos y el acumulado, dependiendo de las características que se le hayan configurado. Esta clase únicamente maneja instancias de la clase *genericSaliency*, cuya interfaz ofrece un método virtual que es el encargado de analizar la imagen y establecer la saliencia dependiendo del interés. En nuestro proyecto hemos implementado tres tipos de saliencia ascendente y dos tipos de saliencia descendentes, que quedan plasmadas en las clases *colorSaliency*, *edgeSaliency*, *movementSaliency*, *explorerSaliency* y *cognitiveSaliency* que heredan de la clase *genericSaliency* e implementan el método virtual *calculateSaliency*, que es el encargado de obtener la saliencia instantánea en cada iteración.

4.4. Visión 3D

Una vez que el sistema de atención nos sitúa los puntos de interés en la imagen, el sistema obtiene su posición en 3D en la escena. El sistema de reconstrucción 3D generará una secuencia de puntos 3D provenientes de la serie de puntos de interés que genera el sistema de atención. Para ello es necesario realizar numerosos cálculos geométricos y seguir los mecanismos clásicos de cálculo de profundidad desde pares estéreos. Estos pasos son: obtención de puntos relevantes, cálculo de homólogos y triangulación.

Como ya explicamos en la sección 3.2.3 hemos utilizado la librería *progeo*, que nos ofrece las herramientas necesarias para trabajar en un mundo 3D. Para que todos los cálculos sean correctos, a partir de este punto trabajamos con las cámaras correctamente calibradas gracias al esquema calibrador que ya explicamos en la sección 3.2.4.

Cualquier algoritmo de reconstrucción 3D que se base en la triangulación necesita conocer la posición de las proyecciones de un punto 3D en cada cámara del par. Con el proceso de atención y filtrado comentados en la sección anterior obtenemos los conjuntos de puntos interesantes de cada imagen.

[1] http://es.wikipedia.org/wiki/Singleton

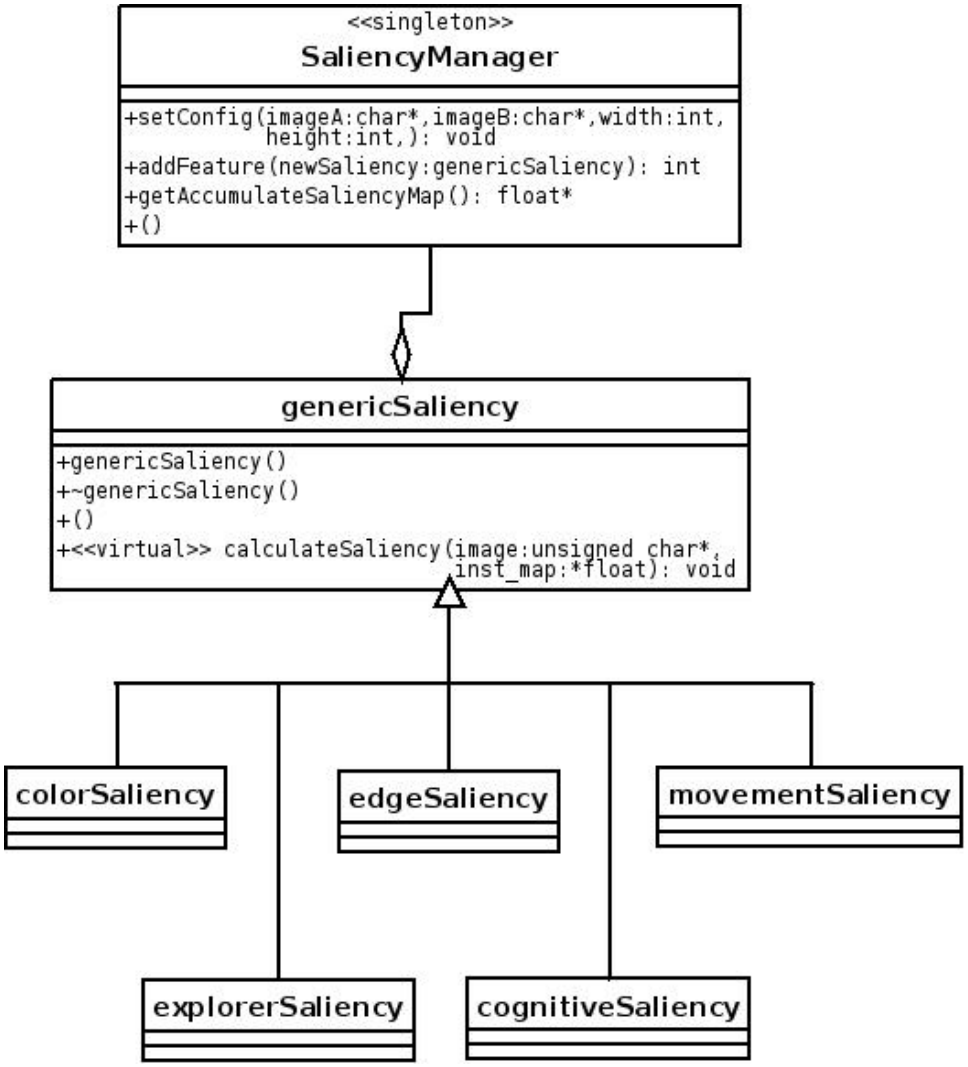


Figura 4.11: Diagrama de clases del sistema de saliencia.

4.4.1. Cálculo de puntos homólogos

Para emparejar cada píxel con su homólogo utilizaremos la técnica de correlación [S. Birchfield, 1999]. La correlación consiste en coger el entorno cercano al píxel a emparejar (su vecindad o ventana de correlación) y buscar en la imagen homóloga el mejor encaje de dicha ventana entre los puntos interesantes y vecindades.

Si por cada punto de atención de la imagen-A tuviéramos que comparar el parche generado con todos los puntos que componen la imagen B tendríamos un tiempo de cómputo excesivo para obtener un sólo punto homólogo. Por ello, y para mejorar el tiempo de cómputo, hemos añadido dos restricciones: el punto homólogo debe situarse cerca de la *línea epipolar* y además debe ser un punto de interés en el sistema. Estas restricciones han sido claves para obtener un comportamiento en tiempo real.

Restricción epipolar

La restricción epipolar consiste en limitar la búsqueda del punto homólogo sobre el *epipolo* de la imagen homóloga. En la figura 4.12 podemos ver el método para obtener la recta epipolar. Dado un punto P en la cámara A (izquierda), proyectamos una línea que pasa por el foco de A (Fa) y por el punto P. Sobre esta recta de proyección situamos dos puntos cualesquiera Q' y R'. Dichos puntos los proyectamos hacía la cámara B, y dónde estas proyecciones corten al plano imagen, obtendremos los puntos Q y R. Estos dos puntos sabemos que pertenecen a la recta epipolar, por tanto ya podemos dibujar la recta e. Así, dado el punto P en la imagen A podemos concluir que su homólogo estará en el conjunto de puntos que componen la recta epipolar e en la imagen B.

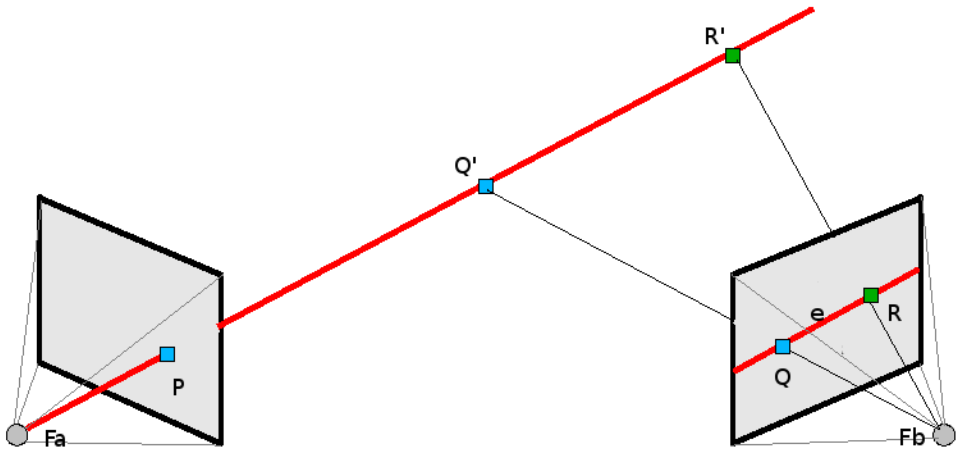


Figura 4.12: Restricción epipolar.

En la imagen 4.13, podemos ver la aplicación de la teoría anterior sobre dos imágenes reales. En la imagen A seleccionamos un punto de atención (cuadrado rojo), y obtenemos en la imagen B la recta epipolar (recta blanca) que nos indica que en esa recta está el homólogo que buscamos. Es importante que destaquemos que todos estos cálculos y operaciones las hemos realizado con la librería *progeo* y que únicamente se obtienen buenos resultados si las cámaras están correctamente calibradas.

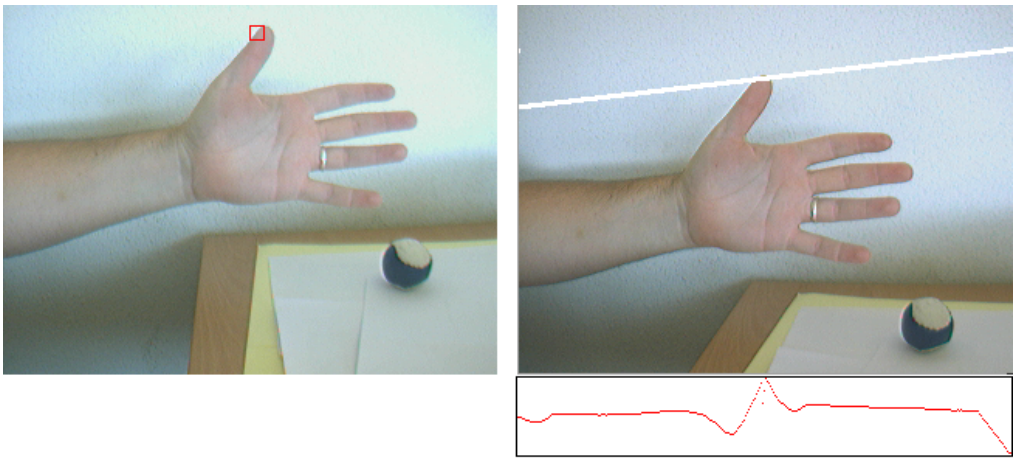


Figura 4.13: Restricción epipolar aplicada a dos imágenes reales

Aunque la calibración de las cámaras la realizamos con mucha exactitud, nunca la calibración es perfecta. Que ésta no sea perfecta implica que la línea epipolar no se sitúe realmente donde debería estar. Por ello hemos creado una banda alrededor de la línea epipolar que se sitúa dos píxeles por encima y dos por debajo. De esta forma el análisis de la epipolar queda: Dado un punto P de la epipolar generamos una ventana de 5x5 píxeles con centro dicho punto P, donde calcularemos el parche de comparación de 45x45 únicamente en los píxeles en los que exista saliencia instantánea. El de mayor correlación de todos ellos se tomará como ganador, y se incluirá en la lista de posibles candidatos a ser el punto homólogo. Esta acción se repetirá para todos los puntos de la epipolar, generando finalmente una lista de puntos candidatos, cuyo ganador será el que más se asemeje al parche de la imagen maestra.

Restricción atentiva

Para obtener un comportamiento más vivaz y rápido tenemos que reducir los puntos sobre los que crear el parche y compararlo con el de la imagen maestra. Ésto lo vamos a conseguir utilizando el sistema de atención. Utilizaremos los mapas de saliencia

instantáneos para únicamente generar el parche en aquellos puntos que pertenezcan a la epipolar y además tengan saliencia instantánea. Como podemos observar en la figura 4.14 las dos imágenes están representando la saliencia instantánea que hay en esta iteración. En este caso la recta epipolar pasa por un conjunto de puntos donde existe saliencia, y es sólo en estos puntos donde se genera y calcula el parche. De hecho en la gráfica que representa la *correlación*, podemos observar cómo la saliencia es 0 para los puntos de la recta epipolar que no tienen saliencia, ya que para estos puntos no llegamos a calcular el parche, con el consiguiente ahorro computacional. Concretamente para éste ejemplo analizamos 20 posibles candidatos en lugar de los 320 totales. En la sección 5.4 veremos la mejora notable que se consigue con esta técnica.

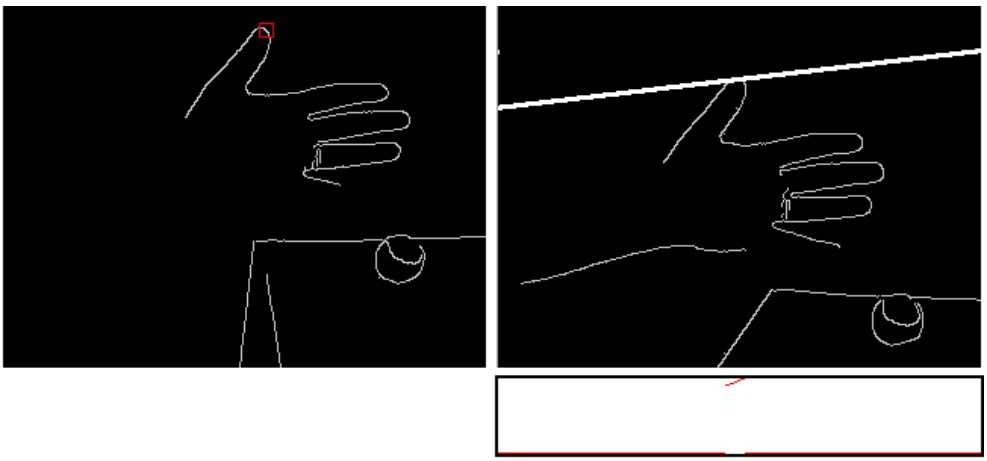


Figura 4.14: La recta epipolar y la saliencia determinan donde buscar el punto homólogo.

Franja epipolar

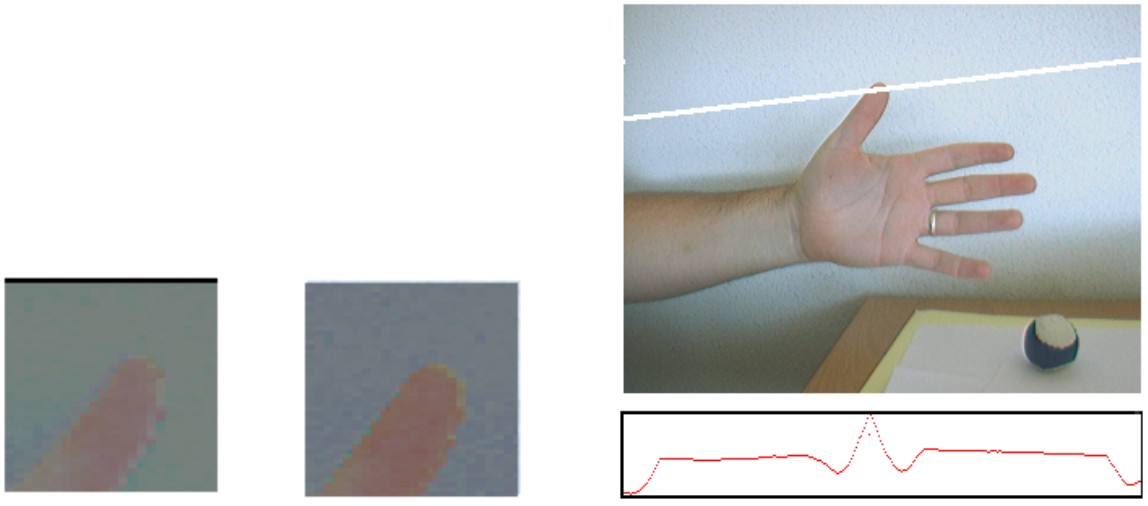
Aplicando la restricción epipolar, tenemos que para el cálculo de un punto homólogo es necesario realizar la comparación del parche por cada píxel que compone la recta epipolar. Sabiendo que el parche que generamos es de 45x45 píxeles y que la epipolar contiene 320 píxeles, obtenemos una multitud de operaciones de cómputo para la obtención de un único punto homólogo. Como podemos observar en la imagen 4.13, se ha realizado un emparejamiento comparando todos los parches correspondientes a los píxeles que componen la epipolar, dando como resultado la gráfica de correlación que se observa en la parte inferior de dicha imagen. Como veremos en la sección 5.4, esto resulta todavía muy costoso y el realizar el emparejamiento utilizando todos los puntos de la línea epipolar ralentiza el sistema.

Correlación de puntos mediante parches cuadrados en HSV

La correlación de puntos de interés la hemos realizado creando parches cuadrados de 45x45 píxeles tomando como centro dicho punto de interés. De esta manera tanto el punto de atención como su entorno se tendrán en cuenta para decidir el emparejamiento. Para realizar la comparación de parches en este espacio de color realizaremos la resta absoluta de parches, píxel a píxel, excluyendo la componente V, que representa la luminosidad de la imagen, tal como muestra la formula 4.3. Debido a que las componente H tiene un rango entre 0 y 360 y las componente S tiene un rango entre 0 y 1, hemos añadido las constantes α y β para normalizar y que las dos componentes tengan el mismo peso de decisión.

$$Correlacion_{ParcheAB} = \sum_{p=0}^{TamParche} \alpha * |(H_{pa} - H_{pb})| + \beta * |(S_{pa} - S_{pb}) \qquad (4.3)$$

El correlación del parche HSV consiste en comparar parches cuadrados de 45x45 píxeles en el espacio de color HSV, reduciendo así los errores debidos a la diferencia de luminosidad, como ya explicamos en la sección 4.3.3. La figura 4.15(a) representa los parches en el espacio de color HSV. Es una buena aproximación para poder realizar comparaciones cuando las imágenes tienen diferente intensidad de luz. Para calcular el parecido entre estos parches, también lo hemos realizado haciendo la resta de parches, píxel a píxel, en valor absoluto.



(a) (b)

Figura 4.15: Parche máster ((a)-izquierda), parche homólogo ((a)-derecha) e imagen homóloga junto con la función de parecido (b).

La función de correlación que observamos en la figura 4.15(b) es la mejor obtenida de todas nuestras pruebas. Esta función de correlación o parecido la hemos analizado también desde el punto de vista de otros espacios de color e intensidad y utilizando parches circulares. En el experimento (5.5) se detallarán estas otras alternativas y se razonará el porqué hemos optado por utilizar parches cuadrados en HSV.

4.4.2. Triangulación

Una vez que hemos realizado el emparejamiento correctamente mediante el algoritmo de correlación, disponemos de las proyecciones de un mismo punto 3D desde las dos cámaras. Las rectas de retro-proyección de cada cámara se cortarán en la posición asociada a ese punto 3D.

Conocida la posición en 3D del foco de la cámara y apoyándonos en la función *retro-proyección* de Progeo, que a partir de un punto proyectado y los parámetros de la cámara devuelve un punto 3D contenido en la recta de retro-proyección (recta 3D que desemboca en ese punto 2D en el plano imagen), podemos definir la recta de retro-proyección asociada a cada cámara. Resolviendo el sistema de ecuaciones que se plantea con las dos rectas obtenemos la posición 3D correspondiente a ese par de píxeles emparejados.

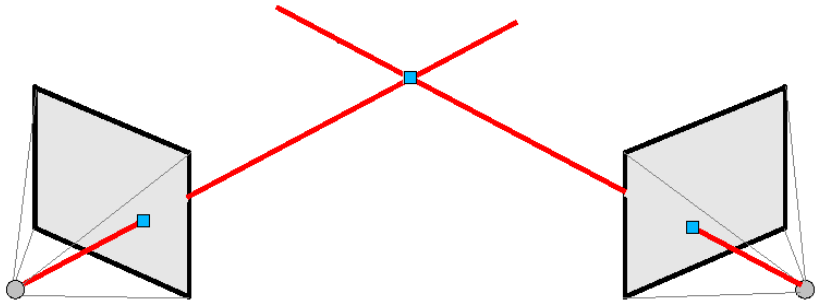


Figura 4.16: Triangulación ideal con corte de las rectas.

Como anteriormente comentamos, los parámetros intrínsecos y extrínsecos de las cámaras pueden no ser exactos debido a que la calibración nunca es perfecta. Ésto puede producir que las rectas de proyección obtenidas no se corten sino que se crucen. En un entorno ideal obtendríamos una situación como la que se muestra en la figura 4.16. Pero en las pruebas que hemos realizado, casi nunca se cortan exactamente las rectas en el espacio 3D, por ello en vez de calcular el punto de corte, calculamos el punto medio del segmento de distancia mínima entre las dos rectas. Utilizando esta técnica obtenemos una buena aproximación que vemos reflejado en la figura 4.17.

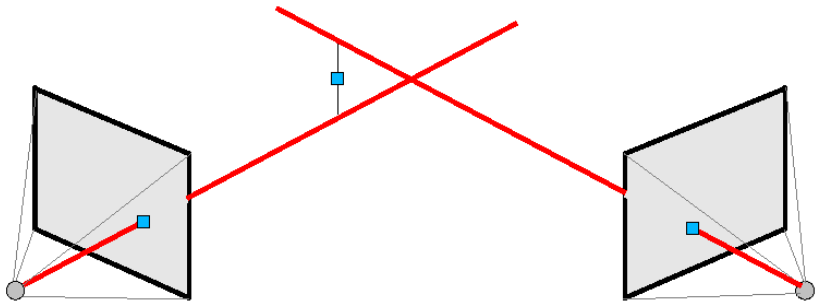


Figura 4.17: Triangulación mediante cruce de las rectas.

Es importante destacar que hemos utilizado la *geometría proyectiva planar* para realizar los cálculos de todas las operaciones relacionadas con puntos, rectas y planos en 2D. Gracias a esta geometría proyectiva las operaciones geométricas se resuelven de una manera fácil y eficiente, contemplando todos los casos posibles sin excepciones explícitas.

A partir de este momento, tendremos almacenados todos los puntos 3D que hemos obtenido de realizar los emparejamientos de los puntos que ha ido obteniendo del sistema de atención visual. Con estos puntos 3D podemos realizar diferentes acciones: representarlo en 3D, analizarlos y tomar decisiones respecto a ellos, etc.

4.5. Interfaz gráfica de la aplicación

La interfaz gráfica del esquema atentivo está implementada con la librería *xforms* y apoyándonos en la arquitectura de *JDE*. Además de permitirnos depurar y mostrar los resultados del comportamiento, la interfaz permite al usuario interactuar y ajustar diferentes opciones del esquema.

La idea básica de tener una interfaz (a parte de poder mostrar las imágenes en nuestro caso), es que posibilite en tiempo de ejecución probar distintas configuraciones del comportamiento. Con esta idea hemos dotado de numerosos botones y funcionalidad, como la de activar o desactivar técnicas de emparejamiento, parches, tamaño del parche, mover las cámaras virtuales, activar o desactivar diferentes características de saliencia.

En la figura 4.18 podemos observar una vista general del interfaz. Se ha hecho mucho énfasis en parametrizar muchas características para poder estudiar ciertos comportamientos y razonar los algoritmos elegidos. A continuación, y con más detalle, pasamos a comentar las partes más importantes de la interfaz gráfica.

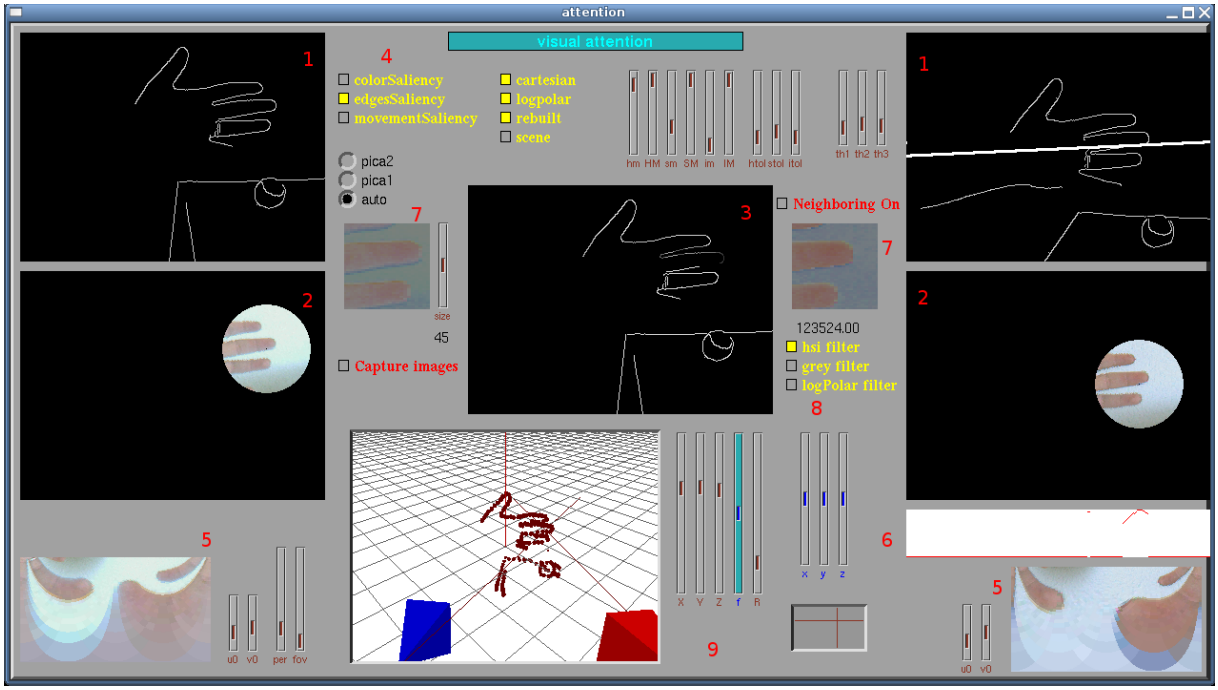


Figura 4.18: Interfaz general del esquema.

1. Captura de las imágenes de la cámara. En tiempo de ejecución del comportamiento se representa la saliencia instantánea de cada cámara si existe alguna característica activada, o la imagen de entrada en cualquier otro caso.

2. Representación en vista de retina de la imagen logpolar. Además sitúa el ojo virtual en el punto de atención.

3. Representa la saliencia acumulada.

4. Diferentes características que llaman la atención al sistema: color, bordes o movimiento. Pueden ejecutarse individualmente o en conjunto.

5. Representación en logpolar de la zona de atención.

6. Gráfica de correlación de los parches a lo largo de la franja epipolar.

7. Parches correspondientes al punto de atención y a su homólogo encontrado.

8. Algoritmo utilizado para la comparación de los parches: RGB, HSV, luminancia y logpolar.

9. Representación en 3D de la escena y ajustes de la cámara virtual.

Representación en openGL

Uno de los aspectos más interesantes de la interfaz gráfica desarrollada es poder visualizar la escena 3D reconstruida y comprobar, al observarla, si se está realizando bien la representación. Con este fin se dotó a la interfaz gráfica de una cámara virtual que se sitúa en la escena reconstruida por el esquema. Para esta representación hemos utilizado la librería OpenGL que ya explicamos en la sección 3.2.6. OpenGL permite representar multitud de figuras y formas, nosotros hemos optado por pequeñas esferas para representar los puntos de atención y pirámides de color para representar las propias cámaras del par estéreo.

De esta forma hemos podido dibujar en la imagen virtual todas las partes de la escena que tenemos en 3D y cualquier objeto o lugar de la realidad. Otra de las ventajas de usar esta cámara virtual es que se puede mover su posición y orientación colocándola en el lugar que más interese y poder comprobar si se está calculando correctamente la posición de los puntos de atención. Por ejemplo, en la figura 4.19 podemos observar diferentes vistas desde la cámara virtual de la mano representada en 3D. También se pueden observar los rayos de proyección que salen de la representación en 3D de las cámaras reales.

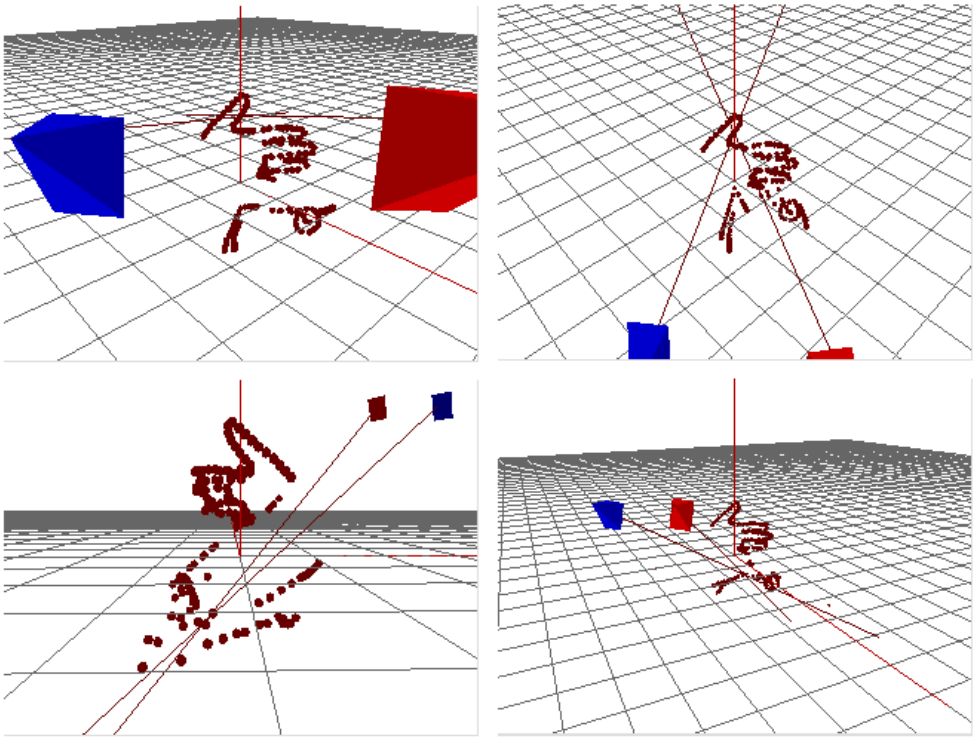


Figura 4.19: Diferentes vistas en 3D de la mano.

Capítulo 5

Experimentos

Una vez que hemos abordado e implementado un conjunto de soluciones al problema de reconstrucción 3D visual atentiva, vamos a poner a prueba el funcionamiento del sistema y comprobar que cumple correctamente el objetivo principal: representar en 3D las zonas de atención que el sistema detecta mediante imágenes reales. Para ello hemos realizado una serie de pruebas experimentales, que nos han ayudado a validar los algoritmos desarrollados y poder caracterizarlos correctamente. Por ejemplo mostraremos el rendimiento temporal del sistema aportando tiempos de cómputo.

Además, incluimos varios experimentos con implementaciones alternativas que han ayudado a afinar y ajustar el algoritmo final y elegirlo bien entre las opciones probadas, como por ejemplo la técnica de correlación y tamaño del parche.

5.1. Sistema de atención

En esta sección ponemos a prueba al sistema de atención. Mostraremos cómo se comporta cuando en una escena estática debe prestar atención a varios objetos a la vez. También veremos cómo se comporta cuando aparecen nuevos objetos interesantes. Además mostraremos el comportamiento del sistema de atención en una escena dinámica con mucho movimiento.

5.1.1. Reparto de mirada con irrupción de un nuevo objeto

En este primer experimento sobre el sistema de atención vamos a comprobar el correcto funcionamiento de éste y cómo es capaz de comportarse ante escenas que muestran cambios

puntuales.

Para este experimento utilizaremos la imagen de partida de la figura 5.1 y configuraremos el sistema de atención para que muestre interés por los bordes (activando sólo la saliencia por derivada espacial en luminosidad). La intención de este experimento es doble: ver cómo reparte la atención entre varios objetos y cómo afecta al sistema de atención la incorporación de un nuevo elemento en la escena. Veremos cómo el sistema de atención analiza los objetos que muestran interés en la imagen de la figura 5.1, y cómo es capaz de mantener la atención sobre éstos y sobre los nuevos objetos de atención que aparecen en la imagen.

Figura 5.1: Imagen original sobre la que el sistema de saliencia trabaja en este experimento.

Como podemos observar en la figura 5.2, el sistema de atención va generando iteración a iteración el mapa de saliencia acumulado de la escena. Se puede observar cómo en la iteración 0, el sistema de saliencia aún no tiene información suficiente para saber si los objetos salientes son de interés o no (recordamos que ésto lo deduce cuando el valor de saliencia no supera un umbral determinado, por eso el foco de atención está en su posición inicial, el centro de la imagen). En la iteración 5 se puede observar cómo la representación del mapa de saliencia es más oscura, ya que iteración a iteración la saliencia acumulada va creciendo (tal como se explicó en la sección 4.3.1). En la iteración 11 podemos ver cómo el sistema ya ha encontrado una zona de atención que supera el umbral, y por tanto es visitada. Como vemos, el sistema reparte la mirada entre la taza y el reloj digital.

En la iteración 130 se introduce un nuevo objeto que genera atención, una pelota. Como vemos, el simple rodar de la pelota genera saliencia en el sistema, eso si, una saliencia mínima que no es impedimento para que se siga atendiendo a los demás objetos.

En la iteración 184 se ha estabilizado la saliencia correspondiente a la pelota, pero aún no tiene el umbral necesario para generar interés. Finalmente en la iteración 241 la saliencia de la pelota gana a la saliencia de cualquier objeto de la escena y por tanto es visitado.

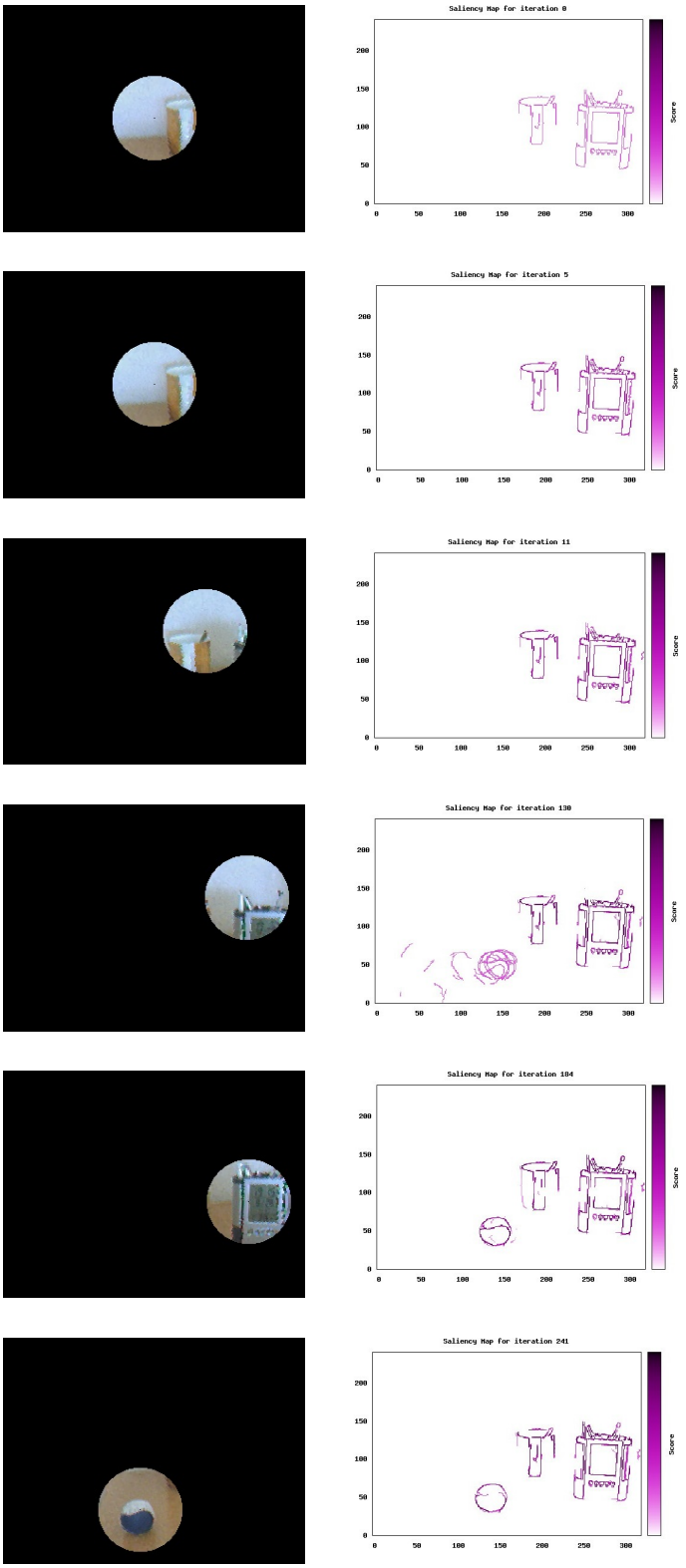


Figura 5.2: Secuencia de atención de saliencia dinámica.

De este experimento podemos sacar las siguientes conclusiones:

- El sistema de atención desarrollado reparte la mirada correctamente tanto para escenas estáticas con varios objetos, como para escenas dinámicas donde aparecen y desaparecen los objetos. Todas estas escenas están representadas con imágenes reales.
- El sistema de atención evita la hambruna. Esto es, asegura que tarde o temprano un punto con saliencia será visitado, independientemente de la velocidad de crecimiento de su saliencia.

5.1.2. Reparto de mirada en una escena con movimiento

En este experimento vamos a analizar una escena utilizando exclusivamente la característica de movimiento que incorpora nuestro proyecto. El fin de este experimento es repartir la mirada correctamente a todos los objetos en movimiento que existen en la escena.

La escena en este experimento es un tanto peculiar, ya que vamos a introducir como fuente de imágenes un vídeo donde pueden observarse multitud de coches pasando por una autopista. En la secuencia de imágenes que se muestran en la figura 5.3 podemos ver cómo el sistema de atención va siguiendo a los coches que pasan por la autopista. Es interesante ver como la estela del movimiento de los coches va perdiendo poco a poco interés, ya que lo que verdaderamente provoca atención es el movimiento continuado de los coches.

De este experimento podemos sacar las siguientes conclusiones:

- Nuestro sistema de atención funciona correctamente con imágenes reales y cotidianas. Sin necesidad de preparar el entorno, hemos comprobado que el sistema atentivo realiza una secuencia de movimientos sacádicos para observar a los coches que se mueven.
- Hemos obtenido un comportamiento de atención muy similar al que tiene el ser humano, en cuanto a movimiento se refiere. Existe un experimento que se realizó para estudiar el sistema atentivo humano ([Turégano Pedruelo, 2008]), y que se puede observar en la figura 5.4 donde los puntos de atención a los que mira el ojo humano se han señalado con una cruz roja. Claramente guarda una gran similitud con el sistema atentivo que hemos creado, ya que es capaz de repartir la mirada entre los distintos objetos en movimiento que hay en la escena, que en este caso son coches.

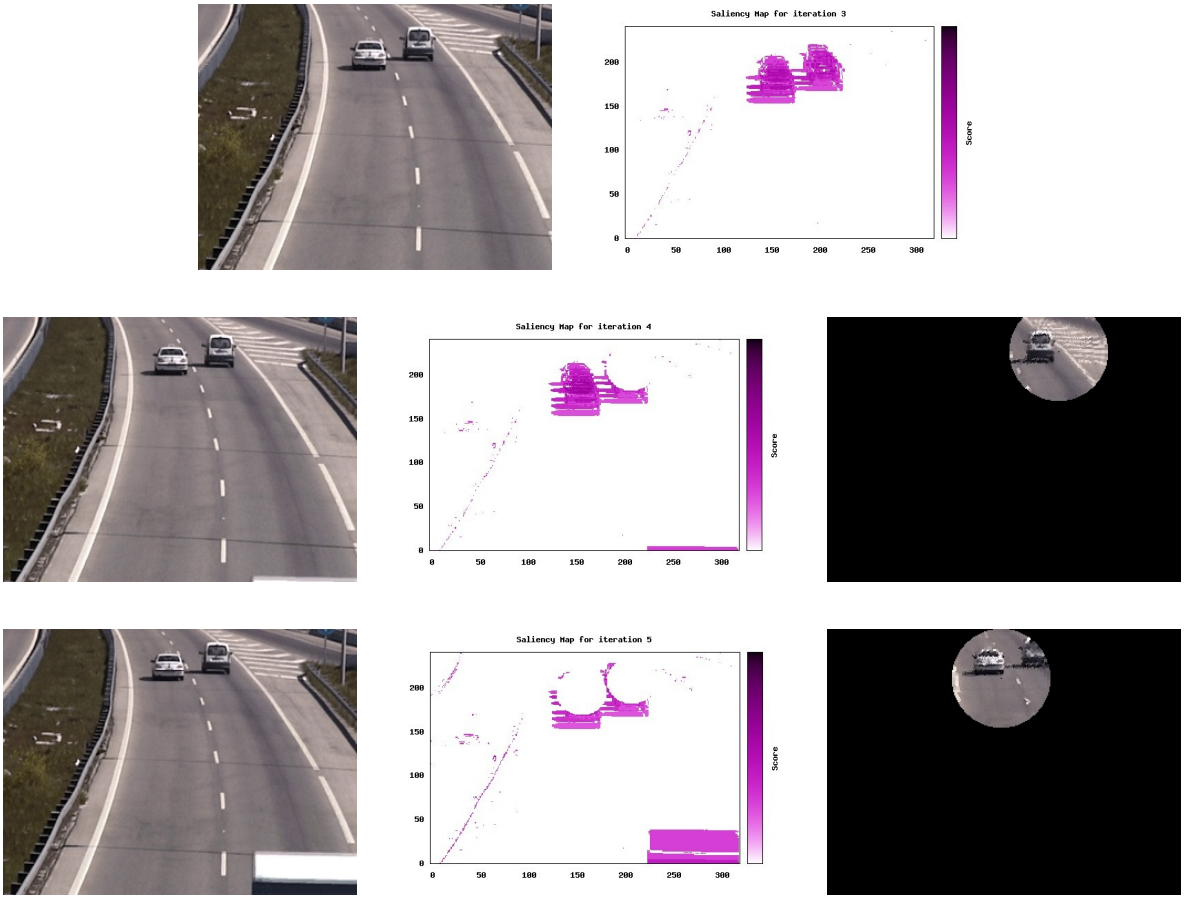


Figura 5.3: Secuencia de atención sobre una escena con movimiento.

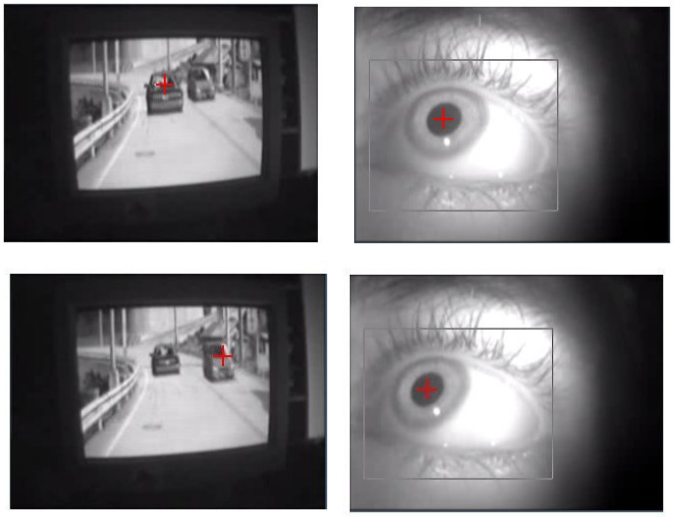


Figura 5.4: Secuencia de atención real del ser humano para imágenes con coches.

5.2. Reconstrucción 3D atentiva

En esta sección vamos a detallar los resultados que hemos obtenido cuando hemos realizado pruebas sobre el sistema de reconstrucción 3D atentivo. Concretamente hemos realizado experimentos sobre escenas sencillas (5.2.1) y sobre escenas complejas (5.2.2), haciendo énfasis en la representación tridimensional conseguida.

Es relevante destacar que la secuencia de puntos atentivos de la imagen que produce el sistema de atención se convertirá en una secuencia de puntos 3D representados por el sistema de reconstrucción 3D.

5.2.1. Reconstrucción 3D de una escena sencilla

En este experimento vamos a comprobar cómo se comporta nuestro sistema ante un objeto relativamente sencillo en cuanto al número y disposición de bordes. El objeto que hemos utilizado para este experimento es un cubo que hemos preparado de tal forma que cada lado tenga un color llamativo para así provocar bordes en las aristas del cubo. Debido a ésto hemos configurado nuestro sistema de atención para que preste interés por los bordes.

En la figura 5.5 podemos ver cómo el sistema de atención se fija únicamente en los bordes, calcula los puntos homólogos, y triangula para reconstruir en 3D. Como vemos en las imágenes de representación en 3D, la imagen parece bien representada cuando la miramos desde un ángulo parecido al de las cámaras. Sin embargo si miramos desde un ángulo muy distinto observamos que el cubo no es perfecto. Esto es debido a que los puntos homólogos en esta situación no son muy exactos, ya que a lo largo de una arista entre dos lados hay muy poca diferencia entre todos esos puntos. Esta dificultad en el emparejamiento se debe a la falta de textura en las regiones de interés por la utilización de imágenes sintéticas, y ya se detectó [Pacios and Cañas Plaza, 2007].

De este experimento podemos sacar varias conclusiones:

- El sistema de atención funciona correctamente con objetos simples como un cubo, mostrando interés por todos los puntos con saliencia.
- El cálculo del punto homólogo y el correcto funcionamiento de la función de correlación necesita de imágenes con textura para poder diferenciar correctamente puntos muy similares a simple vista. En imágenes sintéticas no existe apenas textura, por lo que resulta difícil obtener un buen emparejamiento y reconstrucción de los puntos de interés.

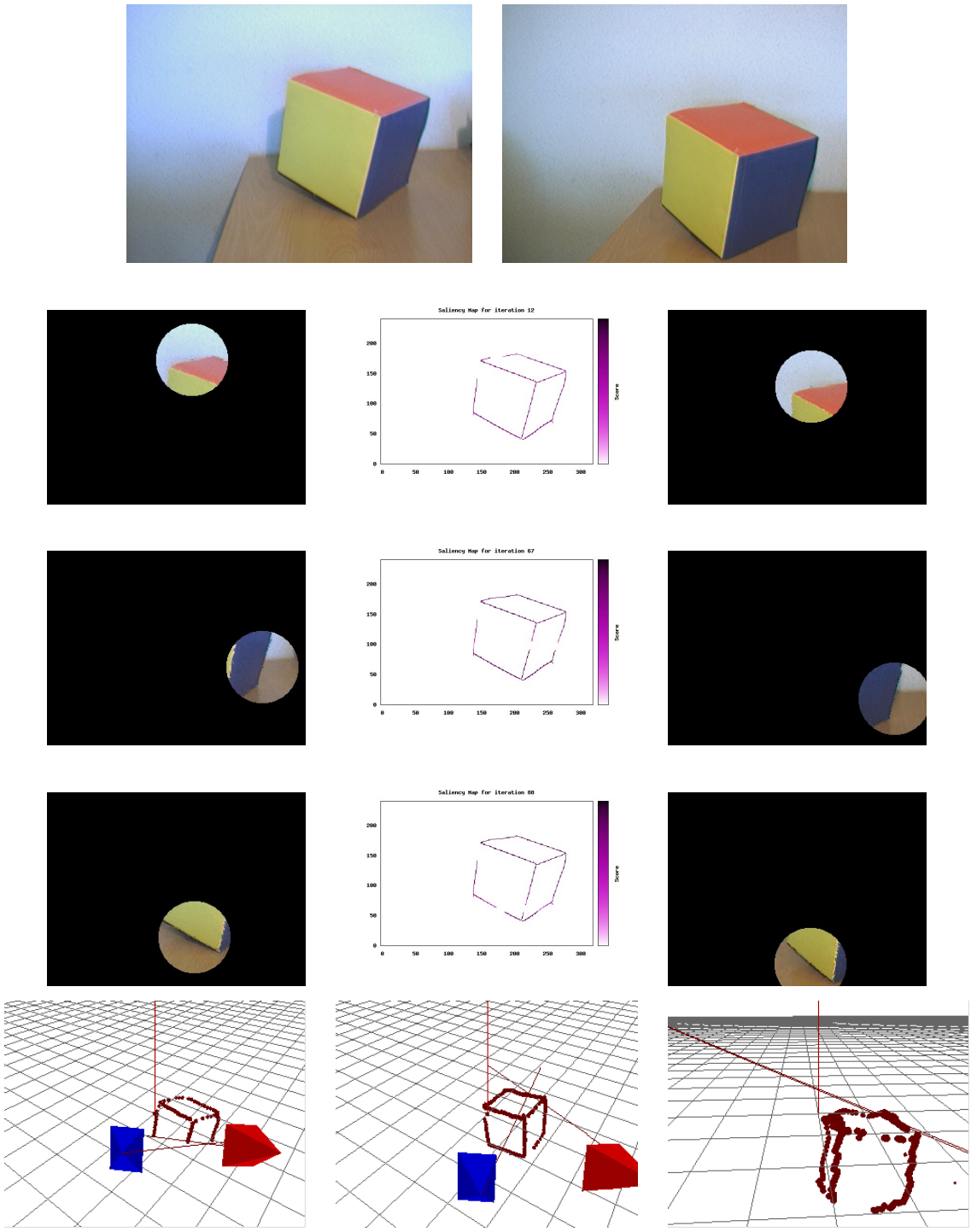


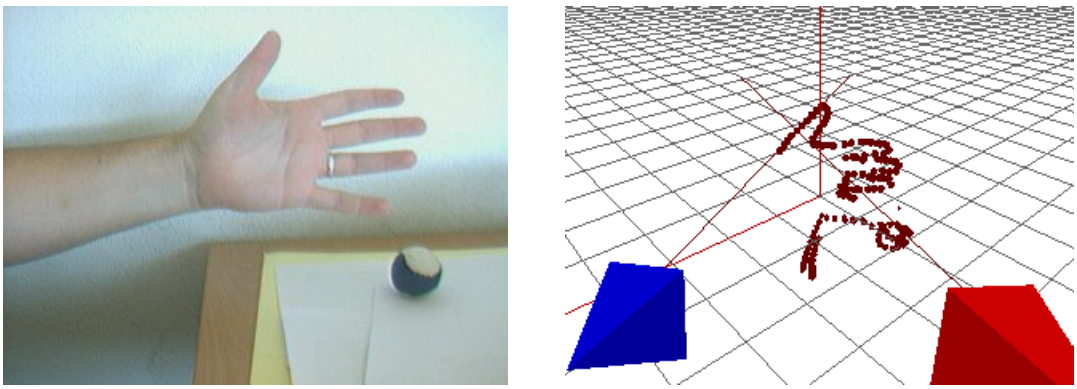
Figura 5.5: Reconstrucción 3D de una escena sencilla.

- El correcto funcionamiento del sistema 3D depende en gran medida del correcto emparejamiento de los puntos homólogos.

5.2.2. Reconstrucción 3D de una escena compleja

En este experimento vamos a comprobar cómo se comporta el sistema cuándo le situamos ante una escena complicada. Para nuestro sistema, la complejidad de una escena es directamente proporcional a los puntos con saliencia que existen en la epipolar. Dicho de otro modo, en el cálculo del punto homólogo, cuantos más puntos tengan saliencia en la línea epipolar, más cómputo de comparación de parches hay que realizar, y más número de candidatos existen a la hora de emparejar.

En este experimento vamos a configurar al sistema de atención del mismo modo que en el anterior experimento, activando únicamente la característica de bordes. Como podemos observar en la figura 5.6(a) la escena muestra un objeto a priori bastante complicado, como puede ser la mano. Sin embargo y como podemos ver en la figura 5.6(b) la reconstrucción ha sido bastante fiel a la realidad.



(a) (b)

Figura 5.6: Imagen de una escena compleja (a) y su reconstrucción en 3D (b).

De este experimento podemos sacar varias conclusiones:

- La reconstrucción de objetos a priori complejos como la mano, da como resultado una reconstrucción 3D bastante fiable.
- La complejidad de una imagen no se encuentra en el número de bordes, o en la naturaleza de los mismos. Dicha complejidad se produce debido al uso de imágenes o zonas sintéticas que no poseen textura, como en el ejemplo anterior. Gracias a que la

mano no es un objeto sintético y posee textura sobre toda su superficie, el algoritmo de emparejamiento no tiene demasiados problemas en localizar el punto homólogo correctamente.

5.2.3. Reconstrucción 3D mediante reparto de mirada sobre objetos móviles

Mediante este experimento vamos a demostrar la solvencia de nuestro sistema cuando varios objetos móviles aparecen en la escena y además mostraremos cómo la trayectoria de estos objetos es representada correctamente en 3D.

El experimento consiste en tener dos objetos en la escena que llamen la atención al sistema y que describan trayectorias diferentes. Así, en las dos primeras imágenes de la figura 5.7 podemos observar dos pelotas de color naranja, que son los objetos que van a llamar la atención al sistema, debido a que hemos activado la saliencia por color. La pelota de la izquierda describirá movimiento circular, y la pelota de la derecha describirá un movimiento vertical.

El comportamiento del sistema presta atención a los dos objetos alternando la mirada entre ellos y siguiendo su recorrido en todo momento. Como podemos observar en las imágenes de las iteraciones del sistema de atención (figura 5.7), el sistema de atención es capaz de mantener la atención sobre las dos pelotas repartiendo de esta manera el interés en la escena. Los mapas de saliencia son muy ricos en información ya que con un simple vistazo podemos ver qué zonas van a ser visitadas próximamente (regiones con color oscuro), y que zonas están perdiendo interés (regiones con color claro). Podemos apreciar que las zonas que pierden interés son justamente las estelas que van dejando el movimiento de los objetos.

Por último podemos ver la representación en 3D del movimiento de los dos objetos que llaman la atención al sistema. Esta representación es bastante fiel al movimiento realizado por los dos objetos, de hecho podemos apreciar claramente cómo uno de ellos ha realizado un movimiento circular y el otro objeto ha realizado un movimiento vertical.

Otro ejemplo ligado a este experimento que hemos realizado es utilizar un único objeto móvil para atraer el interés del sistema de atención. En la figura 5.8 podemos observar cómo el objeto ha seguido una trayectoria en espiral y nuestro sistema de reconstrucción 3D lo ha representado con gran fidelidad.

De este experimento podemos sacar las siguientes conclusiones:

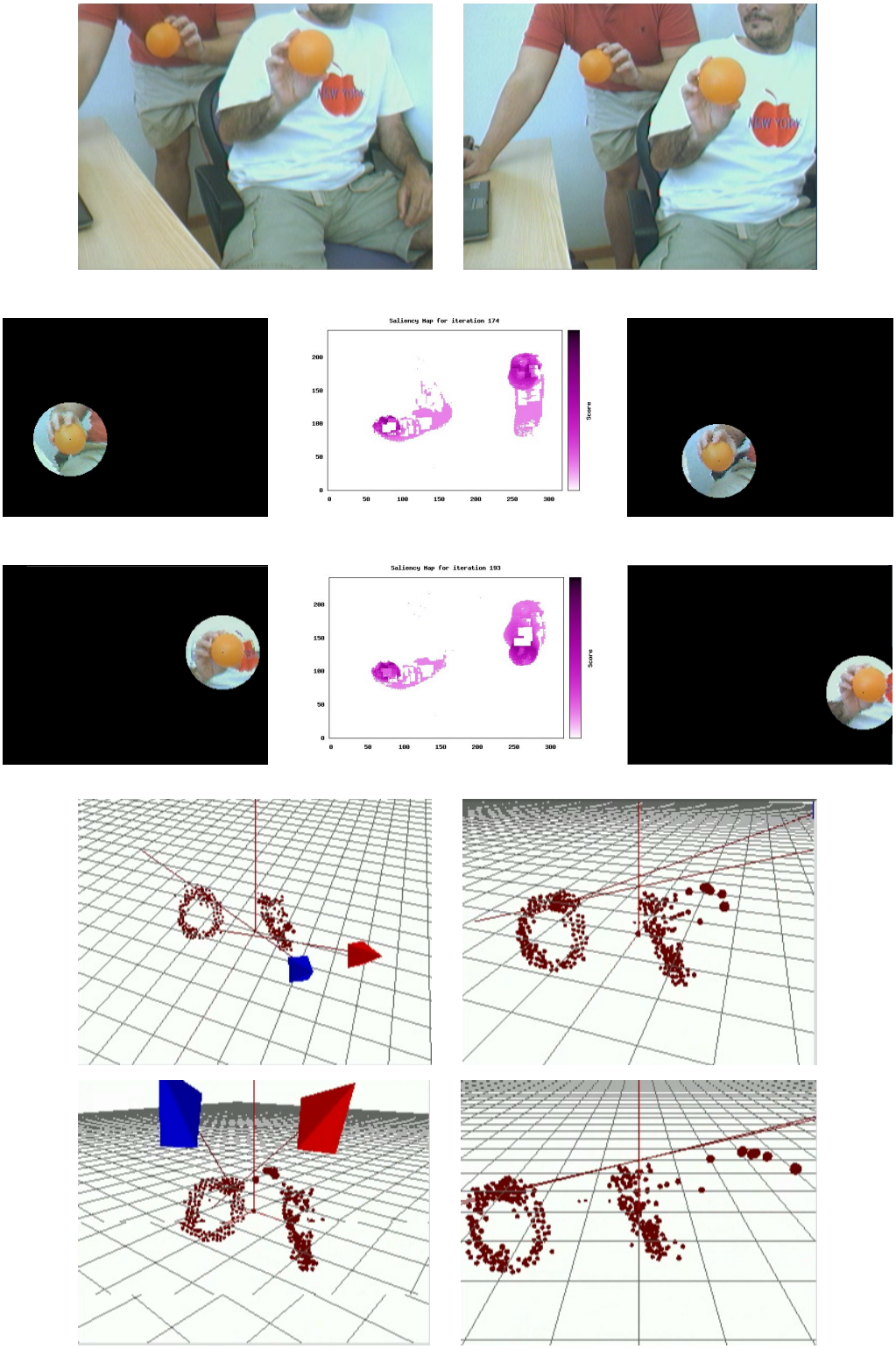


Figura 5.7: Reparto de mirada sobre objetos móviles y su reconstrucción en 3D.

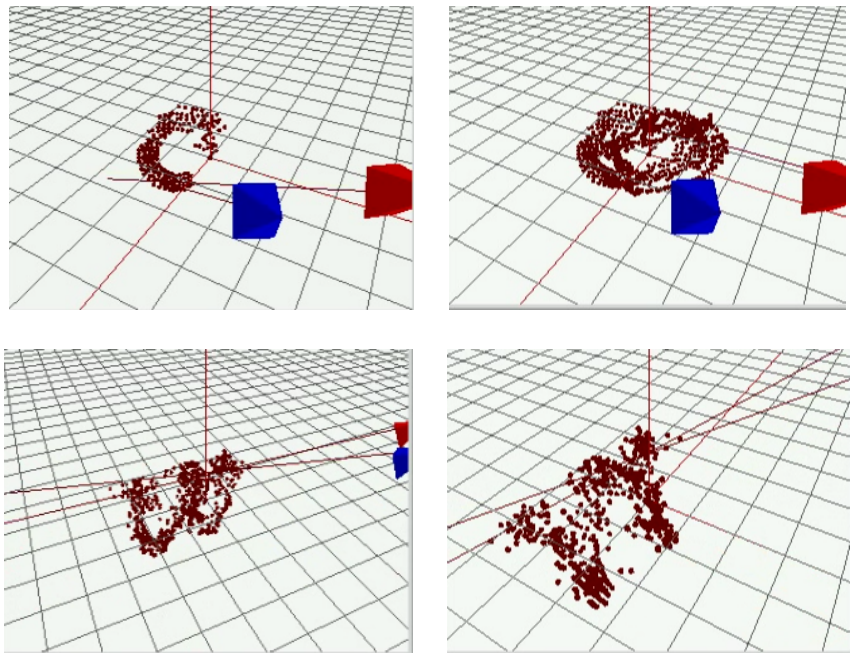


Figura 5.8: Reconstrucción 3D de un objeto móvil en la escena.

- Nuestro sistema de atención es capaz de repartir la mirada adecuadamente entre dos objetos móviles de interés.
- Nuestro sistema de reconstrucción 3D y concretamente el algoritmo de correlación, funciona razonablemente bien cuando existen dos objetos muy similares en la escena. Incluso cuando dichos objetos están situados en la misma recta epipolar.
- Gracias a la representación 3D que ofrecemos en estos experimentos anteriores, abrimos la puerta a la opción de analizar estos puntos 3D para poder estimar velocidad, dirección o posición de un objeto en el mundo real.

5.3. Características cognitivas.

En esta sección vamos a introducir los resultados obtenidos utilizando diferentes razonamientos cognitivos en distintas capas del software. Describiremos el experimento cognitivo que nos lleva a representar en 3D una parte del objeto que no aparece en la escena (5.3.1). Y también veremos los resultados de utilizar la característica cognitiva en el sistema de saliencia, produciendo hipótesis de dónde encontrar una parte del objeto que no se aparece en la escena (5.3.2) y que esas zonas llamen la atención al sistema.

5.3.1. Reconstrucción 3D mediante procesamiento externo

En este experimento vamos a mostrar la potencia de la arquitectura que hemos realizado, modulando ciertos parámetros del sistema, y además añadiendo una carga cognitiva al comportamiento.

El objetivo es mostrar una figura incompleta al sistema y que sea capaz de reconstruir ciertas partes que falten. Como ejemplo concreto, vamos a suponer que le mostramos un cuadrado incompleto, en el que sólo se conocen tres de sus cuatro vértices. El sistema de atención será configurado para visualizar y atender los puntos que forman las esquinas del cuadrado, y una vez obtenga estas tres esquinas calculará y deducirá la cuarta, que lo situará igualmente en 3D.

El sistema de atención ha sido configurado, mediante la interfaz gráfica, para que le llame la atención el color que tienen los vértices del cuadrado. Como podemos observar en la figura 5.10, partimos de un par de imágenes donde está representado un cuadrado incompleto, faltándole una de sus esquinas. En el primer paso, el sistema está obteniendo las zonas de saliencia que son interesantes para la configuración establecida. En el segundo paso, el sistema de saliencia ya tiene un candidato donde mirar, y observa el punto que representa la esquina superior del cuadrado y por tanto desaparece del mapa de saliencia. En el paso siguiente, observamos cómo el sistema decide mirar a la esquina derecha del cuadrado, eliminándola del mapa de saliencia. En el último paso, el sistema decide mirar al último punto que aparece en su mapa de saliencia, la esquina izquierda del cuadrado. A modo de curiosidad, podemos observar en este paso, cómo el sistema de saliencia vuelve a interesarse por uno de los puntos del cuadrado que antes habíamos visitado, aunque éste hecho no es relevante para este experimento.

Para obtener la cuarta esquina del cuadrado es necesario desarrollar fórmulas geométricas. Para ello, vamos a analizar la figura 5.9. Dicha solución se basa en obtener las rectas *r1* y *r2*, para calcular su intersección y así hallar el punto buscado. La característica de la recta *r1* es que debe ser paralela a la recta *s1* y que pase por el punto *P1*, justo lo que refleja la ecuación(5.1). Del mismo modo, la recta *r2* se caracteriza por ser paralela a la recta *s2* y que pasa por el punto *P2*, que matemáticamente queda representado en la ecuación(5.2). Por último, en la ecuación(5.3) se resuelve la incógnita t, que sustituyéndola en cualquiera de las dos ecuaciones de las rectas, nos servirá para calcular las coordenadas

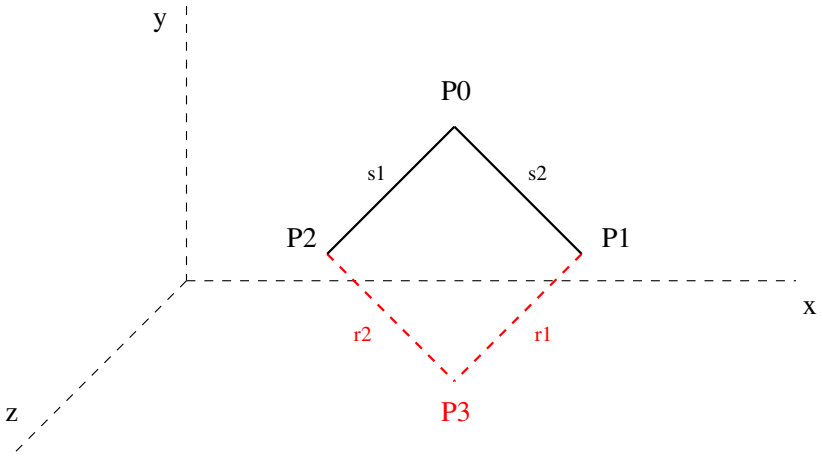


Figura 5.9: Representación del problema del cuadrado.

del punto *P3*.

$$r_1 = \begin{cases} x_3 = x_1 + (x_0 - x_2) * t \\ y_3 = y_1 + (y_0 - y_2) * t \\ z_3 = z_1 + (z_0 - z_2) * t \end{cases} \tag{5.1}$$

$$r_2 = \begin{cases} x_3 = x_2 + (x_0 - x_1) * t \\ y_3 = y_2 + (y_0 - y_1) * t \\ z_3 = z_2 + (z_0 - z_1) * t \end{cases} \tag{5.2}$$

$$t = \frac{x_1 - x_2}{(x_0 - x_1) - (x_0 - x_2)} \tag{5.3}$$

Este componente software, que hemos generado fuera del sistema atentivo, es capaz de analizar la reconstrucción 3D para así deducir dónde se encuentra la cuarta esquina del cuadrado. En cierto modo, hemos dotado de *raciocinio espacial* al sistema.

Como resultado final podemos observar en la figura 5.11 la representación en 3D, desde diferentes ángulos, de los puntos que componen las esquinas del cuadrado. La esquina que no aparecía en la imagen original pero que hemos deducido, se ha representado en color verde. El resultado de la estimación del cuarto punto es realmente buena.

De este experimento podemos sacar las siguientes conclusiones:

- El sistema de saliencia, emparejamiento y triangulación forma una base sólida sobre la que trabajar y abordar numerosos problemas de visión en 3D.

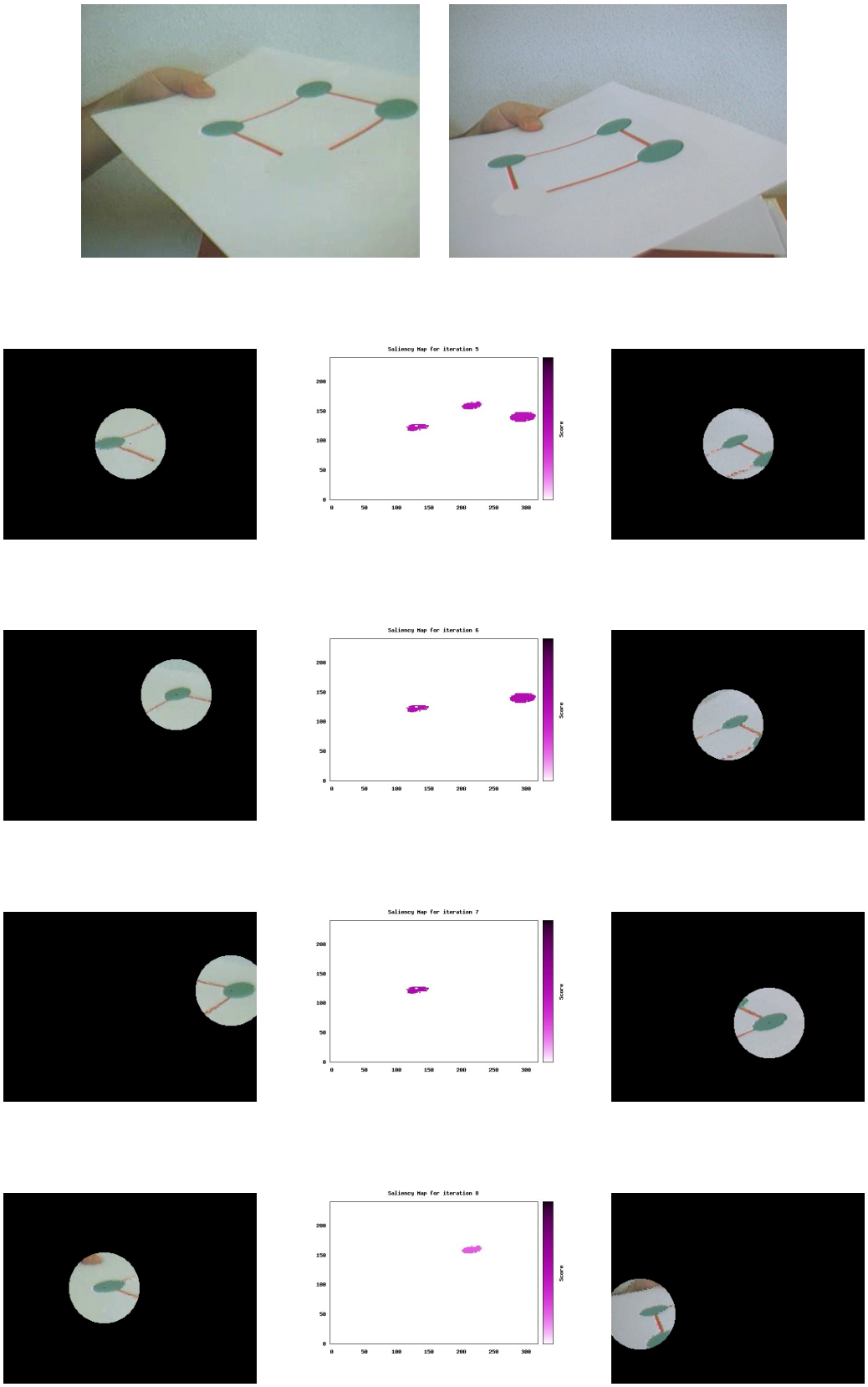


Figura 5.10: Secuencia de la búsqueda del 4º punto del cuadrado.

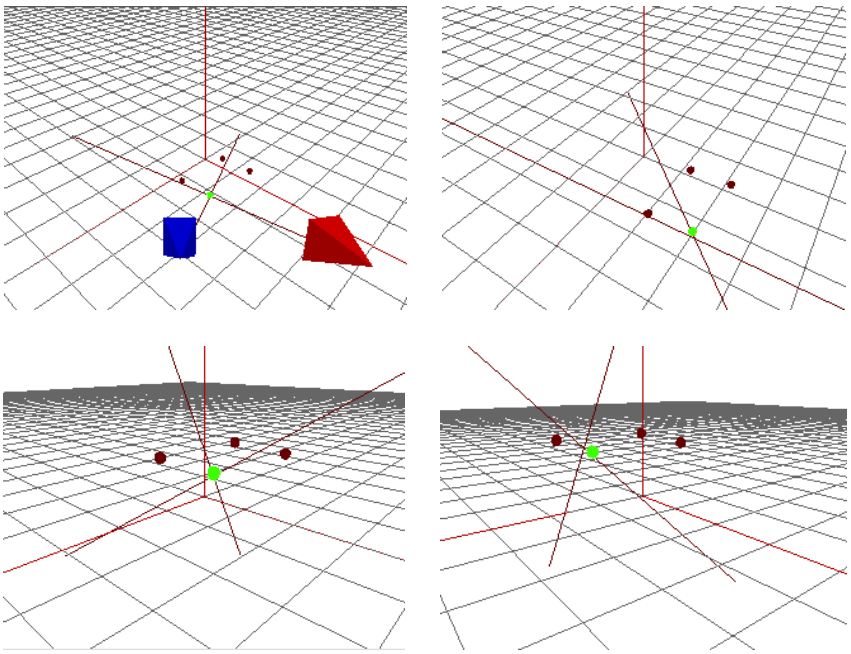


Figura 5.11: Representación final en 3D del cuadrado.

- Hemos dotado al sistema de una capacidad de decisión, utilizando los datos generados en 3D. Razona sobre la representación en 3D que creamos, y no sobre las imágenes que obtiene de la cámara.

5.3.2. Hipótesis de objeto mediante atención cognitiva

En este último experimento vamos a activar dos características de saliencia simultáneamente. Durante todo el proyecto hemos explicado que las características que influyen en el sistema de atención obtienen la saliencia analizando las imágenes (*bottom-up*). Ahora vamos a introducir simultáneamente una característica que influye en el sistema de saliencia, pero que no obtiene la información de las imágenes, sino que utiliza una característica de atención descendente o cognitiva (*top-down*).

La característica que hemos utilizado es un característica cognitiva que nos ofrecerá información sobre dónde puede encontrarse la cuarta esquina de un cuadrado incompleto. A diferencia del experimento anterior, que se basaba en averiguar y calcular la cuarta esquina aunque no apareciera en la imagen, este experimento se basa en ofrecer una hipótesis sobre la situación donde podría estar la cuarta esquina del cuadrado para que dicha zona llame la atención del sistema. Esta hipótesis se refleja como una zona de saliencia en el mapa de saliencia acumulado.

Como vemos en la figura 5.12 partimos de un par de imágenes donde podemos observar un cuadrado incompleto. Los dos primeros pasos que se muestran en el conjunto de imágenes muestran cómo el sistema de atención se fija en las esquinas de color rojo, ya que generan atención en el mapa de saliencia. En el tercer paso podemos ver cómo aparece un zona de saliencia en el mapa con forma cuadrada, que no aparece en la imagen original. Esto implica que la característica cognitiva posee datos suficientes para generar la hipótesis de dónde puede estar la cuarta esquina del cuadrado, y la materializa añadiendo un zona de interés en el mapa de saliencia. Como podemos ver en el último paso, el sistema de atención ha decidido mirar (por poseer saliencia alta) a la región de hipótesis dónde pudiera estar la cuarta esquina del cuadrado. Al no existir dicha esquina, no se empareja ni triangula este punto de interés.

De este experimento podemos sacar las siguientes conclusiones:

- Nuestro sistema, a parte de permitir saliencia ascendente basada directamente en características de la imagen (color, bordes, movimiento), es capaz de integrar homogéneamente una saliencia descendente o cognitiva como puede ser "*la hipótesis de la cuarta esquina de un cuadrado*".

- Es capaz de construir una hipótesis de situaciones que no existen en las imágenes, ofreciendo así una claro matiz inteligente a nuestro comportamiento.

- La potencia de nuestro sistema atentivo queda reflejada en este experimento ya que es capaz de mover el *ojo* para comprobar una hipótesis perceptiva. Incluso puede ser extremadamente útil cuando la región de atención calculada mediante hipótesis se encuentre fuera de la visión instantánea de las cámaras (atención *overt*). Además, abre la puerta a mecanismos de atención para comprobar hipótesis perceptivas.

- Nuestro sistema de atención, en este experimento, comprueba constantemente la hipótesis generada de la cuarta esquina hasta que ahí aparezca la esquina. Del mismo modo, se puede comparar con el sistema de atención humano en el caso que, estando el semáforo en rojo, continuadamente reparte la mirada a la parte inferior del mismo, para ver si el semáforo pasa a verde.

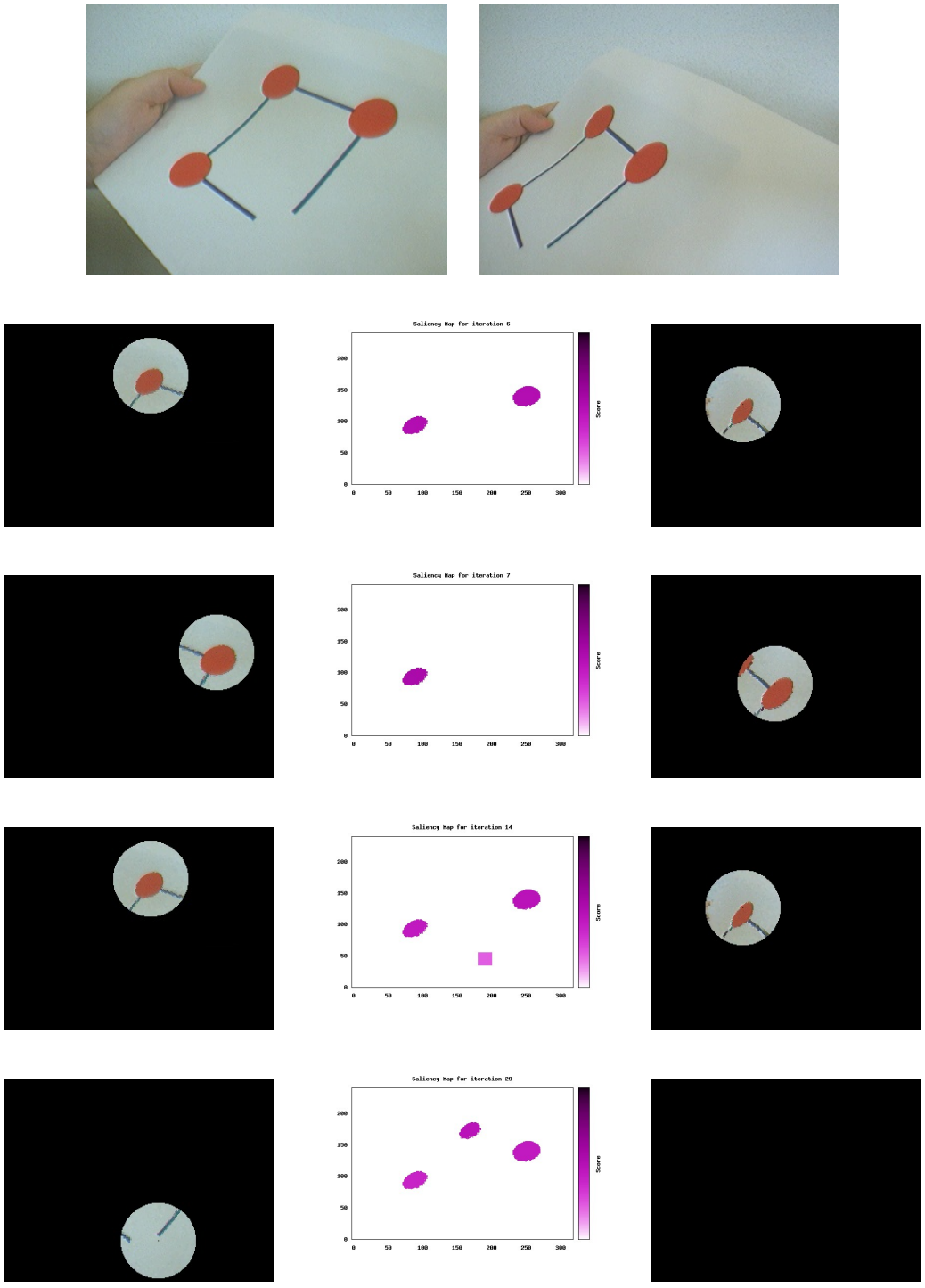


Figura 5.12: Característica cognitiva que realiza una hipótesis sobre la situación de la cuarta esquina.

5.4. Coste temporal

Una vez mostrada la validez del sistema en los experimentos anteriores, vamos ahora a caracterizar el rendimiento temporal de la solución desarrollada.

En la tabla 5.1 podemos ver los tiempos de las tareas más identificativas de nuestro proyecto. Para plasmar estos datos de cómputo en la tabla hemos utilizado una iteración que poseía números candidatos a punto homólogo. El cálculo del punto homólogo, es la única tarea que varia el tiempo de cómputo dependiendo de la iteración. Esto es debido a que si en una iteración dada, existen numerosos puntos dónde calcular y comparar el parche gastará mucho más tiempo que en una iteración donde no existan tantos puntos candidatos donde realizar el parche. Como vemos la tarea 4 utiliza el 50 % de cómputo de la iteración completa. En los experimentos realizados hemos obtenido una media de cómputo por iteración de *0,250 - 0,300* segundos, lo que nos permite procesar 4-5 iteraciones por segundo.

id	Nombre Tarea	Tiempo (seg)	%
1	Cálculo del mapa de saliencia y punto interesante	0,11	25 %
2	Cálculo de la imagen log-polar	0,03	7 %
3	Cálculo de la recta epipolar	0,055	12,5 %
4	Cálculo del punto homólogo	0,22	50,11 %
5	Cálculo de la gráfica de parecido	0,02	4,55 %
6	Representación en 3D	0,004	0,91 %
7	**Total tiempo iteración**	**0,439**	-

Cuadro 5.1: Tiempos de las tareas realizadas en una iteración.

Representación	Tamaño de parche	Tiempo (seg.)	Puntos de interés	Resultado
Cubo	90x90	13	95	Muy buena pero lenta
Cubo	45x45	12	95	Bastante buena
Cubo	15x15	11	95	Irregular
Mano	90x90	22	125	Muy buena pero lenta
Mano	45x45	18	125	Bastante buena
Mano	15x15	16	125	Irregular

Cuadro 5.2: Tiempos generales de representación 3D.

Es importante también reflejar cuánto tiempo tarda el sistema en reconstruir en 3D los objetos completos que hemos utilizado para la explicación del proyecto. Por ello hemos

reflejado en la tabla 5.2, los tiempos de los ejemplos más característicos de este proyecto en las diferentes configuraciones que hemos utilizado.

5.5. Alternativas de correlación entre parches

Como ya vimos en la sección 4.4.1, en nuestro proyecto hemos utilizado la comparación de parches cuadrados en el espacio de color *HSV* como base para el algoritmo de emparejamiento. Esta elección la hemos hecho gracias a los experimentos y pruebas que hemos realizado con otras técnicas que mostramos en esta sección. A continuación veremos los resultados que obtuvimos comparando el rendimiento de los parches cuadrados en el espacio de color *RGB*, en luminancia normalizada y utilizando parches circulares en HSV (*log-polar*).

Hemos realizado numerosos experimentos con diferentes imágenes reales para comprobar el verdadero rendimiento de las 4 técnicas de emparejamiento que hemos analizado.

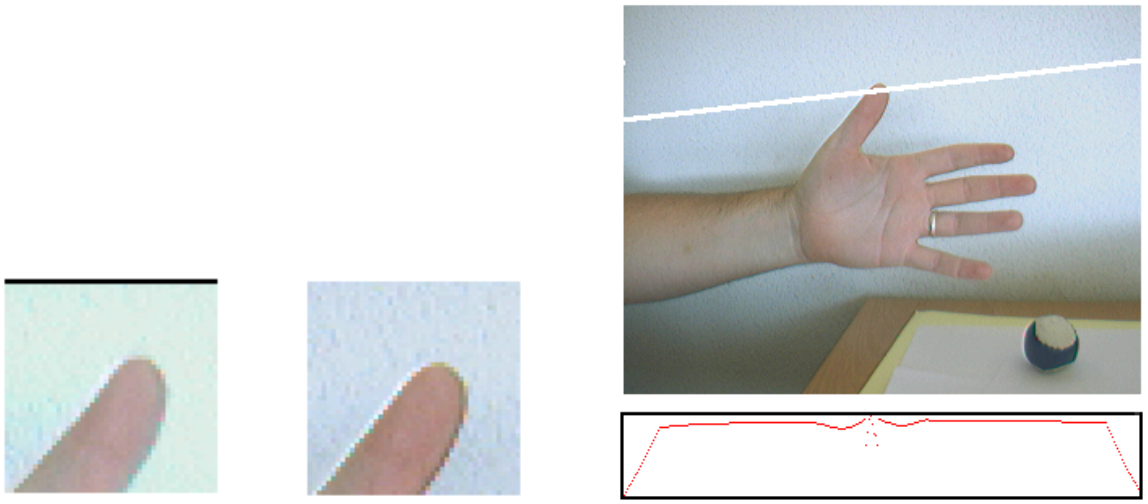
Correlación de parches en RGB

Para realizar la correlación en *RGB* tenemos que comparar los diferentes parches en el espacio de color *RGB*. Para ello realizaremos la resta de parches, píxel a píxel, componente a componente y en valor absoluto, tal como muestra la formula 5.4

$$Correlacion_{ParcheAB} = \sum_{p=0}^{TamParche} |(R_{pa} - R_{pb})| + |(G_{pa} - G_{pb})| + |(B_{pa} - B_{pb})| \qquad (5.4)$$

En la figura 5.13(b) podemos observar la gráfica de *correlación* que obtenemos para este emparejamiento. La función de correlación utilizando parches en *RGB* es muy poco discriminante comparada con la función de correlación que obtuvimos utilizando el espacio *HSV*, como ya vimos en la figura 4.15(b).

En la función de correlación vemos que en los límites de la imagen nos da que no es nada parecido, ya que cuánto más nos acercamos a los límites de la imagen, no tenemos información para generar el parche de 45x45 píxeles y por tanto esa parte de la que no tenemos información la rellenamos con píxeles negros (como podemos ver en el parche de la izquierda de la figura 5.13(a)).



(a) (b)

Figura 5.13: Parche máster ((a)-izquierda), parche homólogo ((a)-derecha) e imagen homóloga junto con la función de correlación (b).

Como podemos observar en la figura 5.13(a) la diferencia de intensidades de las imágenes es notable. A simple vista se ve que una de las imágenes tiene colores más vivos que la otra. Aún así el parche en RGB funciona bastante bien y encuentra al homólogo porque no tenía muchos más candidatos parecidos. Hemos comprobado que el parche en RGB funciona bien en imágenes que tienen la misma intensidad de color, algo difícil de obtener en nuestro sistema ya que las dos cámaras están situadas en posiciones y ángulos distintos y por tanto reciben diferente información de la luz.

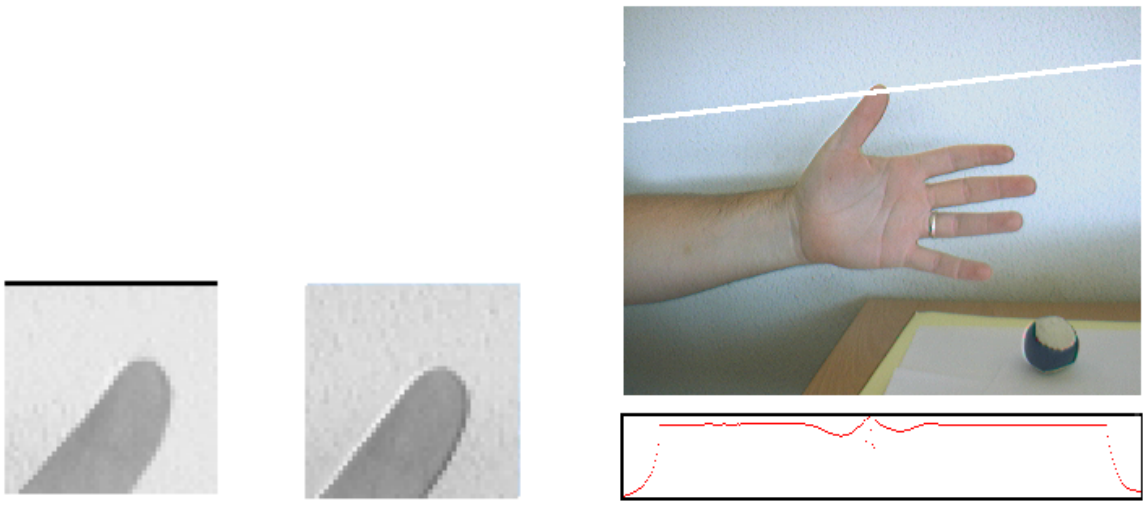
Correlación de parches luminancia normalizada

Esta técnica consiste en tratar los parches en escala de grises normalizada y compararlos. Con ello conseguimos no tener en cuenta la intensidad de la imagen (es una alternativa al HSV). Para normalizar, primero tenemos que obtener la luminosidad media de cada parche por separado, utilizando la fórmula (5.5). Después, para obtener el parecido tenemos que aplicar la fórmula (5.6), que al haber normalizado, obtendremos valores entre *-1* (muy distinto) y *1* (muy parecido).

$$I_{parche} = \frac{\sum_{p=0}^{TamParche} R_p * 0{,}30 + G_p * 0{,}59 + B_p * 0{,}11}{TamParche} \tag{5.5}$$

$$Correlacion_{ParcheAB} = \frac{\sum_{p=0}^{TamParche} (I_{pA} - I_{parcheA}) * (I_{pB} - I_{parcheB})}{\sqrt{\sum_{p=0}^{TamParche} (I_{pA} - I_{parcheA})^2 * \sum_{p=0}^{TamParche} (I_{pB} - I_{parcheB})^2}} \quad (5.6)$$

Aplicando este parche obtenemos las imágenes que se ven en la figura 5.14(a), que como se puede observar son casi idénticas en cuanto a luminosidad. Como podemos apreciar, la función de correlación (figura 5.14(b)) es ligeramente mejor que la del parche en *RGB*. Aunque en este caso concreto que hemos realizado no supone mucha diferencia, sí hemos comprobado que el parche en luminancia en la mayoría de las ocasiones mejora la función de correlación que genera el parche en *RGB*.



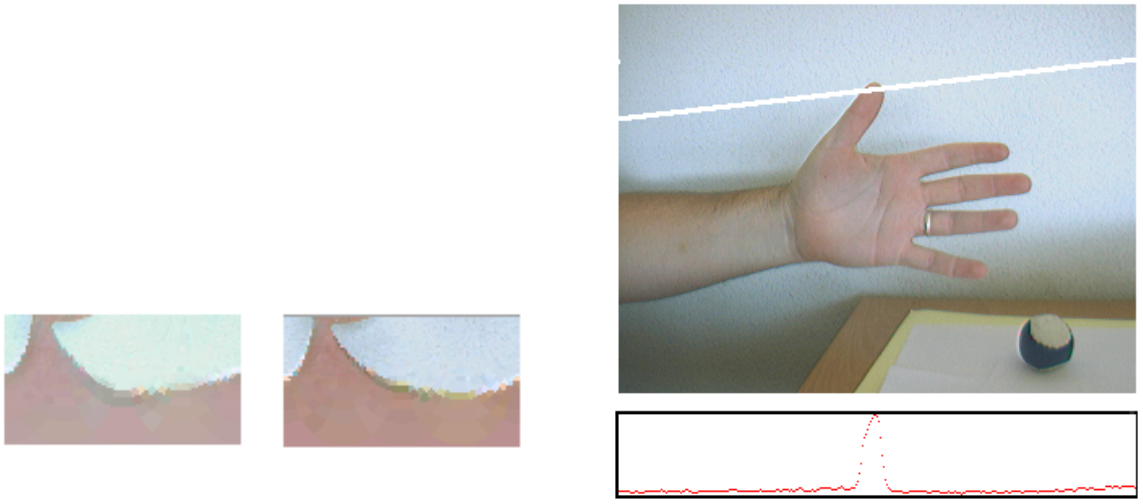
(a) (b)

Figura 5.14: Parche máster ((a)-izquierda), parche homólogo ((a)-derecha) e imagen homóloga junto con la función de correlación (b).

Correlación con parches circulares en HSV

La tercera técnica para generar el parche que hemos estudiado es mediante la combinación de log-polar y HSV. Hemos utilizado los beneficios de logpolar y de HSV. El parche en este caso es circular (en vez de rectangular), ya que logpolar nos provee de una imagen circular allí donde posamos la atención. Aún siendo circular, podemos representarlo

como un parche de 55x100 píxeles (figura 5.15(a)). Hemos optado por este tamaño de parche ya que para parches más pequeños no funciona correctamente el algoritmo de logpolar. La comparación entre los parches logpolar se realiza en el espacio de color *HSV*, para evitar errores debidos a la diferencia de luminosidad.



(a) (b)

Figura 5.15: Parche máster ((a)-izquierda), parche homólogo ((a)-derecha) e imagen homóloga junto con la función de correlación (b).

Como podemos observar en la figura 5.15(b) el resultado es bastante discriminante. La unión de estas dos técnicas (log-polar y HSV) nos proporciona una función de correlación muy buena para algunos de los ejemplos probados.

El problema de esta técnica es que es demasiado lenta debido al mayor tamaño del parche, y que conlleva el cálculo adicional de la imagen logpolar por cada punto *interesante* en la epipolar. Aproximadamente es 10 veces más lento que las técnicas explicadas anteriormente.

Correlación con parches cuadrados en HSV

Como ya comentamos en la sección 4.4.1 la técnica por la que hemos apostado para incluir en nuestro proyecto es la técnica que se basa en *parches cuadrados en HSV*. Como queda reflejado en la figura 4.15(b) la función de correlación utilizando esta técnica es muy discriminante, ofreciendo de una manera más segura un punto homólogo fiable.

La elección del parche *cuadrado HSV* para nuestro proyecto, viene fundada en que: (1)

nos ofrece una mayor velocidad de proceso y una mejor relación fiabilidad-cómputo que el parche *circular HSV*, (2) salva de una manera más eficaz las diferencias en intensidad que el parche *cuadrado RGB* y (3) discrimina la luminosidad de una manera más acertada que el parche *cuadrado en luminancia*. De hecho, con este tercer argumento, podemos concluir que el color aporta un gran poder discriminante en el algoritmo de correlación entre parches frente a sólo la luminosidad.

5.6. Alternativas de tamaños de parches

En este experimento vamos a analizar la influencia que tiene en nuestro algoritmo de emparejamiento el tamaño de parche utilizado. Vamos a realizar las pruebas utilizando diferentes tamaños de parche. El parche es el punto clave para el emparejamiento óptimo. Por ello vamos a probar utilizando parches cuadrados de 90x90 píxeles, de 45x45 píxeles y de 15x15 píxeles. Estos experimentos los hemos realizado utilizando para el emparejamiento la correlación entre parches en *HSV*.

Antes de realizar este experimento cabe esperar a priori que cuanto mayor sea el parche, mejor será el emparejamiento y más lento será el tiempo de cómputo. Por el contrario cuanto menor sea el parche, peor emparejamiento obtendremos y mayor rapidez en el cálculo de los algoritmos.

Hemos realizado estas pruebas con varias imágenes distintas, a continuación mostramos el funcionamiento con una par de ellas, las mismas usadas en el experimento 5.5.

Parche de 90x90

Utilizando parches con un tamaño de 90x90 píxeles obtenemos la representación que se puede observar en la figura 5.16.

El tiempo aproximado en realizar la representación 3D de la figura 5.16 es de 22 segundos. Como podemos ver, dicha representación 3D es bastante fiel a la realidad de las imágenes.

Parche de 45x45

Utilizando parches con un tamaño de 15x15 píxeles para realizar el emparejamiento, obtenemos la representación en 3D que aparece en la figura 5.17.

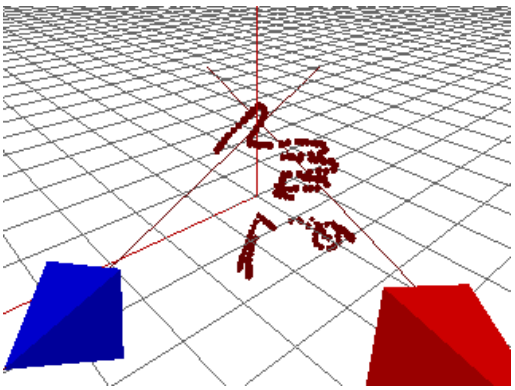


Figura 5.16: Representación 3D de la escena utilizando parches cuadrados de 90x90 píxeles.

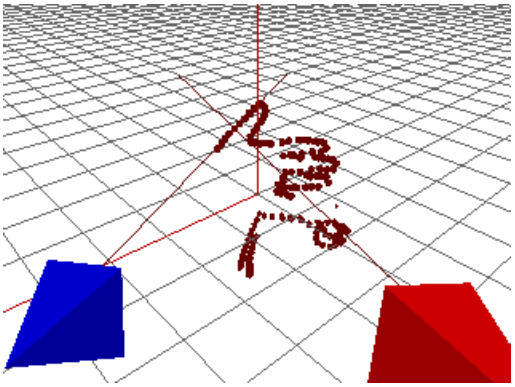


Figura 5.17: Representación 3D de la escena utilizando parches cuadrados de 45x45 píxeles.

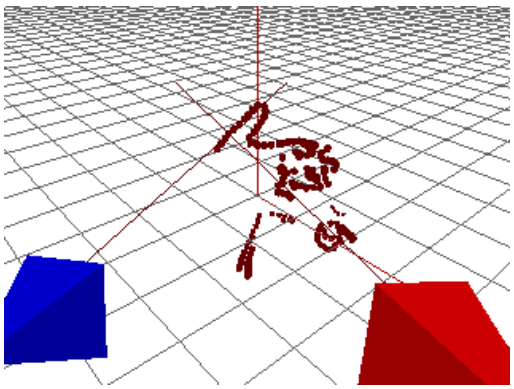


Figura 5.18: Representación 3D de la escena utilizando parches cuadrados de 15x15 píxeles.

El tiempo aproximado en realizar la representación 3D de la figura 5.17 es de `18` segundos. En dicha representación podemos observar que la figura se sitúa bien en 3D, a excepción de algunos puntos espúreos.

Parche de 15x15

Por último, utilizando parches con un tamaño de 15x15 píxeles para encontrar el punto homólogo, obtenemos la representación en 3D que mostramos en la figura 5.18.

El tiempo aproximado en realizar la representación 3D de la figura 5.18 es de `16` segundos. Esta representación no es nada fiable, podemos observar en la imagen 3D como bastantes puntos no se emparejan correctamente.

De este experimento podemos sacar dos conclusiones:

- El tiempo de cómputo del algoritmo de emparejamiento es directamente proporcional al tamaño del parche, y con ello a la calidad de la representación 3D de la escena.
- Gracias a este experimento, hemos optado por utilizar el tamaño de parche de 45x45 píxeles, ya que nos ofrece una representación en 3D bastante buena (comparado con el parche de tamaño 15x15) y además ganamos tiempo con respecto al parche de tamaño 90x90. Supone un buen valor de compromiso.

Capítulo 6

Conclusiones y trabajos futuros

En los capítulos anteriores hemos descrito el problema que abordamos en este proyecto, los objetivos iniciales, y la solución propuesta junto con un conjunto de experimentos que la validan y caracterizan. En este capítulo expondremos las conclusiones obtenidas y las posibles líneas por las que se puede continuar el trabajo aquí realizado.

6.1. Conclusiones

Los objetivos marcados inicialmente para este proyecto fin de carrera implicaban que la aplicación fuera capaz de representar en 3D por medio de un sistema atentivo las figuras u objetos que llamaran la atención al sistema, mediante el uso de imágenes reales (no simuladas). Este sistema de atención tenía el fin de identificar las partes interesantes dependiendo de la configuración establecida. También hemos considerado importante la búsqueda del píxel homólogo, donde reside la eficiencia y rapidez del sistema. Por último hemos empleado un sistema de virtualización 3D para la representación de los objetos interesantes de una escena.

La conclusión general es que hemos cumplido satisfactoriamente los objetivos marcados en el capítulo 2. El sistema de atención funciona de modo correcto, repartiendo la mirada alternativamente entre los objetos relevantes de la escena. Gracias a una buena búsqueda del punto homólogo hemos obtenido una representación 3D bastante fiable (incluso con imágenes con poca textura).

A continuación comentaremos en qué medida se han satisfecho los sub-objetivos marcados en el capítulo 2:

1. *Sistema de atención:* En este sub-objetivo se proponía la realización de un sistema de atención que prestara interés a varios objetos relevantes en la escena. El sistema de atención que se ha desarrollado funciona correctamente tal como muestran los experimentos del capítulo 5. Las diferentes características de atención vuelcan su saliencia instantánea al mapa de saliencia acumulado, el cual es analizado para decidir qué región visitar. Gracias a la inhibición de retorno (*IOR*) podemos establecer por debajo de cero la saliencia de un objeto recién visitado, lo que nos ayuda a no tener *hambruna* de atención.

   Hemos desarrollado tres saliencias ascendentes y dos saliencias descendentes. Las tres saliencias ascendentes, que se basan en la información de la imagen para calcular las regiones de interés, son por bordes, color o movimiento. La saliencia por bordes permite fijar la atención sobre el contorno de los objetos, como por ejemplo una mano o un cubo (experimentos 5.2.2 y 5.2.1). La saliencia por color permite fijar la atención sobre objetos de un color determinado, como por ejemplo un par de pelotas de color naranja (experimento 5.2.3). La saliencia por movimiento nos permite prestar atención en los objetos móviles, como por ejemplo el paso continuado de coches (experimento 5.1.2).

   La generación de saliencias descendentes era uno de los requisitos más ambiciosos del proyecto y más complicado de realizar. Una de estas saliencias descendentes que hemos añadido al proyecto genera puntos de atención exploratorios y aleatorios en la escena, memorizando solamente los puntos visitados recientemente. La segunda saliencia descendente que hemos incluido es una prueba de concepto para hipotetizar dónde podría estar la cuarta esquina de un cuadrado incompleto (experimento 5.3.2). Este experimento muestra la potencia del algoritmo diseñado y abre la puerta a la atención guiada por conocimiento de alto nivel, no basado directamente en las características de la imagen.

2. *Reconstrucción 3D:* En este sub-objetivo se pedía un sistema de reconstrucción 3D que fuera capaz de situar en el espacio en tiempo real, los puntos de interés procedentes del sistema de atención. Para ello hemos tenido que resolver la problemática típica de la representación en 3D.

   El algoritmo de emparejamiento, y más concretamente el cálculo del punto homólogo, es el gran responsable de obtener una buena representación 3D. Por ello, hemos dedicado tiempo y esfuerzo a estudiar diferentes implementaciones de emparejamiento utilizando parches cuadrados o circulares, o utilizando diferentes espacios de color para la comparación de parches: RGB, luminancia normalizada y HSV. Debido a

que el color aporta una gran discriminación entre los parches y que la rapidez de la representación 3D es proporcional al tamaño de parche, optamos finalmente por configurar el algoritmo de emparejamiento mediantes parches cuadrados de 45x45 píxeles (experimento 5.6) y realizar la comparación de los mismos en el espacio de color HSV (experimento 5.5).

Un sub-requisito de la representación 3D era que debía ser en tiempo real, o al menos aproximarse a estos tiempos. Por ello hemos minimizado de una manera efectiva el tiempo de cálculo del punto homólogo, restringiendo la búsqueda de dicho punto en las zonas de la imagen donde se sitúa la recta epipolar y que además dicha zona posea saliencia o interés para el sistema en la imagen B.

Hemos constatado que en imágenes sintéticas o imágenes con relativamente poca textura, el algoritmo de emparejamiento no funciona tan eficientemente como en imágenes reales.

El sistema ha sido validado y experimentado con imágenes reales (no simuladas) como se muestra en el capítulo 5. Los diferentes experimentos nos han sido de gran ayuda para ir comprobando cómo se van cumpliendo los objetivos, y para concluir cuál es la mejor configuración del sistema para casos determinados.

Además de los objetivos, se han cumplido los requisitos establecidos en la sección 2.2:

1. Las cámaras debían estar calibradas con precisión, para conocer sus parámetros intrínsecos y extrínsecos y de esta manera poder realizar correctamente los cálculos geométricos. Se ha realizado una calibración bastante buena, utilizando el calibrador que ofrece la plataforma JDE. Además le hemos realizado algunas mejoras descritas en la sección 3.3.3.

2. El sistema debía ser ágil y bastante rápido, por lo que los algoritmos debían ser eficientes. Hemos conseguido que el comportamiento se ejecute a 8-10 iteraciones por segundo. Varios de los aportes más importantes en este sentido son: la librería libcolorspaces, que genera una tabla precalculada (3.3.2), y la decisión de comparar los parches en aquellos puntos que existía saliencia y pertenecían a la línea epipolar (4.4.1). Los movimiento sacádicos del ser humano oscilan entre promedios de 100-200 milisegundos, nosotros en nuestro proyecto hemos sido capaces de obtener estos promedios de atención, lo que supone un sistema bastante ágil y rápido.

3. El sistema debía ser robusto en cuanto al cambio de luminosidad que existe entre las imágenes tomadas desde las diferentes cámaras del par estéreo. Las mejoras

que hemos realizado para obtener esta robustez se basan en utilizar la librería libcolorspaces (HSV), y en aplicar los parches en luminancia normalizada (escala de grises).

4. El diseño e implementación (sección 4.1) se han concebido siguiendo la arquitectura JDE. De este modo el software se ha implementado siguiendo la arquitectura de esquemas, adaptando algunos existentes y creando otros específicos para el proyecto.

5. El sistema funciona correctamente bajo la plataforma GNU/Linux, concretamente sobre la distribución archiconocida *Debian* y ejecutando en un kernel *2.6.18*.

6. Por último y no menos importante, la licencia con la que se libera el código es *GPLv3* y la licencia con la que se libera esta memoria es *Creative Commons*.

Este proyecto de ingeniería tiene un matiz de investigación. Al principio se tenía el problema de un contexto bastante abierto del que no se sabía hasta dónde íbamos a llegar, o cuán positivos iban a ser los resultados obtenidos. Por ello ha sido muy importante tener un diseño preliminar sujeto a cambios, una implementación que nos permitiera experimentar con diferentes configuraciones, y un plan de pruebas robusto para poder decidir la configuración ideal en cada situación.

El sistema que hemos realizado incorpora componentes software adicionales, ya existentes de otros proyectos, que a veces han resultado difícil de integrar en nuestro sistema. El calibrador ([Kachach, 2008]) y la extracción de recta epipolar ([Mendoza Baños, 2008]) son ejemplos de ello. Sobre estos antecedentes hemos desarrollado mejoras significativas que han llevado a una funcionalidad completamente nueva.

Son aportes genuinos de este proyecto:

1. El sistema de atención en 2D mediante la técnica de mapa de saliencia.

2. Utilización de un par estéreo de cámaras reales. Analizamos imágenes reales en vez de simuladas (línea futura expuesta en el proyecto [León Cadahía, 2006]).

3. La carga cognitiva del sistema, dándole así cierto *raciocinio* espacial al comportamiento. Este punto es una línea futura de investigación que quedó reflejada en el proyecto [León Cadahía, 2006], y que hemos sido capaces de abordar.

4. La rapidez y vivacidad con la que es capaz de representar objetos complejos, comparado con otros proyectos [Mendoza Baños, 2008].

5. La representación en OpenGL de la escena tridimensional

6. Los aportes y mejoras que este proyecto ofrece a la infraestructura software de *JDE*: (a) la creación y utilización de la librería *libcolorspaces*, que minimiza enormemente el coste temporal de la conversión entre espacios de color, (b) las mejoras realizadas en el calibrador de cámaras y (c) el soporte de acceso a las cámaras firewire.

A grandes rasgos este proyecto fin de carrera ha supuesto un paso adelante significativo sobre los proyectos de [Pacios and Cañas Plaza, 2007] y [Mendoza Baños, 2008] añadiendo a la reconstrucción 3D un sistema atentivo y una carga cognitiva en el comportamiento. Y sobre los proyectos de: [León Cadahía, 2006] y [Martínez De la Casa Puebla, 2005] introduciendo un mapa de saliencia y un comportamiento rápido y genérico para cualquier escena.

Los resultado de este proyecto, documentación, código, vídeos y demás material está disponible en la página web de JDE [1].

6.2. Trabajos futuros

Uno de los caminos más interesantes por los que podríamos continuar este trabajo, es intentar una reconstrucción 3D más densa y agrupando objetos. Ahora mismo en este proyecto se representan los objetos mediante puntos 3D, pero cabría la posibilidad de representar la información mediante primitivas más abstractas y potentes como segmentos, figuras, coches, etc.

Un paso más en la línea de atención visual sería realizar seguimiento atentivo y situación en 3D sobre objetos que están en movimiento muy rápido. En este proyecto hemos realizado seguimiento visual sobre objetos en movimiento, pero su ubicación en 3D es bastante costosa y no tan vivaz, ya que está continuamente emparejando y triangulando.

Otro de los *objetos de interés* de la escena podrían ser personas o caras de personas. Sería interesante emular una atención muy parecida a la de los seres humanos, fijando la atención sobre estímulos más naturales. De esta manera el sistema sería capaz de prestar atención a las caras humanas y a los ojos, que normalmente es donde solemos mirar cuando hablamos con una persona.

Podríamos incrementar el número de objetos sobre los cuales el sistema es capaz de hacer una hipótesis de alguna parte que no aparece de dicho objeto. En este proyecto

[1] http://jde.gsyc.es/index.php/Rocapal_visual_attention_3D

hemos realizado experimentos sobre cómo averiguar la cuarta esquina de un cuadrado. Del mismo modo y ligado al punto anterior, podríamos realizar hipótesis perceptivas sobre una cara humana; presentando atención a la cara y a uno de los ojos, podemos hipotetizar dónde estaría el segundo ojo y saltar allí con un movimiento sacádico y verificar que efectivamente es otro ojo.

Por último, otra línea en la que parece interesante es investigando sería implementar una atención global (*overt*) con cámaras móviles, de esta manera la escena no se reduce únicamente a la imagen que muestran las cámaras sino que las cámaras, mediante un cuello mecánico, pueden moverse en horizontal y vertical para analizar el resto de la escena. De esta manera las cámaras se moverían mediante *movimientos sacádicos* reales muy parecidos a los del ser humano.

Apéndice A

Librería de filtrado: libcolorspaces

Cabeceras de la librería *libcolorspaces*

```
/*
 *  Copyright (C) 2007 Roberto Calvo Palomino
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU Library General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 *
 *  Authors : Roberto Calvo Palomino <rocapal@gsyc.es> ,
 *         Jose Maria Cañas Plaza <jmplaza@gsyc.es>
 */
```

```
#ifndef _COLOR_SPACES_H
#define _COLOR_SPACES_H

#define NAME     "colorspaces"
#define VERSION  "1.3.0"

/// *** RGB to HSI  *** ///

struct HSV
{
double H;
double S;
double V;
};

extern struct HSV * LUT_RGB2HSV [64][64][64];
extern int isInitTableHSV;
extern pthread_mutex_t mutex;

/// \brief Init the RGB2HSV
void RGB2HSV_init();

/// \brief Create a translate RGB2HSV table with resolution of 6bits (64x64x64)
void RGB2HSV_createTable();

/// \brief Free memory of RGB2HSV
void RGB2HSV_destroyTable();

/// \brief Print the struct HSV
void RGB2HSV_printHSI (struct HSV*);

/// \brief Test
void RGB2HSV_test();

/// \brief Returns the translation from RGB to HSV
static inline const struct HSV* RGB2HSV_getHSV (int R, int G, int B)
```

```
{ return LUT_RGB2HSV[R>>2][G>>2][B>>2]; }

/// \brief Returns the translation from HSV to RGB
void hsv2rgb(double H, double S, double V, double *r, double *g, double *b);

#endif
```

Bibliografía

[Bachiller Burgos, 2008] Pilar Bachiller Burgos. Percepción dinámica del entorno en un robot móvil. *Tesis Doctoral - UNEX*, 2008.

[Barrera González, 2008] Pablo Barrera González. Aplicación de los métodos secuenciales de monte carlo al seguimiento visual 3d de múltiples objetos. *Tesis Doctoral - Universidad Rey Juan Carlos*, 2008.

[Calvo Palomino, 2004] Roberto Calvo Palomino. Seguimiento de persona mediante visión direccional. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2004.

[Cañas Plaza, 2003] Jose María Cañas Plaza. Jerarquía dinámica de esquemas para la generación de comportamiento autónomo. *Tesis Doctoral - Universidad Politécnica*, Diciembre 2003.

[Cañas Plaza, 2004] Jose María Cañas Plaza. Manual de programación de robots con jde. *Universidad Rey Juan Carlos*, pages 1–36, abril 2004.

[Cañas Plaza, 2005] Jose María Cañas Plaza. Overt visual attention inside jde control architecture. *EPIA*, 2005.

[García Martínez, 2007] Iván García Martínez. Reconstrucción 3d visual con algoritmos evolutivos. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2007.

[Gómez, 2002] Victor Gómez. Comportamiento sigue pared en un robot con visión local. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2002.

[Itti and Koch, 2004] Laurent Itti and Chistof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 2004.

[Itti and Koch, 2005a] Laurent Itti and Chistof Koch. Computation modelling of visual attention. *Nature Reviews Neuroscience*, 2005.

[Itti and Koch, 2005b] Laurent Itti and Chistof Koch. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[Javier Traver, 2005] Filiberto Pla Javier Traver. Similarity motion estimation and active tracking throught spatial-domain projections on log-polar images. *Science Direct*, 2005.

[Kachach, 2008] Redouane Kachach. Calibración automática de cámaras en la plataforma jdec. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2008.

[Lenser and Veloso, 2003] Scott Lenser and Manuel Veloso. Visual sonar: fast obstacle avoidance using monocular vision. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 886–891, Las Vegas (Nevada, USA), October 2003.

[León Cadahía, 2006] Olmo León Cadahía. Navegación de un robot con un sistema de atención visual 3d. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2006.

[Martínez De la Casa Puebla, 2005] Marta Martínez De la Casa Puebla. Sistema de atención visual en escena. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2005.

[Mendoza Baños, 2008] Manuel Mendoza Baños. Reconstrucción 3d visual mediante triangulación con cámaras. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2008.

[Ortiz, 2004] Ricardo Ortiz. Comportamiento sigue persona con visión. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2004.

[Pacios and Cañas Plaza, 2007] Esteban Pacios and Jose María Cañas Plaza. Reconstrucción 3d visual con triangulación. *Proyecto fin de carrera. ITIS - Universidad Rey Juan Carlos*, 2007.

[S. Birchfield, 1999] C. Tomasi S. Birchfield. Depth discontinuities by pixel-to-pixel stereo. *International Journal of computer Vison*, 1999.

[Sebastian Thrun, 2004] Frank Dellaert Sebastian Thrun. Minerva: A second-generation museum tour-guide robot. *Science Direct*, 2004.

[Sethu Vijjayakumar, 2001] Jörg Conradt Sethu Vijjayakumar. Overt visual attention fo a humanoid robot. *International Conference on Intelligence in Robotics and Autonomous System*, 2001.

[Turégano Pedruelo, 2008] Enrique Turégano Pedruelo. Eyeboard: Un periférico alternativo visual. *Tesis de maestría - UNEX*, 2008.