Desarrollo de un sistema de Realidad Aumentada que incluya reconocimiento de gestos

por

José López Bolós

Proyecto Final de Carrera asociado a

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

Directores: Roberto Paredes Palacios

M. Carmen Juan Lizandra

Valencia, Noviembre 2010

Índice

Capítulo 1

Introducción

En el mundo de la informática, ha ido ganando cada vez más importancia la interacción entre el usuario y el computador. Desde los años en los que únicamente se usaba el teclado, la posterior implantación del ratón, trackballs, joysticks, dispositivos hápticos, etc., se ha buscado que ese proceso de interacción fuese lo más natural posible para el usuario. Por lo tanto, un paso lógico en la evolución en que nos comunicamos con una máquina es aproximarse a una de las formas que los humanos usan entre ellos para hacerse entender: el lenguaje gestual. Conseguir dar este paso de una forma robusta y que a la vez sea simple puede abrir un nuevo camino a la comunicación no sólo para aplicaciones desde un punto de vista tradicional de la informática, ya que se podría extender a campos tan variados como la robótica, a la domótica o a la gestión. Se eliminaría un intermediario físico entre el ordenador y el usuario, dándole a este último más libertad para realizar acciones, que con un dispositivo de control "inutilizando" la mano no serían posibles. Además, si se compara con, por ejemplo, el ratón que funciona sobre un plano de dos dimensiones, se ganaría la tercera dimensión que es la profundidad.

Mediante el uso de técnicas de Visión por Computador y Reconocimiento de Formas, se intenta que el ordenador identifique la mano de un usuario y sea capaz de mostrar una respuesta. Para ser más precisos, el computador reaccionará a la mano del usuario y éste visualizará en tiempo real tanto su entorno como la respuesta generada, gracias a técnicas de Realidad Aumentada (RA). Para eso, se ha desarrollado una aplicación y una librería de funciones que reconocen la mano del usuario a través del color y la forma, y muestra un eje de coordenadas virtual sobre la palma o el revés de la mano. Inicialmente el gesto a reconocer será la mano extendida.

Se podría decir que en este proyecto la mano es la protagonista, ya que funcionará tanto de elemento de interacción como de marcador. Un marcador es un punto de referencia espacial de la realidad física para el ordenador sobre el que se basará para introducir o procesar la información digital. Hasta ahora y debido a que es una forma fácil de reconocer, para un programa de RA, se suele usar un marcador con un cuadrado negro sobre fondo blanco. Este tipo de seguimiento es muy invasivo, para evitarlo y hacer más natural la comunicación se ha establecido como objetivo el desarrollar las herramientas necesarias para un sistema de RA sin marcadores.

El objetivo principal del proyecto es realizar un sistema de RA que detecte la mano en cualquier posición, distinguiéndola de la escena. Una vez hecho esto el

sistema debe saber en qué posición está y que reconozca el gesto básico de la mano extendida y todo debe funcionar en tiempo real.

Como objetivos secundarios se plantea:

- Creación de una librería de las funciones que acelere el desarrollo de futuros programas.
- Un sistema que reconozca más de un gesto.
- Conseguir un sistema lo más robusto posible a la hora de reconocer la mano contra problemas lumínicos.

Capítulo 2

Estado del Arte

2.1. Realidad Aumentada

2.1.1. Definición

La tecnología de *RA* nos presenta la realidad y le añade información artificial. Generalmente esto es una captura de imágenes por un dispositivo de vídeo (ej.: una webcam) y tras cálculos varios, la implantación sobre la imagen de cierta información que puede ir desde figuras 2D y 3D, una secuencia animada de video, texto estático o dinámico, etc. Idealmente, el usuario debe tener la sensación de que los objetos virtuales y los objetos reales coexisten en el mismo espacio, es decir, sin distinguir la diferencia entre los objetos reales y los virtuales.

Los sistemas de RA requieren el uso de unas tecnologías y dispositivos de visualización para aprovechar al máximo sus posibilidades. Una definición de RA comúnmente aceptada es la siguiente, un sistema de RA cumple [AZU97]:

- Combinación de imagen real y virtual.
- Interacción en tiempo real.
- Localización 3D.

Dentro de esta definición pueden englobarse diversas tecnologías para los sistemas de RA, acepta interfaces basadas en monitor, sistemas monoculares, *seethrough* Head-Mounted Displays, y otras diversas tecnologías combinadas.

El objetivo final ideal es crear un sistema donde el usuario no pueda notar la diferencia entre ver el mundo real y ese mismo mundo pero complementado con elementos virtuales.

Un usuario necesita que el sistema responda en tiempo real para poder interactuar con el sistema de forma efectiva. Un sistema de RA complementa el mundo real siendo necesario que el usuario mantenga el sentido de presencia en ese mundo. Las imágenes virtuales se mezclan con la vista real para crear el sistema de RA. Los objetos virtuales generados por ordenador deben estar ajustados con el mundo real en todas las dimensiones. Si existen errores en el ajuste, el usuario no tendrá la percepción de ver ambas imágenes, virtual y real, fusionadas. Además, el ajuste de las imágenes debe ser correcto en todo momento, incluso cuando el usuario se esté moviendo, los cambios en la visión debidos al movimiento se deben tener en cuenta y realizar las operaciones oportunas para la situación de los objetos virtuales. Discrepancias o cambios en estos ajustes harán el trabajo con la vista de la RA mucho

más difícil, llegando al punto de ser molesto para el usuario y haciendo que el sistema sea inutilizable. Se pueden dar errores a nivel visual o incluso de percepción, pero son los errores entre la fusión de la imagen real y la virtual a los que el usuario es más sensible y en los que una aplicación de RA se debe centrar principalmente.

Milgram describe una taxonomía de cómo la RA y la Realidad Virtual se relacionan (Figura 1) [MIL94].

Figura 1: Continuo Realidad-Virtualidad de Milgram

El mundo real y un mundo totalmente virtual son los dos extremos de esta continuidad en cuyo punto intermedio se encuentra lo que Milgram denomina Realidad Mezclada. La RA la encontramos cerca del extremo del entorno real, siendo el mundo real complementado con datos generados por ordenador. La *Virtualidad Aumentada* es un término creado por Milgram para identificar sistemas que son principalmente sintéticos pero que agregan ciertas imágenes del mundo real como vídeos y texturas sobre objetos virtuales.

2.1.2. Aplicaciones

El uso de los sistemas de RA se ha extendido a muchos sectores en los últimos años. Cada vez es más evidente para los grupos de investigación, tanto en el ámbito docente como en el ámbito industrial, de la potencia que posee la RA para transmitir de forma visual una información digital que potencie o sustituya parcialmente el entorno percibido. Esto ha sido potenciado gracias a que las capacidades de procesamiento de imagen y de vídeo en tiempo real son cada vez más mayores y que van apareciendo nuevas posibilidades en los sistemas gráficos digitales se consigue cada vez efectos más realistas en los sistemas de RA y esto permite reducir la sensación de artificialidad de los elementos virtuales añadidos.

Algunos de los sectores que se han hecho eco de la aplicabilidad de la RA, que se mencionan en este capítulo.

2.1.2.1. Medicina

Los médicos pueden usar la RA como una ayuda a las prácticas durante la formación. También puede ser de ayuda a cirujanos durante la preparación de una operación o durante la misma. Utilizando estos sistemas, el tiempo de la intervención se reduce, consiguiéndose así un mayor beneficio y menor riesgo para el paciente. Como se puede ver en la figura 2 [NAK08].

Figura 2: Visualización aumentada de una nefrectomía parcial laparoscópica.

2.1.2.2. Psicología

En psicología también se han desarrollado aplicaciones de RA. Por ejemplo para el tratamiento de fobia a los animales pequeños (figura 3 y 4) [JUA05] o para la acrofobia, es decir fobia a las alturas [JUA06][JUA10].

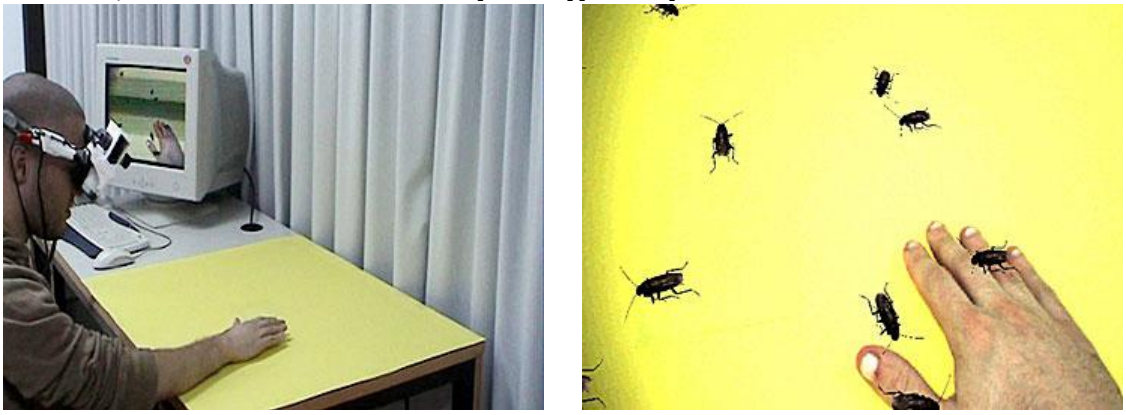


Figura 3 y 4: Sistema contra fobia a los animales pequeños con marcadores invisbles.

2.1.2.3. Fabricación: mantenimiento y reparación

Otra categoría donde se pueden aplicar las posibilidades de la RA es en el montaje, mantenimiento y reparación de maquinaria compleja. Por ejemplo un operario que necesite realizar una labor de montaje o mantenimiento puede tener con un sistema de RA un manual visual que le indique cómo montar y desmotar una pieza, además de permitirle tener las manos mucho más libres que si tuviese que mirar manuales de texto tanto si son en papel como en un formato digital a través de un dispositivo móvil. La figura 5 muestra un ejemplo de cómo BMW está aplicando RA a su cadena de montaje [http://www.bmw.com/com/en/owners/service/augmented_reality_workshop_1.html].

Figura 5: Mantenimiento de un BWM.

2.1.2.4. Anotación y visualización

También podemos usar los sistemas de RA para comentar objetos y entornos ya sea con información pública o privada. Por ejemplo, un usuario podría ir en su coche con un sistema de visualización integrado sobre el parabrisas por una carretera y éste proporcionarle información acerca de los elementos de señalización conforme el usuario avanza o la pide, o bien señalándole dónde se encuentra un posible peligro en concreto. Esto podría ser especialmente aconsejable en días de escasa visibilidad. General Motors está desarrollando un sistema de estas características [http://www.youtube.com/watch?v=J0xn1BA4mQc]. Esto puede aplicarse también al reconocimiento de la posición del usuario, haciendo que las anotaciones aparezcan si está cerca el usuario, o bien le sigan de tal forma que aparezcan siempre delante de su visión. Podemos conseguir este efecto mediante un dispositivo de rastreo que se colocaría el usuario y con lo que el ordenador conocería su localización. Esto permite que la información siempre aparezca a la vista del usuario aunque éste cambie de orientación o de posición con respecto al objeto (figuras 6 y 7).

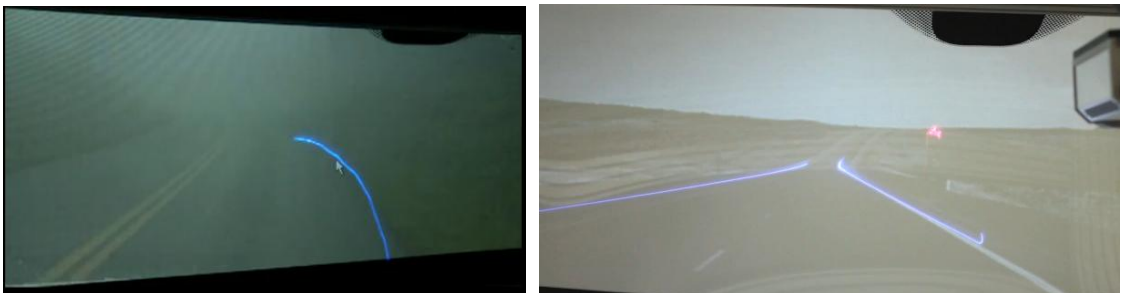


Figura 6 y 7: Prototipo de visualización en el parabrisas de GM.

2.1.2.5. Entretenimiento

Una forma simple de RA se viene usando ya hace tiempo en el entretenimiento y en el mundo de las noticias en televisión. En cualquier noticiario de televisión, la información meteorológica se muestra mediante un presentador junto con las imágenes por satélite y los mapas del tiempo cambiando tras de él.

También se emplea este método últimamente en la televisión para generar platos virtuales, colocan el presentador con una silla y una mesa reales y el resto es virtual. A esta técnica se le conoce con el nombre de *Chroma Key*. Recientemente se han desarrollado otros proyectos, dónde ya no es necesario que exista esta pantalla de fondo verde o azul. Se usa la temperatura del cuerpo de la persona para mantener la imagen de la persona y luego cambiar el fondo (Figura 8) [ART03].

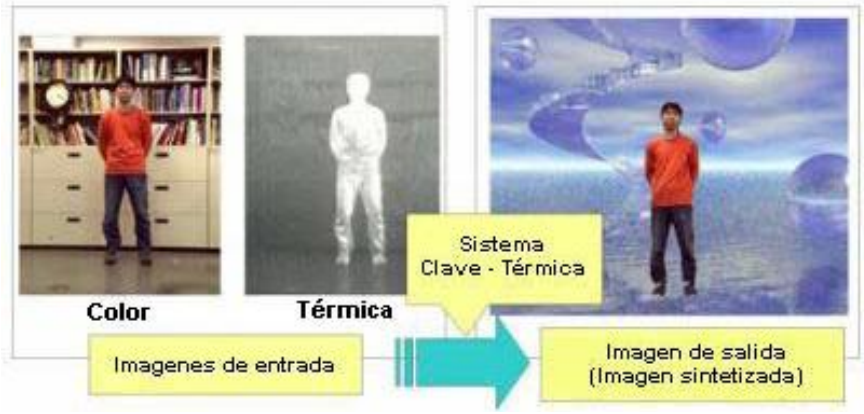


Figura 8: Se genera un fondo virtual detrás del actor

Por supuesto, el campo del ocio ofrece un amplio abanico de posibilidades dónde aplicar los sistemas de RA. Varios grupos y empresas están investigando y desarrollando juegos dentro de este campo. Existen una infinidad de posibilidades, desde juegos de mesa hasta simulaciones de juegos de combate, por ejemplo Invizimals (figura 9) [NOV10].

Figura 9: Juego Invizimals para PSP de Novarama

2.1.3. Aprendizaje

Para aprendizaje se han desarrollado sistemas para distintas materias. Por ejemplo, para contar historias interactivas [JUA04], conocer animales exóticos[JUA09] o en peligro de extinción [JUA10]. Aprender a deletrear palabras [JUA10b] o a reciclar (figura 10) [JUA11].

Figura 10: RA aplicado al aprendizaje del reciclaje con dispositivos móviles.

2.1.4. Aeronáutica y entrenamiento militar

Durante muchos años, la aeronáutica militar y los helicópteros han usado visores en la cabeza (*Head-Up Displays*) y en los cascos (*Helmet-Mounted Sights*) para superponer vectores gráficos en la visión del mundo real.

Además suelen proporcionar también información básica de vuelo y navegación, estas imágenes son también utilizadas en ocasiones a la hora de marcar el objetivo de las armas.

Además de la aplicación en aeronáutica, los sistemas de RA también pueden aplicarse para entrenamiento, tanto en la simulación del manejo de distintos aparatos como en el entrenamiento del personal militar en situaciones de combate (figura 11).

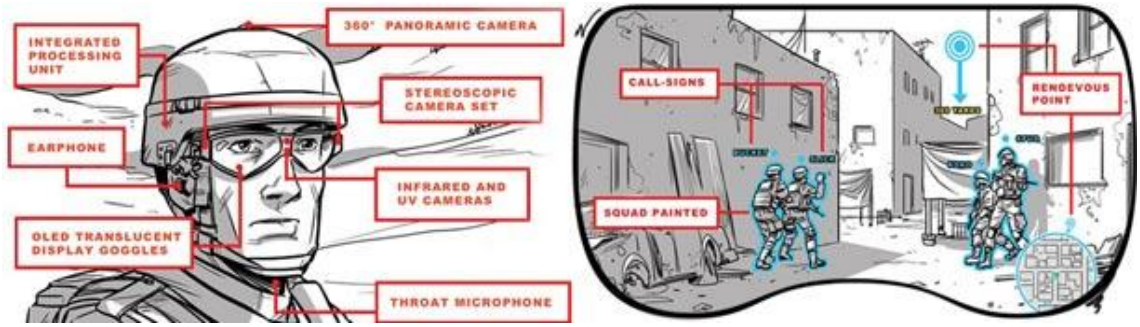


Figura 11: Diseño conceptual de un posible uso de la RA en el ámbito militar.

2.1.5. Diseño comercial y de interiores

Los sistemas de RA se usan ya para el diseño comercial. Por ejemplo, si vamos a cambiar algún mueble de casa, puedes conseguir un marcador y una cámara (webcam o de dispositivo móvil), entrar en el programa a través de la web de la empresa y ver cómo quedaría ese mueble en el sitio que se desea. Por ejemplo, IKEA lanzó un catálogo de productos con sus marcadores (figura 12).

Figura 12: Ejemplo del catálogo de IKEA visto a través de un móvil.

2.2. Reconocimiento de Formas

La disciplina del Reconocimiento de Formas tiene como objetivo la obtención de una representación invariante y discriminativa de las imágenes

La obtención de una representación invariante y discriminativa de las imágenes es el objetivo declarado de la disciplina del *Reconocimiento de Formas*, que se engloba en un conjunto de técnicas mucho más amplio: la *Inteligencia Artificial* [COH82]. El Reconocimiento de Formas, sin embargo, ha ido ganando importancia por sí mismo, hasta el punto de que en la actualidad constituye un campo de investigación que evoluciona con dinámica propia, a menudo independiente de la Inteligencia Artificial.

En Reconocimiento de Formas se ha trabajado mucho en el problema de la segmentación: la diferenciación de los subobjetos de un objeto o la de un objeto respecto del fondo que lo rodea. Cuando el objeto se halla sobre un fondo tiene una característica obvia que lo diferencia (mucha menor amplitud, color distinto,...) la solución puede ser sencilla, pero si el objeto está con otros objetos o es parte de un objeto mayor o la imagen ha sufrido distorsiones puede ser extremadamente difícil.

A la hora de buscar características de los objetos de interés que intentan reconocer en la imagen se puede enfocar la tarea, principalmente, de dos formas: basándose en la apariencia global de la imagen o bien en la apariencia local.

2.2.1. Descriptores globales

Al realizar un reconocimiento de una imagen con descriptores globales comúnmente calculan un vector de características multidimensional y permiten el uso de distancias bien conocidas para usar en clasificadores, como K-nn, Super Vector Machines, Mixturas de Gaussianas o Redes Neuronales.

Cuando se aplican los cálculos se considera la imagen al completo por lo que tiene la desventaja de que no se tendrá en cuenta la información que se podría extraer al considerar la zona de la imagen que se está comprobando. Sin embargo se evita el problema de decidir dónde buscar zonas de interés en la imagen dónde se pueda extraer información útil.

Dentro de la representación global podemos encontrarnos distintas formas de solucionar el problema que representa la segmentación: directa, basada en histogramas, basada en estadísticos, basada en filtros de Gabor,…

2.2.2. Descriptores locales

Los descriptores locales permiten tener una visión de la imagen particularizada a zonas concretas de la misma en las que las características de interés para el reconocimiento no se vean difuminadas por el total de la imagen. Esto puede ser de interés cuando se busca diferenciar algún elemento entre los distintos que pueda de la imagen, por ejemplo encontrar las matrículas de coches en una imagen de una carretera; o puede servir para buscar características que diferencien a la imagen entre un grupo de éstas, como por ejemplo a la hora de verificar en un sistema biométrico si la persona que intenta ser validado es quien dice ser y para ello se comprueban rasgos de su cara (ojos, nariz, etc.) que lo definan.

Es importante la elección de las apariencias locales que se usarán para obtener características. Normalmente se buscarán zonas que ofrezcan más información y sean lo más discriminantes posible y que sean robustas a transformaciones (rotación, perspectiva, distorsión,…).

Capítulo 3

Técnicas propuestas

En este capítulo se describen las técnicas que se han propuesto para solucionar la segmentación de una imagen y, a partir de ahí, poder construir el sistema de RA. Todas ellas se han implementado y se han incluido en la librería creada tal y como se describe en los capítulos 4 y 5: Librería y Aplicaciones implementadas, o bien se ha experimentado con ellas pero no han sido incluidas en la librería, ver el capítulo Experimentos.

Para poder realizar un tracker que detecte una mano en las imágenes capturadas y que posteriormente aplique RA, es necesario: diferenciar la mano del fondo y del resto de elementos existentes en la imagen, extraer las características de ésta y realizar cálculos sobre los datos obtenidos, y que la función de renderizado reciba o calcule la posición, rotación y escalado del sistema de coordenadas de los objetos sintéticos en el espacio virtual.

Es la primera fase, la diferenciación o segmentación de la mano, tal vez la más importante y la que presenta más complicaciones para obtener buenos resultados. Si ésta no se realiza adecuadamente, el resto de la aplicación arrastrará los errores cometidos, dando lugar a resultados no deseados. Estos serán o bien que no reconozca una mano presente en la imagen (falso negativo) o que reconozca como mano a un objeto que no lo es (falso positivo).

En cualquier aplicación de RA es imprescindible que la parte del tracker sea lo más rápida posible tanto a la hora de reconocer el objeto a tratar como a la hora de calcular la posición, rotación y escala buscados, ya que la aplicación ha de funcionar en tiempo real. Esto significa que el retraso entre las acciones del usuario y la respuesta obtenida del ordenador sea reducido o se corre el riesgo de perder la sensación de realismo necesaria. Además, cuanto más rápida sea esta parte más tiempo de cómputo podrá dedicarse a la parte de visualización dando lugar a resultados más atractivos para el usuario.

Binarización

La binarización de una imagen consiste en convertir el color de cada píxel únicamente a blanco o a negro, permitiendo así un tratamiento posterior de la imagen mucho más simple, ya que sólo se trabaja con los dos valores correspondientes a “objeto de interés” y “fondo”. Además como sólo hace falta un canal de color el tamaño de estas imágenes es menor con lo que el coste en memoria al manejarlas también es más reducido.

3.1. Umbralización

La umbralización (thresholding) es uno de los métodos más simples y rápidos de binarización que existen. Inicialmente se elige un valor umbral que servirá de factor de decisión. Al ejecutar el algoritmo, se comprobarán todos los píxeles de la imagen y si el valor del píxel a comprobar supera el umbral se etiquetará a un valor binario y si no es así, se le asignará el otro valor binario. Por lo tanto el coste de una segmentación es:

El resultado de ese algoritmo es una imagen binarizada.

3.2. Clasificador de K vecinos más cercanos

El clasificador del tipo “K-vecinos más cercanos” (“K-nearest neighbor”) es un método de clasificación supervisada que clacula la probabilidad de que un elemento pertenezca a un clase conocida. En este caso, para cada píxel de la imagen se comparan sus características con las de todos los píxeles etiquetados como “mano” y “no-mano” de un fichero de entrenamiento. Este fichero ha sido creado previamente con un conjunto de imágenes de muestra en distintas situaciones lumínicas y con manos en distintas posiciones. La comparación se resuelve con el cálculo de la distancia euclídea entre los valores de RGB del píxel a etiquetar y los valores de cada píxel de entrenamiento, esto es:

Distancia(,)=

Mientras se realiza este cálculo se van guardando las etiquetas de aquellos K píxeles de entrenamiento que estén más cerca del píxel a comprobar. Una vez este cálculo se ha realizado para todos los píxeles que se hayan suministrado de entrenamiento, se etiqueta el píxel con la etiqueta mayoritaria en el conjunto de las K etiquetas de píxeles más cercanos. También es posible comparar ventanas de píxeles; en este caso, en vez de ser solamente un píxel se selecciona un píxel y aquellos que están a su alrededor. Por ejemplo, si la ventana es de 3x3 se comprobarán los valores del píxel central y los de sus colindantes (ver figura 13). Usar ventanas proporciona información de un área de píxeles en vez de un solo punto, por lo que se puede adquirir una información de conjunto que con un único píxel podría pasar desapercibida.

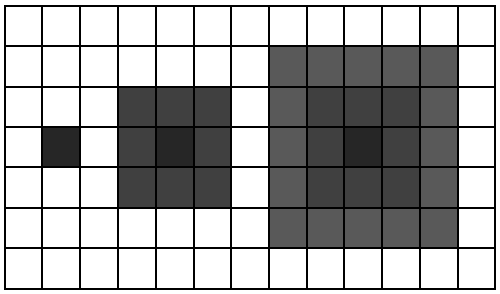


Figura 13: Ejemplos de ventanas de 1 píxel, 3x3 píxeles y 5x5 píxeles

3.3. Método de clustering: K-medias

Este método permite reducir el número de muestras con las que tratar la clasificación de píxeles, extrayendo del conjunto de muestras un número determinado por K de elementos. En este proyecto cada elemento es un vector de características calculados como la media de los subconjuntos de píxeles más cercanos.

El algoritmo empieza así: Se elige aleatoriamente K píxeles del subconjunto de píxeles etiquetados como “mano”, considerando que ese valor es la media de los K conjuntos en los que se agrupará a los píxeles. Iterativamente se va estimando la media de cada grupo y una vez hecho se calculará qué píxeles son los más cercanos. Al cambiar los píxeles su agrupación hacia su media más cercana, los K conjuntos irán cambiando. Este bucle seguirá hasta que, entre una iteración y otra, no haya ninguna reordenación de píxeles entre los K conjuntos. Una vez finalizado se procederá a hacer lo mismo para el subconjunto de píxeles “no-mano”. De esta forma tenemos la media de 2xK subconjuntos que representan al conjunto total de píxeles de entrenamiento, y siendo éstas las que se comparan con los píxeles de la imagen.

3.4. Detección de Bordes: Algoritmo de Canny

El algoritmo de Canny sirve para detectar bordes existentes en una imagen. Este algoritmo emplea de máscaras de convolución y está basado en la primera derivada gaussiana. Se considera que en una imagen en escala de grises aquellas zonas en las que existe un cambio brusco de nivel de gris es un borde. Dado que en el tratamiento de imágenes, se trabaja con píxeles, y en un ambiente discreto, el algoritmo de Canny hace uso de máscaras, las cuales representan aproximaciones en diferencias finitas.

Se ha empleado Canny para extraer los bordes de todos los objetos de la imagen cápturada con el objetivo de encontrar el contorno de una mano y realizar la segmentación por comparación de contornos.

Capítulo 4

Librería de Realidad Aumentada

Con el objetivo de simplificar desarrollos de otros posibles sistemas de RA o de ampliaciones a éste, se ha creado una librería con las funciones desarrolladas. Se han incluido en este capítulo las librerías externas que se han usado: OpenCV, OpenGL y Dibujo.

4.1 Librerías externas

4.1.1. OpenCV

La librería libre OpenCV, creada inicialmente por Intel, es una potente herramienta a la hora de implementar aplicaciones que hagan uso de la visión por computador. Una de sus principales ventajas es ser multiplataforma y su simplicidad para gestionar tanto imágenes estáticas de diversos formatos como flujos de imágenes capturadas por una cámara digital. Además tiene implementadas una gran cantidad de funciones optimizadas para el tratamiento de imágenes y reconocimiento de formas.

Su función en este proyecto ha sido el de tratamiento de la imagen capturada y su procesamiento para extraer la información necesaria para la localización de la mano en la imagen.

4.1.2. OpenGL

Esta famosa librería gráfica sirve de puente entre las tarjetas gráficas y la aplicación a desarrollar gracias a que es multiplataforma y multilenguaje. El objetivo de sus desarrolladores siempre ha sido el de la eficiencia, por lo que es un API de coste computacional reducido.

En este proyecto ha sido empleada para generar el sistema 3D que empleará el sistema de RA. Provee tanto de las transformaciones (Rotación, Translación y Escalado) como del renderizado de la imagen capturada y de los objetos 3D que se superponen a ésta.

4.1.3. Dibujo

Esta famosa librería define las clases color, punto y flecha. Ha sido creada a partir de la librería Dibujo2 usada en la asignatura Gráficos por Computador (GPC) de la UPV. Se ha usado para implementar en OpenGL la geometría de las flechas que simularán los ejes de coordenadas en el programa Tracker v.4, que se explicará más adelante. Con esta librería, además se puede modificar el color, el punto de inicio y el punto de fin de la flecha.

4.2. Umbralizar

Con el objetivo de minimizar ese tiempo de respuesta, se ha elegido como método de segmentación una umbralización. Inicialmente se probó una modificación de la umbralización para que etiquetase un píxel como mano si el valor para cada componente de color (RGB, YCbCr, etc.) estaba dentro del rango umbral y no-mano si se encontraba fuera [MAN05] y [SAN03].

Lamentablemente, el color obtenido por el sensor de la cámara usada en los experimentos es muy dependiente de la iluminación existente en el momento de la captura. Por ejemplo, una mano en una imagen capturada puede tener una fuerte presencia de un color en concreto si se compara con la siguiente imagen que llegue de la cámara, simplemente porque la cantidad de luz que ha incidido, tanto directa como indirectamente, ha variado ligeramente. Debido a esto, la umbralización presentaba grandes errores y una gran presencia de ruido si la iluminación no era uniforme en toda la habitación. Estas situaciones se dan si entra luz solar en la habitación por una ventana (luz direccional fuerte) o si hay luces con base de color distinta (ej.: la luz amarillenta de la bombilla de un flexo).

Con esta modificación se obtuvieron los mejores resultados en habitaciones grandes iluminadas con tubos fluorescentes, con limitada presencia de luz solar. De esta forma la luz emitida al rebotar múltiples veces por la habitación se podría considerar que existe una fuerte iluminación ambiental.

La siguiente modificación que se ha realizado es un cambio que da al sistema un poco más de resistencia a las variaciones de color en los píxeles por los cambios de la luz incidente en el sensor. Considerando que si el color de la imagen se ve modificado por la luz, aclarándola u oscureciéndola; al menos la proporción entre los valores RGB debe más o menos mantenerse. Por lo que en vez de umbralizar los valores RGB directamente, se han comparado a los valores umbrales el valor R y la proporción R-G y R-B. [STÖ04]

La elección de los valores umbral se explica más adelante en la implementación del programa Info.

4.3. Detección de contornos

Esta función hace uso la función `cvFindContours` implementada en la librería OpenCV. Una vez umbralizada la imagen, se le aplica una erosión y una dilatación para borrarle parte del ruido que pueda existir, dejándola más limpia para que `cvFindContours` dé mejores resultados y funcione más rápidamente tras eliminar todas las pequeñas áreas de unos pocos píxeles que puedan existir y que no forman parte de la mano, sino de algún posible brillo o sombra puntual de otro objeto con un color semejante. La función `cvFindContours` devuelve todos los contornos de la figuras presentes en la imagen umbralizada (sólo funciona sobre una imagen de un solo canal de color) aunque se parametriza para que el resultado sea sólo aquellos más externos, así se ignora cualquier hueco en la mano que no haya sido eliminado en la dilatación, y del resultado devuelto se elige aquel con el mayor área, dado que se supone que la mano será el objeto presente de mayor tamaño en la imagen. Como resultado final de la función `contornos`, se devuelve una caja de recubrimiento mínimo del objeto y una imagen con un objeto "mano" de color blanco sobre fondo negro (figura 14).

Figura 14: Ejemplo de una imagen devuelta por el detector de contornos

4.4. Pintado de caja de recubrimiento mínimo

El resultado de la función de detección de contornos es una imagen y una estructura de OpenCV llamada CvBox2D. Estas cajas de recubrimiento mínimo de dos dimensiones están implementadas por un rectángulo definido la altura y anchura, un centro y un ángulo respecto al eje vertical de la imagen. Respresenta el rectángulo de menor área no alineado con los ejes en el que se puede contener el contorno del objeto. Tanto el ángulo de la caja como sus lados se usan para el posterior cálculo del sistema de coordenadas virtual.

Esta función `pintarCajaMinima` muestra, sobre la imagen, el rectángulo que se le pasa por parametro (figura 15).

Figura 15: Ejemplo de una caja de recubrimiento mínimo

4.5. Volteo vertical de la imagen

La función `voltear_VERT_imagen` se ha creado porque OpenGL utiliza un sistema de píxeles distinto al de OpenCV. Para OpenCV el píxel (0,0) es el superior izquierdo, mientras que en OpenGL era el inferior izquierdo. La función de OpenGL `glDrawPixels` sirve para pintar en el buffer de color el vector de píxeles que se le pasa como parámetro, por lo que para una visualización correcta hace falta voltear verticalmente la imagen.

4.6. Distancia al borde del objeto

Una vez obtenida la imagen de la función de detección de contornos se le aplica la función de cálculo de distancia. Este proceso sirve para calcular en píxeles la distancia de cada píxel al borde más cercano. Se considera que el píxel con el valor más alto es el centro de la mano. En caso de empate entre dos o más píxeles, el centro será la media entera de esos píxeles.

Primero se da una pasada a todos los píxeles de la imagen desde el píxel (0,0) hasta el último. En este recorrido a todos los píxeles etiquetados de un color distinto al del fondo se les renombra a un valor igual al valor menor de sus 8 vecinos colindantes más uno. Una vez ha finalizado esta primera pasada se hace una segunda, esta vez desde el último píxel hasta el primero, realizando la misma operación para cada píxel de color distinto al del fondo, el resultado se puede ver en la figura 16.

Figura 16: Ejemplo de la imagen resultante tras un cálculo de distancia

A la hora de definir dónde se encontraba el centro de la mano, se consideró que el centro de la mano podía ser el centro de la caja de recubrimiento mínimo, pero en los experimentos se vio que no era un centro preciso que tendía más a estar en la raíz de los dedos en vez de en medio de la palma, mientras que el que se extrae con esta función estaba mucho más ajustado a la realidad. Sin embargo, si es necesario acelerar el tiempo de respuesta del sistema a cambio de perder algo de precisión, se puede prescindir de la función de distancia a favor del centro de la caja, como una posible solución.

Capítulo 5

Implementaciones

6.1. K-nn

En las primeras pruebas realizadas para detectar el objeto en la imagen que debería ser reconocido como mano se implementó una simple umbralización con unos valores fijos predefinidos. Tras diversas pruebas, explicadas posteriormente, con ese método se pudo observar que se podía localizar la mano si la iluminación era adecuada. Lamentablemente la cantidad que recibe la cámara varía mucho ante el más simple cambio de la escena observada, esto puede ser: que la iluminación de toda la sala cambie, que caiga una sombra sobre la mano o la cámara, que la mano se aleje o se acerque haciendo que la luz reflejada sobre ella sea más o menos potente, etc. Esta variabilidad afectaba gravemente al sistema ya que la mano podía no sólo ser más o menos oscura sino que la imagen cambiaba significativamente de color, esto queda mejor reflejado con los siguientes ejemplos (figuras 17 a 20):

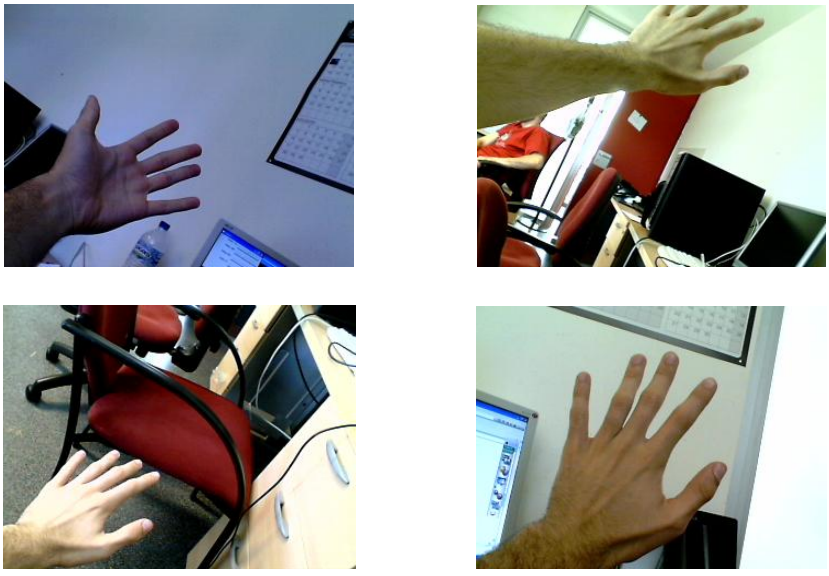


Figura 17-20: Ejemplo de la variación de color por los cambios en la iluminación

Para solventar este problema se decidió hacer uso de un clasificador del tipo “K-vecinos más cercanos” (“K-nearest neighbor”). El programa K-nn implementa este algoritmo de clasificación y sirve para calcular la tasa de error que se obtiene al clasificar los píxeles de un fichero de Test con el algoritmo de vecino más cercano. Al ejecutar K-nn se le pasa como parámetro el fichero de Entrenamiento y de Test, y el valor de K. Estos ficheros han sido creados previamente con un conjunto de imágenes de muestra en distintas situaciones lumínicas y con manos en distintas posiciones

como las de la figuras 16 a 19. Se muestra por pantalla el porcentaje de error de clasificación para los K primero vecinos.

En la siguiente tabla se pueden ver los resultados que este programa mostró durante los experimentos con una K entre 1 y 9 y tamaño de ventana de un único píxel.

K	Error
1	3.4%
3	3.4%
5	3.0%
7	2.5%
9	2.3%

Tabla 1: Tasa de error de K-nn para distintas K con tamaño de ventana 1

Como se puede observar las tasas de error son buenas y permitirían un reconocimiento de la mano sin problemas, por desgracia para llevar a cabo la clasificación de todos los píxeles de la imagen se tardaba demasiado para las exigencias del proyecto, ya que en el ordenador de prueba el tiempo superaba el minuto y se necesitaba que funcionase por debajo de 20 milisegundos. Esta exigencia era necesaria para que el sistema funcionase en tiempo real. Así que se implementó el algoritmo de K-medias.

6.2. K-medias

Tras las pruebas realizadas con el clasificador K-nn se pudo observar que aunque el resultado obtenido era bueno en cuestión de error, era demasiado lento ya que se estaban comparando 320x240 = 76800 píxeles de una imagen contra unos 540000 píxeles de entrenamiento y que si se usaba ventanas el coste aumenta proporcionalmente al tamaño de éstas. Por estas razones, el algoritmo de K-nn no servía como método de selección para el sistema y había que reducir ese número de comprobaciones. Para esto se implementó el algoritmo de clustering de K-medias (K-means) [MAC67], reduciendo el número de muestras desde varios cientos de miles a decenas.

Este algoritmo elige aleatoriamente K píxeles del subconjunto de píxeles etiquetados como “mano” (teniendo en cuenta que este K y el K del algoritmo K-nn no tienen por qué ser el mismo valor), considerando que ese valor es la media de los K conjuntos en los que se agrupará a los píxeles. Iterativamente se va estimando la media de cada grupo y una vez hecho se calculará qué píxeles son los más cercanos. Al cambiar los píxeles su agrupación hacia su media más cercana, los K conjuntos irán cambiando. Este bucle seguirá hasta que entre una iteración y otra no haya ninguna reordenación de píxeles entre los K conjuntos. Una vez finalizado se procederá a hacer lo mismo para el subconjunto de píxeles “no-mano”. De esta forma tenemos la media de 2xK subconjuntos que representan al conjunto total de píxeles de entrenamiento, y siendo éstas las que se comparan con los píxeles de la imagen.

Este programa tiene que recibir como parámetro un fichero de entrada que será el fichero de Entrenamiento y un número entero que será K, por cuestiones prácticas que se ha limitado K a valores menores de 100, ya que a partir de esa cifra el algoritmo ya funcionaba a una velocidad limitada. Adicionalmente se puede pasar como parámetro el número de iteraciones máximas (ITER_MAX) que se desea que realice el bucle por si la convergencia a una solución tarda en exceso, en caso de no introducir ningún valor está definido por defecto: ITER_MAX=10000 iteraciones. La salida se guarda en ficheros que empiezan en 10 medias para cada etiqueta ("mano" y "no-mano") hasta el K definido por el usuario en incrementos de 10; dando así diversos ficheros de entrenamiento para realizar los tests.

Aplicando la técnica de K-means se redujo drásticamente el tiempo de procesado de la imagen y con unas pocas decenas y ventanas de píxel no superiores a 7x7 ya era alcanzable procesar la imagen en unos 20-30 milisegundos, pero se sacrificó mucha precisión. Los errores pasaron del 2-3% a un 12% en el mejor de los casos en los experimentos, como se ve en las siguientes tablas:

K-medias	10		20		30		40		50	
	K	Error	K	Error	K	Error	K	Error	K	Error
	1	15.84%	1	19.35%	1	20.79%	1	17.87%	1	17.94%
	3	23.74%	3	22.33%	3	21.26%	3	12.39%	3	12.21%
	5	22.55%	5	40.50%	5	24.22%	5	13.87%	5	13.07%
	7	24.30%	7	46.20%	7	44.81%	7	16.44%	7	14.06%
	9	34.14%	9	48.23%	9	46.79%	9	18.76%	9	16.39%

Tabla 2: Tasa de error de K-nn con K-medias para distintas K con tamaño de ventana 1

K-medias	10		20		30		40		50	
	K	Error	K	Error	K	Error	K	Error	K	Error
	1	20.54%	1	13.89%	1	12.96%	1	12.37%	1	12.15%
	3	44.71%	3	25.12%	3	19.67%	3	16.58%	3	17.32%
	5	47.76%	5	44.05%	5	24.78%	5	18.51%	5	20.48%
	7	50.56%	7	45.31%	7	26.67%	7	23.01%	7	24.40%
	9	51.55%	9	55.91%	9	43.41%	9	24.50%	9	25.86%

Tabla 3: Tasa de error de K-nn con K-medias para distintas K con tamaño de ventana 3

K-medias	10		20		30		40		50	
	K	Error	K	Error	K	Error	K	Error	K	Error
	1	17.58%	1	15.29%	1	15.36%	1	15.49%	1	16.07%
	3	17.66%	3	16.71%	3	14.94%	3	14.93%	3	15.66%
	5	20.03%	5	15.10%	5	14.90%	5	14.37%	5	15.55%
	7	26.43%	7	17.09%	7	14.55%	7	14.18%	7	15.43%
	9	42.89%	9	23.63%	9	13.70%	9	13.50%	9	14.65%

Tabla 4: Tasa de error de K-nn con K-medias para distintas K con tamaño de ventana 5

K-medias	10		20		30		40		50	
	K	Error	K	Error	K	Error	K	Error	K	Error
	1	17.52%	1	16.67%	1	15.61%	1	16.84%	1	17.13%
	3	17.67%	3	16.23%	3	16.73%	3	15.62%	3	15.99%
	5	19.90%	5	15.31%	5	17.00%	5	15.37%	5	15.09%
	7	22.99%	7	18.24%	7	15.77%	7	15.20%	7	15.66%
	9	27.07%	9	23.96%	9	15.01%	9	14.62%	9	14.97%

Tabla 5: Tasa de error de K-nn con K-medias para distintas K con tamaño de ventana 7

Como se puede ver en los resultados de los experimentos para una cantidad reducida de K-medias se obtiene mejor tasa con un número bajo en el K del K-nn y con un tamaño pequeño de ventana. Según va creciendo el número de K-medias se consigue mejores resultados cuanto mayor es el número de K del K-nn y mayor el tamaño de ventana.

Al realizar las pruebas también quedó patente que al aumentar a más de 50 K-medias el programa iba demasiado lento para trabajar en tiempo real y que siendo un 12.15% la mejor tasa de error obtenida esta técnica no funcionaba lo bastante bien ya que el reconocimiento de la mano presentaba una gran cantidad de ruido y no era posible hacer los cálculos necesarios para posicionar el sistema de coordenadas 3D sobre la mano de la imagen.

6.3. Bibliotecario + lector+ lector2clasificador

Estas tres implementaciones sirven para generar los ficheros de Entrenamiento y de Test que utiliza K-nn. El programa Bibliotecario abre una imagen .PNG que se le pasa como parámetro y la muestra en una ventana. El usuario puede, sobre la ventana y mediante el uso del ratón, elegir dinámicamente regiones rectangulares de píxeles. Dado que la visualización se ha realizado con OpenCV y esta librería no implementa botones se puede elegir si se quiere seleccionar regiones de “mano” o de “no-mano” gracias a la trackbar que se puede ver en la figura 21. También se puede observar los distintos rectángulos siendo los rojos los que pertenecen a regiones de “no-mano” y los azules los de “mano”.

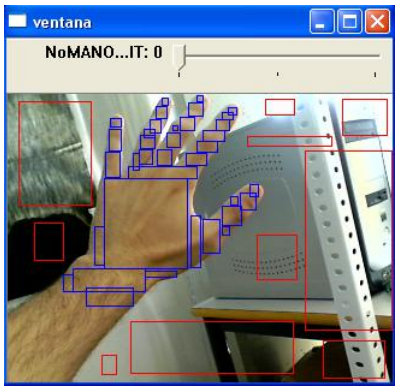


Figura 21: Seleccionando regiones con el Bibliotecario

Las áreas seleccionadas se guardarán en dos ficheros “MANO.txt” y “NoMANO.txt” respectivamente definiendo para cada imagen el píxel donde empieza cada región y en el que ésta acaba. Abriendo diversas imágenes se irán añadiendo a los dos ficheros las nuevas regiones rectangulares y a qué imagen pertenecen. Se realizó este programa con el objetivo de crear ficheros mucho más ligeros que las listas de vértices y que acompañasen a las imágenes de muestra.

Esos ficheros los lee la aplicación Lector y abre las imágenes que se encuentran en ellos, asignando a cada imagen dos vectores con todas las áreas de píxeles que les corresponde según el usuario había definido en Bibliotecario. Una vez hecho esto se procede a recorrer todas las imágenes y convertir cada área de píxeles en vectores de características, etiquetarlos y guardarlos en un fichero. Las características son los valores RGB de los píxeles y los vectores tienen un tamaño que depende de la ventana de píxeles que el usuario pida a la aplicación, que puede ser de 1x1, 3x3, 5x5, 7x7, o 9x9. Esto se hace con las teclas 1 a 9. La aplicación permite también extraer como características las componentes HSV de una ventana de un píxel con la pulsación de la tecla 0. De esta forma se crean listas de valores RGB de ventanas de píxeles etiquetadas que pueden ser usadas para entrenamiento y test de un clasificador. De esta forma funciona como traductor del formato de áreas de píxeles al formato que necesita para el programa K-nn.

Lector2clasificador recibe el fichero creado por el Lector y mezcla de forma aleatoria los vectores de características leídos. Esto se hace para que el fichero resultante pueda tener características de todas las imágenes de muestra que se le pasen. El usuario elige la cantidad de vectores que se guarda en el fichero que quiera usar para Test o para Entrenamiento. Estos ficheros, ahora cortados a un tamaño a gusto del usuario y con sus elementos mezclados aleatoriamente, resultantes son los que se pasarán al clasificador para comprobar las tasas de error o para clasificar píxeles en el sistema de RA.

La implementación de la funcionalidad de los tres programas se podría haber realizado unificada en un único programa pero se ha preferido hacer por separado por motivos prácticos a la hora de realizar los experimentos, ya que de esta forma se podía comprobar que en cada paso la información era correcta.

6.4. Info

Dado que se ha decidido no usar el clasificador K-nn por no funcionar lo bastante rápido para su uso en tiempo real y decantarse por un sistema de umbralizado con unos valores umbrales preestablecidos, se ha creado el programa Info. Sirve para valorar según los píxeles seleccionados en Bibliotecario qué umbrales escoger para que el sistema de RA funcione lo mejor posible. Las correspondencias de los valores en R, G y B de las regiones etiquetadas como “mano” tienen una distribución similar a una gaussiana, por lo que se puede modelizar con una media y una varianza. Se muestra, en ventanas separadas con unos histogramas (figura 22), como se distribuyen los

valores relacionados de R-B, R-G y B-G. Esta relación permite encontrar umbrales más resistentes a los cambios de luz y ligeramente más resistentes a los cambios de color.

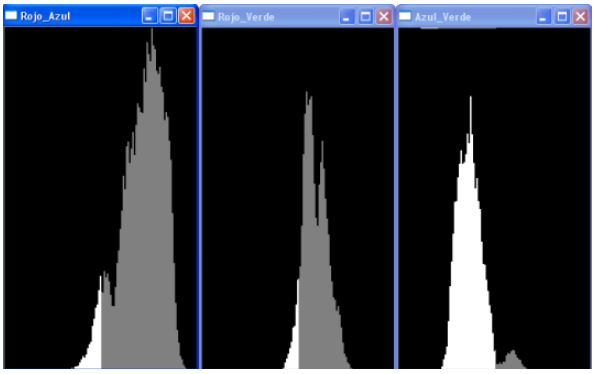


Figura 22: Los histogramas de las diferencias entre canales RGB. El color blanco son valores menores de 0 y el gris mayores o iguales.

Este programa da información sobre los píxeles de entrenamiento y ayuda a la elección de valores umbrales adecuados. Se realizó unas pruebas para calcular, según los datos obtenidos por Info, que tasa de error ofrecían distintos umbrales. Las pruebas se hicieron sobre dos conjuntos de imágenes de prueba: El primero son capturas de la cámara en las que hay gran variedad de objetos de distintos colores y una iluminación variable (imágenes 0 a 18 en la figura 22). El segundo grupo tiene un fondo oscuro y los cambios de luz son menos intensos (imágenes 18 a 30 en la figura 22).

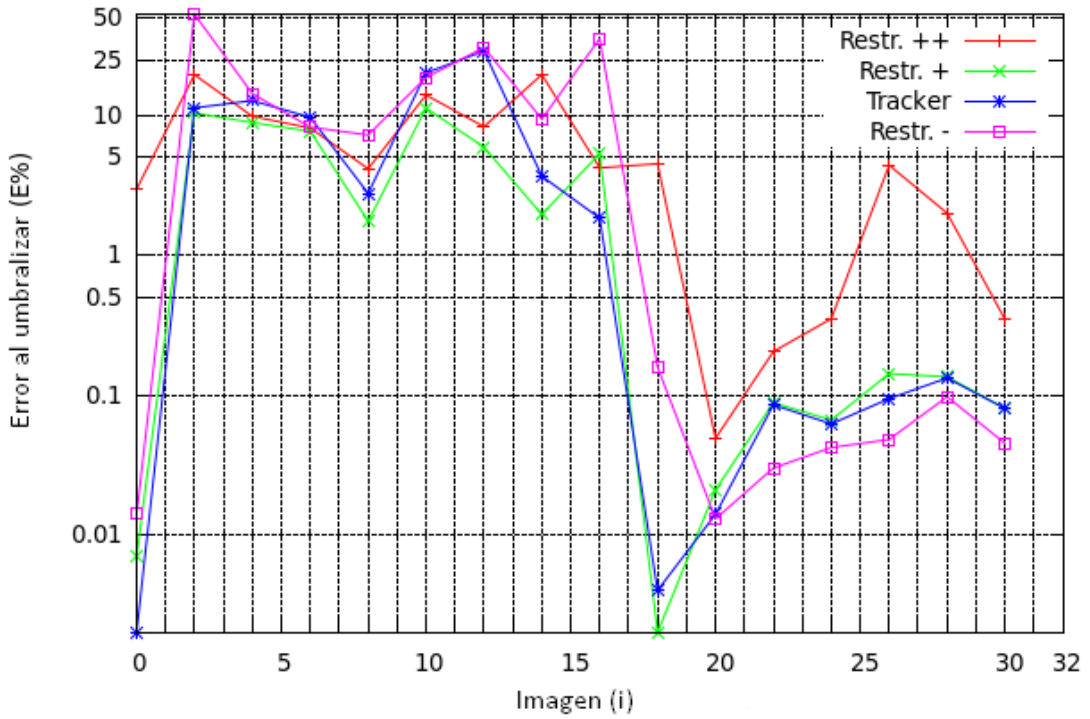


Figura 23: Gráfica con el error obtenido al umbralizar (E) para cada imagen (i). Cada línea de color representa unos valores de umbralización.

En la figura 23 se representa los valores de error optenidos para distintos valores umbral. Siendo la línea de color azul la que se ha elegido para el Tracker v.4, se han

definido el resto como más o menos restrictiva respecto a la elegida. Siendo el orden de más restrictiva a menos el siguiente: línea roja, línea Verde, línea azul y línea rosa. Cuanto más restrictiva es un umbralización menos píxeles serán catalogados como mano. Se puede observar que el umbral representado con el color verde y el elegido para el Tracker tienen una tasa de error semejante, siendo incluso peor la de Tracker, esto es así porque al ser un poco más restrictiva se han evitado una parte de los Falsos Positivos que sí se dan en la umbralización del Tracker aumentando sólo un poco los Falsos Positivos, sin embargo se ha elegido el umbral representado por la línea roja porque el sistema creado funciona mejor con una tasa más alta de Falsos Positivos que de Falsos Negativos, dando así mejores resultados. Las restricciones mucho más o mucho menos restrictivas tienen una tasa de error peor, por lo que el valor a elegir debería estar en el intervalo entre el umbral verde y el azul.

6.5. Tracker v.4

Tracker v.4 es la versión final de una serie de pruebas que han ido incluyendo características y funcionalidades de forma progresiva. En este programa se juntan las funciones necesarias para conseguir un sistema de RA que reconozca el gesto de mano extendida de una persona para la cual ha sido entrenado.

En la función principal main, el programa empieza inicializando las librerías externas y se abre el flujo de imágenes que pueden proceder de la cámara o de un video usando respectivamente `cvCaptureFromCAM` y `cvCreateFileCapture`. Con la primera imagen que se captura empieza el proceso de búsqueda de la mano. Lo primero es voltearla con la función `voltear_VERT_imagen` para ajustarla al sistema de OpenGL. Luego se binariza con los valores definidos en la función de umbralización, se buscan los contornos y se selecciona el que más probablemente sea la mano. Además, se calcula la caja de recubrimiento mínimo y se rellena una cola de posiciones con la posición encontrada y otra con el ángulo. Estas colas se usan para implementar un filtro de medias que evite saltos bruscos por culpa de un fallo en la detección en un frame aislado o para evitar cambios muy bruscos en la orientación del sistema 3D.

Tras la fase de inicialización, se pasa al bucle de funcionamiento propio OpenGl: MainLoop. Se lanzan así las callbacks `onDisplay` y `onIdle`. Estas dos funciones sirven para el renderizado y la visualización de la escena, y para actualizar los estados de los objetos mientras el sistema está a la espera.

Concretamente, la función onIdle comprueba si ha pasado un número definido de milisegundos desde la última vez que se actualizó la pantalla. Si no es así sigue a la espera, pero si ha pasado ese tiempo se hace una captura de imagen y se voltea, para después hacerle pasar por los mismos tratamientos de imagen y cálculos descritos anteriormente, con la excepción de que ahora no se inicializan las colas de los filtros, sino que se borran la posición y el ángulo más antiguos, éstos se encuentran al principio de las colas, y se añaden los nuevos valores que acaban de ser calculados. Sacando la media de los valores de las colas se obtienen la posición y el ángulo que se

usarán para posicionar y rotar el sistemas de coordenadas. Finalmente actualiza el tiempo de espera y hace una llamada a `glutPostRedisplay` que fuerza a que se ejecute la función onDisplay.

En la función onDisplay se empieza por borrar el buffer de profundidad de OpenGL para evitar problemas entre un frame y el siguiente. Normalmente, en una aplicación típica de OpenGL también se borraría el buffer de color, pero en este caso vamos a pintar como fondo la imagen real obtenida mediante las funciones de captura de la librería OpenCV, así ese buffer será sobrescrito por el fondo con la llamada a la función `glDrawPixels` en cada frame. Hay que tener cuidado en desactivar el test del Z-buffer antes de usar esa función, ya que dado que la imagen se pinta directamente sobre el raster estará lo más cerca posible que un objeto puede estar de la cámara y por lo tanto mucho más de lo que lo estará cualquier objeto 3D que queramos pintar posteriormente. Si no se desactivase, el fondo taparía por completo a los objetos por estar delante de ellos espacialmente. Una vez se ha producido la llamada a glDrawPixels se debe volver a activar el test de profundidad y los objetos 3D se ocluirán entre ellos normalmente. Después, se realizan los cálculos que definirán dónde se posiciona el sistema de coordenadas y cuánto se rotan los objetos 3D que se dibujen.

Para calcular la posición del sistema 3D en el plano 2D de la imagen que nos da OpenCV hubo que definir en los experimentos un caso base en el que la imagen está a una distancia conocida y sobre la que poder calcular las posiciones relativas que se calculan en tiempo real cuando se ejecuta Tracker v.4. Este caso base es la profundidad y el . Esa distancia servirá para normalizar la distancia a la que está la mano en cada frame. Se hace de la siguiente forma:

- El área de la mano en el instante t se normaliza dividiéndola por el , que es el área de la mano en la distancia . Esto nos dirá cuanto ha aumentado o disminuido la mano respecto al caso base.
- El factor diferencial de distancia se define como inversamente proporcional a la raíz cuadrada del área normalizada.
- El valor de distancia en el instante t será: = * .

Ahora ya sabemos a qué profundidad está la mano, por lo que podemos calcular la equivalencia de una unidad de distancia de OpenGL (GLunit) en píxeles, porque además sabemos el tamaño total de la imagen, la posición del centro de la mano en la imagen y el ángulo de la cámara:

- Suponemos que la altura del Viewport (*H*) de OpenGL es del mismo tamaño que la imagen 2D.
- Tenemos .
- Suponemos conocido un ángulo del fustrum (*FoVy*) de la cámara. Éste se define en la función `gluPerspective`. En este programa, *FoVy* ha sido definido como 60º.
- Si – (figura 24). Siendo b = , – y = ——. Entonces y *H* =

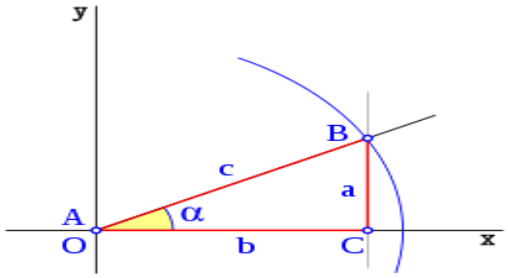


Figura 24: Ejemplo gráfico de tangente

- Si la altura de la imagen en píxeles es ***height*** y el desplazamiento en GLunit por píxel en Y es ***DespY*** = ——.
  Entonces ***DespY*** =

Este cálculo ha sido para la coordenada Y de la imagen 2D, para la componente X el cálculo del desplazamiento es:

DespX = (***Ratio.***

Siendo la anchura de la imagen en píxeles y el ***Ratio*** la resolución de la ventana, normalmente 4:3.

Así se consiguen los valores que convierten las coordenadas 2D en coordenadas 3D dependientes de la profundidad a la que se encuentre la mano. Sólo queda llamar a la función `gluLookAt` para posicionar la cámara respecto a la imagen. En este punto se debe señalar que aunque en todo momento se ha hecho referencia a que lo que se posiciona es el sistema 3D sobre la imagen, esto no es del todo cierto; pero por no adelantar explicaciones se ha dejado así hasta ahora.

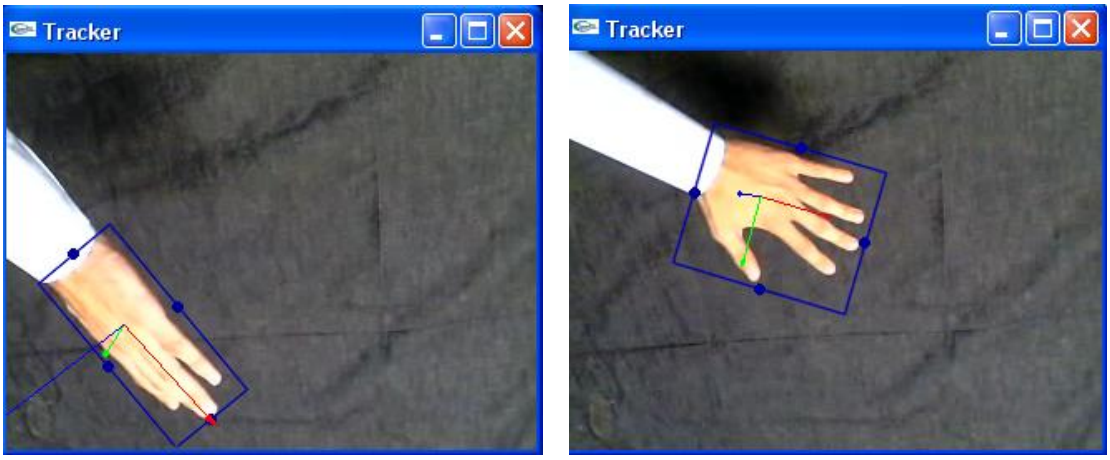
El sistema de coordenadas 3D no se mueve en ningún momento, y aunque parezca que vaya moviéndose por toda la imagen, realmente es la cámara de OpenGL la que se traslada por el sistema. Esto se hace mediante esta llamada a gluLookAt:

```
gluLookAt ( despX*(pos.x-(img->width)/2.0),
            despY*(pos.y-(img->height)/2.0),
            Z,
            despX*(pos.x-(img->width)/2.0),
            despY*(pos.y-(img->height)/2.0),
            0.0,
            0.0, 1.0, 0.0);
```

Las coordenadas se calculan respecto al centro de la imagen, ya que, antes de mover la cámara, el sistema 3D está en ese punto. Como se puede observar, el punto de vista y el punto donde se encuentra la cámara coinciden en las Xs y las Ys. Además, la cámara de OpenGL se sitúa en la coordenada Z y mira al centro de coordenadas. Esto es así para dar la sensación en todo momento de correspondencia entre la cámara real capturadora y la cámara virtual.

Para realizar la rotación de la mano en el plano de la cámara, se hace uso del ángulo calculado mediante el filtro de medias implementado con colas. Como se ha

explicado, cada ángulo del filtro se obtiene al realizar el cálculo de la caja de recubrimiento mínimo. Para la rotación que tiene la mano sobre el eje de giro de la muñeca se ha implementado como una transformación sobre el eje X del sistema y el valor a girar se obtiene como una proporción entre la anchura y altura de la mano sobre el plano de la cámara o, lo que es lo mismo, la anchura y altura de la caja de recubrimiento. Como se puede ver en las figuras 25 y 26, la anchura y altura son parecidas cuando la mano está extendida y paralela al plano de la cámara, y más dispares cuando la mano está más ortogonal al mismo.



Figuras 25 y 26: Ejemplos del funcionamiento del sistema de RA

Finalmente, entre un PushMatrix() y un PopMatrix() se llaman a las funciones que dibujan 3 flechas que simulan los 3 ejes de coordenadas, definidas en la librería externa Dibujo.

Capítulo 6

Experimentos

En este apartado se explican los distintos programas de prueba y experimentos que se hicieron durante la realización de este proyecto. Estos han servido de puente entre los programas principales, para comprobar el desarrollo de ciertas funciones y para la implementación de funcionalidades, aunque algunas luego se desestimaron por no funcionar lo bastante rápido o porque no solucionaban adecuadamente el problema planteado.

6.6.1. Clasificador

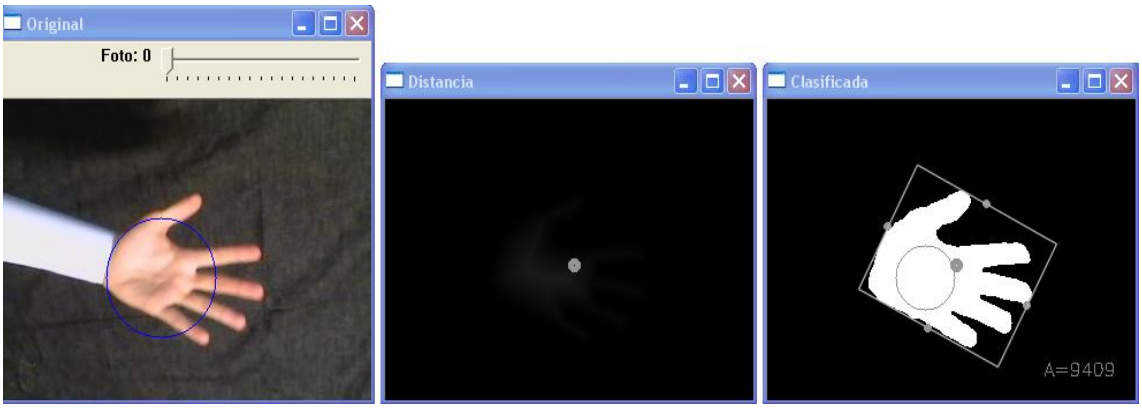
Este programa realiza la búsqueda del punto central del contorno más grande que, en principio, es la mano, mediante la función de distancia. Como ejemplo carga las fotos de muestra y busca el centro de la mano. En las tres ventanas que se abren se puede ver respectivamente la imagen capturada con un círculo azul dónde estaría el centro, la superficie en distintos tonos de gris de los objetos que tienen un color similar al de la piel, siendo la que se ha decidido como mano la que es de color blanco y el resultado de aplicar la función distancia (figuras 27-28).

Figuras 27-28: Las ventanas del clasificador haciendo uso de las funciones de contornos y de distancia

6.6.2. Tracker v.1

En esta primera versión del Tracker se sigue probando con imágenes estáticas. Una vez se ha observado en el programa anterior que se consigue un resultado

aceptable con la umbralización, se ha pasado a extraer información sobre el área, la proporción entre largo y ancho, y la inclinación de la mano para calcular posteriormente la posición, el escalado y la rotación del sistema de coordenadas 3D sobre la mano. Para esto se ha hecho uso de la función OpenCV que calcula la caja de recubrimiento mínimo y se ha pintado sobre la imagen con el área de la mano. Como ejemplo, este programa hace uso de las imágenes de muestra y para cada una abre tres ventanas en las que se muestra la original con el círculo azul, la imagen con la distancia y la que tiene la caja de recubrimiento y el centro de ésta (figuras 29 a 31). Adicionalmente, se muestra por pantalla el tamaño de la caja, la proporción de cada lado y el área de la mano.



Figuras 29-31: El Tracker v.1 añade ya el pintado del centro y la caja de recubrimiento mínimo.

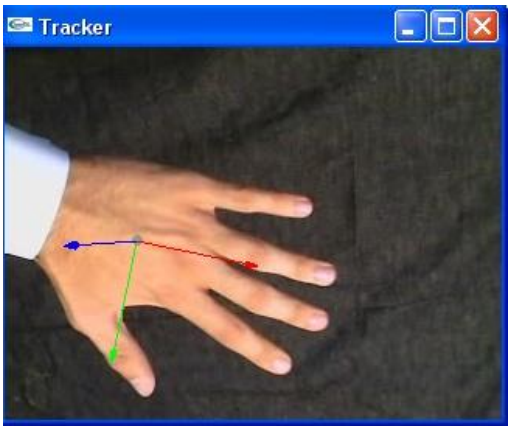
6.6.3. Tracker v.2

En Tracker v.2 el cambio que se ha implementado ha sido la eliminación de las imágenes de muestra a favor del flujo de imágenes que suministra la webcam, por lo que es necesario tener una conectada para que el programa funcione. El objetivo de esta prueba era comprobar que el sistema no daba error al funcionar con una webcam y que la calidad no se resentía. El resultado fue positivo y se obtuvo unos resultados similares a los de Tracker v.1 pero respondiendo a un flujo constante de imágenes. Además se pudo observar que el programa no se ralentizaba y podía mantener un Frame Rate de unos 60 fps.

6.6.4. Tracker v.3

Esta prueba final vuelve a usar una imagen de muestra. Esto es así ya que usar imágenes estáticas es más cómodo que un flujo de imágenes. Se ha implementado el sistema 3D sobre la mano y como ejemplo se ha dibujado tres flechas que reflejan los ejes de coordenadas. Aquí ya está incluida la librería OpenGL, en las anteriores pruebas había sido OpenCV, que es la que realiza el proceso de visualización ya que se ha incluido los objetos 3D. Una vez comprobado el correcto

funcionamiento de este paso, simplemente había que cambiar la imagen por el flujo de imágenes. Cómo se ha integrado OpenGL se ha explicado en Tracker v.4, ya que ahí fue donde se parametrizó para ajustar al máximo la correspondencia entre espacio 2D y el espacio 3D.

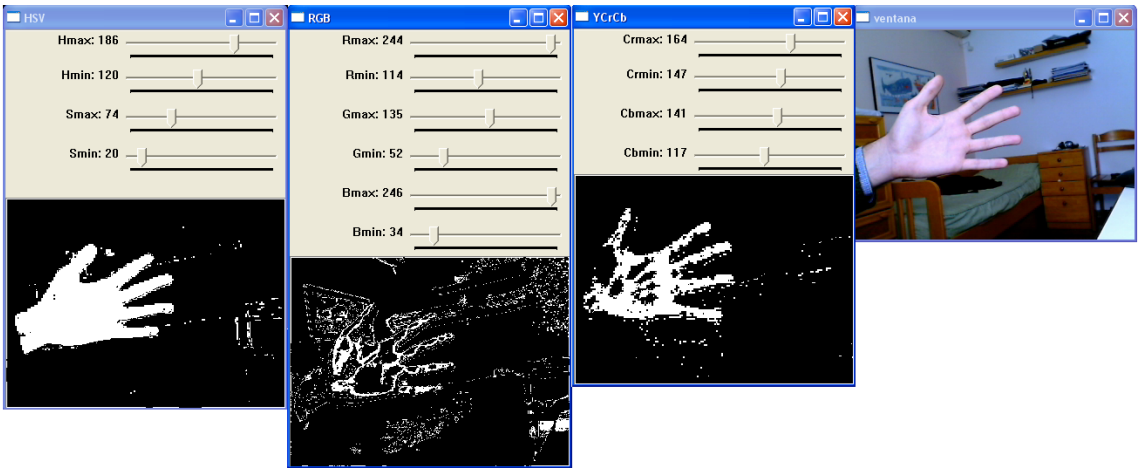


Figuras 32: En Tracker v.3 posicionando el sistema 3D sobre la imagen 2D

6.6.5. Cajas de Prueba

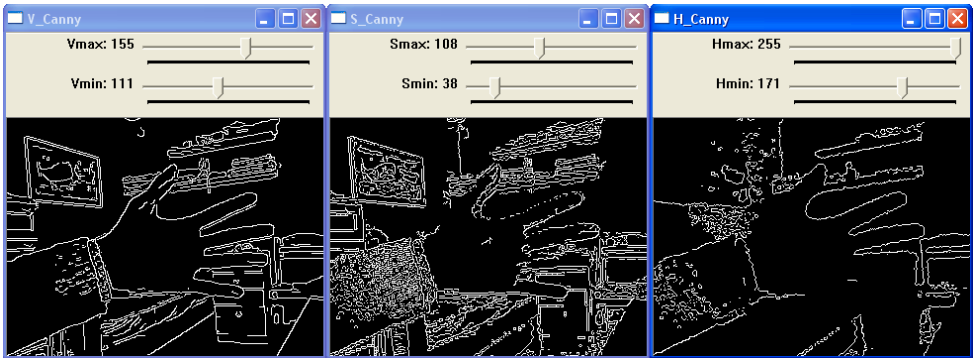
Hay cuatro Cajas de Prueba e implementan cuatro grupos de experimentos que se realizaron al principio del desarrollo del proyecto. Sus objetivos eran probar diferentes técnicas de tratamiento de imagen, sobre todo umbralización, para encontrar la más adecuada para realizar la binarización de la imagen, considerando como adecuada aquella que simplificase lo máximo posible la detección de la mano eliminando gran parte de los objetos distintos a la mano y que no ralentizase el funcionamiento del programa.

Los experimentos realizados en Caja de Prueba1 fueron el primer intento de realizar una umbralización de los canales de color de la imagen. Como se quería comprobar en qué espacio de color se conseguía mejor resultado se crearon 3 ventanas con los espacios HSV, RGB e YCrCb y se configuraron las trackbars para valores máximos y mínimos de cada componente que podía influir significativamente.



Figuras 33-36: Ejemplo de mano umbralizada en los espacios de color HSV, RGB y YCrCb

Tras observar los resultados de esta prueba se decidió probar dos nuevos enfoques. Por un lado implementar el algoritmo K-nn como se ha descrito previamente y por otro lado realizar en Caja de Prueba 3 una experimentación con la detección de bordes de Canny [CAN86] con los que intentar encontrar la silueta de la mano. El algoritmo que implementa esta técnica en OpenCV requiere una imagen de un solo canal de color, por lo que se hicieron pruebas con las tres componentes HSV y unas trackbars permitían parametrizar la función.



Figuras 37-39: Resultado de aplicar Canny a cada una de las componentes H, S y V

El resultado no fue lo bastante satisfactorio, porque en los mejores casos, como el de la figura 37, se podía distinguir la mano visualmente, pero presentaba pequeñas aperturas o defectos en el borde que impedían una posterior comparación de contornos con los de una mano base.

La caja de Pruebas 3 permite saber qué valores de RGB existen en un píxel en concreto en el que haya pulsado el usuario.

La última de estas pruebas está también orientada a los contornos pero sin hacer uso del algoritmo de Canny. Aquí se abren cuatro ventanas de visualización más una donde se encuentran dos trackbars para modificar los valores máximo y mínimo de los valores umbral que se aplican a las componentes H y S del espacio HSV. El resultado de la umbralización se puede ver en la figura 40 (superior izquierda). Después se aplica una serie de erosiones, dilataciones, rellenado de huecos y otros filtros de limpieza de imagen dando un resultado similar al de la figura 41 (inferior izquierda). Aquí quedaba

patente que era posible reconocer la mano si las condiciones de luz eran adecuadas. También se intentó hacer un umbralizado adaptativo sobre la componente V y se observó que se obtenía un resultado con cierto potencial. Como se puede ver en la figura 42 (superior derecha) se consigue ver el contorno de la mano bastante bien, así que se le aplicó cvFindContours a la imagen resultante. En general se podía conseguir un contorno de la mano bastante aceptable pero lamentablemente no era constante debido a que de vez en cuando el contorno se abría y su interior se juntaba con el del fondo obteniendo un contorno muy diferente al esperado, o debido a las líneas de la mano se formaban contornos separados para los dedos y para la palma.

Figuras 40-43: (Izquierda) Umbralización y limpieza de imagen. (Derecha) Substracción de valores V a H y contornos sobre el resultado.

Capítulo 5

Conclusiones

Se ha desarrollado un sistema de RA sin los marcadores típicos, usando únicamente la mano del usuario. Esa mano capturada es el punto de unión entre la realidad tal y como la percibimos y los objetos virtuales creados para aumentar tal realidad. Se ha limitado el número de gestos a reconocer a la palma extendida debido a las dificultades que se obtuvieron para poder realizar un reconocimiento adecuado.

Se ha implementado un conjunto de funciones que agrupan las acciones necesarias para crear este sistema y que pueden ser de utilidad para generar otros en el futuro.

Se ha experimentado con diversas técnicas considerando tanto la exactitud de la localización de la mano como el coste computacional que condiciona este tipo de sistemas, obteniendo la umbralización con valores preestablecidos como aquella que presenta mejores resultados, aunque esta técnica presenta algún problema ante cambios de iluminación capturados por la cámara.

Como trabajo futuro queda añadir el reconocimiento de más de una mano, así como añadir la posibilidad de que se interpreten una serie de gestos [MAN05], [STÖ04] con los que obtener más posibles respuestas. Una posible solución sería reconocer las puntas de los dedos [LEE08].

Una solución planteada para mejorar la resistencia del sistema a los contrastes de iluminación es añadir una cámara de infrarrojos para calcular profundidad o el calor corporal [OKA02]. Ésta ofrecerá una información muy valiosa al sistema: la posibilidad de diferenciar objetos en la imagen tanto por el color como por la distancia a la cámara, eliminando muchos de los problemas que se tienen con un sistema con una sola cámara RGB. Esto ya lo ha puesto en práctica la empresa Microsoft con su dispositivo Kinect en el ámbito de entretenimiento y que ya empiezan a aparecer proyectos que hacen uso de tal herramienta para robótica, diseño, medicina, etc.

Bibliografía

[AZU97] Azuma, R. T., A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, Vol. 6, N. 4, pp. 355-385, 1997.

[ART03] Thermo-Key: Human Region Segmentation from Video Using Thermal Information, SIGGRAPH2003. http://www.hc.t.u-tokyo.ac.jp/project/thermo-key/

[COH82] Cohen, P. R. and Feigenbaum, E., The Handbook of Artificial Intelligence, volume 3. William Kaufmann, Inc., 1982.

[FIX51] Fix, E. and Hodges, J.L., Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[JU04] Juan, M.C., Canu, R., Cano, J., Gimenez, M., "Augmented Reality Interactive Storytelling systems using tangible cubes for edutainment", IEEE International Conference on Advanced Learning Technologies Learning technologies in the Information society (ICALT'08), pp. 233-235 , 2008.

[JUA06] Juan, M. C., Joele, D., Botella, C., Baños, R., Alcañiz, M., Van der Mast, Ch. . The use of a visible and/or an invisible marker augmented reality system for the treatment of phobia to small animals. Annual review of Cybertherapy and Telemedicine, vol. 4, pp. 31-36. 2006.

[JUA09]M.C. Juan, "Advanced Learning. Chapter 10: Augmented Reality and Tangible interfaces for learning, Ed. In-teh", pp.153-166, 2009.

[JUA10] Juan, M.C., Toffetti, G., Abad, F., Cano, J., "Tangible cubes used as the user interface in an Augmented Reality game for edutainment", The 10th IEEE International Conference on Advanced Learning Technologies (ICALT'2010), pp. 599-603, 2010 .

[JUA10b] Juan, M.C., Llop, E., Abad, F., Lluch, J., "Learning words using Augmented Reality", The 10th IEEE International Conference on Advanced Learning Technologies (ICALT'2010), pp. 422-426, 2010.

[JUA11] Juan, M.C., Furió, D., Alem, L., Ashworth, P., Giménez, M., " Chapter: An Augmented Reality Library for Mobile Phones and its Application for Recycling", Open Source Mobile Learning: Mobile Linux Applications, Ed. IGI Global, (accepted, to be published by 2011).

[LEE08] Taehee Lee and Tobias Höllerer. Hybrid Feature Tracking and User Interaction for Markerless Augmented Reality. IEEE Conf. Virtual Reality (VR '08), pp. 145-152, 2008.

[MAC67] MacQueen, J.. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, volume 1, pages 281{297, Berkeley, 1967.

[MAN05] Manresa, C., Varona, J., Mas, R., and Perales, F. J.. Hand Tracking and Gesture Recognition for Human-Computer Interaction. Electronic Letters on Computer Vision and Image Analysis *5*(*3*): pp. *96-104*, 2005.

[MIL94] Milgram, P. and Kishino, F., A taxonomy of mixed reality visual displays, /IEICE (Institute of Electronics, Information and Communication Engineers) Transactions on Information and Systems/, Special issue on Networked Reality, E77-D(12), pp. 1321-1329, 1994.

[NAK08] Nakamoto, M., Ukimura, O., Gill, I. S., Mahadevan, A., Miki, T., Hashizume, M. and Sato Y., Realtime Organ Tracking for Endoscopic Augmented Reality Visualization Using Miniature Wireless Magnetic Tracker, Medical Imaging and Augmented Reality, 4th International Workshop Tokyo, Japan, pp. 360, 2008.

[NOV] Página web de Novarama, desarrolladora de Invizimals: http://www.novarama.com/, 2010.

[OKA02] Oka, K., Sato, Y. and Koike, H., Real-Time Fingertip Tracking and Gesture Recognition, IEEE Computer Graphics and Applications: pp. 64-71, 2002.

[SAN03] Singh, S. K., Chauhan, D. S., Vatsa M. and Singh, R., A Robust Skin Color Based Face Detection Algorithm. Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227-234, 2003.

[STÖ04] Störring, M., Moeslund, T. B., Liu, Y., and Granum, E., Computer vision-based gesture recognition for an Augmented Reality interface**.** 4th IASTED International Conference on Visualization, Imaging, and Image Processing, pp. 766-771, 2004.