
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN


# REALIDAD AUMENTADA EN INTERFACES HOMBRE MÁQUINA

## TESIS

QUE PARA OBTENER EL GRADO DE

**MAESTRO EN CIENCIAS DE LA COMPUTACIÓN**

PRESENTA:


**GILBERTO NÁJERA GUTIÉRREZ**

DIRECTORES:

**DR. RICARDO BARRÓN FERNÁNDEZ**
**DR. GUSTAVO OLAGUE CABALLERO**



México, D.F., junio de 2009

# Realidad aumentada en interfaces hombre-máquina

Gilberto Nájera Gutiérrez

Junio de 2009

SIP-14

# INSTITUTO POLITECNICO NACIONAL

## SECRETARIA DE INVESTIGACIÓN Y POSGRADO

*ACTA DE REVISIÓN DE TESIS*

En la Ciudad de México, D.F. siendo las 12:00 horas del día 17 del mes de Junio de 2009 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

***Centro de Investigación en Computación***

para examinar la tesis de grado titulada:

**"REALIDAD AUMENTADA EN INTERFACES HOMBRE-MÁQUINA"**

| **NÁJERA** | **GUTIÉRREZ** | **GILBERTO** |
|---|---|---|
| Apellido paterno | materno | nombre(s) |

Con registro: | A | 0 | 7 | 0 | 2 | 4 | 2 |

aspirante al grado de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

**Presidente**

Dr. Juan Humberto Sossa Azuela

**Secretario**

Dra. Nareli Cruz Cortés

**Primer vocal**
**(Director de tesis)**

Dr. Ricardo Barrón Fernández

**Segundo vocal**

Dr. Salvador Godoy Calderón

**Tercer vocal**

Dr. Rolando Quintero Téllez

**EL PRESIDENTE DEL COLEGIO**

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

# *INSTITUTO POLITECNICO NACIONAL*

## *SECRETARÍA DE INVESTIGACIÓN Y POSGRADO*

## *CARTA CESION DE DERECHOS*

En la Ciudad de MÉXICO, D. F. el día 18 del mes JUNIO del año 2009, el (la) que suscribe GILBERTO NÁJERA GUTIÉRREZ alumno (a) del Programa de MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN con número de registro A070242, adscrito a CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de DR. RICARDO BARRÓN FERNÁNDEZ Y DR. GUSTAVO OLAGUE CABALLERO y cede los derechos del trabajo intitulado REALIDAD AUMENTADA EN INTERFACES HOMBRE-MÁQUINA, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección gilnajera@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

GILBERTO NÁJERA GUTIÉRREZ

Nombre y firma

**Resumen**

En el presente trabajo se propone un modelo de desarrollo de aplicaciones basadas en realidad aumentada sin marcadores, que incorpora un algoritmo sencillo de simulación de la superposición de objetos y con bajos requerimientos tanto en *hardware* y *software*. Para ello se propone la utilización de detectores y descriptores de puntos de interés, aprendizaje de máquina y geometría proyectiva para la calibración de la cámara y un método de seguimiento para la posterior operación. También se muestra la aplicación de el modelo propuesto en un sistema asistente de dibujo basado en realidad aumentada.

**Abstract**

In this work, we propose a model for the development of augmented reality based applications, those applications can be markerless. The proposed model incorporates an easy-to-implement occlusion simulation algorithm, that is to simulate occlusion between real and virtual objects in the scene. The model uses interest point detectors and descriptors, machine learning and projective geometry to calibrate the camera and to set the virtual objects in the scene. Also uses a tracking algorithm. At the end, the model is implemented in an application, an augmented reality based drawing assistant.

# Índice general

# Índice de tablas

# Índice de figuras

# Capítulo 1

# Introducción

El ser humano está naturalmente provisto de sentidos que le permiten interactuar con su entorno y con otros seres, puede ver, oler, tocar, escuchar y gustar cualquier cosa que esté al alcance de sus sentidos, esto le permite identificar, entender e incluso transformar los elementos que lo rodean. A pesar de contar con estos sentidos, han habido siempre tareas peligrosas, costosas o difíciles de ejecutarse, para facilitar la realización de dichas tareas, el ser humano crea máquinas, las mismas que requieren ser operadas por seres humanos y requieren, a su vez, un mecanismo simplificado de interacción con sus operadores.

Conforme la tecnología ha avanzado, el diseño de la máquinas se ha vuelto más complicado y la necesidad de mecanismos sencillos y efectivos de comunicación entre estas y sus usuarios es cada vez mayor.

Recordando un poco la historia de las máquinas, desde las más simples, nos encontramos con que las primeras maneras que se tenían para interactuar con ellas era la manipulación directa de máquinas simples, después, cuando se fueron haciendo combinaciones de dichas máquinas; la operación se hacía por medio de ruedas, cuerdas, poleas y palancas. Con el desarrollo de la electricidad y la electrónica, estos mecanismos se fueron sustituyendo por botones y palancas más pequeñas que hacían de interruptores para accionar o apagar máquinas que a su vez encendían o apagaban otras máquinas.

Con el nacimiento de las computadoras digitales y los principios de la robótica, estas formas de interacción se han ido modificando de tal manera manera que, actualmente, en varios campos de actividad humana, no hay una interfaz directa entre la máquina que realiza el trabajo y su operador, sino que éste da órdenes a una computadora y es ella quien hace que el resto de la maquinaria actúe de una manera u otra.

Las formas de interfaz humano - computadora también han sufrido grandes cambios, quizá más grandes que la interacción con cualquier otra máquina; podemos comenzar hablan-

Figura 1.1: La realidad mezclada (mixed reality) son todas aquellas técnicas que presenten información al ser humano por medio de su inmersión en un entorno que combina objetos virtuales con el mundo real. Ordenadas de izquierda a derecha, según la proporción de elementos sintéticos con respecto a los reales: realidad física, realidad aumentada, virtualidad aumentada, realidad virtual

do de las enormes tarjetas perforadas que contenían las instrucciones que la computadora debía ejecutar, estas eran introducidas por los programadores, que recibían un resultado algunas horas o incluso días después. En los últimos años se han buscado maneras para hacer más intuitiva y agradable la experiencia de operar una computadora, de manera que se han desarrollado técnicas como las interfaces hápticas que permiten al usuario, mediante el uso de un equipo especial, sentir o tocar objetos virtuales; Se han hecho grandes avances también en las áreas de realidad virtual y realidad aumentada, donde el usuario se introduce en un mundo creado por la computadora con la posibilidad de realizar cambios en él (realidad virtual) o en una captura de video del mundo real se introducen objetos virtuales con los que el usuario puede interactuar en tiempo real (realidad aumentada).

## 1.1. Realidad aumentada

La *Realidad Aumentada* (RA) es una variación de los *ambientes virtuales* o *realidad mezclada*, ver figura 1.1. A diferencia de las otras técnicas de realidad mezclada, la realidad aumentada permite que el usuario perciba el entorno real *"aumentado"* con algunos objetos virtuales (creados por computadora). En circunstancias ideales, debería parecer al usuario que los objetos reales y virtuales coexisten en el mismo espacio.

### 1.1.1. Características de la realidad aumentada

Para que una aplicación pueda ser definida como una aplicación de realidad aumentada deberá cumplir con los siguientes requisitos [6]:

- Combinar objetos virtuales con el mundo real
- Ser interactiva en tiempo real

- Los objetos virtuales se presentan en tres dimensiones

**Tiempo real** [1] Se dice que una aplicación es de tiempo real cuando ésta requiere que un programa responda a estímulos dentro de un límite de tiempo muy pequeño (usualmente milisegundos o microsegundos).

### 1.1.2. Reseña histórica

Si se expande el concepto de realidad aumentada a los diferentes medios por los que los humanos percibimos el mundo, es decir, aumentar la realidad no solo para la vista, sino también para el tacto, oído, olfato e incluso el gusto; podremos ver que el ser humano ha buscado "complementar" la realidad con ciertos elementos que le pueden ayudar, tanto a realizar sus tareas básicas como a explicarse lo que pasa a su alrededor. Tal podría ser el caso de los seres y personajes mitológicos de las culturas antiguas, que aunque no formaban parte del mundo en que las personas de la época vivían, si formaban parte de su realidad, algunos incluso eran objeto de culto y se les atribuía la posibilidad de afectar sucesos tan importantes como el cambio entre día y noche.

En un contexto más actual y enfocado en los desarrollos tecnológicos, podemos citar los siguientes hitos en el desarrollo de las técnicas de aumentado de la realidad[2].

- El cineasta Morton Heilig (1962), crea un simulador de motocicleta (*sensorama*) con sonido, vibración, efectos visuales y olores.

- Ivan Sutherland (1966) inventa el Monitor Montado en la Cabeza (Head Mounted Display, HMD), con el cual podía visualizar objetos virtuales y el mundo real al mismo tiempo.

- En 1975 Myron Krueger presenta *Videoplace*, máquina que permitía por primera vez a los usuarios interactuar con objetos virtuales.

- El término "Realidad virtual" nace en 1989, cuando Jaron Lanier crea el primer negocio comercial en mundos virtuales.

- En 1990, mientras trabajaba para Boeing en un sistema que mostraba diagramas de cableado en un monitor montado en la cabeza (HMD[3]), Tom Caudell crea el término "Realidad aumentada" [17].

[1] "real time" Free OnLine Dictionary of Computing. http://foldoc.org
[2] Wikipedia. $http://en.wikipedia.org/wiki/Augmented_reality$
[3] Del inglés *Head Mounted Display*

- A partir de mediados de los 90, debido a las nuevas capacidades tecnológicas y a la búsqueda de alternativas que faciliten y hagan más productiva la interacción entre personas y computadoras, la investigación y el desarrollo en realidad aumentada se han incrementado enormemente [7].

- En noviembre de 1998 se realiza el primer Taller Internacional de Realidad Aumentada (IWAR, International Workshop on Augmented Reality) que en 2000 se convertirá en ISAR *(International Simposium on Augmented Reality)* y en 2002 en ISMAR (*International Simposium on Mixed and Augmented Reality*).

### 1.1.3. Aplicaciones de la realidad aumentada

Aunque las técnicas de realidad mezclada se pueden implementar en cualquier medio en el que se requiera que el usuario obtenga información adicional del mundo o se involucre en mayor medida con los programas o realice tareas que implicarían cierto riesgo si se hicieran en entornos reales, no en todos estos campos puede resultar ventajosa su utilización, ya sea por cuestiones de practicidad en la implementación, costos, entre otras. Los campos en los que las soluciones en realidad aumentada han probado ser eficientes o al menos muestran tener posibilidades de una implementación eficiente y, por tanto, en los que la mayoría de los investigadores [6] se han enfocado son:

- Medicina

  Los médicos podrían usar la realidad aumentada como ayuda en la visualización y el entrenamiento en las cirugías. Es posible recolectar datos tridimensionales del paciente en tiempo real, usando sensores no invasivos como resonancia magnética, ultrasonido o tomografía por computadora. Estos datos se presentarían y combinarían con una vista real del paciente dándole al médico una especie de *visión de rayos X*, [74].

  Hay incluso intentos de tratamiento de algunas fobias agregando virtualmente el objeto causante de la fobia en el campo de visión del paciente [40].

- Manufactura y reparación

  El ensamblaje, mantenimiento y reparación de maquinaria compleja es un área en la que la RA puede rendir grandes frutos. Sería mucho más sencillo para un técnico si, en lugar de leer un manual y después proceder a hacer la reparación de determinado aparato, pudiera tener las instrucciones (que incluso podrían ser animadas, interactivas y sensibles a los cambios del entorno) a la vista mientras realiza su trabajo. Es decir, se tendría un asistente que vaya indicando que partes cambiar, quitar o mover, cuando hacerlo y de que manera mientras está realizando dichas operaciones [64].

- Anotación y visualización

  Aunque el ser humano puede percibir gran porción de la realidad a través de sus sentidos, siempre hay cosas que escapan a ellos, no se perciben con la precisión requerida para ciertas operaciones o se perciben, pero se desconocen algunos detalles. En estos casos tal información podría ser captada por sensores u obtenida de bases de datos y mostrada al usuario del sistema de RA en una posición significativa según el entorno.

  Algunos ejemplos de estas aplicaciones son las guías, mayormente turísticas, en las que ya hay algunos trabajos de investigación [26], sistemas mediante los cuales una persona puede probarse prendas y accesorios virtuales antes de adquirirlos o los que permiten observar los planos de una construcción superpuestos sobre la misma [61], entre muchos otros.

- Planeación de rutas de robots

  Para el caso de operar un robot a distancia, podría ser una buena alternativa que el usuario pueda planear los movimientos del mismo por medio de una versión virtual que se ubique en el ambiente real, una vez que se ha decidido la ruta idónea entonces se envían los datos al robot para que ejecute las operaciones previamente realizadas en su versión virtual[6].

- Entretenimiento

  Sin duda una de las grandes posibilidades de la RA se encuentra en la industria del entretenimiento, actualmente podemos ver varias aplicaciones sencillas de realidad aumentada en algunas transmisiones por televisión: la línea virtual de primera y diez en los partidos de futbol americano o la de fuera de lugar en el futbol, anotaciones personalizadas siguen a cada auto, durante una carrera, publicidad virtual en los programas en vivo [7], entre otras aplicaciones.

  Los juegos de video son otro campo en el cual la realidad aumentada podía tener una muy provechosa aplicación. En la actualidad hay varios grupos y empresas realizando investigación y desarrollo en materia de realidad aumentada aplicada a la creación de juegos que aprovechen las nuevas tecnologías para brindar al usuario una mayor sensación de realismo a la hora de jugar.

- Educación

  Al igual que muchas otras tecnologías, la realidad aumentada puede ser una herramienta que colabore en mejorar la experiencia de aprendizaje de gran cantidad de personas,

desde museos que creen una experiencia prehistórica en la sala donde se exhiben fósiles [37] o libros para niños que muestren escenas tridimensionales en lugar de fotografías y dibujos planos incluso cursos de geometría, cálculo, entre otras, donde se puedan manipular puntos tridimensionales (¡En el espacio tridimensional!) o conversaciones en el salón de clase con personajes "traídos virtualmente" del pasado.

- Milicia

  Las actividades y entrenamiento militares siempre han tenido, en mayor o menor grado, algún reto a vencer para la ciencia, podemos mencionar el cálculo de las trayectorias de los proyectiles (desde los arcos y las catapultas hasta los tanques y los aviones bombarderos), entre un sinnúmero de aplicaciones. En cuanto al tema que nos interesa, se puede mencionar que los cascos de los pilotos de aeronaves militares cuentan desde hace varios años con visores semirreflejantes sobre los cuales se proyectan gráficos con información sobre rutas, objetivos, el estado de la nave y otros [83]. También se están desarrollando simuladores de operaciones en ambientes urbanos y ayudas visuales para tales situaciones en las cuales, gracias a un conjunto de sensores, un soldado podría saber dónde se encuentran sus compañeros aún si una o varias paredes obstruyen su visión [37].

Las aplicaciones posibles de la realidad aumentada pueden ser tantas y tan variadas como la imaginación pueda concebirlas. Sin embargo, debido a que esta técnica se basa en la visión artificial y los gráficos por computadora en tiempo real, estando ambas ramas en desarrollo y con mucho camino por recorrer, dichas aplicaciones se ven acotadas por los avances tecnológicos de la época y, por lo tanto, muchas de ellas se quedan como proyectos de laboratorio o se ven restringidas al grado de volverse infactibles. Aún así, la investigación y lo adelantos continúan y cada día hay más proyectos funcionales dando buenos resultados y aumentando el interés de la comunidad científica, comercial e industrial por el área.

### 1.1.4. Consideraciones para un sistema de RA

En este apartado se verán algunos aspectos importantes de diseño a considerar a la hora de elaborar un sistema de RA, así como las características principales del mismo. Se comienza por la descripción de lo que se entiende por aumentado. Luego se analizan las características, ventajas y desventajas de los dos enfoques de aumentado más utilizados actualmente, el óptico y el de video. También se explican brevemente los problemas con el foco de la cámara y el contraste que supone la mezcla de objetos virtuales en el mundo real, además, debido a que algunas aplicaciones requieren sistemas móviles de RA para ser en verdad efectivos se presentan también dificultades relacionadas con la portabilidad. Finalmente se comparan los requisitos de la RA con las de otro tipo de ambientes virtuales o realidades mezcladas.

### Aumentado

Aunque hasta ahora el enfoque principal de los investigadores ha sido aumentar la realidad por medio de la adición de imágenes y gráficos, el aumentado podría ser para todos los sentidos y no sólo para la vista, el uso de sonidos (de preferencia envolventes) agregaría una sensación mayor de inmersión y de interacción. Otro ejemplo son las interfaces hápticas, el uso de un guante que simulara ciertas estímulos táctiles en los dedos podrían hacer sentir al usuario que está realmente tocando determinado objeto virtual.

### Óptico o video

Una decisión básica a la hora de construir un sistema de RA es cómo realizar la combinación de lo real y lo virtual. Las dos opciones básicas son las tecnologías ópticas o las de video. Como se verá a continuación, cada una tiene sus ventajas y desventajas.

Un monitor montado en la cabeza (HMD[4]) óptico funciona por medio de la colocación de combinadores ópticos en frente de los ojos del usuario. Estos combinadores son semitransparentes, de manera que se puede ver el mundo real a través de ellos e imágenes virtuales reflejadas sobre los mismos. La figura 1.2 muestra el esquema clásico de un sistema de RA que utilice combinadores ópticos.

Los combinadores ópticos, al no ser completamente transparentes, reducen la cantidad de luz que el usuario percibe del mundo. Como ejemplo, algunos dejan pasar sólo un 30 % de la luz. Esto hace que la elección del nivel de mezclado y los dispositivos a utilizar constituyan un problema importante de diseño. Hay algunos combinadores bastante sofisticados que permiten seleccionar que longitudes de onda de luz serían reflejadas y cuales no, lo que sería ideal para sistemas monocromáticos.

A diferencia de los anteriores, los sistemas de realidad aumentada por medio de video utilizan un HMD de vista cerrada, es decir, que una vez colocado impide ver el mundo real. A este HMD se le agregan una o dos cámaras que hacen las veces de "ojos" del usuario hacia el mundo real. El video que estas cámaras captan se combina con los gráficos virtuales creados por el generador de escenas y el resultado se muestra en las pantallas internas del HMD frente a los ojos del usuario. La figura 1.3 muestra un esquema de como funciona generalmente una construcción de este tipo.

Hay varias maneras de hacer la composición del video, una simple podría ser usar la técnica de pantalla verde o azul (en inglés *chroma-keying)* usada en muchos efectos especiales

[4]Del inglés *Head Mounted Display*

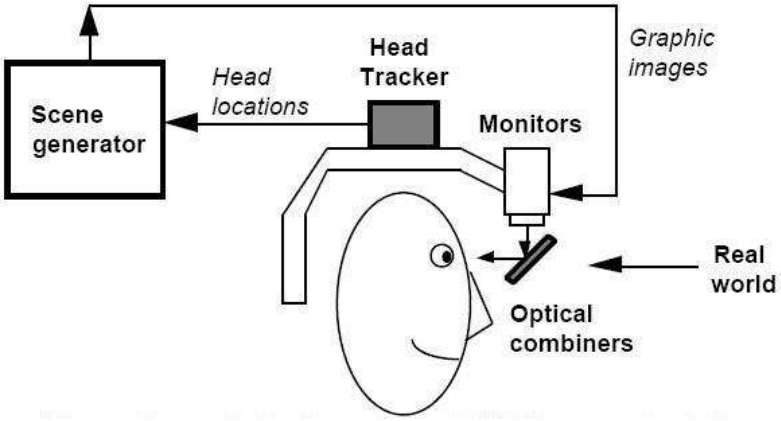



Figura 1.2: Realidad aumentada usando un HMD óptico

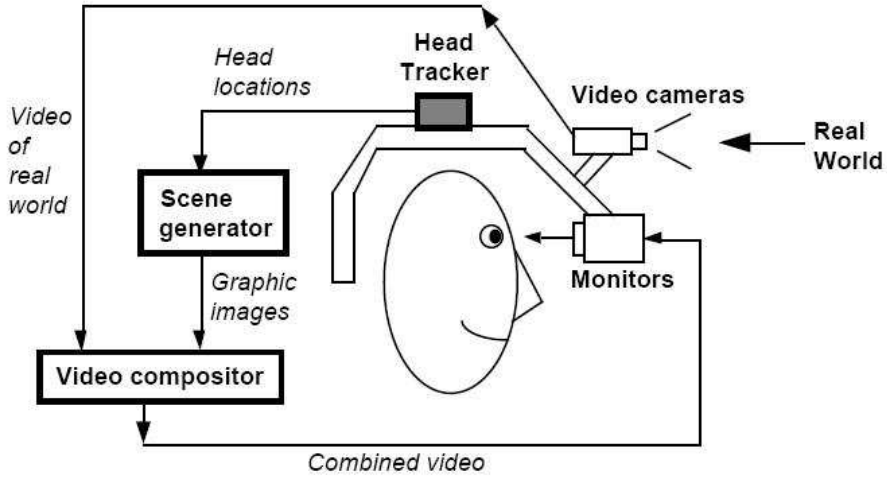



Figura 1.3: Realidad aumentada usando un HMD basado en video

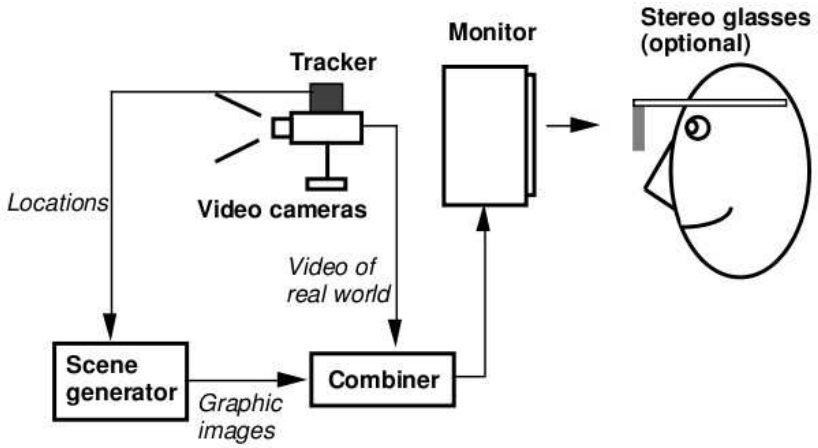



Figura 1.4: Realidad aumentada con un esquema basado en monitor de escritorio

en video. El fondo de los gráficos virtuales se hace de un color fijo, verde, por ejemplo, el cual ninguno de los objetos en la escena utilizan. A la hora de hacer la combinación se reemplazan todas las áreas verdes con las partes de video del mundo real correspondientes, causando el efecto de superposición de los gráficos virtuales sobre el mundo real. Una composición más sofisticada podría utilizar información de profundidad. Si el sistema tiene información de la profundidad de cada píxel en las imágenes del mundo real, se puede realizar la combinación de objetos reales y virtuales por medio de una comparación de profundidad pixel a pixel. Esto permitiría que objetos reales "cubran" a los virtuales y viceversa.

También se puede construir sistemas de RA usando configuraciones basadas en un monitor de escritorio en lugar de usar algún tipo de HMD. La figura 1.4 muestra un esquema de como se podría organizar un sistema de RA basado en monitor de escritorio. En este caso, una o dos cámaras de video, que pueden ser móviles o fijas, capturan el ambiente. En el caso móvil, las cámaras pueden ser portadas por un robot cuyas posiciones serán constantemente rastreadas. La combinación del video del mundo real y objetos virtuales se hace de la misma manera que en el caso de la configuración basada en HMD de video y se muestra en un monitor frente al usuario, el cual no tiene que cargar el dispositivo de visualización. Opcionalmente, se puede utilizar un monitor estereoscópico, en cuyo caso será necesario que el usuario use lentes especiales para dicho monitor.

Finalmente, también es posible una configuración óptica basada en monitor. Esto sería similar a la figura 1.4, sólo que el usuario no usaría los combinadores ópticos en su cabeza, sino que estos estarían fijos en el espacio y se vería a través de ellos.

El resto de esta sección compara las ventajas y desventajas relativas a cada enfoque.

Iniciamos viendo las ventajas del enfoque óptico sobre el de video:

1. **Simplicidad:** La mezcla óptica es más sencilla y barata que la de video. Los enfoques ópticos tienen únicamente una secuencia de video de la cual preocuparse: los gráficos artificiales. El mundo real se ve directamente a través de los combinadores y el retraso es generalmente de unos nanosegundos. Por otro lado, la mezcla de video debe tratar con secuencias de video diferentes para el mundo real y los objetos virtuales, ambas con retrasos inherentes de decenas de milisegundos. Digitalizar imágenes de video usualmente añade al menos un *tiempo de cuadro*[5] de retraso a la secuencia de video. Un monitor que actualiza la pantalla a 60 Hz tiene un tiempo de cuadro de 16.67 ms. Además, las dos secuencias de imágenes reales y virtuales deben ser sincronizadas adecuadamente o de lo contrario se producirán distorsiones. También, algunos HMD ópticos con combinadores con campo de visión limitado presentan vistas del mundo real con una pequeña distorsión. Debido a que el enfoque por video requiere cámaras y otros dispositivos que los enfoques ópticos no necesitan, un sistema basado en video podría resultar más caro y difícil de construir que un basado en óptica.

2. **Resolución:** La mezcla de video limita la resolución de los objetos, tanto reales como virtuales, a la resolución de los dispositivos de visualización. Las pantallas actuales están muy lejos aún del poder de resolución de la visión humana. Es cierto que los combinadores ópticos también muestran los gráficos virtuales a la resolución que el dispositivo permite, pero la vista del mundo real no es degradada.

3. **Seguridad:** Un HMD con visión por video es esencialmente un HMD de vista cerrada al que se le integraron cámaras. Si se interrumpe la energía del HMD, el usuario quedaría prácticamente ciego, lo significa un riesgo de seguridad en algunas aplicaciones. Por el contrario, si se diese un fallo de energía en un HMD óptico el usuario aún podría ver el mundo real y el el de video se convertiría en unos lentes de sol muy pesados.

4. **Desplazamiento de los ojos:** Con la visión por video, la visión del usuario es la que obtiene de las cámaras de video. En esencia, sus ojos están donde están las cámaras. En la mayoría de las configuraciones las cámaras no se colocan exactamente donde deberían ir los ojos del usuario, lo que crea un *desplazamiento* entre las cámaras y los ojos del usuario. Esta diferencia entre colocaciones (de los ojos y las cámaras) crea discordancias entre lo que el usuario ve y lo que espera ver. Aunque se puede minimizar este problema por medio de la utilización de espejos en una configuración tal que hagan que las cámaras capten lo que los ojos del usuario captarían, esto añade complejidad

[5]Tiempo necesario para procesar un cuadro o fotograma de video

al diseño. Los HMD ópticos, por otro lado, no tienen o al menos no con tal dificultad, el problema del desplazamiento. Aunque los ojos del usuario pueden rotar con respecto a la posición del HMD, lo errores resultantes son muy pequeños. Además, si se usa el centro de rotación de los ojos como punto de visión de los gráficos por computadora el modelo podría eliminar cualquier necesidad de rastrear los ojos en un HMD óptico.

Una configuración con visión por video tiene las siguientes ventajas sobre una óptica:

1. **Flexibilidad en estrategias de composición**:Un problema básico con las opciones ópticas es que lo objetos virtuales no cubren completamente a los reales, debido a que los combinadores ópticos permiten ver la luz tanto del mundo real como de la fuente virtual, esto que provoca que los objetos virtuales se vean semitransparentes y por consiguiente afecta en gran medida la sensación de realidad, ya que la oclusión es una de las principales señales de profundidad. En contraste, la visión por video es mucho más flexible en cuanto a la manera de mezclar las imágenes reales y virtuales ya que ambas las tiene en forma digital. También se puede generar la sensación de transparencia tanto en los objetos virtuales como en los reales. Debido a esta flexibilidad, la visión por video podría producir entornos más convincentes que aquellas configuraciones basadas en combinadores ópticos.

2. **Campo de visión amplio**: Las distorsiones en los sistemas ópticos se dan en función de la distancia radial desde el eje óptico. Entre más lejos del centro de visión se mire, más grandes serán las distorsiones. Una imagen digital tomada a través de un sistema con distorsión se puede corregir usando técnicas de procesamiento de imágenes. Aunque esto requiere un tiempo de cómputo significativo, pero esto será cada vez menos importante conforme las computadoras se van haciendo más rápidas. Es más difícil construir monitores con campo de visión amplio usando técnicas de visión a través de óptica. Cualesquiera distorsiones en la vista del mundo real se deberían corregir ópticamente, en lugar de digitalmente, porque el sistema no tiene un versión digital del mundo real que pueda manipular. La óptica compleja es costosa y añade peso al HMD. Los sistemas de campo de visión amplio son una excepción a la tendencia general de los enfoques ópticos a ser más baratos que los enfoques por video.

3. **Sincronización de retardos en la vista real y virtual**: El video ofrece un enfoque para reducir o evitar los problemas causados por discordancias temporales entre las imágenes virtuales y reales. Los HMD ópticos ofrecen una visión instantánea del mundo real y una un tanto retardada del mundo virtual. Este error temporal puede causar problemas. En los enfoques con video es posible retrasar un poco las imágenes del mundo real para que concuerden con las virtuales.

4. **Estrategias de alineación adicionales**: En un HMD óptico, la única información que el sistema tiene sobre la localización del usuario es el rastreador en la cabeza. La mezcla por video ofrece otra fuente de información: la imagen digitalizada de la escena real. Esta imagen puede ser utilizada para aplicar otras estrategias de alineación que no están disponibles en el enfoque óptico.

5. **Control de propiedades visuales**: Mayor facilidad para empatar el brillo de objetos reales y virtuales. Esto se trata en el siguiente apartado.

Cada uno de los enfoques cubre las necesidades de determinadas aplicaciones y la elección de uno u otro dependerá de las mismas. Muchos de los prototipos para ensamblaje y reparación usan equipos con combinadores ópticos, posiblemente por el costo y cuestiones de seguridad. Si el equipo tiene éxito, podría ser replicado a gran escala para equipar a los trabajadores de una línea de producción. Por otro lado, la mayoría de los prototipos para aplicaciones médicas usan el enfoque de video, probablemente por la flexibilidad a la hora de mezclar lo real y lo virtual y por las estrategias de alineación que ofrece.

**Enfoque y contraste**

El enfoque (o foco) puede ser un problema tanto para las opciones ópticas como para las de video. Idealmente, los objetos virtuales deberían concordar con los reales. En un sistema basado en video, las imágenes (real y virtual) combinadas serán proyectadas a la misma distancia por la pantalla o el HMD. Sin embargo, dependiendo de la profundidad de campo de la cámara y la configuración del foco, algunas partes del mundo podrían no enfocarse correctamente. Para simular esto, los gráficos deberán *renderizarse* de manera que simulen el efecto de desenfoque según la distancia y la cámara podría tener una lente con autoenfoque.

En un caso óptico típico, la imagen virtual se proyecta a cierta distancia del usuario. Esta distancia puede ser ajustable, aunque comúnmente se mantiene fija. Por lo tanto, mientras que los objetos reales cambian su distancia constantemente con respecto al usuario, los objetos virtuales se proyectan siempre a la misma distancia. Si las distancias real y virtual no se ajustan automáticamente para los objetos que el usuario está viendo en particular, puede no ser posible ver claramente ambas secuencias (real y virtual) a la vez.

El contraste es otro aspecto problemático debido al amplio rango en los ambientes reales y el lo que el ojo humano puede ver. Idóneamente, el brillo de los objetos reales y el de los virtuales en la misma escena deberían coincidir; desafortunadamente, en el peor de los casos esto significa que el sistema debería poder igualar un enorme rango de niveles de brillo. El ojo es un detector logarítmico, donde la luz más brillante que puede manejar es alrededor

de once niveles mayor que la menos brillante, incluyendo la adaptación a la oscuridad y a la luz. En cualquier estado de adaptación el ojo puede manejar hasta seis niveles de magnitud. La mayoría de los dispositivos de visualización no se acercan aún a este nivel de contraste. Esto es particularmente un problema con las tecnologías ópticas, ya que el usuario tiene vista directa del mundo real. Si el ambiente real es muy brillante estropeará la imagen virtual; si el ambiente es muy oscuro, entonces la imagen virtual resaltará sobre el mundo real. Los problemas de contraste no son tan graves en el caso del video, ya que las mismas video cámaras tienen una respuesta dinámica limitada y tanto la vista real como la virtual son generadas por el monitor, así que todo debe ser ajustado al rango dinámico del mismo.

### Portabilidad

En casi todos los sistemas de realidad mezclada, el usuario no se ve motivado a caminar a su alrededor. En lugar de eso, navega a través del ambiente volando, caminando en una banda móvil o conduciendo alguna especie de vehículo. Cualquiera que sea la tecnología usada, el resultado es que el usuario siempre permanece en el mismo lugar en el mundo real.

Algunas aplicaciones de Realidad Aumentada, sin embargo, necesitarán dar asistencia a un usuario que camine por ambientes extensos. La RA requiere que el usuario esté realmente en el lugar donde la tarea se ha de realizar. El "vuelo" como se realiza en otros ambientes virtuales, ya no es una opción. Si un mecánico debe ir al otro lado del motor de un avión, deberá moverse físicamente junto con el dispositivo de visualización que utilice. Por lo tanto, los sistemas de RA deberán poner énfasis en la portabilidad, especialmente en la posibilidad de operar en exteriores y otros ambientes no controlados. El generador de escenas del HMD y el sistema de rastreo deben ser independientes de equipos no portátiles y capaces de operar y resistir en exteriores. Si se alcanzan esta capacidades, quedarán disponibles muchas aplicaciones que no se han intentado. Por ejemplo, la habilidad de hacer anotaciones sobre los alrededores podría ser útil para soldados, turistas o viajeros que se encuentren en algún lugar desconocido para ellos.

### Comparación con los ambientes virtuales

Todos los requisitos de la RA pueden ser resumidos en una comparación contra los ambientes virtuales (realidad virtual) con base en tres subsistemas básicos presentes en ambos.

1. Generador de escenas

   El *renderizado* no es uno de los mayores problemas de la RA actual. Los sistemas de realidad virtual tienen requisitos mucho más altos en cuanto a imágenes realistas porque remplazan completamente el mundo real con un ambiente virtual. En RA las imágenes

sólo complementan el mundo. Además son pocos los objetos que requieren ser dibujados y no tienen que ser presentados de manera fotorrealista para cumplir con los propósitos de la mayoría de las aplicaciones, aunque idealmente deberían ser presentados de esta manera.

2. Dispositivos de visualización

   Los dispositivos usados para RA pueden no tener requisitos tan rigurosos como los de los usados para Ambientes Virtuales, debido nuevamente a que la RA no remplaza el mundo real. Por ejemplo, algunas aplicaciones de RA se pueden servir de monitores monocromáticos, mientras que todas las aplicaciones actuales de Ambientes Virtuales utilizan dispositivos a todo color. Los HMD con combinadores ópticos con un campo de visión pequeño aún son útiles, ya que el usuario puede ver el mundo real con su visión periférica. Además, la resolución de estos monitores puede ser más baja de lo que un usuario toleraría en una aplicación de RV.

3. Rastreo y sensado

   Mientras que en los casos anteriores los requisitos de la RA son menores que los de otro tipo de ambientes virtuales, no es el mismo caso para los procesos de rastreo y sensado. En esta área los requisitos de la RA son mucho más estrictos que los de los sistemas de Realidad Virtual. La principal razón para esto es el problema de la alineación, el cual se describe en el siguiente apartado. Otros factores se tratarán en secciones posteriores.

### 1.1.5. El problema de la colocación (alineación) de objetos

En este apartado se describe el ya mencionado problema de la colocación de los objetos virtuales y su correcta alineación[6] con los elementos del mundo real en aplicaciones de realidad aumentada.

El objetivo principal de la RA es lograr la complementación de la realidad que el usuario percibe por si solo, con información virtual que sea de utilidad para que dicho usuario realice determinada tarea. Esta información deberá, por tanto, estar coordinada con los objetos y sucesos del mundo real.

En el aumentado de la realidad por medio de objetos visuales, que es la forma más común de realidad aumentada y la que compete a este documento. Un aspecto básico para que la información que se agrega sea de utilidad es que los elementos virtuales se coloquen de una manera coherente con respecto a los objetos reales. Es decir, que su alineación en las tres dimensiones sea correcta. Por ejemplo, en una aplicación de asistencia para el ensamblado o

[6]En inglés; *Registration*

reparación de maquinaria es necesario que los nombres de las piezas, así como las instrucciones sobre cuales de ellas se deben reemplazar, o mover en determinado sentido, sean colocados de manera que resulte claro a que pieza se refiere cada etiqueta o instrucción, ya que de no ser así, se corre el riesgo de que el técnico confunda dos partes, cambie alguna que no requería ser cambiada, entre otras cosas que sería mejor evitar.

Hay otras aplicaciones donde la precisión en la alineación de los objetos es aún de mayor importancia. Supóngase el caso de una cirugía asistida por medio de realidad aumentada. En una aplicación de ese tipo una falla de unos cuantos milímetros en la colocación de la guía para la aguja o el bisturí puede resultar en una intervención fallida o incluso en la pérdida de la vida del paciente, en un caso extremo.

Por otro lado, en un sistema de anotación de lugares turísticos se pueden tolerar fallas en la colocación de las etiquetas de hasta algunos metros reales. Esto debido al tamaño de los edificios, a que no importa si el nombre de algún lugar aparece arriba, abajo o en medio de este y a que no hay ningún riesgo importante en caso de que algún nombre o dato aparezca colocado en un lugar al que no corresponde.

Aunque la importancia del problema de la alineación es dependiente de la aplicación que se esté realizando y aunque puede que nunca se llegue a solucionar por completo [21], gran parte de la investigación actual en RA se enfoca a eliminar, minimizar o predecir el error la alineación con el fin de tener sistemas con una mejor interacción con el mundo real.

### 1.1.6. El problema de seguimiento

En este apartado se describe el problema del seguimiento[7] o rastreo del usuario y los objetos reales en una aplicación de realidad aumentada.

Como ya se mencionó anteriormente, para que un sistema de RA se considere útil, deberá complementar la información que el usuario percibe del mundo; de manera que la información que tal sistema agregue ha de ser acorde a lo que se percibe a través del medio de visualización seleccionado. Entonces, es necesario poder ubicar al usuario y a los objetos reales que sean de interés para la aplicación con respecto a los dispositivos de entrada (cámaras y otros sensores) o viceversa. Se deberá también poder también realizar el seguimiento de dichos objetos y la actualización de la vista de los objetos virtuales mientras dure la ejecución.

[7] En inglés: *Tracking*

### 1.1.7. El problema de la superposición de objetos

Aunque para algunos usos de la realidad aumentada puede ser suficiente presentar información virtual sobre los elementos reales, sin preocuparse por cuál objeto debería aparecer delante o detrás de algún otro, para otros esta tarea se hace vital para su funcionamiento.

Cuando se agregan objetos virtuales a una secuencia de video, éstos generalmente se insertan sobre la imagen que se presentará en en pantalla, de manera que de forma automática tales objetos estarán por encima de cualquier objeto real que la cámara haya captado. En ocasiones será necesario que algunos objetos virtuales parezcan ocultos total o parcialmente por los objetos que aparecen en el video (reales), entonces se requiere encontrar un método que permita ocultar o no dibujar las partes de objetos virtuales que aparecerán detrás de objetos reales y viceversa.

Cómo se podrá inferir, mediante el aumentado por video, esta tarea se hace mucho más sencilla que utilizando un aumentado con monitores ópticos, dónde no se puede modificar la información visual del mundo que el usuario percibe.

## 1.2. Objetivos

El objetivo central del presente trabajo es proponer un modelo genérico de aplicación de realidad aumentada que dicte una pauta para elaborar aplicaciones basadas en realidad aumentada que cuenten, al menos, con las siguientes características:

- Capacidad de prescindir de marcadores predefinidos.
- Independencia de sensores diferentes a una cámara de video monocular.
- Tolerancia a pérdida parcial de información.
- Tolerancia a cambios ligeros en el nivel de iluminación de la escena.
- Tolerancia al movimiento de la cámara y de los objetos de referencia.
- Desmempeño aceptable en equipos de escritorio de uso común.
- Simulación de la superposición entre objetos virtuales y reales.

## 1.3. Alcances

Este trabajo de tesis buscará la propuesta de un modelo de aplicación de realidad aumentada, el desarrollo de una aplicación como tal o la resolución de un problema mediante el uso de la realidad aumentada se consideran objetivos secundarios y no indispensables.

Así mismo, el alto desempeño de las aplicaciones, el empleo de gráficos tridimensionales complejos y la tolerancia en ambientes sin objetos de referencia se descartan dentro del los alcances del presente trabajo.

También, las aplicaciones elaboradas bajo este modelo, aunque podrían utilizar varios objetos de referencia, sólo podrán utilizar uno a la vez para la realización de los cálculos.

## 1.4. Planteamiento de la tesis

Para lograr los objetivos planteados en el apartado anterior se sugiere un modelo de realidad aumentada basado en dispositivos de video (una cámara web comercial) con las siguientes características:

Para afrontar el problema de la alineación y posicionamiento:

- Utiliza un método de detección y descripción de puntos de interés para caracterizar un con unto de objetos conocidos.
- Los puntos de interés serán procesados por medio de una técnica de clasificación para su posterior reconocimiento en tiempo real.
- Los puntos reconocidos se utilizan para conocer la ubicación de la cámara con respecto a los objetos conocidos, con esto se crea un modelo virtual básico del mundo.
- Los objetos virtuales se colocan en el mundo virtual basándose en el modelo obtenido
- Cada cuadro de video se presentará como una imagen (con segmentos transparentes) a manera de *ventana* hacia el mundo virtual, ver figura 1.5.

Para el problema de seguimiento:

- Realizar seguimiento de los puntos de interés detectados e identificados mediante algún método del estado del arte.
- Calcular los vectores de movimiento de cada punto y compararlos con los anteriores.
- Realizar una prueba de consistencia con respecto al conjunto de puntos detectados, si el resultado no es satisfactorio, realizar nuevamente la detección.

Para resolver las oclusiones entre objetos reales y virtuales:

Como se puede ver en la figura 1.5, la imagen interfiere entre el usuario y el mundo virtual, de esta manera, cuando un objeto virtual deba aparecer sobre uno real un fragmento de la

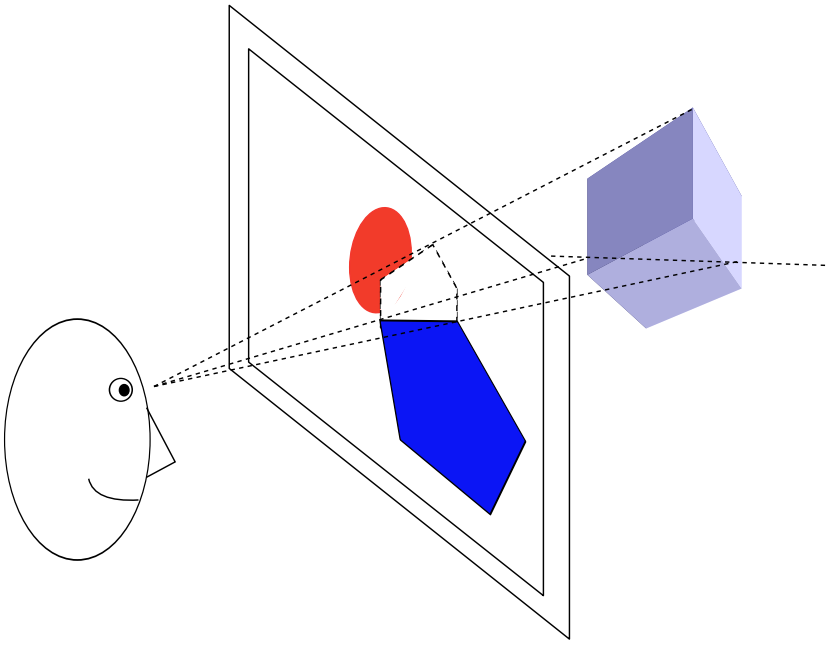


Figura 1.5: Cada fotograma del video, tratado como una imagen, se presentará "por encima" del mundo virtual (y los objetos que contiene); en dicha imagen habrá zonas transparentes para ver ciertos objetos virtuales o parte de ellos

imagen no se dibujará (o se le agregará una transparencia total o parcial) para simular dicho efecto. Este proceso se detalla en el capítulo 5.

La figura 1.6 presenta un diagrama de componentes.

## 1.5. Descripción del contenido

El siguiente capítulo muestra los enfoques hasta ahora utilizados en la resolución de los problemas básicos a resolver en una aplicación de realidad aumentada. En el tercero se detalla el contexto teórico de las herramientas empleadas por el modelo propuesto. Debido a la importancia que tiene la calibración de la cámara en gran cantidad de aplicaciones de visión por computadora, especialmente en las de realidad aumentada; se le dedica el cuarto capítulo a la explicación detallada del método de calibración utilizado. En el quinto capítulo se explica a detalle cada aspecto del modelo, enfocándose en la resolución de los problemas básicos cuyas soluciones existentes se muestran en el capítulo 2, también se describe la estructura de una aplicación de ejemplo. En "Resultados experimentales", capítulo 6, se muestran los resultados de los experimentos realizados a la solución propuesta a cada problema planteado, además de los resultados de desempeño de la aplicación de ejemplo. Finalmente, en el capítulo 7, se dan

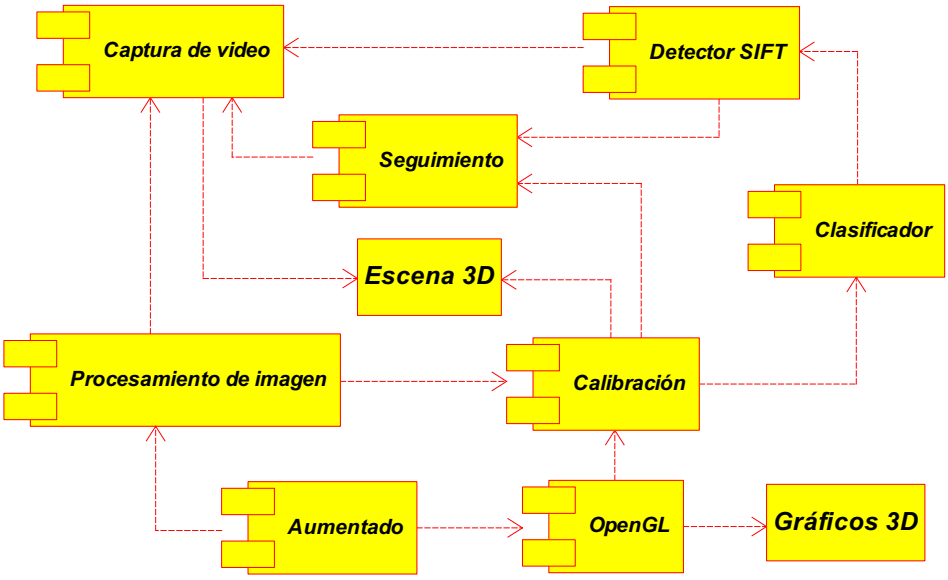



Figura 1.6: Diagrama de componentes del modelo de RA propuesto.

las conclusiones del trabajo realizado, así como también se propone una serie de temas que se considera podrían ser de importancia en el futuro desarrollo del modelo que se propone.

# Capítulo 2

# Propuestas existentes

La realidad aumentada es un área de investigación eminentemente multidisciplinaria, implica la resolución de una multitud de problemas relacionados con el procesamiento de imágenes, visión por computadora, aprendizaje de máquina, gráficos por computadora, entre otros; además de aquellos específicos del área en que se enfoque de cada aplicación en particular.

Si se estableciera un conjunto de tareas a resolver para desarrollar un sistema basado en realidad aumentada, se debería iniciar con la captura de información, esto es, que herramientas se van a utilizar para obtener la información del entorno que nos permitirá colocar la información agregada correctamente; luego, ¿De qué manera se deberá procesar la información obtenida?, es necesario que dicha información permita realizar cálculos sobre la métrica de la escena con los cuales será posible colocar eficientemente los objetos virtuales y lograr algunos efectos que incrementen la sensación de realidad; finalmente, la RA no busca crear aplicaciones estáticas, será necesario mantener la secuencia de ejecución durante el tiempo que el usuario lo requiera, actualizando siempre los datos iniciales y verificando que éstos se mantengan confiables.

En los apartados siguientes se explicarán brevemente los métodos hasta ahora utilizados para la resolución de los problemas más comúnmente enfrentados en el desarrollo de una aplicación basada en RA, estos son: la obtención de información, la calibración automática de la cámara (en el caso de las aplicaciones que la utilizan), la colocación de los objetos virtuales, la superposición entre los mismos y los objetos reales en la escena y el seguimiento de aquellos que sean de interés a la aplicación a través de los cuadros de video.

## 2.1. Captura de datos

El proceso de captura de datos en una aplicación de realidad aumentada depende de las necesidades de la misma y de las herramientas con que se cuente. Se pueden fabricar sistemas

de RA que obtengan información de una o varias cámaras [30; 65] con lentes comunes o convexos [30], de sistemas de posicionamiento (GPS) [66; 77] o de sensores de movimiento [8; 43], entre otras fuentes.

## 2.2. Calibración automática

Aunque la calibración de la cámara es un tema específico de la visión computacional, en el campo de la realidad aumentada (principalmente en la basada en video) juega un papel primordial. Ya que es la cámara la que se utiliza para obtener la mayor parte de la información que se procesa, es de vital importancia poder utilizarla para realizar ciertas tareas que implican comparaciones y transformaciones entre el sistema coordenado del mundo y el sistema coordenado de la imagen.

Además, algunas tareas pueden no permitir que se inicie la operación con una cámara previamente calibrada, como cuando el sistema debe permitir cambios frecuentes en el equipo o cuando se realizarán modificaciones en propiedades como ángulo de visión, apertura del foco, *zoom*; en estos casos, se hace necesario contar con un método que permita realizar el proceso de calibración con la menor asistencia posible por parte del usuario. A esto se le conoce como calibración automática.

En años recientes ha habido un creciente interés por el trabajo enfocado a lograr que la calibración de una cámara se realice de manera automática, a continuación se describen algunos esfuerzos en ese sentido.

ARToolkit ([42], [1]) utiliza un método basado en aprendizaje de máquina para detectar algunos marcadores predeterminados en la escena. Las características geométricas tridimensionales de dichos marcadores se conocen previamente. Una vez que un marcador es detectado, se utiliza la información de sus líneas en la imagen y en el mundo tridimensional para obtener los parámetros de calibración de la cámara y con ellos, se puede hacer que el sistema inserte objetos virtuales en la escena, cuya posición y orientación depende de la de los marcadores. Este mismo procedimiento es seguido por la mayoría de los sistemas de realidad aumentada basados en marcadores.

Szenberg et. al. [76] utilizan las líneas de un campo de futbol para calibrar automáticamente una cámara. Es posible crear un modelo tridimensional del campo y las distancias entre líneas son conocidas en su totalidad. De esta manera, es posible obtener los parámetros de calibración mediante la correspondencia entre las líneas en la imagen de la cancha real y las del modelo creado.

En [22], Deutscher et. al. presentan un algoritmo para aproximar la calibración de una cámara partiendo de una sola imagen de una escena desconocida, suponiendo que la imagen cumple con las condiciones de un mundo *Manhattan*, esto es, que la imagen contiene tres ejes ortogonales dominantes.

Ribeiro, Dihl y Jung [67], propusieron un método de calibración para sistemas de apoyo a conductores. El usuario del sistema debería proporcionar el ancho del carril y, asumiendo que el vehículo se mueve en un tramo recto del camino, el sistema es capaz de detectar las líneas de la carretera y calcular la transformación proyectiva de un segmento rectangular del plano que es el camino y un plano virtual de las mismas medidas. Una vez hecho esto, se pueden obtener los parámetros intrínsecos[1] de la cámara y realizar mediciones útiles para el sistema.

Gordon y Lowe [33], utilizan el detector y descriptor de puntos de interés SIFT para identificar un objeto conocido en una escena. En una primera etapa, previa al ciclo de ejecución principal del programa, se extraen las características (descriptores) de los puntos de una imagen de referencia y de la escena y se forman correspondencias entre ellos. Estas correspondencias y la información tridimensional del objeto se utilizan para crear un modelo métrico del mundo, al mismo tiempo que se obtienen los valores de proyección y posición de la cámara.

Como se habrá podido observar, debe existir cierto grado de conocimiento previo de la escena mediante la cual se desean obtener los parámetros de calibración de la cámara, este puede ser sobre el ambiente en que el sistema operará ([76], [67], [22]) o la existencia de algún objeto conocido el cual se deba de localizar [42], [27], [33].

## 2.3. Colocación de objetos virtuales

Como se habrá podido notar en apartados anteriores, el diseñador de un programa basado en realidad aumentada puede echar mano de tantos medios para obtener de información como sus recursos e ingenio lo permitan. El objetivo principal de este proceso de recopilación de datos es siempre uno: conocer en la mayor medida posible la escena para poder así colocar los objetos virtuales en el lugar que se requieran, al momento que se requieran. De esta manera, en un sistema que opere al aire libre, si se tiene la posición exacta del usuario (mediante GPS, por ejemplo) y se sabe hacia donde está mirando (usando una brújula electrónica), se puede decidir qué elementos se agregarán al mundo real, en qué posición e incluso a qué escala. Entonces, el método que se utilice para la colocación de los objetos virtuales depende en gran

[1] De proyección

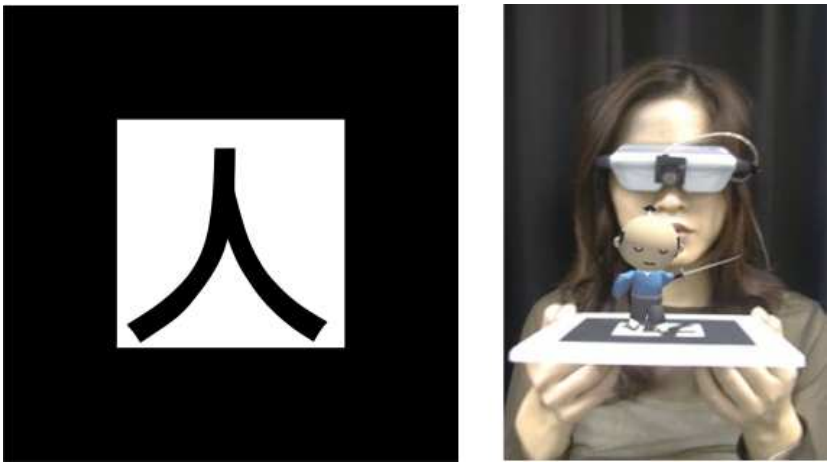


Figura 2.1: A la izquierda, un marcador predefinido. Por su estructura interna se puede observar que, al ser identificado en una imagen, se puede conocer también su orientación. A la derecha, Utilización de ese marcador para ubicar un objeto virtual en un fotograma [42]

medida, si no es que totalmente, de los medios utilizados para recopilar información sobre la escena.

Cuando se trata de aplicaciones de realidad aumentada que utilicen de manera preponderante la información obtenida de las imágenes, la colocación de los objetos virtuales se realiza usando como base principal los objetos y formas que se encuentran en la escena. Para esto, las propuestas existentes se pueden clasificar en dos principales enfoques: usando marcadores predefinidos y sin marcadores.

### 2.3.1. Marcadores predefinidos

Se entiende por un marcador predefinido un objeto plano con una textura sencilla, generalmente monocromática, que será reconocido y ubicado por el sistema y con la información obtenida a partir de uno o varios marcadores se realizará la calibración de la cámara y la colocación de los objetos virtuales en la escena, ver figura 2.1.

A grandes rasgos, el proceso de aumentado involucrando marcadores es como sigue: [1]:

1. Se parte de el conocimiento previo de los parámetros intrínsecos de la cámara
2. Aplicar un umbralado a la imagen original. Debido al diseño y color del marcador, será fácil identificarlo del resto de la imagen.
3. Ejecutar una análisis de componentes conectados para definir la localización y orientación del marcador.

4. Detectar los contornos y esquinas del marcador.

5. Calcular la homografía entre el marcador original (figura 2.1 izquierda) y el capturado por la cámara.

6. Calcular la transformación de la cámara (parámetros extrínsecos)

7. Colocar los objetos virtuales según la ubicación y orientación de la cámara

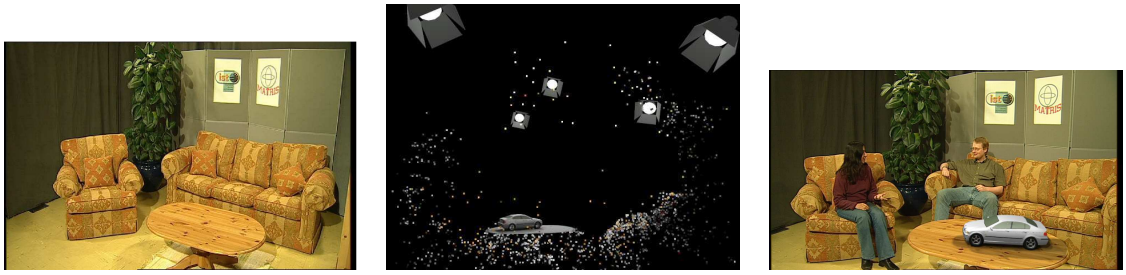
Este método presenta grandes beneficios para ciertas aplicaciones ([40], [77], [10], entre muchos otros), sobre todo por su simplicidad de aplicación, además de la existencia de creciente número de herramientas capaces de manejar este tipo de objetos para facilitar la creación de sistemas de RA (ARToolkit [1], OSGART [20], Designer's ARToolkit[51], por mencionar los más comunes).

Como desventajas de este enfoque, se puede mencionar el hecho de que los marcadores no son parte natural de la escena, deberán ser insertados en ella para poder agregar los objetos virtuales teniendo con esto una influencia negativa en la experiencia del usuario en cuanto a sensación de realidad. Además, debido a que cada marcador otorga determinada información al sistema, estos deberán ser únicos y detectados en su totalidad en la imagen; esto significa que si cualquier objeto real cubre parte de un marcador, éste dejará de ser detectado y por lo tanto los objetos asociados a él desaparecerán del video aumentado. Estas cuestiones representan problemas graves cuando no se puede tener un gran control de la escena o de las acciones del usuario.

### 2.3.2. Realidad aumentada sin marcadores

Cuando se desea prescindir de los marcadores en una aplicación de realidad aumentada, se debe recurrir a técnicas de visión computacional que consumen mayor cantidad de recursos, por eso, en ocasiones resulta necesario realizar un procesamiento de la escena previo a la operación del sistema. Generalmente se parte de algunos elementos conocidos en la escena para realizar la calibración de la cámara y localizar los objetos importantes y luego se procede con la colocación de los objetos virtuales. A continuación se describen algunos métodos de trabajo sin marcadores reportados a la fecha.

Frahm et al. [30] presentan una aplicación para la televisión que permite agregar objetos virtuales iluminados de acuerdo a la iluminación real, incluyendo sombras y reflejos. Cuentan con un escenario preparado con varias cámaras, una de ellas con lente convexo para el monitoreo de las luces, ubicada en el techo de la habitación. Todas estas cámaras están orientadas



(a) Escena original, las texturas del mobiliario son utilizadas para la detección de puntos de interés

(b) La ubicación 3D previamente conocida de los objetos y las fuentes de iluminación permiten hacer un modelo tridimensional de la escena

(c) Escena aumentada

Figura 2.2: Aplicación de realidad aumentada sin marcadores y con cálculo de iluminación [30]

hacia un escenario con objetos conocidos. En una primera etapa de funcionamiento se detectan e identifican los puntos de interés de la escena, se calcula la calibración de las cámaras y se identifican los objetos que contendrán los elementos virtuales. Luego se colocan los objetos en el mundo real y se calcula la iluminación correcta para ellos. Ver figura 2.2.

Yuan [87] propone un enfoque basado únicamente en la detección e identificación de puntos de interés. Se tiene una escena conocida (una imagen tomada previamente) que se utiliza como referencia y una posición en esa escena donde los objetos virtuales deberán aparecer. Con base en los puntos de interés detectados en la imagen de referencia y se realiza un apareo de puntos con los de las capturas hechas durante la ejecución, con esto se conoce la ubicación de los puntos de la primera imagen en la segunda y se puede calcular la transformación geométrica sufrida por los mismos y con ella calibrar la cámara con respecto a los mismos puntos y entonces colocar los objetos virtuales, ver figura 2.3.

Una metodología similar es la seguida por Xu, Chia y Cheok [85], parten de dos imágenes de referencia de una misma escena tomadas desde distintas posiciones y conociendo los parámetros de calibración de la cámara en ambas, además de conocer de antemano el lugar donde los objetos virtuales habrán de aparecer. Cada cuadro de video que se procesa será sometido a una detección de puntos de interés y al apareo de éstos con las imágenes de referencia, para luego calcular el movimiento de la cámara y las transformaciones sufridas por los objetos virtuales, ver figura 2.4.

Chekhlov et al.[19] utilizan un método para descubrir estructuras planas en una escena no preparada y sobre ellas colocan algunos objetos virtuales, también realizan un proceso de detección y descripción de puntos de interés una vez que los planos fueron encontrados, esto

(a) Imagen de referencia (b) Imagen aumentada

Figura 2.3: La posición predefinida del cubo es sobre los números del teléfono y se debe presentar de esa manera independientemente de la posición del mismo [87]

Figura 2.4: Aunque no hay ningún marcador en la escena, el cubo aparece siempre sobre el libro [85]

para posibilitar la recuperación en caso de oclusión o movimiento.

El trabajo de Gordon y Lowe anteriormente referido [33] se aplica a un sistema de realidad aumentada libre de marcadores utilizan el detector y descriptor de puntos de interés SIFT para identificar un objeto conocido en una escena. En una primera etapa, previa al ciclo de ejecución principal del programa, se extraen las características (descriptores) de los puntos de una imagen de referencia y de la escena y se forman correspondencias entre ellos. Estas correspondencias y la información tridimensional del objeto se utilizan para crear un modelo métrico del mundo, al mismo tiempo que se obtienen los valores de proyección y posición de la cámara.

## 2.4. Superposición de objetos

Para ser efectivo, un sistema de realidad aumentada, debe incrementar la información que el usuario percibe del mundo por medio de objetos generados por la computadora. Y para que esa información sea confiable deberá ser colocada en el mundo real de manera convincente.

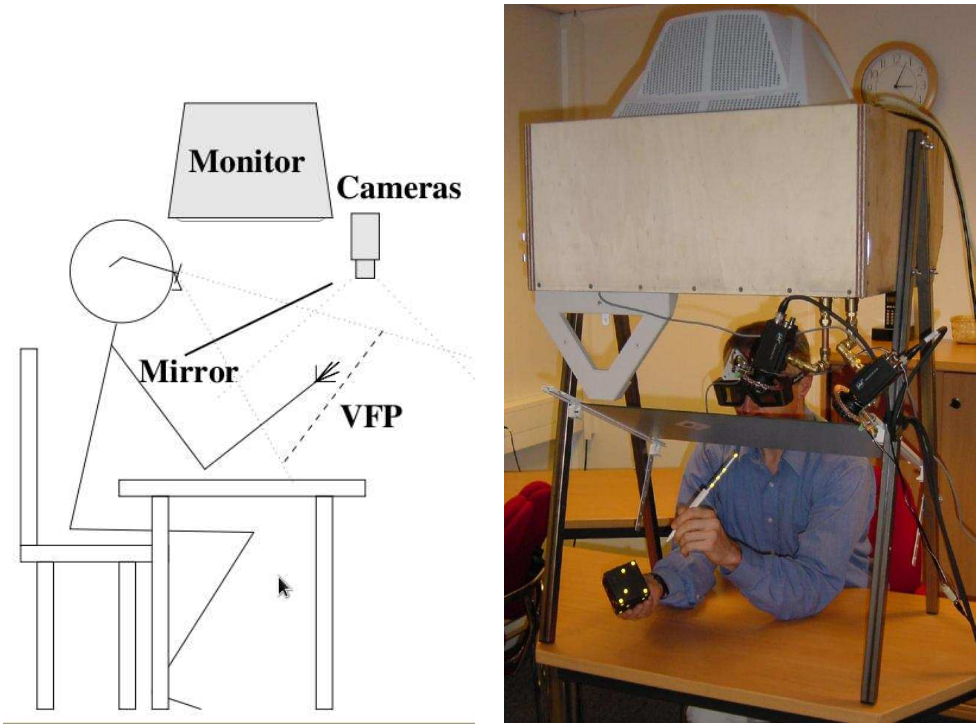
Si la sensación de profundidad y el hecho de poder saber cuando un objeto está detrás de otro proveen al ser humano de una gran cantidad de información visual, un sistema de realidad aumentada debería proporcionar esa información también. En condiciones ideales debería ser posible que los objetos reales y virtuales interactuaran de manera tal que fuera posible dejar de ver total o parcialmente un objeto virtual cuando alguno real obstruyese su vista, al igual que permitir que los objetos virtuales cubriesen a los reales.

Aunque hay una gran cantidad de trabajos de investigación en el área de la realidad aumentada ([6], [7], [69]), son relativamente pocos los que proponen alguna solución al problema de la oclusión de la visibilidad entre objetos reales y virtuales. En los párrafos siguientes se describen brevemente algunas propuestas representativas.

Wloka y Anderson [84] utilizan un sistema con dos cámaras alineadas (estereoscópico), éstas capturan sendas imágenes en escala de grises de la escena y dichas imágenes son procesados de la siguiente manera:

1. Se analizan en franjas verticales en busca de cambios abruptos en el nivel de intensidad de los pixeles.

2. Con el proceso anterior se generan bloques de pixeles con cambios ligeros en la intensidad. Estos bloques se caracterizan por parámetros como su posición $x$, $y$, su longitud, promedio de intensidad y desviación estándar de la misma.




(a) Esquema de vista lateral

(b) Prototipo de la "Estación espacial personal"

Figura 2.5: Aplicación propuesta por Mulder [58]

3. Se comparan los bloques obtenidos en ambas imágenes, se definen correspondencias entre ellos y se calcula la información de profundidad para cada bloque. Al final de esto se tienen dos mapas de profundidad (para la imagen izquierda y para la derecha).

4. Esta información de profundidad se utiliza para crear un modelo del mundo y determinar así que bloques serán cubiertos por objetos virtuales.

Mulder [58] presenta un sistema de RA basado en espejos semitransparentes que presenta la capacidad de generar el efecto de superposición, ver figura 2.5. Utilizando, además de la construcción básica mostrada en la figura 2.5(a), una pantalla de cristal líquido (LCD) para bloquear la luz y material polarizable por la luz[2] logran filtrar la luz del monitor para evitar que se muestren determinados pixeles en el espejo, esto es, el espejo sólo reflejaría aquellos objetos que no sean cubiertos por un objeto real. El cálculo de las oclusiones se basa en la profundidad conocida de los objetos virtuales y en el cálculo de profundidad de los objetos reales por medio de imágenes obtenidas desde varios ángulos con distintas cámaras.

[2] $\mu$Pol comercializado por vRex http://www.vrex.com/

Berger, en [9], propone un método de resolución de oclusiones basado mayormente en el procesamiento de la imagen, logrando prescindir de la necesidad de tener información a priori de la posición tridimensional de los objetos en la escena. El proceso seguido por este método se puede resumir en cuatro pasos:

1. Se calcula una *máscara* inicial $m_1$ que es la región de la imagen donde el objeto virtual aparecería de no ser cubierto por ningún objeto real, también se calculan los contornos que causarían oclusión de la vista del objeto (un rectángulo blanco), figura 2.6(b).

2. Se extraen las cadenas de contornos en la región de la imagen correspondiente a $m_1$, figura 2.6(c), se consideran todas las cadenas que crucen la máscara inicial. Estas cadenas se rastrean en la imagen siguiente 2.6(d) usando un sistema de seguimiento basado en curvas. Finalmente se calculan las correspondencias entre los contornos de ambas imágenes por medio de geometría epipolar.

3. Con base en la comparación de el desplazamiento 2D asociado al punto real en la escena que se proyecta en $m$ con el desplazamiento asociado al punto 3D que pertenece al objeto virtual que se proyecta en $m$, se calcula una etiqueta *delante*, *detrás* o *dudoso* para cada punto en $m$, dependiendo se este debe aparecer enfrente o detrás del objeto virtual o si no se puede determinar su ubicación. Ver figura 2.6(e), 2.6(f).

4. Se calcula la nueva máscara de oclusión que será aplicada a los objetos virtuales, esto con base en el etiquetado de los contornos previamente hecho y en *snakes*[3] generados con base en los mismos contornos. Figuras 2.6(g), (h), (i) y (j).

## 2.5. Seguimiento de los objetos de referencia

El seguimiento de objetos[4] es una tarea importante en el campo de la visión por computadora. Su objetivo es, dada la descripción de un objeto en una imagen $I_t$ capturada en el tiempo $t$ de una secuencia, encontrar dicho objeto en una ubicación cercana en una segunda imagen $I_{t+1}$ y calcular el movimiento y, en el caso de algunos algoritmos, deformación que el objeto sufrió de una imagen a otra. En otras palabras, el seguimiento puede ser definido como el problema de calcular la trayectoria de un objeto (mediante su detección en cada cuadro) en el plano de la imagen según este se mueve en la escena.

En la literatura se reportan varios métodos de seguimiento de objetos, estos se pueden agrupar en tres enfoques principales [86]:

[3]Contorno activo: plataforma para delinear un objeto en una imagen.
http://en.wikipedia.org/wiki/Active_contour

[4]En inglés *Tracking*

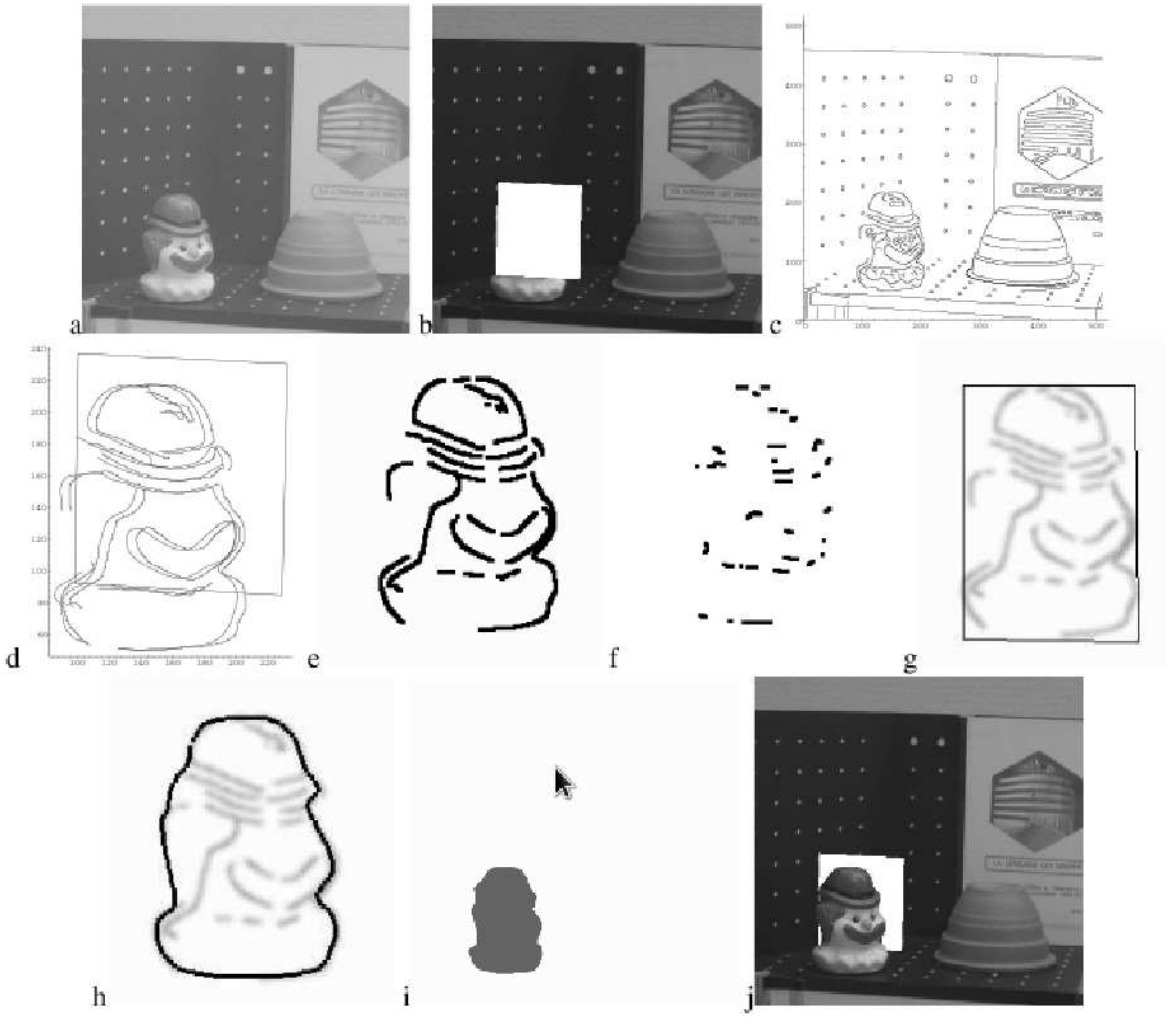



Figura 2.6: (a) Imagen original. (b) La máscara del objeto virtual que será mezclado con la imagen. (c) El mapa de contornos. (d) Resultado del proceso de seguimiento. (e) Contornos etiquetados como *delante*. (f) puntos no clasificados. (g) Campo de gradiente creado por los contornos y un *snake*. (h) La máscara de oclusión después del procesamiento con el *snake*. (i) La máscara de oclusión final. (j) El resultado

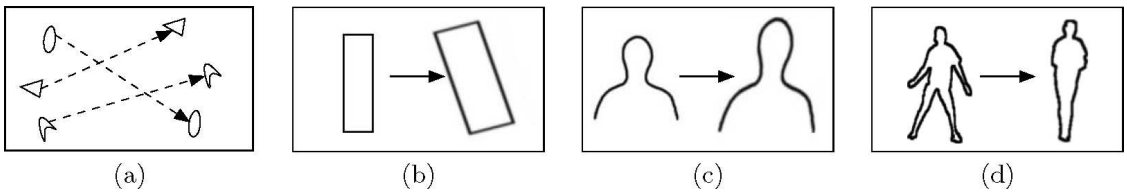



Figura 2.7: Diferentes enfoques de seguimiento. (a)Correspondencia por puntos. (b) Transformación paramétrica de un segmento rectangular. (c) y (d) Seguimiento por siluetas (evolución de contornos)

- Seguimiento por puntos, ver figura 2.7a.
- Seguimiento por núcleos, ver figura 2.7b.
- Seguimiento por siluetas, ver figura 2.7c y d.

### 2.5.1. Seguimiento de puntos

En este enfoque el seguimiento se formula como la correspondencia entre objetos detectados entre fotogramas diferentes, estos objetos se representan por puntos. El problema consiste, entonces, en identificar determinado conjunto de puntos correspondientes a un objeto, de una imagen $I_t$ a una imagen $I_{t+1}$ y calcular el desplazamiento entre ambas. Algunos ejemplos de métodos que siguen este enfoque pueden ser: filtros de Kalman [15], filtros de partículas, Seguimiento por hipótesis múltiples [75].

### 2.5.2. Seguimiento por núcleos

En este enfoque los objetos se representan por medio de una región o primitiva[5] del objeto. El movimiento del objeto se da, generalmente, en forma de movimiento paramétrico (traslación, conforme, afín, etc.). KLT [73], seguimiento con SVM [5], seguimiento por apariencia [38], son algunos ejemplos de este enfoque.

### 2.5.3. Seguimiento por siluetas

Algunos objetos pueden no ser fácilmente representables con puntos o figuras sencillas. Para seguir este tipo de objetos en una secuencia de imágenes, se puede utilizar un método basado en siluetas. Estos métodos se basan en la representación de la región que ocupa el objeto por medio de un modelo, que puede ser un histograma de color, el contorno de objeto u otra información que lo caracterice. El seguimiento se puede realizar por coincidencia de formas o por evolución de contornos. En el primer caso se busca encontrar al objeto en cada

[5]Elemento geométrico sencilla (elipse, polígono, línea, etc)

fotograma de la secuencia. En el segundo, se deforma (evoluciona) el contorno de la imagen previa con base en la optimización de alguna función de energía. Ver figura 2.7c y d. Algunos métodos que utilizan este enfoque son el usado por Haritaoglu para el seguimiento de personas [34], Kang, Cohen y Medioni [41] utilizan histogramas de color e información de los bordes para modelar los objetos; Sato [71] proponen la generación de rastros de un objeto por medio de la transformada de Hough en el espacio de la velocidad entre imágenes consecutivas.

# Capítulo 3

# Marco teórico

Este capítulo trata los fundamentos teóricos empleados en la elaboración de la propuesta de modelo detallada en el capítulo 5, con un enfoque central en aquellos temas involucrados en la resolución de los problemas básicos antes mencionados: calibración automática de la cámara, colocación de objetos virtuales, simulación de la superposición entre elementos reales y virtuales y seguimiento de características de referencia.

Cabe hacer notar que, aunque el método de calibración utilizado una vez que se tienen las correspondencias entre puntos del mundo y puntos en la imagen podría ser incluido en este capítulo, con el doble objetivo de darle la relevancia debida a dicho método y mantener el equilibrio (cuantitativo y cualitativo) entre las partes del documento, se ha decidido dedicar un capítulo completo a la explicación de la geometría de la cámara y el proceso de calibración. Por lo tanto, en los próximos apartados se verán los aspectos necesarios para obtener las correspondencias entre los sistemas de coordenadas del mundo y de la imagen, la deducción de las transformaciones necesarias para mapear un punto de uno a otro se verá en el capítulo siguiente.

## 3.1. Detección y descripción de puntos de interés

Históricamente, la mayoría de los sistemas de visión por computadora incluían un proceso de segmentación de la imagen en una etapa de bajo o mediano nivel [52; 24]. La segmentación es un proceso mediante el cual se intenta agrupar pixeles u otros rasgos de bajo nivel de una imagen para formar regiones conectadas y homogéneas, utilizando una medida de similitud y un análisis global de la imagen. Con base en este proceso, se espera que cada región cubra objetos reales o conceptos reales, por ejemplo una casa, el cielo, una pieza de herramienta, etc. De esta forma el análisis de alto nivel se podría simplificar gracias a la división semántica de las regiones.

Sin embargo, segmentar una imagen es un problema mal planteado para el caso general [45; 79], razón por la cual esta tarea ha resultado ser difícil en la práctica. Por lo tanto, desde la década de los noventa muchos de los sistemas de visión se han diseñado utilizando enfoques que no requieran una segmentación previa para resolver tareas como la detección y reconocimiento de objetos, recuperación de imágenes basada en contenido, entre otras.

Entonces, para algunas aplicaciones puede resultar más práctica la utilización de un análisis local de la imagen, esto mediante la detección estable y la descripción representativa de regiones locales en la imagen. Este tipo de rasgos son muy pequeños en relación con la imagen y se les llama regiones de interés porque transmiten información que se considera *visualmente interesante*.

Las regiones de interés contienen pixeles que exhiben algunas propiedades distintivas que las hacen adecuadas para aplicaciones en donde sea necesario identificar ciertas características en una secuencia de imágenes o en imágenes tomadas bajo distintas condiciones. Ahora bien, la estabilidad de un detector se puede medir con base en diferentes tipos de transformaciones: cambios de iluminación, proyección, escala o rotación. Entonces, la forma de las regiones de interés dependen del tipo de invarianza requerido, por ejemplo, cuando un detector sólo detecta pixeles, o puntos de interés, entonces sólo se espera que sea invariante a transformaciones de iluminación y rotación. Cuando un detector también es invariante a escala, entonces es capaz de detectar regiones de interés isotrópicas.

Una medida de cuan interesante es un pixel o región se puede extraer empleando un operador $K = \mathbb{R}^+ \rightarrow \mathbb{R}$. Cabe hacer la distinción entre un operador y un detector: el primero sólo se utiliza para calcular la medida de lo *interesante* que es cada pixel, mientras que el detector es el proceso algorítmico empleado para identificar todos los pixeles o regiones interesantes dentro de una imagen. De tal forma que diferentes detectores emplearán operadores $K$ diferentes durante la detección. Cuando se aplica $K$ a una imagen se obtiene lo que se conoce como una imagen de interés $I^*$. Después, la mayoría de los algoritmos emplean un proceso similar que consiste en:

1. Supresión de no máximos.

2. Umbralado para identificar los picos en la respuesta al operador $K$.

### 3.1.1. Métodos de detección de puntos de interés

Las técnicas para la detección de puntos de interés se desarrollaron como resultado del trabajo realizado para resolver el problema de la detección de esquinas [57; 2; 35; 72; 63]. La clase de los detectores de esquinas que operan directamente sobre los valores de intensidad en

una imagen [35; 72] son conocidos como detectores de puntos de interés, la diferencia puede parecer sutil, pero es importante conceptualmente; las esquinas son rasgos que se encuentran en uniones entre líneas y superficies, por otro lado, el concepto de punto de interés puede o no incluir estos rasgos además de otros que carecen de una interpretación semántica estricta o clara.

Algunos detectores basan la medida de interés que arroja el operador $K$ en la matriz de autocorrelación local $A$ que caracteriza la distribución del gradiente dentro del vecindario de cada pixel, dicha matriz está dada por:

$$A(\mathrm{x}, \sigma_I, \sigma_D) = \sigma^2 \cdot G_{\sigma_I} * \begin{bmatrix} L_x^2(\mathrm{x}, \sigma_D) & L_x(\mathrm{x}, \sigma_D) L_y(\mathrm{x}, \sigma_D) \\ L_x(\mathrm{x}, \sigma_D) L_y(\mathrm{x}, \sigma_D) & L_y^2(\mathrm{x}, \sigma_D) \end{bmatrix} \tag{3.1}$$

donde $\sigma_D$ y $\sigma_I$ son las escalas de derivación en integración, respectivamente, $L_u(\mathrm{x}, \sigma_D)$ es la derivada gaussiana en dirección $u$ de la imagen $I$ en el punto x dado por:

$$L_u(\mathrm{x}, \sigma_D) = \frac{\delta}{\delta u} G_{\sigma_D} * I(\mathrm{x}), \tag{3.2}$$

siendo $G_\sigma$ una función gaussiana de suavizado con desviación estándar $\sigma$. Los detectores que utilizan $A$ incluyen a los propuestos por Förstner y Gülch [29], Harris y Stephens [35], Shi y Tomasi [73], con los operadores de cada uno definidos de la siguiente manera:

$$K_{Forstner}(\mathrm{x}) = \frac{det(A)}{Tr(a)}, K_{Harris}(\mathrm{x}) = det(A) - k \cdot Tr(a)^2, K_{Shi\&Tomasi}(\mathrm{x}) = min\{\lambda_1, \lambda_2\},$$

donde $\lambda_1$ y $\lambda_2$ son los valores propios de $A$; la definición de $A$ se tomó de [72], en donde se aplica con el detector de *Harris mejorado*, descrito en ese mismo trabajo.

### El detector de puntos de SIFT

En [48] se presenta, junto con un descriptor, un algoritmo de detección de puntos de interés basado en el método de diferencia de gaussianas (DoG)[1] que se explica a continuación:

Se desea identificar ciertas posiciones en el espacio escalar de la imagen que sean invariantes a traslaciones, cambios de escala y rotaciones, así como mínimamente afectados por ruido y distorsiones ligeras. Lindeberg [47] ha mostrado que bajo algunas presuposiciones generales sobre la invarianza a la escala, el núcleo gaussiano y sus derivados son los únicos núcleos de suavizado posibles para el análisis de espacios escalares.

Para lograr la invarianza a la rotación y un alto nivel de eficiencia se seleccionan las posiciones correspondientes a los máximos y mínimos de una diferencia de gaussianas, definiendo la

[1] Acrónimo el inglés *Difference of Gaussians*

función gaussiana como en la ecuación 3.3. Esto se puede lograr de una manera eficiente construyendo una pirámide de imágenes con muestreo a cada nivel. Además, se localizan puntos de interés en regiones y escalas con alta variación, haciendo estas posiciones particularmente estables para la caracterización de las imágenes.

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x^2+y^2)/2\sigma^2} \tag{3.3}$$

Para la localización de puntos de interés, todas las operaciones de suavizado se realizan usando $\sigma = \sqrt{2}$, lo cual se puede aproximar con suficiente precisión usando un núcleo unidimensional con siete puntos de muestra.

La imagen de entrada se convuelve primero con la función gaussiana usando $\sigma = \sqrt{2}$ para obtener una imagen $A$. A esta se le aplica un nuevo suavizado cd $\sigma = \sqrt{2}$ para obtener una imagen $B$, la cual tiene un suavizado efectivo e $\sigma = 2$. La diferencia de gaussianas se obtiene de sustraer la imagen $B$ a la imagen $A$. De manera que:

$$\begin{aligned} rclD(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma) \otimes I(x, y)) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \\ L(x, y, \sigma) &= G(x, y, \sigma) \otimes I(x, y) \end{aligned} \tag{3.4}$$

donde $D$ es la diferencia de gaussianas, $I$ es la función que representa una imagen en niveles de gris, $\otimes$ representa la operación de convolución y $k$ un factor multiplicativo, en este caso $\sqrt{2}$.

Para generar el siguiente nivel de la pirámide se hace un muestreo de la imagen $B$ usando interpolación bilineal con un espaciado de 1.5 pixeles en cada dirección. Esto significa que cada nueva muestra será una combinación lineal constante de cuatro pixeles adyacentes. Y así sucesivamente para todos los niveles que se requieran.

Los máximos y mínimos de este espacio escalar se determinan comparando cada pixel en la pirámide con sus vecinos. Primero, un pixel se compara con sus ocho vecinos en el mismo nivel, si es máximo o mínimo en ese nivel se calcula la posición más cercana en el siguiente nivel, tomando en cuenta el muestreo de 1.5 veces. Si el pixel sigue siendo mayor (o menor) que sus ocho vecinos se continúa la prueba con el siguiente nivel. Ya que la mayoría de los pixeles se eliminarán en las primeras comparaciones, el costo de detección es bajo y mucho menor que el de la construcción de la pirámide.

Figura 3.1: De izquierda a derecha: imagen original ($I$), primer suavizado gaussiano ($A = G(I)$), segundo suavizado gaussiano $B = G(A)$, diferencia de gaussianas $D = A - B$

**Cálculo de la orientación**

También se puede asignar una orientación a cada punto detectado basándose en las propiedades locales de la imagen, el descriptor del punto podría ser representado de acuerdo a su orientación y así se obtendría invarianza a la rotación.

Según [49], el enfoque que proporciona los resultados más estables en el cálculo de la orientación consiste en calcular la magnitud del gradiente $m(x, y)$ y la orientación $\theta(x, y)$ usando diferencias entre pixeles:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \theta(x,y) &= tan^{-}1((L(x+1,y) - L(x-1,y))/(L(x,y+1) - L(x,y-1))) \end{aligned}$$

Con ésto se forma un histograma de orientación que consta de 36 secciones, cubriendo el rango de $360^o$ de las orientaciones. Cada muestra que se agrega al histograma se mide con respecto a la magnitud de su gradiente y por una ventana gaussiana circular con una $\sigma$ que es 1.5 veces la $\sigma$ de la escala del punto.

Los picos en el histograma de orientación corresponden a las orientaciones dominantes de gradientes locales. El pico más alto del histograma será el detectado, así como cualquier otro pico con al menos el 80 % de la altura de éste. De esta manera se pueden obtener múltiples puntos de interés de una misma posición y por lo tanto múltiples descriptores de un mismo punto, lo que contribuye en gran medida a la estabilidad de la localización y posterior identificación.

### 3.1.2. Métodos de descripción de puntos de interés

Una vez que se tiene localizado un conjunto de regiones que aportan información distintiva acerca de una imagen, puede resultar útil tener la capacidad de distinguir entre cada una de las

regiones detectadas o incluso definir si una región detectada en una imagen corresponde a otra detectada en una segunda imagen. Para esto, es necesario diseñar un método que cuantifique ciertas características distintivas de una región en particular, es decir, que describa a una región en términos de un conjunto específico de características.

A la fecha se han desarrollado muchas técnicas de descripción de regiones locales en imágenes. El más sencillo podría ser un vector de los pixeles de la imagen, después se puede realizar una correlación cruzada para comparar dos descriptores. Aunque la alta dimensionalidad del descriptor lo hace inviable para el reconocimiento. Los métodos de descripción de puntos de interés reportados se pueden agrupar de la siguiente manera [78]:

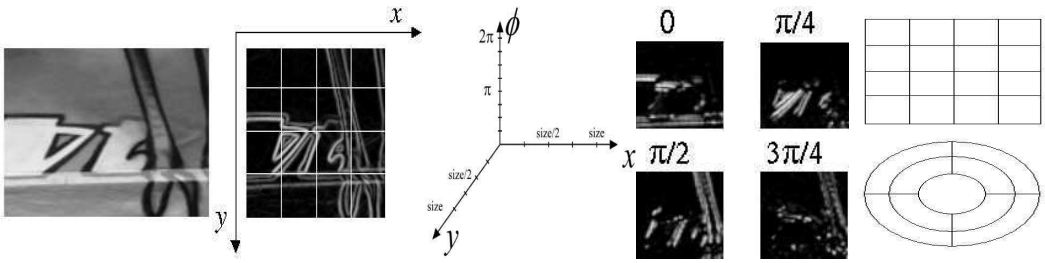
**Técnicas de espacio-frecuencia** Muchas técnicas describen el contenido de frecuencia de una imagen. La transformada de Fourier descompone el contenido de la imagen en funciones base. Sin embargo, en esta representación, las relaciones espaciales entre puntos no son explícitas y las funciones base son infinitas, además de ser difícil de adaptar a un enfoque local. La transformada de Gabor [32] supera estos problemas, pero se requiere un gran número de filtros de Gabor para capturar cambios pequeños en frecuencia y orientación. Los filtros de Gabor y las ondeletas[2][82] son utilizadas con frecuencia en aplicaciones de clasificación de texturas.

**Descriptores diferenciales** Un conjunto de derivadas de imágenes calculadas hasta un determinado orden aproxima un vecindario de puntos. Las propiedades de las derivadas locales (*jet local*) fueron investigadas por Koenderink [44]. Florack et al. [28] derivaron invariantes diferenciales, las cuales combinan componentes del *jet local* para obtener invarianza a las rotaciones. Freeman y Adelson [31] desarrollaron filtros dirigibles, los cuales dirigen derivadas en una dirección particular dados los componentes del *jet local*. Dirigir las derivadas en la dirección del gradiente hace que éstas sean invariantes a la rotación. Una estimación estable de las derivadas se obtiene mediante la convolución de la imagen con derivadas gaussianas.

**Descriptores basados en distribución** Estas técnicas utilizan histogramas para representar diferentes características de apariencia y forma. Un descriptor muy simple sería la distribución de intensidades de los pixeles representada por un histograma. Una representación más expresiva fue presentada por Johnson y Hebert [39] para el reconocimiento de objetos tridimensionales en imágenes de rango[3]. Su representación es un histograma de las posiciones relativas de un punto de interés tridimensional en el vecindario. Este

[2]En inglés *wavelets*

[3]Imágenes que contienen información de profundidad en vez de niveles de intensidad




(a) (b) (c) (d) (e)

Figura 3.2: Descriptor SIFT. (a)Región detectada. (b). Imagen gradiente y rejilla de localización. (c)Dimensiones del histograma. (d). 4 de 8 planos de orientación. (e) Rejillas de localización cartesiana y polar. La rejilla polar muestra 9 secciones de localización usadas en el contexto de la forma (4 en dirección angular)

descriptor ha sido ya adaptado a las imágenes [46]. Las dos dimensiones del histograma son la distancia al punto central y el valor de intensidad.

**Otras técnicas** Los momentos invariantes generalizados fueron presentados por Van Gool et al. [80] para describir la naturaleza multi-espectral de los datos en la imagen. Estos invariantes combinan los momentos centrales definidos por $M^1_{pq} = \iint_\omega x^p y^q \left[I(x,y)\right]^a$ con orden $p+q$ y grado $a$. Los momentos caracterizan la distribución de forma e intensidad de la región $\omega$, son independientes y se pueden calcular fácilmente para cualquier orden y grado. Sin embargo, los momentos de alto orden y grado son sensibles a pequeños cambios geométricos y distorsiones fotométricas [55]. Estos descriptores son más adecuados para imágenes a color donde se puedan calcular los invariantes para cada canal y entre canales.

### El descriptor SIFT

Lowe propone en [48] una transformación de rasgos invariantes a escala (SIFT, acrónimo de *Scale Invariant Feature Transform*), que combina un detector de regiones invariante a escala y un descriptor basado en la distribución del gradiente en las regiones detectadas. El descriptor se representa por medio de un histograma tridimensional de orientaciones y localizaciones del gradiente, ver figura 3.2. La contribución a las secciones de localización y orientación se mide por la magnitud del gradiente. La cuantificación de las posiciones y orientaciones del gradiente hace al descriptor robusto a pequeñas distorsiones geométricas y pequeños errores en la detección de regiones.

Se usa una función gaussiana con desviación estándar $\sigma$ igual a la mitad del ancho de la ventana del descriptor para asignar un peso a la magnitud de cada punto muestreado, en

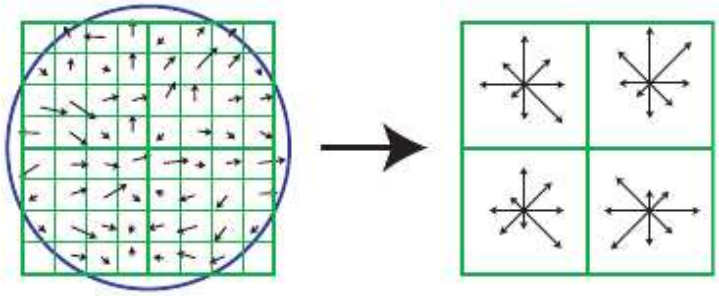


Figura 3.3: Para crear un descriptor primero se calculan la magnitud y orientación del gradiente en cada punto muestreado (izquierda). Estos son procesados con una ventana gaussiana (indicada por el círculo). Estas muestras se acumulan en histogramas de orientación conjuntando los contenidos sobre subregiones de 4x4 (derecha), la longitud de cada flecha representa la suma de las magnitudes de los gradientes cercanos a esa dirección dentro de la región. En la figura se muestra un descriptor de 2x2 calculado de un conjunto de 8x8 muestras

la figura 3.3 (izquierda) se ilustra con una ventana circular, aunque, por supuesto, el peso disminuye suavemente. El propósito de esta ventana gaussiana es evitar cambios repentinos en el descriptor con cambios pequeños en la posición de la ventana y dar menos énfasis a los gradientes que están lejos del centro del descriptor [49].

El descriptor de puntos se muestra en el lado derecho de la figura 3.3. Éste permite cambios significativos en las posiciones del gradiente creando histogramas de orientación sobre regiones muestra de 4x4. La figura muestra ocho direcciones para cada histograma de orientación, la longitud de la flecha corresponde a la magnitud en el histograma. Una muestra de gradiente a la izquierda puede cambiar hasta cuatro posiciones de muestra y seguir contribuyendo al mismo histograma en la derecha, permitiendo así el objetivo de tolerar grandes cambios de posición.

Cada descriptor consiste en un vector que contiene los valores de todas las entradas al histograma de orientación. en la figura 3.3 se muestra un arreglo de histogramas de orientación de 2x2, según [49] los mejores resultados se obtienen con arreglos de histogramas de 4x4, con 8 espacios de orientación en cada uno. Obteniendo así un vector de $4 \times 4 \times 8 = 128$ elementos para describir cada punto de interés.

Finalmente, el vector de características se modifica para reducir los efectos de los cambios de iluminación. Primero se normaliza el vector a un valor constante, ya que un cambio en el contraste de la imagen multiplicaría los valores de cada pixel por un valor constante, al igual que los gradientes, por lo tanto, el efecto de esta multiplicación se anula al realizar la normalización del vector. Un cambio de brillo, en el cual se suma una constante a la intensidad

de cada pixel no afectaría los valores de los gradientes, ya que se componen de diferencias entre pixeles. Además, el descriptor es invariante a transformaciones afines [49].

## 3.2. Clasificación y reconocimiento

Una vez que se tiene una manera de detectar y describir consistentemente ciertos puntos en una secuencia de imágenes, resulta necesario saber, además de la posición y descripción de un determinado punto de interés, a cuál punto detectado en una imagen $A$ corresponde el detectado en otra imagen $B$. O más aún, puede resultar necesario, habiendo definido cierto grupo de puntos, saber si éstos se encuentran o no entre los puntos que se han detectado en determinada imagen.

El problema que ocupa a este trabajo es el segundo, se tiene un cierto conjunto de puntos previamente definidos (aunque no necesariamente se tienen sus descriptores) en un objeto real conocido y se desea saber si estos puntos aparecen o no en un grupo de imágenes de las que se han detectado y descrito sus puntos de interés. Este problema puede ser planteado, desde el enfoque del aprendizaje de máquina [56], como un problema de clasificación supervisada donde se tienen $k + 1$ clases, donde $k$ es el número de puntos que se busca identificar en las imágenes y la clase restante sería la no pertenencia a cualquiera de las $k$ clases, es decir, el punto detectado en la imagen no se puede identificar como ninguno de los puntos que se espera encontrar. Cabe recordar que el objetivo del presente trabajo es detectar e identificar un conjunto de puntos en una imagen para después utilizar la ubicación de tales puntos en un proceso de calibración relacionándolos con un grupo de puntos tridimensionales, previamente conocido, en la escena.

Aunque se reportan algunos trabajos en el área de reconocimiento de objetos por medio de los descriptores SIFT, es decir, caracterizar un objeto por una cierta cantidad de descriptores de puntos encontrados en las imágenes obtenidas de él; luego encontrar dicho objeto en otras imágenes ([49; 60; 59; 54], entre otros); no se tiene conocimiento de algún intento de clasificación de los puntos individualmente.

A continuación se muestran algunos métodos de clasificación que podrían resultar útiles en el problema particular de la identificación de puntos de interés, más propiamente de sus descriptores, detectados por medio de SIFT.

### 3.2.1. Clasificador Bayes normal

El clasificador Bayes normal o ingenuo[4] basa su funcionamiento en el teorema de Bayes, expresado en la ecuación 3.5, añadiéndole una regla de decisión dada en la ecuación 3.6. La idea básica del algoritmo es: conociendo un conjunto de datos previamente clasificados, calcular la probabilidad de que cada clase se presente, luego calcular la probabilidad de que cada rasgo se presente en cada clase. Con esto, cuando se tiene un objeto no clasificado, se evalúa la probabilidad de pertenencia a una clase con base en los rasgos que lo caracterizan y en la probabilidad de que los mismos pertenezcan a cada una de las clases. Finalmente se elige a la clase con mayor probabilidad de pertenencia.

$$p(C_j|X) = p(C_j)\frac{p(X|C_j)}{p(X)} \tag{3.5}$$

donde $p(C_j|X)$ se entiende como la probabilidad de que un objeto, cuyo vector de características es $X = \{x_1 \dots x_l\}$, pertenezca a la clase $C_j$, entonces $p(C_j)$ es la cantidad de objetos conocidos en la clase $C_j$ dividido por la cantidad total de objetos en la muestra de entrenamiento, siendo $j = \{1 \dots k\}$ y $k$ el número de clases. De la misma manera, $p(X|C_j)$ es la probabilidad de que un vector de características $X$ pertenezca a la clase $C_j$, eso es:

$$p(X|C_j) = \prod_{i=1}^{l} p(x_i|C_j)$$

debido a que $p(X)$ es constante para todas las clases para un mismo $X$, se puede tomar como un factor de escala y prescindir de ese factor para efectos de clasificación. Esto gracias a la regla de decisión:

$$C = \arg\max_j \left(p(C_j|X)\right) \tag{3.6}$$

Como se podrá observar, esté método tiene una aplicación directa a problemas cuyas variables $x_i$ sean categóricas o cualitativas, es decir, que presenten un rango pequeño de valores posibles ("si" y "no", "alto", "bajo" y "mediano", listados de colores, etc.). Pero se presentan algunos problemas al pretender aplicarlo a problemas con valores continuos en sus rasgos. En éste último caso, una solución puede ser conocer la función de densidad de probabilidad (FDP) para los datos de muestra o bien asumir que se representa por alguna función conocida, por ejemplo:

[4]En inglés *Naïve Bayes*

Gaussiana o normal:

$$p(x_i|C_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{\left(\frac{-(x_i-\mu_{ij})}{2\sigma_{ij}}\right)}$$

donde $\mu_{ij}$ es la media de los valores de $x_i$ presentes en la clase $j$ en la muestra de entrenamiento y $\sigma_{ij}$ la desviación estándar de los mismos.

Gamma:

$$p(x_i|C_j) = \frac{\left(\frac{x}{b_y}\right)^{c_{ij}-1}}{b_{ij}\Gamma(c_{ij})} e^{\frac{-x}{b_{ij}}},\ b_{ij}, c_{ij} > 0$$

siendo $b_{ij}$ el parámetro de escala y $c_{ij}$ el parámetro de forma.

Poisson:

$$p(x_i|C_j) = \frac{\lambda_{kj} e^{-\lambda_{kj}}}{x!},\ \lambda_{kj}, x \in > \mathbb{N}$$

$\lambda_{ij}$ es la media del rasgo $i$ en la clase $j$.

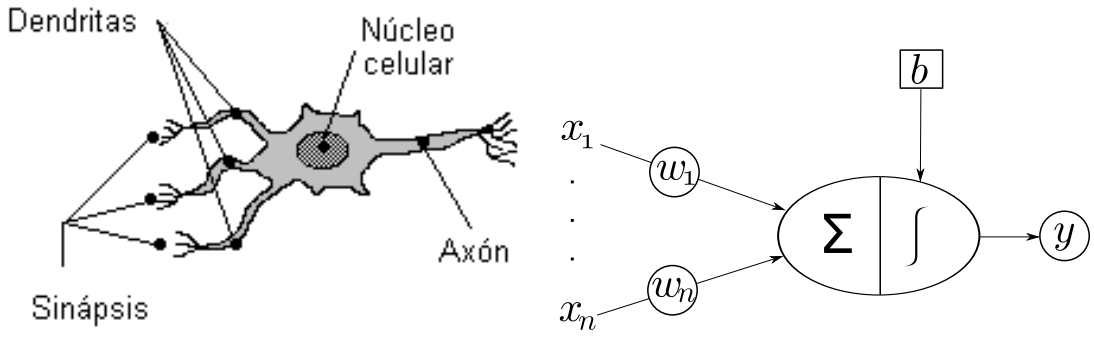
Se han desarrollado otros métodos como por ejemplo, la estimación de la FDP por medio de núcleos gaussianos [12], que mejoran la clasificación en varios problemas ya que permiten un mejor modelado del problema y, por lo tanto, mejores resultados en la predicción.

### 3.2.2. Redes neuronales artificiales

Como su nombre sugiere. este tipo de clasificadores basan su funcionamiento en la estructura de las redes de células cerebrales (neuronas) de los animales [70]. Una neurona animal consta de un conjunto de entradas (dendritas) por medio de las cuales recibe impulsos eléctricos que procesa en su núcleo y, según el resultado de ese procesamiento, emite una carga eléctrica a través de su salida (axón) como resultado del procesamiento, ver figura 3.4a. El axón de cada neurona se conecta a las dendritas de otras neuronas (sinapsis), de esta manera se puede realizar tareas complicadas por medio de la operación de varios elementos sencillos.

En el caso de una neurona artificial, o perceptrón [68], ésta cuenta con un conjunto de entradas $x_i$, a las cuales se asigna determinado peso $w$. Estas entradas son parámetros de una función de activación $\Sigma(\mathbf{x}) : \mathbb{R}$, cuyo resultado, generalmente entre 0 y 1 o -1 y 1, es multiplicado por un factor de tendencia $b \in (0, 1)$. Ver figura 3.4b.

Una red neuronal artificial es un conjunto de neuronas artificiales conectadas entre si. Esta red puede incluir varias capas de estás, mínimamente una capa de entrada y una de salida. Las capas entre los elementos de entrada y los de salida se conocen como capas ocultas. Ver figura 3.5.




(a) (b)

Figura 3.4: Representación de una neurona (a) biológica, (b) artificial

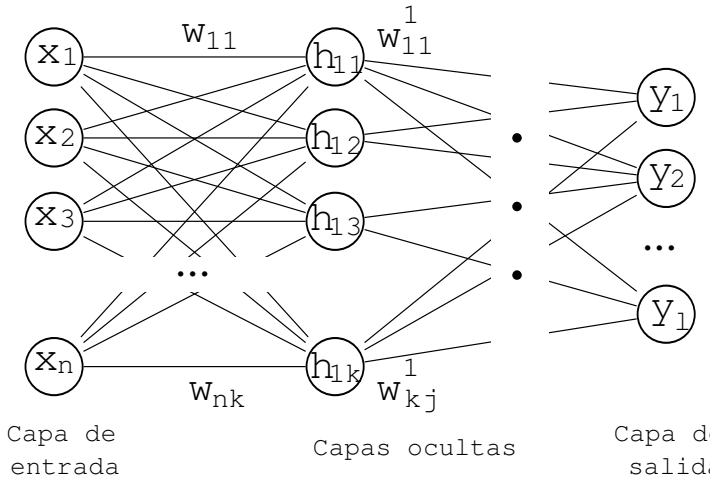
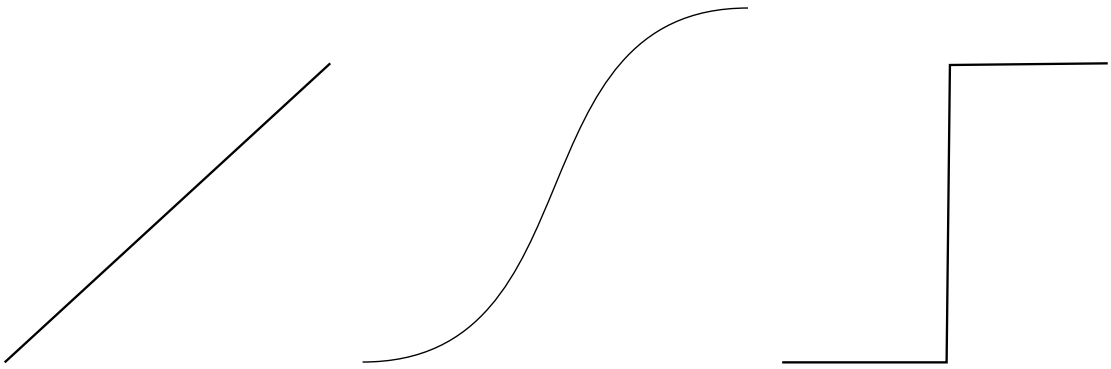



Figura 3.5: Una red neuronal artificial con $n$ elementos de entrada y $l$ elementos de salida



(a) $f(a) = \lambda a$ (b) $f(a) = \tanh(a),\ f(a) \in (-1, 1)$ (c) $f(a) = \{ {}^{1}_{-1} \quad {}^{a>0}_{a\leq 0}$

Figura 3.6: Funciones de activación: (a)Lineal. (b)Sigmoide. (c)Umbral

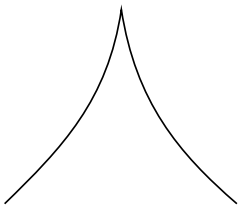


Figura 3.7: Gráfica de la función de activación Elliott

Las funciones de activación para cada neurona se eligen dependiendo del problema de clasificación que se trate, algunos ejemplos de funciones se pueden ver en la figura 3.6.

En [23], Elliott propone una función de activación que, según experimentación reportada más adelante en este trabajo permite buenos resultados en la clasificación de descriptores SIFT, ver figura 3.7:

$$f(a) = \frac{a}{1 + |a|} \tag{3.7}$$

### 3.2.3. Máquinas de soporte vectorial

Si se tiene un conjunto de objetos, representados por vectores de características $\mathbf{x} = \{x_1 ... x_n\}$ con dimensionalidad $n$ y divididos en dos clases: $C_+$ y $C_-$; se dice que las clases son linealmente separables si y solo si existe un hiperplano $H$ que las separe, ver figura 3.8.

El objetivo de una máquina de soporte vectorial (SVM[5]), es maximizar el margen de separación entre dos clases linealmente separables, esto por medio del cálculo de un hiperplano $H$ que permita tal característica [16].

[5]Por el inglés: *Support Vector Machine*

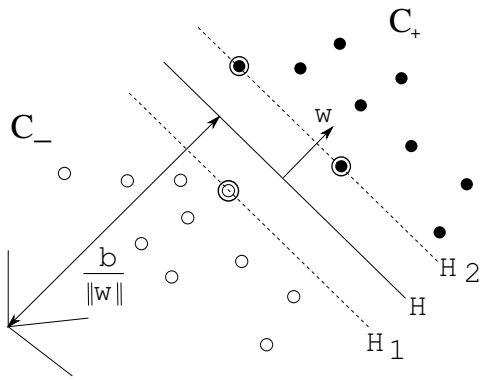



Figura 3.8: Las clases $C_+$ y $C_-$ son linealmente separables por el hiperplano $H$

Supóngase que se asigna un valor escalar $y_i$ de 1 a cada objeto $\mathbf{x}_i$ perteneciente a la clase $C_+$ y $y_j = -1$ para los objetos en $C_-$. Entonces, todo objeto miembro de la clase $C_+$ deberá cumplir con la restricción:

$$\mathbf{w} \cdot \mathbf{x} + b \geq 1$$

y para la clase $C_-$ se deberá cumplir que:

$$\mathbf{w} \cdot \mathbf{x} + b \leq -1$$

lo cual se puede combinar en un conjunto de inecuaciones:

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i\right) + b \geq 1 \forall i \tag{3.8}$$

donde $\mathbf{w}$ es un vector normal al hiperplano de separación $H$ cuya dirección indica la ubicación de la clase $C_+$ y $b$ es un escalar mediante el cual se puede conocer la distancia perpendicular de $H$ al origen.

Los vectores que yacen en los hiperplanos $H_1 : \mathbf{w} \cdot \mathbf{x} + b = 1$ y $H_2 : \mathbf{w} \cdot \mathbf{x} + b = -1$, marcados en la figura 3.8 con un círculo externo, cuya distancia $H$ es mínima con respecto al resto de los miembros de la misma clase; son llamados *vectores soporte*. La distancia entre los hiperplanos $H_1$ y $H_2$ que contienen a los vectores soporte se denomina margen $m = \frac{2}{||w||}$.

Entonces, dado un conjunto de $l$ vectores de entrenamiento $\mathbf{x}_i, i = 1 \ldots l$ con sus respectivos valores de clasificación $y_i$ el hiperplano $H$ que proporciona la separación óptima entre clases se puede obtener, primero, encontrando el conjunto de vectores soporte y luego mediante la minimización de $||\mathbf{w}||^2$ sujeto al conjunto de restricciones expresado en la ecuación 3.8. Expresado más formalmente:

$$max\left(||\mathbf{w}||^2\right)$$

Sujeto a: (3.9)

$$y_i\left(\mathbf{w}\cdot\mathbf{x}_i\right)+b\geq 1\forall i=1\ldots l$$

Lo cual se puede expresar, en términos de los multiplicadores de Lagrange como:

$$L_P=\frac{1}{2}||\mathbf{w}||^2-\sum_{i=1}^{l}\alpha_i y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)+\sum_{i=1}^{l}\alpha_i \tag{3.10}$$

Ahora se debe minimizar $L_P$ con respecto a $b$ y $\mathbf{w}$ y buscar que las derivadas de $L_P$ con respecto a todos los $\alpha_i$ se aproximen a 0.

**Clases no separables linealmente**

La idea central en la clasificación de problemas no lineales con SVM se basa en trasladar los elementos de un problema de dimensionalidad $d$ a un espacio euclidiano $\mathcal{H}$ de dimensionalidad superior (incluso infinita) donde la función de decisión[6] sea una función lineal de los datos. Este mapeo $\phi$ se expresa:

$$\phi:\mathbb{R}^n\rightarrow\mathcal{H}$$

Con esto, el algoritmo de entrenamiento sólo dependerá de los datos, a través de productos punto en $\mathcal{H}$, esto es, a través de funciones de la forma $\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. Es posible evitar la necesidad de tratar con estas funciones si se tiene una función núcleo $K(\mathbf{x}_i,\mathbf{x}_j)=\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. Algunas de estas funciones pueden ser:

Polinomial de grado $p$:

$$K(\mathbf{x},\mathbf{y})=(\mathbf{x}\cdot\mathbf{y})^p$$

Función de base radial gaussiana:

$$K(\mathbf{x},\mathbf{y})=e^{-||\mathbf{x}-\mathbf{y}||^2/2\sigma^2}$$

Sigmoidal:

$$K(\mathbf{x},\mathbf{y})=\tanh\left(\kappa\mathbf{x}\cdot\mathbf{y}-\delta\right)$$

[6]Entiéndase por función de decisión la función cuyo signo representa la clase a la que un vector $\mathbf{x}$ pertenece

Se recomienda la lectura de [16] para mayor comprensión del funcionamiento y aspectos formales de las SVM.

**Problemas con más de dos clases**

Hasta ahora se ha mostrado la aplicación de las máquinas de soporte vectorial en problemas con sólo dos clases, pero, ¿Qué sucede cuando se tiene un problema de clasificación que involucre tres o más clases? Al respecto se han propuesto algunas alternativas [11], las más representativas se explican enseguida.

Una solución inmediata es la estrategia *uno contra todos* llamada también WTA_SVM[7], ésta consiste en tener un conjunto de $k$ clasificadores, siendo $k$ el número de clases, de manera que exista una máquina $S_j$ capaz de distinguir elementos de la clase $j$ de los del resto de las clases, para $j = 1 \dots k$; asignándole a los correspondientes a $j$ la clasificación positiva. Cuando un objeto $\mathbf{x}$ no clasificado se presenta es sometido a todos los clasificadores y se asigna a la clase cuyo clasificador haya arrojado un valor positivo.

Otra opción es contar con un clasificador binario para cada posible par de clases en el problema, de esta manera un objeto no clasificado se somete a todos los clasificadores y aquella clase con mayor número de *victorias* o clasificaciones positivas es a la que se asigna dicho objeto. Este método se conoce como *max-wins voting* (MWV_SVM).

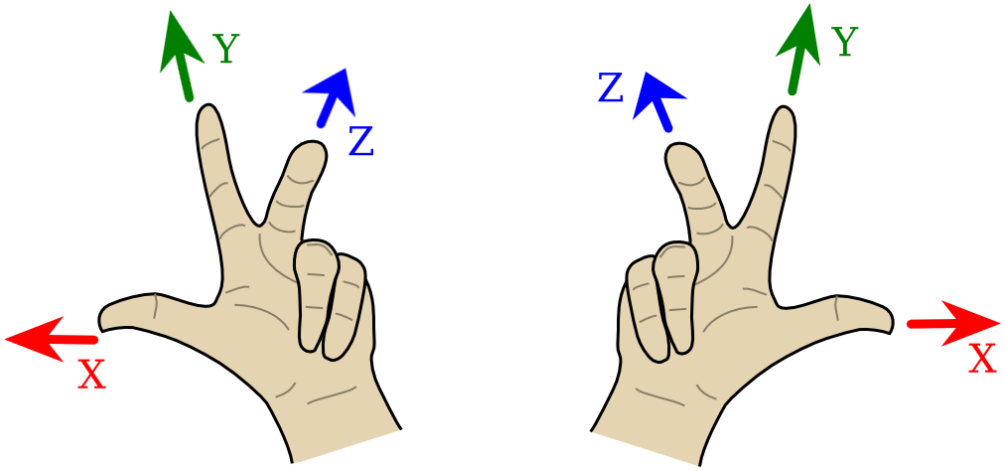
Una variante de MWV_SVM es PWC_SVM[8] que utiliza las salidas de cada clasificador binario para calcular la probabilidad de que un objeto nuevo pertenezca a cada clase y finalmente lo asigna a la clase con mayor probabilidad.

## 3.3. El mundo de OpenGL

OpenGL[9] es una especificación estándar que define una interfaz de programación multilenguaje y multiplataforma para el desarrollo de aplicaciones que presenten gráficos en dos y tres dimensiones. Teniendo como objetivos principales dos aspectos:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.

[7] Del inglés: *Winner Takes All*, el ganador se lleva todo
[8] Del inglés: *Pairwise coupling*
[9] Open Graphics Library




(a) Regla de la mano izquierda (b) Regla de la mano derecha

Figura 3.9: Un sistema de coordenadas se denomina de mano izquierda o de mano derecha según la orientación de sus ejes

- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

El mundo virtual en el que OpenGL permite la creación de gráficos tridimensionales tiene un sistema de coordenadas fijo basado en la regla de la mano derecha, ver figura 3.9, y con una cámara u observador virtual ubicada de forma predefinida en el origen y mirando hacia el lado negativo del eje $Z$ y con el lado positivo del eje $Y$ apuntando hacia arriba, ver figura 3.10. La posición y orientación de la cámara se puede modificar por medio de transformaciones rígidas (rotaciones y traslaciones), la manera como se presentan los objetos en pantalla, por medio de transformaciones proyectivas y escalamientos.

Además, OpenGL permite la utilización de sistemas de coordenadas locales para los objetos, esto es, habiendo $n$ objetos en una escena se tienen $n+2$ sistemas de coordenadas, uno por cada objeto y además el de la cámara u observador y el del mundo. De estos, se pueden realizar transformaciones en todos ellos excepto en el del mundo.

### 3.3.1. Transformaciones en OpenGL

Antes de ser presentados en pantalla y con este fin, los objetos son sometidos a cuatro transformaciones [3]:

**Transformación de modelado** Esta mueve los objetos a través de la escena y transforma de coordenadas locales (del objeto) a coordenadas globales (del mundo).

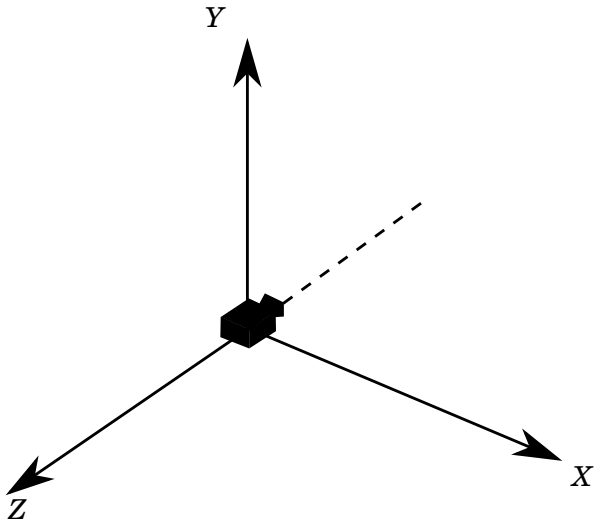



Figura 3.10: El sistema de coordenadas de OpenGL y la ubicación inicial del observador

**Transformación de vista** Especifica la localización de la cámara y transforma coordenadas globales a coordenadas de la cámara.

**Transformación de proyección** Define el volumen de visión y los planos de corte, ver figura 3.11, también mapea los objetos del sistema coordenado de la cámara a coordenadas en el plano.

**Transformación de la ventana** Realiza un mapeo de las coordenadas del plano de corte a coordenadas bidimensionales en la ventana de visualización.

OpenGL maneja todas estas transformaciones mediante operaciones con matrices, estás operaciones se pueden realizar directamente sobre la matriz predefinida que afecta a cada transformación o por medio de instrucciones especiales para cada operación. OpenGL combina las transformaciones de vista y modelado en un sólo grupo de transformaciones representado por la matriz de modelado-vista (*modelview*) y la de proyección a través de la matriz del mismo nombre.

En cuanto a las transformaciones de vista, son las que involucran el movimiento de la cámara, estas pueden ser de rotación y de traslación y se manejan, en C y C++, con la funciones `glRotate()` y `glTranslate()` indicando como parámetros, en la rotación: el ángulo de rotación y el vector (con coordenadas $x$, $y$ y $z$) que sirve de eje para la misma; y en la traslación el desplazamiento sobre los ejes $x$, $y$ y $z$. También se puede modificar la orientación de la cámara llamando a la función `glLookAt()` que se usa para especificar posición y la línea de visión de la cámara, proporcionando como parámetros las coordenadas $x$, $y$ y $z$ (con

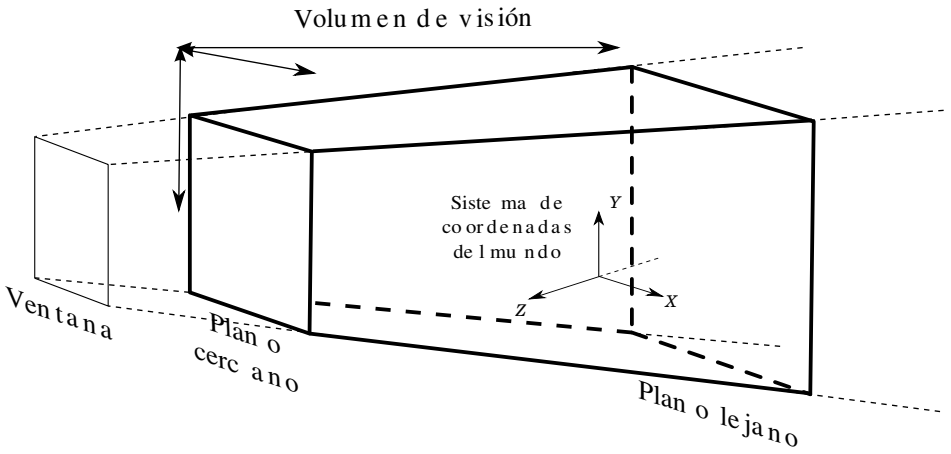



Figura 3.11: El volumen de visión es el segmento del mundo que será visible a través de la ventana

referencia al mundo) de la cámara, el punto (también con tres coordenadas en referencia al mundo) con el que formará la línea de visión y finalmente el vector que indica la dirección que será *arriba* para la cámara.

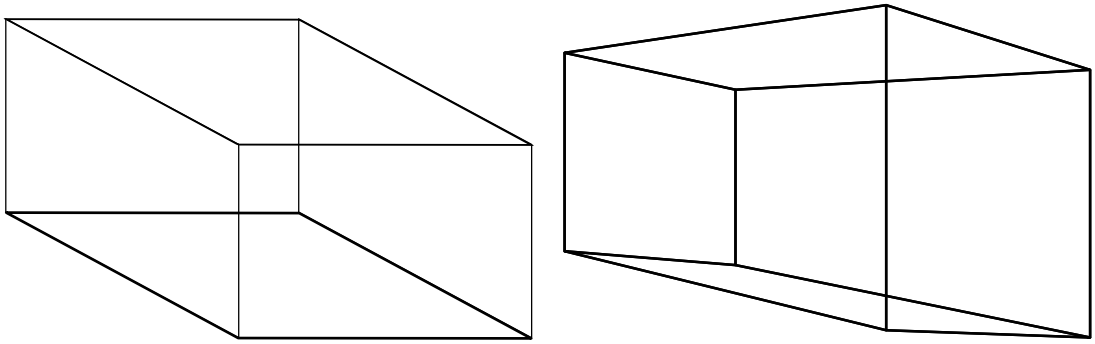
Las transformaciones de modelado son rotación, traslación y escalamiento. La rotación y traslación se pueden manejar de la misma manera que se explicó antes. El escalamiento se controla a través de la función `glScale()` que toma como parámetros los factores de escala a aplicar por cada eje.

La transformación de proyección define el volumen de visión y los planos de corte (figura 3.11). Mediante esta transformación se identifican los puntos que pertenecen al volumen de visión y se calcula su proyección sobre el plano cercano. OpenGL permite dos tipos de proyección, ver figura 3.12:

**Proyección perspectiva** Este tipo de proyección es el más parecido a la forma como son percibidos el mundo, por ejemplo en este tipo de proyección los objetos más lejanos aparecen más pequeños en la pantalla.

**Proyección ortográfica** Los objetos se muestran siempre del mismo tamaño, independientemente de su distancia a la cámara.

Tanto la forma en que los objetos se proyectan como la configuración del volumen de visión se pueden configurar por medio de funciones OpenGL:



(a) Proyección ortográfica (b) Proyección perspectiva

Figura 3.12: Los dos tipos de proyección en OpenGL

Para definir los aspectos propios del volumen de visión, en una proyección ortográfica, se puede utilizar:

```
void glOrtho(izq, der, abajo, arriba, cerca, lejos);
```

`izq` y `arriba` indican la esquina superior izquierda (en coordenadas de OpenGL) del plano cercano, `der` y `abajo` las coordenadas de la esquina inferior derecha, `cerca` y `lejos` indican las coordenadas z del plano cercano y lejano respectivamente.

Para la proyección perspectiva se utiliza `glFrustum()`, con parámetros idénticos que para `glOrtho`.

Los parámetros de la ventana de visión se establecen por medio de `glViewport`, indicando en los parámetros las coordenadas $x$ e $y$ de la esquina inferior izquierda y el tamaño horizontal y vertical de la misma.

### 3.3.2. Matrices y pila de matrices

Como ya se mencionó, OpenGL realiza todas las transformaciones a través de operaciones con matrices. Para esto cuenta con un sistema de pilas de matrices que permiten que se puedan realizar operaciones sobre los distintos objetos de una escena sin afectar al resto. La pila de matrices se controla por medio de las funciones `glLoadIdentity()`, `glLoadMatrix()`, `glPushMatrix()`, `glPushMatrix()` y `glPopMatrix()`. Por ejemplo, en el siguiente fragmento de código se realiza primero una traslación de la cámara, luego se coloca un objeto rotado sobre su propio sistema de coordenadas y otro objeto escalado en un factor de 1.5 en el eje $X$ (se omiten los acentos por cuestiones de compatibilidad con el conjunto de caracteres):

```
//Se elige la proyeccion perspectiva
glMatrixMode(GL_PROJECTION);
```

```
//Se carga la matriz identidad sobre la de modelado-vista
//que esta en la cima de la pila al inicio
glLoadIdentity();

//Se mueve la camara sobre el eje Z
//(la f de glTranslatef indica parametros tipo float)
glTranslatef(0,0,3.5);

//Se carga una nueva matriz a la pila
glPushMatrix();

//Se define una rotacion de 45 grados sobre el eje Y
glRotatef(45,0,1,0);

//contiene las instrucciones de dibujado de un objeto
objeto1();

//Retirar la matriz de la pila
glPopMatrix();

//Se carga una nueva matriz a la pila
glPushMatrix();

//Se realiza el escalamiento y se dibuja el objeto
glScalef(1.5,1,1);
objeto1();

//Retirar la matriz de la pila
glPopMatrix();
```

También es posible generar las transformaciones modificando directamente la matriz indicada, esto se puede realizar seleccionando la matriz mediante `glMatrixMode()` incluyendo como parámetro `GL_PROJECTION` para la matriz de proyección y `GL_MODELVIEW` para la de modelado-vista. Una vez seleccionada la matriz se le debe agregar los nuevos valores mediante `glLoadMatrix()`, que recibe como parámetro un arreglo de tamaño 16 que contiene los nuevos valores que deberá tener la matriz, organizados por columnas.

# Capítulo 4

# Geometría y calibración de la cámara

En una aplicación de realidad aumentada basada en video, la calibración de la cámara es una tarea esencial, tanto para las etapas posteriores del funcionamiento como para el resultado final que se mostrará en pantalla. Por medio de los parámetros de la cámara es que se pueden realizar los cálculos necesarios para la ubicación del usuario y las referencias en el mundo, la localización de aspectos interesantes en la escena y la colocación de los objetos en el mundo. Es decir, el correcto funcionamiento del proceso de aumentado, cuando se basa en la información obtenida del video, depende en primer lugar de un resultado preciso en la calibración de la cámara.

En el capítulo anterior se trataron los métodos de detección e identificación de puntos de interés en una imagen. Si se logra obtener la posición tridimensional en la escena de estos puntos, entonces se tendrá un conjunto de correspondencias 2D - 3D que pueden ser utilizadas para obtener los parámetros intrínsecos y extrínsecos de la cámara.

En el presente capítulo se describirá el modelo geométrico de una cámara y el proceso que se sigue para relacionar los sistemas de coordenadas involucrados en la adquisición de una imagen, asumiendo que se cuenta con un conjunto de puntos 3D donde al menos uno no es coplanar al resto y sus correspondientes proyecciones en la imagen.

Para la formación de una imagen intervienen tres sistemas de coordenadas, todos en el espacio euclidiano:

- El sistema coordenado del mundo (SCM), basado en unidades métricas (usualmente mm) y expresado como $X, Y, Z$.

- El sistema coordenado de la cámara (SCC), basado en unidades métricas (mm) y expresado como $x, y, z$.
- El sistema coordenado de la imagen (SCI), basado en pixeles[1] y expresado como $u, v$.

En el proceso de pasar de coordenadas definidas en el SCM a coordenadas en el SCI expresadas en pixeles, se deben realizar tres transformaciones:

1. *Transformación rígida mundo-cámara.* Los puntos tridimensionales expresados en el sistema coordenado del mundo experimentan un cambio al pasar al sistema coordenado de la cámara. Este cambio de sistema coordenado comprende una transformación rígida compuesta por una rotación, integrada a su vez por tres rotaciones, sobre los ejes $X$, $Y$ y $Z$ y una traslación, igualmente dividida en un valor de traslación por cada eje. Estos parámetros son la posición y orientación que tiene la cámara con respecto a la escena.

2. *Transformación proyectiva cámara-imagen.* Una vez que se ha realizado el cambio del SCM al SCC, los puntos tridimensionales expresados en el SCC son proyectados sobre el plano de la imagen. Las nuevas coordenadas reciben el nombre de coordenadas normalizadas.

3. *Transformación afín cámara-imagen.* Finalmente, se necesita obtener las coordenadas en pixeles correspondientes al SCI. Para ello, las coordenadas normalizadas se someten a una transformación afín en el plano.

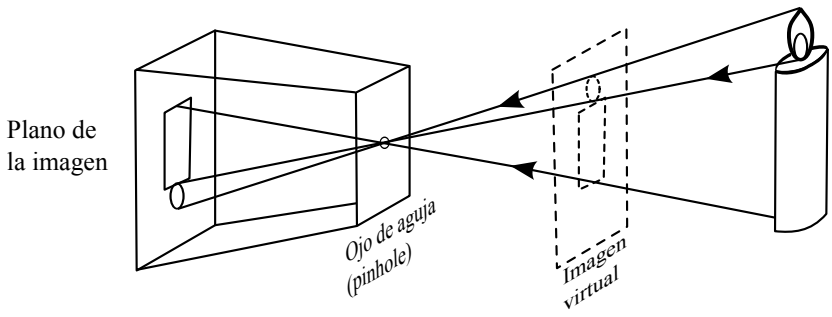



Figura 4.1: *Modelo de cámara oscura.*

Una cámara es una herramienta que proyecta objetos 3D (de la escena) en el espacio 2D (imagen de los objetos). Para representar geométricamente tal proyección se usa el modelo de

[1]Del inglés *pixel*: *picture element*

cámara oscura[2] (ver figura 4.1); en este modelo la imagen resulta de la proyección de todos los puntos de la escena sobre el plano de la imagen a través de un solo punto llamado centro de proyección, centro focal o simplemente foco[3]. En la figura 4.1 el plano de la imagen se encuentra detrás del centro de proyección, por tanto, la imagen es invertida; sin embargo, es posible modificar la geometría para que el centro de proyección corresponda a un punto de visión colocado detrás del plano de la imagen, ver figura 4.2.

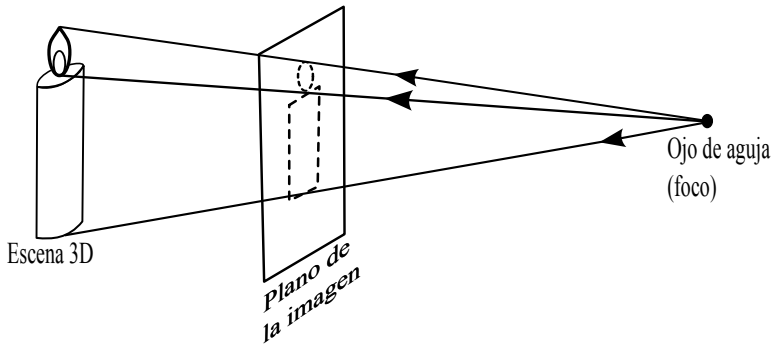



Figura 4.2: Modelo de *cámara oscura* con el foco detrás de la imagen.

## 4.1. Geometría del modelo de cámara oscura

Como ya se mencionó anteriormente, a la cámara se encuentra ligado un sistema coordenado (SCC) con origen que denominaremos $F$. $(F, x)$ y $(F, y)$, ver figura 4.3, serán los ejes paralelos al plano de la imagen correspondientes a la dirección de las filas y las columnas de los pixeles de la imagen. El eje $(F, z)$, llamado eje óptico, está orientado hacia la escena y es perpendicular al plano de la imagen.

El modelo de cámara oscura está basado en que un punto en el espacio con coordenadas $B = (X, Y, Z)$ es mapeado al punto $b = (u, v)$, siendo $b$ la intersección del plano de la imagen con la línea que une al punto $B$ y al centro de proyección $F$. Este modelo indica que cada punto en la escena pasa a través del centro óptico para proyectarse en el plano de la imagen. Por lo tanto, existen dos transformaciones principales que son llevadas a cabo en el proceso de formación de la imagen. La primera es la transformación del sistema coordenado del mundo al sistema coordenado de la cámara $M \rightarrow C$ y la segunda, compuesta a su vez de dos transformaciones (afín y proyectiva), es la transformación del sistema coordenado de la cámara al sistema coordenado de la imagen $C \rightarrow I$. Mediante estas dos transformaciones se obtiene finalmente la imagen.

[2] Conocido en inglés como *Pinhole camera model*
[3] *Pinhole*

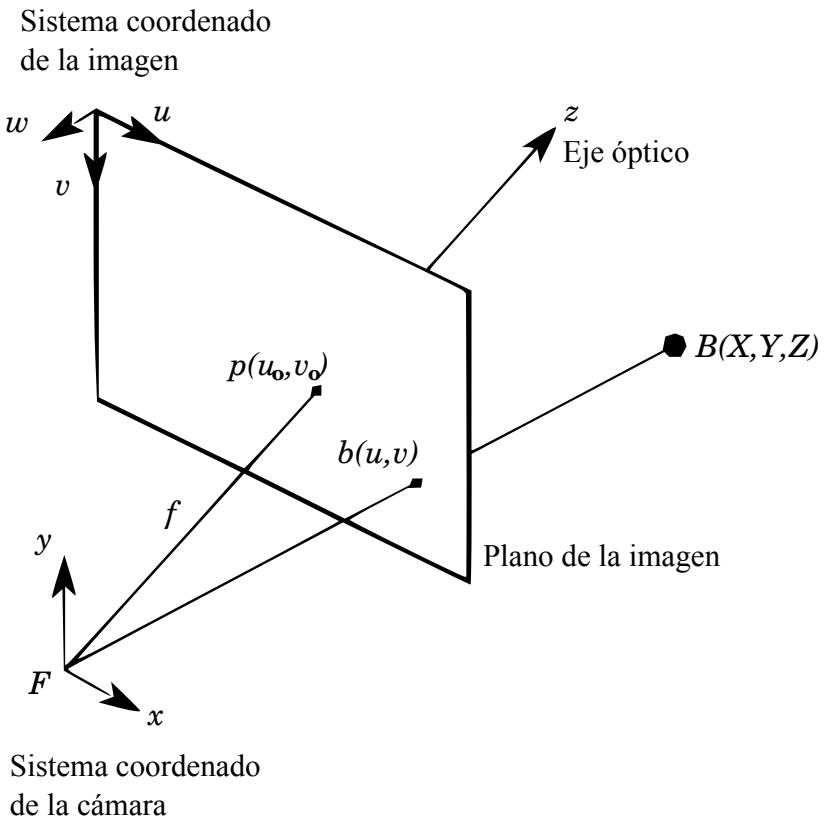



Figura 4.3: *Modelo geométrico de una cámara.* La figura muestra la geometría de un modelo de cámara oscura. $F$ es el centro de proyección, centro focal o foco y $p$ el punto principal. El plano de la imagen está situado al frente del centro de la cámara.

En la figura 4.3 se observa que $F$ es proyectado sobre el plano de la imagen en $p$, llamado punto principal, la distancia entre $p$ y $F$ se conoce como distancia focal y es representada por $f$. El punto $B$, con coordenadas $(X, Y, Z)$ se proyecta largo de una recta que pasa por $b$ y $F$. Al elegir un sistema coordenado fijo en la cámara, el plano $x$-$y$ de este sistema coordenado es paralelo al plano de la imagen, y el eje $z$ se convierte en el eje óptico. El punto $b$ tiene coordenadas $(u, v)$ en el sistema coordenado de la imagen, las coordenadas $(x', y', z')$ del mismo punto basadas en el SCC están dadas por las siguientes ecuaciones:

$$x' = \frac{fX}{z}, \qquad y' = \frac{fY}{z}, \qquad z' = f \tag{4.1}$$

En forma matricial y utilizando coordenadas homogéneas tenemos que si $B = (X, Y, Z, 1)$, su proyección en la imagen $b = (x', y', z')$, según el SCC, están dada por las siguientes ecuaciones:

$$\begin{bmatrix} sx' \\ sy' \\ sz' \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix}}_{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{4.2}$$

y

$$x' = \frac{X}{s}, \qquad y' = \frac{Y}{s}, \qquad z' = \frac{Z}{s}, \qquad s = \frac{Z}{f} \tag{4.3}$$

donde, $P$ representa la matriz de proyección, la cual nos sirve de apoyo para calcular las coordenadas cartesianas del punto. $s$ representa un factor de escala y $f$ la distancia focal.

## 4.2. Transformación del sistema coordenado de la cámara al sistema coordenado de la imagen

Los puntos de la imagen son medidos en pixeles en un sistema coordenado bidimensional $u$-$v$ asociado a la imagen, ver figura 4.3. Con el propósito de poder escribir la matriz de transformación del sistema coordenado de la cámara al sistema coordenado de la imagen, debemos introducir los siguientes parámetros: $u_0$, $v_0$ y $w_0$, que son las coordenadas del centro de proyección $F$ en el sistema coordenado de la imagen, $k_u$ es el factor de escala horizontal y $k_v$ el factor de escala vertical. Las unidades de ambos factores de escala están dadas en pixeles/mm. Se manejan dos factores de escala debido a que los pixeles de una cámara rara vez son cuadrados. La transformación del sistema coordenado de la cámara al sistema coordenado de la imagen se escribe de la siguiente manera para el punto $b$:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix}, \tag{4.4}$$

donde $u, v, w$ son las coordenadas del punto $b$ en el sistema coordenado de la imagen. La transformación que se presenta es una transformación afín que representa un cambio de escala, una rotación y una traslación. Cabe observar que los valores de $y'$ y $z'$ se multiplican por $-1$ debido a que los ejes que se corresponden, $v$ con $y$ y $w$ con $z$, están en sentidos opuestos, ver figura 4.3. La componente $w$ es nula ya que el sistema coordenado de la imagen sólo maneja dos dimensiones, por lo tanto, se ignora el tercer el renglón y se escribe la transformación como una matriz $K$ de tamaño $3 \times 4$. La transformación que representa el cambio de coordenadas

del espacio proyectivo al plano de la imagen es:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = K \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}, \tag{4.5}$$

donde, $s$ es el factor de escala vertical y

$$K = \begin{bmatrix} -k_u & 0 & 0 & u_0 \\ 0 & k_v & 0 & v_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

### 4.2.1. Parámetros intrínsecos

Al multiplicar las matrices **K** y **P** (proyección perspectiva seguida de una transformación afín) podemos escribir las ecuaciones del modelo geométrico de la cámara, es decir, la relación entre las coordenadas $(x, y, z)$, según el SCC, del punto $B$ y las coordenadas $(u, v)$, según el SCI, del punto $b$:

$$u = k_u \frac{fx}{z} + u_0 \qquad v = -k_v \frac{fy}{z} + v_0$$

de esta forma obtenemos el producto de $KP$:

$$KP = \begin{bmatrix} k_u & 0 & 0 & u_0 \\ 0 & -k_v & 0 & v_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} = \begin{bmatrix} k_u & 0 & \frac{u_0}{f} & 0 \\ 0 & -k_v & \frac{v_0}{f} & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \tag{4.7}$$

Al multiplicar todos los elementos de la matriz por $f$ (lo cual no afecta el resultado porque las coordenadas homogéneas de la matriz son definidas a un factor multiplicativo) obtenemos:

$$I_c = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.8}$$

lo cual es una aplicación lineal del espacio proyectivo hacia el plano proyectivo, por lo que es posible representar la transformación perspectiva de la siguiente manera:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = I_c \begin{bmatrix} x \\ y \\ z \\ s \end{bmatrix} \tag{4.9}$$

Como se puede ver, este modelo se compone de cuatro parámetros: $\alpha_u = k_u f$, $\alpha_v = -k_v f$, $u_0$ y $v_0$. Estos son los parámetros intrínsecos de la cámara estimados en el proceso de calibración, siendo $\alpha_u$ el factor de escala horizontal, $\alpha_v$ el factor de escala horizontal y $(u_0, v_0)$ las coordenadas del punto principal en la imagen. Al introducir las coordenadas de la cámara sin dimensiones, $x_c$, $y_c$ y $z_c$, tenemos que:

$$x_c = x/z \qquad y_c = y/z \qquad z_c = 1$$

Ahora se puede describir la relación entre las coordenadas de la imagen y las coordenadas de la cámara con las siguientes ecuaciones:

$$u = \alpha_u x_c + u_0 \qquad con \qquad \alpha_u < 0$$
$$v = \alpha_v y_c + v_0$$

En forma matricial tenemos:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = C \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \tag{4.10}$$

donde

$$C = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

Entonces, la matriz $I_c$, que contiene los parámetros intrínsecos, se compone de una transformación afín cámara-imagen (matriz $C$) y una transformación proyectiva:

$$I_c = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

## 4.3. Parámetros extrínsecos

Con el fin de conocer los parámetros extrínsecos del modelo de la cámara se coloca delante de esta un objeto de referencia que contenga un conjunto de puntos cuyas coordenadas en el sistema coordenado del mundo (SCM) sean perfectamente conocidas, dicho objeto será conocido como mira de calibración o, por simplicidad, mira. Cada uno de los puntos antes mencionados se proyecta en la imagen y se miden sus coordenadas en el SCI, ver figura 4.4.

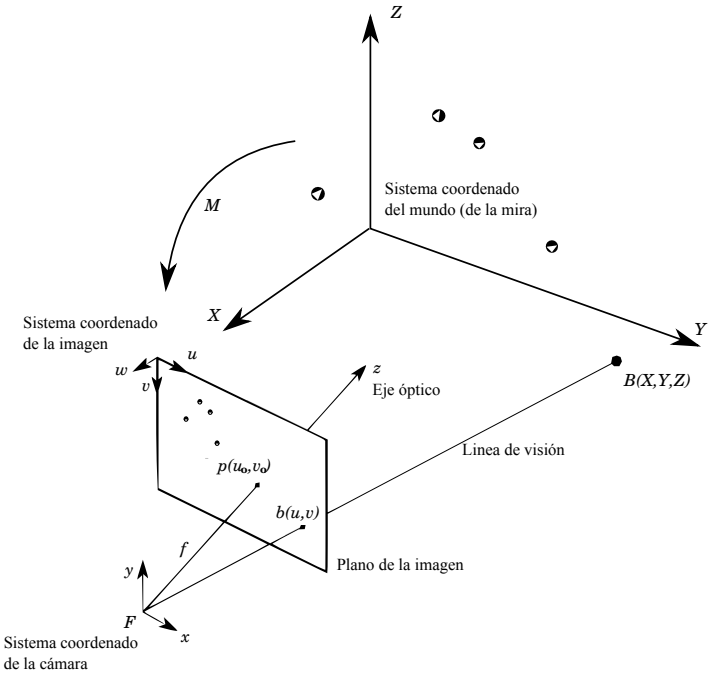



Figura 4.4: El principio de calibración de una cámara

La transformación mira-imagen se divide en una transformación mira-cámara, seguida de una proyección y finalmente de una transformación cámara-imagen.La transformación mira-cámara se compone de tres rotaciones y tres traslaciones:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.12}$$

Esta transformación rígida puede ser presentada en coordenadas homogéneas como sigue:

$$A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0^t} & 1 \end{bmatrix} \tag{4.13}$$

La matriz $A$ representa una transformación rígida de seis grados de libertad: rotaciones en los ejes $x$, $y$ y $z$ y traslación sobre los mismos. A sus elementos se les conoce como parámetros extrínsecos.

## 4.4. Transformación del sistema coordenado del mundo al sistema coordenado de la cámara

Ahora se puede escribir la transformación mira-imagen en forma de una matriz M de $3 \times 4$, llamada matriz de proyección perspectiva y que puede ser descompuesta como sigue:

$$M = I_c A$$

$$M = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u r_{11} + r_{31} u_0 & \alpha_u r_{12} + r_{32} u_0 & \alpha_u r_{13} + r_{33} u_0 & \alpha_u t_x + u_0 t_z \\ \alpha_v r_{21} + r_{31} v_0 & \alpha_v r_{22} + r_{32} v_0 & \alpha_v r_{23} + r_{33} v_0 & \alpha_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

$$M = \begin{bmatrix} \alpha_u \mathbf{r_1} + u_0 \mathbf{r_3} & \alpha_u t_x + u_0 t_z \\ \alpha_v \mathbf{r_2} + v_0 \mathbf{r_3} & \alpha_v t_y + v_0 t_z \\ \mathbf{r_3} & t_z \end{bmatrix} \tag{4.15}$$

El resultado de $M = I_c A$ está escrito de forma compacta al utilizar la notación $\mathbf{r_i} = (\mathbf{r_{i1}}\ \mathbf{r_{i2}}\ \mathbf{r_{i3}})$. Entonces tenemos que la matriz $A$ puede representarse de la siguiente forma:

$$A = \begin{bmatrix} \mathbf{r_1} & t_x \\ \mathbf{r_2} & t_y \\ \mathbf{r_3} & t_z \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.16}$$

Ahora se puede expresar la transformación mira-imagen con base en $M$:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}}_{MatrizM} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.17}$$

Las coordenadas $X$,$Y$ y $Z$ representan las coordenadas de un punto $B$ tridimensional en el sistema coordenado del mundo. Si consideramos $\mathbf{m_1} = (m_{11}, m_{12}, m_{13})$, $\mathbf{m_2} = (m_{21}, m_{22}, m_{23})$, $\mathbf{m_3} = (m_{31}, m_{32}, m_{33})$, entonces:

$$M = \begin{bmatrix} \mathbf{m_1} & m_{14} \\ \mathbf{m_2} & m_{24} \\ \mathbf{m_3} & m_{34} \end{bmatrix} \tag{4.18}$$

Al calcular $M$ con $I_cA$, teniendo en cuenta propiedades de ortonormalidad de la rotación y resaltando que se debe obtener un valor negativo para $\alpha_v$, obtenemos un conjunto de ecuaciones que permiten calcular los parámetros intrínsecos y extrínsecos en función de los coeficientes de $M$. De esta forma, tenemos:

$$\begin{aligned} \mathbf{r_3} &= \mathbf{m_3} \\ u_0 &= \mathbf{m_1} \cdot \mathbf{m_3} \\ v_0 &= \mathbf{m_2} \cdot \mathbf{m_3} \\ \alpha_u &= \|\mathbf{m_1} \wedge \mathbf{m_3}\| \\ \alpha_v &= -\|\mathbf{m_2} \wedge \mathbf{m_3}\| \\ t_x &= 1/\alpha_u(m_{14} - u_0 m_{34}) \\ t_y &= 1/\alpha_v(m_{24} - u_0 m_{34}) \\ t_z &= m_{34} \\ \mathbf{r_1} &= 1/\alpha_u(\mathbf{m_1} - u_0\mathbf{m_3}) \\ \mathbf{r_2} &= 1/\alpha_v(\mathbf{m_2} - v_0\mathbf{m_3}) \end{aligned} \tag{4.19}$$

donde $\cdot$ es el producto punto y $\wedge$ es el producto cruz. Para conocer los parámetros intrínsecos se debe primero estimar los coeficientes de $M$ y después extraer los parámetros de la cámara a partir de los coeficientes dados por las ecuaciones anteriores, a este proceso se le conoce como calibración de la cámara.

## 4.5. Calibración de una cámara

Se conoce como calibración al proceso de encontrar los parámetros intrínsecos y extrínsecos de la cámara, para esto se debe de:

1. Estimar los coeficientes de la matriz de proyección **M** y
2. extraer los parámetros de la cámara según las ecuaciones (4.19)

Las coordenadas en la imagen de un punto de la escena pueden escribirse usando la matriz $M$ y las coordenadas tridimensionales $(X, Y, Z)$ como:

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \tag{4.20}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \tag{4.21}$$

Estas dos ecuaciones describen la línea que pasa por el centro de proyección y el punto de la imagen $(u, v)$, en el sistema coordenado del mundo. Esta línea es la recta que pasa por $F$ y el punto $b$ en la figura 4.4, también llamada línea de visión.

Con el fin de calcular los coeficientes de la matriz $M$ es necesario escribir un sistema de ecuaciones a partir de los puntos de la mira de calibración y su proyección en la imagen. Cada punto $(X_i, Y_i, Z_i)$ se proyecta en $(u_i, v_i)$ generándose dos ecuaciones (4.20) y (4.21). Estas ecuaciones son linealmente independientes con respecto a los coeficientes de la matriz. Las ecuaciones (4.20) y (4.21) pueden ser reescritas como una combinación lineal de los $m_{ij}$ parámetros:

$$X_i m_{11} + Y_i m_{12} + Z_i m_{13} + m_{14} - u_i X_i m_{31} - u_i Y_i m_{32} - u_i Z_i m_{33} = u_i m_{34} \tag{4.22}$$

$$X_i m_{21} + Y_i m_{22} + Z_i m_{23} + m_{24} - u_i X_i m_{31} - u_i Y_i m_{32} - u_i Z_i m_{33} = u_i m_{34} \tag{4.23}$$

De esta forma, se obtienen $2n$ ecuaciones para $n$ puntos. Las ecuaciones pueden ser reescritas en forma matricial de la siguiente forma:

$$\mathbf{Qx} = \mathbf{u} \tag{4.24}$$

donde $\mathbf{Q}$ es una matriz de tamaño $2n \times 11$, el vector $x$ de $11 \times 1$ y el vector $u$ de $2n \times 1$. De forma más detallada tenemos que:

$$\mathbf{Q} = \begin{bmatrix} & & & & & & \vdots & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\ & & & & & & \vdots & & & & \end{bmatrix} \tag{4.25}$$

$$\mathbf{x} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{21} & m_{22} & m_{23} & m_{24} & m_{31} & m_{32} & m_{33} \end{bmatrix}^t \tag{4.26}$$

$$\mathbf{u} = \begin{bmatrix} \vdots \\ u_im_{34} \\ v_im_{34} \\ \vdots \end{bmatrix} \tag{4.27}$$

Por lo tanto, tenemos que:

$$\begin{bmatrix} & & & & & & \vdots & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\ & & & & & & \vdots & & & & \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \begin{bmatrix} \vdots \\ u_im_{34} \\ v_im_{34} \\ \vdots \end{bmatrix} \tag{4.28}$$

Los valores reales de los doce elementos que componen la matriz de proyección $M$, permiten conocer los parámetros intrínsecos y extrínsecos a partir de la ecuación (4.19). Para ello, se hace uso del método de calibración Faugueras-Toscani[25], el cual da una solución robusta al problema que se describe en la siguiente sección.

### 4.5.1. Método Faugeras-Toscani

En el caso del presente documento, la expresión (4.28) no es del todo funcional, ya que hace necesario que $m_{34} = 1$, cosa que no coincide con lo que se presenta en un problema de calibración real, por tanto es necesario redefinir la expresión de la siguiente manera:

$$\mathbf{Q_{2n\times 12}} = \begin{bmatrix} & & & & & & \vdots & & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \\ & & & & & & \vdots & & & & & \end{bmatrix} \tag{4.29}$$

$$\mathbf{x_{12}} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{21} & m_{22} & m_{23} & m_{24} & m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}^t \tag{4.30}$$

Lo que deja un sistema de ecuaciones:

$$\mathbf{Qx} = \mathbf{0} \tag{4.31}$$

que puede ser resuelto como un problema de optimización sujeto a restricciones.

Al calcular los coeficientes de $M$ en función de los coeficientes que componen las matrices $I_c$ y $A$, encontramos que:

$$m_{31} = r_{31}, \qquad m_{32} = r_{32}, \qquad m_{33} = r_{33}$$

y ya que la matriz de rotación formada por $r_1$, $r_2$ y $r_3$ es ortonormal se asume que $r_{31}^2 + r_{32}^2 + r_{33}^2 = 1$; lo cual puede ser utilizado como restricción

$$\| \mathbf{m}_3 \|^2 = m_{31}^2 + m_{32}^2 + m_{33}^2 = 1 \tag{4.32}$$

A continuación se muestra cómo calcular $M$ tomando en cuenta la restricción anterior. La ecuación (4.31) se puede desglosar de la siguiente manera:

$$\mathbf{Bx_9} + \mathbf{Cx_3} = 0 \tag{4.33}$$

donde $\mathbf{B}$ es una matriz de tamaño $2n \times 9$ y $\mathbf{C}$ una matriz de $2n \times 3$:

$$\mathbf{B} = \begin{bmatrix} & & & & \vdots & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i \\ & & & & \vdots & & & & \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \vdots & & \\ -u_iX_i & -u_iY_i & -u_iZ_i \\ -v_iX_i & -v_iY_i & -v_iZ_i \\ \vdots & & \end{bmatrix} \tag{4.34}$$

Como se puede observar en la ecuación (4.33), es necesario descomponer el vector $\mathbf{x_{12}}$ en dos incógnitas $\mathbf{x_9}$ y $\mathbf{x_3}$, así tenemos:

$$\mathbf{x_9} = \begin{bmatrix} \mathbf{m_1} & m_{14} & \mathbf{m_2} & m_{24} & m_{34} \end{bmatrix}^t \tag{4.35}$$

y

$$\mathbf{x_3} = (\mathbf{m_3}) \tag{4.36}$$

El vector $\mathbf{x_3}$ representa a las variables necesarias para aplicar la restricción indicada en (4.32). Entonces, el problema de optimización para resolver el sistema de ecuaciones es el siguiente:

$$L = \underset{x_3}{Argmin} \quad \left( \| \mathbf{Bx_9} + \mathbf{Cx_3} \|^2 \right) \tag{4.37}$$

con la restricción:

$$\| \mathbf{x_3} \|^{\mathbf{2}} = \mathbf{1}$$

Utilizando multiplicadores de Lagrange, la función puede ser escrita de la siguiente manera:

$$\mathbf{L} = \| \mathbf{Bx_9} + \mathbf{Cx_3} \|^2 + \lambda(1 - \| \mathbf{x_3} \|^2) \tag{4.38}$$

desarrollando esta expresión tenemos:

$$\mathbf{L} = \mathbf{x_9^t B^t B x_9} + \mathbf{x_3^t C^t C x_3} + bf x_9^t B^t C x_3 + \mathbf{x_3^t C^t B x_9} + \lambda(1 - \mathbf{x_3^t x_3}) \tag{4.39}$$

calculando e igualando a cero las derivadas parciales con respecto a $\mathbf{x_9}$ y $\mathbf{x_3}$, resultan las siguientes dos ecuaciones:

$$\mathbf{B^t B x_9} + \mathbf{B^t C x_3} = \mathbf{0}$$

$$\mathbf{C^C x_3} + \mathbf{C^t B x_9} - \lambda \mathbf{x_3} = \mathbf{0},$$

de las cuales obtenemos:

$$\mathbf{x_9} = -(\mathbf{B^t B})^{-\mathbf{1}} \mathbf{B^t C x_3} \tag{4.40}$$

$$\mathbf{D x_3} = \lambda \mathbf{x_3} \tag{4.41}$$

$$\mathbf{D} = \mathbf{C^t C} - \mathbf{C^t B} (\mathbf{B^t B})^{-\mathbf{1}} \mathbf{B^t C} \tag{4.42}$$

finalmente, sustituyendo estas ecuaciones en (4.39) tenemos:

$$L = \mathbf{x_3^t D x_3} = \lambda \mathbf{x_3^t x_3} = \lambda \tag{4.43}$$

**D** es una matriz simétrica y definida positiva de tamaño $3 \times 3$. Ésta tiene sus valores propios reales y positivos. Como se puede ver en la ecuación (4.41), $\mathbf{x_3}$ es un vector propio de **D** asociado al valor propio de la variable $\lambda$. Por lo tanto, para encontrar el valor mínimo de $\mathbf{x_3}$ y resolver el sistema de ecuaciones es necesario seguir el siguiente procedimiento:

1. Formar la matriz D
2. Calcular los valores propios de **D**
3. Escoger el valor propio más pequeño (este valor es el que minimiza Q)
4. Calcular el vector propio ($\mathbf{x_3}$) asociado al valor propio seleccionado.
5. Normalizar el vector propio ($\mathbf{x_3}$)
6. Finalmente, calcular el vector $x_9$

Los elementos de $M$ están dados por los vectores $\mathbf{x_3}$ y $\mathbf{x_9}$. El signo del vector propio $\mathbf{x_3}$, no está definido, teniendo así dos soluciones: $M$ y $-M$. Podemos escoger una de estas dos soluciones tomando en cuenta que la mira a calibrar se encuentra frente a la cámara y no atrás de ésta. Así pues $m_{34} = t_z > 0$.

# Capítulo 5

# Propuesta de solución

El modelo de aplicación de realidad aumentada que se propone cuenta con los siguientes módulos:

- Captura de video u obtención de datos.
- Detector de puntos de interés (SIFT).
- Clasificador.
- Seguimiento.
- Calibración.
- Dibujado de gráficos 3D (OpenGL).
- Procesamiento de la imagen.
- Aumentado.

El módulo de obtención de datos se encarga de la captura de video de la escena real. Cuando el usuario de una orden, la imagen captada por este módulo será procesada por un módulo detector y descriptor de puntos de interés; una vez que se tiene el conjunto de descriptores, estos son clasificados (en el módulo clasificador) para definir a cuál de los puntos tridimensionales corresponden. Hasta este punto se completa la etapa de detección y clasificación.

Con los puntos identificados se forman las correspondencias 2D - 3D necesarias para la calibración de la cámara. En el módulo de calibración se obtienen los parámetros intrínsecos y extrínsecos que son transmitidos al componente encargado del dibujado de los gráficos, también se obtiene una matriz de transformación que se utilizará para simular la superposición

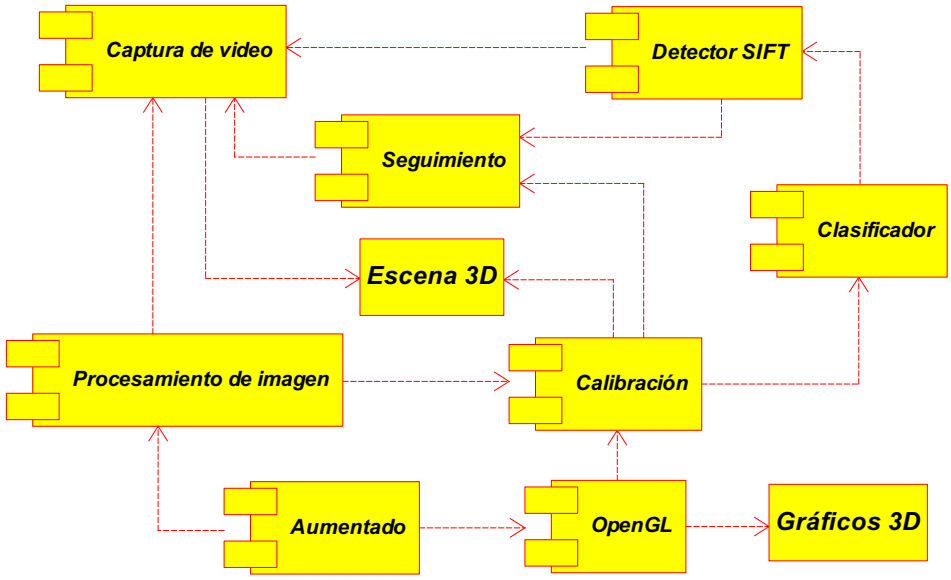



Figura 5.1: Componentes del modelo propuesto

de objetos en el módulo de procesamiento de la imagen. Finalmente, el módulo de aumentado se encargará de unir la imagen procesada y los gráficos 3D y presentarlos en el dispositivo de salida. Hasta este punto se completa la primera iteración del ciclo de realidad aumentada.

Los puntos 2D utilizados para la calibración se almacenan y en la siguiente iteración no se realiza detección ni clasificación alguna, sino que estos puntos son procesados en el módulo de seguimiento, junto con la nueva imagen capturada por la cámara, para obtener de ahí su nueva posición y con ella calibrar nuevamente la cámara. Este ciclo se repite hasta que el usuario decida finalizarlo o bien cuando la prueba de integridad realizada a los puntos durante el seguimiento no se supere. En este último caso, será necesario volver a la detección de puntos en la nueva imagen. La estructura de componentes del modelo se puede ver en la figura 5.1.

En los siguientes apartados se mostrarán los detalles de cada componente del modelo de aplicación de realidad aumentada

## 5.1. Obtención de datos

En los equipos de computación de uso común, generalmente no se cuenta con sensores de movimiento, cámaras infrarrojas, lentes convexos o sistemas GPS; las herramientas de obtención de información externa más comunes que una computadora de escritorio posee son: teclado, ratón, micrófono y cámara de video. Las cámaras de video más populares en estos equipos son las llamadas cámaras web, usadas principalmente con fines de comunicación.

Debido a esto, en un intento por acercar las aplicaciones de RA al uso cotidiano de la computación, en el caso particular de este trabajo, se busca crear un modelo de aplicación de realidad aumentada que utilice como única fuente de captura de datos una cámara web genérica.

El *software* involucrado en el proceso de adquisición de video es la biblioteca de procesamiento de imágenes y visión por computadora OpenCV [13] para lenguaje C y C++. Misma que se eligió debido a su facilidad de uso, probado funcionamiento en experiencias anteriores, soporte multiplataforma y licencia de código abierto.

### 5.1.1. Captura de video con OpenCV

Para capturar imágenes de un dispositivo de video cualquiera con OpenCV se utilizan las estructuras cvCapture e IplImage [14], además de algunas funciones que se explican a continuación:

```
CvCapture *capture;
IplImage *img;
capture=cvCaptureFromCAM(0);

cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 320);
cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 240);

if ( cvGrabFrame( capture )){
    IplImage *img= cvRetrieveFrame( capture );
}
```

En este fragmento de código se puede ver la forma básica de capturar un fotograma de video utilizando OpenCV, primero se definen los apuntadores a las estructuras `CvCapture` y e `IplImage`, luego se asigna un valor a `capture` por medio de `cvCaptureFromCAM(0)` que indica que se utilizará la cámara predefinida del sistema (la cámara 0), las siguientes dos instrucciones indican el tamaño de la imagen a capturar; luego `cvGrabFrame( capture )` envía a la cámara la señal de iniciar la captura de un fotograma, devolviendo `NULL` si ocurre algún error. Finalmente se asigna a `img` la imagen capturada por la cámara y será el valor apuntado por `img` del que se obtenga la información que se someterá a procesamiento.

La estructura IplImage está definida como sigue:

```
typedef struct _IplImage
{
  int  nSize;
```

```
    int  ID;
    int  nChannels;
    int  alphaChannel;
    int  depth;
    char colorModel[4];
    char channelSeq[4];
    int  dataOrder;
    int  origin;
    int  align;
    int  width;
    int  height;
    struct _IplROI *roi;
    struct _IplImage *maskROI;
    void  *imageId;
    struct _IplTileInfo *tileInfo;
    int  imageSize;
    char *imageData;
    int  widthStep;
    int  BorderMode[4];
    int  BorderConst[4];
    char *imageDataOrigin;
}
IplImage;
```

Son de particular interés las variables `width` y `height` que indican el ancho y alto de la imagen en pixeles, respectivamente; `nChannels` que indica el número de canales que tiene la imagen; `imageData` que contiene la información de intensidad de color de la imagen, esto es, la imagen propiamente dicha; esta información se presenta por medio de una arreglo de caracteres de tamaño `width` x `height` x `nChannels` para el caso de 8 bits de profundidad en el color[1]. Por ejemplo, en el caso de una imagen RGB de 320x240, el tamaño del arreglo `imageData` sería de 320x240x3 y cada pixel estaría representado por 3 espacios contiguos en el arreglo siendo los primeros 8 bits (el primer carácter) el valor de intensidad correspondiente al color rojo, los segundos al verde y los terceros al azul.

Para mayor información sobre esta estructura y las funciones de OpenCV en general, se recomienda la consulta de [14] y [13].

[1]Indica que cada canal está representado en un pixel por 8 dígitos binarios

## 5.2. Calibración automática

Para lograr la calibración automática de la cámara, se propone un procedimiento que se divide en dos etapas, la primera, que llamaremos de entrenamiento, se debe realizar antes de iniciar el proceso de calibración; la segunda, de calibración, consiste en el proceso de calibración propiamente dicho.

### 5.2.1. Etapa de entrenamiento

Se requiere contar con un objeto de referencia, mismo que deberá encontrarse en la escena al momento de realizar la calibración. En esta primera etapa se realiza el entrenamiento de un clasificador que será capaz de reconocer, posteriormente, un grupo de puntos cuya información tridimensional se conoce. El procedimiento para la obtención de la muestra de entrenamiento y el entrenamiento mismo es el siguiente:

1. Tomar varias fotografías del objeto de referencia, buscando contar con muestras suficientes de cada parte del mismo y muestras con variaciones de escala, orientación, condiciones de luz.

2. Ejecutar un detector y descriptor de puntos de interés sobre las imágenes de la muestra.

3. Seleccionar aquellos puntos del objeto que hayan sido más consistentemente detectados en la muestra, cuidando que la distribución de los mismos en el espacio cubra el mayor volumen posible y que no sean todos coplanares.

4. Obtener la información tridimensional de los puntos seleccionados, tomando como origen un punto cualquiera en el objeto de referencia.

5. Agrupar los descriptores de los puntos seleccionados por su correspondencia a los puntos en el objeto, es decir, se forman las clases.

6. Numerar o etiquetar desde 1 hasta $n$ ($n$ es el número de grupos) cada uno de los grupos de puntos formados en el paso anterior.

7. Formar un conjunto de descriptores de puntos de aquellos que no pertenezcan a ninguna de las clases etiquetadas, este conjunto será la clase 0 o clase negativa.

8. Entrenar un clasificador con el conjunto de entrenamiento formado por las clases positivas y la negativa (clases 0 hasta $n$).

Por lo expuesto en [55], en el presente trabajo se utilizará el método SIFT para la detección y descripción de puntos de interés, en particular la implementación de Vedaldi [81]. En esta implementación de código abierto, se procesa una imagen en niveles de gris dando como resultado un arreglo que contiene las coordenadas $(u, v)$ del punto en la imagen con precisión de centésimas de pixel, la magnitud y orientación del gradiente y los 128 descriptores del punto normalizados entre 0 y 255.

### 5.2.2. Etapa de calibración

Una vez que se tiene un clasificador entrenado para identificar un cierto conjunto de puntos, se puede realizar la calibración de la cámara con una imagen desconocida que contenga al objeto de referencia. El algoritmo 1 muestra cómo se realiza este procedimiento, se tiene una imagen con la que se desea calibrar la cámara, un clasificador entrenado y la información tridimensional correspondiente a cada clase positiva del clasificador.

**Algoritmo 1** Procedimiento de autocalibración, una vez que se tiene el clasificador entrenado

```
puntos ← Detectar_puntos(imagen)
for x = 1 → total_puntos do
  clase ← Clasificar(puntos(x))
  if clase > 0 then
    agrega puntos(x) a puntos_calibración
  end if
end for
Calibrar(puntos_calibración,puntos_3D)
```

## 5.3. Colocación de objetos

Una vez que se tienen los valores de calibración de la cámara, se pueden aplicar las transformaciones al mundo de OpenGL para empatar el origen y la orientación del mismo con las del mundo real.

El primer paso es empatar el sistema de coordenadas del mundo OpenGL con el del mundo real, para esto se hace los siguiente:

- Establecer el origen del mundo real en un punto del objeto de referencia. Este mismo punto será el origen del mundo OpenGL.
- Se considerará que las unidades adimensionales de OpenGL son milímetros, esto para utilizar las mismas unidades en OpenGL y el mundo real.

De esta manera, solo restaría simular las propiedades intrínsecas de la cámara real (longitud de foco, punto central y escala) en la cámara de OpenGL y realizar las transformaciones (parámetros extrínsecos) de rotación y traslación sobre la misma. Lo primero se hace mediante la modificación de la matriz de proyección de OpenGL:

```
float proyMat[16]={
  (2*(Uo))/ancho,          0,                                  0,  0,
               0,  (Vo))/alto,                                 0,  0,
(2*(Ku))/ancho-1, (Kv)/alto-1,  -(lejos+cerca)/(lejos-cerca), -1,
               0,           0,-(2*lejos*cerca)/(lejos-cerca),  0
};
glViewport(0, 0, ancho, alto );
glMatrixMode( GL_PROJECTION );
glLoadMatrixf( proyMat );
```

donde `Uo`, `Vo`, `Ku` y `Kv` son los parámetros intrínsecos de la cámara obtenidos mediante la calibración, ancho y alto son las dimensiones de la ventana en la que se mostrarán los gráficos y cerca y lejos son las distancias del plano cercano y el plano lejano al observador.

El siguiente paso sería aplicar a la cámara virtual las mismas transformaciones que se aplicaron a la real para capturar la imagen. Esto es, trasladarla y rotarla de la misma manera que se trasladó y rotó la cámara real. Esto se logra modificando la matriz de modelado del mundo:

```
float modelMat[16]= {
  R[0][0], R[1][0], R[2][0],0,
  R[0][1], R[1][1], R[2][1],0,
  R[0][2], R[1][2], R[2][2],0,
     T[0],    T[1],    T[2],1
};
glMatrixMode( GL_MODELVIEW );
glLoadMatrixf( modelMat );
```

`R` representa la matriz de rotación perteneciente a los parámetros extrínsecos y `T` el vector de traslación $(x,y,z)$.

Cabe hacer notar que mediante las transformaciones indicadas y utilizando las mismas unidades y el mismo punto de origen, no es necesario aplicar ninguna otra forma de procesamiento a los objetos virtuales para que puedan aparecer en la escena con la correcta ubicación y alineación.

Finalmente se agrega la imagen como una cubierta a la ventana que deja ver el mundo virtual (figura 5.2). Se hace uso de la función glDrawPixels de OpenGL para esa tarea. El código en C para OpenGL sería el siguiente (se omiten los acentos por cuestiones de compatibilidad con el conjunto de caracteres):

```
glPushMatrix();
  //Aqui se agregan los objetos virtuales
glPopMatrix();

//--------Poner imagen de la camara
begin2D ();
  glRasterPos3i(0,0,0);
  glDrawPixels(ancho,alto,color,GL_UNSIGNED_BYTE,imageData);
end2D();
```

Las variables `ancho` y `alto` son las dimensiones de la imagen dadas en pixeles, `color` es la constante OpenGL que indica el formato de color de la imagen, en este caso deberá ser `GL_BGRA` o `GL_RGBA`, `GL_UNSIGNED_BYTE` es la profundidad de color de cada pixel (aquí, 8 bits por canal) e `imageData` es un arreglo de bytes que contiene la información de la imagen. `glRasterPos3i(0,0,0)` indica la posición en la que se empezará a dibujar la imagen.

Las funciones `begin2D` y `end2D` se encargan de preparar el entorno de OpenGL para el dibujado en dos dimensiones:

```
void begin2D (){
  int width,height;
  GetSize(&width,&height);

  glMatrixMode (GL_PROJECTION);
  glPushMatrix ();

  glLoadIdentity ();
  glOrtho (0, width, 0, height, -1.0f, 1.0f);

  glMatrixMode (GL_MODELVIEW);
  glLoadIdentity ();
}
void end2D (){
  glMatrixMode (GL_PROJECTION);
```

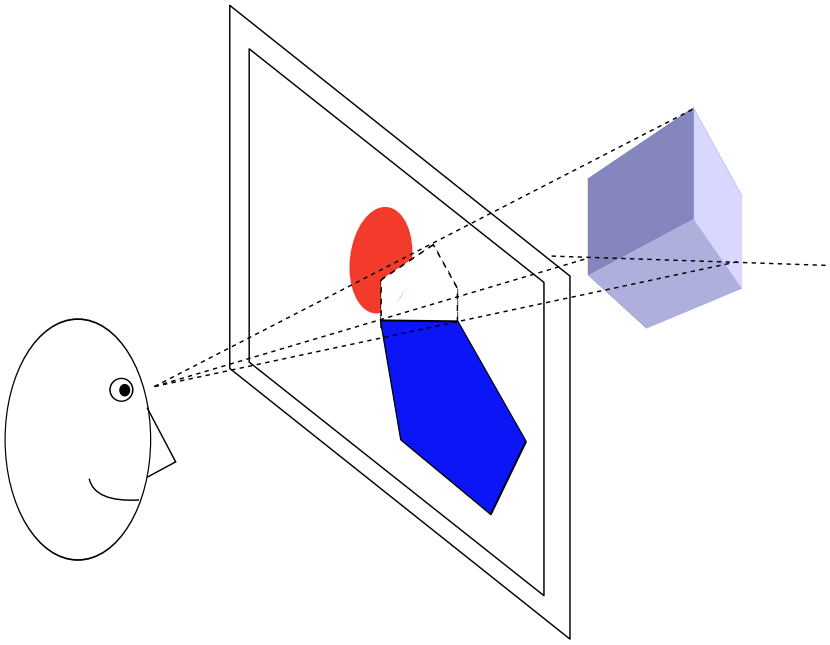


Figura 5.2: La imagen de la escena real deberá ser "perforada" en aquellas zonas donde algún objeto virtual deba cubrir a los capturados por la cámara

```
    glPopMatrix ();
    glMatrixMode (GL_MODELVIEW);
}
```

## 5.4. Simulación de la superposición de objetos

En este trabajo se utilizará una metodología similar a la propuesta por Berger en [9], en donde la máscara de oclusión se calcula con base en segmentos de rectas y no en *snakes* y se aplica a la imagen y no a los objetos virtuales. A continuación se explica su funcionamiento.

Se parte del supuesto de que se cuenta con una cámara calibrada, se conoce la escena y se sabe qué objetos virtuales aparecerán delante de qué objetos reales. La idea general es que los objetos virtuales no se colocan delante de la imagen, si no detrás de ésta. Y es la imagen la que deberá procesarse para eliminar de ella las partes que deberán parecer detrás de los elementos insertados.

Para la eliminación de las zonas cubiertas se propone la utilización de la capacidad de transparencia del canal *alpha* en las imágenes RGBA. El proceso consiste en agregar primero el cuarto canal (*alpha*) de color a la imagen capturada por la cámara en RGB. Luego, habiendo ya obtenido los datos de la calibración de la cámara y conociendo la posición tridimensional

de los objetos virtuales se realiza la proyección sobre la imagen de las esquinas que serán visibles (5.1). Con esto se obtiene una colección de puntos 2D que representan el contorno que tendrá el objeto virtual en la imagen, ver figura 5.2. Dicho de otra forma, si tenemos un objeto virtual con esquinas (o un conjunto de puntos tridimensionales de su borde) $P_i = x_i, y_i, z_i$ y una matriz de transformación $M$ de 3 x 4 obtenida en el proceso de calibración, el conjunto de puntos de la imagen $p_i = u_i, v_i$ donde $P$ se proyecta se obtiene por medio de:

$$p_i = M \otimes P_i \tag{5.1}$$

Una vez que se tienen los puntos de la imagen donde el objeto aparecerá, es necesario hacer transparentes todos aquellos pixeles que estén dentro de este conjunto de puntos, es decir, que sean envueltos por el polígono formado por $p$ en la imagen. El algoritmo sería como sigue:

**Algoritmo 2** Hacer transparentes las zonas de la imagen que serán cubiertas por objetos virtuales

```
for k =min_v → max_v do
  for l =min_u → max_u do
    if x_kl Dentro de p y color >= umbral then
      x ← transparente
    end if
  end for
end for
```

Donde min_u y min_v son los valores más pequeños en $p$ de coordenadas horizontales y verticales respectivamente y max_u y max_v son los más altos valores. Estos valores son útiles para delimitar el área de búsqueda dentro de la imagen, es decir, sólo son de interés aquellos pixeles que están dentro del rectángulo en el que se encuentra la proyección del objeto virtual . *umbral* es un umbral de color previamente definido, para hacer que todos los pixeles debajo de ese umbral permanezcan opacos y se conserven *sobre* el objeto virtual, mientras que los que superen el umbral quedarán *debajo*.

La función Dentro evalúa la posición de las coordenadas del pixel dado con respecto a las líneas formadas por cada par de puntos consecutivos de $p$, considerando que $p$ forma un polígono cerrado y que, para n puntos, $p_1$ es seguido de $p_2$ de la misma manera que $p_n$ es seguido de $p_1$. Para conocer dicha posición se toman los puntos ordenados en un sentido fijo alrededor del contorno del objeto, en este caso en sentido antihorario, ver figura 5.3, entonces, la posición de un punto $x$ con respecto a está dada por:

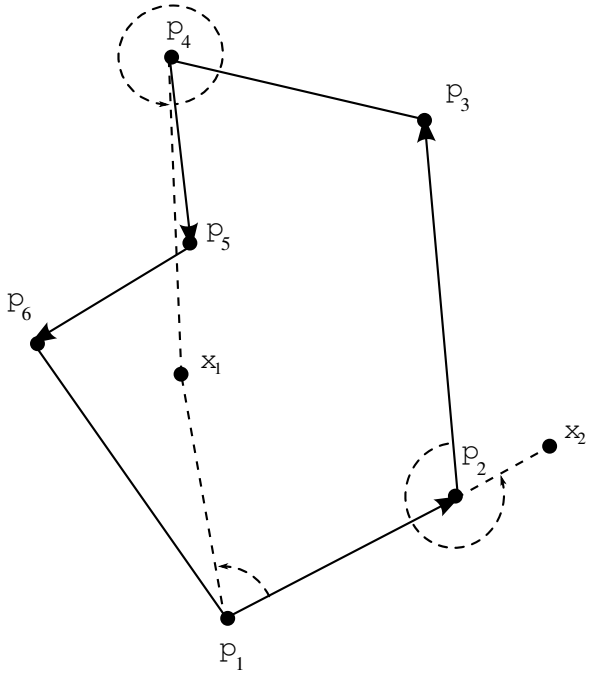



Figura 5.3: Como se puede observar, en las esquinas convexas un sólo ángulo mayor a $180^o$ significa que el punto está fuera del polígono ($x_2$). Mientras que en una esquina cóncava, una de las líneas que la forman tendrá un ángulo superior a los $180^o$ con respecto al punto aún cuando esté dentro del polígono

$$lado(x, \overline{p_i p_j}) = \begin{cases} 1 & \text{Si} \quad det\,(x, p_i - p_j) >= 0 \\ 0 & \text{Si} \quad det\,(x, p_i - p_j) < 0 \end{cases} \tag{5.2}$$

siendo $det\,(x, p_i - p_j)$ el determinante de la matriz formada por:

$$\begin{bmatrix} x_u & p_{iu} - p_{ju} \\ x_v & p_{iv} - p_{jv} \end{bmatrix}$$

y $p_{iu}$ y $p_{iv}$ son las coordenadas horizontal y vertical, respectivamente, del punto $p_i$ en la imagen. Lo mismo para $p_j$.

Es decir, dado que $det(X, X') \geq 0$ si y sólo si$0 \leq \theta \leq \pi$ ($\theta$ es el ángulo entre ambos vectores $(X, X')$) [4], se puede decir que si el ángulo entre las líneas $\overline{p_i p_j}$ y $\overline{p_i x_{kl}}$ menor a $180^o$, el determinante indicado en la ecuación 5.2 es mayor o igual a 0, entonces se considera que el punto está del lado interno (1) de la línea, de lo contrario estaría del externo (0), llamamos lado interno al lado correspondiente a la parte interna del polígono. Cuando un punto está del lado interno de todas las líneas, en el caso de polígonos convexos, se dice que está dentro del polígono. Para polígonos con segmentos cóncavos, será necesario definir las situaciones donde algunos ángulos serán mayores a $180^o$ y el pixel aún está dentro del polígono, ver figura 5.3.

El algoritmo 2 puede ser modificado para permitir transparencias parciales, difuminados, suavizados y otros efectos que podrían ayudar a mejorar la experiencia del usuario. Sólo es necesario especificar una función que determine el grado de transparencia que le corresponde a un pixel según su posición, color o alguna otra característica.

## 5.5. Seguimiento

Para hacer el seguimiento se propone la utilización del algoritmo KLT (Kanade-Lucas-Tomasi) propuesto por Shi y Tomasi [73], basándose en los desarrollo de Lucas y Kanade [50]. La metodología a seguir para el seguimiento, la corrección de errores y determinar cuando se deberá realizar nuevamente la detección e identificación de puntos se define en el algoritmo 3.

Sea $R_t = \{r_t^1 ... r_t^n\}$ el conjunto de $n$ puntos de referencia en fotograma en el tiempo $t$, proyectados en la imagen desde el conjunto de puntos 3D $O = o_1 ... o_n$ y $r_{t+1} = \{r_{t+1}^1 ... r_{t+1}^n\}$ el conjunto de puntos dado por el seguimiento para el fotograma en el tiempo $t + 1$.

Entonces, luego de realizar el seguimiento de los puntos en la imagen $t$, se calculan los vectores de movimiento con respecto a los puntos obtenidos para $t + 1$. Los puntos $t + 1$

**Algoritmo 3** Procedimiento para seguimiento de puntos de referencia y corrección de errores

Seguimiento($R_t$, $R_{t+1}$)
Calcular_vectores($R_t$, $R_{t+1}$) $\rightarrow$ vector
**for** $i = 1 \rightarrow n$ **do**
  $1 \rightarrow$ valido[$i$]
  **if** (vector$\left[r_{t+1}^{i}\right]$ - vector_promedio($R_{t+1}$) ¿umbral **then**
    $0 \rightarrow$ valido[$i$]
  **end if**
**end for**
**if** ConsistenciaAfin($R_1$,$R_{t+1}$)==0 **then**
  calcular_SIFT
  $1 \rightarrow t$
**else**
  **for** $i = 1 \rightarrow n$ **do**
    **if** valido[$i$]==0 **then**
      proyecta ($o_i$) $\rightarrow r_{t+1}^{i}$
      $1 \rightarrow$ valido[$i$]
    **end if**
  **end for**
**end if**
Calibrar
Ajustar_OpenGL
Oclusiones

correspondientes a aquellos vectores que no coincidan, dentro de cierto rango de tolerancia, con el vector de movimiento promedio del conjunto $R_{t+1}$ (o algún subconjunto que represente los $k$ vecinos más cercanos al punto en cuestión) se etiquetarán como *no válidos*. Luego se realiza la prueba de consistencia afín [73] entre los puntos en el primer fotograma de la serie de seguimientos y los del fotograma $t+1$. Si esta prueba se supera, los puntos etiquetados como *no válidos* son reproyectados desde $O$ y se vuelven a etiquetar como *válidos*. En caso de que no se supere la prueba de consistencia, es necesario volver a detectar e identificar nuevamente los puntos. Luego se procede con la calibración de la cámara, los ajustes del mundo virtual y la simulación de la superposición de objetos que ya se han explicado antes.

## 5.6. Ejemplo de uso

Se plantea el desarrollo de una aplicación de realidad aumentada cuya funcionalidad será asistir en el trazado en papel de un dibujo que se tiene almacenado en formato digital. La aplicación que se propone contará con las siguientes características:

- Utilizará el modelo propuesto en este documento.
- El caballete donde se colocará la hoja de papel (ver figura 5.4) contendrá algunas texturas que se utilizarán para la detección de puntos de interés.
- Se deberá simular la superposición del dibujo sobre la hoja de papel, y de la mano del usuario sobre el dibujo.
- En una primera versión de la aplicación, la posición de la cámara con respecto al caballete no cambiará durante toda la ejecución, por lo que se hace innecesario el proceso de seguimiento.
- Se asume que la hoja de papel estará siempre en la esquina inferior izquierda del caballete.

Esta aplicación, que llamaremos Asistente de Dibujo en Realidad Aumentada (ADRA), utilizará una configuración basada en video, específicamente con un monitor (ver Introducción). El *hardware* utilizado por la aplicación consistirá en una computadora personal, una cámara web y una caballete diseñado especialmente para dicha aplicación. La distribución del equipo se puede ver en la figura 5.5.

El funcionamiento de ADRA, que se expresa en forma de diagrama de colaboraciones en la figura 5.6, consiste en que esta iniciará mostrando la captura de video sin nungún objeto virtual en ella (los objetos virtuales están permanentemente dibujándose, aunque no se muestren),

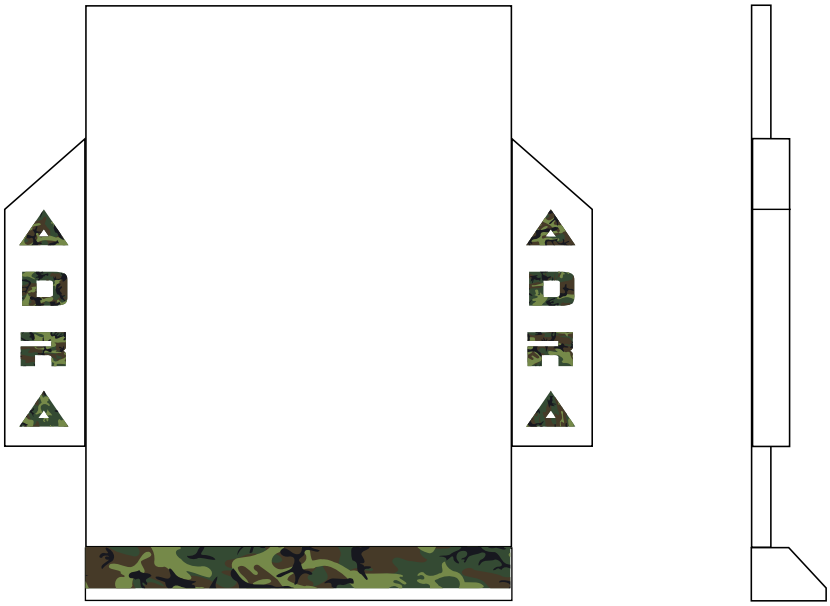



Figura 5.4: Vistas frontal y lateral del modelo de caballete que se diseñó para el ADRA. Se puede observar la existencia de tres zonas con texturas distintivas a la izquierda, en la franja central inferior y a la derecha; éstas serán utilizadas para la detección e identificación de puntos de interés.

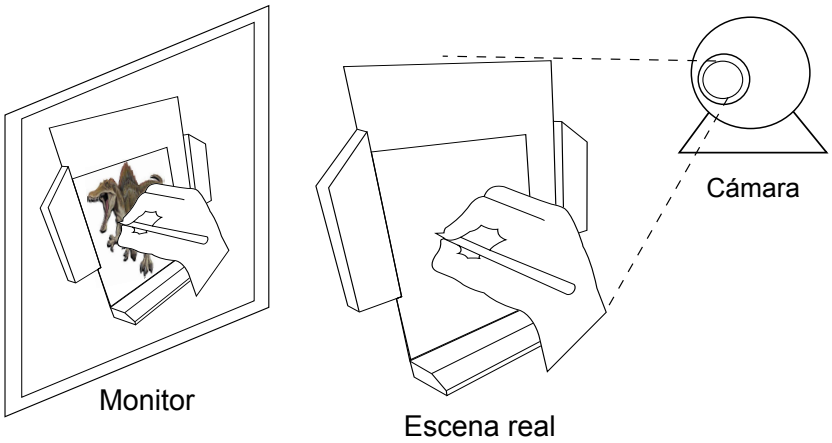



Figura 5.5: Disposición del *hardware* del ADRA

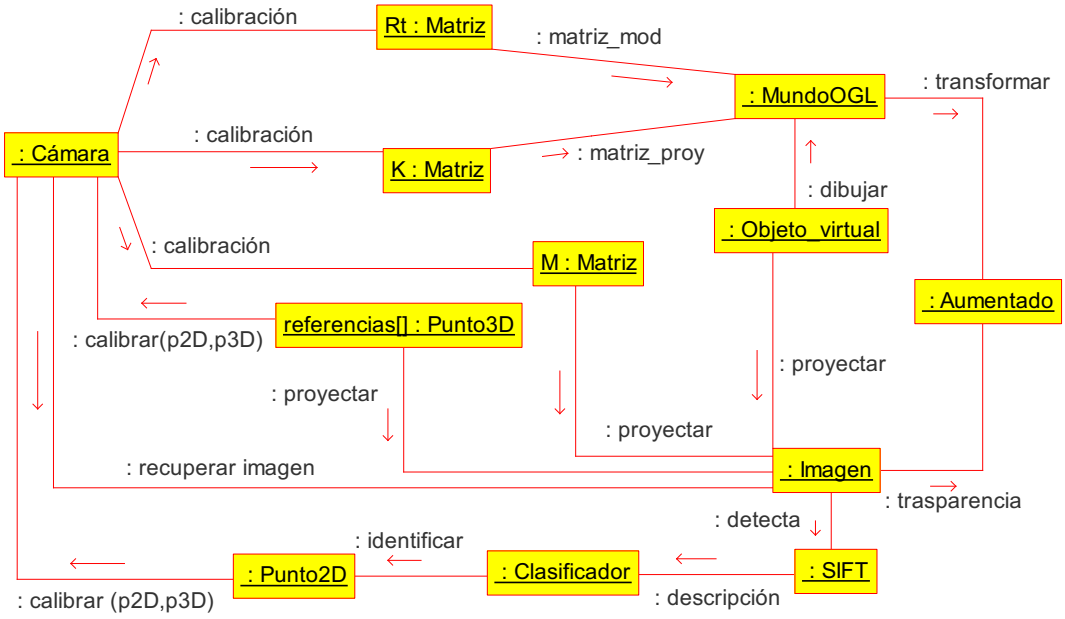



Figura 5.6: Diagrama de colaboraciones entre los objetos de la aplicación

cuando el usuario pulsa una tecla específica ("s", po ejemplo), inicia la detección y descripción de los puntos de interés sobre la imagen capturada en ese momento por la cámara, habiendo obtenido los descriptores, éstos se someten a un proceso de clasificación con SVM, de este proceso resulta un conjunto de puntos 2D cuya localización con respecto al origen del caballete se conoce, con esta información se procede a realizar la calibración de la cámara, de la cual se recuperarán 3 matrices: $Rt$, que representa los parámetros extrínsecos, $K$, o parámetros intrínsecos y $M$, que es la matriz de transformación proyectiva. Con $K$ y $Rt$ se hace el ajuste del mundo OpenGL, para empatarlo con el sistema de coordenadas de la escena. Con $M$, se realiza la proyección de los puntos de referencia no detectados y de los objetos virtuales sobre la imagen, ésto último para aplicar el método de simulación de superposiciones propuesto. Una vez que el mundo virtual fue transformado según los parámetros de la cámara y que la imagen fue procesada con ta transparencia, se realiza el aumentado de la imagen integrándole el mundo virtual como fondo y se muestra el resultado en pantalla. Para el siguiente fotograma, se parte de la posición previa de los puntos de referencia en la imagen, con ellos se realiza la calibración de la cámara y se sigue el proceso ya explicado de ajustar el mundo virtual, simular superposición en la imagen y mostrar los resultados integrados, así sucesivamente hasta que el usuario decida terminar la ejecución.

En resumen, ADRA contará con tres estados básicos, iniciará mostrando el video que captura directamente de la cámara. Luego, ante una orden del usuario, entrará en el estado de detección y clasificación de puntos de referencia una vez que se cumple con una serie de

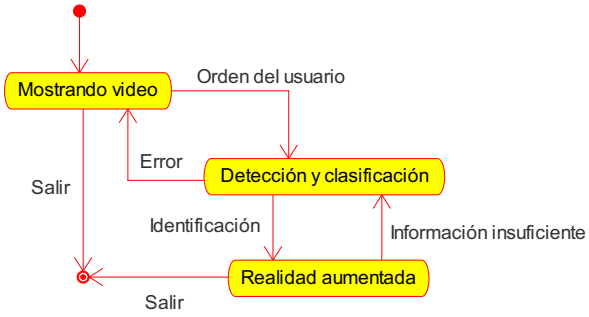



Figura 5.7: Estados por los que atravesará una aplicación

requisitos después de la clasificación (mínimo de puntos, mínimo de certeza en los resultados, etc) se puede pasar a la siguiente etapa o bien, si se produce algún error se regresa a mostrar únicamente el video. Cuando se tiene ya un conjunto de puntos en la imagen que corresponden con algunos de los puntos 3D conocidos, se pasa al estado de realizada aumentada, donde se realizan todas las de calibración y aumentado necesarias para mostrar un video con objetos virtuales incluidos. Esto se representa en el diagrama de estados de la figura 5.7.

A continuación se explican algunos detalles de diseño de la aplicación.

### 5.6.1. Selección de puntos para clasificación

Para elegir los puntos que representarían las clases se tomaron varias imágenes del caballete desde diferentes ángulos, distancias y con cambios en las condiciones de iluminación. De aquí se obtuvo un conjunto final de sesenta imágenes. Se utilizó SIFT con estas imágenes, y se seleccionaron aquellos puntos del caballete que fueran más consistentemente detectados.

### 5.6.2. Selección de clases para la calibración

Un proceso de clasificación rara vez clasifica correctamente el 100 % de los elementos que no conoció durante el entrenamiento; por lo tanto, si se sugiere que se utilice un clasificador para la identificación de los puntos de interés, es recomendable tomar alguna medida que minimice el riesgo de tomar algún elemento mal clasificado para calibrar la cámara.

Para reducir el riesgo arriba mencionado, se tomaron los resultados de varias pruebas de clasificación y se ordenó cada una de las clases de acuerdo a la precisión observada en los diferentes experimentos, otorgándole mayor prioridad a aquellas clases que hubieran sido detectadas con mayor frecuencia y precisión. De esta manera se pueden establecer límites mínimos y máximos en el número de puntos a utilizar en la calibración, ésto para cumplir con

el mínimo de puntos que el método Faugeras-Toscani requiere (8) y para evitar seleccionar aquellas clases con menor jerarquía, es decir, las más propensas a error.

# Capítulo 6

# Resultados experimentales

Con base en la aplicación propuesta en el capítulo anterior, se realizaron pruebas a cada uno de los componentes del modelo incluidos en la misma. Para la realización de estos experimentos, además del caballete ya descrito, se utilizó una cámara web marca Logitech, modelo QuickCam 4000, conectada a una computadora portátil con procesador AMD Turion 64 X2 a 2 GHz, con 2 GB de memoria RAM; Como se puede ver, se trata de un equipo de uso genérico, sin equipo especializado adicional. La figura 6.1 muestra una captura de pantalla (antes de realizar el aumentado) de la aplicación desarrollada para fines de prueba.

La interfaz de la aplicación (figura 6.1) se compone de un área donde se presenta el video aumentado (izquierda) y una área de información (derecha). La información mostrada consta de la imagen que se desea trazar (arriba) y datos que pueden ser relevantes tanto para el uso como la prueba, tales como el número de cuadros por segundo, la cantidad de puntos detectados, la cantidad de puntos que se utiliza en la calibración, entre otros.

En los apartados del presente capítulo se describirán los experimentos realizados y los

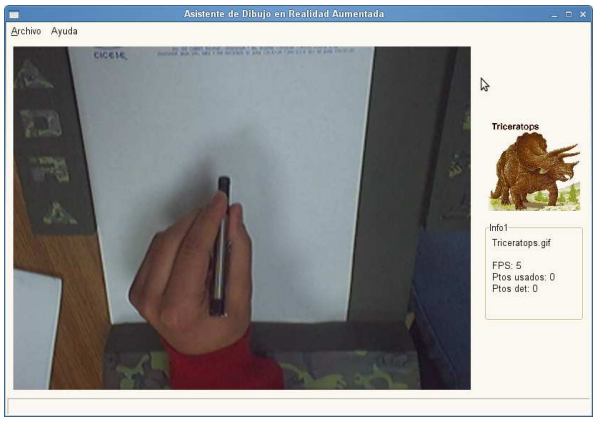



Figura 6.1: Captura de pantalla de la interfaz del ADRA

Figura 6.2: Pares de imágenes, original (izquierda) y el resultado de la detección de puntos SIFT (derecha)

resultados obtenidos en los mismos.

## 6.1. Detección y descripción de puntos de interés

Para realizar la detección y descripción de puntos de interés iniciales sobre el caballete se tomaron varias fotografías con resolución de 320x240 pixeles con diferentes ángulos, posiciones, orientaciones y condiciones de iluminación. Estas fotografías fueron procesadas por medio del detector y descriptor SIFT. En cuanto a los parámetros del detector y del descriptor, son los siguientes: se utilizó una pirámide gaussiana de 3 niveles, con un escalamiento de 3.0 en cada nivel y $\sigma = 1.6 * \sqrt{1/l}$, siendo $l = 3$ el número de niveles. Estos son los valores predeterminados de la implementación y los que dieron una detección más consistente y eficiente en la experimentación. La figura 6.2 muestra el resultado de la detección de puntos sobre dos imágenes de ejemplo.

El tiempo que el equipo arriba descrito tarda en realizar la detección y descripción de puntos en una imagen de 320 x 240 pixeles varía entre los 2.4 y 5.5 segundos, con un promedio de 3.4 segundos. Estos tiempos dependen de la cantidad de puntos detectados. Durante la experimentación dicha cantidad osciló entre 440 y 80 puntos con un promedio de 187 puntos detectados por imagen.

## 6.2. Clasificación y reconocimiento

De entre los descriptores de puntos de interés obtenidos se eligió un conjunto de aquellos correspondientes a los puntos que fueron más consistentemente detectados, etiquetándolos de manera distintiva, ver figura 6.4. Estos descriptores y sus etiquetas conformaron las muestras de entrenamiento, validación y prueba de un posterior proceso de clasificación. A continuación se presentan los detalles de la formación de los subconjuntos y los resultados del entrenamiento y clasificación con diferentes métodos de aprendizaje.

Se seleccionaron un total de 49 clases de puntos, 16 en el texto izquierdo, 20 en la franja

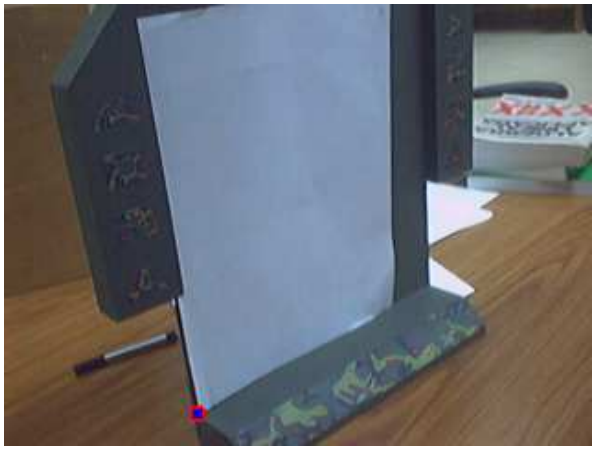


Figura 6.3: Fotografía del caballete sobre el que se colocará la hoja de papel para dibujar. Se marcan con puntos aquellas regiones de interés que se seleccionaron como clases para la identificación. El origen del mundo sería el punto marcado con un círculo grande

central y 13 en el texto derecho, ver figura 6.3. A continuación se muestra la manera en que se formaron los conjuntos de prueba, validación y entrenamiento; así como los resultados de la clasificación por diferentes métodos.

### 6.2.1. Conformación de los conjuntos para aprendizaje

El conjunto de datos total contiene 4140 descriptores de puntos ($T$), obtenidos de 60 imágenes y clasificados manualmente en 50 clases: 49 clases, con un total de 1088 miembros para las correspondientes a las de los puntos que se buscará identificar (ejemplos positivos) y una, con el resto de los miembros, para indicar la no pertenencia a ninguna de ellas (ejemplos negativos). De este conjunto $T$ se formaron tres subconjuntos, entrenamiento ($T_E$), validación($T_V$) y prueba($T_P$), primero cuidando que hubiera suficiente representación de los ejemplos positivos (al menos 5 miembros por clase) en el conjunto de entrenamiento y luego seleccionando los elementos aleatoriamente; con 80 %, 10 % y 10 % de los ejemplos positivos respectivamente y con una proporción aproximadamente similar en los ejemplos negativos.

También, tomando como base el conjunto original $T$, se conformaron 3 conjuntos, con 3 subconjuntos cada uno, para la identificación de los puntos por sectores de la imagen (izquierda, centro y derecha), es decir, se agruparon todos los puntos pertenecientes a la parte izquierda del caballete y con ellos se formaron subconjuntos de entrenamiento ($I_E$), validación ($I_V$) y prueba ($I_P$), con la proporción ya mencionada; lo mismo se hizo para las partes central ($C_E$, $C_V$, $C_P$) y derecha ($D_E$, $D_V$, $D_P$). Esto para poder contar con una manera de verificar (o mejorar), en tiempo de ejecución, la clasificación de los puntos detectados. Cabe mencionar que los conjuntos de entrenamiento $I_E$, $C_E$ y $D_E$ no son subconjuntos de $T_E$, aunque lo sean

| Conjunto | Tamaño | Elem. positivos |
|---|---|---|
| $T_E$ | 2437 | 879 |
| $T_V$ | 340 | 120 |
| $T_P$ | 326 | 89 |
| $I_E$ | 2133 | 33 |
| $I_V$ | 151 | 48 |
| $I_P$ | 140 | 40 |
| $C_E$ | 2164 | 23 |
| $C_V$ | 151 | 47 |
| $C_P$ | 140 | 41 |
| $D_E$ | 2001 | 23 |
| $D_V$ | 231 | 27 |
| $D_P$ | 220 | 23 |

Tabla 6.1: Distribución de los conjuntos de entrenamiento, validación y prueba para cada grupo

de $T$, de esta manera, los modelos entrenados de los primeros no necesariamente coincidirán con el modelo entrenado correspondiente a $T_E$. La tabla 6.1 detalla las características de los conjuntos antes mencionados:

En la figura 6.4 se puede ver los puntos que se han seleccionado por su repetición en las muestras tomadas. Entonces, en la parte izquierda ($I_E$, $I_V$, $I_P$) se concentran las clases 1 a 16, en la central ($C_E$, $C_V$, $C_P$) de la clase 17 a la 36 y en la derecha ($D_E$, $D_V$, $D_P$) de la 37 a la 49. Mientras que el conjunto $T_E$ contiene datos que pertenecen a las clases 1, ..., 43. Y todos los conjuntos contienen elementos de la clase 0, es decir ejemplos de descriptores que no pertenecen a ninguna clase de interés (ejemplos negativos).

Al graficar los valores de los descriptores de algunas clases (figura 6.5) se puede pensar que la simple distancia euclidiana serviría como método de identificación . Sin embargo existen otros conjuntos de descriptores en las clases seleccionadas que pueden no ser tan regulares (figura 6.6), incluso si se comparan los valores promedio por descriptor entre dos clases, puede ser que algunos lleguen a provocar una confusión si se toma la distancia euclidiana como único criterio de clasificación 6.7.

### 6.2.2. Resultados de la clasificación

Se probó la clasificación sobre los conjuntos de descriptores SIFT antes mencionados con tres métodos del estado del arte (Bayes normal, redes neuronales y máquinas de soporte vectorial). La comparación entre los resultados se realiza siguiendo el criterio de precisión y


Figura 6.4: Puntos de interés seleccionados para el proceso de entrenamiento y posterior identificación

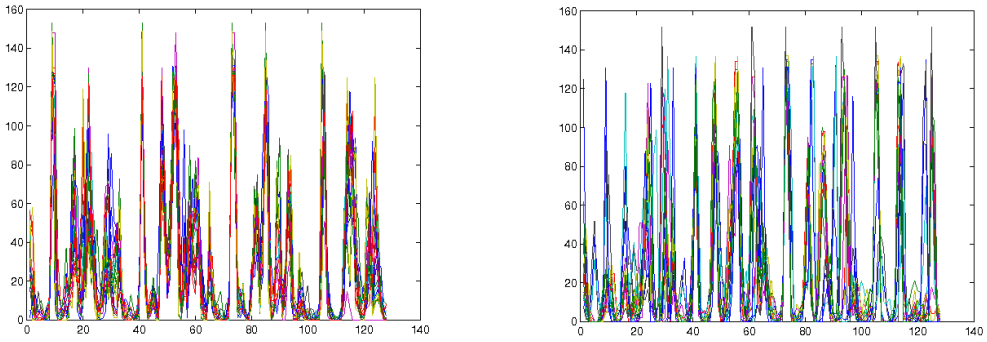


Figura 6.5: Gráfica de los valores de cada uno de los 128 descriptores de los puntos pertenecientes a la clase 11 (izquierda) con 24 objetos y a la clase 21 (derecha) con 16 objetos. El eje horizontal representa el número de descriptor y el vertical el valor asignado.

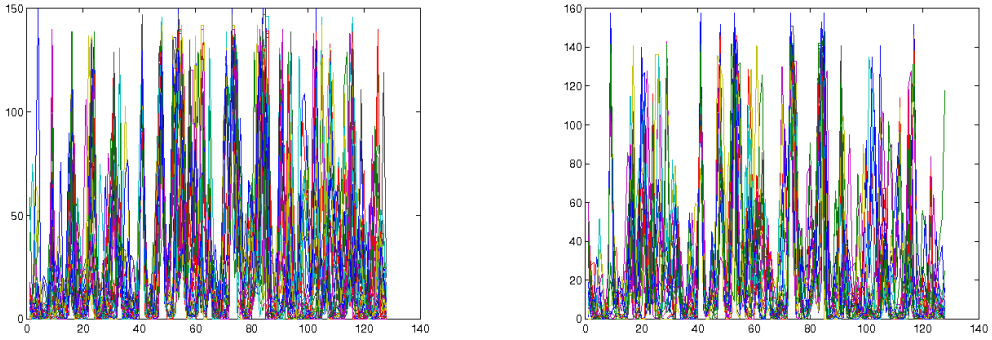


Figura 6.6: Gráfica de los valores de cada uno de los 128 descriptores de los puntos pertenecientes a la clase 5 (izquierda) con 43 objetos y a la clase 9 (derecha) con 30 objetos. Se puede observar fácilmente las diferencias de valor para un mismo descriptor entre los objetos de la clase.

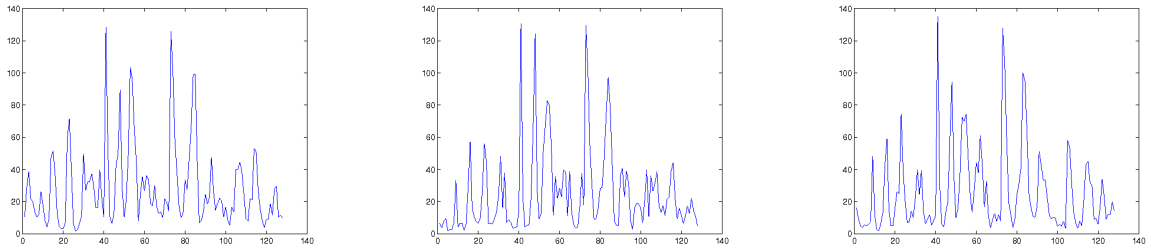


Figura 6.7: Promedios de los valores de los descriptores para la clase 7 (izquierda), la clase 5 (centro) y la clase 14 (derecha). Es posible observar que las diferencias entre los promedios de diferentes clases pueden ser incluso menores que entre miembros de una misma clase.

recuerdo[1], estos valores se definen como sigue:

$$P = \frac{\text{Aciertos en el conjunto de prueba}}{\text{Tamaño del conjunto de prueba}}$$

$$R = \frac{\text{Aciertos en el conjunto de entrenamiento}}{\text{Tamaño del conjunto de entrenamiento}}$$

Entonces, la precisión $P$ describe la capacidad del modelo de clasificar correctamente datos hasta entonces desconocidos obtenida por medio de la relación entre aciertos y el total de la muestra y el recuerdo $R$ mide la capacidad de clasificar correctamente los datos que ya conoce, el conjunto de entrenamiento. Además, en el caso particular de la aplicación que aquí se busca (identificar puntos para calibración de la cámara), se sugiere la utilización de penalizaciones en el caso de falsos resultados positivos (clasificar un elemento de la clase 0 en alguna otra) y de confusión de clases positivas (clasificar un elemento de una clase positiva(1 a 49) en otra positiva diferente). Entonces, se toma la fórmula de la precisión y, agregando las penalizaciones, se obtiene un valor de efectividad ($E$) de la clasificación:

$$E = \frac{A - 2\zeta - f}{n} \tag{6.1}$$

siendo $A$ el total de aciertos en la clasificación de la muestra, $\zeta$ el número de confusiones, $f$ el número de falsos positivos y $n$ el tamaño total de la muestra de prueba.

También se tomará en cuenta, para la selección de un método de clasificación específico, el índice de precisión en el subconjunto de los ejemplos positivos ($P_+$), es decir, la proporción de muestras positivas, con respecto al total de las mismas, que clasificó correctamente.

A continuación se muestran los resultados de efectividad ($E$), precisión ($P$) y recuerdo ($R$) obtenidos en la clasificación de los conjuntos de entrenamiento y validación-prueba ($T_V \cup T_P$, $I_V \cup I_P$, $C_V \cup C_P$, $D_V \cup D_P$).

### Clasificación usando Bayes normal

Se utilizó la implementación del clasificador Bayes normal incluida en el paquete de herramientas estadísticas de Matlab 2009 [53] la FDP de los datos (considerándolos continuos) se estimó por medio de un núcleo gaussiano, opción incluida en las rutinas de clasificación de Matlab.

En la tabla 6.2 se muestran los resultados de la clasificación de los cuatro conjuntos de datos con el método de Bayes normal. $R$ es el recuerdo del conjunto de entrenamiento, $P$ la

[1] En inglés: *Precision* y *Recall*

| **Conjunto** | $R$ | $P$ | $f$ | $\zeta$ | $P_+$ | $E$ |
|---|---|---|---|---|---|---|
| $T$ | 0.963 | 0.791 | 20 | 2 | 0.44 | 0.744 |
| $I$ | 0.984 | 0.832 | 1 | 7 | 0.45 | 0.780 |
| $C$ | 0.967 | 0.804 | 3 | 0 | 0.39 | 0.794 |
| $D$ | 0.994 | 0.850 | 1 | 2 | 0.26 | 0.830 |

Tabla 6.2: Resultados de la clasificación con Bayes normal

| **Conjunto** | $R$ | $P$ | $f$ | $\zeta$ | $P_+$ | $E$ |
|---|---|---|---|---|---|---|
| $T$ | 0.991 | 0.886 | 6 | 2 | 0.68 | 0.879 |
| $I$ | 0.993 | 0.942 | 0 | 2 | 0.81 | 0.928 |
| $C$ | 0.994 | 0.942 | 1 | 1 | 0.82 | 0.931 |
| $D$ | 0.990 | 0.933 | 0 | 1 | 0.48 | 0.925 |

Tabla 6.3: Resultados de la clasificación con redes neuronales

precisión al clasificar, $f$ la cantidad de falsos positivos sobre el conjunto de validación-prueba, $\zeta$ la cantidad de confusiones en el mismo conjunto, $P_+$ la precisión en las clases positivas y $E$ la medida de efectividad definida en la ecuación 6.1.

### Clasificación usando redes neuronales artificiales (RNA)

Para el desarrollo del software de clasificación con RNA, se utilizó la biblioteca de programación en C++ FANN[2] escrita por Nissen [62].

Se realizaron pruebas con varias configuraciones y varias funciones de activación en redes de retropropagación, los mejores se obtuvieron con una función de activación Elliot[23] en redes de 4 capas con 128 neuronas en la capa de entrada, 128 y 64 en las capas ocultas y en la capa de salida, 49 para los conjuntos $T$, 16 para los $I$, 20 para los $C$ y 13 para los $D$. Esto último para utilizar valores normalizados entre 0 y 1, tanto en entradas como en salidas. En la salida, la clase se define por el número de neurona que tenga el valor más cercano a 1 y por encima de cierto umbral (0.7 en este caso). En caso de que ningún valor de salida supere el umbral, la clase de salida se determina como 0.

La tabla 6.3 muestra los resultados obtenidos en la clasificación de los cuatro conjuntos ($T$, $I$, $C$, $D$).

### Clasificación usando maquinas de soporte vectorial (SVM)

Se utilizó la implementación hecha por Chang y Lin [18], libSVM. Utilizando como núcleo una función de base radial y como parámetros $\gamma = 0.5$ y $c = 8$. Los resultados de la clasificación

[2]Por *Fast Artificial Neural Network*

| **Conjunto** | $R$ | $P$ | $f$ | $\zeta$ | $P_+$ | $E$ |
|---|---|---|---|---|---|---|
| $T$ | 0.995 | 0.917 | 8 | 8 | 0.79 | 0.890 |
| $I$ | 0.998 | 0.952 | 0 | 1 | 0.84 | 0.945 |
| $C$ | 0.999 | 0.945 | 2 | 2 | 0.84 | 0.924 |
| $D$ | 0.997 | 0.960 | 0 | 0 | 0.80 | 0.960 |

Tabla 6.4: Resultados de la clasificación con máquinas de soporte vectorial

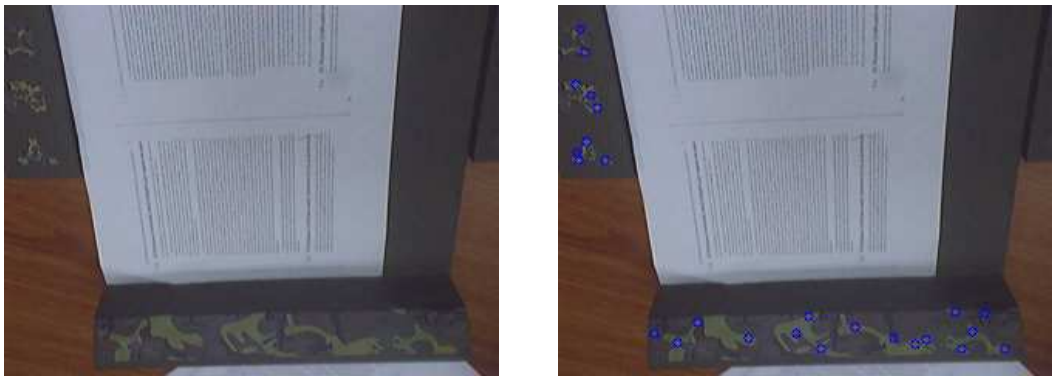


Figura 6.8: Ejemplos de imágenes utilizadas para la detección de puntos de interés. A la izquierda la imagen original, a la derecha la detección e identificación de puntos (marcados por círculos)

con este método se pueden ver en la tabla 6.4. Los valores para $\gamma$ y $c$ se obtuvieron mediante un método de búsqueda iterativo elaborado por Hsu, Chang y Lin [36] incluido en libSVM. Cabe mencionar que la metodología para la resolución de problemas multiclase que se sigue en libSVM es WTA_SVM.

Como los resultados arriba expuestos lo indican, la mejor clasificación se obtiene por medio de las SVM, por lo que se sugiere que sea éste el método utilizado para la clasificación de descriptores generados por SIFT. En las figuras 6.8 y 6.9 se puede ver el resultado de la detección y la identificación de puntos.

Para disminuir la posibilidad de falsos positivos y confusiones en la clasificación que aportará resultados a la calibración, se utilizó un sistema de clasificación por capas. La clasificación por capas se refiere a que se realiza más de una clasificación del mismo punto, con diferentes clasificadores (basados en máquinas de soporte vectorial), en este caso particular, los puntos se clasifican primero por medio del clasificador $T$; luego, dependiendo de la clase obtenida, se utiliza el clasificador $I$, $C$ o $D$. Si la segunda clasificación coincide con la primera el punto se toma como correcto.

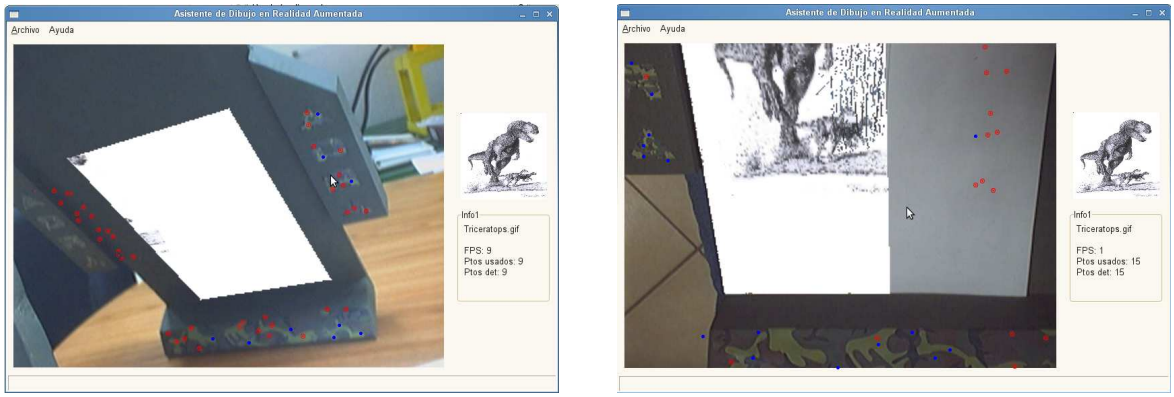
Figura 6.9: Ejemplos de imágenes utilizadas para la detección de puntos de interés. A la izquierda la imagen original, a la derecha la detección e identificación de puntos (marcados por círculos)

## 6.3. Calibración automática y colocación de objetos virtuales

Se establecieron límites mínimos y máximos en el número de puntos a utilizar en la calibración: el método de calibración Faugeras - Toscani requiere al menos seis correspondencias de puntos 2D y 3D, entonces, el límite mínimo se estableció en ocho puntos, para evitar operar con la mínima exactitud y el límite máximo se estableció en quince correspondencias, ya que según experimentos este número da resultados de calibración con la exactitud requerida sin requerir gran carga de procesamiento, además de que permite un gran margen de selección de puntos por prioridad de entre las 49 clases posibles.

Entonces, cuando se tiene el conjunto de puntos de la imagen identificados por el clasificador, éstos se ordenan de acuerdo a la prioridad de la clase. Si son ocho o más puntos, se procede con la calibración, si no, será necesario realizar la detección de puntos en un cuadro de video futuro. Luego de que se cumplió el límite mínimo, se eligen los 15 con más alta prioridad, en caso de ser más de 15, de lo contrario se eligen todos. Con los puntos elegidos se forman las correspondencias con sus equivalentes tridimensionales y se realiza la calibración. La figura 6.10 muestra un par de ejemplos donde, después de identificado un grupo de puntos y calibrada la cámara, se reproyecta el total de las clases, los puntos identificados se señalan con un círculo relleno y los no identificados con un doble círculo.

Se puede notar en la figura 6.10 la disminución en la precisión de los puntos no identificados con respecto a los identificados, sobre todo de aquellos fuera del área enmarcado por los círculos rellenos. Esto se debe a que el método de calibración se basa en un proceso aproximativo (optimización sujeta a restricciones) y sus resultados son precisos en el contexto de sus entradas, fuera de él puede o no serlo. En la figura 6.14 se puede ver un ejemplo de la




(a) (b)

Figura 6.10: Los puntos identificados se reproyectan en la imagen con la matriz de transformación obtenida (círculos rellenos), de la misma manera se proyectan los puntos 3D que no se identificaron (doble círculo)

colocación de objetos tridimensionales correctamente colocados sobre el caballete.

## 6.4. Superposición de objetos reales y virtuales

En el capítulo anterior se explicó el procedimiento para hacer transparente una región de la imagen que esté inscrita en un polígono determinado. A continuación se muestran algunas imágenes que son resultado de la aplicación de dicho método, en algunas de ellas se experimentó con la aplicación de transparencias graduales según un umbral, en otras se utilizó sólo un umbral mínimo y en otras simplemente se aplicó la transparencia total a los pixeles.

Las figuras 6.11, 6.12, 6.13 y 6.14 muestran algunos resultados obtenidos en la aplicación de la transparencia para simular la superposición.

El ADRA, realizando las tareas de calibración, proyección de los puntos, transparencia y mostrando el resultado en pantalla alcanza una tasa máxima de 14 cuadros de video por segundo, lo que la hace viable para operar en un ambiente real que no esté sujeto a movimientos bruscos de la cámara o del objeto de referencia.

## 6.5. Seguimiento de objetos

Aunque la aplicación propuesta, ADRA, no requiere que se realice el seguimiento de objetos, se realizó una prueba aparte del seguimiento de los puntos detectados con SIFT. Las imágenes en la figura 6.15 fueron extraídas de una secuencia de video donde se realizó la detección y descripción SIFT, luego se utiliza el algoritmo KLT (la implementación incluida en

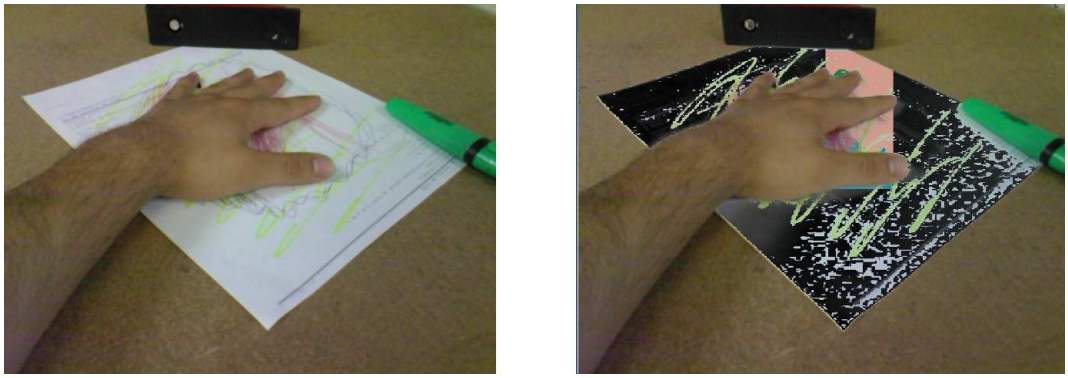


Figura 6.11: Se puede ver la imagen original a la izquierda, a la derecha el resultado de aplicar la transparencia al rectángulo formado por la hoja de papel. Dicha transparencia se aplicó de forma proporcional a la diferencia del tono de gris de cada pixel con un umbral predefinido.

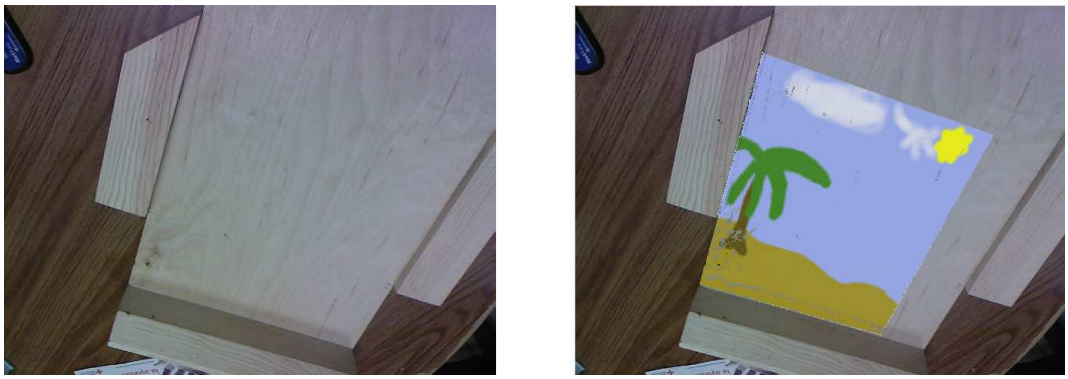
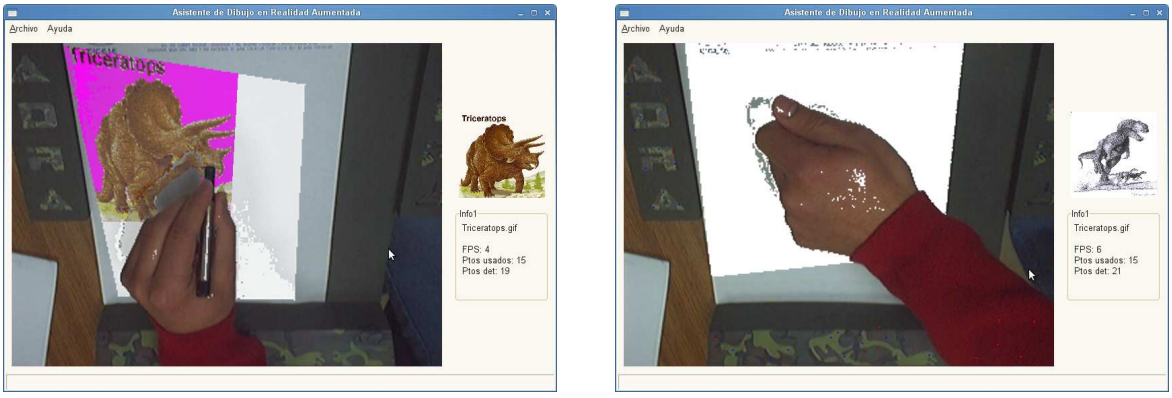


Figura 6.12: Después de calibrar la cámara se proyecta un plano virtual texturizado detrás de la pieza de madera haciendo transparente la zona de la imagen correspondiente, con esto da la impresión de que el plano virtual está delante de la pieza de madera.




(a) (b)

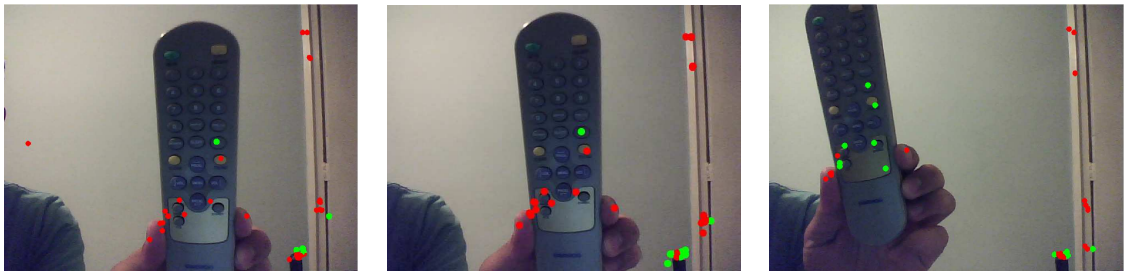
Figura 6.13: En estas imágenes se puede ver el efecto de superposición utilizando un umbral fijo para aplicar la transparencia.

(a) Imagen original

(b) Se agrega un plano texturizado

(c) Se agrega un cubo después del plano

Figura 6.14: Al igual que en la figura 6.12, se obtienen los puntos de interés y se calibra la cámara, para después agregar objetos virtuales con base en la posición del objeto real de referencia.



(a) (b) (c)

Figura 6.15: Seguimiento de puntos detectados con SIFT

OpenCV [13]) para dar seguimiento a los puntos. En la secuencia se puede ver que los puntos detectados (marcados con círculos) se mantienen en su posición con respecto a los objetos sobre los que fueron detectados originalmente.

# Capítulo 7

# Conclusiones y trabajo futuro

Según se ha mencionado antes, un sistema de realidad aumentada debe cumplir al menos con tres características para ser clasificado en esta área [6]:

- Combinar objetos virtuales con el mundo real
- Ser interactiva en tiempo real
- Los objetos virtuales se presentan en tres dimensiones

Una aplicación desarrollada con el modelo aquí presentado permitiría incrustar gráficos tridimensionales generados por computadora de una manera geométricamente realista (combina objetos reales y virtuales), la posición de dichos objetos virtuales se actualiza según los movimientos que se realizan con los objetos de la escena (interactivo en tiempo real), dicho movimiento puede ser en cualquier dirección posible en la escena (Los objetos se presentan en tres dimensiones). Con esto se puede decir claramente que el modelo propuesto es un modelo de aplicación de realidad aumentada.

El modelo detallado en capítulos anteriores basa su funcionamiento en la detección de puntos de interés para evitar el uso de marcadores predefinidos, estos puntos de interés se detectan y describen utilizando el algoritmo SIFT (Scale Invariant Feature Transform). Por medio de la identificación, hecha mediante máquinas de soporte vectorial, de un conjunto de tales puntos en la imagen cuya información tridimensional en la escena se conoce se realiza la calibración de la cámara y con ella el ajuste de la posición de los objetos tridimensionales en la escena. Finalmente, y para no requerir nuevamente la detección de puntos, se realiza el seguimiento de los puntos detectados (y de la proyección de los no detectados) para mantener actualizados los datos de calibración durante el tiempo que dure la ejecución del programa. En caso de que en algún momento el seguimiento de puntos sea insuficiente para la colocación de los objetos, es necesario volver a realizar la detección de puntos de interés. También se

incorpora un método sencillo de manejo de la superposición entre objetos reales y virtuales, particularmente la oclusión de la vista de un objeto virtual por uno real. Las oclusiones son un aspecto importante en un sistema de realidad aumentada, uno de los aspectos que distinguen a la realidad de las imágenes bidimensionales que se obtienen de ella es la posibilidad de que un objeto se encuentre detrás o delante de otro, y no siempre en el mismo plano; por lo tanto, un sistema de realidad aumentada deberá encontrar la manera de proporcionar al usuario esa sensación para mejorar la experiencia de uso.

Como se habrá podido observar, este modelo en su conjunto, presenta algunas ventajas ante otros enfoques de desarrollo de este tipo de aplicaciones. Entre esas ventajas, se puede destacar las siguientes:

**Independencia de marcadores predefinidos.** Si bien es cierto que la aplicación aún dependerá de la existencia de determinados objetos en la escena. El aprendizaje de puntos de interés resulta más conveniente que el uso de figuras predefinidas debido, en primer lugar, a que el objeto de referencia puede ser uno inherente a la escena y no necesariamente una placa bicolor que se inserta de manera premeditada. Luego, cuando se utilizan marcadores cuya estructura debe ser reconocida por el sistema, se tiene poca tolerancia a la oclusión parcial de los mismos, cosa que, como se ha demostrado en la experimentación anterior, sucede más bien poco si se utiliza un conjunto debidamente seleccionado de puntos de interés.

**Simulación de superposición.** Generalmente se piensa en sistemas de realidad hasta el punto en que los objetos virtuales se colocan sobre las imágenes de una manera realista, pocos son los intentos que se han hecho por lograr que, en la imagen resultante, los objetos virtuales puedan cubrir a los objetos reales y ser cubiertos por los mismos. El modelo propuesto plantea una opción sencilla y de bajo costo computacional para el manejo de las oclusiones entre objetos reales y virtuales.

**Bajos requisitos de hardware.** Se ha utilizado una computadora de escritorio para todas las etapas del procesamiento, el único sensor empleado para el cálculo del movimiento y localización de los objetos de interés es una cámara web. Con esto se evita el uso de sensores de movimiento, localización u otros que pueden resultar costosos o poco convenientes en ciertas condiciones.

Como aspectos menos favorables que se pueden mencionar respecto modelo, se tiene la necesidad de recolectar grandes cantidades de variantes del mismo punto para obtener una clasificación confiable, aunque esto redunda después en una mejor tolerancia a cambios en las condiciones ambientales de la escena, lo que da una mayor independencia a la aplicación.

También puede resultar poco favorable el hecho de que la ejecución del programa se detiene mientras se ejecuta SIFT, aunque en condiciones ideales este proceso se realizaría una vez cada que inicie la ejecución, posteriormente el algoritmo de seguimiento se encargaría de mantener actualizados los puntos de referencia.

En cuanto a los componentes y la forma específica en cómo se decidió resolver la problemática que cada uno representaba, se concluye lo siguiente:

Al respecto de la clasificación de los descriptores obtenidos por SIFT y de los métodos utilizados en la experimentación, se eligió finalmente SVM por los resultados en la clasificación. Aunque el método Bayes ingenuo ha mostrado buenos resultados en otro tipo de problemas, se pudo observar que para el caso de los descriptores SIFT y de las necesidades del modelo, incluso con estimación de la FDP, sus tasas de precisión difícilmente son útiles.

Las redes neuronales dieron resultados aceptables, pero presentan algunas desventajas, principalmente durante el entrenamiento; la forma en que se logró la mejor tasa de clasificación con redes neuronales fue utilizando una salida para cada clase posible, una entrada para cada elemento del descriptor y 2 o 3 capas ocultas de al menos la mitad de las neuronas de la capa de entrada cada una, esto vuelve el proceso de entrenamiento altamente costoso en cuanto a tiempo y en cuanto a recursos computacionales, tanto que en algunas implementaciones (matlab para Unix, por ejemplo) no se pudo ni siquiera procesar el entrenamiento por falta de memoria y en la implementación utilizada finalmente, el tiempo de entrenamiento usado para obtener valores de error aceptables fue desde varios minutos hasta horas. Además de que los parámetros como número de capas ocultas, tamaño de dichas capas, funciones de activación, método de entrenamiento, son factores que afectan grandemente el resultado y tienen que ser establecidos, en general, mediante prueba y error.

Máquinas de soporte vectorial (SVM) fue el método que obtuvo las mejores tasas de clasificación, además, su proceso de entrenamiento es bastante más rápido que el de las redes neuronales (segundos o pocos minutos). En cuanto a los parámetros para el entrenamiento, se requiere de una función núcleo y del establecimiento de dos valores $c$ o corte y $\gamma$. Estos valores también tienen un efecto importante en el clasificador resultante, para calcularlos ($c$ y $\gamma$) se utilizó una herramienta incluida en la implementación seleccionada cuyo gasto de recursos computacionales se puede comparar al entrenamiento de una red neuronal. Por medio de experimentación se determinó que la mejor función núcleo para el problema de la clasificación de descriptores SIFT es la función de base radial.

La calibración de la cámara (como se pudo ver en capítulos anteriores) es una operación de aproximación y cómo tal, la efectividad o inefectividad de su resultado depende de los datos

en que se base para realizar los cálculos, es decir, para que el resultado de la calibración sea confiable los puntos 3D y sus correspondientes 2D deben ser lo más exactos posibles. Además, al tratarse de una aproximación, se busca obtener un resultado que satisfaga los datos de entrada y por lo tanto, si la información de entrada es la mínima requerida, el resultado se ajustará a los datos proporcionados y difícilmente a otros fuera de ese rango. Entonces, para obtener una calibración confiable no sólo es necesaria la exactitud en los datos de entrada, también se requiere que la información se proporcione en la mayor cantidad posible.

En este trabajo, por medio de la clasificación por capas y la jerarquización de las clases, se incremente la calidad de la información proporcionada al método de calibración. El límite mínimo de correspondencias de puntos, establecido por encima del mínimo requerido por el método Faugeras - Toscani, permite que la falta de información sea un problema menos relevante. El límite máximo ayuda a mejorar la calidad de los datos, pero impide también mejorar la cantidad.

Uniendo la detección y descripción de puntos de interés con SIFT, la clasificación con SVM y el método de calibración Faugeras - Toscani se tiene un enfoque ante el problema de la calibración automática novedoso y competitivo con otros del estado del arte.

También se propone un método de simulación de superposiciones entre objetos reales y virtuales fácil de implementar en cualquier tipo de computadora, gracias a sus bajos requisitos de poder de procesamiento: la proyección de los puntos se hace mediante una multiplicación de matrices, el cálculo de la transparencia implica un par de multiplicaciones entre escalares y una resta por cada punto del polígono a procesar y el umbralado de una región limitada de la imagen. El grado de realismo proporcionado por este método depende de la exactitud y número de puntos del polígono, del método de umbralado y de la forma en que la transparencia que se utilice, que puede ser con un umbral duro, haciendo transparente todo nivel de intensidad superior o inferior al umbral o bien aplicando una transparencia proporcional a la diferencia entre el umbral y el nivel de intensidad a evaluar.

## 7.1. Trabajo futuro

Son variados los aspectos en que futuros trabajos de investigación pueden ayudar a mejorar el desempeño y las funcionalidades del modelo aquí propuesto.

Se sugiere la experimentación con diferentes algoritmos de clasificación, incluso se podría utilizar un método de agrupamiento no supervisado para seleccionar aquellos puntos que se buscarán en determinada escena. También puede resultar benéfico agregar los valores de

rotación generados por SIFT a los rasgos para el entrenamiento, se cree que esto podría mejorar sustancialmente la tasa de recuperación al agregar información de la orientación de un punto al momento de obtener un conjunto de descriptores específico. El incrementar el número y la variedad de las muestras también ayudará a hacer del reconocimiento un proceso más robusto.

En cuanto al seguimiento de objetos, en este documento se propone el método KLT, aunque se podrían probar otros, incluso se podría intentar el uso de clasificadores no supervisados para localizar el punto a seguir en la vecindad de su posición anterior.

Recientemente, en varios trabajos de investigación se ha buscado la implementación de algoritmos que requieren de gran poder de cómputo en las arquitecturas paralelas proporcionadas por los procesadores gráficos de algunas tarjetas de video. La detección y descripción de puntos de interés, e incluso la calibración podrían ser planteadas como procesos paralelos e implementadas en este tipo de arquitecturas con el fin de que se puedan ejecutar en tiempo real sin disminuir el rendimiento de otros procesos. De lograrse esto, se podría prescindir de cualquier tipo de seguimiento posterior en la escena, ya sea por hardware o por software, ya que se podría obtener de primera mano la ubicación de las regiones de interés por medio de una detección, descripción, e identificación en tiempo real.

Otro aspecto en el que el modelo podría crecer, sería en la adición de un conjunto de objetos que permitan la creación de interfaces de usuario de propósito general basadas en realidad aumentada, esto es, agregar al video objetos virtuales que reaccionen ante ciertos gestos del usuario y le permitan realizar tareas genéricas como el manejo de archivos (abrir, cerrar, borrar, etc), dar instrucciones a la aplicación, incluso el manejo de la misma cámara y de la manera como se presenta la información al usuario.

# Glosario

| | |
|---|---|
| Aprendizaje de máquina | En inglés, *machine learning*, rama de la inteligencia artificial que trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos |
| Calibración | Hablando de una cámara, calibrarla consiste en obtener los valores las variables que afectan la manera en cómo los objetos son proyectados sobre la imagen (intrínsecas) y las referentes al movimiento realizado para capturar una imagen específica (extrínsecos). |
| Entrenamiento | En aprendizaje de máquina, es el proceso por el cual se da la información de los ejemplos al método elegido y se inicia el proceso de aprendizaje con base en ellos. |
| Estereoscópico | Se dice de cualquier técnica de grabación de la información tridimensional visual o a la creación de la ilusión de profundidad en una imagen. Generalmente se hace utilizando dos cámaras que captan la misma escena desde ángulos distintos a la vez. |

| | |
|---|---|
| Fotograma | En una secuencia de video, un fotograma es una fotografía fija de las muchas que componen la secuencia. |
| GPS | Sistema de Posicionamiento Global, tecnología que, valiéndose de las comunicaciones satelitales, permite ubicar la posición de un dispositivo en cualquier punto de la superficie del planeta. |
| Háptico | Del griego *Haphe* relativo a la percepción táctil (interfaces hápticas). |
| Interfaz | Conexión física y funcional entre dos aparatos o sistemas independientes. |
| Marcador | En visión computacional, es un objeto generalmente plano y bicolor (blanco y negro) con trazos sencillos que pueden ser reconocidos por la computadora. Se utilizan como guías para el análisis de video. |
| Máquina de soporte vectorial (SVM) | Técnica de aprendizaje de máquina que basa su funcionamiento en el cálculo de los llamado vectores de soporte. |
| OpenCV | *Open Computer Vision library* conjunto de herramientas de programación especializado en el procesamiento de imágenes y la visión computacional. |
| OpenGL | Del inglés *Open Graphics Library*, es una interfaz de programación que permite el manejo y la presentación en pantalla de objetos tridimensionales. |

| | |
|---|---|
| Parámetros extrínsecos | Indican el movimiento (rotación y traslación) que sufrió una cámara hasta llegar a la posición en donde capturó una imagen. |
| Parámetros intrínsecos | Son los que especifican como realiza la proyección de los objetos sobre la imagen una cámara. |
| Punto de interés | Son aquellos que, en una imagen, proporcionan mayor cantidad de información distintiva sobre la misma. Por ejemplo una esquina |
| Realidad aumentada | (RA) es una variación de los *ambientes virtuales* o *realidad mezclada*, consiste en agregar objetos virtuales tridimensionales a secuencias de video de escenas reales. |
| Redes neuronales | Técnica del aprendizaje de máquina cuyo funcionamiento está inspirado en las redes que forman las células cerebrales (neuronas) de los animales. |
| Seguimiento | En visión computacional, el seguimiento o *tracking* consiste en, dado un objeto en movimiento en una secuencia de video, identificar el movimiento del objeto, de manera que en cada fotograma se conozca su localización . |
| SIFT | *Scale Invariant Feature Transform.* Transformación de características invariantes a escala. Método de detección y descripción de puntos de interés propuesto por Lowe en 1999 |
| Superposición | De superponer, poner por encima de. Es el hecho de que un objeto esté encima (o antes, en el orden de visibilidad) de otro. |

| | |
|---|---|
| Tiempo real | Se dice que una aplicación es de tiempo real cuando ésta requiere que un programa responda a estímulos dentro de un límite de tiempo muy pequeño (usualmente milisegundos o microsegundos). |
| Virtual | Que tiene existencia aparente, no real. |

# Bibliografía


[1] Augmented reality toolkit homepage.

[2] H. Asada and M. Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8, 1986.

[3] Dave Astle and Kevin Hawkins. *Beginning OpenGL Game Programming*. Premier Press, 2004.

[4] Autor. *Title*. Publisher, Year.

[5] Shai Avidan. Support vector tracking. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 184–191, 2001.

[6] Ronald Azuma. A survey on augmented reality. *Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[7] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steve Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, December 2001.

[8] Michael Bajura and Ulrich Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60, 1995.

[9] M. O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *In Proceedings of CVPR (IEEE Conference on Computer Vision and Pattern Recognition), Puerto Rico*, pages 91–96, 1997.

[10] M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook - moving seamlessly between reality and virtuality. *Computer Graphics and Applications, IEEE*, 21(3):6–8, 2001.

[11] Kai bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 278–285, 2005.

[12] Remco R. Bouckaert. Naïve bayes classifiers that perform well with continuous variables. In *Proceedings of the 17th Australian Conference on AI (AI 04)*. Springer, 2004.

[13] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[14] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, October 2008.

[15] T J Broida and R Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):90–99, 1986.

[16] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[17] Tom Caudell and D. W. Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Hawaii Int'l Conference on System Scieneces*, pages 659–669, 1992.

[18] Chih C. Chang and Chih J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[19] Denis Chekhlov, Andrew Gee, Andrew Calway, and Walterio Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, November 2007.

[20] Enylton Machado Coelho, Blair MacIntyre, and Simon J. Julier. Osgar: A scene graph with uncertain transformations. *Mixed and Augmented Reality, IEEE / ACM International Symposium on*, 0:6–15, 2004.

[21] Enylton Machado Cohelo. *Spatially Adaptive Augmented Reality*. PhD thesis, Georgia Institute of Technology, 2005.

[22] Jonathan Deutscher, Michael Isard, and John Maccormick. Automatic camera calibration from a single manhattan image. In *Eur. Conf. on Computer Vision (ECCV*, pages 175–205, 2002.

[23] D.L. Elliott. A better activation function for artificial neural networks. Technical Report 93-8, Institute for Systems Research, Enero 1993.

[24] Olivier Faugeras. *Three-Dimensional Computer Vision (Artificial Intelligence)*. The MIT Press, November 1993.

[25] Olivier Faugeras and G. Toscani. The calibration problem for stereo. In IEEE Computer Society Press, editor, *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 15–20, June 22-26 1986.

[26] Steve Feiner, Blair MacIntyre, Tobias Höellerer, and Anthony Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. Cambridge, MA., October 1997.

[27] M. Fiala and C. Shu. Fully automatic camera calibration using self-identifying calibration targets. Technical Report NRC-48306 ERB-1130, National Research Council of Canada, 2005.

[28] L. Floarck, B. Haar, J. Koenderink, and M. Viergever. General intensity transformations and second order invariants. In *Proceedings of 7h Scandinavian Conference on Image Analysis*, pages 338–345, 1991.

[29] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centers of circular features. In *ISPRS intercomission conference on fast processing of photogrametric data*, pages 149–155, 1987.

[30] J. Frahm, K. Koeser, D. Grest, and R. Koch. Markerless augmented reality with light source estimation for direct illumination. pages 211–220, 2005.

[31] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. on Machine Intelligence*, 13(9):891–906, 1991.

[32] D. Gabor. Theory of communication. *Journal I. E. E. E.*, 3(93):429–457, 1946.

[33] Iryna Gordon and David Lowe. What and where: 3d object recognition with accurate pose. *Toward Category-Level Object Recognition*, pages 67–82, 2006.

[34] Ismail Haritaoglu, Davis Harwood, and Larry S. David. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):809–830, August 2000.

[35] Chris Harris and Mike Stephens. A combined corner and edge detector. pages 147–151, 1988.

[36] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Taipei, 2003.

[37] Charles E. Hughes, Christopher B. Stapleton, Darin E. Hug, and Eileen M. Smith. Mixed reality in education, entertainment, and training. *IEEE Computer Graphics and Applications*, pages 24–30, December 2005.

[38] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003.

[39] A. Johnson and M. Hebert. Object recognition by matching oriented points. In *Proceedings of the Conference on Computer Vision and pattern recognition*, pages 684–689, 1997.

[40] M. Carmen Juan, Mariano Alcañiz, Carlos Monserrat, Cristina Botella, Rosa M. Baños, and Belen Guerrero. Using augmented reality to treat phobias. *IEEE Computer Graphics and Applications*, pages 31–37, December 2005.

[41] Jinman Kang, Isaac Cohen, and Gérard Medioni. Multi-views tracking within and across uncalibrated camera streams. In *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, pages 21–33, New York, NY, USA, 2003. ACM.

[42] H. Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, San Francisco, USA, October 1999.

[43] Mathias Koelsch, Ryan Bane, Tobias Hoellerer, and Matthew Turk. Multimodal interaction with a wearable augmented reality system. *IEEE Comput. Graph. Appl.*, 26(3):62–71, 2006.

[44] J. J. Koenderink and A. J. van Doom. Representation of local geometry in the visual system. *Biol. Cybern.*, 55(6):367–375, March 1987.

[45] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.*, volume 1, pages 18–25 vol. 1, 2005.

[46] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Sparse texture representation using affine-invariant neighborhoods, 2003.

[47] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *J. of Applied Statistics*, 21(2):224–270, 1994.

[48] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, Washington, DC, USA, 1999. IEEE Computer Society.

[49] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[50] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

[51] B. Macintyre, M. Gandy, J. Bolter, S. Dow, and B. Hannigan. Dart: the designer's augmented reality toolkit. pages 329–330, 2003.

[52] David Marr. *Vision: A computational investigation into the human representation and processing of visual information.* W.H. Freeman, July 1982.

[53] MathWorks. *Statistics Toolbox For use with Matlab® - User's Guide, Ver. 5.0.2*, 2009. Cited By (since 1996): 1Export Date: 16 March 2009Source: Scopus.

[54] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local features for object class recognition. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1792–1799, Washington, DC, USA, 2005. IEEE Computer Society.

[55] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[56] Tom M. Mitchell. *Machine Learning.* McGraw-Hill Science/Engineering/Math, March 1997.

[57] Hans Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, August 1977.

[58] Jurriaan D. Mulder. Realistic occlusion effects in mirror-based co-located augmented reality systems. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 203–208, 326, Washington, DC, USA, 2005. IEEE Computer Society.

[59] M. E. Munich, P. Pirjanian, E. Di Bernardo, L. Goncalves, N. Karlsson, and D. Lowe. Sift-ing through features with vipr. *Robotics & Automation Magazine, IEEE*, 13(3):72–77, 2006.

[60] Jim Mutch and David Lowe. Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 2008.

[61] Nassir Navab. Scene augmentation via the fusion of industrial drawings and uncalibrated images with a view to markerless calibration. In *Proceedings of the 2nd International Workshop on Augmented Reality*, pages 125–133. IEEE CS Press, 1999.

[62] Steffen Nissen. Implementation of a fast artificial neural network library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU)*, 31, 2003.

[63] Gustavo Olague and B. Hernández. A new accurate and flexible model based multi-corner detector for measurement and recognition. *Pattern Recognition Letters*, 26(1):27–41, 2005.

[64] Juri Platonov, Hauke Heibel, Peter Meier, and Bert Grollmann. A mobile markerless ar system for maintenance and repair. In *ISMAR '06: Proceedings of the 2006 Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*, pages 105–108, Washington, DC, USA, 2006. IEEE Computer Society.

[65] Juri Platonov and Marion Langer. Automatic contour model creation out of polygonal cad models for markerless augmented reality. In *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.

[66] Bernhard Reitinger, Christopher Zach, and Dieter Schmalstieg. Augmented reality scouting for interactive 3d reconstruction. In William R. Sherman, Ming Lin, and Anthony Steed, editors, *VR*, pages 219–222. IEEE Computer Society, 2007.

[67] Anderson André Genro Alves Ribeiro, Leandro Lorenzett Dihl, and Cláudio Rosito Jung. Automatic camera calibration for driver assistance systems. In *Proceeedings of 13th International Conference on Systems, Signals and Image Processing*, pages 173 —- 176, 2006.

[68] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.

[69] Lawrence Rosenblum. Virtual and augmented reality 2020. *IEEE Comput. Graph. Appl.*, 20(1):38–39, 2000.

[70] S. J. Russell and Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.

[71] Koichi Sato and J. K. Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vis. Image Underst.*, 96(2):100–128, 2004.

[72] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[73] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[74] Andrei State, Mark A. Livingston, Gentaro Hirota, , William F. Garrett, Mary C. Whitton, Henry Fuchs, and Etta D. Pisano. Pose tracking from natural features on mobile phones. In *Proceedings of SIGGRAPH '96*, pages 125–134, 1996.

[75] R. L. Streit and T. E. Luginbhul. Maximum likelihood method for probabilistic multihypothesis tracking. In *In Proceedings of the International Society for Optical Engineering (SPIE.)*, pages 394–405, 1994.

[76] Flávio Szenberg, Paulo Cezar Pinto Carvalho, and Marcelo Gattass. Automatic camera calibration for image sequences of a football match. In *ICAPR '01: Proceedings of the Second International Conference on Advances in Pattern Recognition*, pages 301–310, London, UK, 2001. Springer-Verlag.

[77] Bruce Thomas, Ben Close, John Donoghue, John Squires, Phillip De Bondi, and Wayne Piekarski. First person indoor/outdoor augmented reality application: Arquake. *Personal and Ubiquitous Computing*, 6:75–86, 2002.

[78] Leonardo Trujillo-Reyes. *Cómputo evolutivo aplicado en el diseño de métodos para la detección y descripción de regiones de interés*. PhD thesis, Centro de Investigación Científica y Estudios Superiores de Ensenada, B.C., 2008.

[79] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):929–944, 2007.

[80] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/ photometric invariants for planar intensity patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 642–651, London, UK, 1996. Springer-Verlag.

[81] A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.

[82] Martin Vetterli and Jelena Kovacevic. *Wavelets and Subband Coding*. Prentice Hall PTR, April 1995.

[83] Brian Wanstall. Hud on the head for combat pilots. *Interavia*, (44):334–338, 1989.

[84] Matthias M. Wloka and Brian G. Anderson. Resolving occlusion in augmented reality. In *in Symposium on Interactive 3D Graphics Proceedings*, pages 5–12, 1995.

[85] Ke Xu, Kar Wee Chia, and Adrian David Cheok. Real-time camera tracking for marker-less and unprepared augmented reality environments. *Image Vision Comput.*, 26(5):673–689, 2008.

[86] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[87] Chunrong Yuan. Markerless pose tracking for augmented reality. In *ISVC (1)*, pages 721–730, 2006.