UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO Y CONSTRUCCIÓN DE INTERFAZ HUMANO ROBOT UTILIZANDO GESTOS REALIZADOS CON LAS MANOS

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL ELECTRICISTA

HARDY EINAR FRANCKE HENRÍQUEZ

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR

MIEMBROS DE LA COMISIÓN:
HECTOR AGUSTO ALEGRIA
RODRIGO VERSCHAE TANNENBAUM

SANTIAGO DE CHILE
OCTUBRE 2007

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO
DE INGENIERO CIVIL ELECTRICISTA
POR: HARDY FRANCKE HENRÍQUEZ
FECHA: 18 DE OCTUBRE DE 2007
PROF. GUÍA: SR. JAVIER RUIZ DEL SOLAR

"DISEÑO Y CONSTRUCCIÓN DE INTERFAZ HUMANO ROBOT UTILIZANDO GESTOS REALIZADOS CON LAS MANOS"

En el presente trabajo de título, que se presenta para obtener el título de Ingeniero Civil Electricista, se diseña y construye un sistema de capaz de interpretar una serie de gestos realizados con las manos. Este sistema hace uso de una cámara web con la que se obtienen imágenes de manera periódica, que son procesadas y analizadas individualmente para poder luego interpretar el gesto ejecutado mediante métodos estadísticos.

El problema se ha separado en dos etapas distintas: la localización inicial de la(s) mano(s) en una imagen cualquiera y la posterior interpretación del gesto que se realiza. Para este fin se hace uso de varios subsistemas dentro del campo de la visión computacional, como lo son clasificadores estadísticos basados en 'Adaboost', seguimiento de objetos mediante 'Meanshift' y detección de piel con métodos adaptivos.

El sistema diseñado funciona de la manera siguiente: en una serie de imágenes obtenidas a través de una cámara se ejecuta un detector de caras, continuándose este proceso hasta que se logre encontrar alguna. Una vez detectada una cara se le aplica un algoritmo de seguimiento de objetos para conocer su ubicación dentro de la imagen a lo largo del tiempo, usándose además la información del color presente en esta cara para construir un modelo que identifique la piel. Este modelo de piel se aplica sobre todas las imágenes siguientes para detectar zonas que posean este color, y que por lo tanto tienen alta probabilidad de corresponder a alguna parte del cuerpo humano. Estas zonas de interés son analizadas en mayor profundidad, aplicando sobre ellas un detector de manos, entrenado en el marco de este trabajo de título y que hace uso de un procedimiento similar al del detector de cara. Al igual que en el caso de las caras, este detector se aplica sobre todas las imágenes de entrada hasta que se obtenga una detección positiva. Una vez detectada una mano se usa el mismo sistema de seguimiento empleado para las caras, y se aplica sobre esta zona otra serie de detectores. Estos nuevos detectores cumplen la función de identificar cual es el gesto que el usuario está realizando en ese momento, basados en un diccionario gestual previamente definido y que consta de cuatro tipos diferentes para este trabajo. Los resultados de estos detectores son finalmente analizados por un clasificador que finalmente toma la decisión de cual gesto está realizando el usuario. Este proceso se continúa efectuando iterativamente mientras se tenga un vídeo de entrada al sistema.

Los resultados obtenidos en la etapa de detección de los gestos son para todos los casos superior al 60%, obteniéndose para un caso detecciones correctas mayores al 95%. El sistema final logra un 69% de éxito en la clasificación de los gestos conocidos por él, debiéndose la mayor cantidad de errores a confusiones entre gestos parecidos entre si. La totalidad del algoritmo puede llegar a funcionar a una velocidad promedio de 6,6 [cuadros/seg.], lo que se considera un número aceptable para su uso sin molestias para el usuario del mismo.

A mi familia y amigos.

Índice general

Índice de figuras

Capítulo 1
Introducción

1.1. Introducción a la memoria

La masificación de los computadores a finales del siglo XX trajo consigo muchos cambios que eran imposibles de prever en un comienzo de la fabricación de estas máquinas: nuevas formas de resolver problemas, de comunicarse, de entretenerse; una nueva forma de relacionarse con el mundo. La ubicuidad de estos artefactos ha hecho que la relación con ellos sea un asunto normal y cotidiano para una gran cantidad de personas. Por la gran frecuencia en que se realizan este tipo de interacciones ente personas y máquinas, y dado que sólo se puede esperar que estas interacciones sean aún más comunes en el futuro, es que el modo en que se realiza esta comunicación sea hoy en día sujeto de amplio estudio en el mundo científico y comercial. Se busca que en el futuro sea posible lograr una comunicación más natural con las máquinas, logrando que éstas se adapten a los modos de comunicación propios de las personas, evitando así desviar la atención hacia la forma de la interacción en vez de la tarea a realizar. Este requerimiento es especialmente necesario en el campo de la robótica, en que si se desea una naturalidad en su manejo y explotar de mejor forma sus capacidades se hace necesario imitar la comunicación que realizan comúnmente las personas entre si. Es por esto que muchas investigaciones se han centrado en lograr que un computador pueda entender órdenes verbales hechas por un usuario humano y transmitir por la misma vía información relevante. Otra forma común de interaccionar es la que se produce a través de los gestos que se pueden realizar con distintas partes del cuerpo, y

que son capaces de transmitir información muy compleja al igual que la comunicación oral, como por ejemplo con el lenguaje de señas de lo mudos.

Gesto se define como el movimiento de alguna parte del cuerpo con la intención de transmitir información, conciente o inconscientemente, o interactuar con el ambiente. Debido a lo amplio de esta definición, muchas definiciones alternativas han surgido para señalar lo que es un gesto. En algunos casos, por ejemplo, se separan los elementos según la función del gesto: aquella realizada para comunicar información significativa, la hecha con el fin de manipular algún objeto y aquella hecha para obtener y transmitir información por medio del tacto. Otra separación que se hace comúnmente es la de categorizar los gestos según su componente temporal; así, se llama gesto estático a aquél en que el significado del gesto viene dado solamente por su forma, mientas que un gesto dinámico considera también la manera en que el gesto cambia durante el tiempo dentro de su significado. Una clasificación distinta es la que se realiza según el tipo de gesto que se hace, como por ejemplo pantomimas, lenguaje de signos o gesticulación (entre otros); existiendo también otras subdivisiones en cada una de las categorías. Aunque los gestos realizados de manera involuntaria corresponden a casi un 90% del total (incluso gente ciega gesticula al hablar entre ellos y las personas también tienen la tendencia a gesticular al hablar por teléfono), éstos se suelen dejar fuera de la investigación en interacción humano-computador (HCI por sus siglas en inglés) y se favorecen otros gestos que sean menos ambiguos y espontáneos, como son los gestos aprendidos. Estos tipos de gestos contienen una carga semántica más clara y parecieran ser más apropiados para interacciones enfocadas en controlar un sistema como es el objetivo en HCI.

Uno de los problemas que surgen con la comunicación gestual es que los mensajes pueden ser expresados mediante ellos de muchas formas. Por ejemplo, una emoción como tristeza puede comunicarse a través de expresiones faciales, posición de la cabeza (cabizbajo), músculos relajados y movimientos lentos. De manera similar, un mensaje como "detenerse" puede gesticularse simplemente como una mano con la palma hacia delante del cuerpo, o con un movimiento exagerado de ambas manos sobre la cabeza. En general, existen muchos mensajes asociados con un gesto (los gestos son ambiguos), y además existen muchos gestos asociados con un concepto (los gestos no son únicos). Aún más, los gestos varían entre culturas y entre individuos, y una misma persona cambia la ejecución del gesto en cada nueva ejecución.

Para poder realizar de mejor manera la comunicación es necesario, no sólo seguir el movimiento del usuario, si no que interpretar ese movimiento para reconocer gestos que carguen algún significado. Aunque seguir el movimiento de manos y cara puede llegar a ser útil para enviar directamente información a un sistema o controlarlo de manera básica, la comunicación

generalmente se realiza en un nivel de abstracción más alto. Dado que el cuerpo humano puede expresar una gran cantidad de gestos, cabe preguntarse que es lo que corresponde medir. Expresiones faciales y gestos hechos con las manos transmiten gran cantidad de información, por lo que los trabajos realizados en el tema suelen centrarse en ellos. Algunas características más sutiles también han sido usadas, como tensión muscular y dilatación de las pupilas.

Este trabajo de memoria busca aportar en el campo de la HCI, en particular en el campo relacionado a la comunicación mediante gestos hechos con las manos. Dentro de este terreno el trabajo se centrará en aquellos gestos que son realizados de manera conciente para interactuar con una máquina de manera básica.

A lo largo del informe aquí presente se irán mostrando las distintas alternativas para la resolución del problema de la identificación de gestos, señalándose las dificultades presentes y algunos de los métodos considerados como estado del arte en la visión computacional y que encuentran una de sus aplicaciones en esta área. Se finalizará con la implementación de un sistema que cumpla los objetivos que se plantean en la siguiente sección.

1.2. Objetivos

A continuación se detallan los objetivos, tanto generales como específicos, que se busca cumplir con la realización de esta memoria.

1.2.1. Objetivos generales

El propósito de este trabajo es implementar un programa computacional que permita realizar una interacción ente una persona y un computador mediante gestos visuales que la máquina interpretará de acuerdo a un diccionario gestual preestablecido. El programa debe funcionar en tiempo real con el fin de poder emplearse en un sistema que tenga una salida pronta hacia el usuario y éste no se vea afectado por retrasos que hagan poco natural la interacción.

Para esto se analizarán los métodos existentes actualmente en el campo de la visión computacional y se elegirá un enfoque que permita resolver la problemática planteada.

1.2.2. Objetivos específicos

Los objetivos específicos de este tema de memoria son los siguientes:

1. Estudiar lo realizado en trabajos similares, tanto aquéllos hechos en temas de memoria dentro del Departamento de Ingeniería Eléctrica como en aquellos efectuados externamente y cuyos resultados hayan sido publicados.
2. Hacer un catastro de las imágenes y videos disponibles para realizar el entrenamiento de los algoritmos, y en caso de ser necesario crear una nueva base de imágenes para ser usadas en esta tarea.
3. Proponer estructura para detector y clasificador de manos usando 'Adaboost'.
4. Implementar estructura propuesta y usarla dentro de una interfaz gráfica construida para ese propósito.
5. Hacer ajustes necesarios al algoritmo de acuerdo a los resultados obtenidos y los deseados hasta obtener lo requerido en cuanto a tasas de detección, clasificación y velocidad del sistema en su conjunto.

1.3. Estructura de la memoria

El presente trabajo está dividido en seis capítulos, incluyendo el actual. Los capítulos siguientes se resumen a continuación.

En el Capítulo 2 se presenta la definición del problema de detección de manos y gestos, con el fin de mostrar la problemática a resolver dentro de esta memoria. Dentro de este capítulo se pretende describir la razón de la complejidad de este problema en relación con otros problemas de detección de objetos o figuras en vídeos. Además se indicarán algunas de las distintas soluciones que se le han dado a este tema con una revisión de la bibliografía existente en el tema.

En el Capítulo 3 se muestran los antecedentes generales que es necesario conocer para abordar este problema. Se presentan en este capítulo el marco teórico para un sistema de clasificación estadístico basado en 'Adaboost', para el seguimiento con 'Meanshift' y para la detección de piel.

En el Capítulo 4 se indica en detalle el diseño y la implementación realizada para resolver la problemática planteada.

En el Capítulo 5 se presentan los resultados obtenidos con el desarrollo realizado en esta experiencia, además de compararlo con sistemas similares cuyos resultados hayan sido publicados.

Luego, en el Capítulo 6 se dan a conocer las conclusiones de este trabajo y algunas recomendaciones para el trabajo futuro en el tema.

Por último, se presenta a manera de Anexo el artículo realizado durante la ejecución de este trabajo y que ha sido aceptado en la conferencia PSIVT 2007 (2007 IEEE Pacific-Rim Symposium on Image and Video Technology).

Capítulo 2

Definición del problema

El uso de las manos como dispositivo de entrada es una forma atractiva de proveer formas más naturales de interacción humano-computador (HCI). Su gran plasticidad permite transmitir una amplia cantidad de gestos, debido a las múltiples formas que puede tomar gracias a sus 27 grados de libertad [46] (ver Figura 1). Pero por el mismo motivo, la obtención de la información del gesto que se realiza no es trivial por lo deformable que es la mano.

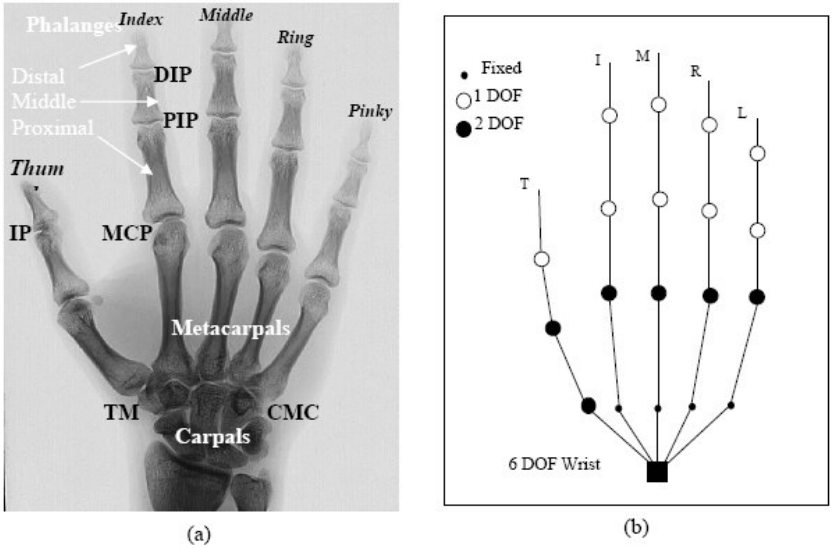


Figura 1: (a) Anatomía de la mano. (b) Grados de libertad de cada juntura de la mano. Imagen extraída de [47].

Actualmente existen dos maneras de obtener información de la posición de las distintas partes de una mano: por medio de dispositivos físicos que tengan sensores que detecten la posición y orientación de las manos, o por medio de visión computacional. El primero de ellos posee la ventaja de entregar resultados muy precisos de la ubicación de cada una de las partes móviles, que luego puede ser interpretada por un computador; pero requiere equipamiento costoso, largos procesos de calibración y puede ser molesto para el usuario, por lo que su aplicación fuera de los laboratorios es limitada. Por otro lado, los métodos de visión computacional tienen el potencial de proveer de soluciones más naturales, por lo que se han hecho considerables esfuerzos en investigación para poder usar las manos en HCI de esta forma. Distintas alternativas han surgido dentro de esta misma categoría, diferenciándose en los siguientes elementos: número de cámaras usadas, velocidad del algoritmo, requisitos para su uso (tanto del usuario como del ambiente, como por ejemplo usar guantes de un color particular o que el fondo no presente tonalidades parecidas a la piel), las características usadas para el reconocimiento (bordes, siluetas, momentos, histogramas, etc.), si la representación de la mano será bidimensional o tridimensional y finalmente si el tiempo será considerada como una variable dentro del sistema. Debido a que el enfoque de este trabajo está en lograr un sistema que funcione en tiempo real[1] con la menor cantidad de restricciones posibles, no se considerarán en el estudio sucesivo aquellos sistemas que usen modelos tridimensionales (que requieren una alta capacidad de cómputo y por lo tanto no funcionan en tiempo real) ni aquellos que impongan condiciones muy estrictas para su uso.

A continuación se detallarán algunas de las soluciones planteadas para resolver el problema del reconocimiento de gestos mediante visión computacional. Esta revisión no pretende ser un compendio completo de lo hecho en el campo del reconocimiento de gestos, si no que más bien una muestra de algunos de los sistemas más destacados y actuales en este campo. El problema se dividirá en dos partes: la localización de la mano dentro de la imagen de entrada del sistema, y el reconocimiento del gesto como tal. Luego en la sección 'Estado del arte' se presentarán los resultados obtenidos en aquellos trabajos publicados que contengan enfoques similares al desarrollado en esta memoria, y que por lo tanto son atingentes a la misma.

[1] Tiempo real: la definición de este término en la literatura es un poco vaga, aunque varios trabajos la entienden como la capacidad de un sistema de visión computacional de funcionar al menos a 30 [cuadros/seg] (fps), variando este número según el tamaño de las imágenes o 'frames' usadas. En este trabajo se usará una definición más general y se hablará de tiempo real si el sistema funciona a una velocidad suficiente para que no exista un retardo molesto (fijado arbitrariamente en medio segundo) en la respuesta del sistema.

2.1. Localización de manos

La segmentación por color de piel es un método comúnmente usado para la localización de manos, debido a que el tiempo de procesamiento requerido suele ser muy bajo, permitiendo dejar libres recursos que pueden ser necesarios posteriormente. Algunos trabajos que hacen uso de este enfoque son [48], [49], [50] y [51]. El principal problema de este tipo de métodos es que no necesariamente todo lo detectado como piel corresponda a una mano, pudiendo tratarse de otra parte del cuerpo (caras o brazos por ejemplo) o de detecciones de piel erróneas (objetos con color parecido, como madera). Algunos trabajos ignoran este problema imponiendo como restricción que la mano sea el único objeto con color piel en la imagen, mientras que otros aplican algún procesamiento extra con el fin de realizar la discriminación, por ejemplo detectando caras y considerando aquellas regiones de piel que no corresponden a la cara como manos [52]. Otros métodos hacen uso de la substracción de fondos [53], en caso de que estos sean estáticos, o de modelos adaptivos de fondo en caso contrario [54]. Algunos estudios también usan cámaras infrarrojas calibradas a la temperatura del cuerpo humano para encontrar las manos [55], pero al igual que en el caso de la detección de piel, procesamientos posteriores son requeridos para eliminar zonas no deseadas, además de presentarse la dificultad extra de que estas cámaras son mucho más costosas que las regulares.

Otra forma de enfrentar el problema es el usar métodos de clasificación de objetos para localizar la(s) mano(s). Estos buscan una serie de características en distintas regiones de la imagen para determinar si la zona en cuestión contiene una mano o no. Debido a que generalmente las subimágenes a revisar son muchas, y las características a analizar son también numerosas si se desea una detección confiable, con métodos convencionales este enfoque no se podría ejecutar en un tiempo razonable. Sin embargo, gracias al trabajo desarrollado en [11], esto ha cambiado gracias al uso de cascadas de clasificadores 'Adaboost', los cuáles serán explicados en detalle en el 0. Estos clasificadores pueden funcionar muy rápido sobre imágenes al descartar zonas que no contengan el objeto que se desea encontrar, en este caso manos, de manera eficiente. En [56] se hizo uso de este sistema para detectar manos en distintas poses con buenos resultados, lo que luego fue aplicado en un programa de HCI [57]. En [58] también se empleo un sistema similar, con la diferencia de que en este caso la localización de la mano está integrada junto con el sistema de la identificación del gesto. Esto se realiza a través de un árbol, en que se comienza detectando la presencia o no de manos en la raíz y a medida que se avanza hacia las hojas de éste se va

acotando el tipo de gesto que se realiza. La desventaja de este método es que requiere un gran número de imágenes de entrenamiento de la clase que se desea detectar, por lo que el proceso de recolección de estas imágenes puede ser un trabajo arduo. Pero una vez en funcionamiento, un buen sistema de este tipo debería presentar un error en la detección mínimo y un tiempo de procesamiento relativamente bajo.

Otra característica usada para la localización de manos es la detección de "primitivas de dedos" [59], en que se buscan en la imagen características que son propias de los dedos, como por ejemplo color uniforme, líneas paralelas y puntas de los dedos curvas. Una ventaja de este sistema es su invariabilidad a rotaciones y traslaciones, pero presenta un alto número de detecciones erradas en comparación con las técnicas anteriores que le resta viabilidad práctica.

Como se puede ver, existen distintos enfoques para intentar resolver este problema, lo que habla de que la detección de manos es aún un problema abierto que no tiene una respuesta completamente satisfactoria para todos los casos. Por lo mismo, dependiendo de la aplicación que se desea y de las restricciones que se consideren permisibles se debe evaluar para cada situación que sistema conviene emplear.

2.2. Detección de gestos

Entendiendo que ya se ha resuelto la dificultad inicial de encontrar la mano en la imagen de entrada, se puede enfrentar el verdadero problema que se busca resolver, cual es identificar el gesto que se está realizando. Para esto primero hay que definir que es lo que se entenderá por gesto dentro del marco del trabajo y acotarlo a las necesidades y posibilidades del mismo, ya que construir un sistema que abarque absolutamente todo lo que se entiende por gesto (y que se explicitó en el capítulo introductorio de esta memoria) es una tarea demasiado ambiciosa e irrealizable actualmente. Una separación básica es la que se da entre gestos estáticos, en que el usuario asume cierta posición sin movimiento de las manos, o si es que hay movimiento este no es considerado como una variable por el sistema; y los gestos dinámicos, en que la variable temporal (movimiento) si es usada. El análisis siguiente se hará considerando esta subdivisión, presentando algunos de los trabajos más relevantes en estas dos áreas. Hay que hacer notar que todos los trabajos actuales consideran un vocabulario gestual limitado, es decir, no distinguen entre todas las posturas que puede llegar a presentar una mano si no que eligen un número limitado de gestos a analizar. Este número limitado se mueve en los trabajos actuales entre un par

hasta cerca de una treintena de gestos (en el caso de sistemas que buscan interpretar el lenguaje de señas de los mudos).

2.2.1. Detección de gestos estáticos

Uno de los primeros sistemas de detección de gestos estáticos relativamente existoso es el presentado en [60], en donde se hace uso de histogramas de orientación. La idea es calcular los bordes de la imagen que contiene la mano, luego usar esa información para calcular la orientación en cada punto. Con estas orientaciones se construye un histograma que es comparado, usando la distancia euclidiana, con el diccionario de gestos que se haya construido inicialmente. La menor distancia obtenida indicará el gesto que se está haciendo, si corresponde a alguno de la base de datos. La ventaja de este sistema es su sencillez, que lo hace rápido de calcular. Sin embargo, presenta en algunos casos dificultades para clasificar gestos en una misma categoría a pesar de ser casi iguales o clasifica en una misma clase elementos diferentes, además de usar la simplificación de que se trabaja con imágenes que contienen fondos simples (es decir, de un solo color).

Un método similar al anterior es el uso de histogramas con contornos de la mano, como el usado en [61], o el de histogramas de puntos con una distancia mayor a cierto umbral hasta su centro de masa [50]. Aunque estos métodos más recientes presentan varias mejoras sobre la implementación inicial del uso de histogramas, siguen teniendo en mayor o menor medida problemas con la clasificación de los gestos ya que varios gestos diferentes pueden llegar a tener un mismo histograma.

Otros métodos recientes hacen uso también de la silueta, pero sin ocupar histogramas. En [62] la silueta de la mano es muestreada y esos puntos pasados al espacio de la frecuencia a través de la transformada de Fourier. A estos puntos transformados se les llama descriptores de Fourier, y tienen la propiedad de ser aproximadamente invariantes a rotaciones, traslaciones y cambios de escala. Estas propiedades hacen a este método muy útil en clasificación de gestos, aunque el costo computacional de calcular la transformada de Fourier para cada punto y el hecho de que no todo gesto puede ser bien definido en base a su contorno le juega en contra.

También, como ya se mencionó en la sección de 'Localización de manos', clasificadores en cascada de tipo 'Adaboost' también se han aplicado para clasificar gestos estáticos. Algunos de los trabajos que siguen esta línea son los presentes en [58], [63] y [9]. Una de las ventajas de este método es que no se requieren simplificaciones del objeto a detectar, en este caso la mano, como

sí se hace cuando se usa el contorno o imágenes binarias. Sin embargo presenta la desventaja de no ser invariante a rotaciones (si lo es a traslaciones y cambios de escala), por lo que si se desea detectar un gesto girado es necesario entrenar un clasificador con imágenes giradas en ese ángulo, lo que puede ser un proceso bastante engorroso.

2.2.2. Detección de gestos dinámicos

Los gestos dinámicos son básicamente una serie de gestos estáticos que cambian durante el tiempo. Por esta razón se usan los mismos métodos descritos en la sección anterior, pero se le agregan algoritmos que tomen en consideración también el tiempo como una de sus variables. La forma más usada para lograr ésto es ocupar un proceso llamado modelo oculto de Markov (HMM). Una propiedad de un proceso de Markov es que la probabilidad condicional del evento actual depende solamente del *j*-ésimo evento más reciente. Si el estado actual depende solamente del estado anterior más reciente, entonces se denomina como proceso de Markov de primer orden. Esta idea es útil en el caso de las manos, al considerar la posición y orientación de ellas a lo largo del tiempo. En el caso de que los estados del proceso no sean observables directamente, si no que lo son indirectamente a través de variables influenciadas por ellos, se habla de un modelo de Markov oculto.

Un HMM consiste en un número de estados, cada uno de los cuáles tiene asignado una probabilidad de transición de un estado a otro. Durante el tiempo, las transiciones de estados ocurren estocásticamente. En instantes de tiempo discretos, el proceso se encuentra en uno de sus estados y genera una observación de acuerdo a la función aleatoria que tenga asociada ese estado. Cada cambio de estado se define con un par de probabilidades: la probabilidad de transición, que da la probabilidad de que se realice esa transición, y la probabilidad de salida, que define la probabilidad condicional de emitir una salida dado un estado. Para describir matemáticamente un HMM se sigue la siguiente notación:

- T, largo de la secuencia observada.
- $A = \{a_{ij}\}$, es la matriz de transición de estado: $A = p(q_j / q_i)$, donde a_{ij} es la probabilidad de transitar del estado q_i al espacio q_j.
- $Q = \{q_1,...,q_N\}$, son los estados del modelo.

- $V = \{v_1,...,v_k\}$, arreglo de los posibles símbolos de salida.
- $B = \{b_{ij}\}$, es la matriz confusión, donde b_{ij} es la probabilidad de generar el símbolo v_i en el estado x_j: $B = p(v_i / x_j)$
- $\Pi = \{\pi_i\}$, es el vector de probabilidades inicial.
- $\lambda = \{A, B, \Pi\}$ es el conjunto completo de parámetros del modelo.

Un ejemplo de un HMM, usando la notación presentada, se muestra en la Figura 2. La estructura global de un modelo oculto de Markov aplicado en la clasificación de gestos dinámicos se construye con una conexión de varios HMM (λ_1, λ_2,…, λ_M) en paralelo. Aquí, M corresponde al número total de gestos que se reconocen y λ_k es un HMM construido para el gesto k-ésimo.

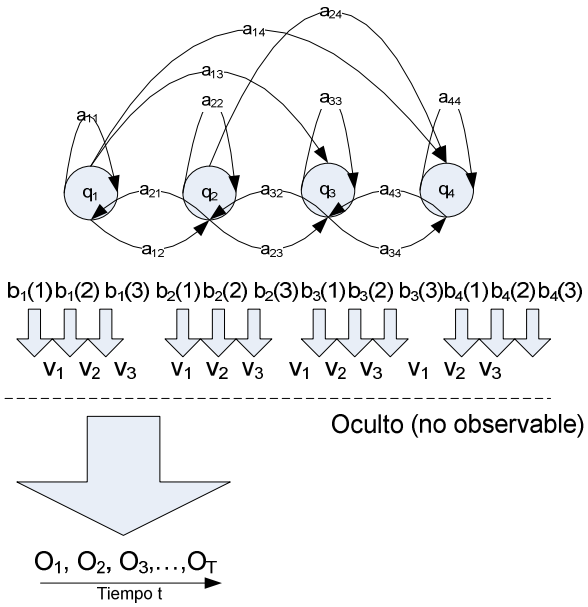


Figura 2: Diagrama de un HMM de 4 estados.

Uno de los primeros trabajos en usar HMM para identificación de gestos fue [65], donde el objetivo era identificar distintos tipos de golpes de un tenista. El vector de entrada correspondía en este caso a una imagen binarizada de la pose del jugador. A partir de este trabajo, HMM ha comenzado a ser la solución preferida, si es que no la única, para problemas de detección de gestos dinámicos. Algunas de estas aplicaciones se pueden encontrar en [66][48][49][65][51].

HMM ha probado ser de gran valor en analizar sistemas reales, aunque no por eso está exenta de desventajas. Una de ellas es que considera que cada estado depende solamente de sus predecesores, y que esa dependencia es independiente del tiempo, lo que en algunos casos es una sobresimplificación del problema real. Pero el hecho de que HMM sea la solución preferida da a entender que sus dificultades son menores, o al menos que es mejor que otros métodos de este tipo.

2.3. Estado del arte

En esta sección se mostrarán los resultados publicados en sistemas de detección de gestos que contengan similaridades con el desarrollado en esta memoria, por lo que se consideran útiles como punto de comparación.

Una de las principales dificultades para comparar resultados en el área de detección y clasificación de gestos es que no existe una base de datos común en que se puedan realizar estas comparaciones. Así, la base de prueba a usar queda a criterio de cada desarrollador y rara vez es publicada, por lo que no es sencillo declarar si un sistema es mejor que otro. El cotejo se complica si se desea tomar en consideración otras variables que se puedan considerar de interés, como la velocidad del algoritmo ('frame rate'), porque está información rara vez se presenta.

De todas formas es interesante saber cuáles son los resultados obtenidos en otras investigaciones, aún cuando la comparación sea subjetiva, para tener un marco de referencia a la hora de evaluar lo obtenido.

Se muestran en esta sección los resultados de cinco métodos que usan como método de detección alguna variante del clasificador 'Adaboost', debido a que este mismo clasificador es el que se usa para la aplicación realizada en esta memoria. Luego en el capítulo de 'Resultados y Análisis' se compararán estos sistemas con el propuesto en esta memoria.

Los primeros resultados que se mostrarán corresponden a los logrados en [9], donde se entrenaron 2 clasificadores de formas distintas: el primero (Protocolo 1) corresponde a sólo usar imágenes con el gesto deseado sobre un fondo simple, es decir, de una sola tonalidad; el segundo (Protocolo 2) incluye imágenes con fondos complejos (varias tonalidades). Los resultados mostrados en la Tabla 1 corresponden a la detección de la mano usando el clasificador entrenado para ese gesto particular, por ejemplo "es gesto A usando reconocedor de gesto A"; mientras que

en la Tabla 2 se muestran los resultados de la clasificación de los gestos ("es gesto A usando reconocedores de gestos A, B,…, Y"), donde la clasificación se realiza considerando como

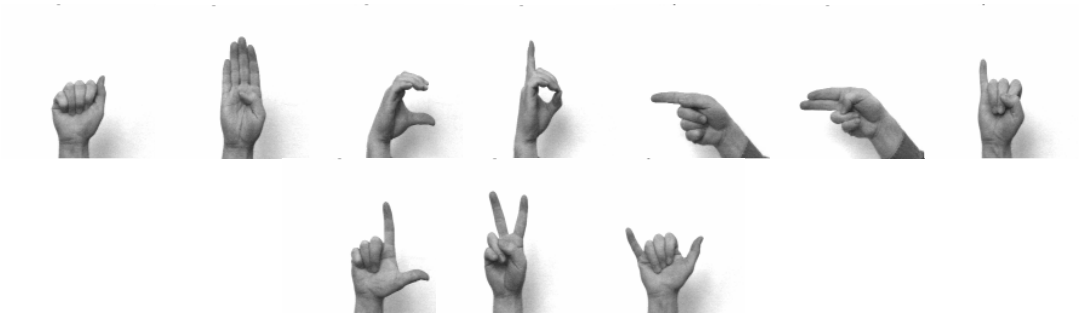


Figura 3: Gestos detectados por el sistema en [9]. De izquierda a derecha: 'A', 'B', 'C', 'D', 'G, 'H', 'I', 'L', 'V' e 'Y'.

	Fondo Uniforme		Fondo Complejo	
Gesto	Protocolo 1 [%]	Protocolo 2 [%]	Protocolo 1 [%]	Protocolo 2 [%]
A	100,00	100,00	91,67	100,00
B	93,75	100,00	75,00	100,00
C	96,88	100,00	66,67	93,75
D	100,00	100,00	87,50	100,00
G	100,00	100,00	87,50	100,00
H	100,00	100,00	100,00	100,00
I	100,00	100,00	95,83	93,75
L	100,00	100,00	100,00	100,00
V	96,77	100,00	54,17	100,00
Y	96,88	100,00	62,50	87,50
Promedio	98,40	100,00	82,10	97,50

Tabla 1: Resultados de la detección de gestos obtenidos en [9]. Protocolo 1 corresponde a entrenamiento sin imágenes con fondo complejo y Protocolo 2 sí incluye fondos complejos.

	Fondo Uniforme		Fondo Complejo	
Gesto	Protocolo 1 [%]	Protocolo 2 [%]	Protocolo 1 [%]	Protocolo 2 [%]
A	100,00	100,00	100,00	100,00
B	93,75	93,75	93,75	93,75
C	93,75	93,75	75,00	93,75
D	93,75	84,38	62,50	81,25
G	96,88	100,00	50,00	68,75
H	84,38	90,63	87,50	87,50
I	84,38	90,63	56,25	62,50
L	84,38	96,88	37,50	75,00
V	87,10	96,77	56,25	87,50
Y	81,25	81,25	25,00	62,50
Promedio	89,97	92,79	64,38	81,25

Tabla 2: Resultados de la clasificación de gestos obtenidos en [9]. Protocolo 1 corresponde a entrenamiento sin imágenes con fondo complejo y Protocolo 2 sí incluye fondos complejos.

verdadera la detección con mayor confidencia de todas. Estos resultados, como se puede ver, no indican las detecciones erróneas que realizó cada detector. Además, al parecer las imágenes de prueba fueron obtenidas de un subconjunto de las que se usaron para entrenar lo que le restaría validez al 'test'. Por otro lado, en este caso sí se cuenta con la base de datos en que fue realizada esta prueba lo que permite realizar una comparación más directa con lo obtenido en esta memoria.

Otro trabajo que hace uso de 'Adaboost' en la detección de manos es [56], cuyos resultados se observan en la Figura 5. Éste fue uno de los primeros trabajos en aplicar clasificadores 'Adaboost' en cascada para la detección de manos. Se evalúo la detección en 5 diferentes gestos (mostrados en la Figura 4) con imágenes obtenidas con una cámara montada sobre el usuario. Para la evaluación se usaron 200 imágenes que no contenían manos y que fueron obtenidas algunas de Internet y otras por los mismos investigadores para calcular la tasa de falsos positivos del detector ('false positive rate'), pero no se explicita como se midió la tasa de detección ('detection rate'). En el caso de este trabajo no se hace una clasificación de los gestos, sólo se aplica la parte de detección, por lo que no hay resultados en esta categoría.

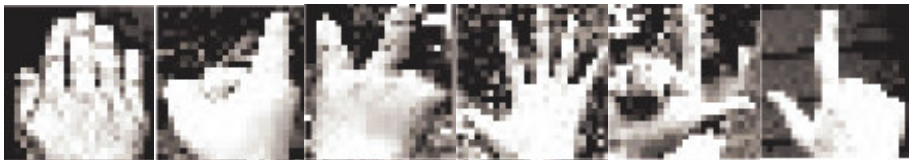


Figura 4: Gestos detectados por el sistema en [56]. De izquierda a derecha: 'closed', 'sidepoint', 'victory', 'open', 'Lpalm' y 'Lback'.

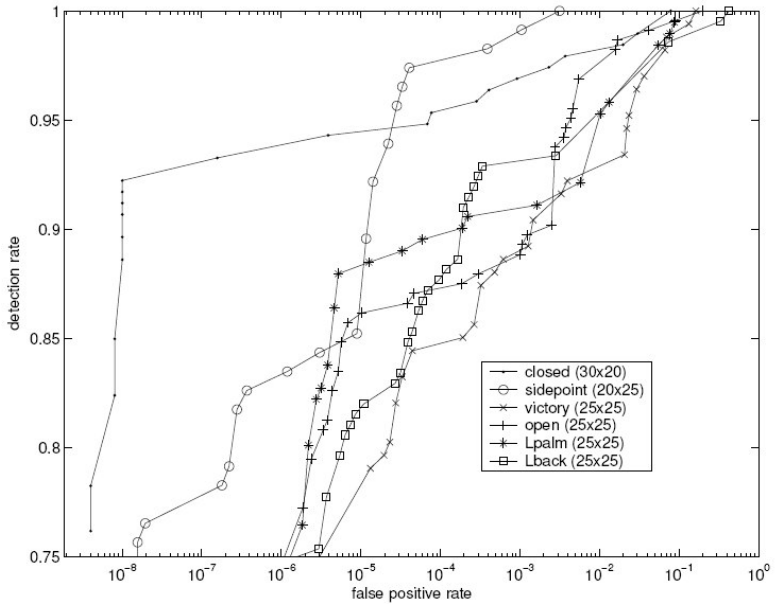


Figura 5: Resultados de la detección gestos obtenidos en [56].

En [58] se usaron clasificadores 'Adaboost' conectados en forma de árbol de manera de detectar y clasificar las manos de una manera más eficiente, según el esquema mostrado en la Figura 6. Para probar este algoritmo se usaron 2509 imágenes obtenidas de distintos vídeos en la etapa de detección y 900 en la etapa de clasificación, con los que se obtiene un 99,8% y 97,4% de tasa de éxito respectivamente. Los resultados presentados no indican por separado los resultados de la clasificación para cada gesto, si no que se enseñan para el sistema total. De hecho, no se indica en ningún momento cual es el número de gestos que el sistema detecta y clasifica ni cuales son. Tampoco se señala el número de falsos positivos del sistema. Aunque las tasas de detección y clasificación son sorprendentemente altas, se señala en el mismo trabajo que estas fueron obtenidas en imágenes con fondos simples, por lo que no se puede esperar resultados tan precisos en imágenes sin esa restricción.

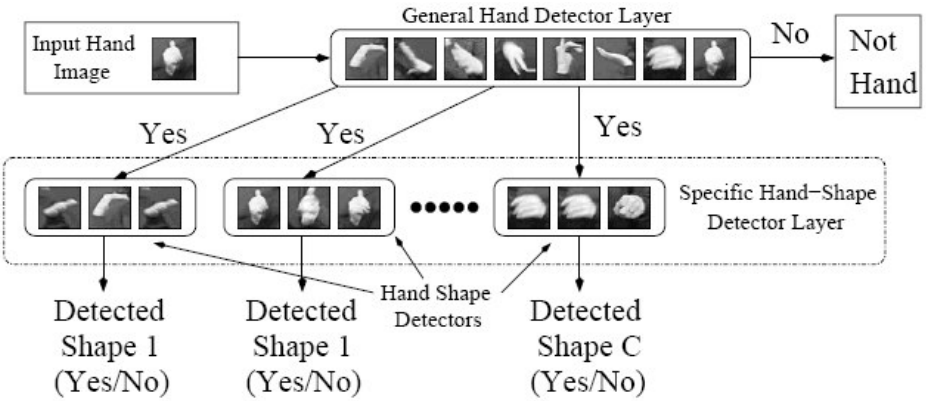


Figura 6: Esquema de detección y clasificación de gestos propuesto en [58].

En [63] se publica un trabajo que, entre otras cosas, crea un sistema de control básico a través de gestos. Aquí se entrenaron 4 detectores para 4 gestos diferentes, los que se muestran en la Figura 7. Los resultados se muestran en la Tabla 3, donde se señalan también el tiempo promedio de procesamiento por imagen. La clasificación se realiza corriendo estos 4 detectores en paralelo y decidiendo luego cual gesto es el correcto a través de un clasificador que no es especificado. La clasificación, según se declara en [63], presenta algunos errores entre gestos similares, pero no se indican resultados numéricos. Las bases de entrenamiento y de prueba (que contiene un total de 100 imágenes) corresponden a imágenes con fondos simples, en que no existe otro elemento en ellas aparte de la mano.

Figura 7: Gestos detectados por el sistema en [63]. De izquierda a derecha: dedo índice y medio, palma, puño y dedo meñique.

Gesto	DR [%]	FP	Tiempo promedio por imagen [seg]
Dedo índice y medio	100,00	29	0,031
Palma	90,00	0	0,019
Puño	100,00	1	0,028
Dedo meñique	93,00	2	0,025

Tabla 3: Resultados de la detección de gestos logrados en [63].

El último trabajo expuesto en esta sección es el publicado en [64], en que se también se hace uso de varios clasificadores 'Adaboost' conectados en cascada. En este caso el interés era mostrar las diferencias en las tasas de detección para varios clasificadores entrenados con imágenes rotadas en distintos ángulos, por lo que se hace uso de un solo gesto (igual al gesto 'palma' mostrado en la Figura 7). La idea de realizar esto viene del hecho de que un clasificador 'Adaboost' que hace uso de características rectangulares (conceptos cuyo significado será explicado en el capítulo siguiente) no detecta objetos rotados, por lo que una forma de detectarlos es entrenar varios clasificadores con las mismas imágenes pero giradas en cierto ángulo. Los resultados de la Tabla 4 muestran que las tasas de detección son dispares dependiendo del ángulo de la rotación, y que aun considerando el mejor caso los resultados son inferiores a los otros trabajos descritos anteriormente. Sin embargo, esta prueba permite comprobar que no se pueden extrapolar los resultados obtenidos con imágenes en una orientación hacia otras rotadas, siendo necesario tal vez otro enfoque para resolver este problema.

Ángulo [°]	Número de imágenes de prueba	DR [%]	FP
-90	109	34,00	7
-60	104	62,50	9
-30	117	78,60	10
0	100	72,00	10
30	121	70,20	17
60	100	84,00	4
90	129	48,80	3

Tabla 4: Resultados de la detección de gesto en [64].

Capítulo 3

Antecedentes Generales

3.1. Clasificadores estadísticos

Una clasificación estadística es un procedimiento por el cual distintos elementos son agrupados entre ellos por medio de información cuantitativa obtenida de una o más características propias de los elementos y basados en conjuntos de entrenamientos previamente etiquetados [13]. En particular estos clasificadores pueden ser usados en imágenes, como en el caso de la detección de objetos. Esta detección suele hacerse entre 2 clases: la del objeto que se desea encontrar y el resto (o lo 'no-objeto'). Algunos ejemplos de clasificadores estadísticos son: redes neuronales, SVM y clasificadores tipo 'Adaboost'. Este trabajo hace amplio uso del sistema de clasificación de tipo 'Adaboost', que es el sistema más usado en la actualidad para resolver problemas de este tipo en imágenes debido a sus buenos resultados y bajo tiempo de procesamiento. Por lo mismo, se procederá a detallar el diseño y la implementación usada en esta memoria.

3.2. Clasificadores estadísticos basados en Adaboost

Dentro de este trabajo se hace uso de un algoritmo de detección de objeto que se basa en la utilización de las características de las imágenes. Este método es conocido como 'Adaboost', introducido inicialmente en [6] y generalizado en [7]. Debido a sus buenos

resultados en problemas de detección y clasificación de objetos en imágenes, pero por sobre todo a su alta velocidad de procesamiento, ha sido empleado ampliamente en este campo.

'Adaboost' es un sistema que construye una regla de clasificación final usando varios clasificadores menores, denominados "débiles" por su sencillez y escasa precisión. En solitario estos clasificadores débiles no constituirían un sistema de clasificación usable debido a su alta inexactitud, pero al usarlos en conjunto es posible construir un clasificador muchísimo más preciso. Los clasificadores débiles corresponden a reglas de clasificación simples y que entregan como resultado un valor de confianza, o confidencia, respecto a la predicción que están haciendo. Estos clasificadores son linealmente combinados en la forma indicada en la ecuación (1). Aquí cada $h_t(x)$ corresponde a un clasificador débil, mientras que T indica el número total de clasificadores usados y b corresponde al umbral de operación del clasificador.

$$H(x) = Signo\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x) - b\right) \qquad (1)$$

Los clasificadores débiles son aplicados sobre características, o 'features', dependiendo cada uno de solamente una característica. Siguiendo el modelo propuesto en [7] el dominio X sobre el que se trabaja (que en este caso es el dominio de la característica usada) es particionado en distintos bloques disjuntos $X_1,\ldots, X_n$, los cuales cubren por completo el dominio en cuestión. Cada clasificador tiene asociado un valor con una de las particiones, dependiendo ese valor solamente de a que bloque de la partición X_j corresponde. Por lo tanto la predicción del clasificador débil corresponderá exclusivamente a la partición X_j de la característica en que caiga, según se indica en (2), donde $f(x)$ indica la característica o 'feature'.

$$h(f(x)) = c_j \ni f(x) \in F_j \qquad (2)$$

Para cada clasificador débil su salida, i.e. el valor asociado a cada bloque de la partición (c_j), se calcula minimizando una función que refleja el error de entrenamiento. Esta salida es calculada entonces a partir del número de veces que la característica (o 'feature') cae dentro de este bloque de la partición (histograma), de la clase de los ejemplos, que llamaremos y_i, y de sus pesos, que denominaremos $D(i)$. El valor de c_j se obtiene, según [7], de la ecuación (3).

$$c_j = \frac{1}{2} \cdot \ln\left(\frac{W_{+1}^{j} + \varepsilon}{W_{-1}^{j} + \varepsilon}\right), \text{ donde } W_i^{j} = \sum_i D(i) = \Pr\left[f(x_i) \in F_j \wedge y_i = L\right], L = \pm 1 \qquad (3)$$

La salida que se obtiene para el valor c_j durante el entrenamiento es guardada en una 'look-up table' (LUT) con el fin de luego poder acceder a esta información de manera más rápida durante la ejecución del programa. Por lo tanto, cada clasificador débil estará definido por [8]:

$$h(x) = h_{LUT}[x] = LUT[index_{bin}(f(x))] \qquad (4)$$

Dos tipos de características son usadas dentro de esta experiencia por los clasificadores débiles: mLBP y características rectangulares, las cuales serán descritas a continuación.

3.2.1. mLBP

mLBP, 'Modified Local Binary Pattern', es también conocido como transformada Census o número de textura y fueron introducidos en [10]. Es una transformada no parámetrica usada ampliamente en el análisis de texturas en imágenes. Consiste en un arreglo ordenado que se obtiene luego de la comparación binaria de intensidades de los píxeles que están dentro de una vecindad de 3x3 en la imagen y la intensidad media de todos los píxeles de ese vecindario [1], como se puede apreciar en la Figura 8.

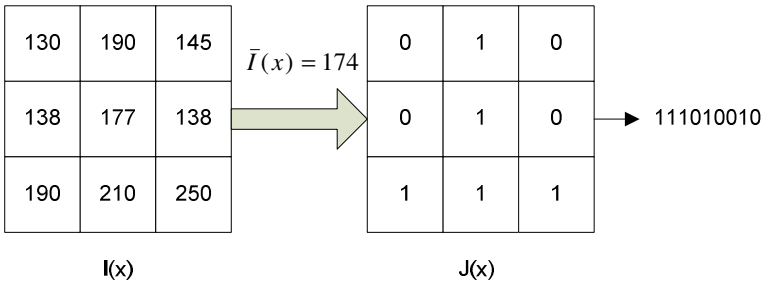


Figura 8: Ejemplo de cálculo de mLBP

Una propiedad interesante de las características mLBP es su invarianza ante cambios lineales en la iluminación: por definición de la transformada, cualquier cambio en la intensidad de los píxeles que se de en forma lineal no cambiará el resultado del mLBP. Además su simpleza computacional permite su cálculo sin grandes costos computacionales lo que favorece uso en aplicaciones que requieran funcionar en tiempo real.

3.2.2. Características rectangulares

Las características rectangulares fueron introducidas por Viola y Jones en [11]. Corresponden a la diferencia entre la suma de la intensidad de los píxeles entre distintas áreas rectangulares de la imagen. Usando la imagen integral [11] es posible calcular esta característica de forma muy rápida y de manera independiente a su tamaño. A modo de ejemplo se presenta la Figura 9, donde se presenta el uso de las características rectangulares en imágenes de manos. Aquí, dentro de las zonas blancas y negras se suman las intensidades de los píxeles, para luego restarse esos valores obtenidos entre si.

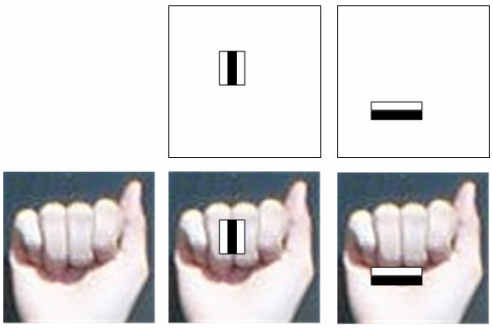


Figura 9: Ejemplo de características rectangulares aplicadas en manos.

3.2.3. Clasificadores 'Adaboost' en cascada

Como se explicó anteriormente, el uso de 'Adaboost' permite crear clasificadores "fuertes" a partir de varios clasificadores menores denominados "débiles". Con el fin de disminuir las falsas detecciones del objeto que se desea detectar se puede usar una combinación de estos detectores fuertes. Al usarlos uno después de otro, a manera de cascada, es posible construir un detector final mucho más eficiente que al tener un solo clasificador que agrupe todo. De esta forma se puede construir la cascada para que las primeras etapas de ella rechacen ventanas que no tengan similitud con el objeto a detectar, que ocurrirá en la mayoría de los casos, y las etapas posteriores se especializarán en descartar objetos que sí tengan más parecido.

Una cascada de clasificadores 'Adaboost' está compuesta por varias etapas, cada una de las cuales corresponde a un clasificador 'Adaboost'. Toda la cascada es un solo clasificador, integrando la información de los clasificadores presentes en cada etapa. Se define como una

cascada anidada C de M etapas a aquélla que es la unión de M clasificadores fuertes H_C^k 'Adaboost', que a su vez están compuestos de una serie de clasificadores débiles h_t^k. Debe tenerse en cuenta que en el caso de los clasificadores fuertes en la cascada, éstos corresponden a la combinación de todos los clasificadores antepuestos a él. Matemáticamente esto se expresa como:

$$C = \bigcup_{k=1}^{M} \left\{ H_C^k \right\} \qquad (5)$$

Donde cada H_C^k se define como:

$$H_C^k(x) = H_C^{k-1}(x) + \sum_{t=1}^{T_k} h_t^k(x) - b_k \qquad (6)$$

Con:

$$H_C^0(x) = 0 \qquad (7)$$

Donde $h_t^k(x)$ son los llamados clasificadores débiles, T^k es el número de clasificadores débiles en la etapa k, y b_k es un umbral predeterminado. Dada la configuración anidada de la cascada C, su salida está dada por:

$$O_C(x) = \begin{cases} signo\left(H_C^q(x)\right) & t.q. \quad H_C^k(x) \geq 0,\ k = 1,\ldots,q-1 \\ & \qquad \wedge \left(H_C^q(x) < 0 \vee q = M\right) \\ signo\left(H_C^1(x)\right) & t.q. \quad H_C^1(x) < 0 \end{cases} \qquad (8)$$

$$conf_C(x) = \begin{cases} H_C^q(x) & t.q. \quad H_C^k(x) \geq 0,\ k = 1,\ldots,q-1 \wedge H_C^q(x) < 0 \\ H_C^M(x) & t.q. \quad H_C^k(x) \geq 0,\ k = 1,\ldots,M \end{cases} \qquad (9)$$

Una representación gráfica de lo aquí expuesto se presenta en la Figura 10.

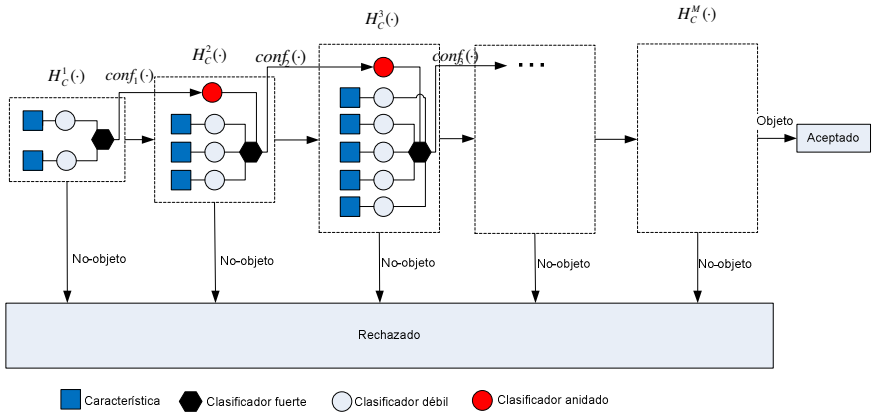


Figura 10: Clasificador anidado en cascada.

3.2.4. Bootstrapping

Durante el entrenamiento de un clasificador 'Adaboost' tanto imágenes que representan al objeto que se desea detectar como de no-objeto son necesarias para el funcionamiento del algoritmo que construye los clasificadores. Cada ventana de cualquier tamaño en cualquier imagen que no contenga el objeto es un ejemplo válido para el entrenamiento como no-objeto. Claramente incluir todos los posibles no-objetos en una base de entrenamiento no es posible por la infinidad de posibilidades. Para definir los límites de esta definición, imágenes que contengan no-objetos que parezcan similares al objeto a detectar deben ser seleccionadas. Esta selección comúnmente se resuelve usando el procedimiento de 'bootstrapping', que consiste en entrenar los clasificadores de manera iterativa, en cada ciclo aumentando el grupo de imágenes negativas en el entrenamiento al incluir imágenes que hayan sido incorrectamente clasificadas por el clasificador ya entrenado. Al entrenar un clasificador en cascada, el 'bootstrap' se puede usar en dos momentos: al comenzar el entrenamiento de cada nueva etapa de la cascada o para volver a entrenar una etapa recién construida. El primero se aplica solamente una vez por etapa, antes de comenzar el entrenamiento, mientras que el segundo puede ser aplicado varias veces durante el entrenamiento de cualquier etapa. Para obtener más información a como es implementado el procedimiento de 'bootstrapping' se refiere al lector a [8].

3.2.5. Active Learning

La creación de conjuntos de imágenes de entrenamiento positivas, es decir, de imágenes que corresponden al objeto que se desea detectar es un procedimiento que toma mucho tiempo, ya que un operador humano debe estar involucrado. Sin embargo, estos ejemplos para el entrenamiento pueden ser obtenidos de manera semi-automática mediante un procedimiento llamado 'active learning'. Esta técnica consiste en usar el mismo sistema que está siendo construido para generar la selección de los ejemplos para el entrenamiento.

En el presente trabajo se hace uso de 'active learning' para asistir en la construcción de una muestra representativa de imágenes de entrenamiento positivas, es decir, conjuntos de entrenamiento que capturen las condiciones reales de uso de la aplicación final. Para generar ejemplos para el entrenamiento de un detector para un gesto específico, el procedimiento consiste en solicitar al usuario que realice ese gesto en particular por un tiempo dado. Durante este tiempo

la mano del usuario será seguida por el sistema de 'tracking', y las ventanas que encierren la zona de interés serán automáticamente incorporadas al conjunto de entrenamiento para ese gesto. Si la mano es seguida por un par de minutos, y el usuario mantiene el mismo gesto a la vez que mueve la mano, miles de ejemplos pueden obtenerse con la variabilidad deseada (iluminación, fondo, rotación, traslación, etc.). Entonces, todas las ventanas clasificadas como positivas por el bloque de 'tracking' son tomadas como imágenes de entrenamiento positivas. Este procedimiento puede ser repetido por varios usuarios. Un operador humano sólo debe verificar que estas ventanas fueron detectadas correctamente, y corregir la alineación de éstas si fuese necesario. Luego, todas las ventanas son escaladas al tamaño en que se desea realizar el entrenamiento (usualmente 24x24 píxeles o tamaños similares, teniendo siempre cuidado en mantener la relación alto/ancho de la imagen).

En una segunda etapa, 'active learning' puede ser también empleado para mejorar un detector para algún gesto ya entrenado. En este caso, se hace uso del mismo procedimiento señalado con anterioridad (el usuario realiza el gesto y la mano es seguida), pero el detector para el gesto que ya ha sido entrenado está a cargo de generar los ejemplos para el entrenamiento. Así, cada vez que el detector de gesto clasifica una ventana proveniente desde el bloque de 'tracking' como un no-objeto (el gesto no es detectado), esta ventana es incorporada al conjunto de entrenamiento positivo para este gesto.

3.2.6. Modelo general de detección

El diagrama de bloques de un sistema de detección generalizado es presentado en la Figura 11. El sistema, como se puede ver en esa figura, contiene 5 bloques básicos. Los dos primeros se encargan de que se cumpla la condición de que el sistema sea capaz de detectar el objeto que se desea en diferentes escalas y posiciones dentro de la imagen que sea analizada. La etapa de análisis multiresolución se encarga de que sea posible la detección a diferentes escalas, al aplicar una reducción al tamaño de la imagen de entrada en un factor que se da como parámetro llamado 'factor de escalamiento' (típicamente se usa igual a 1,2). Esta reducción se realiza hasta que el ancho o el alto de la imagen sean más pequeños que el tamaño de la ventana de procesamiento. El módulo siguiente, que se indica en la Figura 11, es el de "Extracción de ventanas", en donde se obtienen ventanas de tamaño HxW a partir de cada una de las versiones escaladas de la imagen de entrada, obteniéndose todas las subimágenes posibles. La dimensión HxW de las ventanas que

son extraídas queda determinada en la fase de entrenamiento del clasificador, según el tamaño que tengan las imágenes que se usen esa etapa (por ejemplo, para la detección de caras generalmente este tamaño varía entre 19x19 y 25x25). No necesariamente se tienen que obtener todas las subventanas posibles en esta etapa, pudiéndose saltar algunos píxeles entre cada una de ellas con el fin de aumentar la velocidad del algoritmo. Luego de esta etapa se puede realizar algún tipo de preprocesamiento sobre las subventanas, con el fin de obtener invariabilidad a la iluminación u obtener algún otro tipo de propiedad sobre la imagen. Luego, las ventanas son analizadas por el clasificador 'Adaboost' anidado en cascada ('nested cascade of boosted classifiers'). Luego que todas las ventanas son clasificadas, en la etapa de detección de traslapes todas aquéllas clasificadas positivamente son analizadas y fusionadas para determinar la posición y tamaño final de las detecciones. Esta etapa es necesaria porque normalmente muchas detecciones del mismo objeto se obtendrán en diferentes escalas y posiciones. En este módulo los valores de las confidencias asociadas a cada detección son usadas para la fusión de las ventanas. Más detalles sobre el procesamiento de los traslapes en [12].

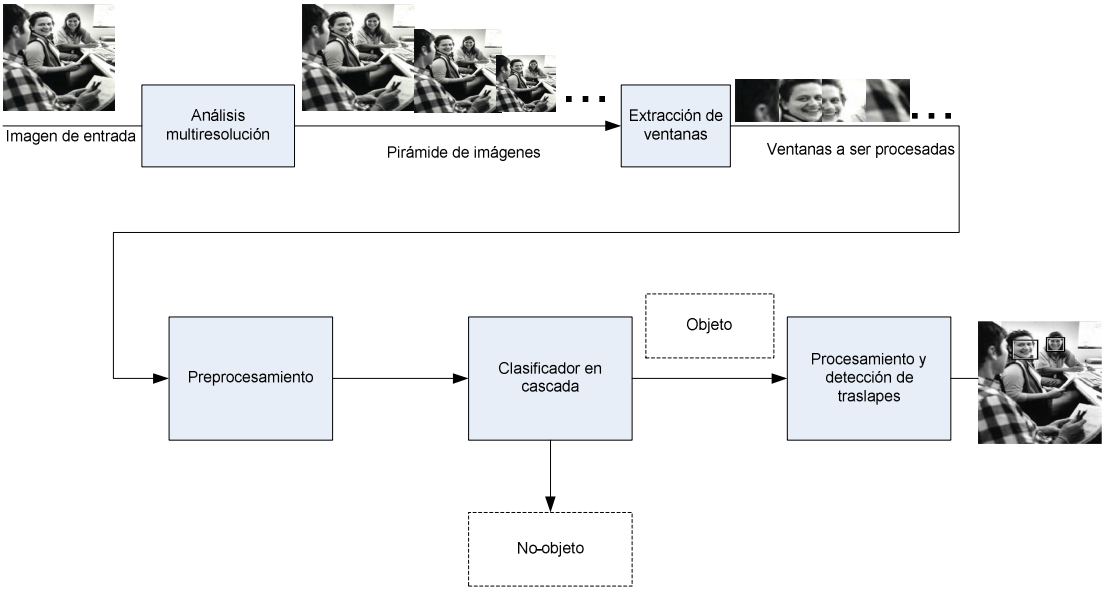


Figura 11: Diagrama bloques de un sistema de detección general.

3.3. Sistemas de Tracking

El seguimiento (o 'tracking' en inglés) de objetos deformables en movimiento en videos es una tarea muy importante en el campo de la visión computacional debido a las múltiples

aplicaciones prácticas, como por ejemplo en vigilancia o en control de tráfico. Por lo mismo ésta es un área muy activa de investigación y muchos enfoques se han tomado sobre este problema con el fin de encontrar formas robustas de hacer 'tracking' en vídeos que presenten distintas dificultades, como oclusiones, ruido, etc. Por ejemplo, en [36] los objetos son seguidos usando información de textura y de bordes junto con una función de correlación, mientras que en trabajos posteriores se agregaron características siguiendo el mismo concepto. En [37] se le añadieron características al mismo modelo con el fin de hacerlo invariante a la iluminación. Otros trabajos se pueden encontrar en [38], [39], [40], todos los cuáles hacen uso de información de color de alguna forma para hacer 'tracking' al objeto. El color es una característica muy usada en estos casos debido a que es invariante ante rotaciones y traslaciones, además de poder calcularse rápidamente. Esto es deseable ya que en aplicaciones que funcionen en tiempo real sólo una pequeña parte de los recursos pueden ser usados para hacer 'tracking', mientras que el resto será requerido generalmente para prepocesamiento o para etapas posteriores como reconocimiento o interpretación de trayectoria que son computacionalmente más costosas.

En este trabajo se hace uso del sistema de 'tracking' llamado 'Meanshift', el que será descrito en mayor a detalle a continuación. Este sistema fue introducido en [41] y [42], teniendo como principal característica el uso de un 'kernel' que permite reducir el tiempo de cómputo en relación a los algoritmos mencionados anteriormente.

3.3.1. MeanShift[2]

En este algoritmo, el objeto al que se desea realizarle el seguimiento es modelado con una función de densidad de probabilidad ('pdf' por sus siglas en inglés) en el espacio de características elegido. Por lo general la característica elegida es el color, debido a las propiedades que presenta y que fueron mencionadas ya anteriormente, pero podrían usarse otras variables como textura o borde, o incluso una mezcla de ellas. Con el fin de usar una baja cantidad de recursos computacionales se usa un histograma de m bines para aproximar la 'pdf'. Una vez modelado el objeto con el histograma $\hat{q}$, en los 'frames' siguientes un candidato a objeto en la posición y se

[2] Extraído de [42]

modela a través de otro histograma, $\hat{p}(y)$, en el mismo espacio. Expresando en forma matemática lo planteado se llega a:

$$Modelo\ objeto: \hat{q} = \{\hat{q}_u\}_{u=1,\ldots,m} \quad \sum_{u=1}^{m} \hat{q}_u = 1 \tag{10}$$

$$Modelo\ candidato: \hat{p}(y) = \{\hat{p}_u(y)\}_{u=1,\ldots,m} \quad \sum_{u=1}^{m} \hat{p}_u = 1 \tag{11}$$

Claramente lo que se busca es encontrar el candidato a objeto que más se asemeja al modelo del objeto calculado inicialmente. Para calcular esta semejanza se usa una función de similaridad entre $\hat{p}$ y $\hat{q}$:

$$\hat{\rho}(y) = \rho[\hat{p}(y), \hat{q}] \tag{12}$$

El problema está en encontrar el máximo local de esta función en la imagen que indica la presencia del objeto en el segundo 'frame' que tenga una representación similar a $\hat{q}$ (objeto en el primer 'frame'). Este puede ser un procedimiento costoso en términos computacionales, por lo que se usa un 'kernel' sobre el dominio espacial con el fin de obligar a que la función $\hat{\rho}(y)$ tenga derivada constante, lo que hace más simple el cálculo del gradiente y por lo tanto la obtención de la dirección del movimiento más sencilla.

Un 'kernel' es una función de peso usada en un método estadístico llamado estimación no-paramétrica, que permite encontrar una función a partir de una serie de datos sin usar ningún parámetro de ajuste [44]. Un tutorial sobre estimación no-parámetrica y 'kernels' se puede encontrar en [45].

La modelación del objeto se realiza de la siguiente forma. Se dice que $\{x_i^*\}_{i=1,\ldots,n}$ representa la posición de los píxeles en la región definida como perteneciente al objeto, y se define el 'kernel' isotrópico *k(x)* que asigna pesos más pequeños a los píxeles más alejados del centro del objeto. Además se define la función $b(x_i^*)$ que asocia al píxel x_i^* el bin que le corresponde en el histograma. Entonces la probabilidad de la característica $u \in \{1,\ldots,m\}$ en el modelo del objeto es calculada como:

$$\hat{q}_u = C \cdot \sum_{i=1}^{n} k\left(\left\|x_i^*\right\|^2\right) \cdot \delta[b(x_i^*) - u] \quad ; \quad C = \frac{1}{\sum_{i=1}^{n} k\left(\left\|x_i^*\right\|^2\right)} \tag{13}$$

Donde δ es la función delta de Kronecker. De forma similar se calcula la probabilidad de cada característica en los candidatos a objeto, pero considerando un ancho de banda *h* junto con el 'kernel'. En este caso se define $\{x_i\}_{i=1,\ldots,n_h}$ como la posición de los píxeles del candidato a objeto:

$$\hat{p}_u(y) = C_h \cdot \sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right) \cdot \delta\left[b(x_i^*) - u\right] \quad ; \quad C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right)} \tag{14}$$

El ancho de banda h define la escala del candidato a objeto, es decir, el número de píxeles que se considera en el proceso de localización. El 'kernel' usado es el de Epanechnikov, el que se define como:

$$k(x) = \begin{cases} \frac{1}{2} \cdot c_d^{-1}(d+2) \cdot (1 - x) & si\ x \le 1 \\ 0 & \sim \end{cases} \tag{15}$$

Donde c_d es el volumen de una esfera de d-dimensiones, y x corresponde a los píxeles normalizados, y relativos al centro, que están dentro del objeto. Dado que el espacio de la imagen en que se trabaja es bidimensional, el 'kernel' usado es igual a:

$$k(x) = \begin{cases} \frac{2}{\pi} \cdot (1 - \|x\|^2) & si\ x \le 1 \\ 0 & \sim \end{cases} \tag{16}$$

Este 'kernel' puede ser visualizado en la Figura 12.

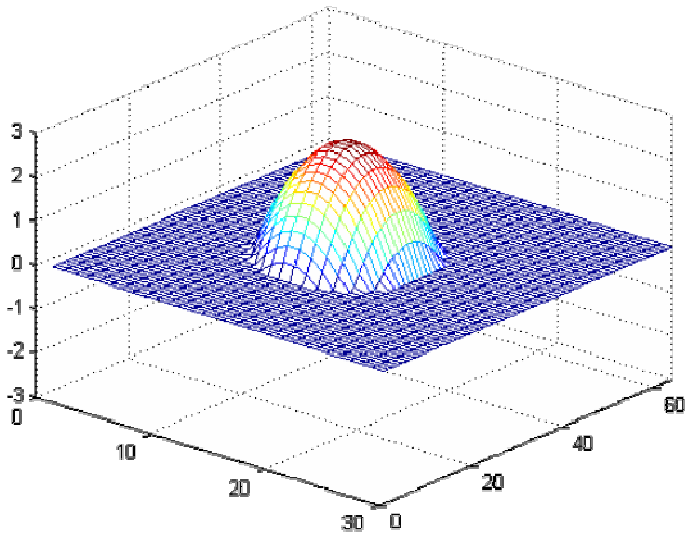


Figura 12: Kernel monótono decreciente de Epanechnikov. Imagen extraída de [43].

Las distancias entre el objeto y los candidatos a objeto se calculan, como ya se mencionó, con una función de similaridad que se conoce como el coeficiente de Bhattacharyya y que se define para dos distribuciones discretas como sigue:

$$\hat{\rho}(y) = \rho[\hat{p}(y), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y) \cdot \hat{q}_u} \qquad (17)$$

El problema a resolver se resume entonces en maximizar la función $\hat{\rho}(y)$ en función de y. La búsqueda del máximo comienza en la ubicación del objeto en el 'frame' anterior, que llamaremos $\hat{y}_0$. Aplicando varias propiedades del 'kernel' y de las funciones antes descritas se llega a la siguiente función sobre la que se itera hasta encontrar el máximo de la función $\hat{\rho}(y)$, donde $\hat{y}_0$ es en cada iteración la posición del candidato a objeto:

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i \cdot w_i}{\sum_{i=1}^{n_h} w_i} \quad ; \quad w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \cdot \delta[b(x_i) - u] \qquad (18)$$

Cuando la diferencia entre $\hat{y}_0$ e $\hat{y}_1$ es menor que cierto umbral dado en alguna iteración, se finaliza la recursión y se considera que se ha encontrado la nueva ubicación del objeto (ver Figura 13).

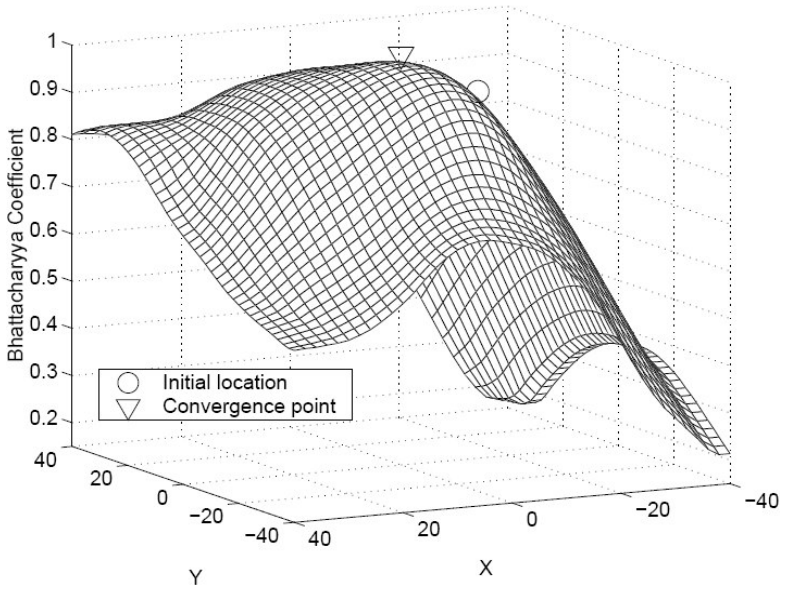


Figura 13: Ejemplo de la función de similaridad (Bhattacharyya) para un caso real, indicando la posición inicial y el punto de convergencia después de la iteración. Imagen obtenida de [42].

Para más detalles acerca de la implementación del algoritmo de 'Meanshift' se refiere al lector a [41] y [42], donde se expone con mayor profundidad el procedimiento a realizar.

3.4. Detección de piel

La detección de piel juega un importante rol en un amplio rango de aplicaciones desde detección de caras, 'tracking' de caras, análisis de gestos, contenido de imágenes y en varios métodos de interacción humano-computador. Métodos de detección de piel basados en la información de color han ganado mucha atención ya que pueden ser calculados de manera muy eficiente computacionalmente, a la vez que proporcionan información invariante a rotaciones y escalamientos, mientras que presenta cierta robustez ante oclusiones parciales. La detección de color de piel es usualmente usada como un paso preliminar para procesamientos posteriores, como por ejemplo reconocimiento de caras. Sin embargo, la detección del color de piel puede ser una tarea muy complicada ya que el color de piel en una imagen es sensible a muchos factores, algunos de los cuales son los siguientes:

- Características de la cámara: la distribución del color de la piel para la misma persona es distinta entre diferentes cámaras dependiendo de las características del sensor.
- Grupo étnico: el color de piel varía entre personas que pertenecen a diferentes grupos étnicos y entre personas de distintas regiones.
- Características individuales: edad, sexo y parte del cuerpo también afectan el color de piel.
- Iluminación: el color de piel es distinto ante diferentes iluminaciones, dependiendo de si es natural o artificial, de día o de noche, de lámpara incandescente o de tubo fluorescente, etc.
- Otros factores: maquillaje, sombras y el movimiento también influyen en el color de piel sensado.

Desde el punto de vista de la clasificación, la detección de piel puede ser visto como un problema entre dos clases: píxeles de piel vs. píxeles de no-piel. Los pasos básicos para la detección de piel en una imagen, usando la información de color, son:

1) Representar los píxeles de la imagen en un espacio de color apropiado.
2) Modelar los píxeles de piel y los de no-piel usando una distribución adecuada.
3) Clasificar las distribuciones modeladas.

Según varios autores, el espacio de color determina que tan efectiva será nuestra modelación de la distribución de colores de piel. La distribución de colores de piel usada es generalmente modelada por un histograma, o por una o varias distribuciones Gaussianas. Varias técnicas en clasificación de modelos de color de piel, que van desde simples 'LUT' hasta complejos reconocimiento de patrones han sido publicados.

3.4.1. Espacio de color

La elección del espacio de color puede ser considerada como la etapa inicial antes de poder realizarse la detección de piel. El espacio de color RGB es el espacio estándar para la mayoría de los formatos de imágenes. Cualquier otro espacio de color puede obtenerse a partir de RGB por medio de transformaciones, sean éstas lineales o no-lineales. Esta conversión de espacio de color se realiza bajo la presunción de que con esto se logrará una mejor separación entre los 'clusters' de pixeles que corresponden a piel y los de no-piel. Aunque algunos autores discuten la veracidad de esta afirmación en algunos casos particulares [14], en general se puede decir que dependiendo de que sistema de clasificación de piel se use se debe determinar que espacio de color es más apropiado para ese caso [15]. Algunos espacios de color típicos son los que se indican a continuación.

Espacios de color básicos (RGB, RGB normalizado): RGB es el espacio de color más utilizado en imágenes digitales debido al amplio uso del filtro de Bayer en cámaras

[16], en que por la implementación física se obtiene inmediatamente la imagen en ese formato. RGB es el acrónimo que se usa para referirse a los tres colores primarios en inglés: rojo, verde y azul. En aplicaciones digitales se suele usar una representación de 24 'bits' para RGB, usando 8 'bits' para cada color. Esto da un rango en [0, 255] en números naturales para representar cada color, o entre [0,1] para números reales. La representación de este espacio de color en forma gráfica usualmente se hace dibujando un cubo, en que cada uno de los ejes indica uno de los tres colores básicos (ver Figura 14).

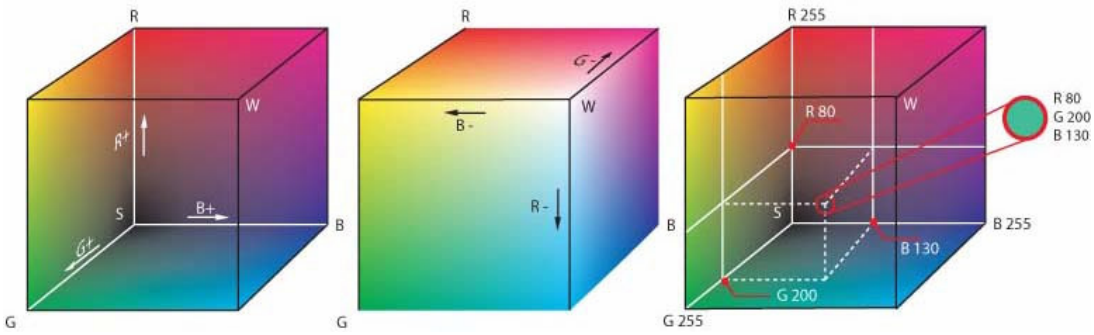


Figura 14: Espacio de color RGB, representado como un cubo. Extraído de [19].

Los píxeles de piel tienden a formar un clúster en prácticamente cualquier espacio de color, en particular RGB (ver Figura 15). Con el fin de obtener una mayor invarianza ante cambios de iluminación en la detección de piel, usualmente se normalizan los componentes de RGB para que la suma sea igual a la unidad (r + g + b = 1). Dado que la suma es igual a 1, se puede despreciar una componente dado que no aportará ninguna información significativa (se suele despreciar 'b'), reduciendo la dimensión del problema. El clúster del color de piel tiene menor varianza en el espacio RGB normalizado que en RGB normal ([17], [18]), lo que produce ciertamente ventajas en su uso para clasificación (Figura 16).

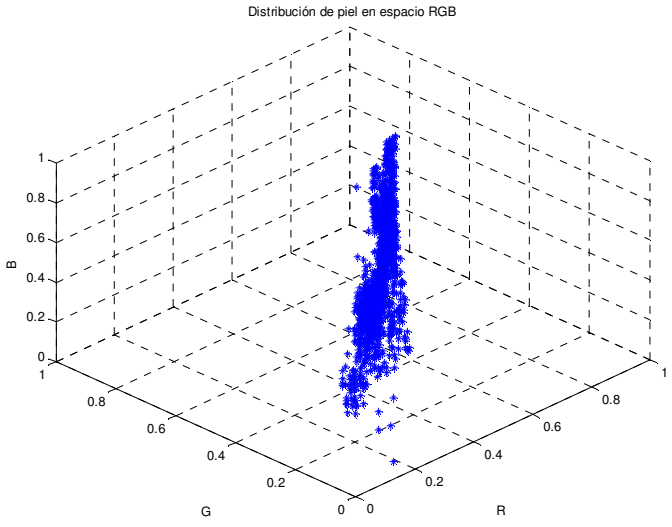


Figura 15: Distribución de color de piel en el espacio RGB

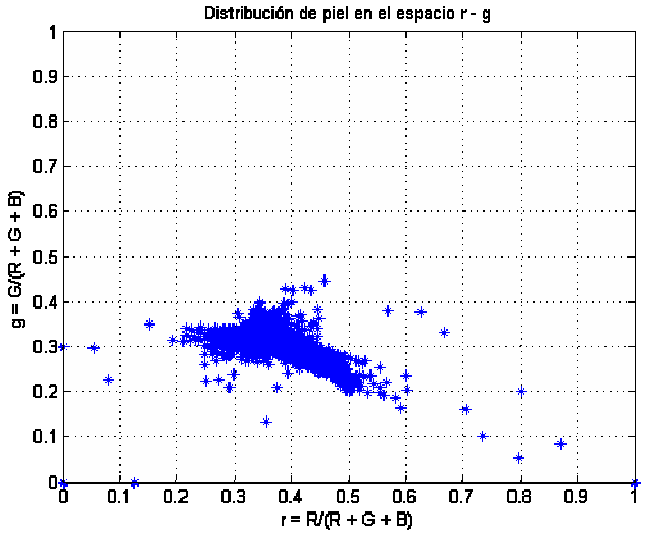


Figura 16: Distribución de piel en el espacio RGB normalizado.

Espacios de color perceptual (HSV): con el fin de lograr espacios de color que sean más parecidos a la manera en que los humanos procesan los colores se han creado varias transformaciones a partir del clásico RGB. Uno de los más usados es HSV, que es el acrónimo en inglés para: tonalidad, saturación y valor. La tonalidad define el tipo de color, saturación es la intensidad de éste y valor el brillo. La transformación de RGB a HSV presenta varias ventajas en cuánto a la manera en como la iluminación la afecta, por lo que este espacio de color se ha usado en varios trabajos ([20],[21],[23]). Un ejemplo de la distribución del color de piel obtenida en este espacio se puede apreciar en la Figura 18.

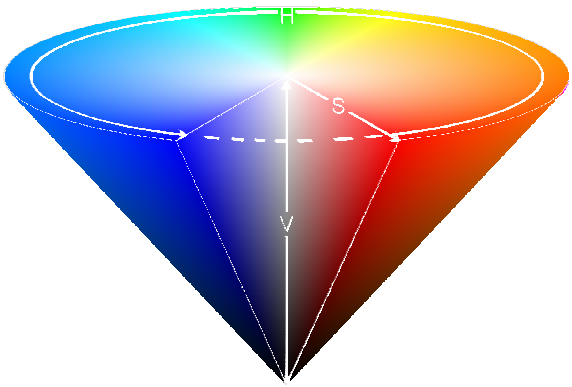


Figura 17: Espacio de color HSV representado como un cono. Imagen extraída de [22].

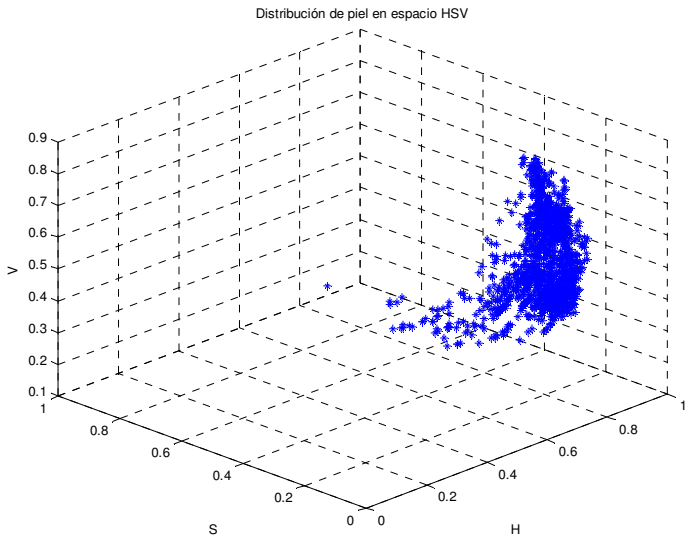


Figura 18: Distribución de color de piel en el espacio HSV.

Otros espacios de color: muchísimos otros espacios de color existen que han sido diseñados para tener algunas propiedades particulares, como tener sus variables ortogonales entre si (como YCbCr [24], YUV [25], YIQ [26]) o ser perceptualmente uniformes (CIE-Lab [27], CIE-Luv [18]). Sin embargo estos espacios de colores son menos usados que RGB o HSV debido a lo costoso de las transformaciones requeridas para llegar a ellos en comparación con las limitadas ventajas que se obtienen.

3.4.2. Métodos de detección de piel

La idea final de la detección de piel es poder llegar a tener una regla que permita decidir si un píxel en particular corresponde a piel o no. Para construir esta regla existen varios métodos, pero casi todos toman como base el medir la distancia (en su sentido más general) a un rango de colores definido como perteneciente a piel [28]. Algunos de los métodos más usados para realizar esta clasificación son los siguientes:

- Regiones de piel definidas explícitamente: uno de los métodos más simples para realizar detección de piel es la de fijar una región que se considerará como piel a partir de los resultados obtenidos con los datos de entrenamiento. Un ejemplo de esto es lo presentado en [29], en que se llega a la siguiente regla:

$$R > 95 \wedge G > 40 \wedge B > 20$$
$$\wedge \max\{R,G,B\} - \min\{R,G,B\} > 15 \qquad (19)$$
$$\wedge |R - G| > 15 \wedge R > G \wedge R > B$$

La dificultad que presenta este método es que la simplicidad del mismo no lo hace aplicable en condiciones en que se presenten grandes cambio de luminosidad. Además es posible que las reglas se sobreajusten a los datos usados durante el entrenamiento, no resultando luego un sistema de clasificación aplicable a las condiciones de uso reales.

- Métodos no-paramétricos: estos métodos consisten básicamente en dividir el espacio de color elegido en un número de bines predefinidos, correspondiendo cada bin a un rango de colores fijo. Cada bin guardará la información del número de veces que un color de la base de entrenamiento cayó dentro de él. Luego estos valores son normalizados, con lo que se obtiene a una distribución de probabilidad discreta (la 'LUT') que se puede usar para detectar piel fijando un umbral. Algunos trabajos que aplican este enfoque son [30] y [32]. El valor de probabilidad computado anteriormente es estadísticamente la probabilidad de encontrar un color en ese bin dado que ese color pertenece al conjunto de piel ($P(color / piel)$). Si se calcula también $P(color / no - piel)$ (la probabilidad del color dado que no es piel) y usando el teorema de Bayes se puede llegar a la siguiente regla de decisión para la detección de piel [31]:

$$\frac{P(color / piel)}{P(color / no - piel)} \geq \theta \qquad (20)$$

  Donde θ indica el umbral elegido para la detección.

- Métodos paramétricos: en vez de usar un método no-paramétrico, como en el caso de los clasificadores bayesianos en que se trabaja con histogramas, es posible emplear funciones de probabilidad conocidas. Este enfoque tiene la ventaja de requerir menor capacidad de almacenamiento que en el caso anterior (ya que en él hay que guardar las 'LUT' usadas) y poder generalizar con menor cantidad de datos de entrenamiento [33]. La función típicamente usada es la de Gauss, de la manera siguiente:

$$p(c / piel) = \frac{1}{2 \cdot \pi \cdot |\Sigma|^{1/2}} \cdot e^{-\frac{1}{2}(c-\mu)\cdot\Sigma^{-1}\cdot(c-\mu)} \qquad (21)$$

Donde c representa el vector de color, mientras que μ y Σ son los parámetros de la función y corresponden a la media y la varianza respectivamente. Por lo tanto:

$$\mu = \frac{1}{n} \cdot \sum_{j=1}^{n} c_j \; ; \; \Sigma = \frac{1}{n-1} \cdot \sum_{j=1}^{n} (c_j - \mu) \cdot (c_j - \mu)^T \qquad (22)$$

Con *n* el número total de muestras de *c*. Esta probabilidad se puede usar directamente para decidir en el momento de la clasificación si el píxel en cuestión es de piel [34], o se puede usar en su defecto a través de la distancia de Mahalanobis [24].

El uso de una gaussiana para modelar la distribución de piel es posible en condiciones de luminosidad controlada, ya que la piel de distintos individuos tiende a formar un cluster más o menos compacto, dependiendo del espacio de color usado. Sin embargo en condiciones en que la luminosidad puede variar se hace necesario ampliar la definición anterior a una que de más flexibilidad para estos casos. Así se llega a la siguiente función:

$$p(c \,/\, piel) = \sum_{i=1}^{k} \pi_i \cdot p_i(c \,/\, piel) \; ; \; \sum_{i=1}^{k} \pi_i = 1 \qquad (23)$$

Donde *k* es el número total de gaussianas, π_i es el parámetro que indica la contribución de cada una y cada $p_i(c/piel)$ es una función de distribución de probabilidad, con su respectiva media y varianza. La clasificación final se puede hacer comparando la probabilidad con algún umbral [18], o usando el teorema de Bayes si se modela también *p(c/no-piel)* [31].

- Modelos dinámicos: los modelos descritos anteriormente están formulados principalmente para operar sobre imágenes estáticas, es decir, no están hechos para aprovechar las ventajas que se pueden tener en imágenes que provengan de vídeos. Una familia de métodos de detección de piel cae dentro de la categoría de ser diseñados específicamente para ser usados en conjunto con métodos de 'tracking' de caras u otras partes del cuerpo, es decir para ser usados particularmente en vídeos. Esto hace el sistema distinto a aquéllos que se usan en imágenes estáticas en varios aspectos. Una etapa de inicialización es posible en el momento en que la región de la cara (o en su defecto otra parte del cuerpo) es discriminada del fondo usando algún clasificador o manualmente. Esto da la posibilidad de obtener un modelo particular para la situación específica en ese momento, con lo que se puede obtener un modelo de clasificación óptimo para esas condiciones de iluminación, fondo, cámara y color de piel de la persona. A diferencia de lo que sucede con un modelo de piel más general que es entrenado previamente (o 'offline'), que está hecho para detectar piel

en cualquier condición, como el descrito en [31], es posible obtener mejores niveles de detección con menos falsos positivos con este modelo que es generado 'online' con la misma información de la imagen. Por otro lado, las distribuciones de color de piel en la imagen pueden variar con el tiempo debido a cambios la iluminación, por lo que muchas veces es necesario ajustar el modelo periódicamente para obtener una buena detección. Por lo mismo es necesario que el tiempo de entrenamiento y el de clasificación sean pequeños, es decir consuman pocos recursos computacionales, con el fin de que el sistema pueda funcionar en tiempo real, que es por lo general lo que se necesita al trabajar con vídeos. Algunos métodos que hacen uso de modelos dinámicos son: [30], [35], [5]. Estos modelos dinámicos se complementan en ciertos casos con métodos paramétricos y en otros con métodos no-paramétricos, dependiendo del enfoque que considere más adecuado cada investigador.

Capítulo 4

Sistema desarrollado

4.1. Descripción general

En este capítulo se presenta el sistema construido para satisfacer los requerimientos planteados en los objetivos de esta memoria. Para ilustrar el diseño planteado se presenta un diagrama de éste en la Figura 19.

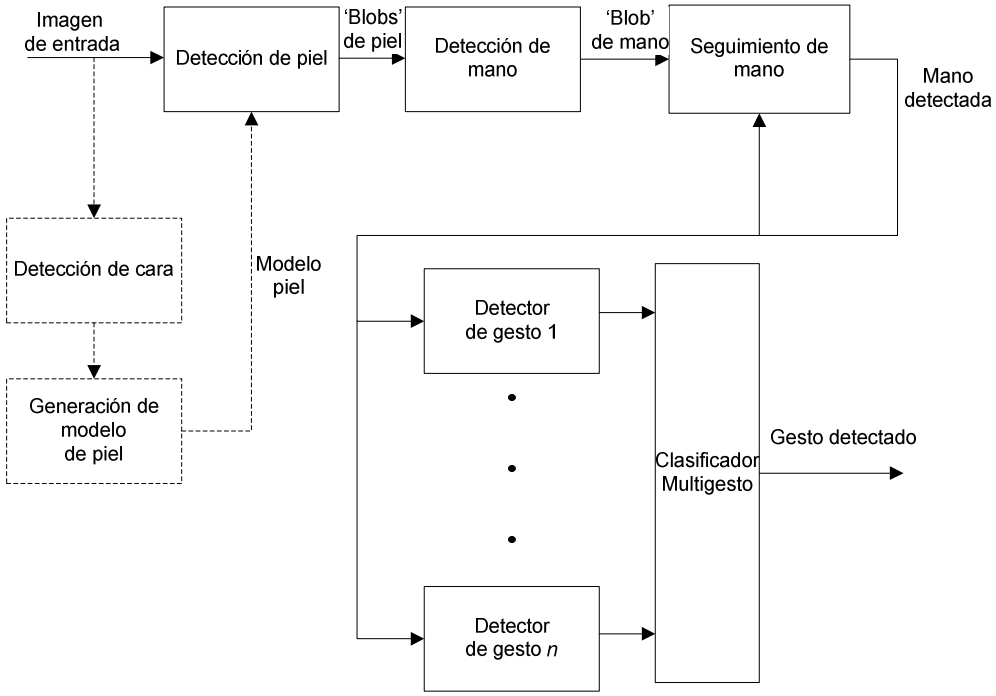


Figura 19: Diagrama de bloques del sistema desarrollado.

4.2. Detalles de la implementación

A continuación se detallará la implementación realizada para los bloques más importantes del sistema descrito en el diagrama de la Figura 19.

4.2.1. Detector de caras

El detector de caras usado en esta etapa no fue desarrollado en el marco de esta memoria, si no que fue usado el implementado en [8]. Este detector hace uso de los clasificadores 'Adaboost' conectados en cascada descritos en el Capítulo 2. Para su entrenamiento se hizo uso de 5000 imágenes de caras, además de usar una versión invertida de izquierda a derecha como validación. Además 3500 imágenes sin caras en ellas se usaron para extraer ventanas de subimágenes de no-caras para el entrenamiento, y 1500 otras imágenes que tampoco contenían caras se usaron para validación. Un ejemplo de la detección de caras que es posible lograr con este sistema se muestra en la Figura 20. Para más detalles de la implementación se recomienda al lector dirigirse a [8], donde además se muestran algunas de las posibilidades de esta forma de detección al detallar la construcción de un clasificador de género (hombre o mujer) y un detector de ojos.

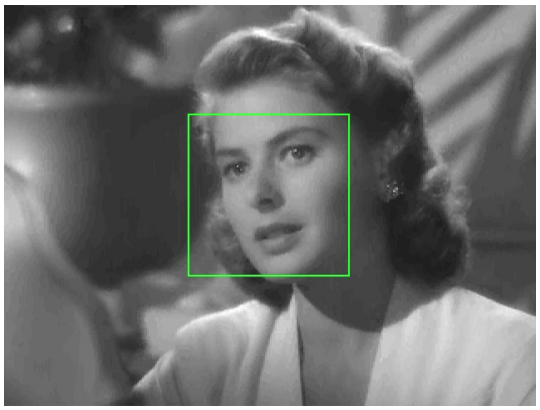
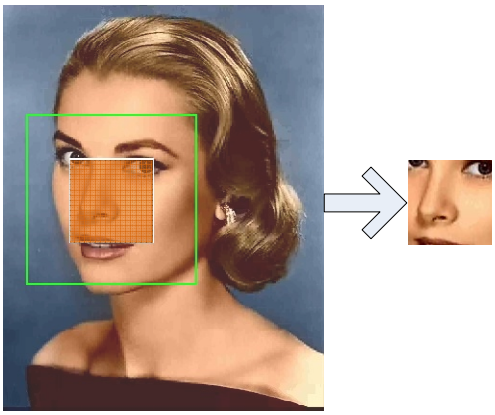


Figura 20: Ejemplo de detección de cara logrado por este bloque.

4.2.2. Detector de piel

La detección de piel se realiza siguiendo un procedimiento similar al detallado en [5]. La idea central de este método es usar la información de la imagen para construir un modelo de piel específico para ella, es decir, generar un modelo adaptivo. Con esto se logra tener una detección de piel más robusta, por ejemplo, al tipo de iluminación o al color específico de piel de la persona. Para lograr esto se aplica sobre la imagen de entrada un detector de caras basado en 'Adaboost'. Como se sabe, este tipo de detectores no usa la información de color de la imagen ya que funcionan sobre imágenes en blanco y negro. Este detector de caras se usa en todos los 'frames' de entrada al programa hasta que es detectada alguna cara. Una vez detectada, se puede construir el modelo del color de piel ya que se tiene la información en los píxeles de la cara. Como es posible que zonas del fondo de la imagen se consideren dentro de la ventana de detección de la cara, se considera para la elaboración del modelo de piel solamente una porción del rectángulo de detección de cara. El procedimiento seguido se ilustra en la Figura 21, donde se muestra la detección de cara en una imagen, la selección de la zona usada para crear el modelo de piel y la fórmula usada para decidir el sector de la imagen que se considerará para el modelo de piel.



(a)

$$x_{0,naranjo} = x_{0,verde} + 0.25 \cdot ancho_{verde}$$
$$y_{0,naranjo} = y_{0,verde} + 0.25 \cdot alto_{verde}$$
$$ancho_{naranjo} = 0.5 \cdot ancho_{verde}$$
$$alto_{naranjo} = 0.5 \cdot alto_{verde}$$

(b)

Figura 21: (a) Detección de cara (rectángulo verde) y selección de la zona usada para crear el modelo de piel. (b) Fórmula usada para calcular la zona de interés para el modelo de piel.

La imagen es entregada por la cámara en el espacio de color RGB. Con el fin de hacer el modelo más invariante a la iluminación, se trabaja con los píxeles en el espacio de color RGB normalizado. La normalización se realiza de la siguiente forma:

$$base = R + G + B$$

$$r = \frac{R}{base}$$

$$g = \frac{G}{base}$$

Esta normalización se aplica sobre la zona de la cara que es considerada apta para construir el modelo, llegándose a una 'cluster' en el espacio RGB normalizado similar al ejemplo que se observa en la Figura 22. La distribución que se obtiene para estos píxeles en cada variable (*r*, *g* y *base*) puede ser aproximada por una gaussiana, como se puede ver al observar la Figura 23 para el histograma de *r*, en la Figura 24 para el histograma de *g* y en la Figura 25 para el histograma de *base*.

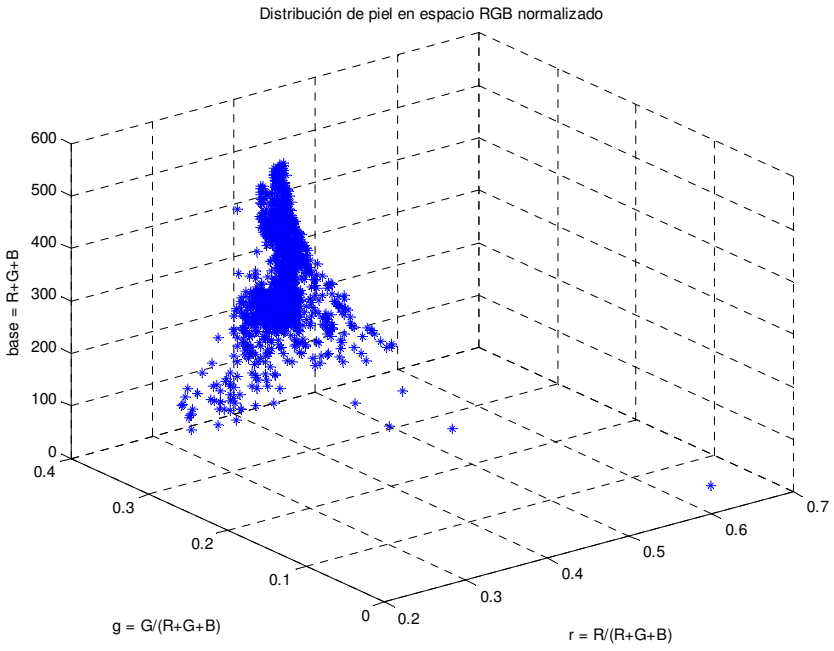


Figura 22: Ejemplo de distribución de piel de la cara en espacio RGB normalizado.

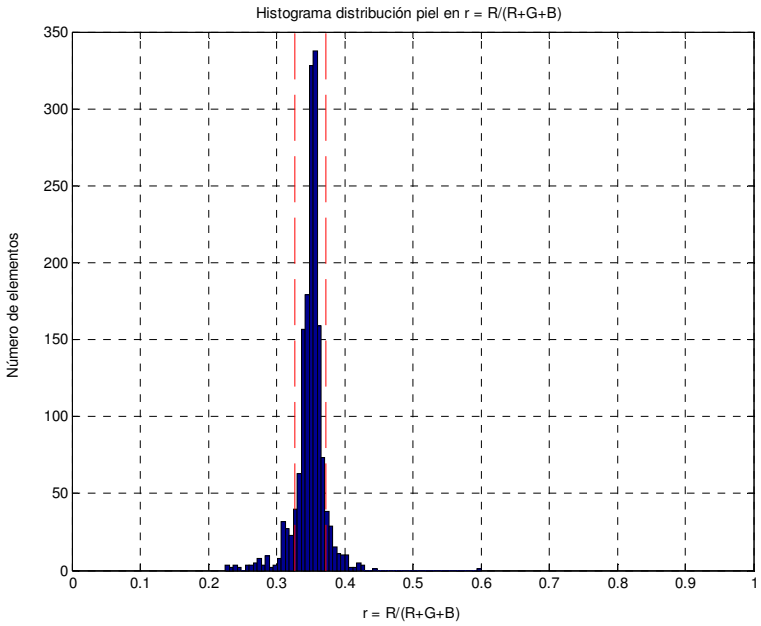


Figura 23: Histograma de distribución de los píxeles de piel en r. Límites de la zona de piel se indican en línea punteada roja, para α = 1.

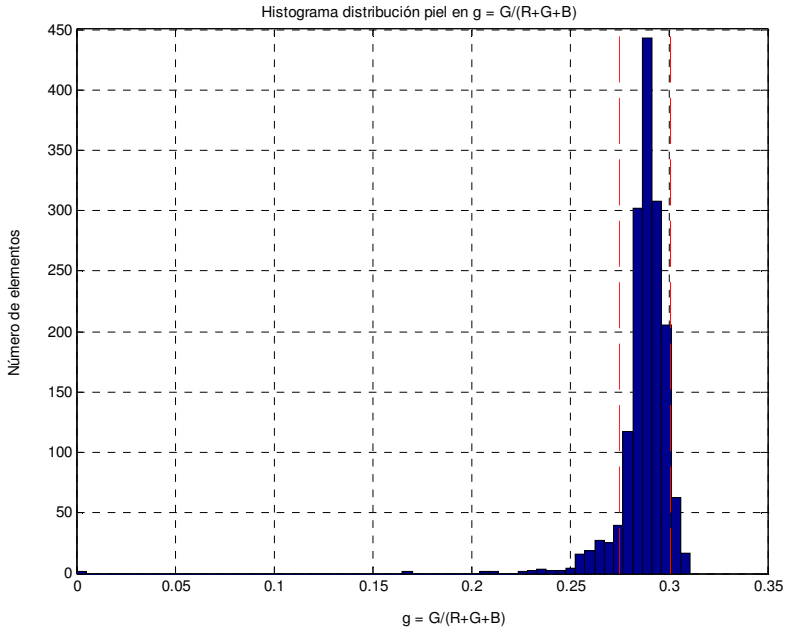


Figura 24: Histograma de distribución de los píxeles de piel en g. Límites de la zona de piel se indican en línea punteada roja, para α = 1.

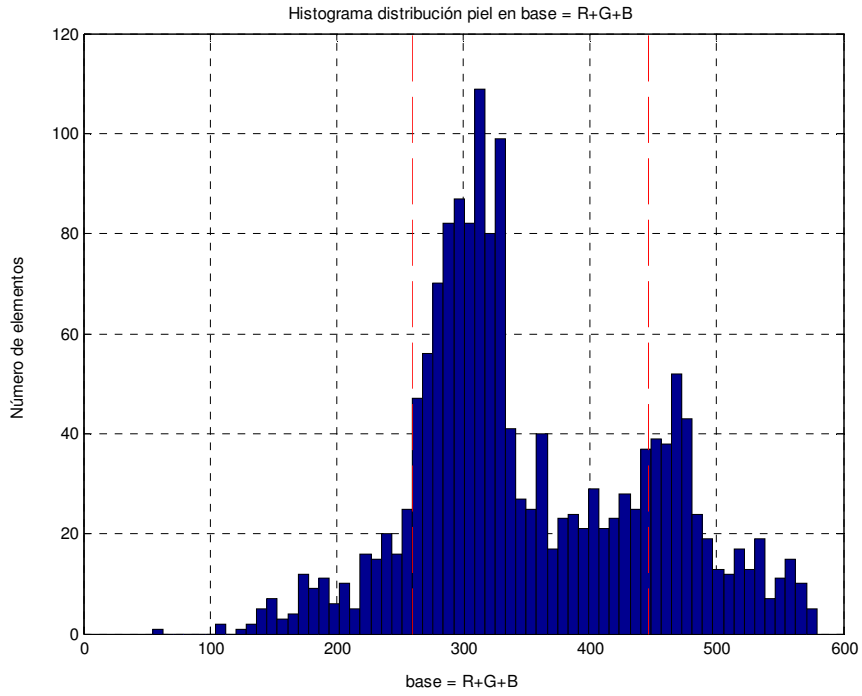


Figura 25: Histograma de distribución de los píxeles de piel en base. Límites de la zona de piel se indican en línea punteada roja, para α = 1.

Una vez normalizados los píxeles de la cara, se procede a modelar con una función gaussiana cada una de las variables de interés antes mencionadas: *r*, *g* y *base*. Se obtienen así 6 nuevos valores, los que corresponden a la media y la desviación estándar para cada una de las variables anteriores, los que llamaremos: μ_r, σ_r, μ_g, σ_g, μ_{base} y σ_{base}. El proceso anterior de la construcción del modelo sólo se realiza una vez para aligerar la carga computacional, además de no ser necesario reajustar el modelo a no ser que se produzca algún cambio de luminosidad muy grande. Con éstos datos se pueden ya clasificar los píxeles de la imagen de entrada según su cercanía a la media de cada una de las gaussianas, siguiendo la siguiente regla de clasificación en cada punto de la imagen de entrada:

$$f(i,j) = \begin{cases} piel & si \quad |c - \mu_c| < \alpha_c \cdot \sigma_c, \quad c = r, g, I \\ nopiel & si \quad \sim \end{cases}$$

Donde i y j representan las distintas coordenadas dentro de la imagen, mientras que a_r, a_g y a_{base} son constantes de ajuste para el clasificador. Por simplicidad se hace todas estas constantes iguales ($a=a_r=a_g=a_{base}$). Se observa en la práctica que este valor debe ser ajustado según la luminosidad de

la imagen, siendo mayor *a* mientras disminuye la luminosidad y viceversa. Esta adaptación se realiza de forma automática de acuerdo al valor obtenido en μ_{base}, variando *a* linealmente según lo que indique μ_{base}. Los límites que se logran aplicando esta clasificación se pueden observan en la Figura 23, Figura 24 y Figura 25 para cada variable. Además en la Figura 26 se han graficado los 'clusters' de piel y no-piel que se obtienen en una imagen estándar. Se ilustra además en esta imagen que el 'cluster' de piel se encuentra incluido dentro del 'cluster' de los de no-piel, lo que hace suponer que ciertas detecciones erróneas y píxeles de piel no detectados ocurrirán al aplicar este método debido al traslape mencionado.

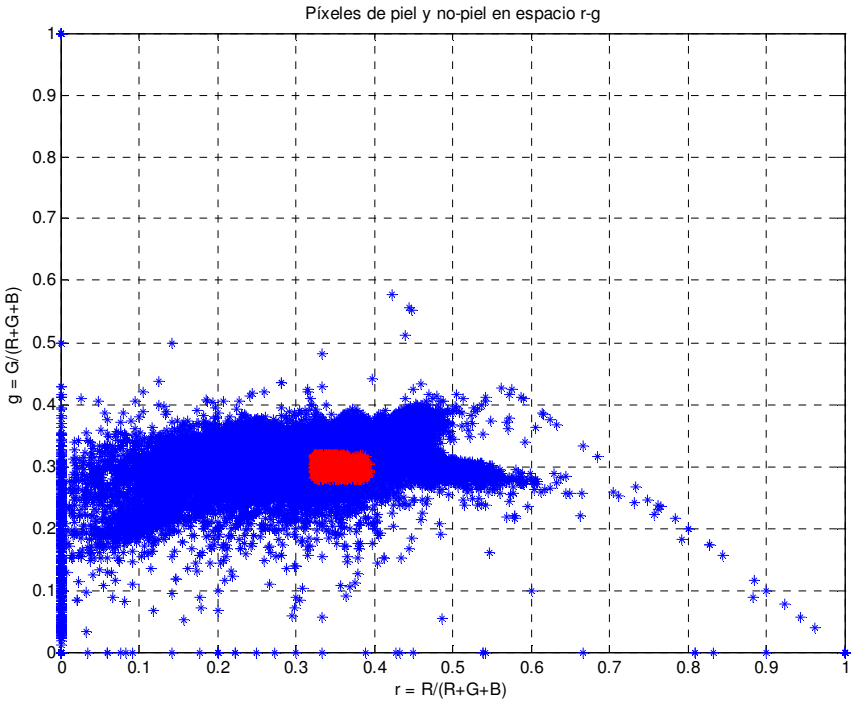
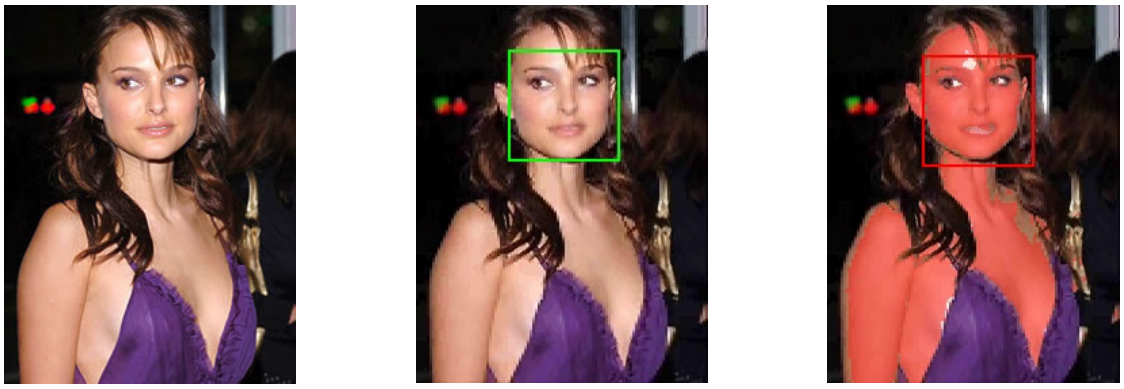


Figura 26: Píxeles detectados como piel (en rojo) y los de no-piel (azul) según el método descrito.

Luego de recorrer la imagen completamente se obtiene como salida todos los píxeles que fueron clasificados como piel según el criterio anterior. Estos píxeles son luego agrupados según su conectividad con los vecinos en 'blobs' de piel. Con el fin de disminuir las detecciones erróneas (falsos positivos) aquellos 'blobs' que tienen un área menor ha cierto tamaño predeterminado son desechados como pertenecientes al conjunto de piel. Finalmente las ventanas que enmarcan a los 'blobs' de piel anteriores son pasadas al módulo de detección de manos, con excepción del 'blob'

que corresponde a la cara. A modo de ejemplo se presenta en la Figura 27 una imagen a la que se le realiza detección de piel con este método.



(a) (b) (c)

Figura 27: Ejemplo de los resultados de la detección de piel: (a) Imagen original, (b) Detección de cara, (c) Detección de piel ilustrada como una máscara de color rojo.

4.2.3. Detector de manos

Para la detección de manos se emplea un clasificador 'Adaboost', al igual que para la detección de caras. Este detector de manos está entrenado con imágenes de personas realizando un gesto de puño. Las imágenes usadas para construir este detector fueron extraídos de la base de datos IDIAP [70]. En total corresponden a 1329 imágenes de puños capturadas en distintas condiciones de iluminación y por distintas personas. Este detector se ejecuta sobre los 'blobs' de piel que, como ya se mencionó, excluyen a aquél que contiene la cara. El detector es ejecutado solamente sobre los 'blobs' de piel como una forma de disminuir los falsos positivos que se podrían producir en otras partes de la imagen, además de disminuir el costo computacional de la detección al no tener que revisar toda la imagen. Una vez detectada una mano, que para el caso de este detector en particular tendrá que estar haciendo un gesto de puño para ser reconocida, la ventana de detección es entregada a un sistema de seguimiento. El sistema mencionado se basa en una implementación del algoritmo de 'tracking Meanshift'. Una vez que el sistema de 'tracking' está haciendo su trabajo no hay necesidad de seguir operando el módulo de detección de manos sobre los 'blobs' de piel, ya que se conocerá la ubicación de la(s) mano(s), por lo que éste deja de funcionar mientras 'Meanshift' este operando. En caso de que la mano salga de la imagen, o el

sistema pierda el seguimiento, el módulo de detección de manos comienza a funcionar nuevamente. Finalmente se obtiene como salida de estos módulos uno o varios región(es) de interés (ROI por sus siglas en inglés, 'region of interest'), los cuáles indican la posición de la(s) mano(s) en la imagen. Estos ROI son entregados al módulo siguiente de detección de gestos para determinar que gesto es el que se está realizando.

4.2.4. Detector de gestos

Con el fin de poder determinar el gesto que se está ejecutando se han construido una serie de detectores, usando clasificadores 'Adaboost'. Las imágenes para entrenar estos clasificadores fueron obtenidos del mismo programa haciendo uso del método de 'Active Learning' descrito en la sección 3.2.5. Como se explicó ahí, el método consiste en que, una vez detectada la mano, se procede a extraer la ventana del 'tracking' de manos durante cierto tiempo, hasta que se considere se cuenta con un número suficiente de imágenes. De esta forma se puede generar una base de imágenes para entrenamiento de forma muy rápida y simple. Aún así se comprobó de forma heurística que se obtenían mejores detectores si las imágenes de entrenamiento estaban alineadas manualmente, por lo que se realizó este procedimiento extra para mejorar los resultados obtenidos. Los gestos seleccionados para ser detectados son los indicados en la Figura 28: 'puño', 'palma', 'índice' y 'cinco'. Una vez que se han entrenado los detectores para cada gesto de interés se puede realizar finalmente la detección de gestos. Para hacer esto se ejecutan todos los detectores sobre el ROI que fue entregado como la salida de la etapa de 'tracking'. Como en muchos casos más de un detector podría tener resultados positivos, se hace necesario crear una regla de decisión para elegir el gesto que realmente se está realizando. Para cumplir esta tarea se ha entrenado un clasificador en el programa WEKA [71], tomando como entrada cuatro atributos que entrega cada uno de los detectores. Los cuatro atributos usados se describen a continuación:

1. Confidencia: es la suma de las confidencias de la cascada para cada ventana en que el gesto es detectado (un mismo gesto es detectado en varias posiciones y escalas).
2. Número de ventanas: número de ventanas en que el gesto es detectado.
3. Confidencia media: confidencia media de la detección, que es calculada como:

$$\text{conf. media} = \frac{\text{confidencia}}{\text{núm. ventanas}}$$

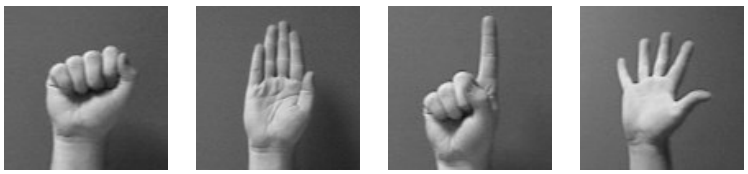
4. Confidencia normalizada: se calcula como:

$$\text{conf. normalizada} = \frac{\text{conf. media}}{\text{conf. máxima}}$$

En que 'confidencia máxima' es el máximo valor que puede obtener la detección en una ventana.

Para generar el conjunto de entrenamiento para el clasificador que fue entrenado en WEKA, se hizo uso de un procedimiento similar al de 'Active Learning'. La diferencia es que en este caso la salida es un vector con los cuatro atributos para cada gesto detectado. Es decir, en el caso del entrenamiento del clasificador se reemplaza la ventana donde se encontraba la mano, como sucedía cuando se quería entrenar los detectores, por un vector que tiene en total 16 elementos para este caso particular (ya que son 4 gestos). Si se calcula este vector para cada uno de los gesto de interés durante algún tiempo, se obtendrá un suficiente número de observaciones para entrenar el clasificador que se necesita.

WEKA permite seleccionar entre varios tipos de clasificadores, como Bayesianos, SVM ('Support vector machines'), árboles de decisión, etc. En este caso se eligió usar un árbol de decisión de tipo J48 por obtenerse mejores resultados en el conjunto de entrenamiento. Este clasificador es la última etapa en el sistema de reconocimiento de gesto, dando como resultado el gesto efectuado en caso de ser alguno existente en la base de datos, además de la ventana en que fue detectado el gesto, con lo que se procede a actualizar la información del 'tracking'.



Puño Palma Índice Cinco

Figura 28: Gestos detectados por el sistema desarrollado.

4.2.5. Restricciones del sistema

Debido al método de resolución usado para el problema del reconocimiento de gestos surgen algunas restricciones implícitas a ellos, los que se detallan a continuación:

- El o los usuarios del sistema deben ubicarse de frente a la cámara usada y realizar los gestos de manera frontal a ella.

- Los gestos no deben presentar rotaciones mayores a aproximadamente 15° de las posiciones indicadas en la Figura 28. Es decir, para obtener una correcta detección de los gestos, no se permiten rotaciones en el plano ni fuera de él.
- Las manos no pueden presentar un color demasiado distinto al que tenga la cara. Por esta razón, la mano debe presentarse libre de vestiduras, como son los guantes. Esto porque inicialmente la detección de piel es usada para identificar posibles zonas con manos en la imagen, y esta detección de piel es calibrada según los colores encontrados en la cara.
- No se permiten interacciones entre las manos ni entre las manos y la cara. Esta restricción viene del hecho de que la etapa de 'Tracking' usa información de color para el seguimiento, por lo que contactos entre estas zonas pueden desorientar el seguimiento de las manos y conducir a resultados errados.
- El reconocimiento de los gestos será estático y no dinámico. Es decir la detección y reconocimiento de los gestos se hará de acuerdo a la información presente en cada imagen de entrada y no como resultado de un grupo de imágenes sucesivas obtenidas a través del tiempo.
- No debieran haber grandes variaciones en la iluminación, ya que esto afectaría al sistema de detección de piel y por consiguiente podría perjudicar la detección y clasificación.
- El grupo de gestos a reconocer corresponde a un número reducido. Gestos diferentes a los mostrados en la Figura 28 no serán detectados, o serán detectados de forma errónea. Sin embargo, el sistema ha sido diseñado pensando en futuras ampliaciones, por lo que el hacer esto no debería traer consigo mayores complicaciones.

4.2.6. Interfaz gráfica

La interfaz gráfica que fue desarrollada para la visualización de los resultados hace uso de la librería OpenCV ('Open Computer Vision Library'), disponible en [73], para la captura de imágenes y la presentación en pantalla de la salida. Esta interfaz permite tres modos de visualización: una presenta las detecciones de caras, de piel y de manos en una misma ventana, además de indicar como que gesto fue clasificada la(s) mano(s), si es que hay alguna presente. El siguiente modo muestra los mismos resultados pero excluyendo la detección de piel, mientras que

el último modo muestra únicamente los 'blobs' de piel detectados en la imagen, enmarcando aquéllos que no corresponden a caras y que por lo tanto serán analizados por el bloque de detección de manos. Ejemplos de los modos de visualización descritos se pueden observar en la Figura 29, que se encuentra en la página siguiente. Nótese en estas imágenes la no uniformidad del fondo y los cambios de iluminación presentes.

4.2.7. Adquisición de imágenes

Para la realización de este trabajo de título fue usada una cámara comercial de gama alta, marca Philips, modelo SPC900NC. Esta 'webcam' tiene como característica el poseer una resolución de 1,3 [MP] (megapíxeles) y tener un sensor de tipo CCD, que lo hace tener una mejor respuesta a cambios de iluminación y a ambientes de baja luminancia en comparación con las cámaras de bajo costo que poseen típicamente sensores de tipo CMOS. El tamaño de imágenes usado durante todo este trabajo fue de 320x240 píxeles, aun cuando esta cámara en particular permite trabajar con vídeos de tamaño de hasta 640x480. Se eligió este tamaño porque dimensiones mayores disminuyen la velocidad en que se puede ejecutar el programa, sin aportar en general mayor información que se pueda considerar relevante para la ejecución del mismo. En general se puede decir que el proceso de adquisición de imágenes con la cámara no presenta grandes restricciones para el proceso de detección y clasificación de gestos, a excepción del hecho de que él o los usuarios deben presentarse de manera frontal a ella.

Figura 29: Capturas de la interfaz gráfica del programa. (a) Detección de cara. (b) Detección de cara y detección de piel. (c) y (d) 'Blobs' de piel, enmarcándose aquellos que no correspondan a la cara. (e)-(i) Región de cara, región de mano, detección de piel y clasificación de gesto. (k) y (l) Región de cara, región de mano y clasificación de gesto sin detección de piel.

4.2.8. Software utilizado

La aplicación se encuentra escrita en su totalidad en C++, haciendo uso de la librería OpenCV [73] para la obtención de imágenes desde la cámara y la posterior visualización de los resultados obtenidos en pantalla. Esta librería es en la práctica un estándar en los trabajos no comerciales, en particular los desarrollados para investigación, en el campo de visión computacional. Por lo mismo se encuentra disponible para su uso en varias plataformas: Windows, Linux y Mac OS X. Debido a que el resto del código está escrito en C++, el programa en su totalidad puede ser usado eventualmente en cualquiera de estas plataformas siempre que se tenga instalada la librería OpenCV. No obstante lo anterior, el sistema fue desarrollado y probado en su totalidad solamente bajo Windows XP. Otro programa que fue usado también es WEKA [71], para la creación del clasificador de gestos. Como el entrenamiento de este clasificador se realiza 'offline', es decir, antes de la ejecución del programa, y los resultados son exportados a un archivo de texto que luego es leído por el programa, no se crea ninguna nueva restricción por el uso de este software respecto a la plataforma que es necesario usar.

Capítulo 5

Resultados y Análisis

5.1. Resultados del sistema desarrollado

En esta sección se presentan los resultados obtenidos con el sistema desarrollado en esta memoria para detectar manos y clasificar gestos. Para una mayor claridad se han separado los resultados obtenidos en las distintas etapas del sistema, enseñándose al final los resultados para el sistema completo.

Los resultados de sistemas de detección de objetos en imágenes son generalmente presentados en la forma de curvas 'ROC' (acrónimo en inglés para 'Receiver Operating Characteristic curve'). En estas curvas se grafica la tasa de detecciones acertadas en función de las detecciones del sistema que estuvieron erradas. Debido a que normalmente una mayor cantidad de detecciones acertadas va de la mano con un crecimiento también en el número de errores por parte del sistema (o de manera inversa, a menor falsas detecciones menos detecciones correctas), la forma de estas curvas es la de una función cóncava monótona y creciente.

En primer lugar se tienen los resultados del detector de caras, usado como etapa inicial para la posterior detección de piel. El gráfico en la Figura 30 muestra el resultado de este detector en la base de imágenes BioID [67], que contiene 1521 imágenes de tamaño 384x286 píxeles. Cada imagen presenta una cara de 23 personas ocupadas para la prueba. Con el fin de ajustarse lo más posible a condiciones reales de uso, esta base contiene una gran varianza en condiciones de iluminación, fondos, tamaños de las caras y expresiones de ellas. Ésta es una base de estándar de pruebas en el campo de la detección de caras por lo que los resultados pueden ser comparados

con otros métodos similares. Para estas comparaciones, usando ésta y otras bases de imágenes, se refiere al lector a [8]. Lo importante a recalcar en la Figura 30 es la alta tasa de detección del algoritmo y su baja tasa de falsos positivos, por lo que se considera una buena forma de inicializar el sistema.

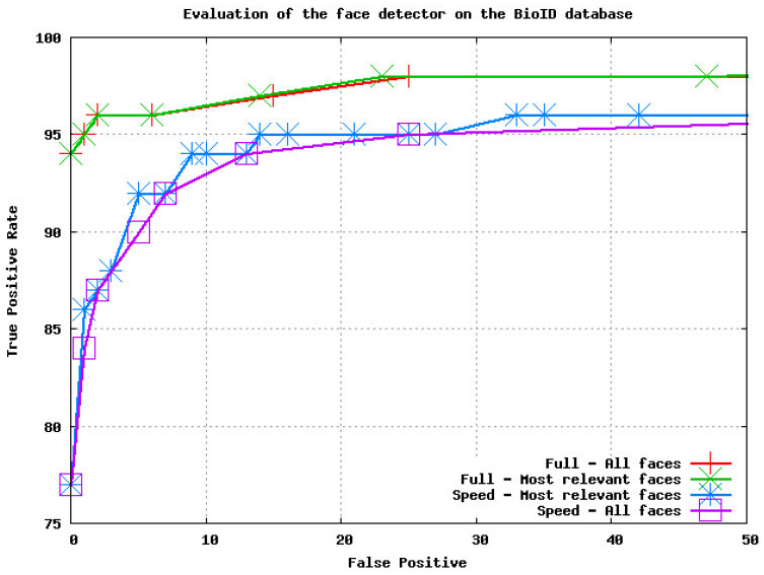


Figura 30: Resultados detector de caras en base de imágenes BioID (1521 imágenes). Extraído de [8].

El siguiente módulo del sistema construido es la detección de piel, la cual usa, como ya está dicho, la información de las regiones de caras para construir el modelo a usar. Por lo mismo es difícil medir la tasa de detección de este sistema, ya que aunque existen bases públicas con imágenes marcadas de piel y no-piel, éstas no necesariamente incluyen zonas con caras por lo que este sistema no sería aplicable en esas condiciones. Sin embargo, el sistema implementado en este trabajo es muy similar al descrito en [5], en donde sí se midieron las tasas de detección usando una selección de imágenes de una base de datos pública en que se incluían caras. Los resultados se presentan en la Tabla 5, en que el sistema se examinó sobre 12 secuencias de vídeos obtenidas en [68]. Se recalca que aunque el sistema no es exactamente el mismo que el de este trabajo, es lo suficientemente parecido como para poder aplicar estos resultados acá.

Secuencia	DR [%]	FP [%]
Secuencia 2	62,10	0,40
Secuencia 4	68,50	2,30
Secuencia 6	82,30	0,20
Secuencia 7	68,40	2,50
Secuencia 8	67,70	1,80
Secuencia 9	83,70	0,00
Secuencia 10	86,90	15,80
Secuencia 11	64,00	0,00
Secuencia 15	74,60	2,60
Secuencia 16	73,20	1,30
Secuencia 18	95,60	3,20
Secuencia 21	68,80	6,50
Promedio	74,60	3,10

Tabla 5: Resultados del sistema de detección de piel descrito en [5] para una selección de 12 vídeos.

Los resultados obtenidos en [5], y que pueden ser extrapolados a lo realizado en este trabajo de título, son comparables a los presentados en [69] usando la misma base de datos, lo que habla de que el sistema implementado, a pesar de su simplicidad, obtiene resultados razonables. Aunque posiblemente el sistema puede ser mejorado, su principal uso es el alivianar la carga computacional al reducir los sectores donde el detector de manos posterior debe funcionar por lo que no es absolutamente necesario tener una detección de piel precisa, ya que no es el objetivo principal de este trabajo de título.

La parte que tomó el mayor tiempo durante la ejecución de este trabajo fue el crear los detectores de cada gesto y el detector de manos de manera que se obtuvieran resultados aceptables. Para esto se hizo necesario realizar muchas pruebas, con distintos conjuntos de entrenamiento, distintos parámetros y distintos tamaños de las imágenes. Para ilustrar el proceso llevado a cabo se muestran los resultados obtenidos con un conjunto de 6 detectores para 3 gestos distintos, como se puede ver en las Figura 31, Figura 32, Figura 33. Estas figuras muestran los resultados iniciales logrados con los detectores y los finalmente obtenidos, para tres gestos en particular. Es necesario hacer notar que estas tres figuras se muestran con el fin de enseñar la

evolución que tuvieron los detectores a lo largo del tiempo, y no representan el total de detectores construidos. La base de prueba sobre la que fueron evaluados se describe en las primeras tres filas de la Tabla 8. Las diferencias entre los detectores iniciales y los finales se observan en la Tabla 6. Aquí se puede notar que una de las principales diferencias entre detectores de un mismo gesto es la base de imágenes usadas, que en un principio fueron recolectadas de Internet (principalmente de [70]) para los detectores D2. Luego se decidió generar una base de imágenes en el marco de este mismo trabajo por medio del sistema de 'Active Learning' ya descrito con anterioridad para los detectores q finalmente fueron usados (D1). Otra diferencia que salta a la vista es el tamaño usado para las imágenes de entrenamiento, que en algunos casos se varío desde 24x24 píxeles en D2 a 24x42 en D1. La razón de este cambio es que se comprobó durante la realización de las pruebas la importancia de mantener la relación alto/ancho de las imágenes, ya que en caso contrario la distorsión producida sobre la imagen impide su detección correctamente. Por esto, durante la construcción de las bases de datos de entrenamiento se mantuvo la relación alto/ancho de las imágenes capturadas aún cuando fuera distinta de 1. Finalmente se nota que al realizar estos cambios el número total de características de los detectores nuevos es mayor, lo que habla de que la cantidad inicial en los detectores D2 no era suficiente para describir los gestos correctamente.

Gesto	Tamaño imág. Entrenamiento	N° imág. Entrenamiento positivas	N° imág. validación	Base de datos	N° imág. negativas	N° total de etapas	N° total de características
Puño (D1)	24x24	1194	1186	Active Learning	46746	9	612
Puño (D2)	24x24	795	606	IDIAP [70]	47950	10	190
Palma (D1)	24x42	526	497	Active Learning	45260	10	856
Palma (D2)	24x24	597	441	IDIAP [70]	36776	8	277
Índice (D1)	24x42	947	902	Active Learning	59364	12	339
Índice (D2)	24x24	836	631	IDIAP [70]	30648	8	173

Tabla 6: Comparación de características entre algunos detectores iniciales y los finales. Los detectores D1 fueron entrenados usando 'Active Learning' mientras que en los D2 se usaron bases de datos estándar.

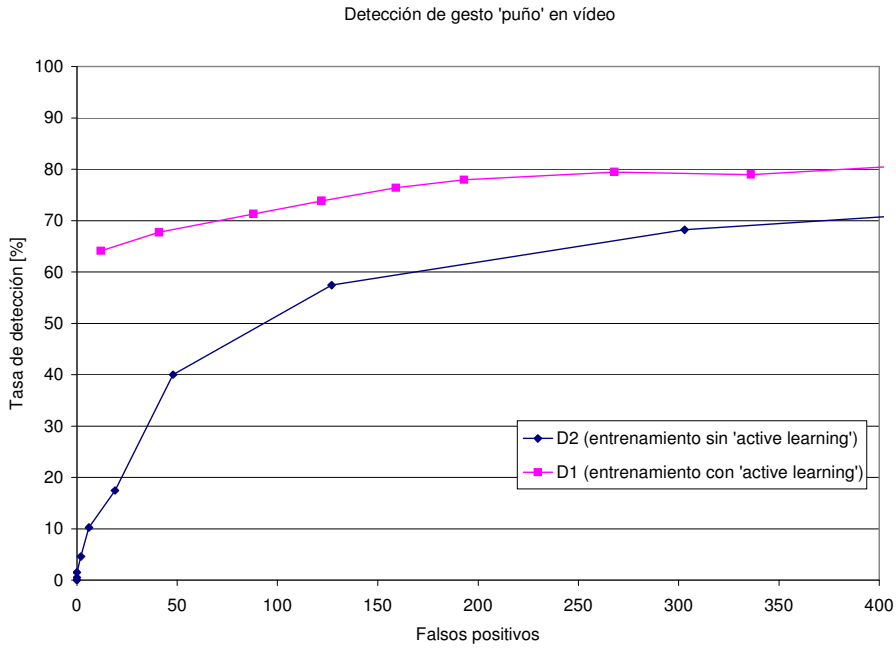


Figura 31: Detección del gesto 'puño' para dos detectores distintos usando base de imágenes descrita en la Tabla 8.

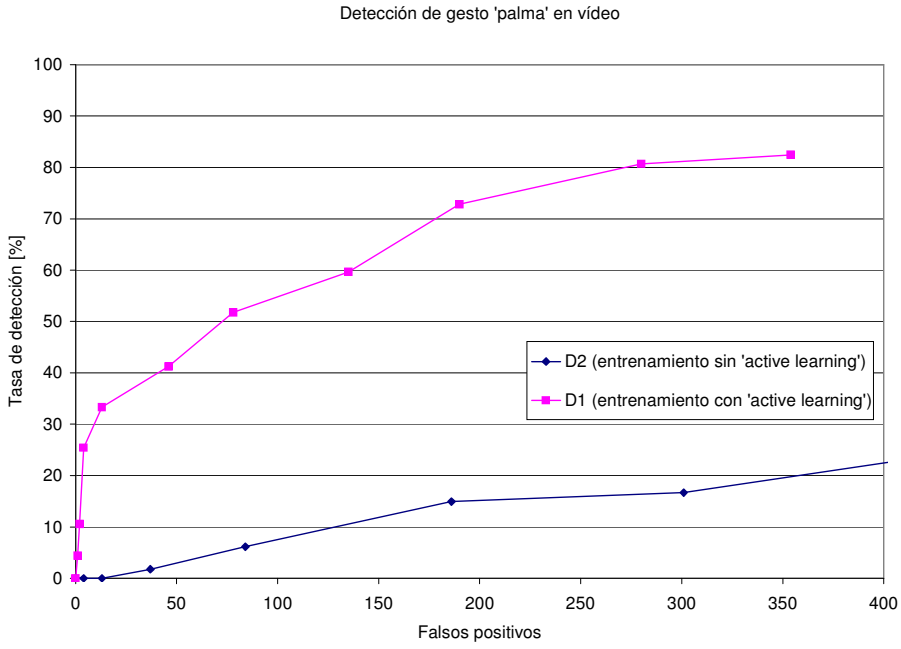


Figura 32: Detección del gesto 'palma' para dos detectores distintos usando base de imágenes descrita en la Tabla 8.

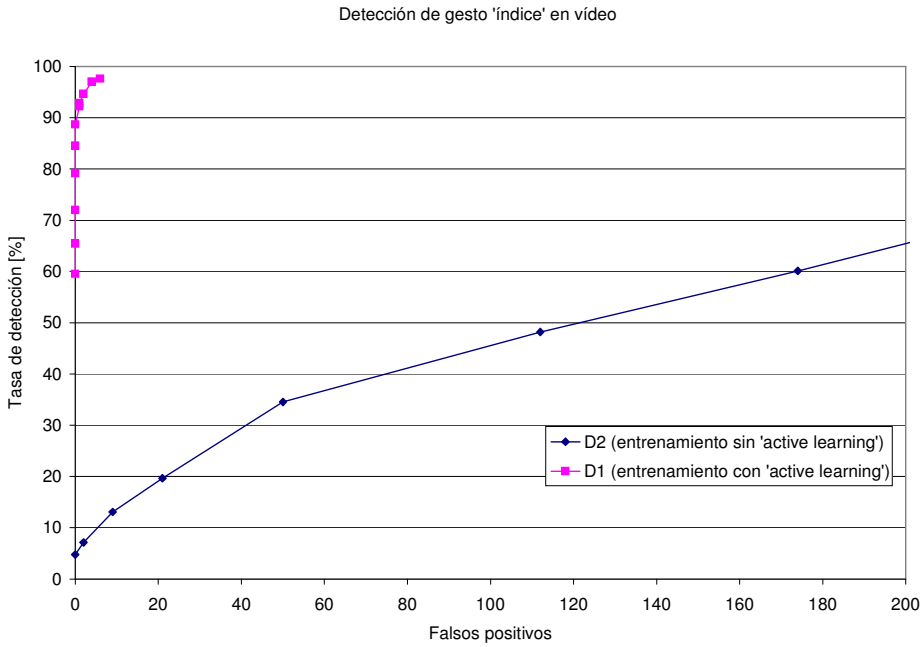


Figura 33: Detección del gesto 'índice' para dos detectores distintos usando base de imágenes descrita en la Tabla 8.

Luego de terminar los entrenamientos de los detectores de gestos, se eligieron aquéllos en que se tenían los mejores resultados, con lo que se consideraron 4 gestos para la aplicación: puño, palma, índice y cinco. Un resumen de estos detectores finales se explicitan en la Tabla 7, donde se indica el tamaño elegido para las imágenes de entrenamiento, el número de ellas y del conjunto de validación, el número de imágenes negativas usadas por el programa de entrenamiento, el total de etapas obtenidas y el número total de características (rectangulares o mLBP). Se observa que las características usadas son básicamente siempre mLBP, a excepción del gesto 'Cinco' en que en sus 2 primeras etapas se usan características rectangulares para luego usar en las siguientes mLBP. La razón de esto es que se encontró que las características mLBP tenían mejores resultados que las rectangulares sobre la detección de manos, pero en algunos casos se podían usar en etapas iniciales características rectangulares para acelerar el proceso de descartar zonas de no-gesto (ya que el cálculo de las características rectangulares es más rápido que el de mLBP). La cantidad de imágenes para entrenamiento necesarias de usar se encontró que debían ser del orden de o mayores a 800 para obtener resultados satisfactorios (con 50% de esas imágenes usadas como imágenes de entrenamiento positivas y el otro 50% como imágenes de validación), lo que concuerda con el número de imágenes usadas en trabajos similares [56] [63]. Los resultados logrados se muestran en la Figura 34 y en la Figura 35.

Gesto	Tamaño imág. Entrenamiento	Características usadas	N° imág. Entrenamiento positivas	N° imág. validación	N° imág. negativas	N° total de etapas	N° total de características
Puño	24x24	mLBP	1194	1186	46746	9	612
Palma	24x42	mLBP	526	497	45260	10	856
Índice	24x42	mLBP	947	902	59364	12	339
Cinco	24x24	Rect + mLBP	651	653	41859	9	356

Tabla 7: Resumen de los detectores de gestos finalmente implementados.

Gesto	Cantidad de imágenes de prueba	Tamaño imágenes de prueba
Puño	195	320x240
Palma	114	320x240
Índice	168	320x240
Cinco	147	320x240

Tabla 8: Base de imágenes de prueba usadas para calcular la tasa de detección en Figura 31, Figura 32, Figura 33, Figura 34 y Figura 36.

Gesto	Cantidad de imágenes de prueba	Tamaño imágenes de prueba
Puño	200	320x240
Palma	200	320x240
Índice	200	320x240
Cinco	200	320x240

Tabla 9: Base de imágenes de prueba usadas para calcular la tasa de detección en Figura 35.

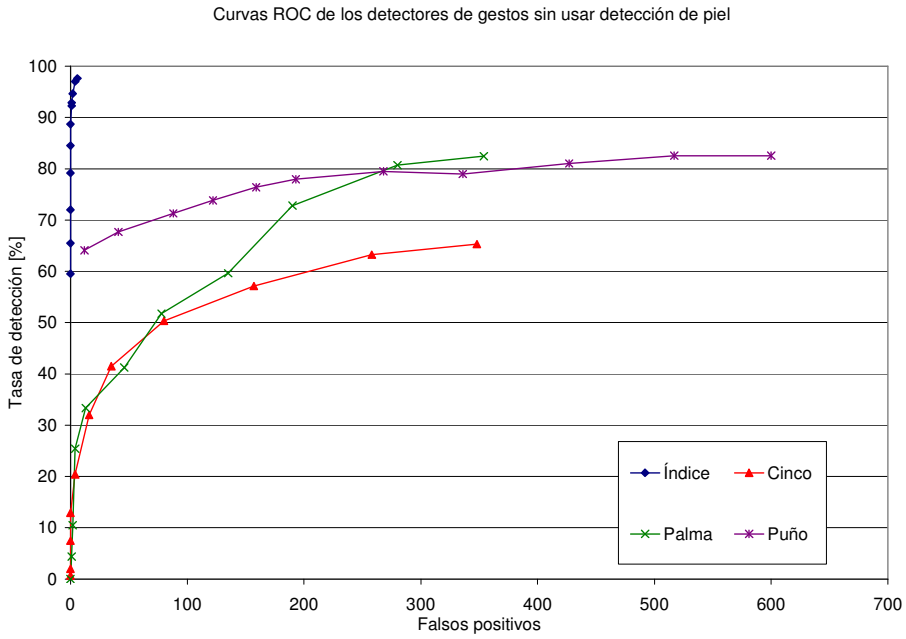


Figura 34: Curvas ROC de los detectores de gestos sobre toda la imagen usando la base de prueba descrita en la Tabla 8.

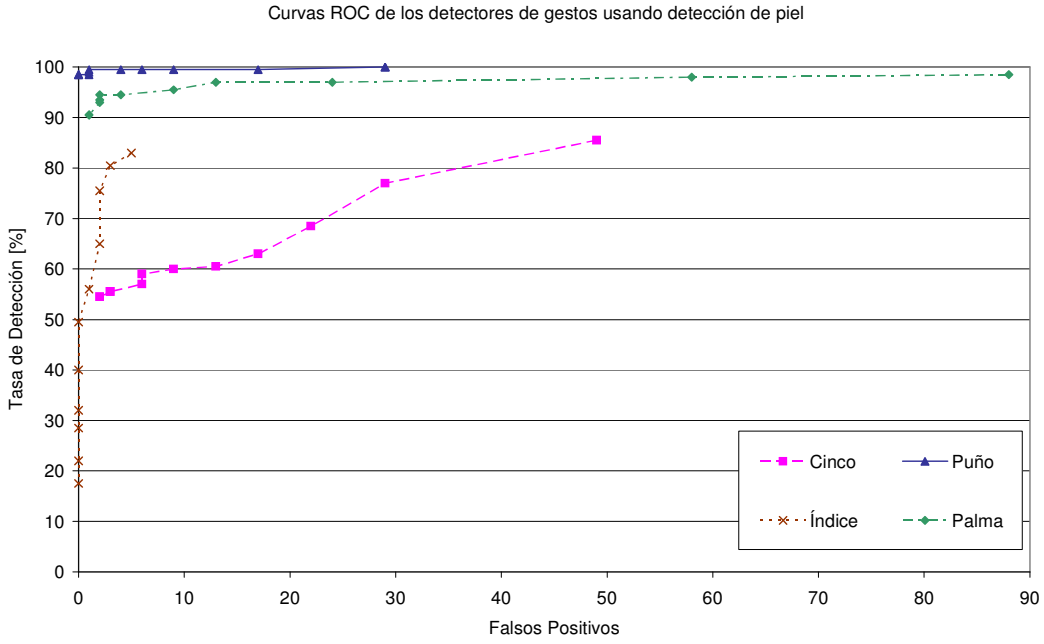


Figura 35: Curvas ROC de los detectores de gestos sobre zonas de piel usando la base de prueba descrita en la Tabla 9.

Es necesario hacer notar que los resultados obtenidos en la Figura 34 para los distintos detectores son sobre la base de pruebas indicada en la Tabla 8, y que en este caso no incluyen procesamientos extras, como por ejemplo de detección de piel. Es decir, estos resultados son los obtenidos al correr los detectores sobre las imágenes mencionadas sin realizar ningún procesamiento previo, por lo que la detección se hace sobre toda la imagen. En el caso en que sí se ocupa la etapa preliminar de selección de zonas de piel, como ocurre en el funcionamiento real del sistema, se obtienen los resultados de la Figura 35, en que se usó la base de prueba detallada en la Tabla 9.

En los resultados indicados en la Figura 34, es notoria la buena respuesta del detector del gesto 'índice', el cuál presenta una tasa de detecciones correctas superior al 95% con sólo 6 falsos positivos sobre un total de 168 imágenes. La razón de que este resultado sea superior a los otros no es clara, pudiendo deberse tal vez a la forma propia del gesto, pero en caso de que ese no sea el motivo, debieran poder obtenerse estos mismos resultados eventualmente con otros gestos si se realizan nuevos entrenamientos. De todas formas, para todos los gestos se logra una tasa de detección superior al 60%, lo que se considera suficiente para el sistema ya que con una alta tasa de cuadros por segundos del programa es casi seguro que el gesto será detectado en corto tiempo.

En el caso de lo obtenido en la Figura 35, como es de esperar al hacer uso del bloque previo de detección de piel, los falsos positivos son mucho menores en relación a lo logrado en los resultados de la Figura 34. Esto es así ya que los detectores funcionan sobre menos partes de las imágenes, disminuyendo la posibilidad de cometer detecciones erróneas. Además se observa que en el caso de algunos detectores la tasa de aciertos sube considerablemente, como el gesto 'puño' que logra tasas cercanas al 98% de efectividad sin ningún falso positivo sobre 200 imágenes. Por otro lado, el gesto 'índice' disminuye su tasa de detección pero mantiene un bajo número de falsos positivos sobre toda su curva. Aunque estos resultados pueden llegar a parecer contradictorios con lo que se obtuvo en la Figura 34, es necesario recalcar que ambas pruebas se hicieron sobre bases de imágenes distintas (ver Tabla 8 y Tabla 9), por lo que no son directamente comparables.

Gesto	Número de ejemplos entrenamiento	Número de atributos usados para entrenamiento
Puño	3838	15352
Palma	3750	15000
Índice	3743	14972
Cinco	3753	15012

Tabla 10: Datos usados para el entrenamiento del clasificador

El siguiente paso en el sistema es la clasificación de los gestos. En este caso el entrenamiento se realizó sobre el software WEKA [71] de minería de datos ('data mining'), usando como entrada los cuatro atributos mencionados (confidencia, número de ventanas, confidencia media, confidencia normalizada) y explicados en detalle en el capítulo 5. Estos cuatro atributos se midieron para los cuatro gestos, o clases, para el número de ejemplos que se detalla en la Tabla 10. En total son 15084 los ejemplos usados para el entrenamiento en WEKA. Se eligió como método de clasificación un árbol de decisión J48, debido a que fue el que presentó mejores resultados en el conjunto de prueba, usando validación cruzada, con un 90,8% de clasificaciones acertadas. Una comparación con otros tipos de clasificador con los que se probó el sistema se muestra en la Tabla 11. El árbol J48 finalmente usado tiene un total de 72 hojas y un tamaño de 143 nodos.

Clasificador	Clasificaciones correctas [%]	Clasificaciones incorrectas [%]	Error medio absoluto	Raíz error cuadrático medio
SVM	90,413	9,587	0,2588	0,3256
Árbol J48	90,824	9,176	0,0707	0,1958
Bayes Multinomial	82,696	17,304	0,0864	0,2918
Bayes	81,516	18,484	0,101	0,2896
Árbol aleatorio	86,316	13,684	0,0688	0,2603
Tabla de decisión	90,122	9,878	0,0737	0,1998

Tabla 11: Resumen de resultados obtenidos con distintos clasificadores.

Este clasificador luego fue aplicado sobre una serie de 5 vídeos realizados por 4 cuatro personas distintas. Estos vídeos muestran al usuario realizando los 4 gestos distintos durante un tiempo determinado, tanto en movimiento como estático. Los 5 vídeos en conjunto suman 8150 'frames', en los que la clasificación obtienen los resultados mostrados en la Tabla 12. Esta tabla indica la matriz de confusión del sistema, es decir, presenta el gesto clasificado según el programa y gesto que realmente estaba realizando el usuario. Se observa que la mayor confusión se produce entre el gesto 'puño' y el 'índice', y entre el gesto 'palma' y 'cinco'; es decir entre gestos que guardan cierto parece entre si.

Puño	Palma	Índice	Cinco	Desconocido	Predicción/Gesto
1533	2	870	9	15	Puño
39	1196	10	659	15	Palma
436	36	1503	27	86	Índice
103	32	6	1446	127	Cinco

Tabla 12: Matriz de confusión del sistema completo.

Un resumen de estos resultados se muestra en la Tabla 13, donde se ven solamente las clasificaciones correctas (TPR por 'true positive rate') y las mal clasificadas (FPR por 'false positive rate'). En promedio se obtiene una clasificación correcta en el 69,66% de los casos.

Gesto	TPR [%]	FPR [%]
Puño	63,112	36,888
Palma	62,324	37,676
Índice	71,983	28,017
Cinco	84,364	15,636
Promedio	69,669	30,331

Tabla 13: Tasas finales de detección y reconocimiento del sistema completo.

En la Tabla 14 se muestra el tiempo promedio de procesamiento del sistema completo, programado en C++, en un computador estándar (Pentium 4, 3.2 [GHz], 1 [GB] Ram, sobre Windows XP) y usando imágenes de tamaño 320x240 píxeles. De aquí se extrae que el sistema funciona en promedio a 5,8 [fps], lo que es suficiente para considerarlo como tiempo real, al menos en el contexto de este trabajo. Esto porque no existe un retraso molesto para el usuario del sistema que perjudique la comunicación hacia la máquina. Si no se considera la etapa de 'Dibujar en pantalla', que no es necesaria para el funcionamiento del algoritmo, se logra en promedio 6,6 [fps] en la operación dadas las mismas condiciones.

Otra información importante que se puede extraer de lo expuesto en la Tabla 14 son las partes más costosas computacionalmente dentro del algoritmo: el reconocimiento del gesto, la captura del cuadro y el dibujar los resultados en pantalla. El primero de ellos, el reconocimiento del gesto, es un resultado esperado, ya que en esta etapa es necesario correr todos los detectores para cada gesto dentro del diccionario, lo que es necesariamente un procedimiento trabajoso. Sin embargo el segundo proceso que más tiempo toma es la captura del cuadro, lo que produce cierta sorpresa ya que la cámara usada durante este proyecto, una webcam Philips modelo SPC 900NC, declara tener una velocidad de captura de hasta 90 [fps]. Claramente esto no sería posible con el

tiempo que demora en realizar este proceso dentro de la aplicación, ya que incluso en el caso de ser el único proceso que funcione dentro del programa igual se lograría solamente un promedio de 19,6 cuadros por segundo. Este resultado se considera que se debe a problemas de OpenCV en el entorno de Windows para capturar cuadros de la cámara, debido al uso de una interfaz antigua de este sistema. Por lo tanto si se quisiera mejorar estos resultados sería necesario usar algún método distinto para capturar cuadros que no sea OpenCV, o usarlo en otra plataforma (por ejemplo Linux). Finalmente el tiempo tomado por la aplicación en graficar en pantalla los resultados, aunque alto, no se considera relevante, ya que no es necesario para el funcionamiento del mismo, si no que tiene solamente finalidades informativas para el usuario.

Etapa del sistema	Tiempo promedio por cuadro [ms]
Captura de cuadro	51,304
Conversión formato (OpenCV a formato propio)	10,844
Detección de piel	4,456
Detección de cara	0,861
Tracking de cara	1,621
Detección de mano	2,687
Reconocimiento de gesto	78,967
Dibujar en pantalla	21,205
Tiempo total	171,945

Tabla 14: Velocidad de procesamiento promedio de las distintas etapas del sistema sobre una imagen de 320x240.

5.2. Comparación con otros métodos similares existentes

La comparación del sistema implementado en este trabajo se realizará en relación con los trabajos presentados en el capítulo 2, en la sección 'Estado del arte', además del trabajo de título [72].

En primer lugar se cotejarán los resultados, específicamente los de la sección de detección, obtenidos en [9] con los logrados en la presente memoria. En [9] se detectan 10 gestos, 3 de los cuales son iguales a los gestos 'puño' (A), 'índice' (L) y 'palma' (B) detectados en el sistema desarrollado, por lo que el paralelo se hará sobre ellos. Debido además a que la base sobre la que

se examinaron los resultados de [9] es conocida, se usa esta misma base sobre los detectores propios. Lo obtenido se muestra en la Tabla 15, donde se ve que los resultados en la detección son bastante similares. La comparación en el plano de la clasificación de lo detectado no es posible hacerla, ya que el sistema aquí desarrollado está pensado para funcionar sobre vídeos, por lo que no es posible medir su funcionamiento sobre imágenes estáticas.

		Fondo simple		Fondo complejo	
	Gesto	FP	DR [%]	FP	DR [%]
Detectores [9]	Puño (A)	S/I	100	S/I	100
	Palma (B)	S/I	100	S/I	100
	Índice (L)	S/I	100	S/I	100
Detectores propios	Puño	0	100	13	100
	Palma	1	100	8	97,2
	Índice	0	100	5	98,6

Tabla 15: Comparación de los detectores en [9] con los de esta memoria.

La otra comparación que es posible hacer es con los detectores de [56], cuyos resultados se dieron en la Figura 5. Como en aquel trabajo no se hace uso de ninguna etapa de preprocesamiento, como es el uso de detección de piel, se usará como comparación los resultados de la Figura 34, en que tampoco se hace uso de etapas anteriores. Debido a que en ese trabajo los falsos positivos se presentan en relación al número de ventanas total analizadas, es necesario cambiar la escala del gráfico mostrado en la Figura 34. Para esto se necesita entonces conocer ese número de ventanas, el que se encuentra a través de la siguiente fórmula que relaciona el tamaño de la imagen y el factor de escalamiento (definido en la sección 3.2.6):

$$N_w \approx H \cdot W \cdot \frac{1}{1 - s^{-2}}$$

Donde H y W es el alto y ancho de la imagen respectivamente, y s es el factor de escalamiento ya señalado. En este caso el factor de escalamiento tiene un valor de 1,1, mientas que el tamaño de las imágenes es de 320x240, según la Tabla 8. Si se multiplica este resultado por el total de imágenes analizadas, y se divide el número de falsos positivos por este resultado se tiene lo buscado. Se construye entonces un gráfico similar al mostrado en la Figura 34 pero con los falsos positivos en esta nueva escala. Este gráfico se aprecia en la Figura 36 y permite comparar de

manera más directa con lo mostrado en la Figura 5 para el sistema [56], aún cuando estos resultados se obtuvieron en una base de prueba de imágenes diferente. En este caso existen tres gestos parecidos que se analizaron en ambos trabajos: 'indice' (o 'sidepoint'), 'palma' ('closed') y cinco ('open'). Estos gestos tampoco son exactamente los mismos ya que en un caso los gestos miran con la palma hacia la cámara y en el otro se miran por el lado anverso, pero se pueden considerar suficiente parecidos como para hacer la comparación. Se observa que aún cuando en el sistema presentado en [56] se logran obtener en general mejores tasas de detección, éstas vienen acompañas de también altas tasas de falsos positivos, lo que no permite que sean usadas en la práctica en ningún sistema de detección de gestos confiable en esos puntos. Por lo tanto se comparan los resultados para una cantidad de falsos positivos de $5 \cdot 10^{-6}$ por cada ventana analizada (o de 0,0005%). Aquí se obtiene que para dos gestos ('cinco' y 'palma') se logran mejores resultados en [56], y que estos resultados son aproximadamente un 15% superior. Mientras que para el gesto 'índice' la salida lograda en esta memoria es muy superior a lo logrado en [56], ya que allí se obtiene una tasa de detecciones correctas de aproximadamente un 98% con $1 \cdot 10^{-4}$ falsas detecciones (o equivalentemente 0,01%), siendo que para la misma tasa de detección los falsos positivos son menores en cuatro órdenes de magnitud en lo logrado en esta memoria.

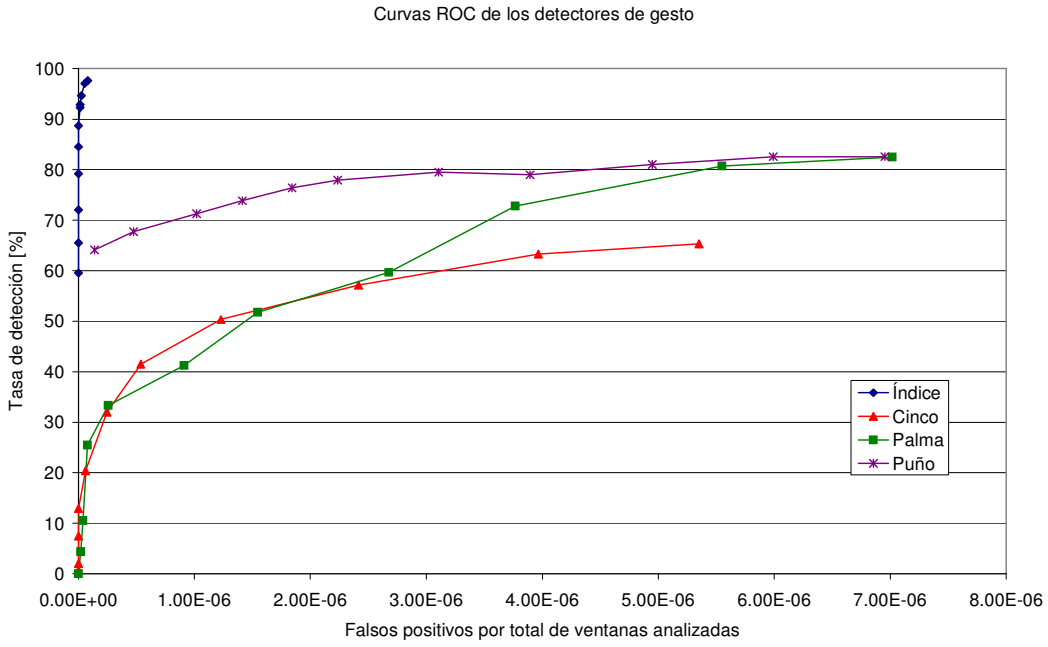


Figura 36: Curvas ROC de los detectores de gestos con falsos positivos en relación al total de ventanas analizadas.

El trabajo desarrollado en [58] presenta los mejores resultados conocidos en sistemas de detección y clasificación de gestos, con tasas de 99,8% y 97,4% respectivamente. Claramente estos números son muy superiores a lo presentado en este trabajo pero, como ya se mencionó en su momento, estos se lograron usando imágenes con fondos relativamente simples y no en condiciones más generales de uso. Además el hecho de que no se muestren los números relativos a los falsos positivos de ese sistema no permite hacer una buena comparación entre ellos.

Otro sistema presentado en la sección de 'Estado del arte' es el que se encuentra en [63], donde se aplican 3 detectores en paralelo sobre una imagen y luego se realiza la clasificación, de manera similar a lo efectuado en el presente trabajo. En el único gesto en que ambos sistemas tienen en común, el gesto 'palma', en [63] se obtiene un TPR de 90% con 0 falsos positivos en 100 imágenes, mientras que en esta memoria el TPR más alto logrado, para ese gesto, es de cerca de un 80% con más de 300 falsos positivos sobre cerca de 120 imágenes. Sin embargo, nuevamente se observa que las pruebas son realizadas sobre imágenes sencillas, en que se presenta el objeto sobre fondos simples, y en que además no existe ningún elemento extra además de las manos. Por lo tanto una comparación directa entre ambos sistemas no sería justa dado la disparidad que hay entre las imágenes de prueba.

Otra variable que es interesante de analizar es el tiempo de procesamiento del sistema. En este punto solamente se conoce la información de dos trabajos: [63] y [72]. El primero indica el tiempo que toma en promedio la detección de cuatro distintos gestos en una imagen, lo que en total suma 0,103 [seg] (ver Tabla 3). Dado que en el sistema implementado en esta memoria, de acuerdo a la Tabla 14, la etapa de detección y clasificación de los cuatro gestos toma 0,079 [seg] en total, se puede concluir que este último funciona a mayor velocidad para el mismo número de gestos. Esta comparación obviamente es válida si es que ambos sistemas se prueban en condiciones similares, es decir se asume que en [63] se usan imágenes de tamaño 320x240, al igual que en este trabajo, y que los computadores usados para realizar los ensayos son equivalentes.

La otra comparación que es posible hacer es con la memoria realizada en [72], donde se usan redes neuronales para la detección y clasificación de gestos. En este caso la comparación es posible de hacer de manera más directa, ya que se cuenta con el software desarrollado en ese trabajo. Al correr este programa sobre el mismo equipo usado anteriormente para medir los tiempos en la Tabla 14, y usando también imágenes de tamaño 320x240, se logra una velocidad promedio del algoritmo de 3,95 [fps], cerca de 3 [fps] más lento que lo logrado en este trabajo de título.

5.3. Discusión y Análisis de Resultados

Se ha presentado en la sección anterior los resultados para el sistema de detección y de clasificación de gestos, además de los tiempos de procesamiento obtenidos en las diversas etapas del sistema. Al comparar los resultados logrados para la detección de gestos en el caso en que no se hace uso de una etapa previa de detección de piel y aquéllos en que sí se usa, se observan varias diferencias. La primera, es que para todos los casos se logra reducir el número de falsos positivos obtenidos. Por otro lado, las tasas de detección de los detectores varían ampliamente entre ambos casos, lográndose mejores resultados para todos los gestos, menos para el denominado 'índice'. Esto a pesar de que los detectores usados son los mismos. Pero dado que las bases de imágenes en que se hicieron las pruebas fueron distintas para ambos casos, estas diferencias no son directamente comparables. Por la misma razón se hace difícil realizar una comparación con otros sistemas similares creados, ya que al no contarse con una base de prueba estándar para estos casos, los resultados necesariamente serán distintos. De todas formas se observa que aún en el peor de los casos se pueden logran tasas de detección en todos los gestos superiores al 60%, número que se considera suficiente para este trabajo, ya que con el 'frame rate' obtenido acá, que es de alrededor 6,6 [cuadros/seg], se asegura que el gesto será detectado en corto tiempo. Finalmente, como puntos de operación de los detectores fueron elegidos aquellos que se encontraban alrededor de los 10 falsos positivos sobre 200 imágenes, según lo visto en la Figura 35, dado que se considera que es un número aceptable para un buen funcionamiento del sistema.

Capítulo 6

Conclusiones

A continuación se procede a extraer las principales conclusiones correspondientes al trabajo realizado en esta memoria. Además se mencionan algunas ideas para posibles trabajos futuros en esta área que complementen lo acá presentado.

A lo largo de este trabajo se ha estudiado el problema de la detección y clasificación de gestos realizados con las manos. Inicialmente, las dificultades que presenta esta tarea y las múltiples formas de solución que han surgido a lo largo del tiempo fueron presentados, haciendo hincapié en la separación entre sistemas de localización de manos en una imagen y sistemas de detección de gestos. Luego de esta presentación de la problemática a resolver, se discutieron las bases teóricas usadas en la solución planteada en este trabajo. En particular se expuso el clasificador 'Adaboost' y su uso en cascadas interconectadas, además de sistemas de detección de piel a través del color. Por último se ha presentado la implementación de un programa capaz de detectar y clasificar gestos hechos con las manos, junto con los resultados obtenidos en su uso.

El programa de detección y clasificación de gestos desarrollado considera el uso de un vocabulario gestual limitado pero demostrativo de las posibilidades del sistema. Para esto se eligieron cuatro gestos: 'puño', 'palma', 'índice' y 'cinco'.

Los resultados obtenidos se dividen en dos partes: detección y clasificación. La etapa de detección considera la ubicación de los gestos en la imagen sin ningún tipo de ayuda de módulos auxiliares, como sería su uso en zonas con detecciones de piel positivas; mientras que la salida de la etapa de clasificación considera los resultados del sistema completo, es decir integrando lo obtenido en la detección de piel y en la detección de gestos.

En la etapa de detección se obtuvo una tasa de éxito mayor a un 90% con menos 10 de falsos positivos en cerca de 150 imágenes de 320x240 con el gesto 'índice', mientras que con los gestos 'palma' y 'puño' las tasas son cercanas al 80% con 300 falsos positivos, y para el mismo número de detecciones erradas el gesto 'cinco' logra un 65% de éxito en igual número de imágenes de prueba. Estos resultados fueron comparados con sistemas similares implementados en otros trabajos, comprobándose que aun cuando en esta memoria las pruebas se realizaron sobre imágenes no controladas, con iluminación variable y fondos complejos, los resultados son en la mayoría de los casos iguales o levemente inferiores a aquéllos en que las pruebas se hicieron sobre conjuntos de pruebas más sencillos.

Para realizar pruebas sobre el sistema completo se capturaron 5 vídeos con 4 personas distintas, cada una realizando los distintos gestos posibles de interpretar por el sistema, con una duración aproximada de 80 [seg.]. Los resultados se obtienen a través de la tasa de clasificaciones correctamente realizadas para el gesto que muestra en cada cuadro de los vídeos. En promedio se obtiene una tasa de resultados positivos de alrededor de un 69% en esta clasificación. En particular lo logrado para cada gesto es: 'puño' 63%, 'palma' 62%, 'índice' 72% y 'cinco' 84%. La mayor confusión en la clasificación se produce entre 'puño' e 'indice' y entre 'palma' y 'cinco'.

El sistema completo funciona a una velocidad media de 6,6 [fps] sobre imágenes de tamaño de 320x240, si no se considera la etapa de graficar en pantalla. Esta velocidad es considerada suficiente para que no se produzcan retrasos molestos para el usuario, por lo que se plantea que el sistema funciona en tiempo real. Esto aun cuando la cifra obtenida es bastante menor a la definición formal de tiempo real, que habla de velocidades mayores a 30 [fps], por lo que este término ha sido usado de manera más libre en este trabajo.

Todo el sistema descrito anteriormente está implementado en C++ y posee una interfaz gráfica que hace uso de la librería OpenCV, mostrándose resultados que podrían ser usados en aplicaciones de control sencillas, como por ejemplo darle algunas órdenes básicas a un robot.

Al analizar el trabajo realizado, las mejoras, proyecciones y ampliaciones del sistema realizado son variadas. Las que se consideran más relevantes serán presentadas aquí, divididas según el área en que está enfocada:

1. Ampliar vocabulario gestual: los cuatro gestos que es capaz de reconocer la aplicación implementada pretende ser solamente demostrativo de las posibilidades del sistema, pero puede llegar a ser un número muy limitado para su uso como medio de control o comunicación real. Por lo mismo, la ampliación del número de gestos reconocidos debiera ser parte de futuros trabajos en el sistema. Sin embargo, debe tenerse en

cuenta que necesariamente un mayor número de gestos a detectar y clasificar traerá consigo un aumento lineal del tiempo de procesamiento de la etapa de 'Reconocimiento de gesto' (ver Tabla 14), con lo que se podría perder la característica necesaria de funcionamiento en tiempo real. Para enfrentar esto se proponen algunas mejoras en la sección siguiente.

2. Aumentar velocidad del algoritmo: La velocidad con la que funciona el programa está determinada, como se vio en la Tabla 14, por dos etapas: la del reconocimiento del gesto y la de captura del cuadro. En la etapa de reconocimiento de gesto, la parte que toma una mayor cantidad de tiempo es la detección en paralelo de todos los gestos. Este tiempo viene dado básicamente porque para cada detector se realiza un análisis multiescala sobre la imagen, es decir, se corre el clasificador sobre múltiples ventanas extraídas de varias versiones de la misma imagen a diferentes escalas. En el sistema implementado, hasta ahora se hace uso de la posición de las manos dentro de la imagen, que se conoce gracias a la etapa de 'Tracking', para saber en que sectores aplicar los detectores. Sin embargo, la escala no se ajusta por si sola en el sistema de 'Tracking' usado, por lo que es necesario hacer el análisis multiescala. Esta parte se podría omitir, o por lo menos reducir considerablemente el tiempo usado en ella, si se realizan algunos cambios sobre la etapa de 'Tracking' para que ajuste por si sola la escala y luego se integre esa información al sistema de detección. Así, el clasificador solo necesitaría funcionar sobre un número mucho menor de ventanas. Se podría disminuir aún más el tiempo de cálculo si se llegan a integrar todos los detectores con el fin de realizar este análisis solamente una vez.

   La segunda etapa de menor velocidad en el algoritmo es el de 'Captura de cuadro'. Una forma de aumentar la velocidad del programa sería usar un sistema distinto a OpenCV para realizar esta función, ya que aquí parece radicar el problema. Una alternativa sería seguir haciendo uso de OpenCV, pero en el entorno de Linux, donde se estima no debiesen ocurrir estos problemas.

3. Mejorar clasificador: el clasificador de gestos que se obtuvo tiene un porcentaje de éxito promedio cercano al 69%, por lo que su uso en condiciones prácticas puede ser algo limitado. Se considera necesario mejorar el sistema de clasificación, para lo que sería necesario posiblemente agregar otros atributos que ayuden a tomar una mejor decisión. Se podrían considerar como nuevos atributos características relacionadas con el 'blob' de piel de la mano (e.g. la relación alto/ancho de esa ventana).

4. Mejorar detección: la tasa de éxito obtenida con el gesto 'índice', de más de un 90% con muy pocos falsos positivos, hace pensar que los resultados en la etapa de detección pueden ser mejorados sustancialmente para los otros gestos, ya que no parecieran haber diferencias importantes entre ellos como para que estos no puedan llegar a ser similares. Para esto sería necesario volver a entrenar los gestos con peores resultados dando especial atención a seguir los mismos pasos que se hicieron al entrenar el gesto 'índice'.
5. Gestos dinámicos: una vez que se tiene un sistema de detección de gestos estáticos robusto el paso lógico siguiente es ampliarlo para hacer uso de gestos dinámicos. Esto significaría probablemente implementar un sistema basado en HMM (Modelos ocultos de Markov), que son el método más usado en aplicaciones que involucran gestos dinámicos.
6. Considerar rotaciones en el plano: una de las limitaciones más grandes del sistema implementado es que no puede detectar un gesto que se encuentre dentro del diccionario gestual usado pero que se encuentre rotado de la posición en que fue originalmente entrenado. Se debiera considerar alguna forma de resolver esta dificultad, como por ejemplo usar la información de la detección de piel para calcular los momentos de esta imagen, a partir de la cual se puede inferir el ángulo de rotación del objeto.

Referencias

[1] F.K. Quek, "Toward a vision-based hand gesture interface", en Proceedings of the Conference on Virtual Reality Software and Technology, 23-26 de Agosto de 1994, Singapur (Singapur), pp. 17-31.

[2] F. Quek, "Eyes in the interface", en Image and Vision Computing, 1995, Vol. 13, No. 6, pp. 511-525.

[3] D. Mcneill y E. Levy, "Conceptual representations in language activity and gesture", Speech, palce and action: Studies in deixis and related topics, Wiley, 1982.

[4] V.I. Pavlovic, R. Sharma, y T.S. Huang. "Visual interpretation of hand gestures for human-computer interaction: A review", en IEEE Transaction on Pattern Analysis and Machine Intelligence, Julio 1997, Vol.19, No.7, pp. 677-695.

[5] M. Wimmer y B. Radig, "Adaptive Skin Color Classificator", en International Journal on Graphics, Vision and Image Processing, 2006, Special Issue on Biometrics, pp. 41-46.

[6] Y. Freund y R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", en Journal of Computer and System Sciences, Agosto 2006, Vol. 55, No. 1, pp. 119-139.

[7] R.E. Schapire y Y. Singer, "Improved Boosting Algorithms using Confidence-rated Predictions", en Machine Learning, 1999, Vol. 37, No. 3, pp. 297-336.

[8] R. Verschae, J. Ruiz-del-Solar, M. Correa y P. Vallejos, "A Unified Learning Framework for Face, Eyes and Gender Detection using Nested Cascades of Boosted Classifiers", en Technical Report UCH-DIEVISION-2006-02, 2006, Dept. of E. Eng., U. de Chile.

[9] A. Just, Y. Rodriguez y S. Marcel, "Hand Posture Classification and Recognition using the Modified Census Transform", en Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, 10-12 de Abril de 2006, Southampton (Inglaterra), pp. 351-356.

[10] B. Fröba y A. Ernst, "Face detection with the modified census transform", en 6th International Conference on Automatic Face and Gesture Recognition, 17-19 de Mayo de 2004, Seúl (Corea del Sur), pp. 91–96.

[11] P. Viola y M. Jones, "Rapid object detection using a boosted cascade of simple features", en Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 8-14 de Diciembre de 2001, Kauai HI (USA), Vol. 1, pp. I-511-I-518.

[12] R. Verschae y J. Ruiz-del-Solar, "A Hybrid Face Detector based on an Asymmetrical Adaboost Cascade Detector and a Wavelet-Bayesian-Detector", en Lecture Notes in Computer Science, Springer, 2003, Vol. 2686, pp. 742-749.

[13] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Statistical_classification, en Agosto 2007.

[14] M. Shin, K. Chang y L. Tsap, "Does Colorspace Transformation make any Difference on Skin Detection?", en Proceeding of the 6th IEEE Workshop on

Applications of Computer Vision, 3-4 de Diciembre de 2002, Orlando FL (USA), pp. 275-279.

[15] A. Albiol, L. Torres E. Delp, "Optimum color spaces for skin detection", en Proceedings of the 2001 International Conference on Image Processing, 7-10 de Octubre de 2001, Thessaloniki (Grecia), Vol. 1, pp. 122-124.

[16] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Bayer_filter, en Agosto 2007.

[17] J. Yang, W. Lu y A. Waibel, "Skin-color modeling and adaptation", en Proceedings of the 3rd Asian Conference on Computer Vision, 8-10 de Enero de 1998, Hong Kong (China), Vol. II, pp. 687-694.

[18] M.H. Yang y N. Ahuja, "Gaussian Mixture Model for Human Skin Color and Its Application in Image and Video Databases", en Proceedings of the SPIE: Conference on Storage and Retrieval for Image and Video Databases, 26-29 de Enero de 1999, San Jose CA (USA), Vol. 3656, pp. 458-466.

[19] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/RGB_color_space, en Agosto 2007.

[20] D. A. Brown, I. Craw y J. Lewthwaite, "A SOM Based Approach to Skin Detection with Application in Real Time Systems", en Proceedings of the British Machine Vision Conference, 10-13 de Septiembre de 2001, Manchester (Inglaterra), pp. 491-500.

[21] C. Garcia y G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis", en IEEE Transactions on Multimedia, Septiembre 1999, Vol. 1, No. 3, pp. 264-277.

[22] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/HSV_color_space, en Agosto 2007.

[23] S. J. McKenna, S. Gong y Y. Raja, "Modelling facial colour and identity with gaussian mixtures", en Pattern Recognition, 1998, Vol. 31, No. 12, pp. 1883-1892.

[24] R. Hsu, M. Abdel-Mottaleb y A. K. Jain, "Face detection in color images", en IEEE Transactions on Pattern Analysis and Machine Intelligence, Mayo 2002, Vol. 24, No. 5, pp. 696-706.

[25] F. Marqués y V. Vilaplana, "A morphological approach for segmentation and tracking of human face", en Proceedings of the 15th International Conference on Pattern Recognition, 3-7 de Septiembre de 2000, Barcelona (España), Vol. 1, pp. 1064-1067.

[26] Y. Dai y Y. Nakano, "Face-texture model based on SGLD and its application in face detection in a color scene", en Pattern Recognition, 1996, Vol. 29, No. 6, pp. 1007–1017.

[27] J. Cai, A. Goshtasby y C. Yu, "Detecting human faces in color images", en Proceedings of the International Workshop on Database Management Systems, 5-7 de Agosto de 1998, Dayton OH (USA), pp. 124-131.

[28] V. Vezhnevets, V. Sazonov y A. Andreeva, "A survey on pixel-based skin color detection techniques", en Proceedings of 13th International Conference on Computer Graphics and Applications, 5-10 de Septiembre de 2003, Moscú (Rusia), pp. 85-92.

[29] P. Peer, J. Kovac y F. Solina, "Human skin colour clustering for face detection", en Internacional Conference on Computer as a Tool, 22-24 de Septiembre de 2003, Ljubljana (Eslovenia), Vol. 2, pp. 144-148.

[30] M. Soriano, S. Huovinen, B. Martinkauppi y M. Laaksonen, "Skin detection in video under changing illumination conditions", en Proceedings of the 15th

International Conference on Pattern Recognition, 3-7 de Septiembre de 2000, Barcelona (España), Vol. 1, pp. 839–842.

[31] M. Jones y J. Rehg, "Statistical color models with application to skin detection", en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 23-25 de Junio de 1999, Fort Collins CO (USA), Vol. 1, pp. 274–280.

[32] Q. Zhu, K. Cheng, C. Wu y Y. Wu, "Adaptive learning of an accurate skin-color model", en Proceedings of the 6th IEEE Conference on Automatic Face and Gesture Recognition, 17-19 de Mayo de 2004, Seúl (Corea del Sur), pp. 37-42.

[33] P. Kakumanu, S. Makrogiannis y N. Bourbakis, "A survey of skin-color modeling and detection methods", en Pattern Recognition, 2007, Vol. 40, No. 3, pp. 1106-1122.

[34] B. Menser y M. Wien, "Segmentation and tracking of facial regions in color image sequences", en Proceedings of the SPIE Conference on Visual Communications and Image Processing, 20-23 de Junio de 2000, Perth (Australia), Vol. 4067, pp. 731–740.

[35] N. Oliver, A. Pentland y F. Berard, "Lafter: Lips and face real time tracker", en Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 17-19 de Junio de 1997, San Juan (Puerto Rico), pp. 123–129.

[36] B. Bascle y R. Deriche, "Region tracking through image sequences", en Proceedings of the 5th Conference on Computer Vision, 20-23 de Junio de 1995, Cambridge MA (USA), pp.302-307.

[37] G. Hager y P. Belhumeur, "Real-time tracking of image regions with changes in geometry and illumination", en Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 18-20 de Junio de 1996, San Francisco CA (USA), pp. 403-410.

[38] S. Sclaroff y J. Isidoro, "Active blobs", en Proceedings of the 6th International Conference on Computer Vision, 4-7 de Enero de 1998, Bombay (India), pp. 1146-1153.

[39] A. Jepson, D. Fleet, y T. El-Maraghi, "Robust online appearance models for visual tracking", en Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 8-14 de Diciembre de 2001, Kauai HI (USA), Vol. 1, pp. I-415-I-422.

[40] C. Hua, H. Wu, Q. Chen y T. Wada, "A General Framework For Tracking People", en Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, 10-12 de Abril de 2006, Southampton (Inglaterra), pp. 511-516.

[41] D. Comaniciu, V. Ramesh, P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift", en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 13-15 de Junio de 2000, Hilton Head Island SC (USA), Vol. 2, pp. 142-149.

[42] D. Comaniciu, V. Ramesh y P. Meer, "Kernel-Based Object Tracking", en IEEE Transactions on Pattern Analysis and Machine Intelligence, Mayo 2003, Vol. 25, No. 5, pp. 564-577.

[43] V. Nedovic, M. Liem, M. Corzilius y M. Smids, "Kernel-based object tracking using adaptive feature selection", en Project Report, 2005, Faculty of Science, Information & Communication Technology, U. van Amsterdam.

[44] Wolfram Mathworld, http://mathworld.wolfram.com/NonparametricEstimation.html, en Agosto 2007.

[45] Kernel Regresion, http://people.revoledu.com/kardi/tutorial/regression/kernelregression/, en Agosto 2007.

[46] J. Lee y T. Kunii, "Constraint-based hand animation", en Models and Techniques in Computer Animation, Springer, Tokyo, 1993, pp. 110–127.

[47] A. Erol, G. Bebis, M. Nicolescu, R. Boyle, X. Twombly, "Vision-Based Hand Pose Estimation: A Review", en Computer Vision and Image Understanding (special issue on Vision for Human-Compute Interaction), Octubre 2007, Vol. 108, No. 1-2, pp. 52-73.

[48] C. Joslin, A. El-Sawah, Q. Chen y N. Georganas, "Dynamic Gesture Recognition", en Proceedings of the IEEE Instrumentation and Measurement Technology Conference, 16-19 de Mayo de 2005, Ottawa (Canadá), Vol. 3, pp. 1706-1711.

[49] A. Ramamoorthy, N. Vaswani, S. Chaudhury y S. Bannerjee, "Recognition of Dynamic Hand Gestures", en Pattern Recognition, 2003, Vol. 36, No. 9, pp. 2069-2081.

[50] X. Yin y M. Xie, "Finger identification and hand posture recognition for human-robot interaction", en Image Vision Computing, 2007, Vol. 25, No. 8, pp. 1291-1300.

[51] K. Nickel y R. Stiefelhagen, "Visual recognition of pointing gestures for human–robot interaction", en Image Vision Computing, 2007, Vol. 25, No. 12, pp. 1875-1884.

[52] K. Kim, K. Kwak y S. Young, "Gesture analysis for human-robot interaction", en The 8th International Conference on Advanced Communication Technology, 20-22 de Febrero de 2006, Gangwon-do (Corea del Sur), Vol. 3, pp. 1824-1827.

[53] S. Malik y J. Laszlo, "Visual touchpad: A two-handed gestural input device", en Proceedings of the 6th International Conference on Multimodal Interfaces, 13-15 de Octubre de 2004, State College PA (USA), pp. 289–296.

[54] J. Letessier y F. Bérard, "Visual tracking of bare fingers for interactive surfaces", en Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, 24-27 de Octubre de 2004, Santa Fe NM (USA), pp. 119–122.

[55] K. Oka, Y. Sato y H. Koike, "Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems", en 5th IEEE International Conference on Automatic Face and Gesture Recognition, 20-21 de Mayo de 2002, Washington DC (USA), pp.411-416.

[56] M. Kolsch y M. Turk, "Robust hand detection", en Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition, 17-19 de Mayo de 2004, Seúl (Corea del Sur), pp.614-619.

[57] HandVu: Hand Gesture Recognition, http://www.movesinstitute.org/~kolsch/handvu/HandVu.html, en Agosto 2007.

[58] E.J. Ong y R. Bowden, "A boosted classifier tree for hand shape detection" , en Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition, 17-19 de Mayo de 2004, Seúl (Corea del Sur), pp. 889–894.

[59] M. Caglar, N. Lobo, "Open Hand Detection in a Cluttered Single Image using Finger Primitives", en Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop, 17-22 de Junio de 2006, New York NY (USA), pp. 148-153.

[60] W. Freeman y M. Roth, "Orientation histograms for hand gesture recognition", en Proceedings of the IEEE International Workshop on Automatic Face and Gesture Recognition, 26-28 de Junio de 1995, Zurich (Suiza), pp. 296–301.

[61] S. Wysoski, M. Lamar, S. Kuroyanagi y A. Iwata, "A rotation invariant approach on static-gesture recognition using boundary histograms and neural networks", en Proceedings of the 9th International Conference on Neural Information Processing, 18-22 de Noviembre de 2002, Singapur (Singapur), Vol. 4, pp. 2137-2141.

[62] P. Harding, y T. Ellis, "Recognizing Hand Gesture using Fourier Descriptors", en Proceedings of the 17th International Conference on Pattern Recognition, 23-26 de Agosto de 2004, Cambridge (Inglaterra), Vol. 3, pp. 286-289.

[63] Q. Chen, N. Georganas y E. Petriu, "Real-time vision-based hand gesture recognition using Haar-like features", en Proceedings of the IEEE Instrumentation and Measurement Technology Conference, 1-3 de Mayo de 2007, Varsovia (Polonia), pp. 1-6.

[64] A. Barczak, F. Dadgostar y C. Messom, "Real-time Hand Tracking Based on Non-invariant Features", en Proceedings of the IEEE Instrumentation and Measurement Technology Conference, 16-19 de Mayo de 2005, Ottawa (Canadá), Vol. 3, pp. 2192-2197 .

[65] J. Yamato, J. Ohya y K. Ishii, "Recognizing human action in timesequential images using hidden Markov model", en Proceedings of the IEEE Computer Vision and Pattern Recognition Conference, 15-18 de Junio de 1992, Champaign IL (USA), pp. 379–385.

[66] J. Weaver, T. Starner y A. Pentland, "Real-time American Sign Language recognition using desk and wearable computer based video", en IEEE Transactions on Pattern Analysis Machine Intelligence, Diciembre 1998, Vol. 20, No. 12, pp. 1371–1378.

[67] HumanScan : BioID : Downloads : BioID FACE Database, http://www.bioid.com/downloads/facedb/index.php, en Agosto 2007.

[68] Color Tracking Home Page, http://www.cs.bu.edu/groups/ivc/ColorTracking/, en Agosto 2007

[69] L. Sigal, S. Sclaroff y V. Athitsos, "Estimation and prediction of evolving color distributions for skin segmentation under varying illumination", en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 13-15 de Junio de 2000, Hilton Head Island SC (USA), Vol. 2, pp. 152-159.

[70] Sebastien Marcel – Gesture Database Web Page, http://www.idiap.ch/resources/gestures/, en Agosto 2007.

[71] I. H. Witten y E. Frank, "Data Mining: Practical machine learning tools and techniques", 2005, Segunda Edición, Morgan Kaufmann, San Francisco.

[72] P. González, "Seguimiento de manos y reconocimiento de gestos mediante métodos estadísticos", Noviembre 2005, Memoria para optar al Título de Ingeniero Civil Electricista, Universidad de Chile.

[73] Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/, en Septiembre 2007.

Anexo A

Paper Realizado

Real-time Hand Gesture Detection and Recognition using Boosted Classifiers and Active Learning

Hardy Francke, Javier Ruiz-del-Solar and Rodrigo Verschae

Department of Electrical Engineering, Universidad de Chile
{hfrancke,jruizd,rverscha}@ing.uchile.cl

Abstract. In this article a robust and real-time hand gesture detection and recognition system for dynamic environments is proposed. The system is based on the use of boosted classifiers for the detection of hands and the recognition of gestures, together with the use of skin segmentation and hand tracking procedures. The main novelty of the proposed approach is the use of innovative training techniques - active learning and bootstrap -, which allow obtaining a much better performance than similar boosting-based systems, in terms of detection rate, number of false positives and processing time. In addition, the robustness of the system is increased due to the use of an adaptive skin model, a color-based hand tracking, and a multi-gesture classification tree. The system performance is validated in real video sequences.

Keywords: Hand gesture recognition, hand detection, skin segmentation, hand tracking, active learning, bootstrap, Adaboost, nested cascade classifiers.

1 Introduction

Hand gestures are extensively employed in human non-verbal communication. They allow to express orders (e.g. “stop”, “come”, “don’t do that”), mood state (e.g. “victory” gesture), or to transmit some basic cardinal information (e.g. “one”, “two”). In addition, in some special situations they can be the only way of communicating, as in the cases of deaf people (sign language) and police’s traffic coordination in the absence of traffic lights. An overview about gesture recognition can be found in [18].

Thus, it seems convenient that human-robot interfaces incorporate hand gesture recognition capabilities. For instance, we would like to have the possibility of transmitting simple orders to personal robots using hand gestures. The recognition of hand gestures requires both hand’s detection and gesture’s recognition. Both tasks are very challenging, mainly due to the variability of the possible hand gestures (signs), and because hands are complex, deformable objects (a hand has more than 25 degrees of freedom, considering fingers,

wrist and elbow joints) that are very difficult to detect in dynamic environments with cluttered backgrounds and variable illumination.

Several hand detection and hand gesture recognition systems have been proposed. Early systems usually require markers or colored gloves to make the recognition easier. Second generation methods use low-level features as color (skin detection) [4][5], shape [8] or depth information [2] for detecting the hands. However, those systems are not robust enough for dealing with dynamic environments; they usually require uniform background, uniform illumination, a single person in the camera view [2], and/or a single, large and centered hand in the camera view [5]. Boosted classifiers allow the robust and fast detection of hands [3][6][7]. In addition, the same kind of classifiers can be employed for detecting static gestures [7] (dynamic gestures are normally analyzed using Hidden Markov Models [4]). 3D hand model-based approaches allow the accurate modeling of hand movement and shapes, but they are time-consuming and computationally expensive [6][7].

In this context, we are proposing a robust and real-time hand gesture detection and recognition system, for interacting with personal robots. We are especially interested in dynamic environments such as the ones defined in the *RoboCup @Home league* [21] (our *UChile HomeBreakers* team participates in this league [22]), with the following characteristics: variable illumination, cluttered backgrounds, real-time operation, large variability of hands' pose and scale, and limited number of gestures (they are used for giving the robot some basic information). In this first version of the system we have restricted ourselves to static gestures.

The system we have developed is based on the use of boosted classifiers for the detection of hands and the recognition of gestures, together with the use of skin segmentation and hand tracking procedures. The main novelty of the proposed approach is the use of innovative training techniques - active learning and bootstrap -, which allow obtaining a much better performance than similar boosting-based systems, in terms of detection rate, number of false positives and processing time. In addition, the robustness of the system is increased thanks to the use of an adaptive skin model, a color-based hand tracking, and a multi-gesture classification tree.

This paper is organized as follows. In section 2 some related work in hand gesture recognition and active learning is presented. In section 3 the proposed hand gesture detection and recognition system is described. In sections 4 and 5 the employed learning framework and training procedures are described. Results of the application of this system in real video sequences are presented and analyzed in section 6. Finally, some conclusions of this work are given in section 7.

2 Related Work

Boosted classifiers have been used for both hand detection and hand gesture detection. In [3] a hand detection system that can detect six different gestures is proposed. The system is based on the use of Viola&Jones' cascade of boosted classifiers [16]. The paper's main contributions are the addition of new rectangular features for the hand detection case, and the analysis of the gesture's separability using frequency spectrum analysis. The classifiers are trained and tested using still images (2,300 in total), which contains centered hands, with well-defined gestures. The performance of the classifiers in real videos is not analyzed. In [6] an extension of [3] is proposed, in which boosted classifiers are employed for hand detection, while gestures are recognized using scale-space derived features. The reported experiments were carried out in a dynamic environment, but using single, large and centered hands in the camera view. In [7] a real-time hand gesture recognition system is proposed, which is also based on the standard Viola&Jones system. New rectangular features for the hand detection case are added. The recognition of gestures is obtained by using several single gesture detectors working in parallel. The final system was validated in a very controlled environment (white wall as background); therefore, its performance in dynamic environment is uncertain. In [9] a system for hand and gesture detection based on a boosted classifier tree is proposed. The system obtains very high detection results, however, the system is very time consuming (a tree classifier is much slower than a single cascade), and not applicable for interactive applications. Our main contribution over previous work are the use of a much powerful learning machine (nested cascade with boosted domain-partitioning classifiers), and the use of better training procedures, which increase the performance of the classifiers.

The performance of a statistical classifier depends strongly on how representative the training sets are. The common approach employed for constructing a training set for a learning machine is to use human labeling of training examples, which is a very time-consuming task. Very often, the amount of human power for the labeling process limits the performance of the final classifier. However, the construction of training sets can be carried out semi-automatically using active learning and the bootstrap procedure. This allows building larger training sets,

and therefore to obtain better classifiers. Thus, the bootstrap procedure can be employed in the selection of negative samples [17]. The procedure requires that the human expert selects a large amount of images that do not contain object instances. During training, the bootstrap procedure automatically selects image areas (windows) that will be used as negative examples. In [11] the bootstrap procedure is extended for the particular case of the training of cascade classifiers. On the other hand, active learning is a procedure in which the system being built is used to lead the selection of the training examples. For instance, in [14] an interactive labeling system is used to select examples to be added to the training set. Initially, this system takes a rough classifier and later, interactively adds both, positive and negative examples. In the here-proposed approach both, bootstrap and active learning, are employed.

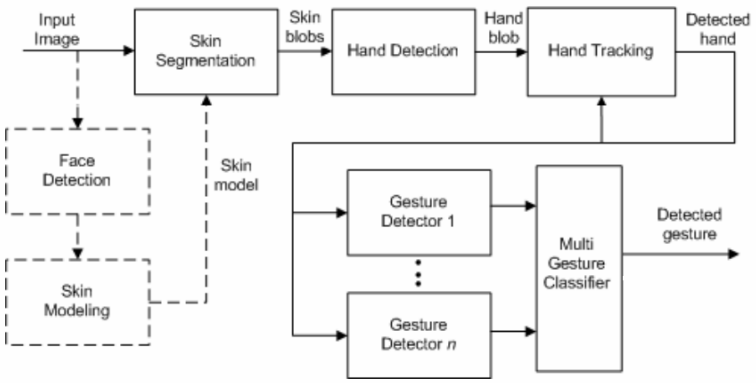


Figure 1: Proposed hand gesture detection and recognition system.

3 Real-time Hand Gesture Detection and Recognition System

3.1 System Overview

The main modules of the proposed hand gesture detection and recognition system are shown in figure 1. The *Skin Segmentation* module allows obtaining skin blobs from the input image. The use of a very reliable face detector (*Face Detection* module) allows the online modeling of the skin, which makes possible to have an adaptive segmentation of the skin pixels. The *Hand Detection* and *Hand Tracking* modules deliver reliable hand detections to the gesture detectors. Hand detection is implemented using a boosted classifier, while hand tracking is implemented using the *mean shift* algorithm [1]. Afterwards, several specific *Gesture Detectors* are applied in parallel over the image's regions that contain the detected hands. These detectors are implemented using boosted classifiers [12]. Finally, a *Multi-Gesture Classifier* summarizes the detections of the single detectors. This multi-class classifier is implemented using a *J48 pruned tree (Weka's* [19] *version of the* C4.5 classifier*)*. In the next subsections these modules are described in detail.

3.2 Adaptive Skin Segmentation

Adaptive skin segmentation is implemented using a procedure similar to the one described in [10]. The central idea is to use the skin color distribution in a perceived face to build a specific skin model. In other words, the skin model uses the context information from the person, given by its face, and the current illumination. With this we manage to have a robust skin detector, which can deal with variations in illumination or with differences in the specific skin's colors, in comparison to offline trained skin detectors. This approach requires having a reliable face detector. We employed a face detector that uses nested cascades of classifiers, trained with the Adaboost boosting algorithm, and domain-partitioning based classifiers. This detector is detailed described in [11].

With the aim of making the model invariant to the illumination level to a large degree, the skin modeling is implemented using the RGB normalized color space:

$$I = R + G + B \;;\; r = \frac{R}{I};\; g = \frac{G}{I} \qquad (1)$$

After a new face is detected, a subset of the face pixels is selected for building the skin model (see figure 2). After pixels' selection and normalization, the *r*, *g* and *I* skin variables are modeled with Gaussian functions. The

skin model parameters correspond to the variables' mean value and standard deviation: μ_r, σ_r, μ_g, σ_g, μ_I and σ_I. In order to lighten the computational burden, this modeling is carried out only once for every detected face (the first time that the face is detected). As long as there is not any major change in the illumination, there is no need to update the model. Having the skin model, the classification of the pixels is carried out as follows:

$$f(i,j) = \begin{cases} skin & if \quad |c - \mu_c| < \alpha_c \cdot \sigma_c, \quad c = r, g, I \\ non-skin & if \quad \sim \end{cases} \tag{2}$$

where *i* and *j* represent the coordinates of pixel being analyzed, and α_r, α_g and α_I are constants of adjustment of the classifier. For simplicity all these constants are made equal. In practice we have observed that this value needs to be adjusted depending on the brightness of the input image, increasing it when the brightness decreases, and vice versa.

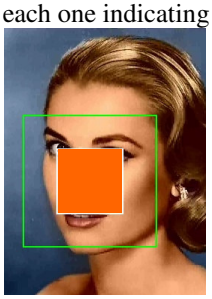
After the skin pixels are detected, they are grouped together in skin blobs, according to their connectivity. In order to diminish the false positives from the skin detection, blobs that have an area below a certain threshold are discarded. Finally, all skin blobs are given to the next stage of the process except the ones containing faces.

3.3. Hand Detection and Tracking

In order to detect hands within the skin blobs, a hand detector is implemented using a cascade of boosted classifiers. Although this kind of classifiers allows obtaining very robust object detectors in the case of face or car objects, we could not build a reliable generic hand detector. This is mainly because: (i) hands are complex, highly deformable objects, (ii) hand possible poses (gestures) have a large variability, and (iii) our target is a fully dynamic environment with cluttered background. Therefore we decided to switch the problem to be solved, and to define that the first time that the hand should be detected, a specific gesture must be made, the fist gesture. Afterwards, that is, in the consecutive frames, the hand is not detected anymore but tracked. The learning framework employed for training the fist detector is described in section 4 and the specific structure of the detector in section 6.

The hand-tracking module is built using the mean shift algorithm [1]. The seeds of the tracking process are the detected hands (fist gestures). We use RGB color histograms as feature vectors (model) for mean shift, with each channel quantized to 32 levels (5 bits). The feature vector is weighted using an Epanechnikov kernel [1].

As already mentioned, once the tracking module is correctly following a hand, there is no need to continue applying the hand detector, i.e. the fist gesture detector, over the skin blobs. That means that the hand detector module is not longer used until the hand gets out of the input image, or until the mean shift algorithm loses track of the hand, case where the hand detector starts working again. At the end of this stage, one or several regions of interest (ROI) are obtained, each one indicating the location of a hand in the image.



$$x_{0,orange} = x_{0,green} + 0.25 \cdot width_{green}$$
$$y_{0,orange} = y_{0,green} + 0.25 \cdot height_{green}$$
$$width_{orange} = 0.5 \cdot width_{green}$$
$$height_{orange} = 0.5 \cdot height_{green}$$

Figure 2: Left: The green (outer) square corresponds to the detected face. The orange (inner) square determines the pixels employed for building the skin model. Right: The orange square cropping formula.

3.4. Hand Gesture Recognition

In order to determine which gesture is being expressed, a set of single gesture detectors are applied in parallel over the ROIs delivered as output of the tracking module. Each single gesture detector is implemented using a cascade of boosted classifiers. The learning framework employed for building and training these classifiers is described in section 4. Currently we have implemented detectors for the following gestures: *first*, *palm*, *pointing*, and *five* (see Figure 3). The specific structure of each detector is given in section 6.

Due to noise or gesture ambiguity, it could happen than more than one gesture detector will give positive results in a ROI (more than one gesture is detected). For discriminating among these gestures, a multi-gesture

classifier is applied. The used multi-class classifier is a *J48 pruned tree (Weka's* [19] *version of C4.5)*, built using the following four attributes that each single gesture detector delivers:

- *conf*: sum of the cascade confidence's values of windows where the gesture was detected (a gesture is detected at different scales and positions),
- *numWindows*: number of windows where the gesture was detected,
- *meanConf*: mean confidence value given by *conf/numWindows*, and
- *normConf*: normalized mean confidence value given by *meanConf/maxConf*, with *maxConf* the maximum possible confidence that a window could get.

Fist Palm Pointing Five

Figure 3: Hand gestures detected by the system.

4 Learning Framework

The learning framework used to train the hand detector and single gesture detectors is presented in the next subsections. An extensive description of this framework can be found in [11].

4.1. Learning using cascade of boosted classifiers

The key concepts used in this framework are nested cascades, boosting, and domain-partitioning classifiers. Cascade classifiers [16] consist of several layers (stages) of increasing complexity. Each layer can reject or let pass the inputs to the next layer, and in this way a fast processing speed together with high accuracy are obtained. Nested cascades [13] allow high classification accuracy and higher processing speed by reusing in each layer the confidence given by its predecessor. Adaboost [12] is employed to find highly accurate hypotheses (classification rules) by combining several weak hypotheses (classifiers). A nested cascade of boosted classifiers is composed by several integrated (nested) layers, each one containing a boosted classifier. The cascade works as a single classifier that integrates the classifiers of every layer. Weak classifiers are linearly combined, obtaining a strong classifier. A nested cascade, composed of *M* layers, is defined as the union of M boosted classifiers H_C^k each one defined by:

$$H_C^k(x) = H_C^{k-1}(x) + \sum_{t=1}^{T_k} h_t^k(x) - b_k \qquad (3)$$

with $H_C^0(x) = 0$, h_t^k the weak classifiers, T_k the number of weak classifiers in layer *k*, and b_k a threshold (bias) value that defines the operation point of the strong classifier. At a layer *k,* processing an input *x*, the class assigned to *x* corresponds to the sign of $H_C^k(x)$. The output of H_C^k is a real value that corresponds to the confidence of the classifier and its computation makes use of the already evaluated confidence value of the previous layer of the cascade.

4.2 Design of the strong and weak classifiers

The weak classifiers are applied over features computed in every pattern to be processed. To each weak classifier a single feature is associated. Following [12], domain-partitioning weak hypotheses make their predictions based on a partitioning of the input domain *X* into disjoint blocks $X_1,\ldots,X_n$, which cover all *X*, and for which $h(x)=h(x')$ for all $x, x' \in X_j$. Thus, a weak classifier´s prediction depends only on which block, X_j, a given sample instance falls into. In our case the weak classifiers are applied over features, therefore each feature domain *F* is partitioned into disjoint blocks $F_1,\ldots,F_n$, and a weak classifier *h* will have an output for each partition block of its associated feature *f*:

$$h(f(x)) = c_j \text{ s.t } f(x) \in F_j \quad (4)$$

For each classifier, the value associated to each partition block (c_j), i.e. its output, is calculated so that it minimizes a bound of the training error and at the same time a loss function on the margin [12]. This value depends on the number of times that the corresponding feature, computed on the training samples (x_i), falls into this partition block (histograms), on the class of these samples (y_i) and their weight $D(i)$. For minimizing the training error and the loss function, c_j is set to [12]:

$$c_j = \frac{1}{2}\ln\left(\frac{W_{+1}^{j} + \varepsilon}{W_{-1}^{j} + \varepsilon}\right), \quad W_l^{j} = \sum_{i:f(x_i)\in F_j \wedge y_i = l} D(i) = \Pr\left[f(x_i) \in F_j \wedge y_i = l\right], \text{ where } l = \pm 1 \quad (5)$$

where ε is a regularization parameter. The outputs, c_j, of each of the weak classifiers, obtained during training, are stored in a LUT to speed up its evaluation. The real Adaboost learning algorithm is employed to select the features and training the weak classifiers $h_t^k(x)$. For details on the cascade's training algorithm see [11].

4.3 Features

Two different kinds of features are used to build the weak classifiers, rectangular features (a kind of Haar-like wavelet) and mLBP (modified Local Binary Pattern). In both cases the feature space is partitioned so that it can be used directly with the domain-partitioning classifier previously described. Rectangular features can be evaluated very quickly, independently of their size and position, using the integral image [16], while mLBP corresponds to a contrast invariant descriptor of the local structure of a given image neighborhood (see [15]).

5 Training procedures

The standard procedure to build *training sets* of objects and non-objects for training a statistical classifier requires that an expert (a human operator) obtains and annotates training examples. This procedure is usually very time-consuming; more importantly, it is very difficult to obtain representative examples. In the following, two procedures for solving these problems are presented.

5.1 Bootstrap Procedure

Every window of any size in any image that does not contain an object (e.g. a hand) is a valid non-object training example. Obviously, to include all possible non-object patterns in the training database is not an alternative. To define such a boundary, non-object patterns that look similar to the object should be selected. This is commonly solved using the bootstrap procedure [17], which corresponds to iteratively train the classifier, each time increasing the negative training set by adding examples of the negative class that were incorrectly classified by the already trained classifier. When training a cascade classifier, the bootstrap procedure can be applied in two different situations: before starting the training of a new layer (external bootstrap) and for re-training a layer that was just trained (internal bootstrap). It is important to use bootstrap in both situations [11]. The external bootstrap is applied just one time for each layer, before starting its training, while the internal bootstrap can be applied several times during the training of the layer. For details on the use of bootstrapping in the training of a cascade see [11].

5.2 Active Learning

As mentioned, the selection of representative positive training examples is costly and very time consuming, because a human operator needs to be involved. However, these training examples can be semi-automatically generated using active learning. Active learning is a procedure in which the system being built is used to lead the selection of the training examples.

In the present work we use active learning to assist the construction of representative positive training sets, i.e. training sets that capture the exact conditions of the final application. To generate training examples of a specific hand gesture detector, the procedure consists of asking a user to make this specific hand gesture for a given time. During this time the user hand is automatically tracked, and the bounding boxes (ROI) are

automatically incorporated to the positive training sets of this gesture. If the hand is tracked for a couple of minutes, and the user maintains the hand gesture while moving the hand, thousands of examples can be obtained with the desired variability (illumination, background, rotation, scale, occlusion, etc.). Thus, all windows classified as positive by the hand tracker are taken as positive training examples. This procedure can be repeated for several users. A human operator only has to verify that these windows were correctly detected, and to correct the alignment of the windows, when necessary. Later, all these windows are downscaled to the window size (24x24 or 24x42 pixels in our case) to be used during training.

In a second stage, active learning can also be employed for improving an already trained specific gesture detector. In this last case, the same procedure is employed (the user makes the hand gesture and the hand is tracked), but the already trained gesture detector is in charge of generating the training examples. Thus, every time that the gesture detector classifies a hand bounding box coming from the hand tracker as a non-object (the gesture is not detected), this bounding box is incorporated in the positive training set for this gesture.

6 Evaluation

In the present section an evaluation and analysis of the proposed system is presented. In this evaluation the performance of the system, as well as, its modules are analyzed. We also analyze the effect over the detector's performance of using Active learning during training. The detection results are presented in terms of Detection Rate (DR) versus Number of False Positives (FP), in the form of ROC curves. An analysis of the processing speed of the system is also presented.

The cascade classifiers were trained using three kinds of hand databases: (i) the IDIAP hand database [20], (ii) images obtained from the Internet, and (iii) images obtained using active learning and our hand gesture detection and recognition system. Table 1 and Table 2 summarize information about these training sets and the obtained cascade classifiers. For the other gesture's databases, the amount of data used to train the classifiers is similar.

On Table 1 and Table 2 we can also observe information about the structure of the obtained classifiers (number of layers and total number of weak classifiers). This information gives us an idea of the complexity of the detection problem, where large values indicate higher complexity and also larger processing times. These numbers are a result of the training procedure of the cascade [11] (they are not set a priori). As mentioned, we have selected a J48 pruned tree as multi-gesture's classifier. This classifier was trained using the training sets described in Table 3, using the Weka package, and 10-fold cross-validation. The obtained tree structure has 72 leaves and 143 tree nodes. In the validation dataset we obtained 90.8% of correct classifications.

To evaluate each single detector, a dataset consisting of 200 examples per class was used. This database contains images presenting a large degree of variability in the shape and size of the hands, the illumination conditions, and in the background. As a reference, this database contains more variability than the IDIAP database [20], and therefore is more difficult. The complete system was evaluated using a database that consists of 8,150 frames coming from 5 video sequences, where 4 different persons performed the 4 considered gestures. The sequences were captured by the same camera used to perform the active learning, and emphasis was given to produce a cluttered background and varying illumination conditions.

To analyze how active learning improves the performance of the boosted classifiers, we studied two cases, a *fist* detector, and a *palm* detector. For each case we trained two classifiers, the first one using active learning and the second one without using it. The training of these detectors was done using the datasets presented in Table 1. The effect of using active learning in the performance of the detector is shown in Figure 4. To better show the effect of using active learning, the evaluation was performed by applying the detectors directly over the skin blobs in the input images that do not correspond to the face, i.e., not over the results of the hand-tracking module. As it can be noticed, the use of active learning during training largely improves the performance of the detectors, with up to a 90 % increase for operation points with low false positive rates. When using the tracking system, the number of false positives is reduced even more, so the complete system has much lower false positive rates than the ones observed here. Even though in the case of using active learning the obtained classifiers have a larger number of weak classifiers, the processing time is not much larger, because there is not a large increase on the number of weak classifier for the first layers of the cascade. As a consequence of this result, we choose to train all our gesture detectors using active learning.

An evaluation of the gesture detectors, trained using active learning, is shown in figure 5. In this case the results were obtained by applying the detectors directly over the detected skin blobs not corresponding to the face, not over the results of the hand-tracking module. The use of the hand tracking before applying the detectors reduces largely the number of false positives. The training was done using the datasets described in Table 2, and

as in the previous experiment, the evaluation was done using a dataset consisting of 200 examples per class, which contains all gestures and a large degree of variability. As it can be observed, the *fist* gesture detector obtains a very high performance, achieving a detection rate of 99%, with just 2 false positives. The other detectors show a lower performance, having a higher number of false positives, which is reduced when the tracking module is used. The main reason for the large number of false positives is the large variability of the illumination conditions and background of the place where the detectors were tested. Figure 6 show some images from the test dataset, where it can be observed that it is an environment with several different light sources, and a lot of reflections, shadows, and highlights.

An evaluation of the complete system, that means the hand tracking and detection module, together with the gesture´s detection module and the gesture recognition´s module, is summarized in Table 4. The results are presented by means of a confusion matrix. The first thing that should be mention here is that the hand detection together with the tracking system did not produce any false positive out of the 8150 analyzed frames, i.e. the hands were detected in all cases. From Table 4 it can be observed that the gesture detection and recognition modules worked best on the *five* gesture, followed by the *pointing*, *fist* and *palm* gestures, in that order. The main problem is the confusion of the *fist* and *pointing* gestures, which is mainly due to the similarly of the gestures. In average the system correctly recognized the gestures in 70% of the cases. If the *pointing* and the *fist* gestures are considered as one gesture, the recognition rate goes up to 86%.

We also evaluated the processing time of the whole system. This evaluation was carried out in a PC powered with a Pentium 4 3.2GHz, 1GB RAM, running Windows XP and the system was implemented using the C language. The observed average processing time required for processing a 320x240 pixel's image, without considering the time required for image acquisition, was 89 milliseconds (see details in Table 5). With this, the system can run at about 11 fps for frames of 320x240 pixel size.

Table 1. Training sets for the *fist* and *palm* detectors. The D1 detectors are built using active learning, while the D2 detectors are built using standard hand databases.

Gesture	Size of training images (pixels)	Database	Training's set size	Validation's set size	Num. negative images	Num. layers	Num. weak classifiers
Fist (D1)	24x24	Active learning	1194	1186	46746	9	612
Fist (D2)	24x24	IDIAP [20]	795	606	47950	10	190
Palm (D1)	24x42	Active learning	526	497	45260	10	856
Palm (D2)	24x24	IDIAP [20]	597	441	36776	8	277

Table 2. Training sets and classifier structure for the definitive gesture's detectors.

Gesture	Size of training images (pixels)	Num. positive training images	Num. positive validation images	Num. negative (no-hand) images	Num layers	Total Num. detectors
Fist	24x24	1194	1186	46746	9	612
Palm	24x42	526	497	45260	10	856
Pointing	24x42	947	902	59364	12	339
Five	24x24	651	653	41859	9	356

Table 3: Training sets for the multi-gesture's classifier.

Gesture	Fist	Palm	Pointing	Five
Number of training examples	3838	3750	3743	3753
Number of training attributes	15352	15000	14972	15012

Table 4: Confusion matrix of the complete system.

Class\Predicted	Fist	Palm	Pointing	Five	Unknown	*Detection and recognition rates [%]*
Fist	1533	2	870	9	15	*63.1*
Palm	39	1196	10	659	15	*62.3*
Pointing	436	36	1503	27	86	*72.0*
Five	103	32	6	1446	127	*84.3*

Table 5: Average processing time of the main modules, in milliseconds

The size of frames is 320x240 pixels.

Skin detection	Face detection	Face tracking	Hand detection	Gesture recognition + Hand tracking
4.456	0.861	1.621	2.687	78.967

7 Conclusions

One of the ways humans communicate with each other is through gestures, in particular hand gestures. In this context, a framework for the detection of hands and the recognition of hand gestures was proposed, with the aim of using it to interact with a service robot. The framework is based on cascade classifiers, a J48 tree classifier, an adaptive skin detector and a tracking system. The main module of the system corresponds to a nested cascade of boosted classifiers, which is designed to carry out fast detections with high DR and very low FPR. The system makes use of a face detector to initialize an adaptive skin detector. Then, a cascade classifier is used to initialize the tracking system by detecting the fist gesture. Afterwards, the hands are tracked using the mean shift algorithm. Afterwards, several independent detectors are applied within the tracked regions in order to detect individual gestures. The final recognition is done by a J48 classifier that allows to distinguishing between gestures.

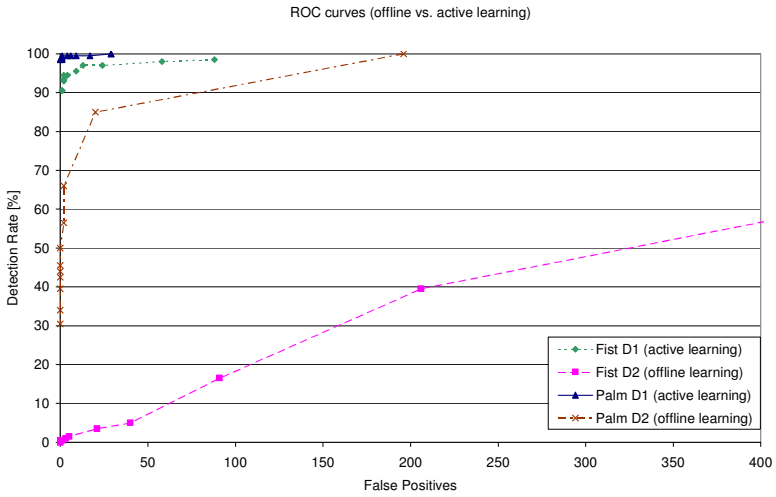


Figure 4: Fist and Palm detector ROC curves, using active learning (D1) and not using active learning (D2). In all cases the tracking system was not used.

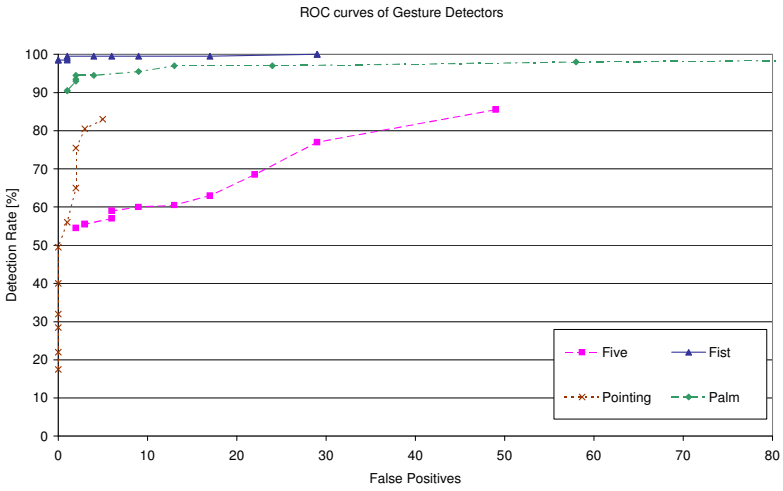


Figure 5: ROC curves of the gesture detectors (trained using active learning) applied without using the tracking system.

For training the cascade classifiers, active learning and the bootstrap procedure were used. The proposed active learning procedure allowed to largely increase the detection rates (e.g., from 17% up to 97% for the Palm gesture detector) maintaining a low false positive rate. As in our previous work [11], the bootstrap procedure [17] helped to obtain representative training sets when training a nested cascade classifier.

Out of the hand detectors, the best results were obtained for the *fist* detection (99% DR at 1 FP), probably because this gesture has the lower degree of variability. The worst results were obtained for the gesture *five* detector (85% DR at 50 FP), mainly because under this gesture the hand and the background are interlaced, which greatly difficult the detection process in cluttered backgrounds. In any case, it should be stressed that these results correspond to a worst case scenario, i.e. when no tracking is performed, and that when using the tracking the FPR is greatly reduced.

The system performs with a reasonable high performance in difficult environments (cluttered background, variable illumination, etc.). The tracking module has a detection rate over 99%, the detection module a 97% detection rate, and the gesture recognition rate is 70%. The main problem is the confusion of the *fist* with the *pointing* gesture and vice-versa. When these two gestures are considered as one, the global recognition rate goes up to 86%. We think that the recognition could be improved by using the history of the detection. The system presents a high processing speed (about 11 fps), and therefore it can be applied in dynamical environments in real time.

As future research we would like to extend our system for recognizing dynamic gestures and to improve the detection module by integrating the classifiers' cascades.

Acknowledgements

This research was funded by Millenium Nucleus Center for Web Research, Grant P04-067-F, Chile.

References

1. D. Comaniciu, V. Ramesh, and P. Meer, Kernel-Based Object Tracking, *IEEE Trans. on Pattern Anal. Machine Intell.*, vol 25, no. 5, (2003) pp. 564 – 575.
2. X. Liu and K. Hand gesture recognition using depth data, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 529 – 534, Seoul, Korea.
3. M. Kolsch, M.Turk, Robust hand detection, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 614 – 619, Seoul, Korea.
4. N. Dang Binh, E. Shuichi, T. Ejima, Real-Time Hand Tracking and Gesture Recognition System, *Proc. GVIP 05*, (2005) pp. 19-21 Cairo, Egypt.

5. C. Manresa, J. Varona, R. Mas, F. Perales, Hand Tracking and Gesture Recognition for Human-Computer Interaction, *Electronic letters on computer vision and image analysis*, Vol. 5, Nº. 3, (2005) pp. 96-104.
6. Y. Fang, K. Wang, J. Cheng, H. Lu, A Real-Time Hand Gesture Recognition Method, *Proc. 2007 IEEE Int. Conf. on Multimedia and Expo*, (2007) pp. 995-998
7. Q. Chen, N.D. Georganas, E.M. Petriu, Real-time Vision-based Hand Gesture Recognition Using Haar-like Features, *Proc. Instrumentation and Measurement Technology Conf. – IMTC 2007*, (2007)Warsaw, Poland
8. A. Angelopoulou, J. García-Rodriguez, A. Psarrou, Learning 2D Hand Shapes using the Topology Preserving model GNG, *Lecture Notes in Computer Science* 3951 (*Proc. ECCV 2006*), pp. 313-324
9. E.-J. Ong and R. Bowden, A boosted classifier tree for hand shape detection, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 889 – 894, Seoul, Korea.
10. M. Wimmer, B. Radig, Adaptive Skin Color Classificator, Int. Journal on Graphics, Vision and Image Processing, Special Issue on Biometrics, vol 2 (2006) pp. 39-42.
11. R. Verschae, J. Ruiz-del-Solar, M. Correa, A Unified Learning Framework for object Detection and Classification using Nested Cascades of Boosted Classifiers, *Machine Vision and Applications* (in press).
12. R.E. Schapire, Y. Singer, Improved Boosting Algorithms using Confidence-rated Predictions, *Machine Learning*, 37(3): (1999) pp. 297-336.
13. B. Wu, H. Ai, C. Huang, S. Lao, Fast rotation invariant multi-view face detection based on real Adaboost, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 79 – 84, Seoul, Korea.
14. Y. Abramson, Y. Freund, Active learning for visual object detection, *UCSD Technical Report* CS2006-0871, Nov. 19, 2006.
15. B. Fröba, A. Ernst, Face detection with the modified census transform, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 91 – 96, Seoul, Korea.
16. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (2001) pp. 511 – 518.
17. K. Sung, T. Poggio, Example-Based Learning for Viewed-Based Human Face Deteccion, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.20, No. 1, (1998) pp. 39-51.
18. The Gesture Recognition Home Page. Available on August 2007 in: http://www.cybernet.com/~ccohen/
19. Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
20. IDIAP hand gesture database. Available in Aug. 2007 in http://www.idiap.ch/resources/gestures/
21. RoboCup @Home Official website. Available in Aug. 2007 in http://www.robocupathome.org/
22. UChile RoboCup Teams official website. Available in Aug. 2007 in http://www.robocup.cl/

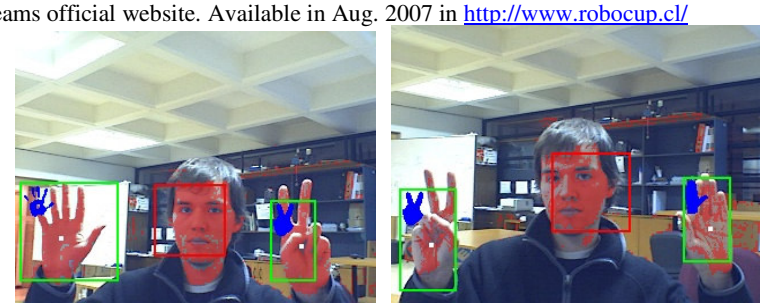


Figure 6: Example results of the system. The five, victory, and the victory gestures are detected and recognized. Notice the cluttered background, the highlights, and skin-like colors.