UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"Seguimiento de la cabeza por medio de un Blob"

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

Fernando Said Ramírez García

DIRECTOR DE TESIS:

M.I.A. Hilda Caballero Barbosa,

HUAJUAPAN DE LEÓN, OAX. SEPTIEMBRE DE 2007

UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"Seguimiento de la cabeza por medio de un Blob"

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

Fernando Said Ramírez García

Jurado

M.C. Zarza López Luis Anselmo

Presidente

M.I.A. Hilda Caballero Barbosa

Vocal

M.C. Rodríguez López Verónica

Secretario

HUAJUAPAN DE LEÓN, OAX. SEPTIEMBRE DE 2007

A mis padres, por dejarme la mejor
herencia en esta vida, mi educación.

Agradecimientos

A mis padres, por estar siempre a mi lado y brindarme el apoyo incondicional para realizar el trabajo de la presente tesis.

A mis hermanos, Ediel, por todo el apoyo y por su admirable pasión en todo lo que realiza. Marvin, por la compresión que ha tenido.

Agradezco de manera muy especial a mi asesora M.C. Hilda Caballero Barbosa por el apoyo, compresión y sobre todo la paciencia durante el desarrollo del presente trabajo de tesis.

A mis sinodales, M.C. Verónica Rodríguez López, M.C. Luis Anselmo Zarza López y Dr. Virginia Berrón Lara, por su tiempo y sus recomendaciones.

A la M.C. Mercedes Martínez González, Director del Instituto de Diseño, por facilitar el uso la cámara de video y las herramientas del Laboratorio de Multimedia.

Al M.C. Mario Moreno Rocha, Director del UsaLab, por permitir el uso de la cámara de video.

Al Dr. Christian Sturm, por permitir el uso de su cámara de video de su propiedad para la realización de las grabaciones ocupadas en la presente tesis.

Al M.C. Joaquín Peña Acevedo, por permitir utilizar la librería implementada por él, para el desarrollo de la aplicación de la presente tesis.

Al P.I.D. Dahir García Merlin, por su tiempo y ayuda en el manejo de las herramientas en el Laboratorio de Multimedia del Instituto de Diseño.

A las personas que participaron en las grabaciones que se utilizaron en la construcción de la presente tesis:

Profesores

L.I. Mónica Edith García García

M.C. Alejandro López López

Ing. Hugo Enrique Martínez Cortés

M.C. Mario Alberto Moreno Rocha

Dra. María del Pilar Pozos Parra

M.C. Felipe Santiago Espinosa

Dr. Christian Bernhard Sturm

Alumnos

Aquino Acevedo Zahedi Armando

Alvarado Legario Selene

Chávez García Ricardo Omar

Cuevas García Enrique Omar

Figueroa Valencia Amarildo

Gómez Jímenez José

García Alvarez Carlos Alberto

González Arévalo Elfrich

Gustavo Adolfo Ramírez Silva

Jarquín González Diana Carolina

Jerónimo Ruíz Belén Rocío

Jiménez Duran Analleli

López Méndez Diego Edson

López Urbina Maritza Elizabeth

Méndez Gaitán Lair Lino

Nuñez Júarez Aristóteles Federico

Nuñez Martínez Lizet

Pacheco López Olivia

Pérez Vásquez Oscar Uriel

Ponce Ortíz Rosario

Ramírez de la Rosa Adriana Gabriela

Ramírez Hernández Arturo

Ramírez Jiménez Lucy

Ramos Ruiz Adrian

Regalado Díaz Dennys

Rodríguez Sánchez Juan Carlos

Santa María Caro Edilberto

Vásquez Limeta Ramón Eugenio

Vásquez Martínez Eric

Villanueva Soriano Mario Alberto

Índice general

Índice de figuras

Capítulo 1

Introducción

Este trabajo de tesis está orientado a resolver uno de los problemas típicos en el área de Visión por Computadora (VC), el cual se refiere al seguimiento de objetos.

El propósito general de la VC y del Procesamiento Digital de Imágenes (PDI) es obtener información relevante de una escena a partir de imágenes digitales, para realizar la toma adecuada de decisiones. La VC y el PDI son disciplinas en creciente auge con una gran variedad de aplicaciones, tales como inspección automática, reconocimiento de objetos, mediciones sin contacto, entre otras. El futuro es aún más prometedor, con la creación de máquinas autónomas capaces de interactuar inteligentemente con el entorno, las cuales necesariamente deben estar dotadas de mecanismos que les permitan percibir éste.

Un sistema de VC captura imágenes mediante una cámara, las analiza, y produce descripciones de lo que fue detectado en ellas.

La mayoría de los sistemas de visión biológicos han evolucionado para contender con el mundo cambiante. Los sistemas de VC han evolucionado de la misma manera. Para un sistema de VC, la habilidad para enfrentarse con el movimiento y objetos cambiantes, cambios de iluminación, y cambios en la perspectiva, es esencial para desarrollar varias tareas. En muchas aplicaciones, una entidad, una característica o un objeto, debe ser

seguido a través de una secuencia de imágenes. Si hay sólo un objeto de interés en la secuencia, el problema es fácil de resolver. En la presencia de muchos objetos de interés moviéndose independientemente en una escena, el seguimiento requiere el uso de restricciones basadas en la naturaleza de los objetos y de su movimiento.

La entrada para un sistema de análisis de escenas dinámicas, es una secuencia de $frames$[1] de imágenes tomadas de un mundo cambiante. La cámara se usa para adquirir la secuencia de imágenes, la cual podría estar, además, en movimiento. Cada $frame$ representa una imagen de la escena en un instante particular de tiempo. Los cambios en una escena podrían darse debido al movimiento de la cámara, al movimiento de los objetos, a cambios en la iluminación, o cambios en la estructura, tamaño o forma de un objeto. Usualmente se asume que los cambios en una escena se deben a la cámara o al objeto en movimiento, y que los objetos pueden ser rígidos o semi-rígidos. El sistema podría detectar cambios, determinar las características del movimiento del observador y los objetos, caracterizar el movimiento usando una abstracción de alto nivel, recuperar la estructura de los objetos, y reconocer a los objetos en movimiento. Una imagen de la escena en un momento dado representa una proyección de la escena, la cual depende de la posición de la cámara. Existen cuatro posibilidades de la configuración de la cámara y el mundo dada la naturaleza dinámica [16]:

1. Cámara estacionaria, objetos estacionarios.
2. Cámara estacionaria, objetos en movimiento.
3. Cámara en movimiento, objetos estacionarios.
4. Cámara en movimiento, objetos en movimiento.

[1] *Frame*: Cada una de las imágenes dentro de una secuencia de video.

Para analizar una secuencia de imágenes, se requieren de diferentes técnicas para cada uno de los casos anteriores.

Son de particular interés para el objetivo de este trabajo, aquellos sistemas de VC que tienen la configuración: Cámara estacionaria y objetos en movimiento, donde los objetos a seguir son personas.

La identificación visual de las personas y sus movimientos es una característica importante en múltiples aplicaciones, tales como interfaces de realidad virtual [3], sistemas de vigilancia [34], percepción robótica [32, 22], reconocimiento de acciones [20, 35], bases de datos de video [17, 24], cuartos inteligentes [2, 5], creación de herramientas que retoman información del cuerpo en movimiento en tiempo real, y creativamente usan esta información para generar o manejar luz, música, gráficos o video [30], entre otras.

Lo que se persigue con lo anterior es mejorar la interacción hombre-máquina y conseguir que las computadoras funcionen como asistentes humanos; un paso importante para ello es dotar a las computadoras de inteligencia perceptual, es decir, dotarlas de habilidades que les permitan determinar qué aspectos de una situación son significativas, y elegir en consecuencia un curso de acción adecuado. La habilidad de encontrar y seguir las partes del cuerpo de una persona, por lo tanto, es un problema visual importante.

Una forma de localizar a las personas es mediante la detección y seguimiento de la cabeza. El seguimiento de la cabeza es un paso importante en una amplia variedad de interfaces de usuario interactivas dirigidas por un sistema de VC [18]. La posición y orientación de la cabeza obtenida en el seguimiento permite la determinación de la inclinación y el movimiento de la cabeza, así como el reconocimiento de gestos simples [29]. Además, mediante el seguimiento de la cabeza, se tiene acceso a la información del rostro de la persona, por lo que se pueden también emplear técnicas para el reconocimiento de rostros, análisis de expresiones faciales [6], lectura de labios [27], y

seguimiento de ojos. Como ejemplos de aplicaciones donde se emplea el seguimiento de la cabeza se tienen los siguientes:

- Comunicación no verbal por medio del seguimiento de la cabeza. Este sistema, llamado HeadDev, utiliza una interface de usuario basada en VC, que reemplaza el uso del ratón para interactuar con un sistema de comunicación no verbal para niños. El sistema permite a niños que no hablan interactuar por medio del movimiento de la cabeza y gestos faciales con otros niños, a través de un programa de comunicación que reproduce verbalmente las sentencias construidas a partir del reconocimiento de tales movimientos y gestos. Por lo que el sistema es una contribución que mejora la integración social de niños con discapacidad física y con problemas de comunicación mediante el habla [21].

- Análisis de los *blobs* de la cabeza y las manos. Este sistema presenta un método para el análisis del movimiento de la cabeza y de las manos basado en la identificación del color de piel, y que es validado con numerosos tonos de piel. El sistema utiliza el *blob* como un método útil y efectivo para investigar el comportamiento no verbal que tienen las personas cuando están mintiendo o tratando de engañar a su interlocutor. Ofrece además un método para la medición precisa del movimiento, el cual no puede ser medido fácilmente por observadores humanos, ofrece flexibilidad en el seguimiento de una variedad de colores de piel (caucásicos, latinos, africanos y asiáticos) y automatiza el análisis del comportamiento observado [20].

- Seguimiento de manos y cara para aplicaciones de realidad virtual. Este sistema realiza el seguimiento en 3D de las manos y rostro de una persona en tiempo real. Este sistema puede ser usado como una interfaz perceptual para actividades de realidad virual en un ambiente *workbench* (o mesa estereoscópica). Una mesa estereoscópica consiste de una estructura donde las imágenes se proyectan en una

pantalla horizontal en forma de mesa. Un diagrama esquemático de una mesa estereoscópica de realidad virtual se muestra en la Figura 1.1. La ventaja principal del sistema es que el humano, que se encuentra enfrente del dispositivo de realidad virtual, no necesita algún tipo de marcador o equipo especial. El sistema incluye un módulo de segmentación de color en tiempo real para detectar los píxeles de color de piel presentes en las imágenes. Los resultados de la segmentación son *blobs* de color de piel que son entradas de un módulo de asociación de datos que etiqueta los píxeles de los *blobs* utilizando un conjunto de hipótesis del estado del objeto, a partir de los $frames$ anteriores. Los resultados del seguimiento en 2D son usados para obtener las posiciones en 3D de las manos y cara y realizar así su construcción. Finalmente, con esta información se crea un avatar (humano virtual) [33].

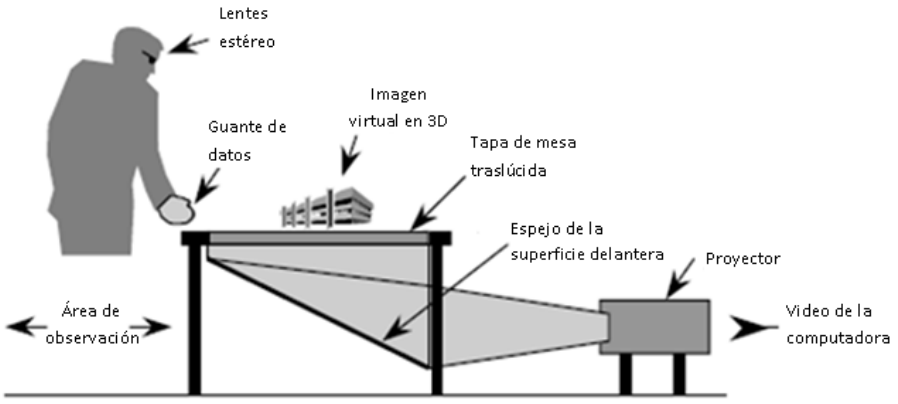


Figura 1.1: Mesa estereoscópica de realidad virtual

- Seguimiento Visual del rostro y su aplicación para codificar video basado en un modelo 3D. Este trabajo es un ejemplo de la tendencia a utilizar modelos 3D en la síntesis de video: a partir de las imágenes de la secuencia de video se obtiene un conjunto de parámetros faciales de animación (Facial Animation Parameters - FAPs -), los cuales especifican las expresiones en el rostro, es decir, describen el

movimiento en 3D y la deformación del rostro en el transcurso del tiempo. Posteriormente, en un módulo decodificador, estos parámetros se utilizan para mover y deformar un modelo texturizado de la cabeza. Para este sistema se desarrolló un método para la estimación de los parámetros FAP, basado en un modelo de deformación del volumen de segmentos de Bézier (Piecewise Bézier Volume Deformation Model -PBVD-), y varios métodos para la codificación de tales parámetros. Requiere, para cumplir con su tarea, localizar y seguir el rostro de la persona en la secuencia de video, para lo cual emplea el modelo PBVD [14].

A partir de estas aplicaciones, puede observarse la importancia de la detección y seguimiento de personas a través de su cabeza. Es por ello que en la presente tesis se abordó este problema. Se realizó la detección analizando el movimiento de la persona y analizando el color de la piel, para el seguimiento se aplicó la similitud de color y posición con respecto a la clase más próxima, las clases consideradas son el fondo de la escena y la cabeza de la persona a seguir.

1.1. Objetivo general

Diseñar e implementar un sistema que localice, segmente y siga la cabeza de una persona en tiempo real, en una escena cerrada, con condiciones de iluminación controladas y con una cámara fija.

1.2. Propuesta de solución

El objeto a seguir puede ser representado por contornos o regiones. Esta clasificación en la representación se origina de los tipos de segmentación que se distinguen en la práctica: Segmentación por contornos y segmentación de regiones o zonas. El resultado

de la segmentación del objeto proporciona la información adecuada para la construcción de su modelo.

Para realizar el seguimiento de la cabeza de la persona en tiempo real, se aplicó una representación por medio de *blobs*, es decir una representación basada en regiones. Un *blob* es definido por un conjunto de píxeles que comparten propiedades similares, tales como el color, la textura, brillantez, movimiento o una combinación de éstas, o algunas otras.

Se utilizó la naturaleza estadística de los *blobs* para el proceso de seguimiento.

Las ventajas que ofrece el modelo de *blobs* son [35, 25, 20]:

1. Analizan a la persona en función de sus características internas (píxeles que definen al *blob*) ya que es una representación basada en regiones.

2. Han tenido éxito en la extracción de la persona y su seguimiento en tiempo real en escenas complejas.

3. No requieren de hardware especializado para cumplir con su tarea.

4. Pueden originarse a partir de similitud en color, textura, movimiento, una combinación de éstas, entre otras.

5. Permiten que el problema del seguimiento sea resuelto en una forma estable.

6. Permiten demostrar la utilidad de las características estocásticas basadas en regiones (vector medio y matriz de covarianza), para el entendimiento en tiempo real de una imagen.

Para alcanzar el objetivo planteado se desarrolló en este trabajo de tesis el diseño e implementación de un sistema de VC enfocado a la localización y seguimiento de una persona, mediante su cabeza, a través de una secuencia de escenas dinámicas, en

una escena cerrada con control de iluminación. Se implementó un modelo estadístico multi-clase, de color y forma, para obtener la representación en 2D de la cabeza.

El diseño antes mencionado consta de dos etapas (las cuales se explicarán a detalle en la sección 3.2):

1. **Etapa de inicialización**: Se almacena la información de la escena del fondo y se obtiene la representación en 2D de la cabeza de la persona a través de un modelo estadístico multi-clase (de color y forma), es decir, por medio de un *blob*.

2. **Etapa de seguimiento:** Se identifican los píxeles del área en movimiento que pertenecen a la cabeza de la persona. Para ello se empleó un clasificador de Mahalanobis.

1.3. Organización de la tesis

La organización de la presente tesis se divide en cinco capítulos y cuatro apéndices. El primer capítulo es una introducción al trabajo desarrollado. Inicia describiendo el área de la VC, profundizando en las características del problema a resolver, presentado el objetivo y la propuesta de solución, finalizando con la descripción breve del diseño utilizado.

En el capítulo dos se presenta la fundamentación teórica sobre la que descansa este trabajo.

El capítulo tres inicia con la descripción del sistema, presenta el diseño y los módulos que lo componen, describiendo detalladamente cada uno de estos módulos. Después se muestran las herramientas de hardware y software utilizadas.

El capítulo cuatro muestra las pruebas y los resultados obtenidos con el sistema desarrollado en esta tesis.

El capítulo cinco presenta las conclusiones y trabajos futuros que se podrían realizar para mejorar la tesis.

El apéndice A describe la librería de OpenCV y las funciones que proporciona.

El apéndice B presenta la instalación de DirectX SDK y OpenCV, además presenta la configuración para que Visual C++ 6.0 trabaje con OpenCV, y finaliza con la descripción para crear un nuevo proyecto que utilice las librerías de OpenCV.

El apéndice C es el manual de usuario para utilizar la aplicación "Seguimiento de la cabeza" desarrollada en la presente tesis.

El apéndice D es el manual de usuario para utilizar la aplicación que genera los archivos de inicialización utilizados en la aplicación "Seguimiento de la cabeza".

Capítulo 2

Marco teórico

2.1. Representación de imágenes

El término *imagen* se refiere a una función bidimensional de intensidad de luz $f(x, y)$, donde x y y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x, y) es el nivel de brillo de la imagen en ese punto.

Una imagen digital es una función $f(x, y)$ que ha sido discretizada tanto en coordenadas espaciales como en la brillantez y puede considerarse como una matriz cuyos renglones y columnas identifican a un punto de la imagen, y el valor correspondiente al elemento de la matriz identifica el nivel de gris de ese punto. Los elementos de tal arreglo digital son llamados elementos de imagen o píxeles [10]. La intensidad de una imagen monocromática f en las coordenadas (x, y) se conoce como el nivel de gris ℓ de la imagen en ese punto, ℓ está dentro del rango:

$$L_{min} \leq \ell \leq L_{max}$$

En teoría, el único requerimiento sobre L_{min} es que sea no negativo, y sobre L_{max} es que sea finita. El intervalo $[L_{min}, L_{max}]$ se llama escala de grises (de la imagen). En la práctica, el intervalo se representa numéricamente como $[0, L]$, donde $\ell = 0$ se considera

negro y $\ell = L$ se considera como blanco en la escala. Todos los valores intermedios son sombras y variaciones continuas de gris que van del negro al blanco.

En el proceso de digitalización se requiere decidir sobre los valores del número de niveles de gris permitidos para cada píxel, el cual puede calcularse mediante la siguiente fórmula [15]:

$$G = 2^m$$

donde G indica el número de niveles de gris y m el número de bits empleados para representar a cada nivel de gris.

De manera que si cada nivel de gris se representa con 8 bits, entonces el nivel de gris permitido es de 256 posibles valores, los cuales corresponden a valores enteros entre 0 y 255, donde el 0 se considera como negro y a 255 como el nivel de intensidad claro.

Una imagen a color tiene una representación similar, comúnmente es descrita por la distribución de tres componentes de color R (Red), G (Green), B (Blue) - los tres colores primarios: rojo, verde y azul -. En una imagen en formato RGB las intensidades de cada una de las tres componentes de color están almacenadas en un byte, por lo tanto se requiere de tres bytes para almacenar el color de un píxel en la imagen (24 bits por píxel).

Una imagen digital se representa por un arreglo bidimensional de píxeles, en este arreglo los índices del renglón y columna de un píxel son valores enteros. El píxel localizado en $[0, 0]$ (origen) está situado en la esquina superior izquierda de la imagen. El índice i apunta hacia abajo haciendo referencia a los renglones, y el índice j apunta a la derecha indicando las columnas [16], este arreglo se muestra en la Figura 2.1. Para una imagen en escala de grises, el valor de la imagen $[i, j]$ es el nivel de gris de ese píxel, mientras que para una imagen a color en el píxel $[i, j]$ se tienen los valores de intensidad R,G,B.

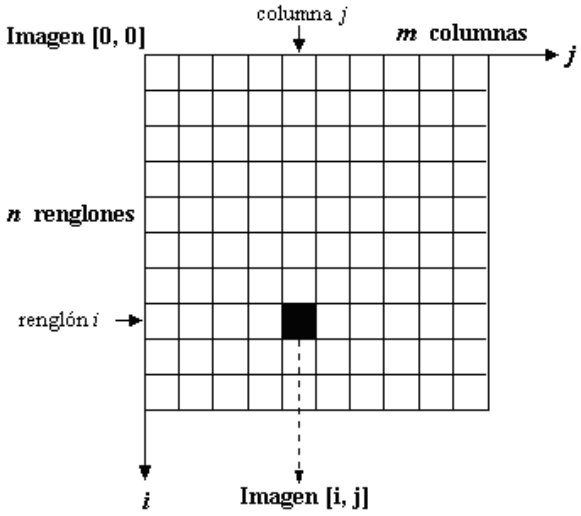


Figura 2.1: Arreglo bidimensional que representa a una imagen digital.

2.2. Diferencia de imágenes

La diferencia entre dos imágenes $f(x, y)$ y $h(x, y)$ se expresa como: $g(x, y) = f(x, y) - h(x, y)$, y se obtiene comparando las dos imágenes píxel por píxel. La diferencia de imágenes tiene numerosas aplicaciones importantes en la segmentación, detección de movimiento, y en la mejora de la calidad de las imágenes [10].

En este trabajo se utilizará a la diferencia de imágenes como una herramienta para detectar el movimiento. El movimiento es una característica poderosa, usada por el humano y por los animales para extraer objetos de interés de un fondo de detalles irrelevantes. En aplicaciones de imágenes, el movimiento crece a partir de un desplazamiento relativo entre el sistema de sensores y la escena que está siendo observada, tal como en las aplicaciones de robótica, la navegación autónoma, y el análisis de escenas dinámicas [10].

2.2.1. Diferencia de imágenes en escenas dinámicas

Una escena dinámica puede considerarse como una secuencia de $frames$, donde cada $frame$ representa una imagen de la escena en un instante particular de tiempo, y se denota por $F(x, y, t)$, donde x y y son las coordenadas espaciales en el $frame$ que representa a la escena en el tiempo t. Debido a que los $frames$ son tomados en intervalos regulares, se asume que t representa el t-ésimo $frame$ de la secuencia, en lugar de considerar que el $frame$ fue tomado en el tiempo absoluto t [16].

La diferencia de imágenes es una técnica espacial para la detección de movimiento. La mayoría de las técnicas de análisis de escenas dinámicas se basan en la detección de cambios en una secuencia de $frames$. Una de las técnicas más sencillas de detección de cambios es la diferencia de imágenes. El aspecto más atractivo de esta técnica espacial para la detección del movimiento es su simplicidad: Supongamos que tenemos una imagen de referencia conteniendo sólo componentes estacionarios. Comparando esta imagen contra una imagen subsecuente que tenga el mismo ambiente pero que incluya también un objeto en movimiento, la diferencia de las dos imágenes cancela los componentes estacionarios, dejando sólo las entradas diferentes de cero que corresponden a los componentes de la imagen no estática [10].

Para detectar el cambio entre dos $frames$, este método realiza una comparación directa entre los píxeles de los dos $frames$ a cotejar. En su forma más simple, una diferencia binaria de imágenes $D_{jk}(x, y)$ entre los $frames$ $F(x, y, j)$ y $F(x, y, k)$ se obtiene de la siguiente forma [16]:

$$D_{jk}(x,y) = \begin{cases} 1 & \text{si} \quad |F(x,y,j) - F(x,y,k)| > \tau \\ \\ 0 & \text{En cualquier otro caso} \end{cases} \tag{2.1}$$

donde τ es un umbral.

En una diferencia de imágenes, los píxeles que tienen un valor de 1 podrían ser considerados como el resultado del movimiento de un objeto o de cambios en la iluminación. El umbral juega un papel importante, ya que si los objetos se mueven lentamente y la intensidad varía lentamente, también estos cambios podrían no ser detectados para un umbral determinado. En su forma más simple, la diferencia de imágenes es susceptible al ruido. Los cambios en la iluminación y el registro de la cámara, además del ruido electrónico de la cámara, pueden dar como resultado falsas alarmas [16]. En el análisis de imágenes dinámicas todos los píxeles que tienen valor de 1 son considerados como el resultado del movimiento del objeto. Esta propuesta es aplicable sólo si las dos imágenes son registradas y la iluminación es relativamente constante en los límites establecidos por el umbral. En la práctica, los valores con 1 frecuentemente se incrementan debido a la presencia del ruido. Comúnmente estas entradas son puntos aislados en la diferencia de imágenes. Una propuesta simple para removerlos es formar regiones conectadas de cuatro u ocho píxeles con valores de 1, y entonces ignorar cualquier región que sea más pequeña u objetos que se estén moviendo más lentamente. Esta propuesta mejora el resultado de la diferencia de imágenes, así se obtiene realmente el resultado del movimiento [10].

2.3. Segmentación

Una de las principales etapas del análisis de imagen y del reconocimiento es la segmentación, la cual consiste en subdividir la imagen en sus partes constituyentes u objetos. El nivel al que se lleva a cabo esta subdivisión depende del problema a resolver [15].

En general, la segmentación autónoma es una de las tareas más difciles en el procesamiento digital de imágenes. Una buena segmentación puede conducir al éxito mientras

que una segmentación errónea casi siempre garantiza el fracaso en algún problema de visión [10].

En el caso de imágenes digitales en blanco y negro los algoritmos de segmentación se dividen de acuerdo a dos propiedades básicas de los valores del nivel de gris, que son discontinuidad y similaridad [11]:

1. La discontinuidad segmenta una imagen en base a los cambios bruscos de nivel de gris. Esta propiedad a su vez puede aplicarse de dos maneras distintas: Detección de puntos aislados y detección de líneas y bordes de una imagen.

2. La similaridad se basa en dividir una imagen de acuerdo a las similitudes de nivel de gris. Los métodos de segmentación de esta categoría se basan en: La umbralización, crecimiento de región y división y fusión de regiones.

Por otro lado, la segmentación de imágenes digitales en color consiste en la adaptación de las técnicas utilizadas para imágenes digitales en blanco y negro [11]. Se han generado una gran cantidad de trabajos que presentan técnicas, modelos y algoritmos para la segmentación de dichas imágenes. Estas técnicas están divididas en cuatro grupos [1]:

1. Segmentación basada en el valor del píxel.

2. Segmentación basada en el área.

3. Segmentación basada en orillas.

4. Segmentación basada en la física.

La segmentación basada en el valor del píxel comprende las técnicas basadas en el histograma, es decir, se obtiene el histograma de una imagen, se localizan puntos máximos y se analizan los intervalos que rodean a éstos durante el proceso de segmentación.

La segmentación por agrupamiento de píxeles se realiza en algún espacio de color de acuerdo a alguna característica homogénea. Los algoritmos que usan agrupamiento difuso en espacios de color, también se considera dentro de este grupo.

La segmentación basada en orillas toma en cuenta la discontinuidad entre regiones disjuntas adyacentes.

La segmentación basada en la física asume que el cambio de color entre dos regiones de la imagen no implica necesariamente una discontinuidad en la forma, iluminación u otras características. Para interpretar correctamente escenas más complejas, las características físicas múltiples se deben examinar para determinar si dos regiones de diferente color de la imagen pertenecen al mismo objeto [23].

El problema de la segmentación ha sido (y aún lo es) un área de investigación importante, y como se explicó en los párrafos anteriores existe una amplia variedad de métodos de segmentación propuestos en la literatura. Como también puede observarse, muchos de estos métodos se basan en alguna propiedad de discontinuidad (llamados métodos basados en contornos), o en alguna propiedad de similitud (conocidos como métodos basados en regiones) [7].

De ahí que en la práctica sólo pueden distinguirse dos tipos de segmentación: La segmentación de regiones o zonas y la segmentación por contornos. Lo que varía para cada tipo de segmentación es aquello que se desea segmentar, para el primer caso se desea segmentar zonas, y para el segundo, ya dentro de una región concreta, se desean segmentar objetos individuales; así que el resultado de la segmentación usualmente son los bordes de una región o todos los puntos que pertenecen a la región en sí misma, en ambos casos se requiere de convertir los datos a una forma adecuada para su procesamiento [10].

La segmentación por contornos y la segmentación por regiones pueden considerarse como complementarias o duales una de la otra, en el sentido que si se conoce comple-

tamente el contorno o frontera de un objeto, se pueden determinar propiedades de la región. De la misma forma, si se identifica completamente la región del objeto, pueden determinarse propiedades del contorno, aunque esto no significa que proporcionen resultados iguales. Cada técnica tiene sus ventajas y limitaciones, y está ligada fuertemente a la aplicación en cuestión. Ninguno de estos dos tipos de segmentación por sí sólo produce una segmentación ideal. Un procedimiento de cooperación entre ambas técnicas permitiría explotar las ventajas de cada método y reducir sus desventajas [7, 15].

Para efectos de este trabajo, a continuación se abundará en las técnicas de segmentación de extracción de regiones, o llamadas también técnicas de segmentación orientadas a regiones. Dentro de los algoritmos de extracción de regiones nos enfocaremos al algoritmo llamado "segmentación por crecimiento de regiones", debido a que este algoritmo fue utilizado para obtener píxeles del color de piel que pertenecen a la cabeza de una persona, a partir de un $frame$. Mediante este algoritmo se obtuvieron los tonos de piel para valores en el espacio de color RGB normalizado, con el fin de crear una distribución normal de color de piel.

El algoritmo de "segmentación por crecimiento de regiones" se utilizó independiente al programa de seguimiento, es decir, en una aplicación diferente se obtuvieron mediante dicha segmentación las regiones de color de piel de un conjunto de personas, se analizaron los píxeles clasificados como piel y con ellos se generó una distribución normal de color de piel, la cual se empleó en el sistema de seguimiento de la presente tesis.

2.3.1. Formulación básica de segmentación orientada a regiones

Sea R la representación de la región que contiene a la imagen completa. Puede verse a la segmentación como el proceso que divide a R en n subregiones, $R_1, R_2, ..., R_n$, tales

que [10]:

1. $\bigcup_{i=1}^{n} R_i = R$
2. R_i es una región conectada, para $i = 1, 2, ..., n$
3. $R_i \cap R_j = \oslash$; para toda i y j, $i \neq j$
4. $P(R_i) =$ VERDADERO para $i = 1, 2, ..., n$
5. $P(R_i \cup R_j) =$ FALSO para $i \neq j$.

donde $P(R_i)$ es un predicado lógico sobre los puntos en el conjunto R_i y $\oslash$ es el conjunto vacío. Estas condiciones pueden resumirse como sigue: La primer condición implica que cada punto de la imagen debe estar contenido en una región, es decir, la segmentación debe ser completa; esto significa que el algoritmo de segmentación no debe terminar hasta que cada punto sea procesado. La segunda condición expone que los puntos en una región deben estar conectados o ser contiguos. La tercera condición indica que las regiones deben ser disjuntas. La cuarta condición determina qué clase de propiedades deben satisfacer los píxeles en una región segmentada. Finalmente la condición cinco sugiere que las regiones R_i y R_j son diferentes en el sentido del predicado P.

2.3.2. Segmentación por crecimiento de regiones

Este método de segmentación se basa en el crecimiento de una región si su interior es homogéneo de acuerdo a ciertas características tales como el nivel de gris, textura, o información de color. La segmentación se inicia dividiendo la imagen en pequeñas regiones, estas regiones iniciales pueden ser pequeños conjuntos de píxeles o sólo un píxel. La implementación más sencilla inicia eligiendo un punto denominado píxel semilla.

Entonces, la región crece cuando se agregan los píxeles vecinos que satisfacen cierto criterio de homogeneidad. Esto es, el criterio de homogeneidad tiene la función de decidir si un píxel pertenece a la región (en crecimiento) o no [10, 12, 7] . El procedimiento de crecimiento de regiones presenta dos problemas inmediatos, uno es la selección de los puntos semilla iniciales, que representarán apropiadamente a las regiones de interés; el otro problema es la selección de propiedades adecuadas, para incluir puntos en varias regiones durante el proceso del crecimiento. La selección de uno o más puntos iniciales frecuentemente depende de la naturaleza del problema.

La selección del criterio de similitud depende no sólo del problema en consideración, sino que también depende del tipo de datos de la imagen. Otro problema en el crecimiento de regiones es la formulación de una regla de parada. Básicamente, el crecimiento de una región debería terminar, cuando no hay más píxeles que satisfacen el criterio de inclusión en esa región [10]. A pesar de que este algoritmo es computacionalmente caro, presenta las siguientes ventajas [4]:

- Permite utilizar varias propiedades de la imagen de manera directa y simultánea para localizar los límites entre regiones. La posición de los límites entre regiones diferentes coincide bien con los límites percibidos subjetivamente.

- El método es extremadamente insensible a la distorsión cuando los puntos iniciales son elegidos correctamente.

- No requiere conocimiento a priori sobre la imagen, por lo que es una opción muy prometedora para realizar la segmentación de escenas naturales donde no se dispone de un amplio conocimiento a priori.

2.4. Modelado de objetos 2D por medio de un *blob*

Después de que una imagen ha sido segmentada en regiones, el siguiente paso es representar estas regiones en una forma apropiada para un procesamiento adicional de la imagen. La representación básicamente puede hacerse por dos medios [10]:

1. La región puede representarse en términos de sus características externas (como sus bordes).

2. La región puede ser representada en función de sus características internas (como son los píxeles contenidos en la región).

Una representación externa es adecuada cuando el punto de interés son las características de la forma, mientras que una representación interna es conveniente cuando lo importante son las propiedades de reflectividad, tales como el color y la textura. En cualquier caso, las características elegidas como descriptoras deben ser insensibles tanto como sea posible a cambios en el tamaño, traslaciones y rotaciones.

La representación que se estudió y aplicó en esta tesis es una representación interna conocida como *blob*. Esta representación fue desarrollada como una forma de extraer una descripción extremadamente compacta y estructuralmente significativa de imágenes satelitales multi-espectrales[1] . En un *blob* las características descriptoras se representan mediante vectores. Los vectores que caracterizan a cada píxel se forman agregando las coordenadas espaciales (x, y) a los componentes espectrales (o de textura) de la imagen, así que las propiedades de la imagen tales como color y similitud espacial se combinan para formar regiones conectadas coherentemente. Con base en lo anterior, un *blob* puede definirse como una nube de puntos, el cual se describe por alguna propiedad visual que

[1] Una imagen satelital multiespectral es aquella que es generada a partir de los datos recolectados por un mismo sensor en más de una banda. Cada banda opera a una longitud de onda diferente generando una imagen, y al conjunto de ellas se le llama imagen multiespectral [19, 10].

es compartida por todos los píxeles del *blob* y que no es compartida por los píxeles del entorno. Esta propiedad podría ser el color, la textura, brillantez, movimiento, una combinación de estas, entre otras [36].

Los *blobs* representan aspectos globales. La reducción de los grados de libertad de los píxeles individuales a los parámetros del *blob* es una forma de regularización, la cual permite que el problema sea solucionado de una forma estable [36, 20]. Cada *blob* puede ser caracterizado por sus estadísticas de bajo orden: Sea el *blob* k, se define $\vec{\mu}_k$ como su vector medio y $\sum_k$ como su matriz de covarianza. Es importante la interpretación física de los parámetros del *blob* [36]:

1. A través de la matriz de covarianza se obtiene la distribución del *blob*, es decir, proporciona una representación detallada de la forma y apariencia del *blob*.

2. Y con el vector medio se calcula el centro geométrico del área del *blob*.

En seguida se abundará sobre el cálculo de éstos.

2.4.1. Vector característico

Un vector característico es [31]: un vector n-dimensional de características numéricas que representan a algún objeto. Los objetos pueden ser modelados como vectores de rasgos distintivos, cada uno de los cuales corresponde a un punto en el espacio de características multidimensional. Para realizar una clasificación con base en el modelo de vectores característicos, se debe seleccionar cuáles características son relevantes, determinando una manera de medirlas, y definir un criterio que permita diferenciar los objetos deseados de los otros. Una vez que el espacio característico es definido, debe ser particionado en regiones que correspondan a diferentes modelos de objetos; de esta manera se permite la asignación de objetos desconocidos a clases de objetos conocidos.

2.4.2. Vector medio

Para el cálculo del vector medio hay que considerar que las posiciones de los píxeles en un espacio multi-espectral[2] pueden describirse por vectores característicos, cuyos componentes son las respuestas espectrales individuales en cada componente (o banda) de la imagen. Entonces, en un espacio multi-espectral con un gran número de píxeles, donde cada píxel es descrito por su correspondiente vector $\vec{x}$, la posición promedio de los píxeles en el espacio se define por el valor esperado del vector del píxel $\vec{x}$, de acuerdo a [28]:

$$\vec{\mu} = \varepsilon\{\vec{x}\} = \frac{1}{k}\sum_{j=1}^{k}\vec{x}_j \tag{2.2}$$

Donde $\vec{\mu}$ es el vector del píxel promedio y $\vec{x}_j$ son los vectores de los píxeles individuales de un número total k. El vector promedio es útil para definir el promedio o la posición esperada de píxeles en el espacio multi-espectral.

2.4.3. Matriz de covarianza

El papel de la matriz de covarianza es describir la dispersión o propagación de la posición de los píxeles en el espacio multi-espectral, así como detectar si existe correlación entre las variables. La matriz de covarianza se define por [28]:

$$\sum = \varepsilon\{(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^t\} \tag{2.3}$$

En la cual el superíndice t denota el vector transpuesto. Si hay correlación entre las respuestas en un par de bandas espectrales, el elemento fuera de la diagonal correspondiente en la matriz de covarianza, será grande en comparación con los términos de la

[2] Un espacio multiespectral o multidimensional tiene tantos ejes o dimensiones como componentes espectrales estén asociados a cada píxel definido en él [10].

diagonal. Por otro lado, si hay poca correlación, los términos alejados de la diagonal serán cercanos a cero. La matriz de covarianza es simétrica, y si en el conjunto de datos de la imagen no hay correlación entre cualquiera de sus componentes multiespectrales, la matriz de covarianza es una matriz diagonal.

2.4.4. Valores y vectores propios: transformación de rotación

Para el desarrollo de la transformación de componentes principales es fundamental encontrar un nuevo sistema de coordenadas en el espacio multiespectral en el cual los datos puedan ser representados sin correlación; en otras palabras, tal que la matriz de covarianza en el nuevo sistema de coordenadas sea diagonal.

Si los vectores que describen los puntos originales del píxel son representados como y en el nuevo sistema de coordenadas, entonces se desea encontrar una transformación lineal G de las coordenadas originales, tal que [13, 28]:

$$\vec{y} = G\vec{x} \tag{2.4}$$

La ecuación 2.4 está sujeta a la restricción de que la matriz de covarianza de los datos del píxel en el espacio y sea diagonal. Esta transformación se puede apreciar en la Figura 2.2 para un caso particular de un espacio vectorial bidimensional, donde se muestra el sistema de coordenadas y modificado, en el cual el vector tiene componentes no correlacionados.

En el espacio y el vector medio es por definición:

$$\vec{\mu}_y = \varepsilon\{\vec{y}\} = \varepsilon\{G\vec{x}\} = G\varepsilon\{\vec{x}\} = G\vec{\mu}_x \tag{2.5}$$

dado que:

$$\varepsilon\{G\vec{x}\} = \frac{1}{K}\sum_{j=1}^{K} G\vec{x}_j = G\frac{1}{k}\sum_{j=1}^{K}\vec{x}_j = G\vec{\mu}_x \tag{2.6}$$

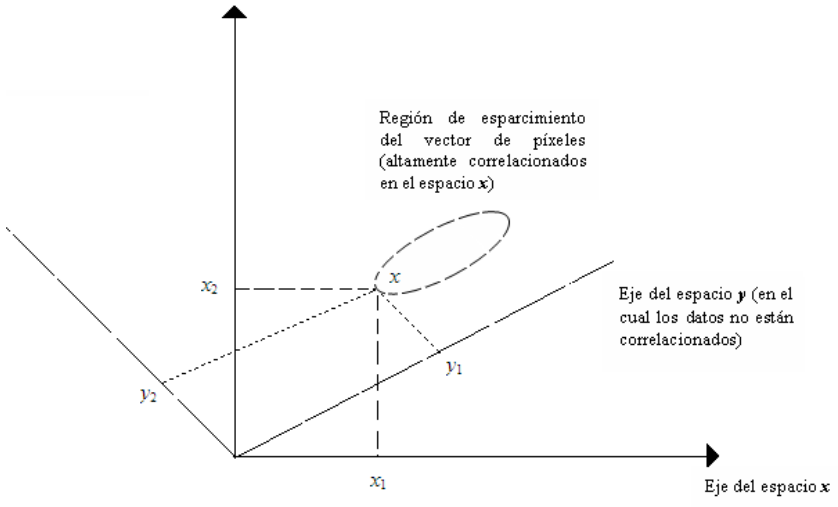


Figura 2.2: Ilustración de un sistema de coordenadas modificado cuyo vector de píxeles tiene componentes no correlacionados [28].

donde $\vec{\mu}_x$ es el vector promedio en el espacio x. Por consiguiente la matriz de covarianza en el espacio y se define por:

$$\sum\nolimits_y = \varepsilon\{(G\vec{x} - G\vec{\mu}_x)(G\vec{x} - G\vec{\mu}_x)^t\} \tag{2.7}$$

la cual también puede escribirse como:

$$\sum\nolimits_y = G\varepsilon\{(\vec{x} - \vec{\mu}_x)(\vec{x} - \vec{\mu}_x)^t\}G^t \tag{2.8}$$

dado que:

$$[G\vec{x}]^t = \vec{x}\ ^t G^t \tag{2.9}$$

por lo tanto se tiene:

$$\sum\nolimits_y = G\sum\nolimits_x G^t \tag{2.10}$$

donde $\sum_x$ es la covarianza de los datos del píxel en el espacio x. Dado que $\sum_y$ debe

ser diagonal, G puede ser reconocida como la matriz traspuesta de vectores propios de $\sum_x$, siempre que G sea una matriz ortogonal. Como resultado, $\sum_y$ puede identificarse como la matriz diagonal de valores propios de $\sum_x$:

$$\sum_{\mathbf{y}} = \begin{pmatrix} \lambda_1 & 0 & & \\ 0 & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix} \tag{2.11}$$

donde N es la dimensionalidad de los datos. Dado que $\sum_y$ es, por definición, una matriz de covarianza y es diagonal, sus elementos serán las varianzas de los datos del píxel en sus respectivas coordenadas transformadas. Estos datos están ordenados tal que $\lambda_1 > \lambda_2 > ... > \lambda_N$, de modo que el dato que exhibe máxima varianza está en y_1, la próxima varianza más grande está en y_2 y así sucesivamente, con la mínima varianza en y_N.

La transformación de los componentes principales efectuada de acuerdo a (2.4) y sujeta a la restricción diagonal de (2.10) es también conocida como Transformación Karhunen-Loeve o Trasformación de Hotelling.

Para determinar la transformación de los componentes principales es necesario encontrar los valores y vectores propios de la matriz de covarianza $\sum_x$. Los valores propios están dados por la solución de la ecuación característica:

$$| \sum_x - \lambda I | = 0 \tag{2.12}$$

donde I es la matriz identidad.

En seguida se mostrará un ejemplo del cálculo de valores y vectores propios para el caso bidimensional, que es el que se emplea en este trabajo: Sea $\sum_x$ la siguiente matriz

de covarianza en el espacio original x, para datos de la imagen altamente correlacionados:

$$\sum_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix} \tag{2.13}$$

Entonces al sustituir (2.13) en (2.12) se tiene:

$$\begin{vmatrix} 1.90 - \lambda & 1.10 \\ 1.10 & 1.10 - \lambda \end{vmatrix} = 0 \tag{2.14}$$

ó $\lambda^2 - 3.0\lambda + 0.88 = 0$, la cual produce los siguientes valores para λ: 2.67 y 0.33.

Una vez resuelta la ecuación característica, la matriz de covarianza en el sistema de coordenadas apropiado y quedaría de la forma de la ecuación 2.11:

$$\sum_y = \begin{bmatrix} 2.67 & 0 \\ 0 & 0.33 \end{bmatrix}$$

Con $\lambda_1 = 2.67$ y $\lambda_2 = 0.33$.

En seguida lo que interesa es encontrar la auténtica matriz de trasformación de los componentes principales G. Considere primero el vector propio que corresponde a $\lambda_1 = 2.67$ como la solución a la ecuación:

$$[\sum_x - \lambda_1 I]\vec{g}_1 = 0 \tag{2.15}$$

con:

$$\vec{g}_1 = \begin{bmatrix} g_{11} \\ g_{21} \end{bmatrix} \tag{2.16}$$

Al Sustituir $\sum_x$ y λ_1 se obtiene el siguiente par de ecuaciones:

$$-0.77g_{11} + 1.10g_{21} = 0$$

$$1.10g_{11} - 1.57g_{21} = 0$$

Las cuales son dependientes, dado que el conjunto de estas ecuaciones es homogéneo. Sin embargo, no existe una solución trivial debido a que el sistema de ecuaciones tiene determinante cero. De cada una de las ecuaciones se puede ver que:

$$g_{11} = 1.43g_{21} \tag{2.17}$$

Y debido a que la matriz resultante G tiene que ser ortogonal: $G^{-1} \equiv G^t$, se requiere que los vectores propios sean normalizados, así que:

$$g_{11}^2 + g_{21}^2 = 1 \tag{2.18}$$

Esta ecuación puede resolverse simultáneamente con la ecuación (2.17) para obtener $\vec{g}_1$:

$$\vec{g}_1 = \begin{bmatrix} 0.82 \\ 0.57 \end{bmatrix}$$

De una manera similar se puede mostrar que el vector propio correspondiente a $\lambda_2 = 0.33$ es:

$$\vec{g}_2 = \begin{bmatrix} -0.57 \\ 0.82 \end{bmatrix}$$

Por lo que la matriz de transformación de componentes principales requerida es:

$$G = \begin{bmatrix} 0.82 & -0.57 \\ 0.57 & 0.82 \end{bmatrix}^t = \begin{bmatrix} 0.82 & 0.57 \\ -0.57 & 0.82 \end{bmatrix}$$

En el caso general, los resultados se pueden interpretar como sigue: Los vectores propios individuales $\vec{g}_1$, $\vec{g}_2$, ..., $\vec{g}_N$ definen los ejes de los componentes principales en términos del espacio de coordenadas original. Es evidente que los datos no están correlacionados en los nuevos ejes y que los nuevos ejes son una rotación del conjunto

original. Por esta razón la transformación de los componentes principales es clasificada como una trasformación rotacional [28].

También es posible conocer el ángulo de rotación del nuevo sistema de coordenadas en el cual los datos no están correlacionados con respecto a los datos originales (Figura 2.3). La ecuación 2.19 permite calcular el coseno del ángulo menor que forman las direcciones dadas por dos vectores. A partir de esta ecuación se obtiene el ángulo que forman dos rectas, dos planos o una recta y un plano (ecuación 2.20), siempre que se conozcan los vectores propios (los cuales se denotan por $\vec{u}$ y $\vec{v}$) de cada uno de ellos:

$$\cos\phi(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\mid\vec{u}\mid\cdot\mid\vec{v}\mid} \tag{2.19}$$

por consiguiente:

$$\phi(\vec{u},\vec{v}) = \cos^{-1}\left(\frac{\vec{u}\cdot\vec{v}}{\mid\vec{u}\mid\cdot\mid\vec{v}\mid}\right) \tag{2.20}$$

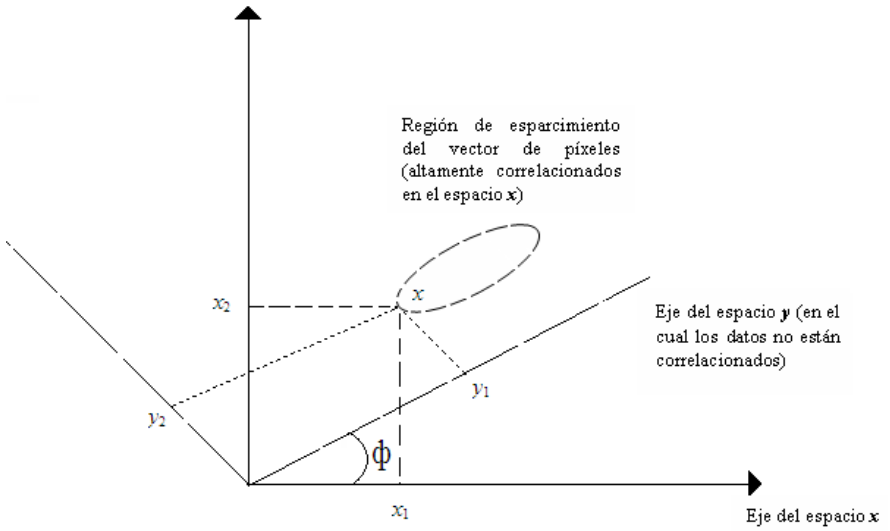


Figura 2.3: El ángulo ϕ indica el ángulo de rotación del nuevo sistema de coordenadas con respecto al espacio de los datos originales.

2.5. Seguimiento de objetos por medio de un clasificador de Mahalanobis

Durante la etapa de seguimiento se requiere identificar qué píxeles del área en movimiento pertenecen a la región de la cabeza de la persona. Cada píxel es representado a través de un vector característico (también llamado patrón vectorial). Una forma de determinar la pertenencia a una clase de un patrón vectorial desconocido, consiste en asignarlo a la clase del prototipo más próximo. En este trabajo se utilizará la distancia de Mahalanobis para determinar el grado de proximidad.

2.5.1. Distancia de Mahalanobis

La distancia Euclidiana entre dos puntos $\vec{x} = (x_1, ..., x_p)^t$ y $\vec{y} = (y_1, ..., y_n)^t$ en el espacio p-dimensional R^p es definida como [8]:

$$d_E(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + ... + (x_p - y_p)^2} = \sqrt{(\vec{x} - \vec{y})^t(\vec{x} - \vec{y})} \tag{2.21}$$

La distancia Euclidiana simplemente es la distancia de una línea recta (como se define la línea recta en la geometría Euclidiana) entre dos puntos dentro de una espacio p-dimensional. Sin embargo en la estadística se prefiere una distancia que por cada componente (variable) tomen la variabilidad de la dispersión respecto a una clase.

La distancia de Mahalanobis es una distancia estadística que generaliza la distancia euclidiana entre dos vectores en la que se tiene en cuenta la dispersión de las variables y su dependencia. Un valor alto de la distancia de Mahalanobis indica que el punto se aleja del centro de la clase [15]. La distancia de Mahalanobis es una forma de determinar la similitud entre dos variables aleatorias multidimensionales, y se define para un objeto $\vec{X}$ y una clase α, de media $\vec{\mu}$ y matriz de covarianza $\sum$, como el escalar:

$$d_m(\vec{X}, \alpha) = (\vec{X} - \vec{\mu})^t \sum^{-1} (\vec{X} - \vec{\mu}) \tag{2.22}$$

Donde $\vec{X}$ es el vector de características, $\vec{\mu}$ es el vector medio, $\sum^{-1}$ es la matriz inversa de la matriz de covarianza de la muestra. Ver Figura 2.4.

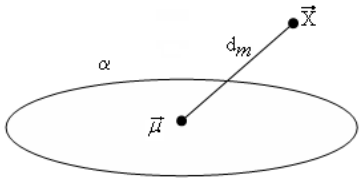


Figura 2.4: d_m es la distancia de Mahalanobis, esta distancia entre $\vec{X}$ y $\vec{\mu}$, la cual toma en cuenta la dispersión intrínseca de la clase α.

2.5.2. Clasificador de Mahalanobis

Para la fase de clasificación se utiliza un clasificador lineal de distancia mínima. El clasificador lineal se construye a partir de los prototipos de cada clase que se desea clasificar. Un prototipo es un vector que contiene los valores ideales de los discriminantes de la clase. A partir del prototipo se construye el vector de pesos aumentado, el cual representa la línea que separa a la clase representada por el prototipo, del resto de las clases [8].

La distancia de Mahalanobis puede usarse como un clasificador de distancia mínima como sigue: Sean $\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_c$ los vectores medios para las c clases, y sean $\sum_1, \sum_2 ..., \sum_c$ sus correspondientes matrices de covarianza; la clasificación de un vector característico $\vec{x}$ a una de las c clases se define por la medida de la distancia de Mahalanobis a cada clase, y $\vec{x}$ se asigna a la clase para la cual la distancia de Mahalanobis es la distancia mínima, se puede ver esto en la Figura 2.5 [8, 10].

El uso de la métrica de Mahalanobis tiene las siguientes propiedades:

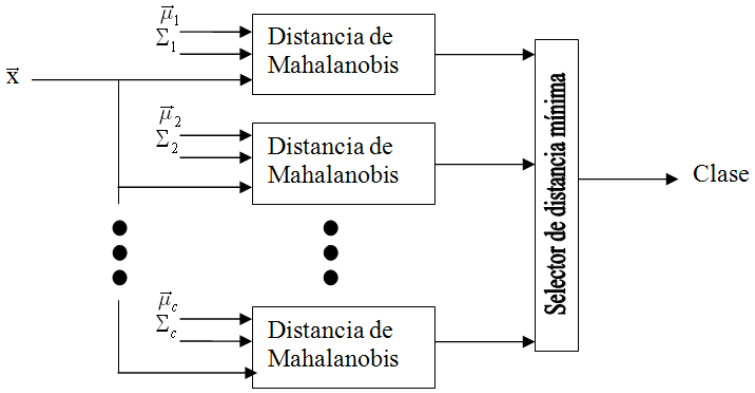


Figura 2.5: Esquema de un clasificador basado en la distancia mínima de Mahalanobis [8].

1. Automáticamente considera la escala de los ejes coordenados.

2. Corrige la correlación entre las diferentes características.

3. Provee de decisiones de límites tanto en curvas como en líneas.

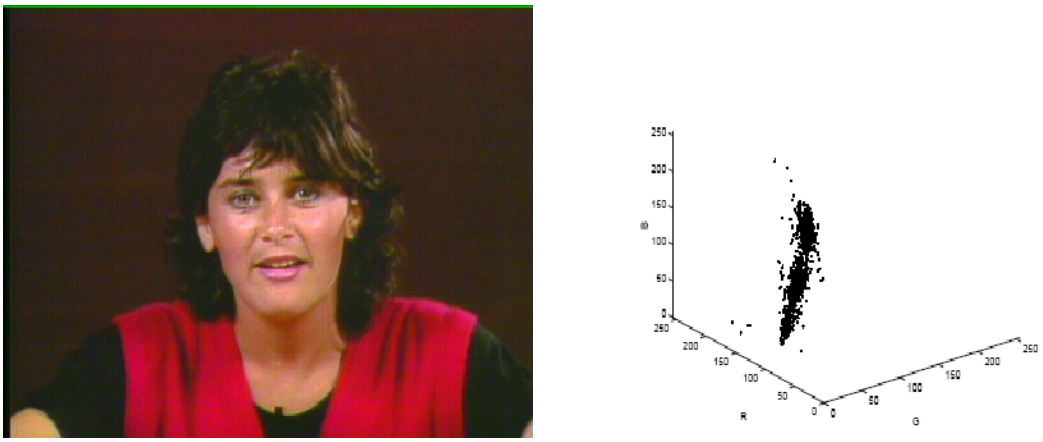
2.6. Técnicas de seguimiento basadas en color

Es conocido que las diferentes personas tienen diferentes apariencias de color de piel, para cada persona la apariencia de color de piel puede variar si viste con distintas ropas o bajo diferentes condiciones de iluminación. En otras palabras, muchos factores contribuyen a la apariencia del color de la piel humana. Para la percepción del color, un espacio en 3D tal como el espacio RGB es escencial, ya que la mayoría de las cámaras utilizan el modelo RBG para representar imágenes en color, además otros modelos de colores pueden ser fácilmente convertidos en un modelo RGB.

La distribución del color de piel en el espacio de color RGB puede obtenerse a partir de un histograma de color de piel. Un histograma de color es una distribución de colores en un espacio de color y ha sido utilizada por la comunidad de VC en el análisis de una imagen. Se ha mostrado que los histogramas de color son representaciones estables de un

objeto que no son afectadas por oclusión y cambios de posición del objeto, y que pueden usarse para diferenciar entre una gran cantidad de objetos. Se ha mostrado además que los colores no caen aleatoriamente en el plano, sino que forman agrupaciones en puntos específicos. Los histogramas de color del color de piel coinciden con estas observaciones.

La Figura 2.6 muestra un rostro humano y las ocurrencias del color de piel en el espacio de color RGB (256x256x256). En esta figura puede observarse que los colores de piel están agrupados en un área pequeña del espacio de color RGB, y sólo unos cuantos de todos los colores posibles son los que efectivamente están presentes en un rostro humano [37].



(a) Rostro humano **(b)** Ocurrencias del color de piel

Figura 2.6: Un ejemplo de un rostro humano y las ocurrencias del color de piel en el espacio de color RGB [37].

Para localizar a la cabeza de la persona en este trabajo se utilizó una segmentación basada en el color de la piel (el color de la piel se representa en el espacio de color RGB), la cual se define en el trabajo de Yang y Waibel [37]. Esta técnica de segmentación considera el hecho de que los colores de la piel se agrupan en una región pequeña, como se explica en las líneas anteriores, en un espacio de color normalizado, formando una distribución de color de piel bien definida. Bajo ciertas condiciones de luz, esta distribución

del color de piel puede caracterizarse por una distribución normal multivariada definida por $N(\vec{\mu}, \sum)$, donde el vector medio $\vec{\mu} = (\bar{r}, \bar{g})$, y la matriz de covarianza $\sum$ se definen por [37]:

$$\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i \tag{2.23}$$

$$\bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i \tag{2.24}$$

$$\sum = \begin{bmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{bmatrix} \tag{2.25}$$

Como puede notarse en las ecuaciones (2.23) y (2.24) sólo se consideraron los colores cromáticos (r, g), conocidos como colores puros en la ausencia de brillantez, los cuales se definen mediante el proceso de normalización como:

$$r = \frac{R}{R+G+B} \tag{2.26}$$

$$g = \frac{G}{R+G+B} \tag{2.27}$$

$$b = \frac{B}{R+G+B} \tag{2.28}$$

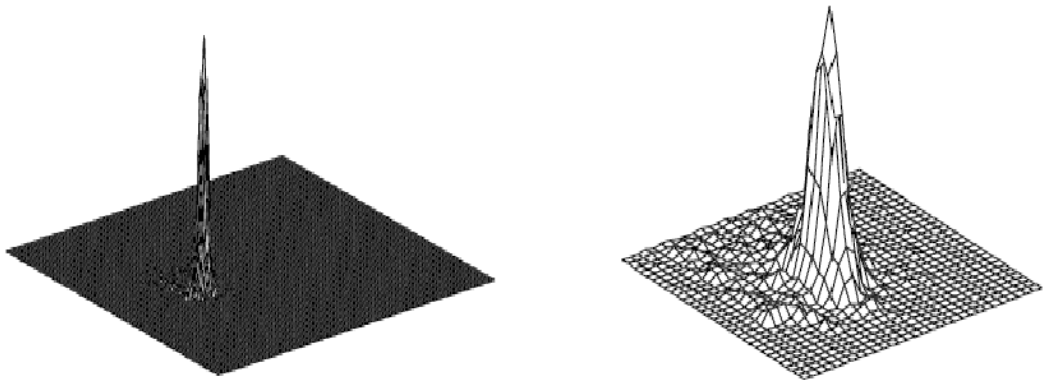
La normalización se realizó para eliminar la brillantez del color, ya que en el problema de localización de rostros humanos la brillantez no es importante. Se ha mostrado que las diferencias del color de piel entre las personas puede reducirse a través de la normalización de la intensidad [37], después de la normalización el color azul (ecuación 2.28) es redundante debido a que $r + g + b = 1$. Con esto se ha mostrado que las diferencias de distribuciones de color han sido reducidas después de la normalización.

En otras palabras, los colores de piel de las diferentes personas varían menos en el espacio de color normalizado. Este resultado es significativo debido a que provee un posibilidad de modelar rostros humanos con apariencias de color diferentes en el espacio de colores monocromáticos.

Hasta ahora se ha establecido que los colores de piel forman un grupo en el espacio de color y que varían menos en el espacio de color monocromático. Sin embargo, es imposible caracterizar todas las distribuciones de color de piel utilizando un modelo fijo, debido a que la distribución del color de piel está relacionada no sólo con el color de piel, sino también con el color de la iluminación, por ejemplo: la luz solar moverá los histogramas de color hacia el azul, debido a que contiene más azul que la luz fluorescente. Por otro lado, aunque los colores de la piel de diferentes personas parecen variar en un rango amplio, es posible modelar la distribución del color de piel de cada individuo bajo cierta condición de iluminación. Dado que la distribución de color de piel tiene sólo dos variables en el espacio de color normalizado, es conveniente analizar tal distribución gráficamente. La Figura 2.7 muestra una distribución de color de piel de la imagen de la Figura 2.6 en el espacio de RGB normalizado [37].

Además, se ha comprobado que la forma de la distribución del color de piel de una persona permanece similar aunque haya un cambio en la distribución bajo condiciones cambiantes de iluminación, y se ha descubierto que tal distribución tiene una forma regular. Por lo que al comparar la forma de las distribuciones de color de piel con una distribución normal bivariada, se puede concluir que es posible usar una distribución normal bivariada para caracterizar las distribuciones de color de piel [37].

Una aplicación directa del modelo de color de piel consiste en localizar a un rostro humano en una imagen. Una forma directa para localizar a un rostro humano es igualar el modelo con la imagen de entrada para encontrar los grupos del color del rostro humano. Cada píxel de la imagen original se convierte al espacio de color cromático



(a) Vista global **(b)** Vista local

Figura 2.7: Distribución de color de piel de la imagen de la Figura 2.6 en el espacio de color normalizado [37].

y entonces se compara con la distribución del modelo de color de piel. Debido a que los colores de piel ocurren en una área pequeña del espacio de color cromático, la equiparación es muy rápida. Lo anterior es muy utilizado en el seguimiento en tiempo real de un rostro humano [37].

En este trabajo se empleó un clasificador de Mahalanobis para determinar si un píxel representado por un vector característico pertenece o no a la clase color de piel. Esta clase de color de piel fue construida mediante un conjunto de píxeles en tonos de color de piel en el espacio de color RGB normalizado y se representa con $N(\vec{\mu}, \sum)$. Para obtener los píxeles de tonos de color de piel, se obtuvo una imagen de cada una de las personas que colaboraron en la realización de este proyecto y a cada imagen se le aplicó el algoritmo de segmentación por crecimiento de regiones en la región del rostro de la persona.

Capítulo 3

Seguimiento de la cabeza de una persona

3.1. Descripción del sistema

El objetivo de la presente tesis es localizar y realizar el seguimiento de la cabeza de una persona por medio de un *blob* en una escena cerrada bajo condiciones de iluminación controladas. Este objetivo fue alcanzado con la implementación de un sistema de VC. En este sistema se utilizó un modelo en 2D de la cabeza de la persona, es decir un *blob*, debido a las ventajas que éste proporciona y por la naturaleza del problema.

El sistema se implementó de acuerdo al diseño de la Figura 3.1 y será explicado a detalle en la sección 3.2.

3.1.1. Alcances del sistema

Como se explicó en la sección de introducción de la presente tesis, se siguió la siguiente configuración: Escena cerrada, cámara estacionaria, con objetos en movimiento (personas), con control de luz. Para obtener una escena cerrada y control de luz se eligió como lugar la Sala de Juntas del Instituto de Electrónica y Computación[1] . En

[1] Ubicado en la Universidad Tecnológica de la Mixteca.

esta escena se grabaron secuencias de video que permitieron realizar las pruebas y observaciones para el desarrollo del sistema.

El sistema está limitado a localizar y realizar el seguimiento de una sola persona, lo cual significa que también construye y modela en 2D una sola representación de *blob*.

3.2. Diseño e implementación

De acuerdo al diseño presentado en la Figura 3.1, el sistema consta de dos etapas, la primera es la inicialización del sistema (localización de la cabeza de la persona), y la segunda etapa corresponde al seguimiento. En la etapa de inicialización se realizó una clasificación del área en movimiento de acuerdo al color de piel, la cual requirió de dos modelos de distribución normal: uno que define a la distribución normal de color de piel y otro a la distribución del fondo. A continuación se detalla como se realizó la construcción de estos modelos y la implementación de cada una de las etapas y subetapas que se aprecian en la Figura 3.1.

3.2.1. Creación de los modelos de color de piel y fondo

La construcción del modelo de la cabeza y del modelo del fondo se realizó de manera previa e independiente al diseño del sistema de seguimiento. En seguida se describe cómo se llevo a cabo este proceso.

Construcción del modelo de color de piel

Para la construcción de la clase cabeza se consideró la información de color de piel, de manera que lo que se realizó fue la construcción de un modelo de color de piel o de una distribución normal del color de piel $N(\vec{\mu}_c, \sum_c)^2$, descrita en la sección 2.6, la cual

[2] El subíndice c denota a la clase cabeza

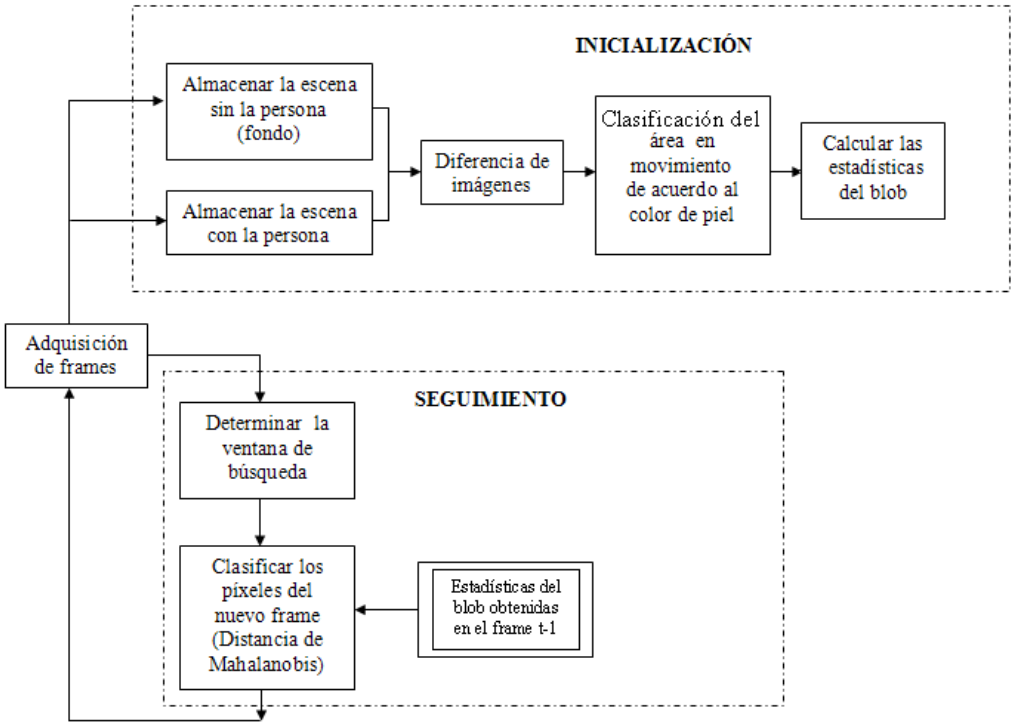


Figura 3.1: Esquema de los módulos que son empleados en la detección y seguimiento de la cabeza de una persona.

se efectuó en un programa independiente al sistema de seguimiento como sigue:

1. Se realizó inicialmente la grabación de 25 personas, para cada una de las grabaciones se almacenó la escena con la persona con vista frontal.

2. Las muestras del color de piel se obtuvieron al aplicar el algoritmo de "Segmentación por crecimiento de regiones"(descrito en la sección 2.3.2) a cada una de las imágenes de las 25 personas. Los puntos semillas se colocaron manualmente sobre el área que se deseaba segmentar. Se utilizó un valor de umbral empírico de 4 como criterio de homogeneidad para realizar la segmentación.

3. Lo siguiente fue almacenar en un archivo de texto todos los valores r, g del espacio de color RGB normalizado de acuerdo a las ecuaciones (2.23) y (2.24) de los píxeles obtenidos por la segmentación, por lo que el archivo contiene sólo dos valores por cada píxel.

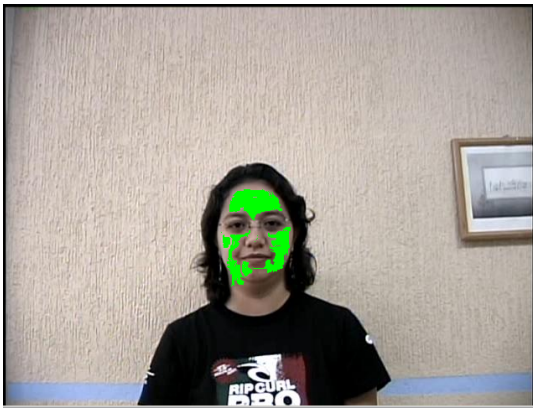
4. Todos los valores obtenidos se utilizaron para calcular las estadísticas de bajo orden del modelo del color de piel $N(\vec{\mu}_c, \sum_c)$: El vector medio ($\vec{\mu}_c$) y la matriz de covarianza ($\sum_c$) (según las ecuaciones 2.2 y 2.3 respectivamente), ambos también se almacenan en un archivo de texto:

$$\vec{\mu}_c = (\bar{r}, \bar{g}) \tag{3.1}$$

$$\sum\nolimits_{\mathbf{c}} = \begin{pmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{pmatrix} \tag{3.2}$$

Las Figuras 3.2(a) y 3.2(b) muestran a una de las personas que colaboró en la construcción de $N(\vec{\mu}_c, \sum_c)$, se puede notar el resultado de la segmentación por crecimiento de regiones con los píxeles en color verde.

(a) Imagen de prueba antes de aplicar la segmentación por crecimiento de regiones



(b) Imagen después de seleccionar los puntos semilla y obtener la segmentación por crecimiento de regiones

Figura 3.2: Aplicación de la segmentación por crecimiento de regiones.

Construcción del modelo del fondo

Se almacenó la escena del fondo ($frame$ sin la persona) de cada una de las 25 grabaciones realizadas, por lo que el modelo del fondo de la escena se construyó con todos los píxeles de estas escenas del fondo. De manera similar al modelo de color de piel, el modelo del fondo está definido por un vector medio $\vec{\mu}_f$ y una matriz de covarianza $\sum_f$ cuyos vectores característicos contienen dos componentes:

$$\vec{\mu}_f = (\bar{r}, \bar{g}) \tag{3.3}$$

$$\sum\nolimits_{\mathbf{f}} = \begin{pmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{pmatrix} \tag{3.4}$$

Los valores r y g de los píxeles de las imágenes del fondo, así como la información de la matriz de covarianza y el vector medio de este modelo también se almacenan en archivos de texto.

Los archivos que contienen la información de la matriz de covarianza y el vector

medio del modelo de color de piel y del modelo del fondo son llamados archivos de inicialización.

Inicialmente se analizaron los videos de 25 personas y posteriormente se incorporaron los videos de otras 7, por lo que se dispone de archivos de inicialización de 25 y 32 personas, esto se explica a detalle en la sección 4.1.

3.2.2. Inicialización

En esta etapa se obtiene el primer modelo de la cabeza de la persona, para realizarlo se detecta el área en movimiento a través de una diferencia de imágenes, una vez localizada dicha área en movimiento, se extrae de ella el área correspondiente a la cabeza a través de una segmentación basada en el color de piel (con ayuda de la distribución normal del color de piel descrita en 3.2.1). En las Figuras 3.3(a) y 3.3(b), se muestran las imágenes con las que se realiza la etapa de inicialización, esto es el fondo y la imagen con la persona en la escena respectivamente. Cada una de las subetapas involucradas en la inicialización se explican enseguida.

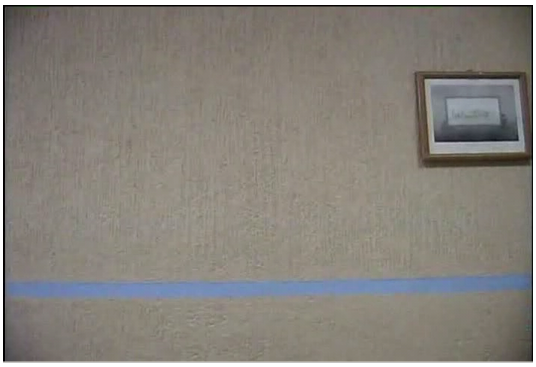
Diferencia de imágenes

Para calcular la diferencia de imágenes se almacena la escena del $frame$ sin la persona, es decir, el fondo del escenario, después se almacena la escena con la persona (ver Figuras 3.3(a) y 3.3(b)) y a estas imágenes se le aplica la ecuación:

$$D_{jk}(x,y) = \begin{cases} 1 & \text{si } dif > \tau \\ \\ 0 & \text{En cualquier otro caso} \end{cases} \tag{3.5}$$

Donde:

- $dif = \sqrt{(F.R - V.R)^2 + (F.G - V.G)^2 + (F.B - V.B)^2}$



(a) Fondo del escenario utilizado para las pruebas

(b) Escenario con la persona

Figura 3.3: Imágenes con las que se realiza la etapa de inicialización

- F es el $frame$ del fondo de la escena
- V es el $frame$ actual del video.
- Las letras R, G y B significan que se comparó cada uno de los componentes de color del espacio RGB.
- El valor del umbral τ se fijó en 45.

Esta diferencia se utilizó para clasificar a los píxeles que pertenecen al área en movimiento y a los píxeles que pertenecen al fondo de la escena. A cada píxel se le aplicó la ecuación 3.5, si la diferencia es mayor al umbral τ (obtenido empíricamente), entonces se clasifica como área en movimiento; de lo contrario se clasifica como área del fondo de la escena. Esto se muestra en las Figuras 3.4(a) y 3.4(b), el área en movimiento se representa en color negro.

Los píxeles que fueron clasificados como fondo de la escena fueron descartados totalmente, mientras que los píxeles que se clasificaron en movimiento se analizaron para determinar cuáles pertenecían a la cabeza de la persona.

(a) Imagen antes de aplicar la diferencia de imágenes con respecto al fondo de la escena

(b) Imagen donde se aprecia el resultado de la diferencia de imágenes mediante puntos negros

Figura 3.4: Resultado del cálculo de la diferencia de imágenes.

Clasificación del área en movimiento de acuerdo al color de piel

Para elegir a los píxeles pertenecientes a la cabeza de la persona se empleó un clasificador de Mahalanobis (sección 2.5.2). Este clasificador de distancia mínima consideró dos clases: el modelo de color de piel y el modelo del fondo (descritos en la sección 3.2.1). Para cada uno de los píxeles que se encontraron en el área en movimiento, se calcula su distancia de Mahalanobis (ecuación 2.22) con respecto a estas dos clases, y la distancia que resulte menor es a la clase que pertenece el píxel en cuestión.

Construcción y modelado del *blob* de la cabeza

A partir de los píxeles que se detectaron como parte de la cabeza (píxeles en movimiento con color de piel), se crea la clase inicial de la *cabeza* (o el *blob* inicial de la cabeza), para la cual se calcula el vector medio ($\vec{\mu}_{bc}$) y la matriz de covarianza ($\sum_{bc}$), $\vec{\mu}_{bc}$ y $\sum_{bc}$ se calculan de acuerdo a las ecuaciones 2.2 y 2.3 respectivamente. Los vectores característicos empleados en el cálculo de $\vec{\mu}_{bc}$ y $\sum_{bc}$ contienen información espacial y de color, por lo que tienen cuatro componentes: las coordenadas x y y del

píxel y sus componentes de color normalizados r y g:

$$\vec{\mu}_{bc} = (\bar{x}, \bar{y}, \bar{r}, \bar{g}) \tag{3.6}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.7}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.8}$$

$$\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i \tag{3.9}$$

$$\bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i \tag{3.10}$$

$$\sum\nolimits_{\mathbf{bc}} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xr} & \sigma_{xg} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yr} & \sigma_{yg} \\ \sigma_{rx} & \sigma_{ry} & \sigma_{rr} & \sigma_{rg} \\ \sigma_{gx} & \sigma_{gy} & \sigma_{gr} & \sigma_{gg} \end{pmatrix} \tag{3.11}$$

donde N es la cantidad de píxeles considerados como pertenecientes a la cabeza.

A partir de $\vec{\mu}_{bc}$ y $\sum_{bc}$ se tiene la información necesaria para poder dibujar el *blob* de la cabeza, el cual se representa por una elipse:

1. El centro de la elipse está descrito por los componentes $\bar{x}$, $\bar{y}$ del vector medio $\vec{\mu}_{bc}$ (ecuación 3.6).

2. Los ejes mayor y menor de la elipse se calculan a partir de la dispersión de los píxeles en x y en y, esta dispersión se obtiene por medio de la desviación estándar en x (V_x) y en y (V_y). Para el cálculo de V_x y V_y se toma la raíz cuadrada positiva de las varianzas en x y en y respectivamente (las cuales corresponden a los dos primeros elementos de la diagonal principal de $\sum_{bc}$):

$$V_x = \sqrt{\sigma_{xx}} \tag{3.12}$$

$$V_y = \sqrt{\sigma_{yy}} \tag{3.13}$$

3. El color de relleno de la elipse se obtiene mediante el promedio de los valores en R, G y B de los N píxeles que pertenecen a la clase *cabeza*:

$$\text{color\_de\_relleno} = (\bar{R}, \bar{G}, \bar{B})$$
$$\bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$$
$$\bar{G} = \frac{1}{N}\sum_{i=1}^{N} G_i$$
$$\bar{B} = \frac{1}{N}\sum_{i=1}^{B} R_i$$

En la Figura 3.5 se aprecian ejemplos del *blob* de la cabeza. Los píxeles en tono rojo que aparecen en la Figuras 3.5(a), 3.5(c) y 3.5(e), son aquellos que fueron detectados como color de piel y por lo tanto clasificados como pertenecientes a la cabeza de la persona; el *blob* se muestra por medio de la elipse en color rojo, y como puede apreciarse representa los aspectos globales de dispersión en x y en y de los píxeles que lo definen. Por otro lado, en la Figuras 3.5(b), 3.5(d) y 3.5(f) se muestra el *blob* con el color de piel promedio de los píxeles que lo componen.

Debido a los movimientos posibles de la cabeza de la persona y con el propósito de que el *blob* se adecuara a tales movimientos, fue necesario calcular un ángulo de rotación para la elipse (de acuerdo a la ecuación 2.20) cuando los datos espaciales están correlacionados. Para el cálculo de los valores propios se utilizó el algoritmo QR implementado en la librería desarrollada por el M.C. Joaquín Peña Acevedo. El Algoritmo QR calcula los valores propios de una matriz tridiagonal, la matriz original queda reducida a su forma diagonal, por lo que los elementos de la diagonal son sus valores propios. Para el cálculo de los vectores propios, una vez obtenidos los valores propios se sustituyen

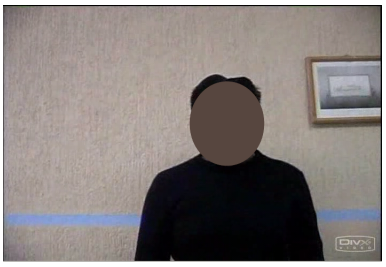
(a) *Blob* con píxeles

(b) *Blob* completo

(c) *Blob* con píxeles

(d) *Blob* completo

(e) *Blob* con píxeles



(f) *Blob* completo

Figura 3.5: Representación del *blob* mediante una elipse.

en la ecuación 2.12 y se genera un sistemas de ecuaciones, el cual se resuelve por el método de eliminacion gaussiana con pivoteo de filas escaladas, dicho método también está definido en la librería antes mencionada (para mayor información sobre el algoritmo QR y el método de eliminación gaussiana con pivoteo de filas escaladas consultar [19]). Ejemplos de esta situación se aprecian en la Figura 3.6.

Construcción y modelado de la clase fondo inicial

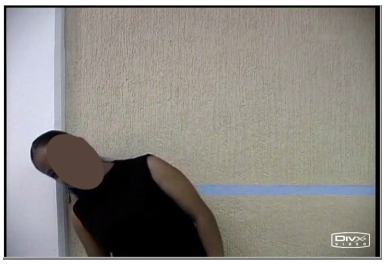
Para la etapa de seguimiento se requiere diferenciar de los píxeles del área en movimiento aquellos que pertenecen a la cabeza de la persona de los que no, para ello se emplea un clasificador de Mahalanobis que considera dos clases: La clase cabeza (el *blob* de la cabeza) y la clase fondo. Para ello se requiere construir una clase fondo diferente a la mencionada anteriormente (ecuaciones 3.3 y 3.4), esta clase se modifica para que los vectores característicos de los píxeles que la definen contengan ahora cuatro componentes: Las coordenadas x y y del píxel y sus componentes de color normalizados r y g, de modo que el cálculo de la matriz de covarianza y el vector medio de está clase se obtienen por:

$$\vec{\mu}_{nf} = (\bar{x}, \bar{y}, \bar{r}, \bar{g}) \tag{3.14}$$

$$\sum\nolimits_{nf} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xr} & \sigma_{xg} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yr} & \sigma_{yg} \\ \sigma_{rx} & \sigma_{ry} & \sigma_{rr} & \sigma_{rg} \\ \sigma_{gx} & \sigma_{gy} & \sigma_{gr} & \sigma_{gg} \end{pmatrix} \tag{3.15}$$

Esta nueva clase fondo (nf) se construye con los píxeles que no fueron clasificados como clase cabeza en la etapa de inicialización y contiene además de la información de color información espacial.

(a) *Blob* con píxeles



(b) *Blob* completo

(c) *Blob* con píxeles

(d) *Blob* completo

(e) *Blob* con píxeles

(f) *Blob* completo

Figura 3.6: Elipse con un ángulo de rotación debido a los movimientos de la cabeza de la persona.

El cálculo de las estadísticas del primer *blob* de la cabeza ($\vec{\mu}_{bc}$ y $\sum_{bc}$) y de la nueva clase fondo ($\vec{\mu}_{nf}$ y $\sum_{nf}$) marca el término de la etapa de inicialización, dichas estadísticas se utilizan en la etapa de seguimiento como se explica en la sección 3.2.3.

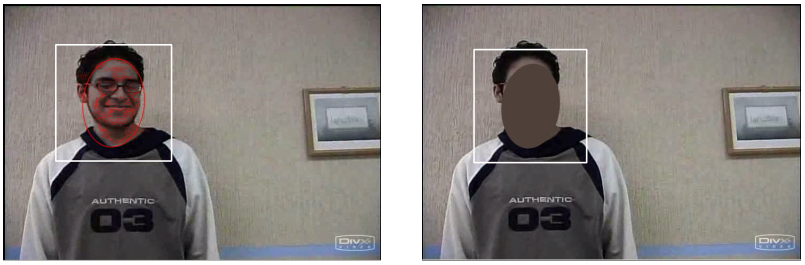
3.2.3. Seguimiento

La etapa de seguimiento de la persona es un ciclo continuo de análisis de $frames$ y como puede observarse en la Figura 3.1, para el $frame$ actual se consideran las estadísticas del *blob* obtenidas en el $frame$ anterior, de este modo cuando se inicia el seguimiento se emplean las estadísticas de *blob* construido en la etapa de inicialización así como las estadísticas de la nueva clase fondo.

Para iniciar el seguimiento primero se determina a partir de la longitud del *blob* una ventana de búsqueda cuadrada. El tamaño de cada lado de esta ventana se delimita a partir del eje mayor de la elipse que representa al *blob*, se obtiene sumándole al eje mayor 50 píxeles. Esta ventana se utiliza para mantener el buen rendimiento del programa, ya que no se busca a la cabeza de la persona en todo el escenario, solamente se busca en esta área. Esto es válido ya que se considera que la adquisición de $frames$ es más rápida que el movimiento de la persona. En la Figura 3.7 se muestra un ejemplo de la ventana de búsqueda representada como un cuadrado de color blanco. Ahora se analizan sólo los píxeles que se encuentran en la ventana de búsqueda sobre la cual se aplica de nuevo la diferencia de imágenes para detectar el área en movimiento actual.

Los píxeles del área en movimiento se pasan por el clasificador de Mahalanobis para determinar qué píxeles pertenecen a la clase cabeza y qué píxeles pertenecen a la clase fondo, y así poder construir el *blob* actual (ver la Figura 3.8).

Con el propósito de mejorar el rendimiento del sistema, se definió un mapa soporte (según se propone en [35, 36]). El mapa soporte es una matriz booleana del tamaño del



(a) *Blob* con píxeles **(b)** *Blob* completo

Figura 3.7: Representación de la ventana de búsqueda.

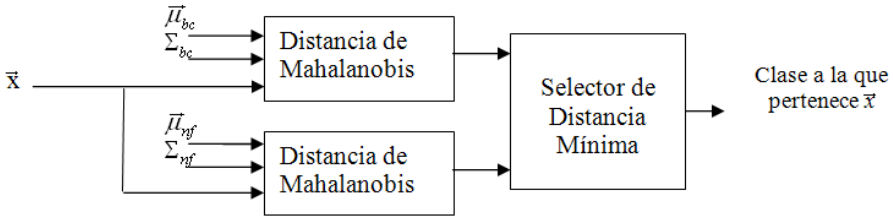


Figura 3.8: Clasificador de distancia mínima para determinar el *blob* actual.

frame (este tamaño varía de acuerdo a la resolución del video que se está analizando), en ella se marcan con 1 las posiciones que forman a la clase de la cabeza, y con 0 a las posiciones de los píxeles que pertenecen a la clase del fondo. De modo que la definición del mapa soporte para el *blob* k está dada por:

$$S_k(x,y) = \begin{cases} 1 & (x,y)\ \epsilon\ k \\ 0 & \text{en otro caso} \end{cases} \tag{3.16}$$

Con ayuda del mapa soporte se tiene acceso a la posición (coordenadas x, y) y al color (valores r, g del espacio de color RGB normalizado) de los píxeles de la cabeza en el *frame* actual, por lo que las estadísticas del nuevo modelo (*blob* actual k) se calculan con base en [35, 36]:

- El vector medio:

$$\vec{\mu}_k = \varepsilon[(\vec{y} - \vec{\mu}_{k-1})(\vec{y} - \vec{\mu}_{k-1})^t] \tag{3.17}$$

donde:

y es el conjunto de vectores característicos de los píxeles pertenecientes al modelo actual.

$\vec{\mu}_{k-1}$ es el vector medio del *blob* obtenido en el $frame$ anterior.

- La matriz de covarianza:

$$\sum\nolimits_k = \varepsilon[(\vec{y} - \vec{\mu}_k)(\vec{y} - \vec{\mu}_k)^t] \tag{3.18}$$

- Se puede simplificar 3.18 para facilitar el cálculo iterativo de $\sum_k$ como se aprecia en 3.19:

$$\varepsilon[(\vec{y} - \vec{\mu}_k)(\vec{y} - \vec{\mu}_k)^t] = \varepsilon[\vec{y}\vec{y}^{\,t}] - \vec{\mu}_k \vec{\mu}_k^{\,t} \tag{3.19}$$

De manera que es suficiente hacer un recorrido sobre $S_k(x, y)$ para poder resolver a 3.17 y el primer término de 3.19, y cómo se indica en 3.19 al finalizar el recorrido se obtiene $\sum_k$ actualizada. Adicionalmente, se calculan los valores y vectores propios del modelo del *blob* (sección 2.4.4), para calcular el ángulo de rotación y de este modo dibujar al *blob* de acuerdo a los movimientos de la cabeza.

Las estadísticas del nuevo modelo del fondo ($\vec{\mu}_{nf}$, $\sum_{nf}$) del frame actual se calculan de manera similar haciendo uso del mapa soporte.

Para mantener la estabilidad durante el seguimiento, se consideró aumentar el tamaño de la ventana de búsqueda en un 50 %, si el número de píxeles detectados como pertenecientes a la cabeza es menor a 45. Si el número de píxeles aumenta al analizar el siguiente $frame$ se detiene el aumento y la ventana de búsqueda depende nuevamente del eje mayor de la elipse del *blob*. El tamaño máximo que puede alcanzar la ventana de búsqueda es el tamaño del $frame$.

3.2.4. Herramientas de programación y hardware utilizado

Para la implementación del sistema se utilizaron la siguientes herramientas de software:

1. El lenguaje de programación Visual C++ 6.0.

2. Para la obtención de *frames* y el manejo de video se utilizó la librería abierta OpenCV. En el apéndice A se reseñan las características de esta librería.

Debido a que la librería OpenCV es compatible con Visual C++, y a las ventajas que éste lenguaje nos ofrece, se decidió desarrollar la tesis en el ambiente de Visual C++. Así, Visual C++ y openCV hacen una herramienta poderosa para desarrollar una aplicación de VC.

Algunas de estas ventajas que ofrece C++ para el objetivo de la tesis [3] son:

- Velocidad. Incluso con la velocidad de las computadoras modernas existen problemas por la eficiencia del lenguaje, el cual es un parámetro importante. C++ hereda la velocidad de C.

- Orientado a objetos. La programación orientado a objetos es una estrategia para la simplificación de tareas programadas.

- Reusable. La capacidad de fácil reuso y modificación de código existente para nuevos problemas. Soporta la herencia de los objetos que permite definir una clase modificando una o mas clases existentes.

- Portabilidad. C++ está estandarizado.

[3] La descripción de funciones de C++ está disponible en: http://msdn.microsoft.com/msdnmag/issues/01/04/STL/

Respecto al hardware utilizado en este proyecto, el sistema se realizó en un sistema de cómputo con las siguientes características:

- Procesador AMD 64 3000+.
- Memoria RAM de 1 Gb.
- Targeta de video: ATI Radeon 9250 de 128 Mb.
- Se requiere de un espacio mínimo en disco duro de 12 Mb para la instalación del sistema.

Se obtuvieron videos de tres cámaras distintas: Canon Xl1, Sony Handycam DCR-TRV130 Digital-8 y Modelo Canon Powershot pro[4] . Las características de los videos obtenidos con cada una de estas cámaras (su modelo, resolución y velocidad en $frames$ por segundo -fps-) se describen en la Tabla 3.1.

Cámara	Modelo	Resolución	Velocidad
A	Canon Xl1	720 x 576	25
B	Sony Handycam DCR-TRV130 Digital-8	320 x 320	15
C	Modelo Canon Powershot pro	640 x 480	15

Tabla 3.1: Características de las cámaras utilizadas en el desarrollo de la tesis.

La frecuencia por $default$ del sistema es de 21.73 Hz. (es decir, cada 0.046 segundos se obtiene una imagen para poder analizarla).

[4] **Disponible en:** http://www.nuevafotografia.com/stamodel.asp?valor=437 (último acceso: 22 de agosto de 2007).

Capítulo 4

Pruebas y Resultados

Se realizaron diferentes pruebas al sistema con el propósito de verificar que éste realizara el seguimiento de la cabeza de la persona. La evaluación del seguimiento es cualitativa y en cada prueba se muestran dos representaciones del *blob* que sigue a la cabeza de la persona, dado que el sistema puede representar al *blob* de dos maneras distintas: El *blob* completo, es decir, con un color de relleno (color promedio de los píxeles del *blob*); y el *blob* con píxeles, es decir, la elipse en color rojo con los píxeles que pertenecen al *blob* en también en color rojo. El desarrollo de las pruebas se hizo en dos etapas:

1. Se realizaron pruebas con los tres tipos de cámaras de video descritas en la Tabla 3.1, para mostrar cómo la calidad de la imagen del video afecta al sistema de seguimiento.

2. Después se probó la robustez del sistema a distintos tipos de movimientos de la persona.

A continuación se describe cada etapa y los resultados obtenidos.

4.1. Pruebas con diferentes tipos de cámaras de video.

Para realizar las pruebas del sistema se consideraron tres tipos distintos de cámaras de video, con el fin de obtener videos con diferentes tipos de resolución y con distintas velocidades en fps (frames por segundo). Primero se probó la estabilidad de la clase de color de piel con un conjunto de nuevos ejemplos, posteriormente se experimentó el sistema de seguimiento con resoluciones de video más bajas. Los resultados se describen a continuación:

- **Cámara de video A:** Con la cámara A se realizaron grabaciones de 25 personas distintas que colaboraron en este trabajo, incluyendo estudiantes y profesores de la institución[1] . Con estas grabaciones se construyeron cuatro archivos de inicialización del sistema, dos archivos para la clase cabeza y dos archivos para la clase fondo (ver apéndice D), para cada clase, en un archivo de texto se almacenó su matriz de covarianza y su vector medio; y en otro archivo se guardaron los valores r y g de los píxeles que la definen. Posteriormente se realizaron 7 grabaciones más con ésta misma cámara y se adquirieron de ellas los valores normalizados de los píxeles en tono de piel y de los píxeles de las escenas del fondo, por lo que se obtuvieron nuevos archivos de inicialización conteniendo las estadísticas de la clase color de piel y de la clase fondo de 32 grabaciones; con estos datos se probó al modelo de representación empleado en este trabajo y al clasificador de Mahalanobis de la siguiente manera:

  - Se realizaron pruebas a los nuevos videos con la información de los 25 videos obtenidos previamente, sin agregar los datos de los tonos de piel correspondientes a las últimas 7 grabaciones de piel a los archivos de inicialización, los

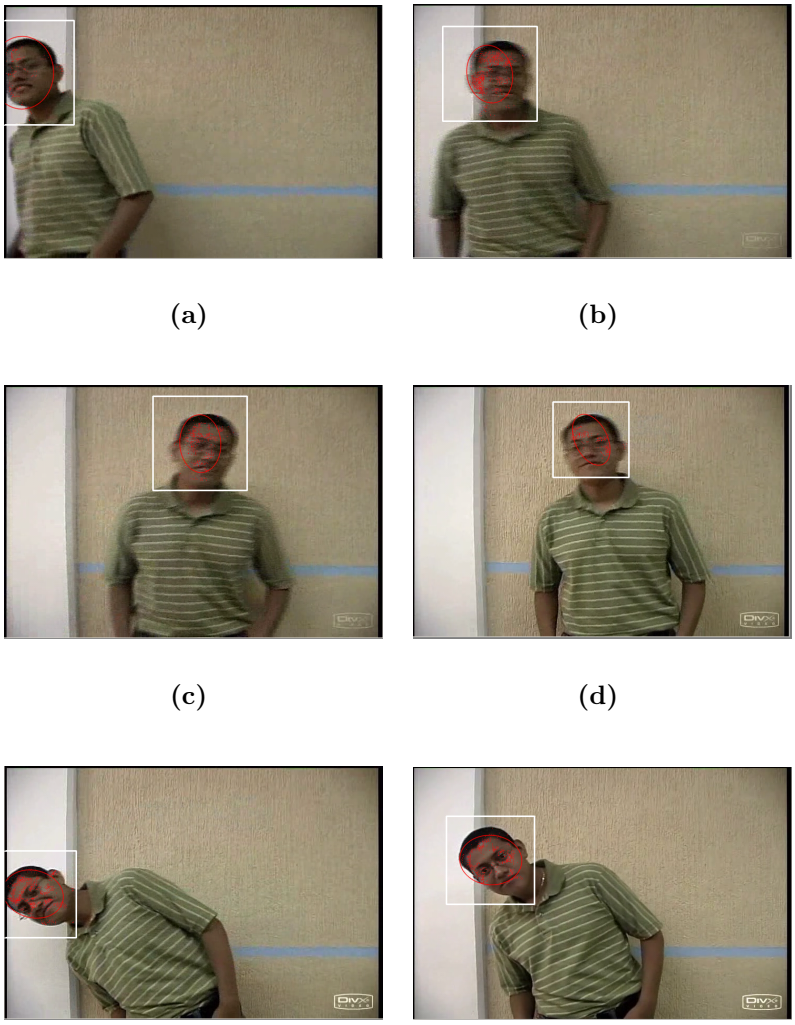
[1] Universidad Tecnológica de la Mixteca

resultados fueron buenos debido a que se realizó el seguimiento de manera correcta. Como se muestra en las Figuras 4.1 y 4.2.

- Más tarde se procedió a almacenar los valores normalizados de los tonos de piel de las personas de las 7 grabaciones a los archivos existentes y probar nuevamente al sistema con los nuevos videos, se detectó que se reconocieron más puntos cómo pertenecientes a la cabeza, pero el aspecto y tamaño del *blob* no cambió. Por lo que se mostró la estabilidad del *blob* al seguir a la cabeza variando la cantidad de información interna empleada en su construcción. Esto se observa en las Figuras 4.3 y 4.4.

- **Cámara de video B:** Para la cámara B (la cual tiene menor resolución con respecto a la anterior) se realizarón grabaciones de 8 personas distintas, los resultados obtenidos fueron muy desalentadores, ya que por la calidad de la imagen, el sistema no pudo clasificar apropiadamente a los píxeles del color de la piel, por lo que el *blob* se dibujó de manera incorrecta y el seguimiento no fue adecuado. Algunos de los resultados obtenidos aparecen en las Figuras 4.5 y 4.6.

- **Cámara de video C:** Para la cámara C sólo se hicieron grabaciones con 2 personas distintas para comprobar el comportambiento del sistema, los resultados fueron mejores comparados con los obtenidos con la cámara B, pero sin llegar a ser tan buenos como los de la cámara A. La calidad de la imagen mejoró, por lo que el sistema sí reconoció a los píxeles de color de piel, sin embargo, para algunos movimientos de la cabeza gran parte de los píxeles del rostro no fueron identificados, por lo que el *blob* por momentos se dibujaba erróneamente. Este comportamiento se aprecia en la Figura 4.7.

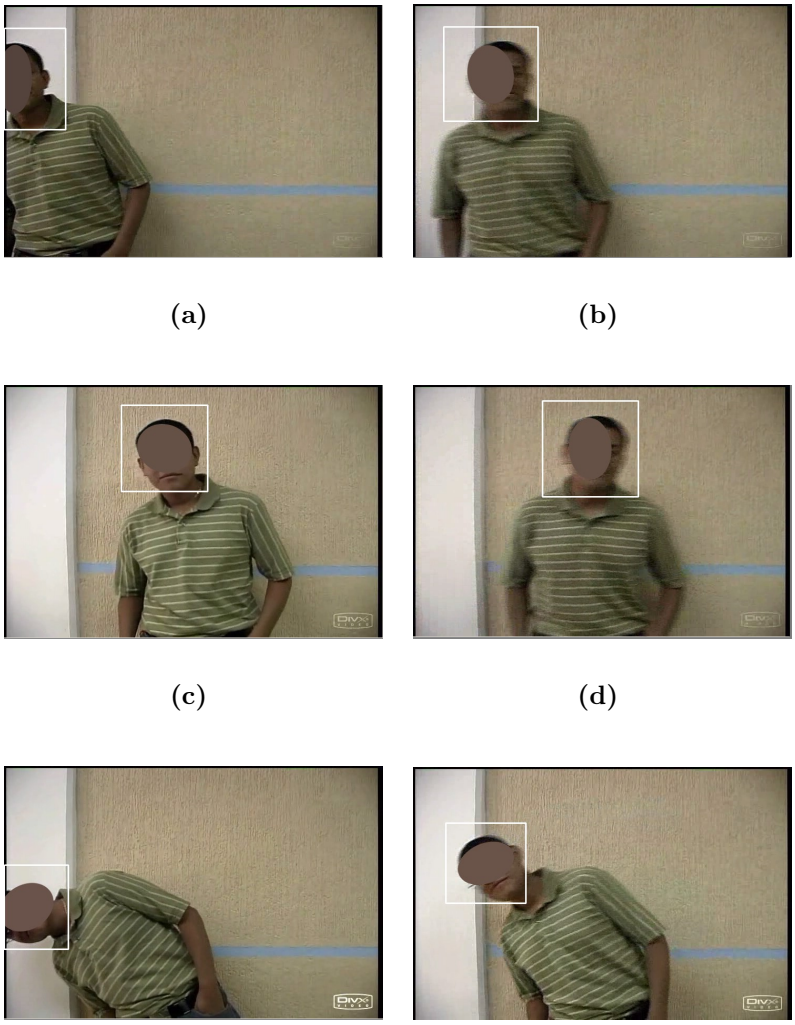


(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.1: Seguimiento sin agregar los datos de tonos de piel de las últimas 7 grabaciones, utilizando el *blob* con píxeles

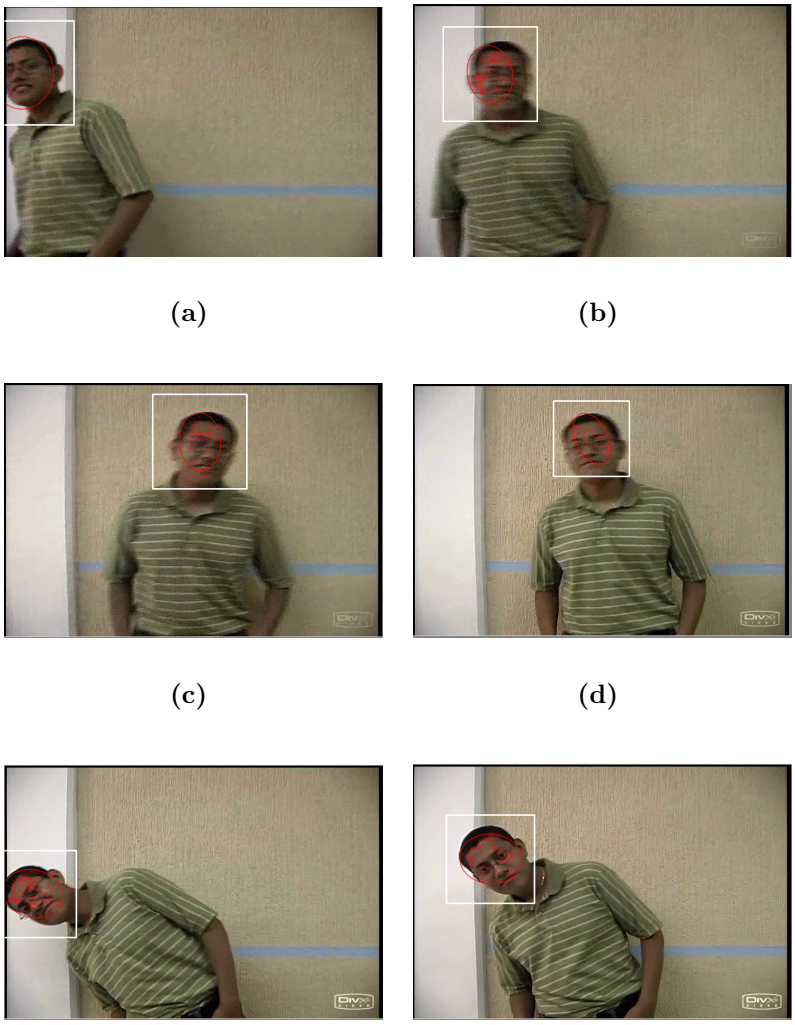


(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.2: Seguimiento sin agregar los datos de tonos de piel de las últimas 7 grabaciones, utilizando el *blob* completo

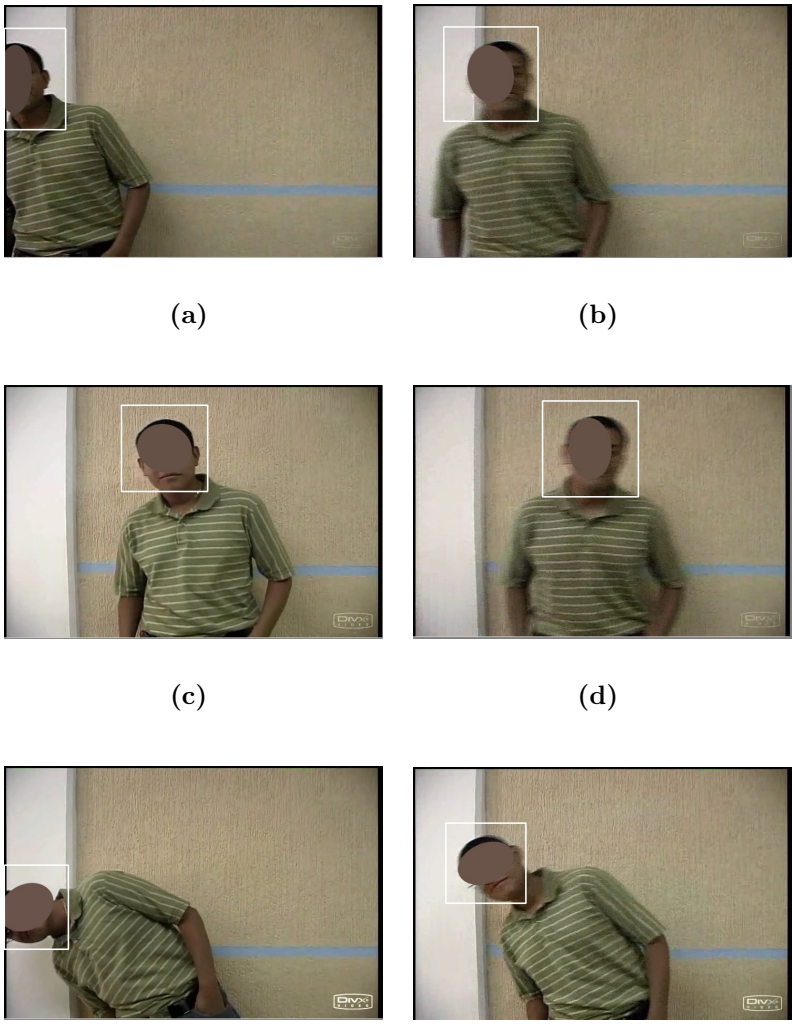


(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.3: Seguimiento después de agregar los datos de tonos de piel de las últimas 7 grabaciones, utilizando el *blob* con píxeles

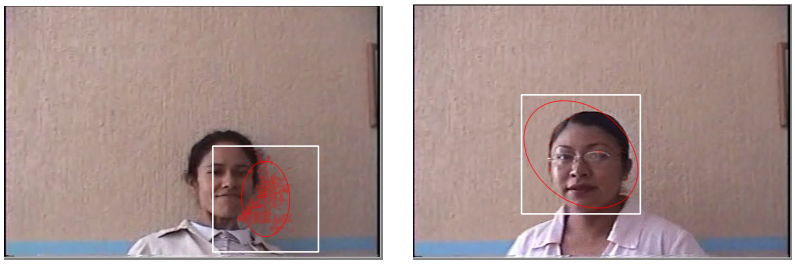


(a) **(b)**

(c) **(d)**

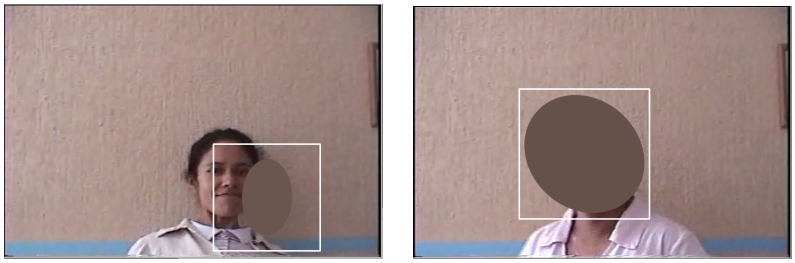
(e) **(f)**

Figura 4.4: Seguimiento después de agregar los datos de tonos de piel de las últimas 7 grabaciones, utilizando el *blob* completo



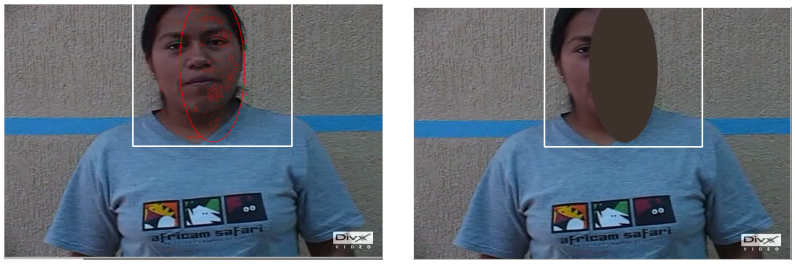
(a) **(b)**

Figura 4.5: Seguimiento con la cámara B mostrando el *blob* mal dibujado, utilizando el *blob* con píxeles



(a) **(b)**

Figura 4.6: Seguimiento con la cámara B mostrando el *blob* mal dibujado, utilizando el *blob* completo



(a) **(b)**

Figura 4.7: Seguimiento con la cámara C

De acuerdo a los resultados mostrados en esta etapa de pruebas, se comprobó que el sistema de seguimiento es dependiente de la calidad del video, ya que se puede notar que debido a la resolución de cada una de las cámaras, la mejor detección de los píxeles de color de piel se presentó en la cámara A. A mayor resolución y mejor calidad del video se obtienen mejores resultados al momento de detectar los píxeles de color de piel, ya que el sistema analiza la información espacial y de color.

Por otro lado, la velocidad en fps de la cámaras también podría afectar el desempeño del sistema, ya que éste analiza 21 fps, esta velocidad es cercana a la velocidad de captura de imágenes de la cámara A. Sin embargo, para la otras cámaras (B y C) su velocidad de captura es menor, lo cual podría propiciar problemas con la aplicación que analiza los frames a una velocidad mayor, especialmente si la persona realiza movimientos rápidos, ya que el sistema supone que entre un frame analizado y el siguiente no hay cambios significativos respecto a la posición del área en movimiento.

4.2. Pruebas con distintos tipos de movimientos de las personas

De acuerdo a los resultados obtenidos en la sección anterior, se decidió probar el sistema con las grabaciones obtenidas con la cámara A, estas grabaciones corresponden a los videos de 32 personas que colaboraron con este proyecto. En estas grabaciones, se trató de realizar el seguimiento a distintos tipos de movimientos de las personas. Los resultados de las pruebas realizadas se describen a continuación, en cada caso se muestran las dos formas de representación del *blob*:

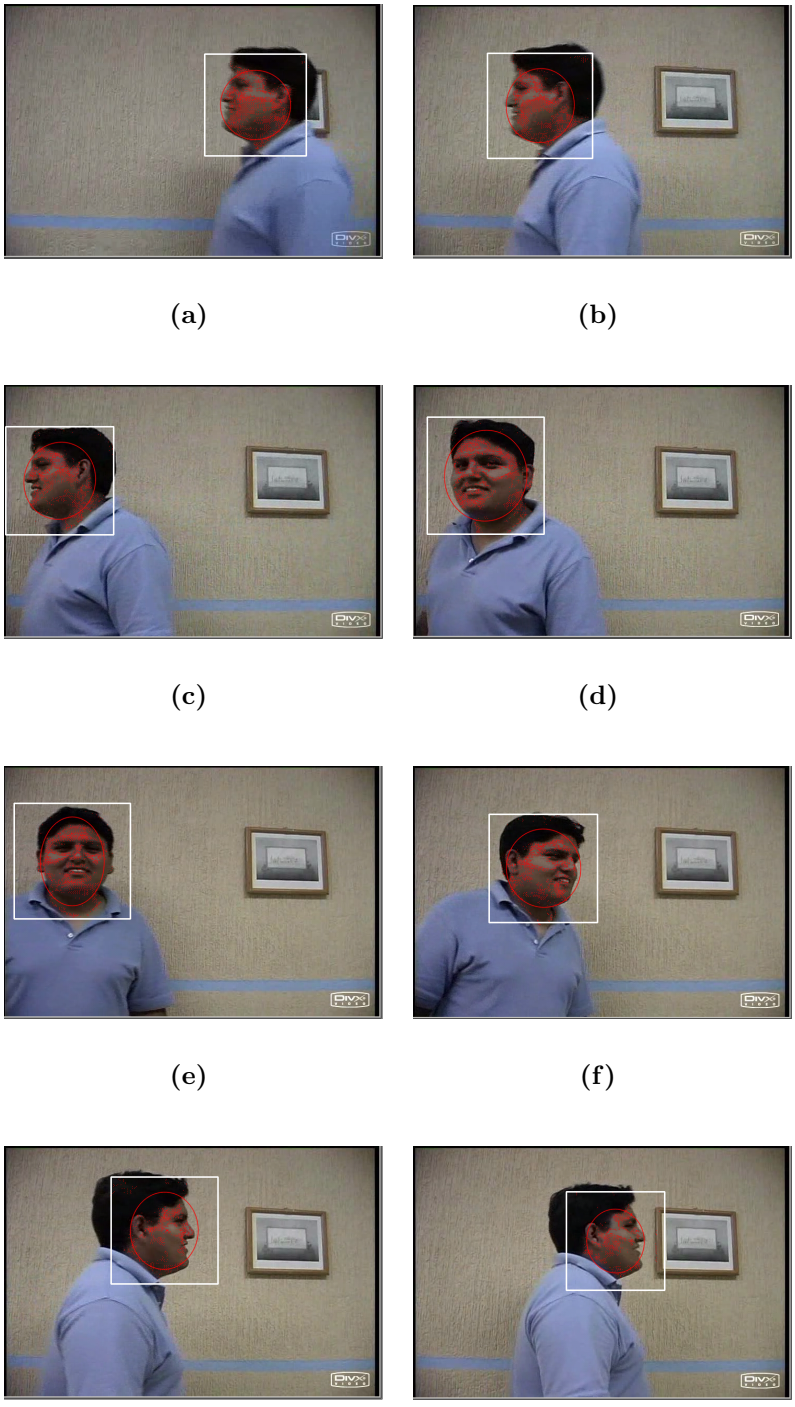
- **Seguimiento con movimientos simples:** Para este tipo de movimientos se tomó en cuenta que las personas caminaran de frente y de lado, realizaran giros,

además de moverse hacia abajo y hacia arriba. Los resultados obtenidos con este tipo de pruebas fueron adecuados, ya que el sistema reconoció en gran medida a los puntos que pertenecen a la cabeza, por lo mismo el *blob* se dibujó de manera apropiada. Este tipo de seguimiento se puede observar en los siguientes ejemplos: Las Figuras 4.8 y 4.9 muestran el seguimiento al movimiento de desplazamiento lateral. Las Figuras 4.10 y 4.11 presentan el seguimiento de movimientos de arriba hacia abajo y movimientos laterales de la cabeza. Las Figuras 4.12 y 4.13 exhiben el seguimiento cuando la persona realiza un giro completo sobre su propio eje.

- **Seguimiento con recuperación de oclusiones:** Este tipo de movimientos consistió en que las personas realizaran oclusión con las manos, es decir, en pasar sus manos por encima del rostro con el propósito de poner a prueba el sistema y que éste confundiera el color de piel del rostro con el de las manos. Cuando las manos hacian la oclusión el *blob* se dibujaba de manera incorrecta, esto debido a que se confundía con el color de piel, sin embargo al retirar las manos se observó la recuperación a la oclusión al seguir nuevamente a la cabeza de la persona. Este tipo de movimientos se pueden observar en las Figuras 4.14, 4.15, 4.16 y 4.17.

- **Seguimiento a diferentes orientaciones de la cabeza:** Este experimento consistió en que la persona realizara diferentes movimientos de la cabeza con el propósito de confirmar que el *blob* se dibujara con respecto a la orientación de ésta. La elipse que representa el *blob* debe tener un ángulo de rotación de acuerdo a la orientación de la cabeza. Los resultados se aprecian en las Figuras 4.18, 4.19, 4.20 y 4.21, donde el *blob* se dibujó de manera correcta además de seguir a la persona.

- **Seguimiento con otra persona involucrada:** Aunque ésta es una limitación del sistema se pudo comprobar su estabilidad al recuperarse de la oclusión que



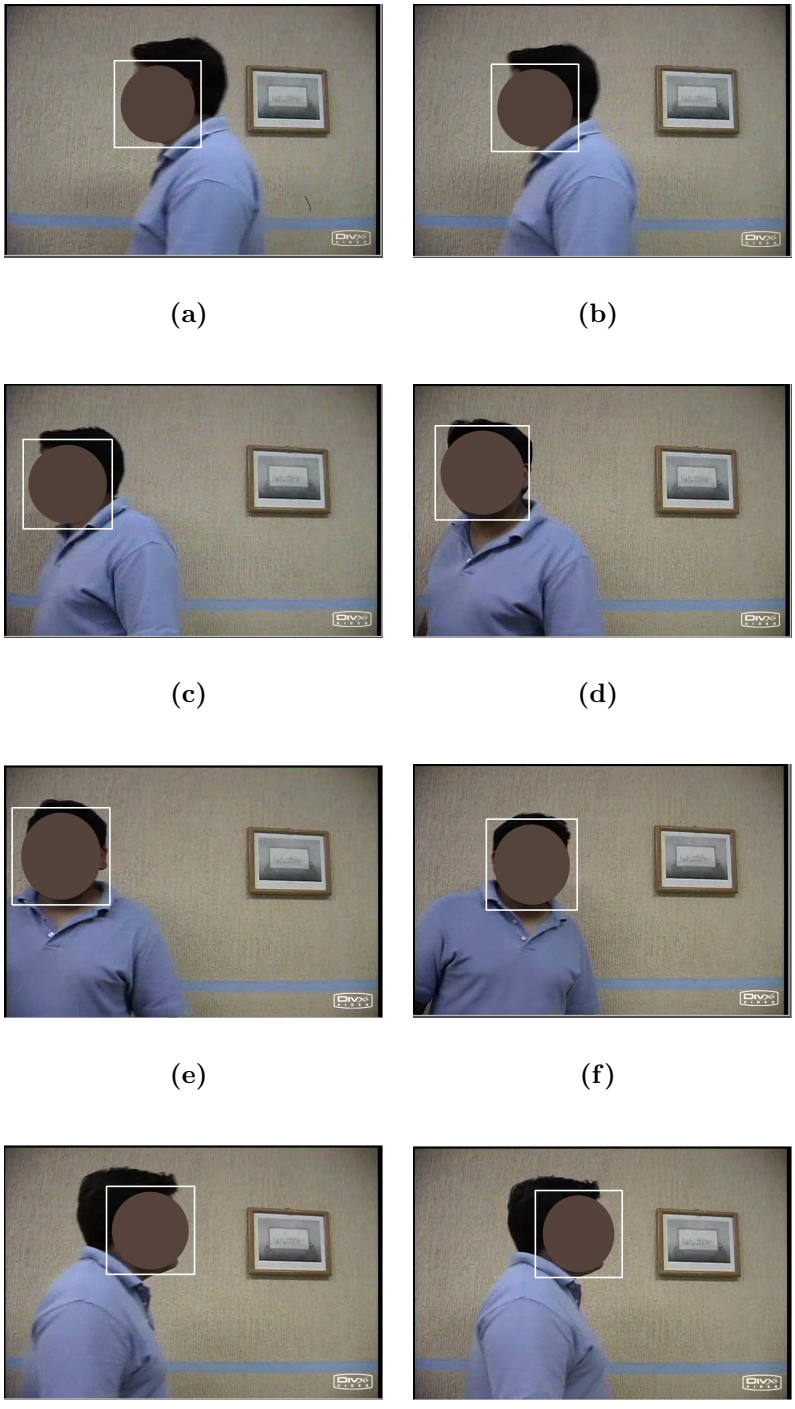
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.8: Seguimiento para el movimiento simple de desplazamiento lateral mostrando el *blob* con píxeles



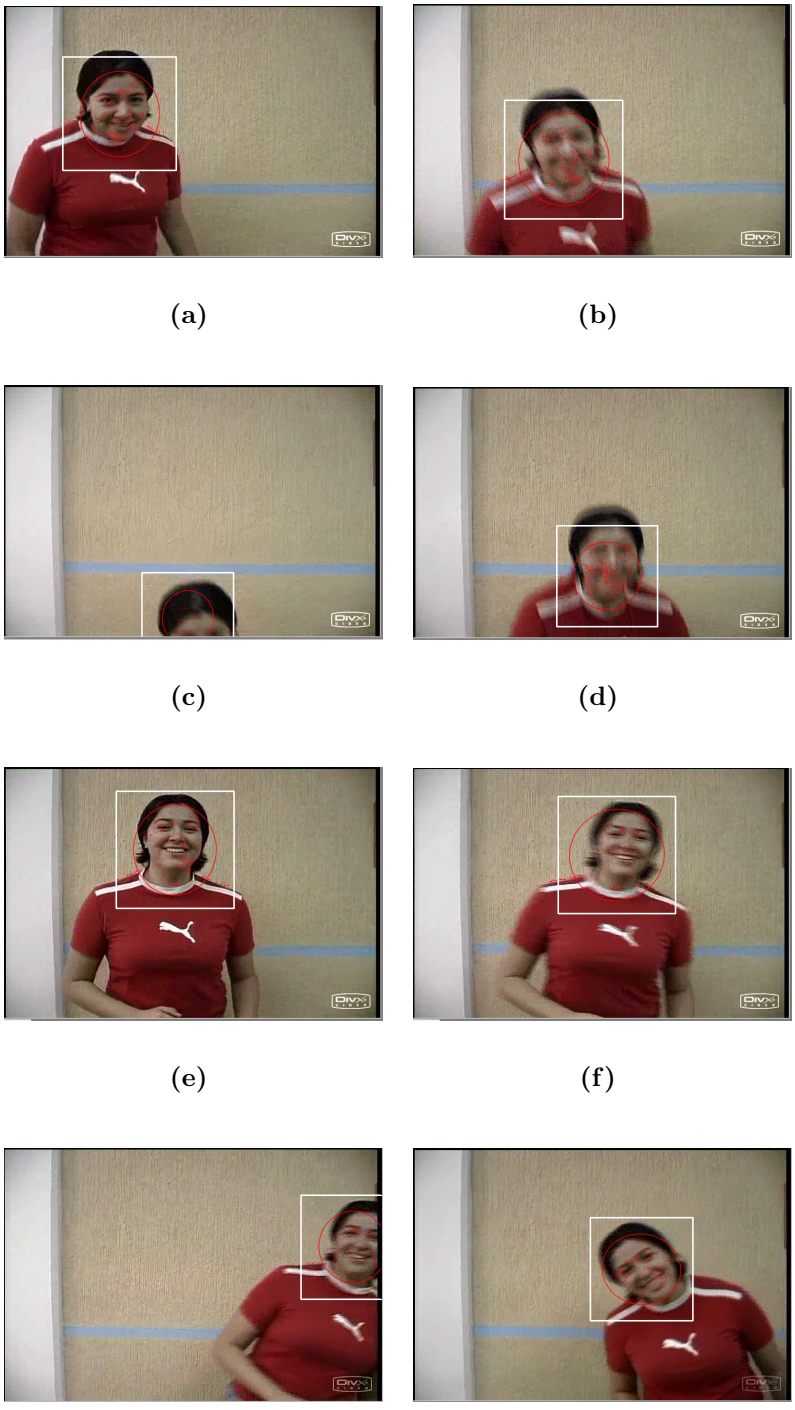
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.9: Seguimiento para el movimiento simple de desplazamiento lateral mostrando el *blob* completo



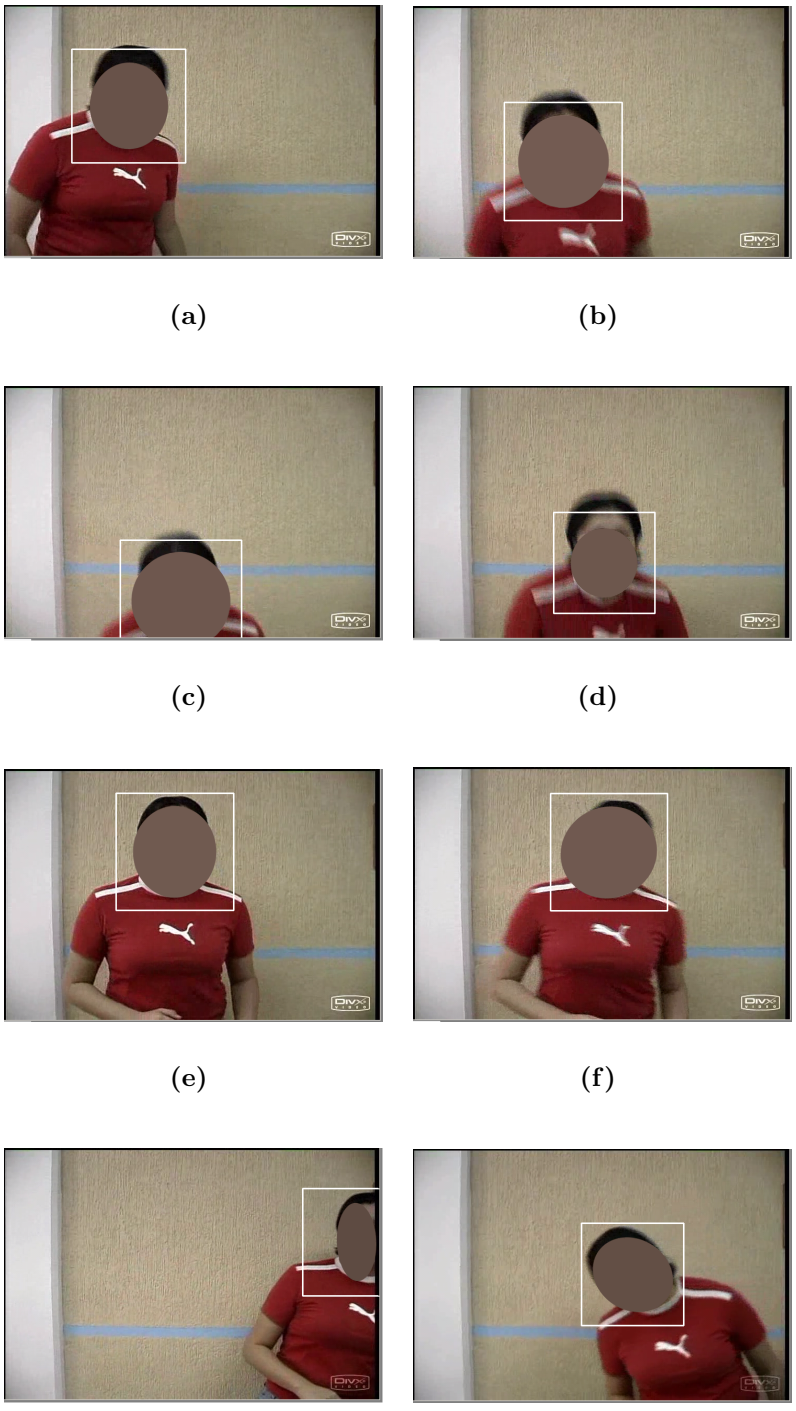
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.10: Seguimiento para el desplazamiento hacia abajo y movimientos laterales mostrando el *blob* con píxeles



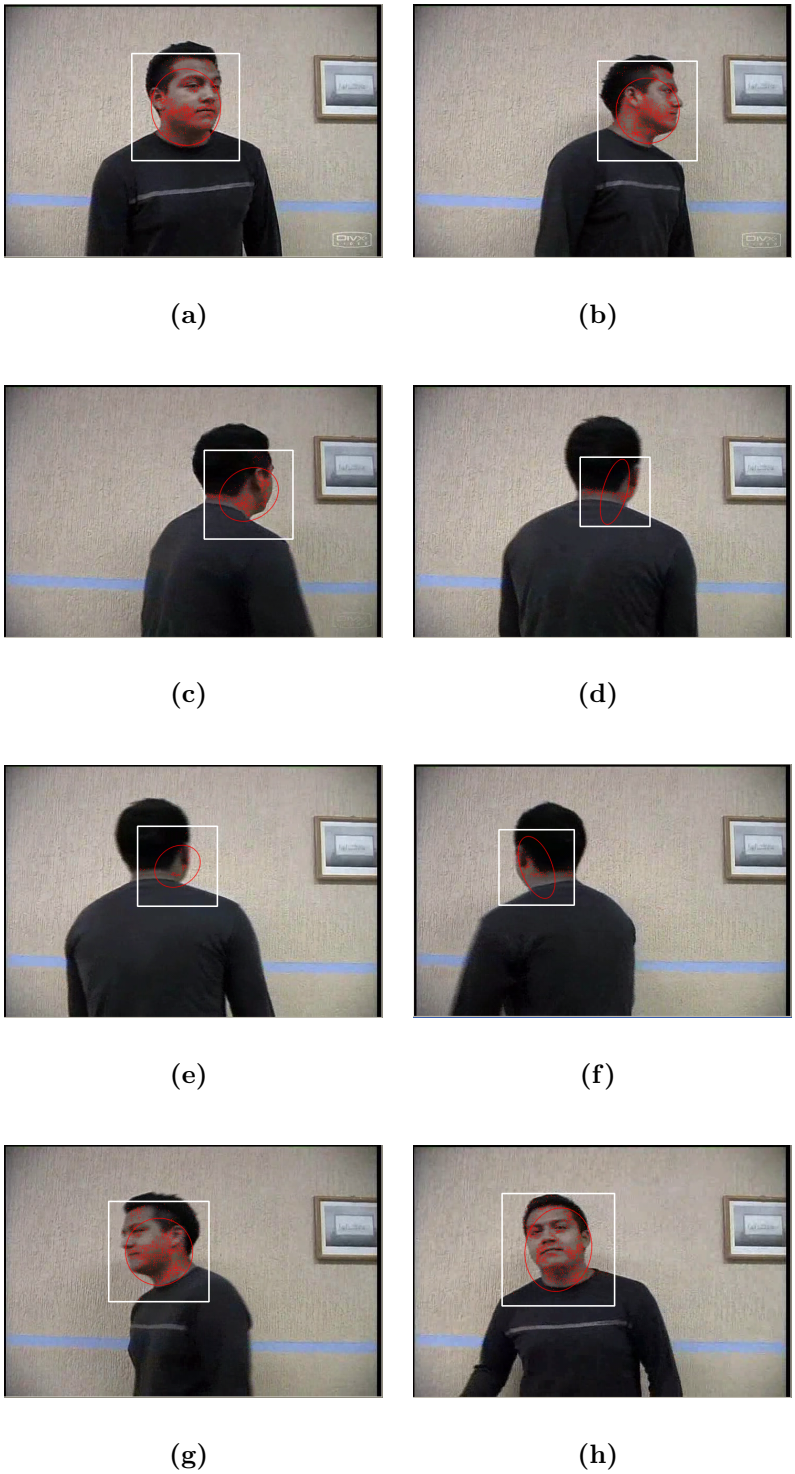
(a) **(b)**

(c) **(d)**

(e) **(f)**

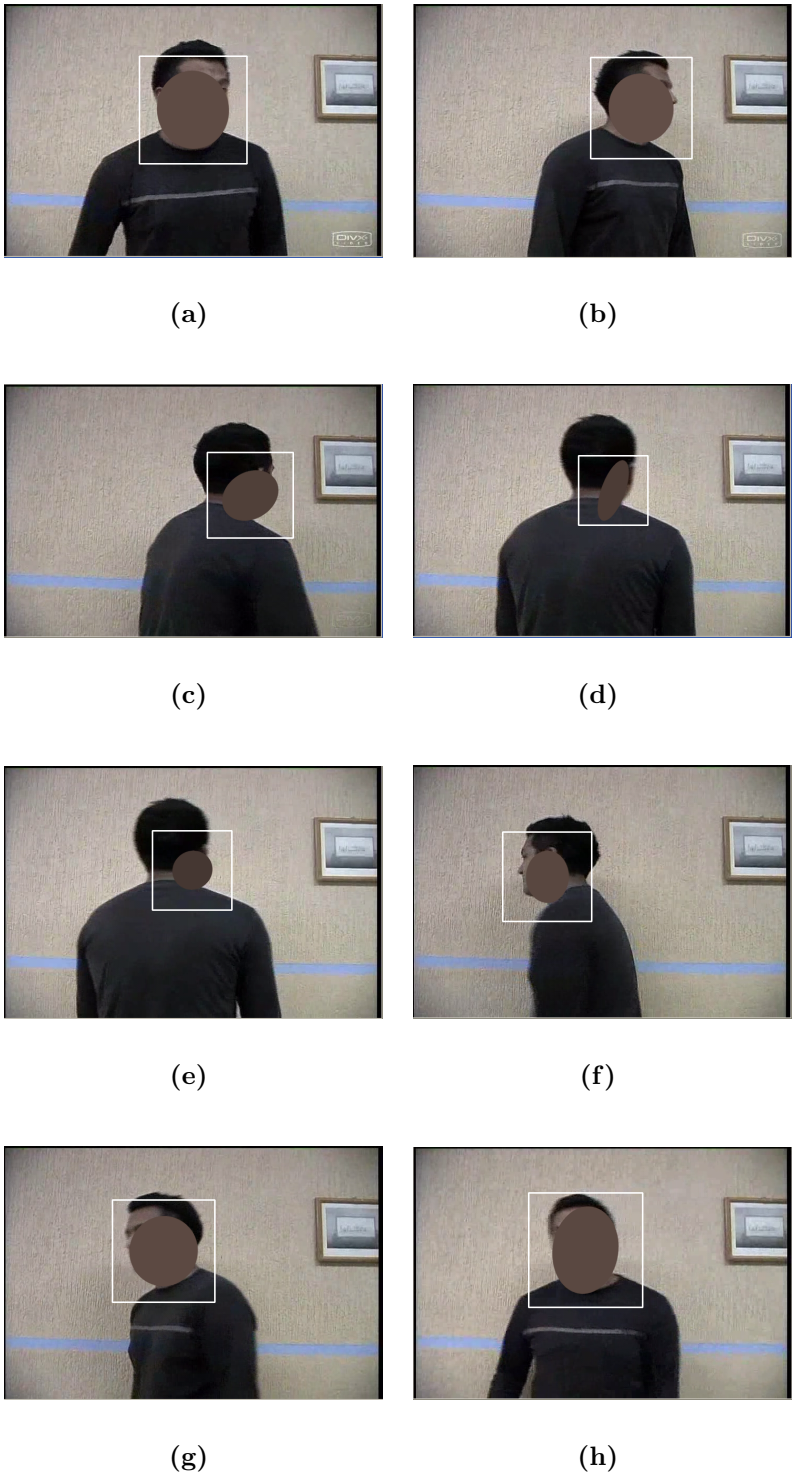
(g) **(h)**

Figura 4.11: Seguimiento para el desplazamiento hacia abajo y movimientos laterales mostrando el *blob* completo



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)**

Figura 4.12: Seguimiento para el desplazamiento de giro completo mostrando el *blob* con píxeles



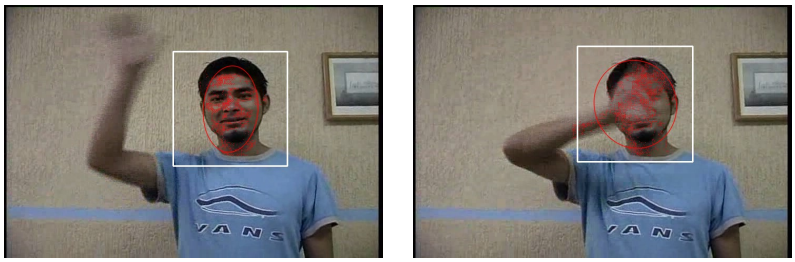
(a) **(b)**

(c) **(d)**

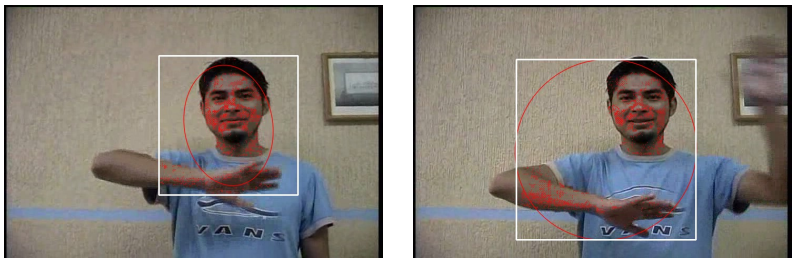
(e) **(f)**

(g) **(h)**

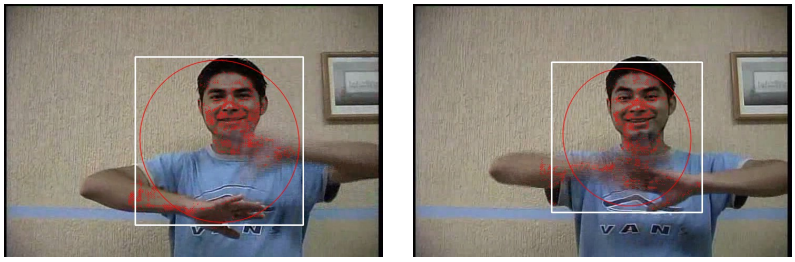
Figura 4.13: Seguimiento para el desplazamiento de giro completo mostrando mostrando el *blob* completo



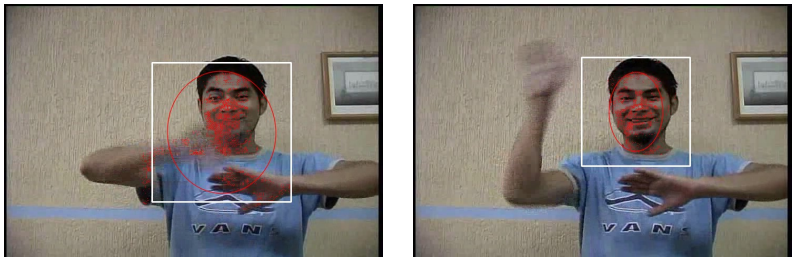
(a) **(b)**



(c) **(d)**

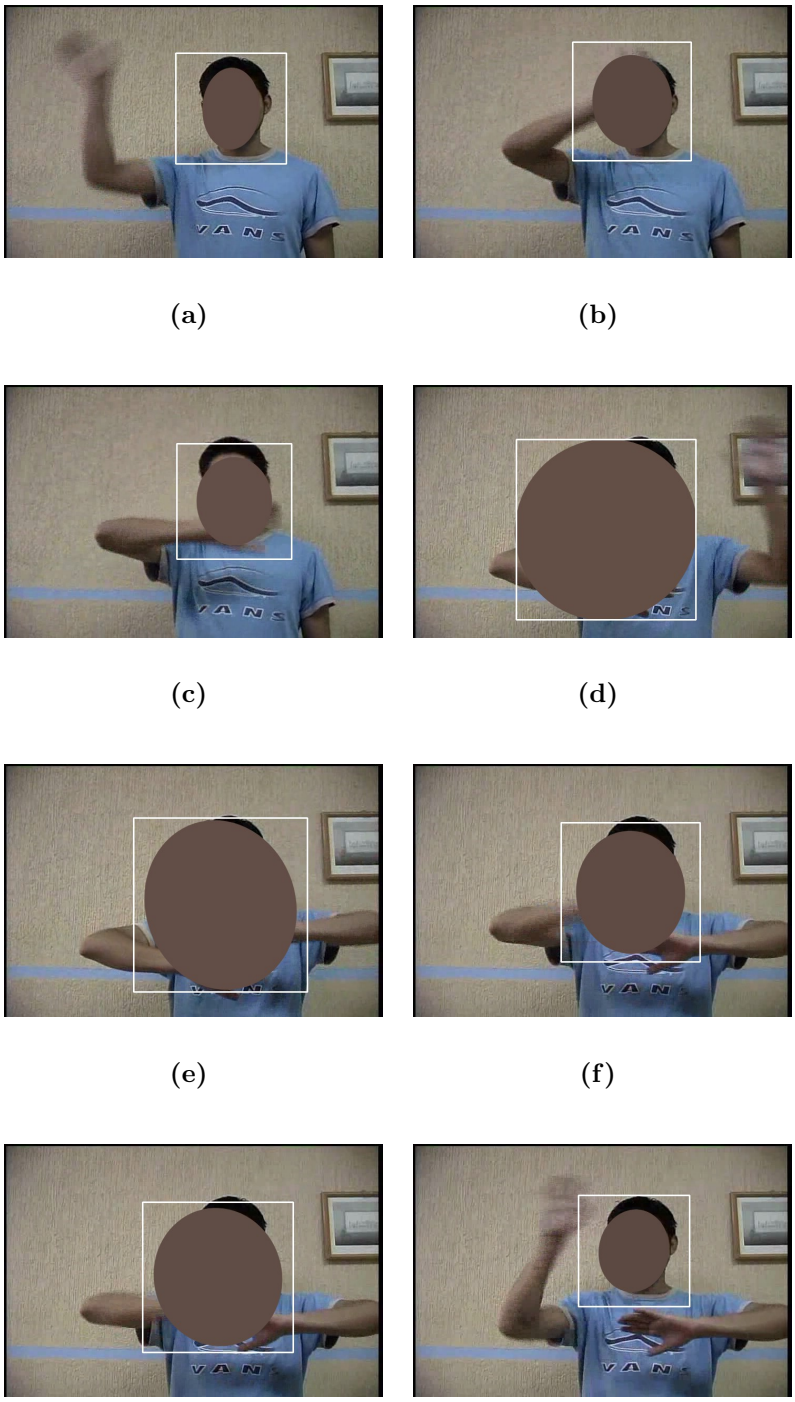


(e) **(f)**



(g) **(h)**

Figura 4.14: Seguimiento con recuperación de oclusiones mostrando el *blob* con píxeles



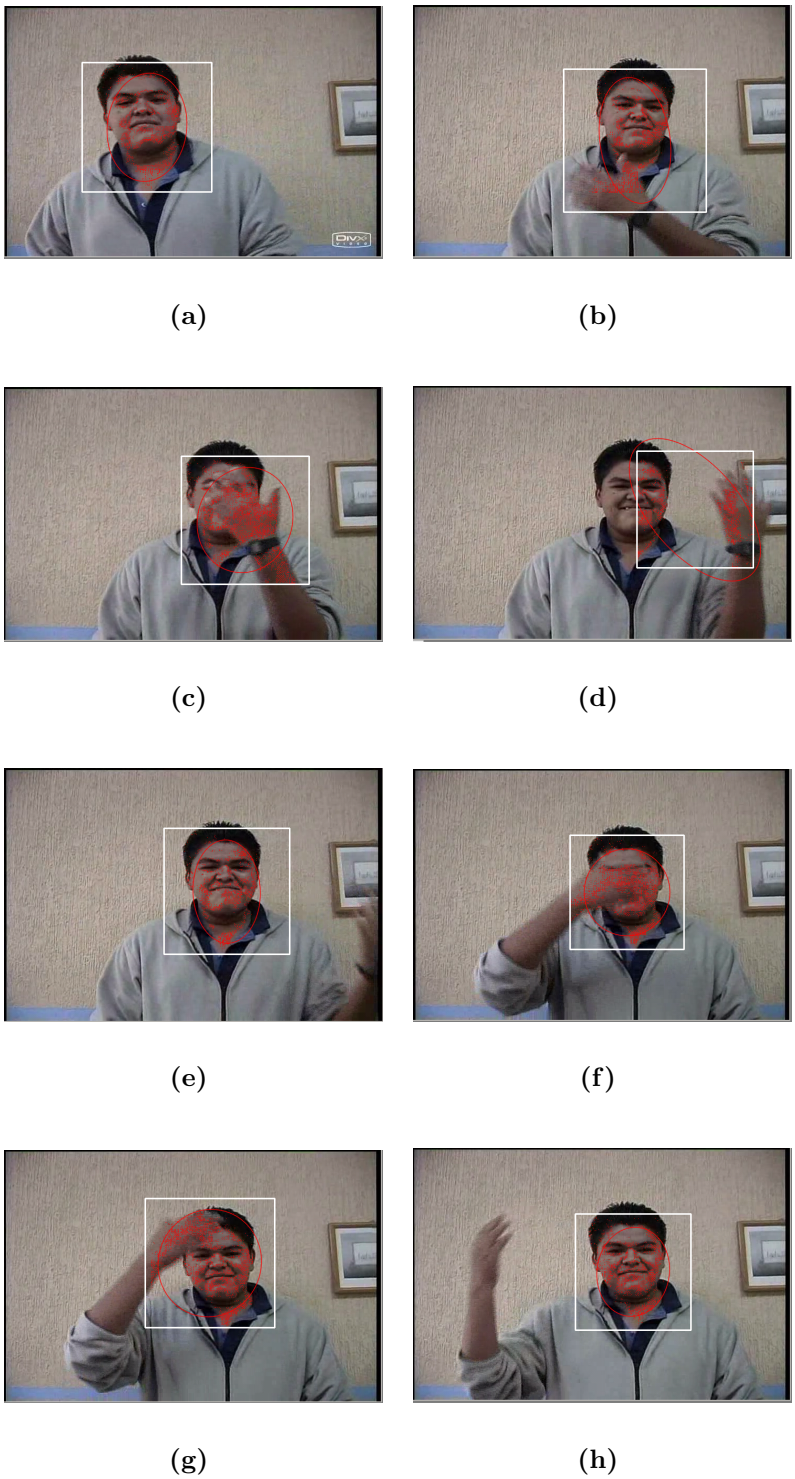
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.15: Seguimiento con recuperación de oclusiones mostrando el *blob* completo



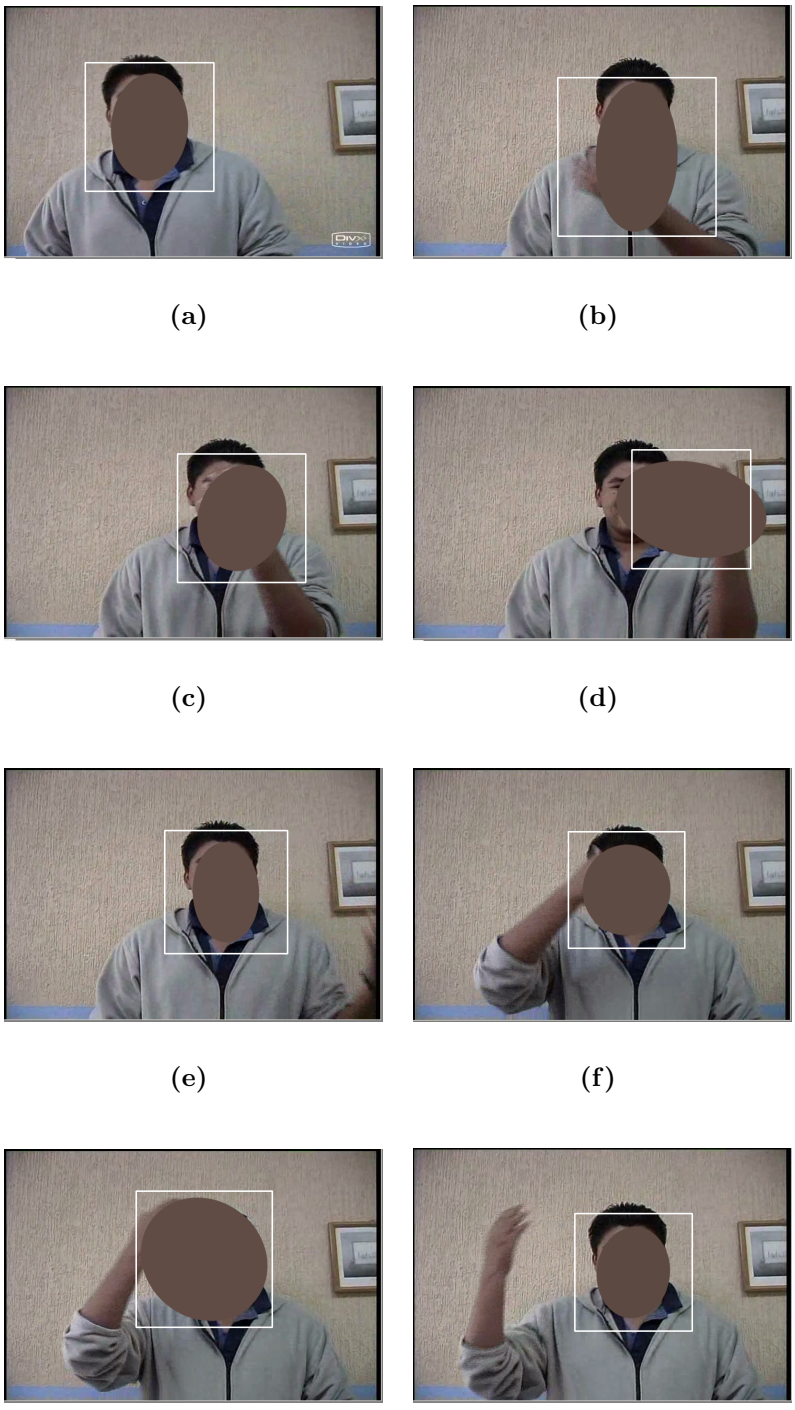
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.16: Seguimiento con recuperación de oclusiones mostrando el *blob* con píxeles



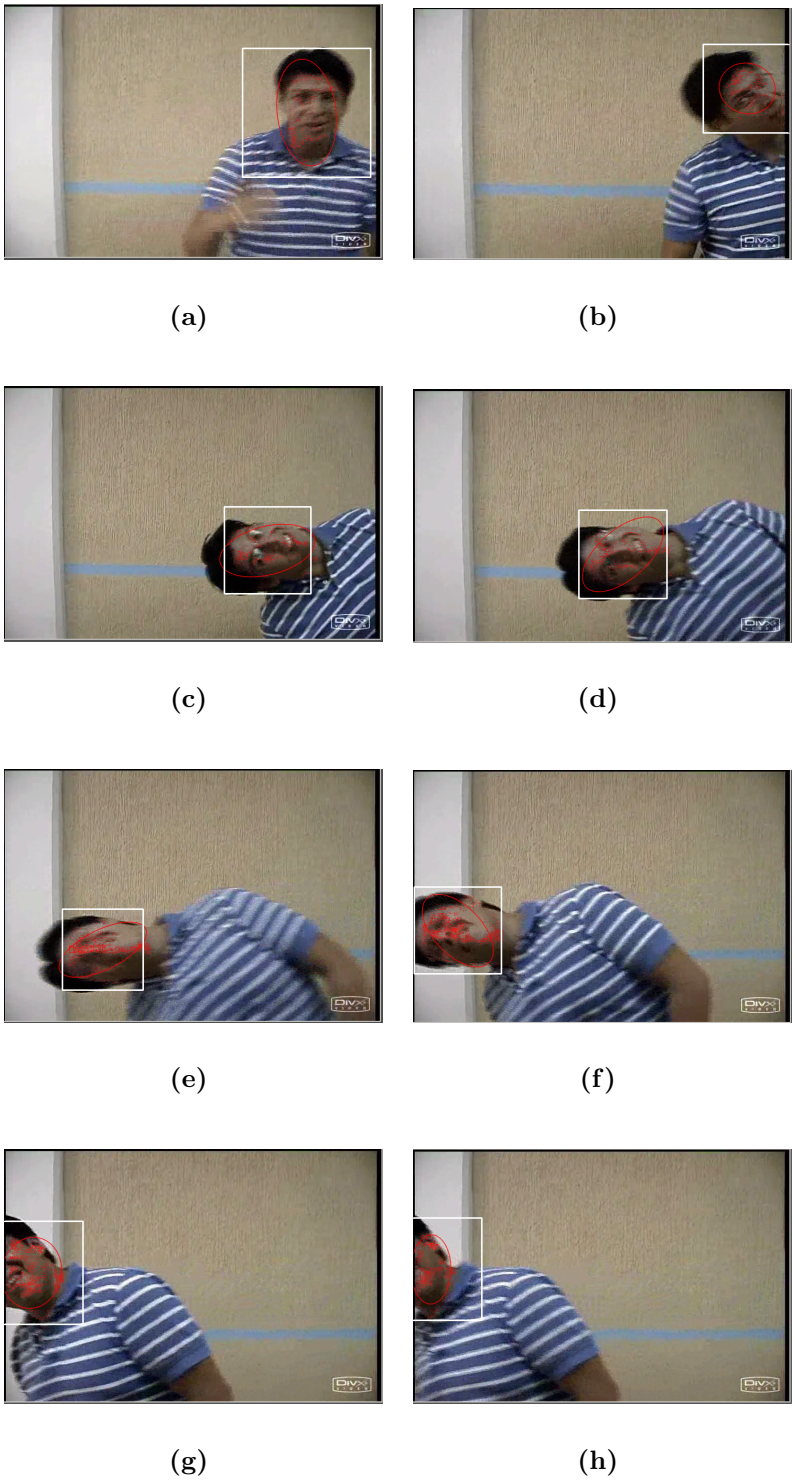
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.17: Seguimiento con recuperación de oclusiones mostrando el *blob* completo



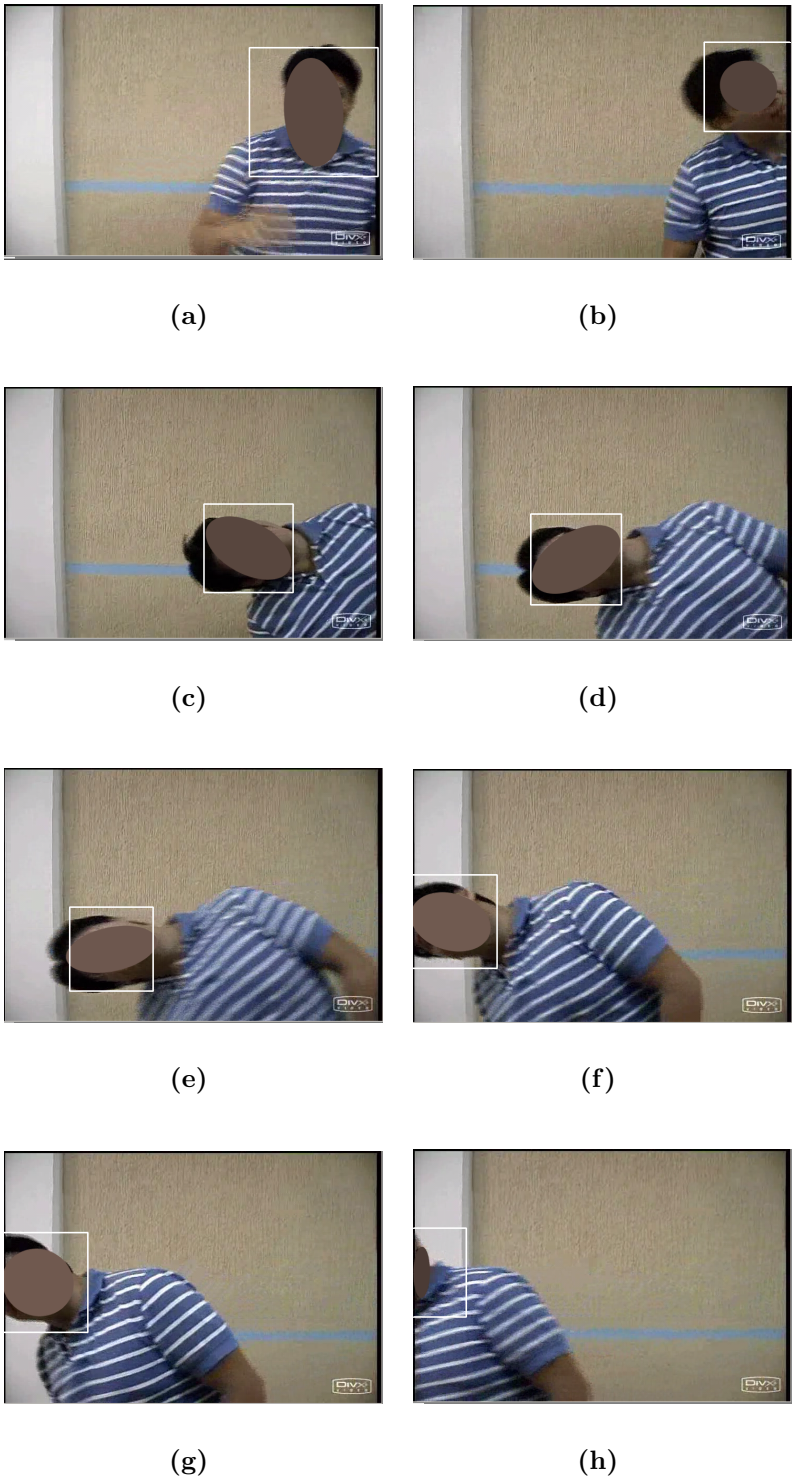
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.18: Seguimiento variando la orientación de la cabeza mostrando el *blob* con píxeles



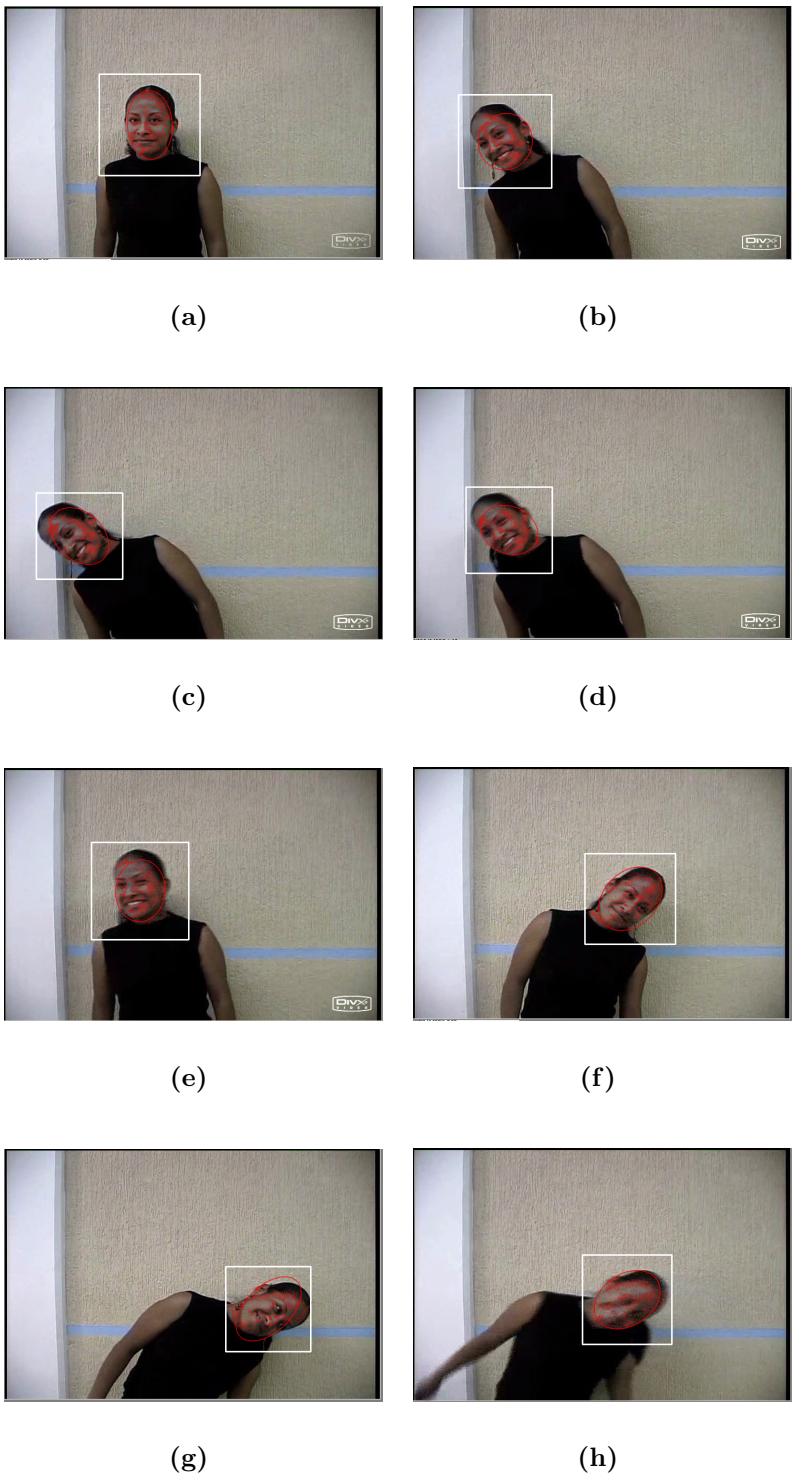
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.19: Seguimiento variando la orientación de la cabeza mostrando el *blob* completo



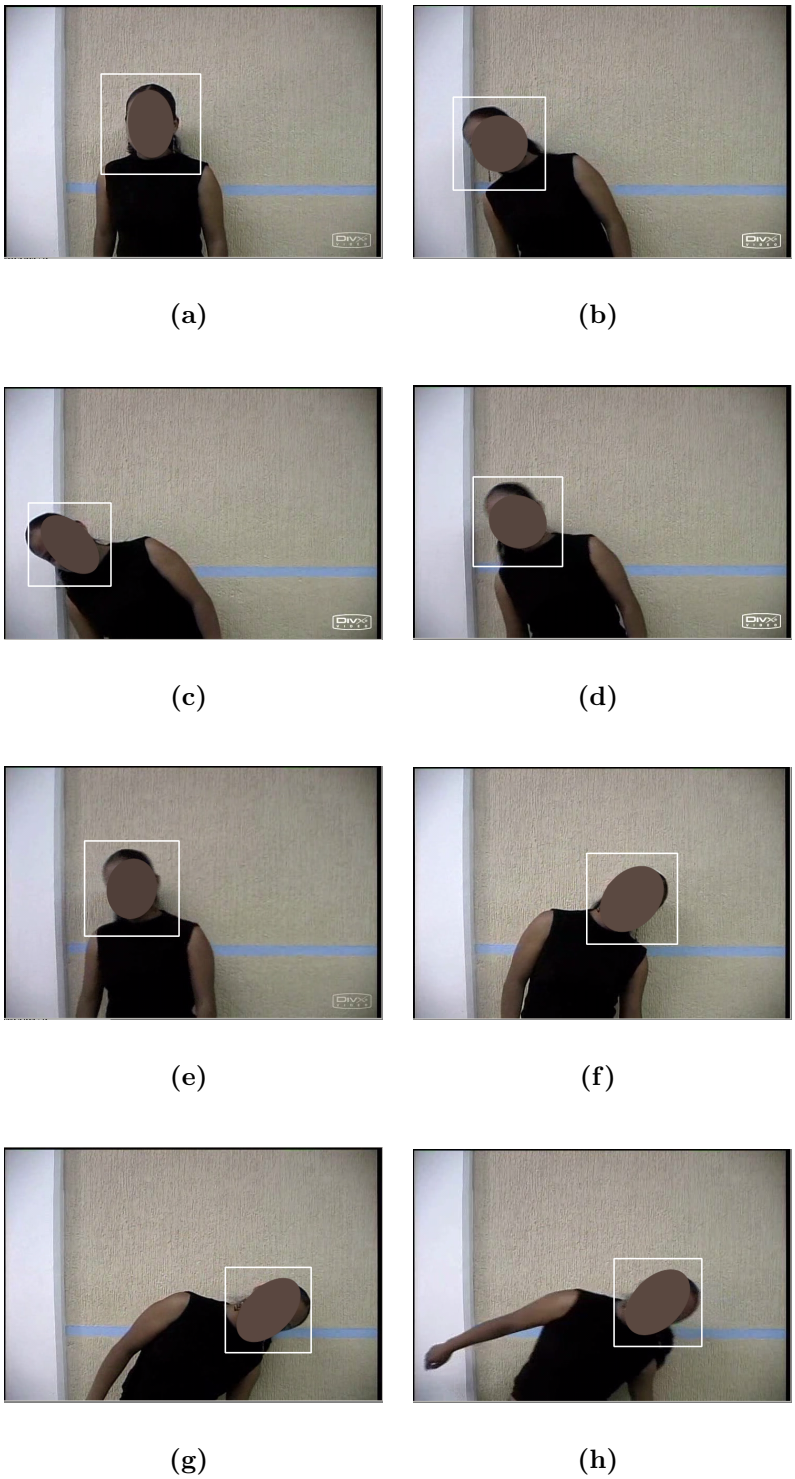
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.20: Seguimiento variando la orientación de la cabeza mostrando el *blob* con píxeles



(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

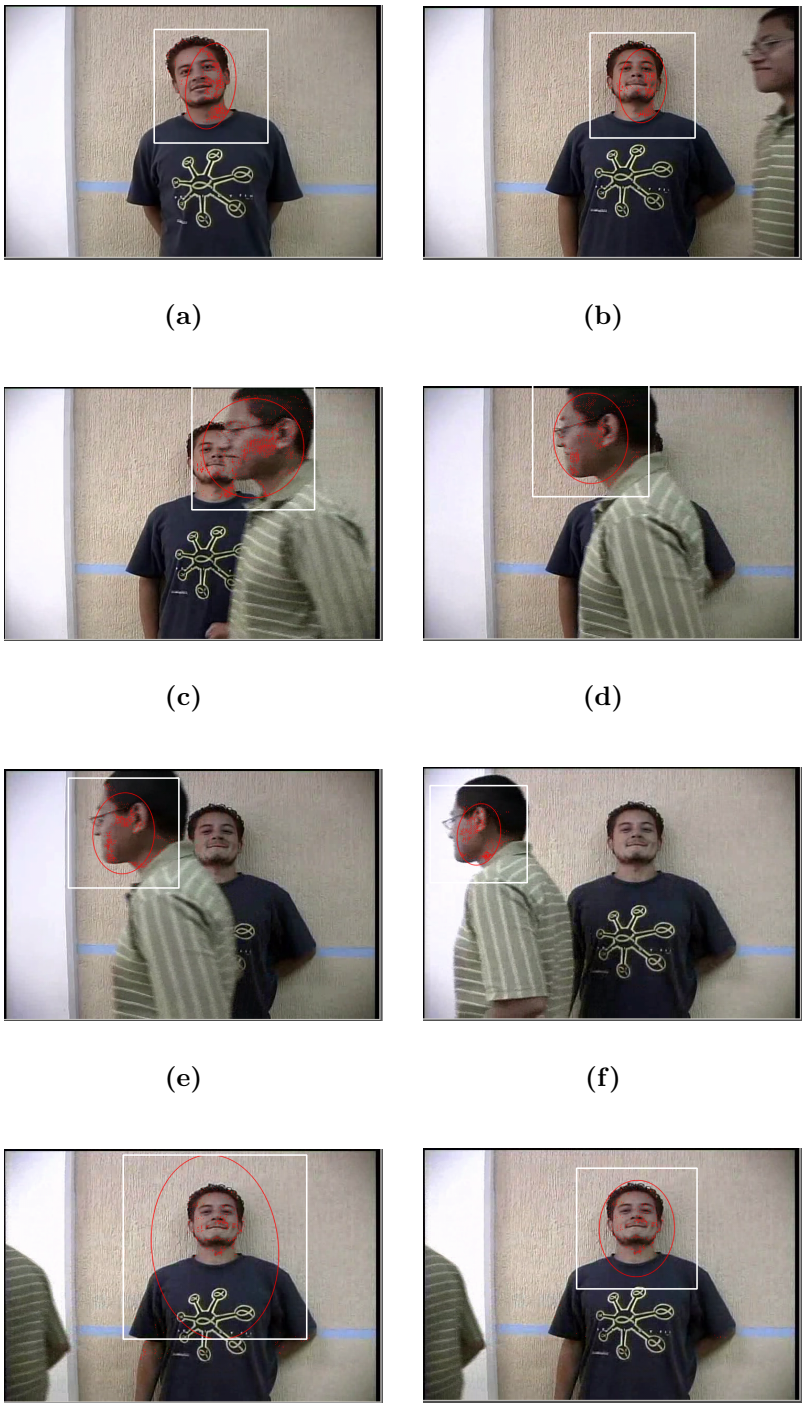
Figura 4.21: Seguimiento variando la orientación de la cabeza mostrando el *blob* completo

hace una segunda persona al introducirse. Primero se inicia el seguimiento de una persona en la escena, y posteriormente se introduce a otra persona, la cual trata de hacer oclusión con la primera. Los resultados de esta prueba se observan en las Figuras 4.22 y 4.23, en las Figuras 4.22(a) y 4.23(a) se aprecia el seguimiento de la primer persona, en las Figuras 4.22(b), 4.23(b), 4.22(c), 4.23(c), 4.22(d), 4.23(d), 4.22(e), 4.23(e), 4.22(f) y 4.23(f) puede verse que al introducir a la segunda persona, el sistema se confunde y la sigue, sin embargo, cuando ésta abandona la escena el *blob* se recupera de manera satisfactoria siguiendo nuevamente a la primer persona, tal y como se ve en las Figuras 4.22(g), 4.23(g), 4.22(h) y 4.23(h).

- **Seguimiento con diferentes tamaños de la ventana de búsqueda:** Como se mencionó en la sección 3.2.3 fue necesario variar el tamaño de la ventana de búsqueda para mantener la estabilidad del sistema cuando el número de píxeles detectados como pertenecientes a la cabeza es menor a 45. El funcionamiento de este proceso se puede observar con el siguiente experimento: La persona sale de la escena por el lado izquierdo y regresa por el otro extremo. La ventana de búsqueda al no encontrar a una persona a quien seguir, aumenta su tamaño hasta encontrar a la persona. Los resultados se muestran en las Figuras 4.24 y 4.25 donde de manera satisfactoria el *blob* sigue nuevamente a la persona cuando ésta entra en la escena. Otro ejemplo de este tipo se seguimiento, donde la persona sale de la escena y retorna por el mismo lado, se presenta en las Figuras 4.26 y 4.27.

Observaciones Adicionales

Para un mejor desempeño del sistema, se deben tomar en cuenta ciertos factores que pueden alterarlo. La siguiente lista describe detalles que se observaron al momento



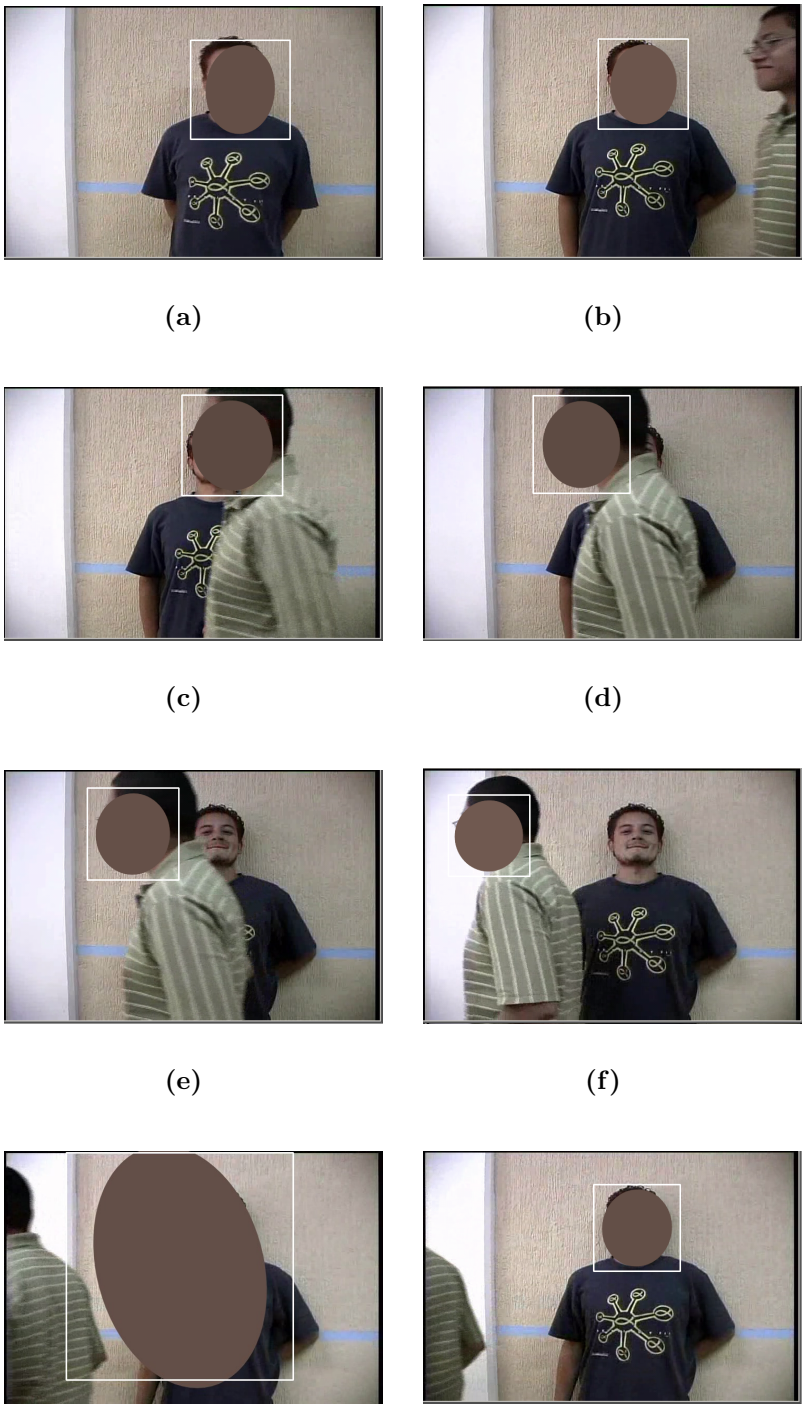
(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.22: Seguimiento con con otra persona involucrada mostrando el *blob* con píxeles



(a) **(b)**

(c) **(d)**

(e) **(f)**

(g) **(h)**

Figura 4.23: Seguimiento con con otra persona involucrada mostrando el *blob* completo

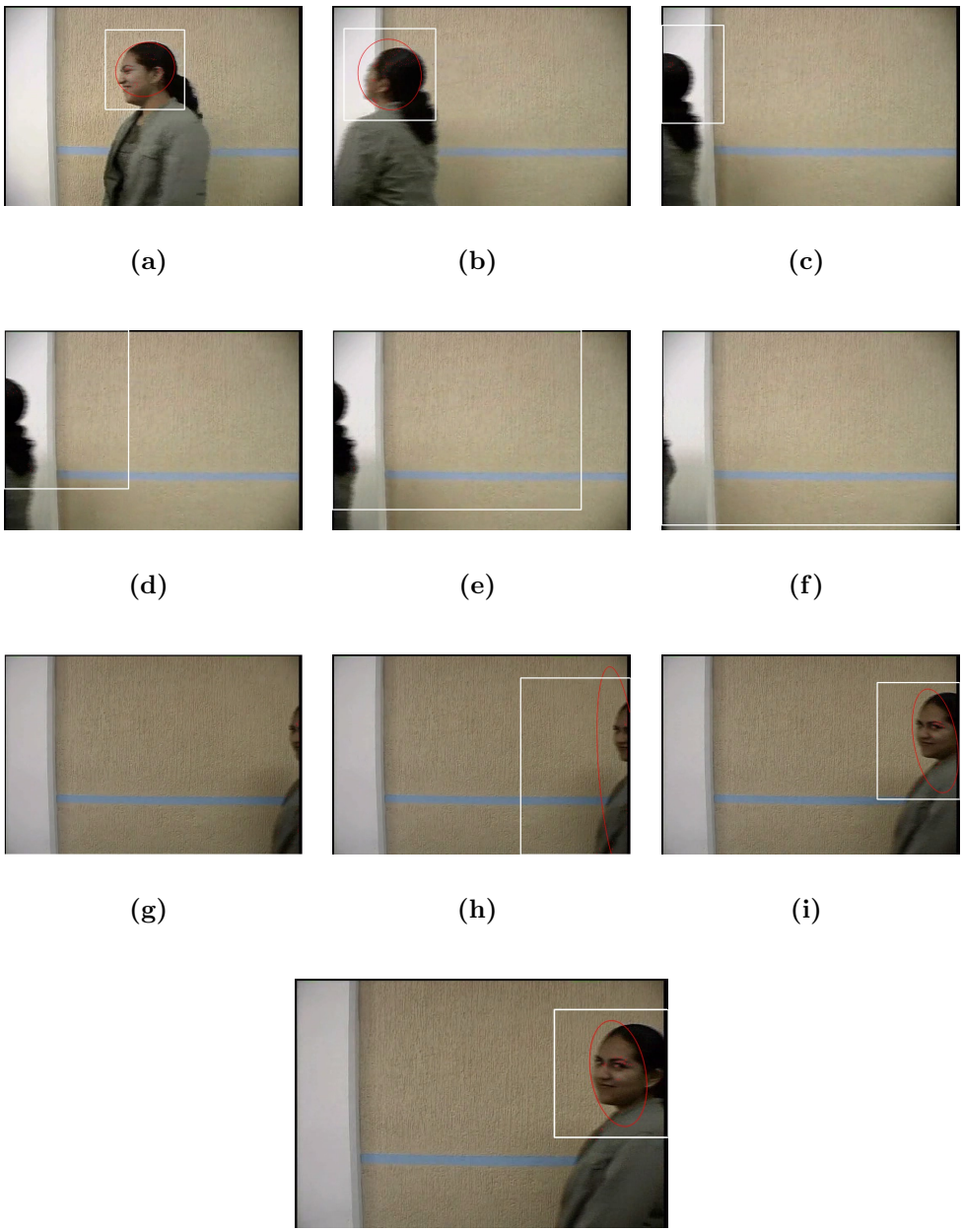


Figura 4.24: Seguimiento donde se aprecia las variaciones en el tamaño de la ventana de búsqueda mostrando el *blob* con píxeles

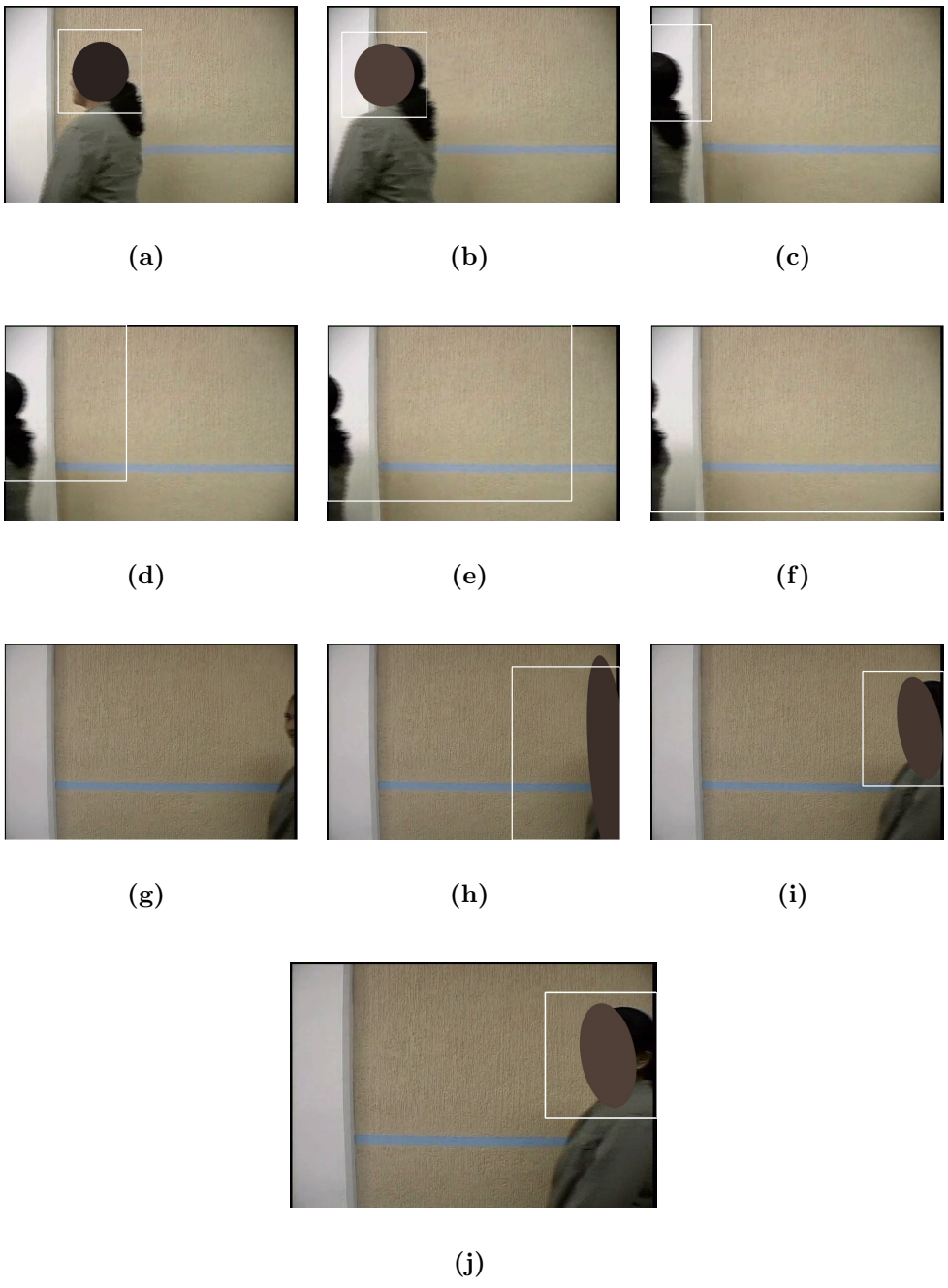
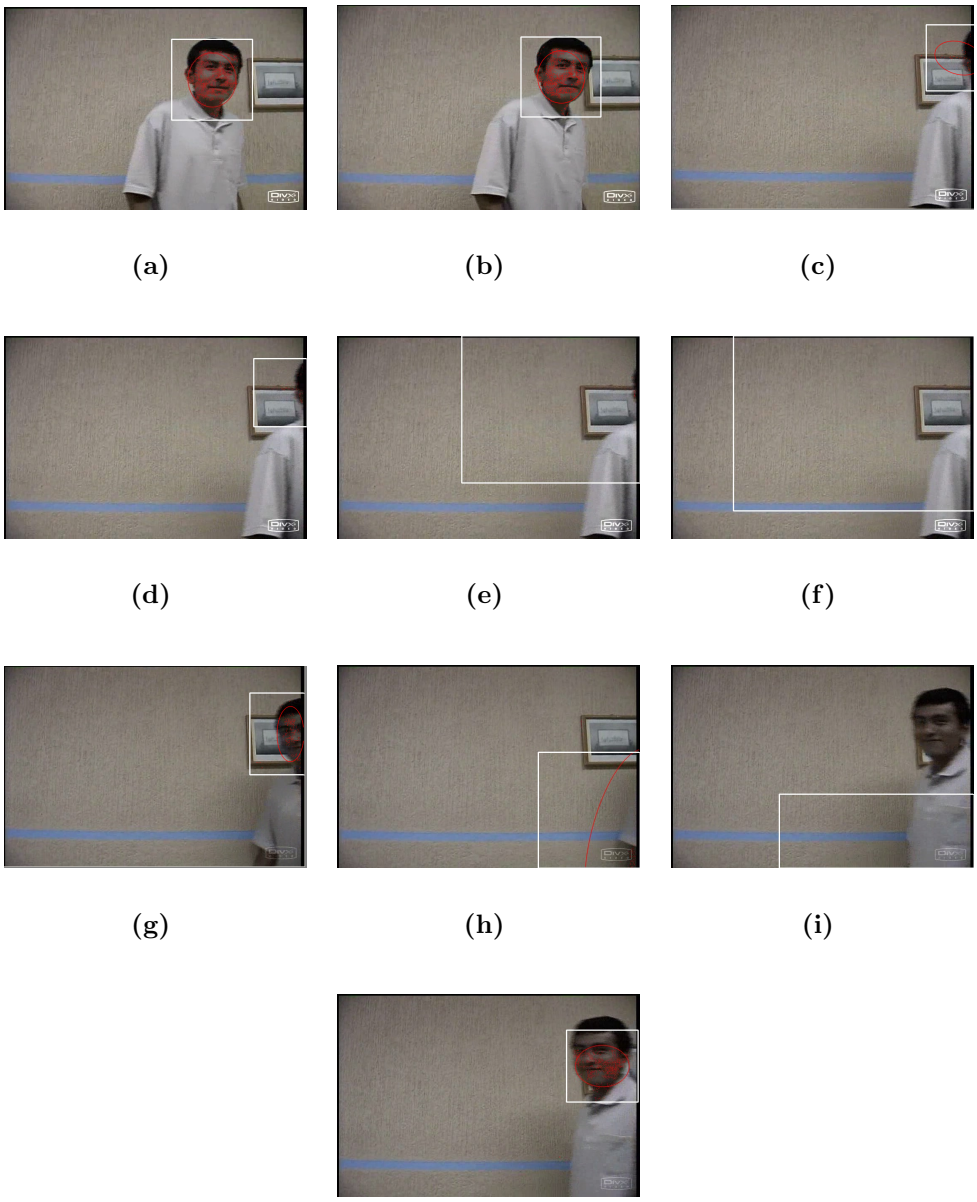


Figura 4.25: Seguimiento donde se aprecia las variaciones en el tamaño de la ventana de búsqueda mostrando el *blob* completo



(a) **(b)** **(c)**

(d) **(e)** **(f)**

(g) **(h)** **(i)**

(j)

Figura 4.26: Seguimiento donde se aprecia las variaciones en el tamaño de la ventana de búsqueda mostrando el *blob* con píxeles

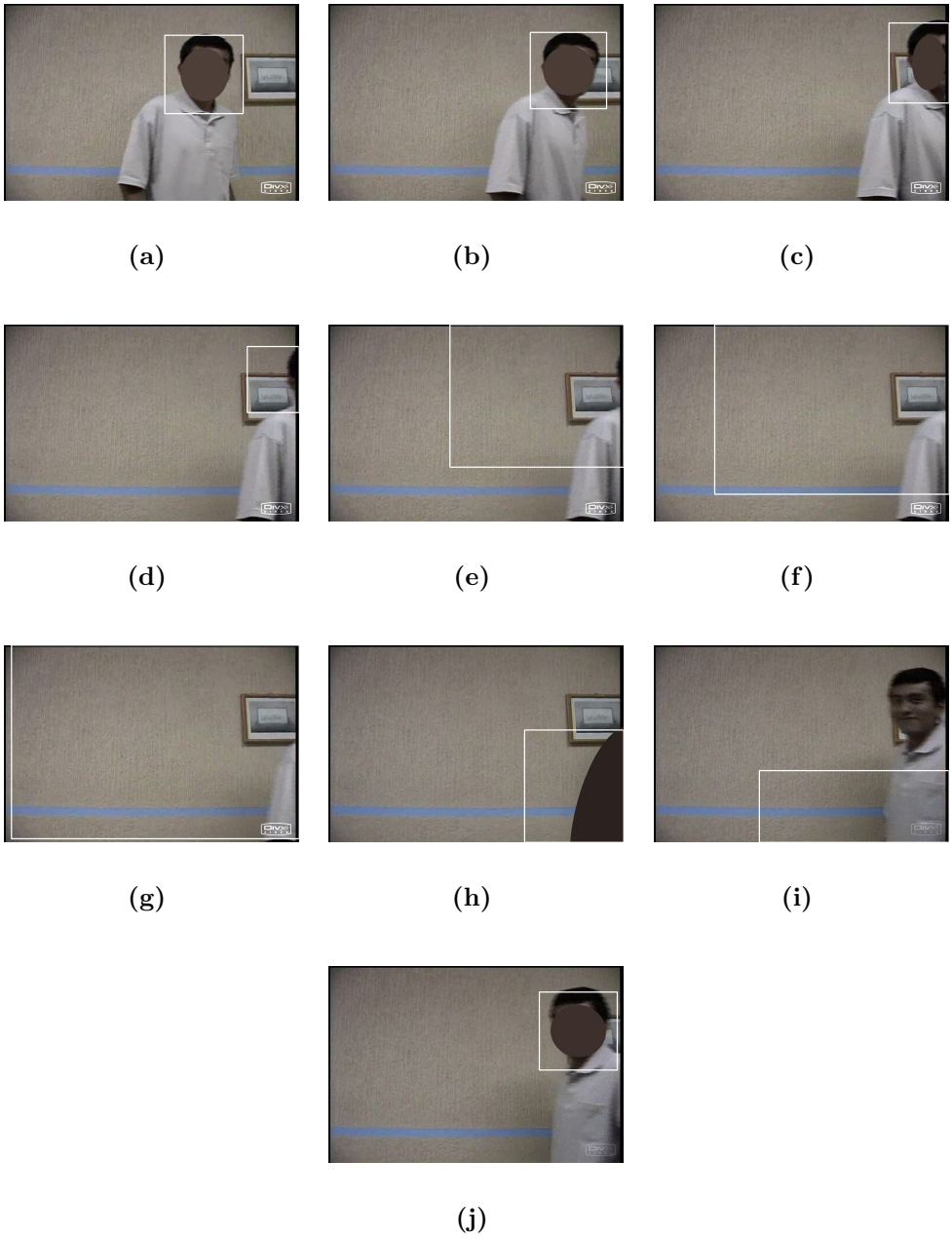
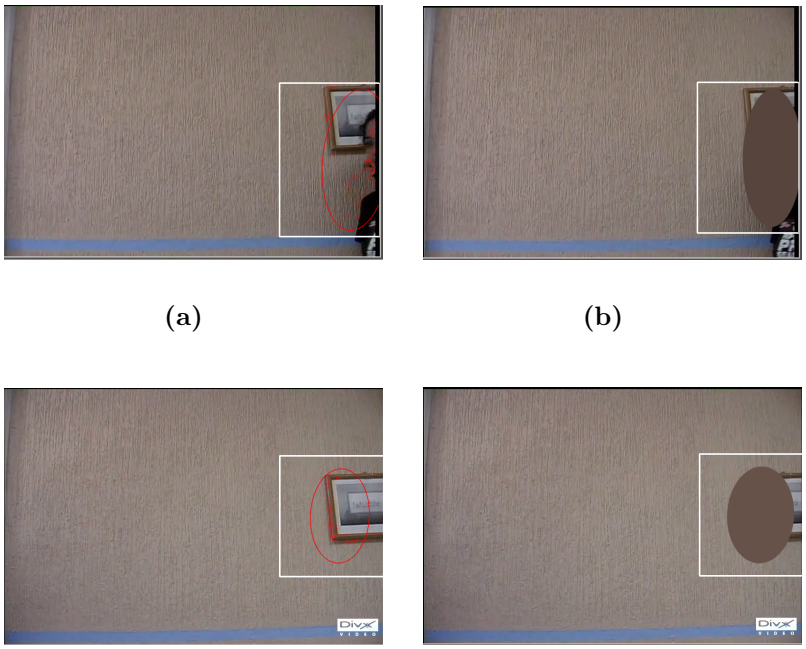


Figura 4.27: Seguimiento donde se aprecia las variaciones en el tamaño de la ventana de búsqueda mostrando el *blob* completo

de realizar las pruebas con los videos grabados. En cada observación se muestra un ejemplo de cómo podría ser afectado el desempeño del sistema:

1. No debe haber distractores que puedan confundirse con el color de piel, esto

representa una limitante del sistema, ya que debido a la salida de la persona de la escena o al cambio de iluminación ciertos distractores pueden confundirse con el color de piel, alterando el aspecto del *blob* o creando el seguimiento de falsos positivos [2] . Un ejemplo de la confusión con un distractor en la escena se muestra en las Figuras 4.28(a) y 4.28(b), donde se observa como se altera el aspecto del *blob*, por otro lado, las Figuras 4.28(c) y 4.28(d) muestran un caso de falsos positivos.



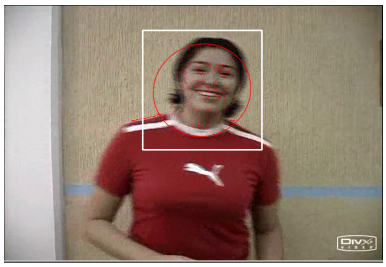
(a) **(b)**

(c) **(d)**

Figura 4.28: Ejemplo de la confusión con un distractor en la escena y de falsos positivos.

2. Los colores rojo y naranja en combinación con la iluminación del ambiente causan falsos positivos. Esto origina una alteración en la forma del *blob*, sin embargo, la información global predomina. Con esto también se prueba la estabilidad del

[2] Los falsos positivos son áreas de la escena que se detectan como la cabeza de una persona cuando no lo son.



(a)

(b)

Figura 4.29: Ejemplo de falsos positivos

(a)

(b)

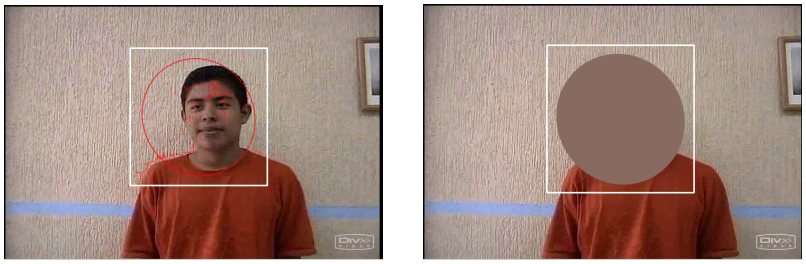
Figura 4.30: Ejemplo de falsos positivos

sistema. Los falsos positivos se aprecian por los píxeles identificados como color de piel que aparecen sobre la ropa en las Figuras 4.29(a), 4.30(a) y 4.31(a), y el seguimiento estable en las Figuras 4.29(b), 4.30(b) y 4.31(b).

4.3. Casos de seguimiento no adecuado

De acuerdo con las pruebas anteriores se pudo comprobar la estabilidad y funcionamiento del sistema, sin embargo, en algunas ocasiones éste realizó un mal seguimiento debido a las limitantes y observaciones descritas en la sección anterior. A continuación se presentan ejemplos del seguimiento no adecuado y las razones del porqué se realiza:

1. La ropa se confunde con el color de piel, ésto es una limitante del sistema, ya que



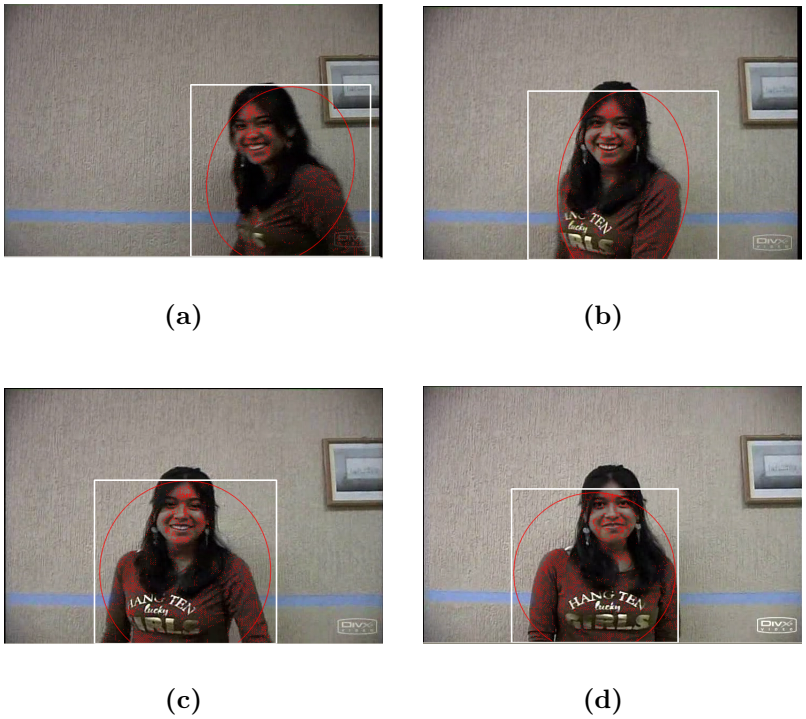
(a) **(b)**

Figura 4.31: Ejemplo de falsos positivos

debido a la iluminación los colores parecidos a la piel (como el color café) se confunden con la misma, por lo que el *blob* se dibuja de manera errónea al reconocer píxeles que no son piel, la solución es que la persona debe vestir colores que contrasten con el color de piel. En las Figuras 4.32 y 4.33 se observa cómo el sistema confunde la ropa con el color de piel, por lo que se da una mala identificación de la cabeza.

2. El proceso para iniciar el seguimiento depende de la diferencia de imágenes, ya que a partir de ésta se realiza la inicialización del *blob*, por lo que el sistema depende de que la región de la cabeza esté contenida en el resultado de la diferencia de imágenes, ya que puede ocurrir un mal seguimiento de la cabeza de la persona si en dicho resultado esta región no se encuentra. Un ejemplo de esta situación se presenta en la Figura 4.34, donde la persona se encuentra situada en la escena desde el inicio del video, y debido a que la persona mueve primero el brazo, en la región resultante de la diferencia de imágenes no se encuentra la cabeza de la persona, y dado que la región del brazo también contiene píxeles de color de piel, ésto provoca un falso positivo y se realiza el seguimiento de éste.

3. El control de la iluminación es otro factor importante durante el seguimiento, ya



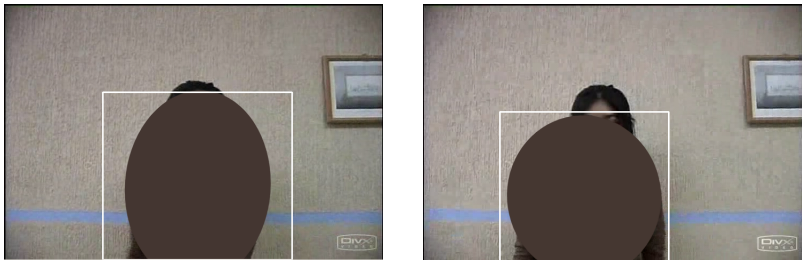
(a) **(b)**

(c) **(d)**

Figura 4.32: Seguimiento incorrecto debido a la confusión de la ropa con la piel, mostrando el *blob* con píxeles.

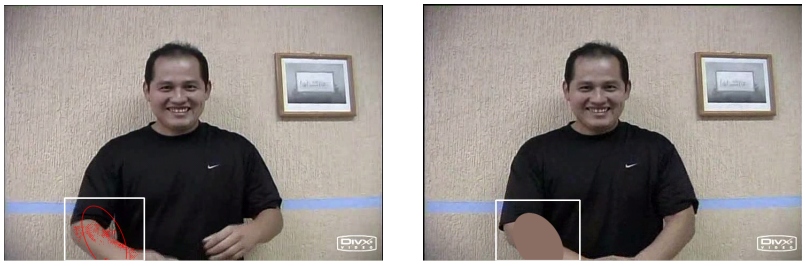
que puede haber confusión con objetos de la escena. Los cambios de iluminación en la escena producen falsos positivos de piel, una solución a este problema sería un mejor control de la iluminación en la escena. Las Figuras 4.35 y 4.36 muestran el seguimiento al cuadro que se encuentra en la escena debido a la salida de la persona de ésta y a los cambios de iluminación. En las Figuras 4.35(a), 4.35(b), 4.36(a) y 4.36(b) se aprecia un seguimiento adecuado, sin embargo, en 4.35(c) y 4.36(c) cuando la persona sale de la escena se realiza el seguimiento del falso positivo (cuadro), lo que provoca que cuando la persona aparece nuevamente en la escena como se observa en 4.35(d) y 4.36(d) el sistema no la sigue de inmediato, pero cuando la persona se mueve como se percibe en 4.35(e) y 4.36(e) esto causa cambios de iluminación en la escena provocando que el número de píxeles

(a) **(b)**



(c) **(d)**

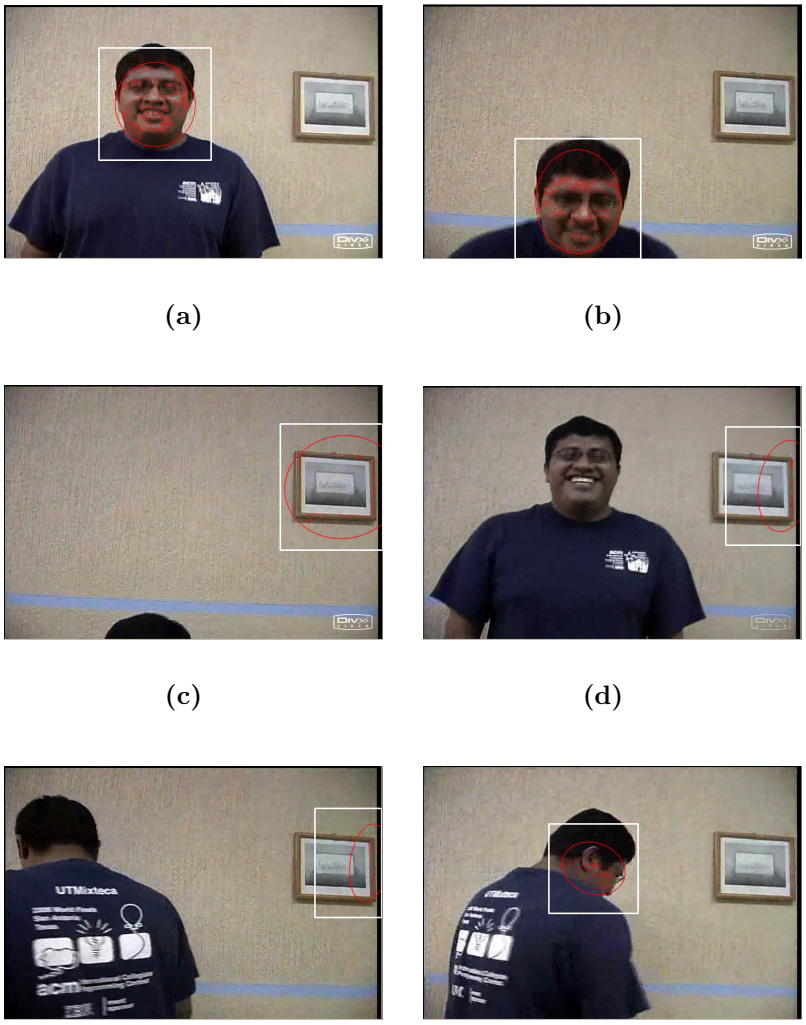
Figura 4.33: Seguimiento incorrecto debido a la confusión de la ropa con la piel, mostrando el *blob* completo



(a) **(b)**

Figura 4.34: Incorrecto seguimiento debido a que la persona se encontraba en la escena desde el principio.

reconocidos como piel sobre el área del cuadro disminuya, por lo que la ventana de búsqueda incrementa su tamaño hasta encontrar nuevamente a la persona como se visualiza en 4.35(f) y 4.36(f).

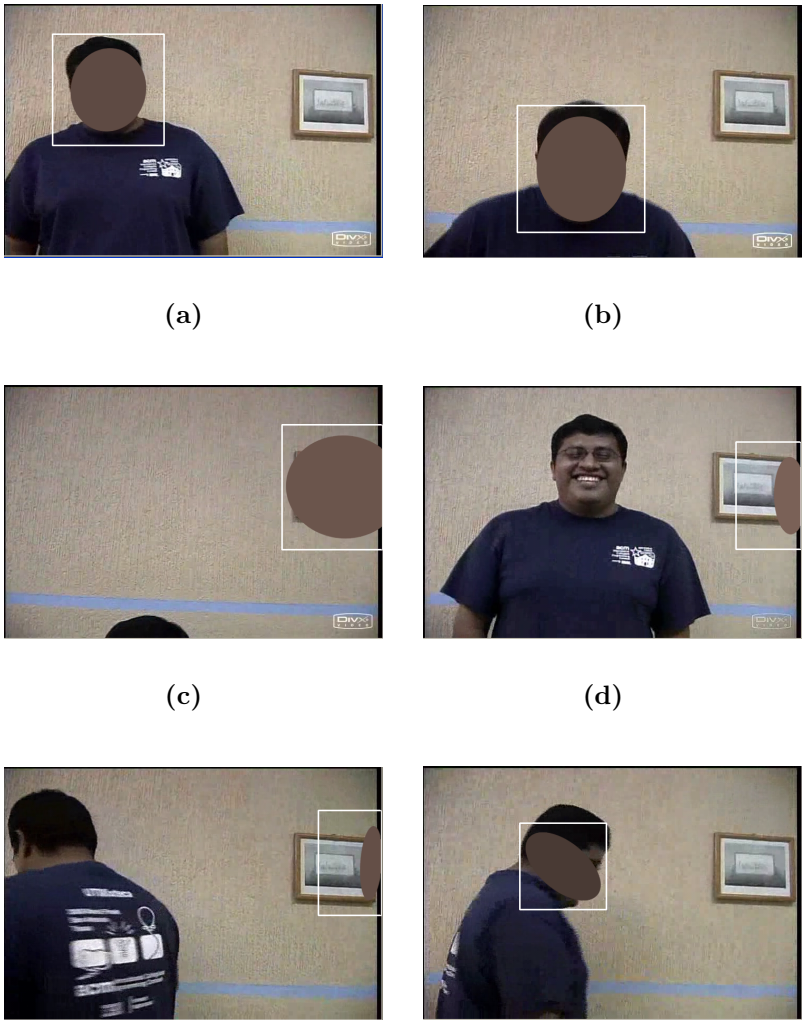


(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.35: Seguimiento de un distractor debido a la salida de la persona y a los cambios de iluminación, mostrando el *blob* con píxeles.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.36: Seguimiento de un distractor debido a la salida de la persona y a los cambios de iluminación, mostrando el *blob* completo.

Capítulo 5

Conclusiones y trabajos futuros

En la presente tesis se ha presentado un método para la localización y seguimiento de la cabeza de una persona en una escena cerrada, es decir, bajo ciertas condiciones de iluminación. Al concluir el desarrollo se cumplió con el objetivo planteado a través de la construcción de un sistema, de acuerdo al diseño presentado en la Figura 3.1. Este sistema recibe de entrada un video, y presenta el seguimiento de la persona involucrada mediante un *blob*.

Los resultados de las pruebas que se realizaron, indican que el sistema es capaz de realizar el seguimiento de la cabeza de la persona en alguna escena cerrada utilizando una elipse como representación del *blob*. El sistema puede representar el *blob* de dos maneras distintas, la primera es una elipse con píxeles de color rojo que indican los puntos que pertenecen a la cabeza, y la segunda, una elipse con color de relleno, que indica el color promedio del tono de piel del rostro. Para hacer más robusto el seguimiento se ha utilizado una ventana de búsqueda para evitar buscar a la persona en toda la escena, y así mantener el buen rendimiento del sistema. Se ha observado que la ventana de búsqueda ayuda además a mantener la estabilidad del sistema cuando se presenta alguna limitación del mismo, es decir, el sistema se mantiene estable ante situaciones que presentan las más importantes limitaciones del sistema como son: Seguimiento a

una sola persona, representación del seguimiento sólo de la cabeza de la persona, la escena tiene que ser cerrada, seguimiento de acuerdo a la posición y color de piel del rostro. La ventana de búsqueda contiene la posible área por donde la cabeza se puede mover, por lo que se optimiza la búsqueda, además de poder aumentar su tamaño al no encontrar suficientes píxeles para dibujar el *blob*. Con estas características mantiene la estabilidad del sistema ante las limitaciones antes mencionadas.

Otro punto importante a considerarse es que el algoritmo presentado en esta tesis depende de las características de la escena, así como del hardware utilizado (cómo la resolución y velocidad de la cámara). Además la escena no debe contener distractores que puedan confundirse con el color de la piel, ya que pueden alterar el aspecto del *blob* o crear el seguimiento de falsos positivos, esto debido a los cambios de iluminación. Otra limitante es que las personas deben vestir colores que contrasten con el color de la piel, ya que ciertos tipos de color de ropa pueden confundirse con la piel.

Se puede resumir las aportaciones de este proyecto como sigue:

1. Se obtuvo un sistema de seguimiento de la cabeza por medio de un *blob* que puede ser la base para la construcción de aplicaciones de VC como las mencionadas en el capítulo 1). Para iniciar el seguimiento se hizo uso de dos modelos de distribución normal generados previamente (color de piel y fondo), además se utilizó una ventana de búsqueda para mejorar el rendimiento y mantener la estabilidad del sistema.

2. Se adquirieron un conjunto de 38 videos (30 videos de la cámara A, 6 videos de la cámara B y 2 videos de la cámara C), los cuales pueden ser utilizados en posteriores investigaciones de seguimiento de personas.

3. Se implementó una herramienta adicional que permite generar modelos de distribución normal multivariada. Esta herramienta ayuda a escalar el sistema, como

se explica enseguida.

4. Otra aportación valiosa de este trabajo consiste en la creación de una distribución normal de color piel en el espacio de color RGB normalizado, la cual caracteriza en su mayoría a un conjunto pequeño de los tonos de piel que se encuentran en la zona de la Mixteca[1] , del estado de Oaxaca. Y aunque el conjunto de tonos de piel es pequeño (32 personas) el funcionamiento del sistema es adecuado. Sin embargo, con la aplicación que se menciona en el apéndice D, este modelo de piel se puede ampliar agregando los tonos de piel de más personas y así obtener una distribución normal que considere un conjunto mayor de tonos de piel, la cual permita mejorar el rendimiento del sistema o ser utilizada en otras aplicaciones.

Como trabajos futuros se podrían eliminar las limitaciones del sistema. Una de ellas y la principal, es la de seguir sólo a una persona; en el futuro se podrían tener más instancias del *blob* y así poder seguir a más de una persona.

En el sistema actual, el *blob* se utiliza como una opción para representar el seguimiento de la cabeza de la persona. Sin embargo, otra modificación muy útil que se podría desarrollar es la de seguir a más partes del cuerpo, creando instancias para cada una de las partes que se desea seguir, así se podrían tener instancias para las manos, piernas, tronco y pies.

Otra limitación es que sólo se consideran escenas cerradas. Se podría desarrollar un módulo para que también pudiera funcionar en escenas al aire libre. Este trabajo, sin embargo, es una tarea difícil, debido a los constantes cambios de iluminación que se tienen, además de las distintas escenas que se pueden encontrar. Estas condiciones adversas hacen que varíe el color y aspecto de la piel.

[1] Es importante mencionar que en el conjunto de tonos de piel también se consideró a una persona de tonalidad clara de origen germánico.

Además, otra modificación importante podría ser con respecto a la configuración de la cámara, hasta ahora se considera que la cámara se encuentra fija, pero se podría considerar la posibilidad que también la cámara estuviera en movimiento.

Es importante mencionar que el *blob* utilizado en este trabajo incluye posición y color de piel, un trabajo a futuro sería el de incluir otras características internas, como textura o movimiento, así como identificar los ojos y/o la boca, y de esta manera hacer más robusto el seguimiento de la cabeza de una persona.

Como se mencionó anteriormente, el trabajo desarrollado es sólo un medio, no un fin, es decir, que se puede utilizar para el desarrollo de aplicaciones más complejas. El sistema detecta y realiza el seguimiento de la cabeza, una vez realizada esta tarea, el resultado del seguimiento se podría aplicar a la detección de gestos simples, análisis de comportamiento, creación de modelos en 3D de rostros humanos y diversas aplicaciones de VC. Una aplicación muy importante en la que se podría aplicar el sistema se menciona en el articulo "Vigilancia avanzada: del *tracking* a la detección de sucesos" [9], la cual se refiere al campo de la vigilancia automática, permitiendo la localización y el seguimiento de múltiples objetos en escenarios complejos. Trabajando de forma automática en un entorno de vigilancia con cámaras en un circuito cerrado, el sistema decide qué cámaras deben ser monitorizadas o grabadas (evitando así la grabación automática de escenas vacías) y en general, ayuda a los operadores a optimizar el uso de los recursos.

Otra aplicación futura del sistema sería la mencionada en [20], donde se analiza el seguimiento de tres *blobs*, los cuales son utilizados para detectar y seguir a la cabeza y manos de una persona. El propósito de tal análisis es explorar el comportamiento de la persona e identificar cuando está mintiendo. De manera similar a nuestro trabajo se basan en el reconocimiento del color de piel y determinan el estado de ánimo de la persona (si está nerviosa, relajada o sobre-controlada) con base en el estudio de la posición y velocidad de los *blobs*.

Apéndice A

OpenCV

OpenCV [1] (Open Source Computer Vision library) es una librería abierta desarrollada por Intel. Esta librería proporciona funciones de alto nivel para el procesado de imágenes y video. Permite a los programadores crear aplicaciones poderosas en el dominio de la VC. OpenCV está disponible en Windows y Linux, es libre y se distribuye en conjunto con el código fuente. Permite realizar tareas relacionadas con la VC en tiempo real y con alta productividad. Dentro de las funciones proporcionadas con la librería se tienen a:

1. Creación y acceso a imágenes o video.
2. Operaciones aritméticas y lógicas de la imagen.
3. Filtrado de la imagen.
4. Transformación lineal de la imagen.
5. Morfología de la imagen.
6. Conversión del espacio de color (RGB, LAB, XYZ, LUV, HSV, HLS, YCrCb y escala de grises).

[1] **Disponible en:** http://www.intel.com/research/mrl/research/opencv/

7. Histograma de la imagen y umbralización.

Los lenguajes con los que tiene compatiblilidad son: ANSI C, ANSI C/C++ y Matlab, mismos a los que les proporciona transparencia para el manejo de imágenes y video.

Apéndice B

Manual de instalación de OpenCV con Visual C++

El presente manual es una guía para la instalación de OpenCV y la configuración de Visual C++ 6.0 para que trabaje con estas librerías. El manual cubre la instalación de OpenCV en Windows XP, sin embargo, OpenCV puede ser instalado en otros sistemas operativos incluyendo Windows 95/98/2000/NT y Linux.

Para iniciar se asume que el entorno de Visual C++ 6.0 se encuentra instalado. A continuación se explica la instalación de DirectX SDK, seguido de la instalación y configuración de OpenCV con el sistema operativo Windows XP, y finalizando con la configuración de OpenCV para que pueda trabajar con Visual C++ 6.0.

B.1. Instalación de DirectX SDK

Para poder utilizar OpenCV se requiere instalar Microsoft DirectX SDK, una versión actualizada de este software puede descargarse de la página: http://msdn2.microsoft.com/es-mx/xna/aa937788.aspx. Esta página contiene las últimas versiones del programa, se escoge la última versión (aunque puede ser cualquiera) y después se descarga el archivo: dxsdk_feb2007.exe (Figura B.1), en el directorio que haya elegido para tal acción. El

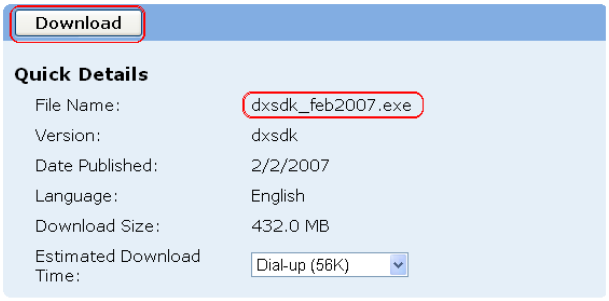


Figura B.1: Descarga de DirectX SDK

nombre varía de acuerdo al mes en que salió la versión, en este caso la versión que se instaló corresponde al mes de febrero.

Una vez descargado el archivo de la página mencionada, se debe hacer doble *click* sobre este archivo para ejecutarlo, aparecerá la ventana de licencia (Figura B.2(a)), se debe dar *click* en el botón "Yes" para continuar con la instalación. Enseguida aparecerá un cuadro de diálogo (Figura B.2(b)) que indicará el avance de la extracción de los archivos necesarios para la instalación. Posteriormente aparecerá un cuadro de diálogo (Figura B.2(c)) que permitirá descomprimir el archivo. Se puede elegir un directorio personal (haciendo *click* sobre el botón *browse*...) o el directorio por $default$, para iniciar el proceso de descompresión se presiona el botón "Unzip". Al terminar de extraer todos los archivos, aparecerá un mensaje que indica que ha terminado el proceso (Figura B.2(d)).

Para iniciar con la instalación se debe hacer doble *click* en el archivo "Microsoft DirectX SDK.exe" que se encuentra en la carpeta que se eligió al descomprimir el archivo (Figura B.3(a)). Al ejecutar el archivo aparecerá una ventana que indica el inicio de la instalación de DirectX SDK (Figura B.3(b)), para continuar se debe presionar el botón "Next", después se debe aceptar los términos de la licencia (Figura B.3(c)) y se presiona nuevamente el botón "Next". En la siguiente ventana se deben especificar los componentes a instalar, así como la ruta de instalación, se recomienda dejar las opciones por

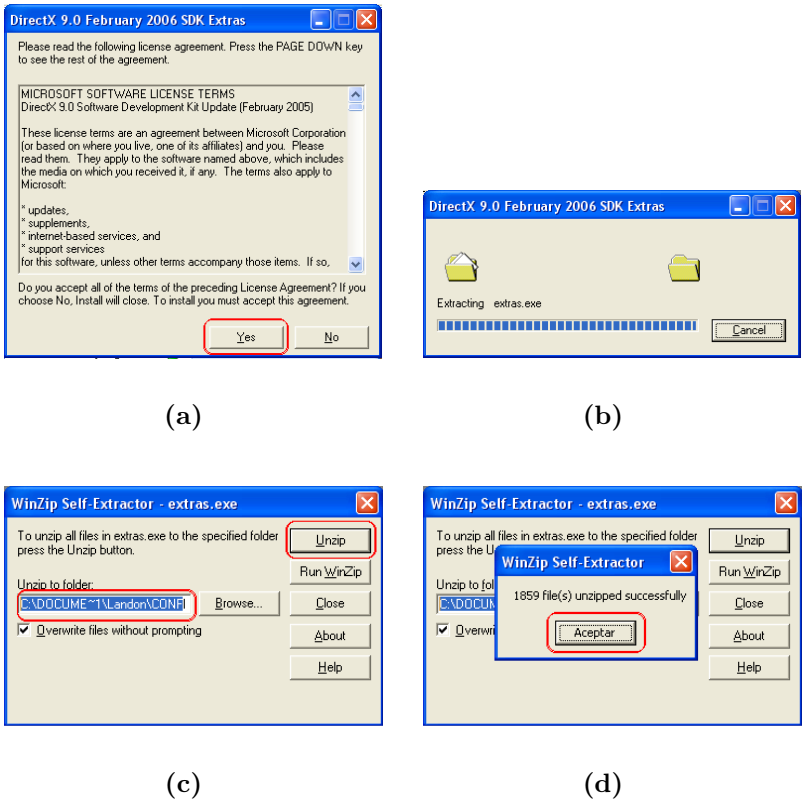


Figura B.2: Extracción de los archivos de instalación de DirectX SDK

default (Figura B.3(d)), de nuevo se presiona el botón "Next". Aparecerá una ventana que mostrará el avance de la instalación (Figura B.3(e)), al terminar aparecerá una última ventana que indicará que la instalación ha finalizado correctamente (Figura B.3(f)), se debe presionar el botón "Finish" para terminar.

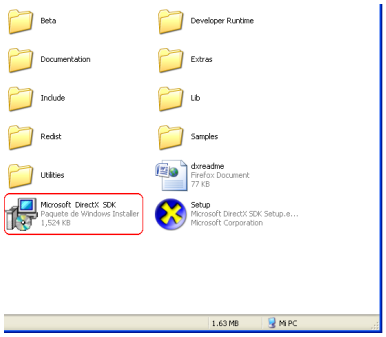
La instalación del software requiere como mínimo 432 Mb de espacio libre en disco duro. Es importante recordar la ruta de instalación dado que más adelante ésta se debe especificar en Visual C++ 6.0.

B.2. Instalación de OpenCV

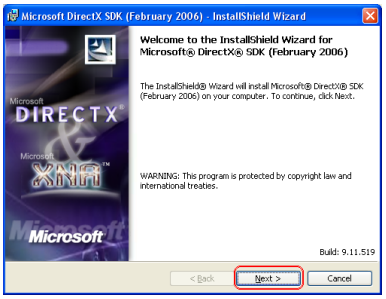
El sitio principal de OpenCV: http://sourceforge.net/project/showfiles.php? group_id=22870 proporciona los archivos necesarios para la instalación, ya sea el código libre para poder compilarlo (OpenCV courses) o el archivo de instalación ejecutable (opencv-win). En la página web mencionada se debe buscar el archivo "opencv-win" para su descarga, la Figura B.4 muestra el archivo que se debe descargar.

Para la instalación del software se requiere como mínimo 36.2 Mb de espacio libre en disco. Y con el propósito de hacer más fácil la configuración que se realizará se sugiere instalar OpenCV en el directorio por *default*, ejemplo: "C:\Archivos de Programa\OpenCV", debido a que esta ruta de instalación será utilizada en Visual C++ 6.0.

Para iniciar con la instalación se deberá hacer doble *click* en el archivo descargado "OpenCV-win.exe", aparecerá una ventana que indica el comienzo de la instalación (Figura B.5(a)), para continuar se presiona el botón "Next". A continuación se deben aceptar los términos de la licencia y posteriomente presionar el botón "Next" (Figura B.5(b)). Ahora se debe especificar la ruta de instalación de OpenCV, se recomienda dejar la ruta por *default*, una vez escogida la ruta se presiona el botón "Next" (Figura

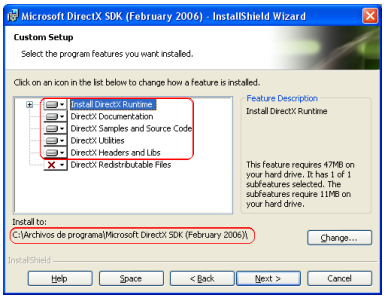


(a)

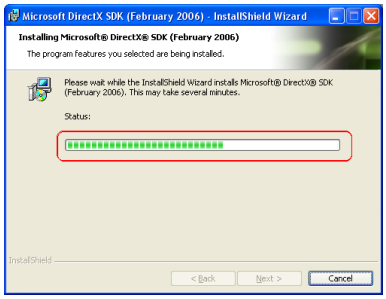


(b)

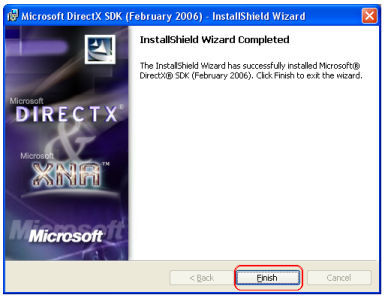
(c)



(d)



(e)



(f)

Figura B.3: Instalación de DirectX SDK

Package	Release	Date	Notes / Monitor	Downloads
ch-opencv	2.3.0	August 3, 2005	-	Download
openAVCSR-win	alpha 1	April 22, 2003	-	Download
OpenCV courses	CVPR'01 course	January 8, 2002	-	Download
opencv-doc	HOWTOs-Tutorials	July 30, 2003	-	Download
opencv-linux	1.0	November 6, 2006	-	Download
opencv-win	1.0	October 19, 2006	-	Download

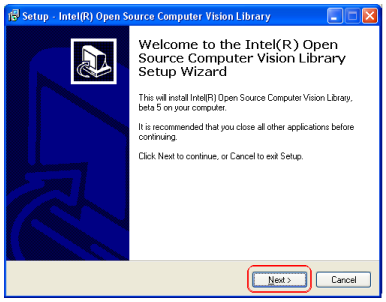
Figura B.4: Lista de descargas de OpenCV

B.5(c)). En la ventana siguiente aparece la opción para especificar el nombre para crear un acceso directo en el menú inicio (Figura B.5(d)), para continuar se presiona el botón "Next". Después aparece un *chekbox* para indicar si se desea agregar la ruta de instalación de OpenCV al PATH del sistema, hay que asegurarse que este marcado éste control y presionar el botón "Next" (Figura B.5(e)). Ahora aparecerá una ventana que indica que el programa está listo para la instalación indicando las opciones que se han escogido anteriormente (Figura B.5(f)), para continuar se presiona el botón "Install", al realizar esta acción aparecerá una ventana que indica el proceso de la instalación (Figura B.5(g)), al terminar, la ventana de la Figura B.5(h) muestra que se ha terminado la instalación, y da la opción de ver la documentación de OpenCV, para terminar se debe presionar el botón "Finish".

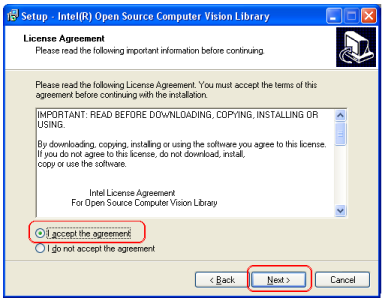
B.3. Configuración de Visual C++ 6.0 para utilizar OpenCV

El siguiente proceso consiste en establecer las variables para que Visual C++ trabaje con OpenCV. Los pasos que se describen en esta sección están dirigidos a usuarios con experiencia mínima en Visual Studio, por lo que los detalles serán cuidadosamente explicados. La siguiente configuración fue realizada en el sistema operativo Windows XP Pro, versión 2002 con Servi Pack 2, y con Visual Studio versión 6.0.

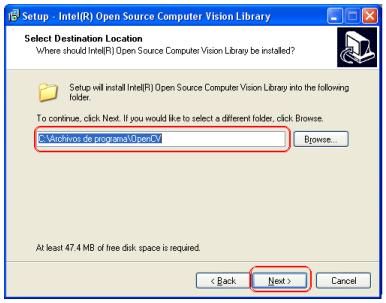
Paso 1: Una vez instalado Visual C++ 6.0 de Visual Studio, se establece la ruta de OpenCV en la variable "*Path*" del sistema. Para realizarlo es necesario abrir el panel de control como lo indica la Figura B.6. Aparecerá entonces la ventana de la Figura B.7, en ella se debe hacer doble *click* sobre el icono "Sistema", se abrirá el cuadro de diálogo de la Figura B.8, en este cuadro de diálogo se selecciona la pestaña de "opciones



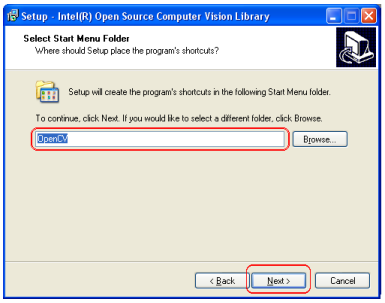
(a)



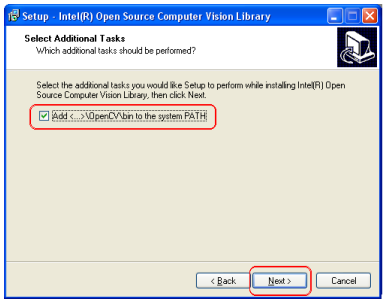
(b)



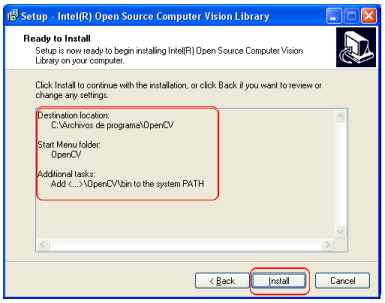
(c)



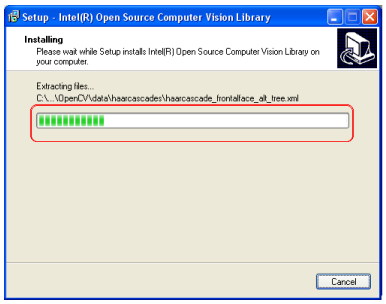
(d)



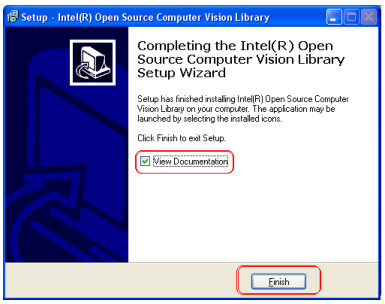
(e)



(f)



(g)



(h)

Figura B.5: Instalación de OpenCV

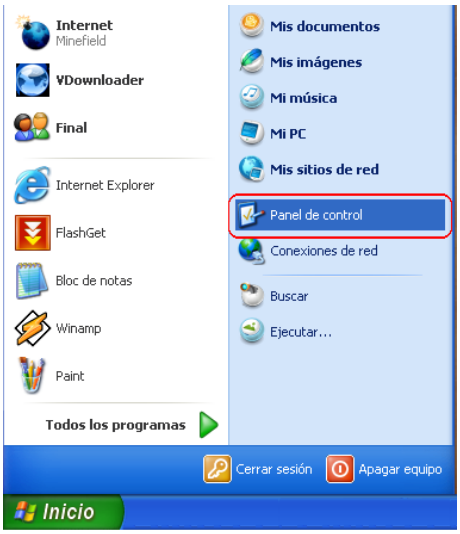


Figura B.6: Selección del panel de control

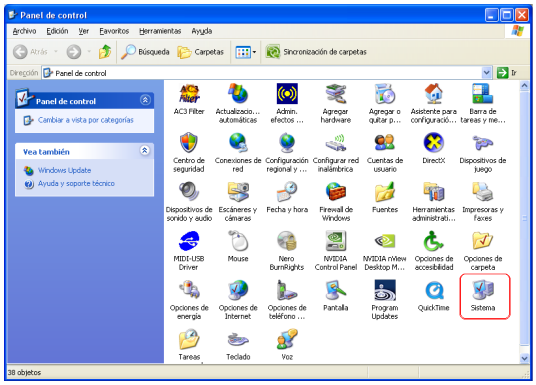


Figura B.7: Selección del icono "Sistema" en el panel de control.

avanzadas", y se selecciona el botón "Variables de entorno" (Figura B.8).

En el cuadro de diálogo "Variables de entorno", en la sección "Variables de sistema" se busca la variable "Path" y se selecciona la opción "Modificar" (Figura B.9). Se agrega la ruta de la carpeta "bin" de OpenCV en el cuadro de texto "Valor de variable". Para hacer esto, se tiene que agregar la ruta al final de la lista, asegurándose que un punto y coma separe las rutas (Figura B.10). Si se seleccionó la ruta de instalación por *default* durante la instalación, la ruta debe ser "C:\Archivos de Programa\OpenCV\bin". De no ser así, localice la ruta de la carpeta "bin" de OpenCV y agrégela como se indicó.

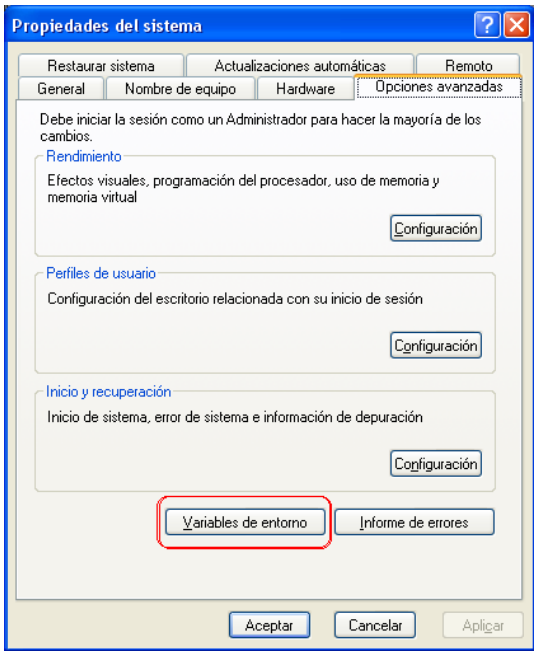


Figura B.8: Cuadro de diálogo "Propiedades de sistema".

Finalmente, seleccione la opción "Aceptar" de todos los cuadros de diálogo abiertos. Ahora para que la variable "Path" del sistema se actualice se debe reiniciar la computadora. A partir de este momento, OpenCV debe estar disponible para poder encontrar las DLL's requeridas. Si se necesita ejecutar la aplicación en otro sistema que no tenga instalado OpenCV, se tienen que copiar las librerías mostradas en la Figura B.11 al directorio \debug de la aplicación. Estas librerías se encuentran ubicadas en "C:\Archivos de Programa\OpenCV\bin".

Paso 2: Visual C++ 6.0 debe ser capaz de encontrar archivos de cabecera, archivos fuente, executables y librerías de OpenCV y DirectX SDK, por lo que se necesita entrar e indicarle al entorno donde encontrar lo único que necesita. Se inicia Microsoft Visual C++ 6.0, el ambiente de programación debe estar sin haber abierto algún proyecto o archivo, tal como se muestra en la Figura B.12.

Del menu "Tools", se selecciona la opción "Options..." (Figura B.13), el cuadro de diálogo de opciones debe ser mostrado ahora.

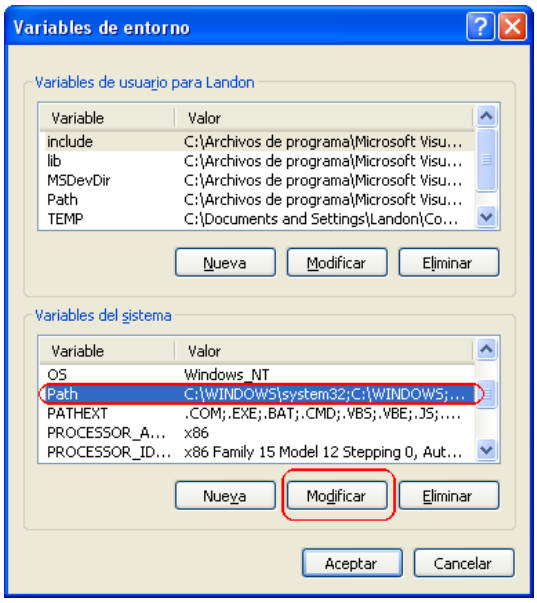


Figura B.9: Cuadro de diálogo "Variables de entorno".

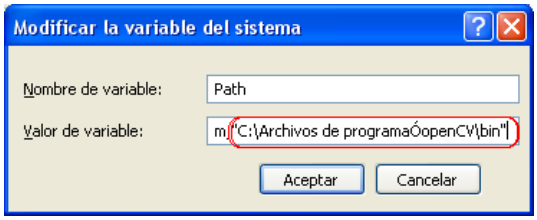


Figura B.10: Cuadro de diálogo "Modificar variable de sistema".

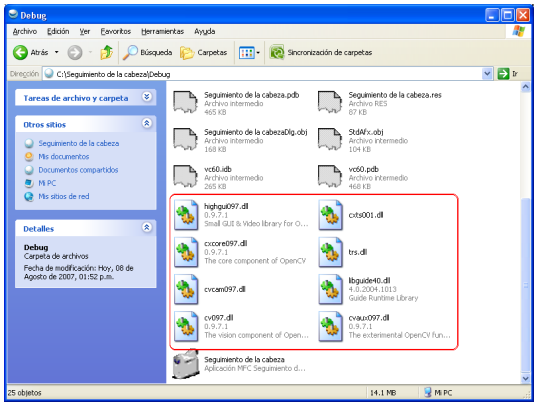


Figura B.11: Archivos con extensión "dll" que se utilizan para ejecutar una aplicación construida con OpenCV

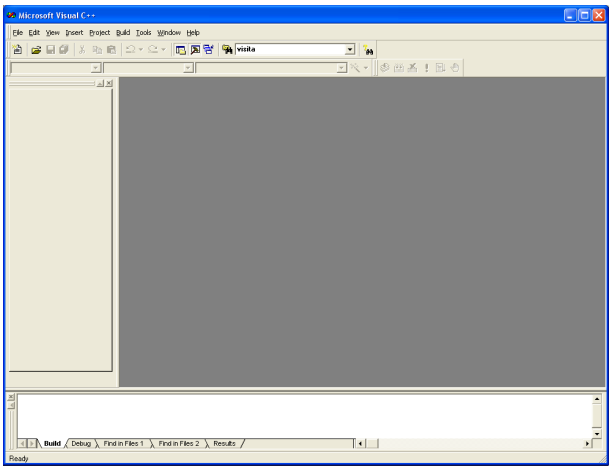


Figura B.12: Interfaz de Microsoft Visual C++ 6.0.

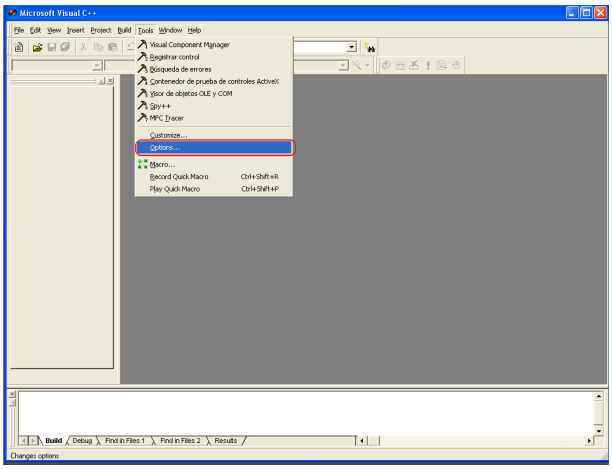


Figura B.13: Ir al menú "Tools" y seleccionar la opción "Options".

En el cuadro de diálogo "Options...", se selecciona la pestaña "Directories". El cuadro de diálogo "Options" con la pestaña "Directories" seleccionado se muestra en la Figura B.14 en dicha figura aparecen los directorios especificados para los archivos de cabecera.

Visual C++ puede tener archivos de cabecera ya especificados. Si no aparece la ruta de la carpeta para los archivos de cabecera de DirectX SDK, entonces se tiene que especificar para que pueda encontrar los archivos necesarios. Si se ha escogido el directorio por $default$ al instalar DirectX SDK, la ruta debe ser la misma como se

muestra en la Figura B.14, la cual es "C:\Archivos de Programa\Microsoft DirectX SDK\include". **Es importante que la entrada de la ruta de DirectX esté al principio de la lista de los archivos de cabecera**, para realizar esta acción se debe seleccionar la ruta que se indicó anteriormente y presionar el botón "Move item up" como se muestra en la Figura B.15, hasta que la ruta llegue al principio de la lista.

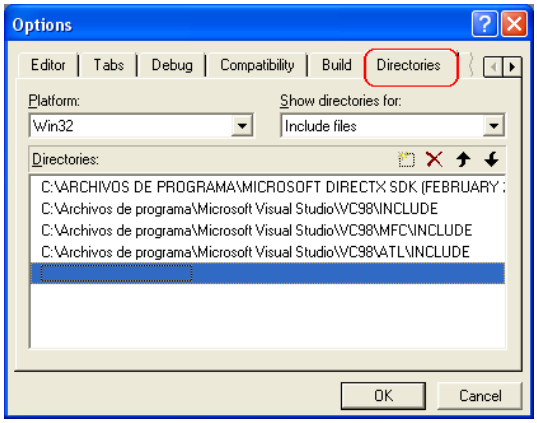


Figura B.14: Cuadro de diálogo "Options" con la pestaña "Directories" seleccionada.

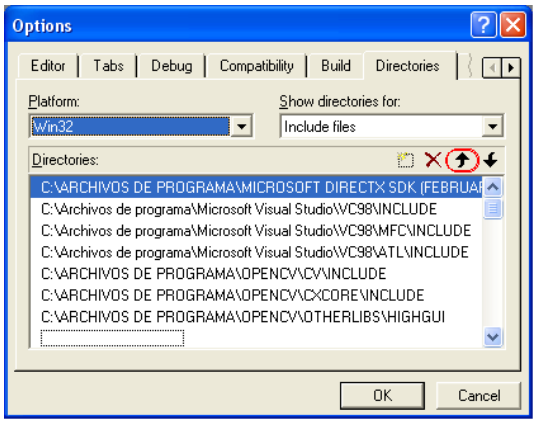


Figura B.15: Mover la ruta de DirectX SDK al principio de la lista.

Ahora se tienen que especificar más directorios, asumiendo que se seleccionó la ruta por *default* durante la instalación de OpenCV, se agregan los siguientes directorios (los directorios pueden ser diferentes si se ha escogido instalar OpenCV en una diferente ruta. Si es así, se tiene que escoger las rutas adecuadamente) ver la Figura B.15:

- C:\Archivos de Programa\OpenCV\cv\include
- C:\Archivos de Programa\OpenCV\cxcore\include
- C:\Archivos de Programa\OpenCV\otherlibs\highgui

Ahora, en la misma pestaña de "Directories", se selecciona "Library files" en la opción "Show directories for" (Figura B.16). Visual C++ 6.0 puede tener especificados directorios a algunas librerías. Si no aparece la ruta de la carpeta para los archivos de las librerías de DirectX SDK, entonces hay que especificarla. Si se ha escogido el directorio por *default* al instalar DirectX SDK, la ruta debe ser la misma como se muestra en la Figura B.16, la cual debe ser "C:\Archivos de Programa\Microsoft DirectX SDK\Lib\x86". Finalmente, se agrega el directorio "C:\Archivos de Programa\OpenCV\lib".

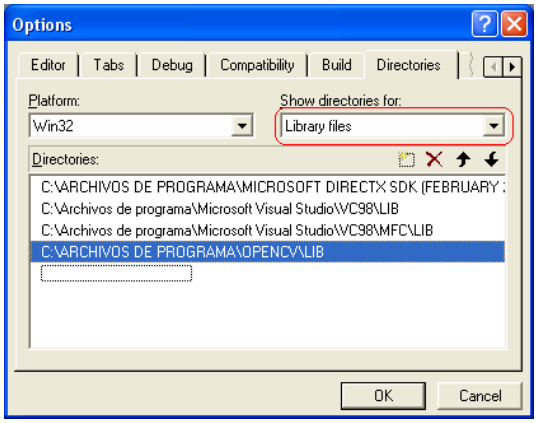


Figura B.16: Lista de los directorios de las librerías.

Ahora se selecciona "Executable files" de la opción "Show directories for", al igual que antes, se debe revisar que se encuentre la ruta para DirectX SDK como se muestra la Figura B.17, si no se encuentra, se debe agregar "C:\Archivos de Programa\Microsoft DirectX SDK\Utilities\Bin\x86" y se debe de mover hacia el inicio de la lista como se mencionó anteriormente. Por último, se agrega "C:\Archivos de programa\OpenCv\Bin".

La especificación de los directorios ha finalizado, hay que asegurarse de seleccionar

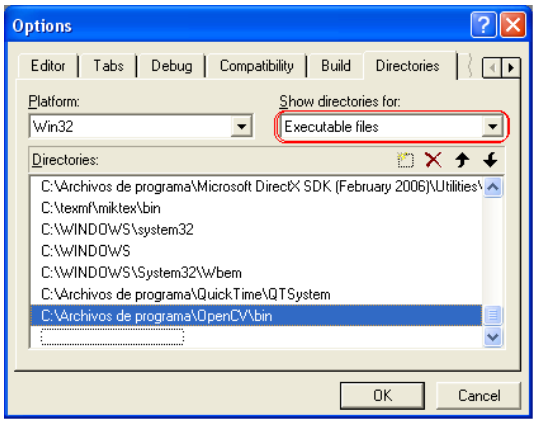


Figura B.17: Lista de archivos ejecutables

"OK" en el cuadro de diálogo "Options", o de lo contrario se tendrá que realizar todo de nuevo.

B.3.1. Creación de un nuevo proyecto que utilice OpenCV

Una vez que se ha configurado Visual C++ 6.0 para trabajar con OpenCV, se explicará la forma de crear una proyecto que utilice OpenCV. Se debe crear un nuevo proyecto normalmente (Seleccionar menú "File" y después seleccionar opción "New..."). Antes de crear alguna aplicación dentro del nuevo espacio de trabajo se necesita modificar la configuración del proyecto. Para realizar esto se debe hacer *click* en el menú "Projects" y seleccionar la opción "Settings..." (Figura B.18).

Un cuadro de diálogo aparecerá, se debe seleccionar "All Configurations" en la opción "Settings for" (Figura B.19). Ahora se hace *click* en la pestaña "Link", se selecciona "General" en la opción "Category". En el cuadro de edición bajo "Object/library Modules" se deben agregar las siguientes librerías (Figura B.20):

- cv.lib
- cxcore.lib

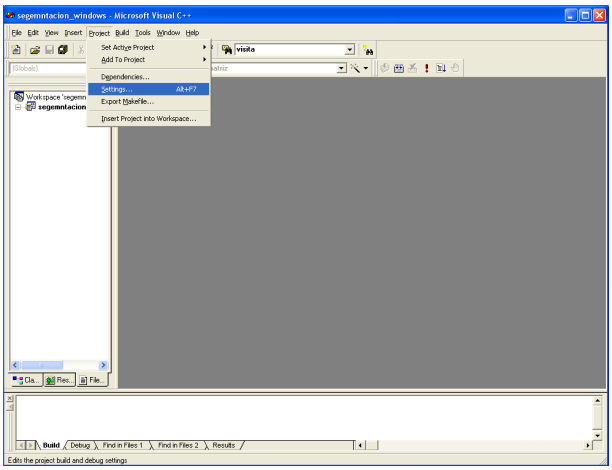


Figura B.18: Selección de la opción "Settings..." en el menú "Projects".

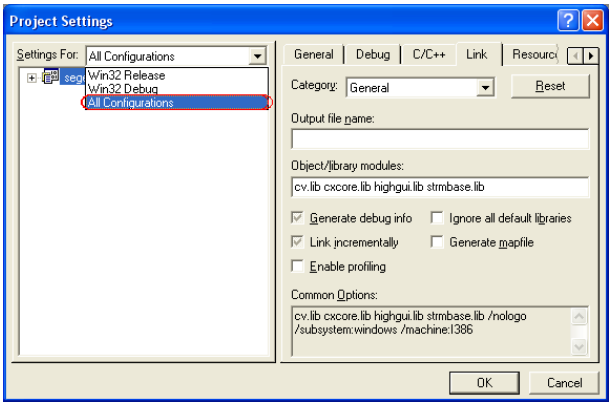


Figura B.19: Selección de "All configurations".

- highgui.lib
- strmbase.lib

Una vez que las librerías han sido ingresadas, se debe hacer *click* en "Ok", ahora el proyecto debe estar listo para trabajar con OpenCV.

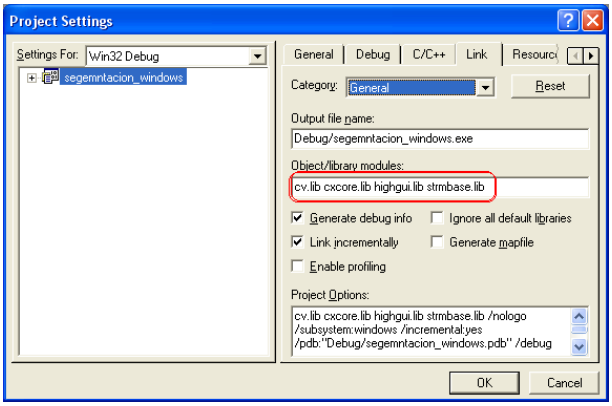


Figura B.20: Cuadro de edición "Object/library Modules" donde se deben agregar las librerías.

Apéndice C

Manual de Usuario

El presente apéndice es un apoyo para los usuarios que utilicen la aplicación desarrollada en este trabajo de tesis. Para poder trabajar con la aplicación se requiere su instalación en la computadora, para iniciar este proceso es necesario ejecutar el archivo "Seguimiento de la cabeza.exe", que se encuentra en el CD-ROM que acompaña la tesis (Figura C.1).

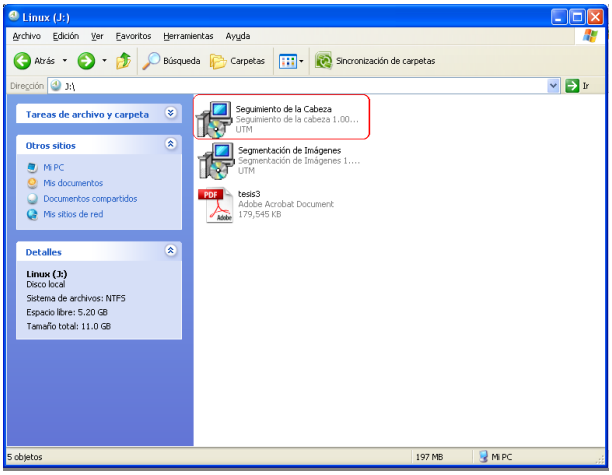


Figura C.1: Instalador de la aplicación.

Al ejecutar el archivo aparecerá una ventana que indica el inicio de la instalación (Figura C.2(a)), para continuar se debe presionar el botón "Siguiente", después se debe establecer la ruta de instalación de la aplicación (Figura C.2(b)), puede escoger la ruta

por *default* o bien, una ruta personalizada presionando el botón "examinar". Una vez elegida la ruta se debe presionar "Siguiente". En la siguiente ventana se debe decidir si se desea un acceso directo en el escritorio (Figura C.2(c)), se recomienda crear el acceso directo, para continuar se tiene que presionar "Siguiente". Ahora aparecerá una ventana indicando las opciones que se han escogido hasta el momento (Figura C.2(d)), para iniciar el proceso de instalación se debe presionar el botón "Siguiente", al realizar esta acción aparecerá una ventana que indica el avance de la instalación (Figura C.2(e)). Al terminar la ventana mostrará una mensaje dando la opción de ejecutar la aplicación (Figura C.2(f)).

Una vez instalado OpenCV para trabajar con Visual C++ 6.0 de acuerdo al apéndice B, se puede ejecutar el sistema de dos maneras distintas: Desde Visual C++ 6.0, abriendo el espacio de trabajo "video.dsw" que se encuentra ubicado en la carpeta donde se ha instalado la aplicación previamente (Figura C.3), para mayor detalle lea el archivo "Instrucciones.txt" que se encuentra en el mismo directorio. La segunda, utilizando el acceso directo que se ha creado al instalar la aplicación, el cual se encuentra en el escritorio.

Una vez iniciada la aplicación se mostrará la ventana principal del sistema, a través de ella se podrá ingresar a las funciones que ésta ofrece.

La ventana principal se muestra en la Figura C.4 y cuenta con las funciones que se describirán a continuación:

- **Tipo de Inicialización:** El sistema da la posibilidad de poder escoger los archivos de inicialización, se pueden utilizar los archivos por defecto (los archivos que contienen la información de los 32 videos) que se han construido a lo largo de la presente tesis, o bien, el usuario puede especificar algún archivo que haya creado con la "Aplicación para obtener los tonos de piel por medio de la seg-

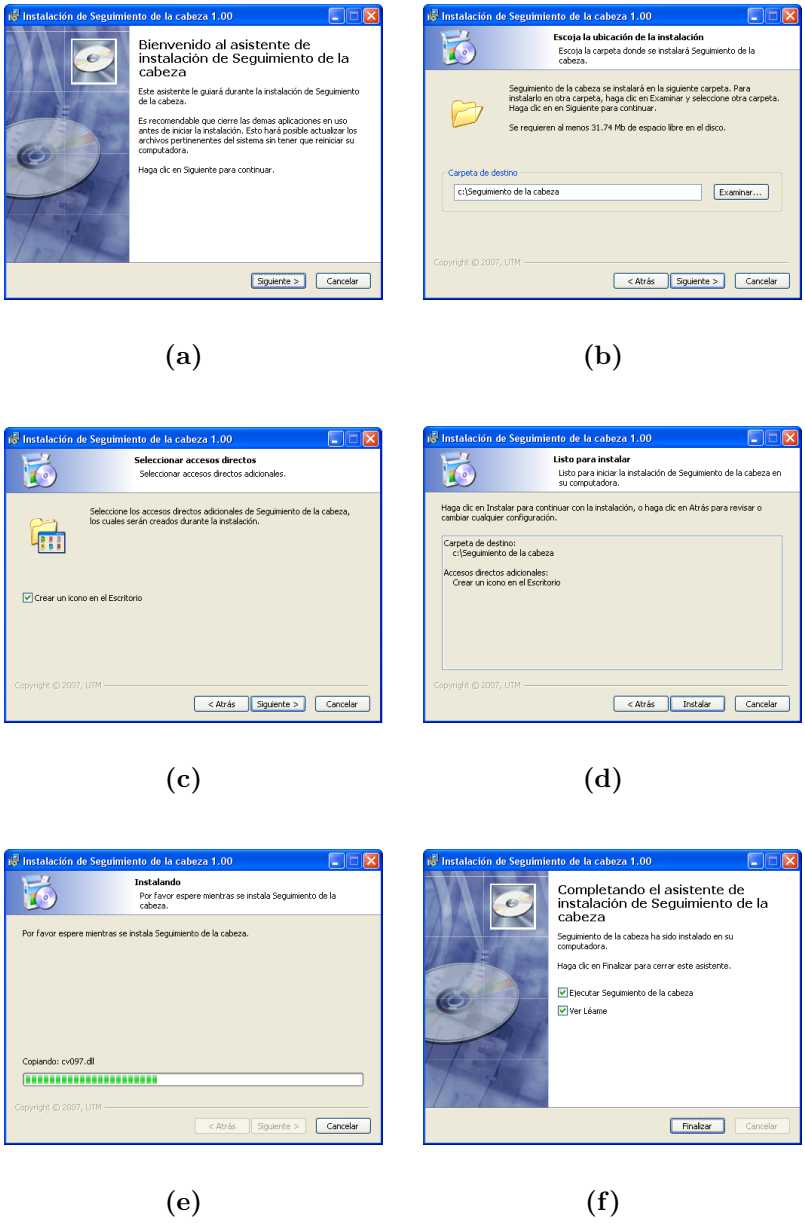


Figura C.2: Instalación de la aplicación "Seguimiento de la cabeza".

mentación por crecimiento de regiones" (ver apéndice D). En la presente sección se debe determinar que tipo de inicialización se desea (Figura C.5): **Por defecto**, el sistema utilizará los archivos que el sistema tiene por defecto. **Personalizada**, el usuario debe establecer los archivos para la clase de *color de piel* o la clase

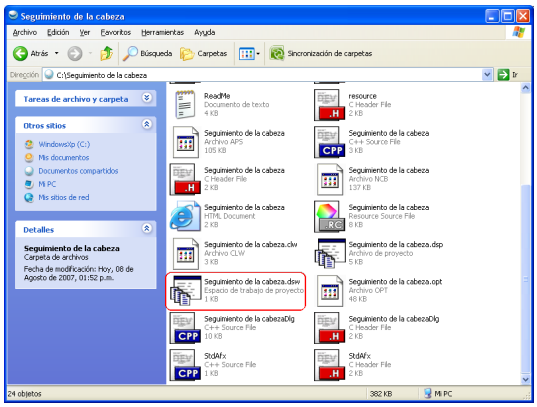


Figura C.3: Espacio de trabajo para editar el programa en Visual C++ 6.0.

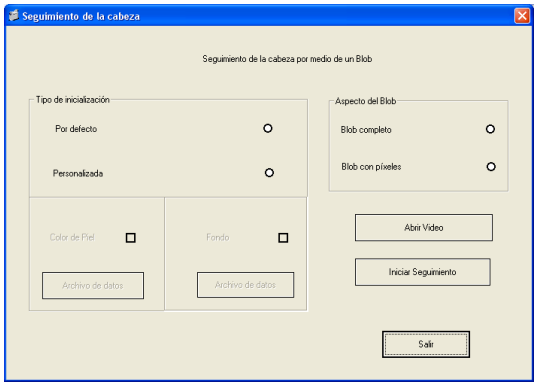


Figura C.4: Ventana principal del sistema.

fondo (al menos una) (Figura C.6). Al principio las casillas para seleccionar las clases que se definirán estarán desactivados, por lo que al elegir esta opción, estas casillas estarán disponibles. Al escoger alguna casilla el botón correspodiente estará disponible, este botón se usa para seleccionar algún archivo que contenga el vector medio y la matriz de covarianza, al presionarlo se abrirá una ventana que permita seleccionarlo, en la Figura C.7 se muestra el caso en que se desea redefinir el archivo de inicialización de la clase de color de piel, la ventana es similar para el caso de la clase fondo. Si se activa cualquiera de estas dos casillas (Color de Piel o Fondo), pero no se selecciona un archivo de inicialización correspondiente, aparecerá el mensaje de error de la Figura C.8.

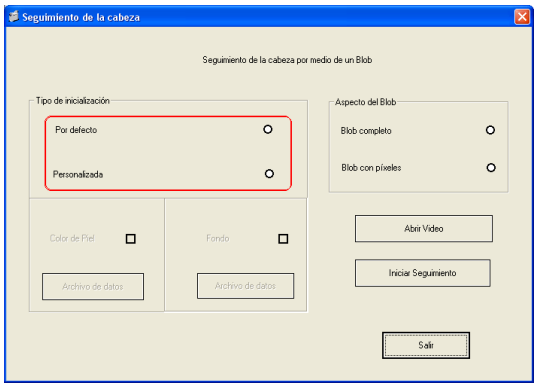


Figura C.5: Selección del tipo de inicialización.

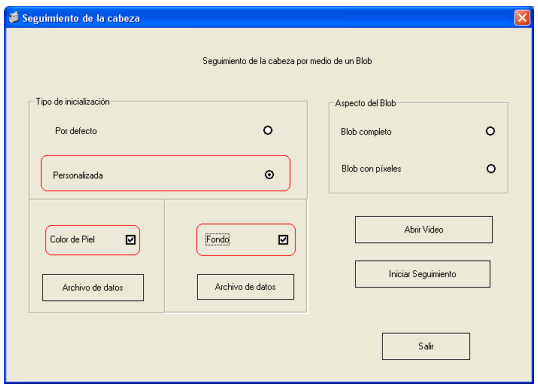


Figura C.6: Selección de la clase a redefinir.

- **Aspecto del *blob*:** Esta parte contiene dos opciones, las cuales definen el aspecto del *blob* durante el seguimiento de la cabeza de la persona. Si se selecciona la primera opción: "*Blob* completo" (Figura C.9), el aspecto del *blob* que se mostrará en el video será de una elipse con un color de relleno (color promedio de los píxeles que lo definen). Mientras que si se selecciona la segunda opción: "*Blob* con píxeles" (Figura C.10), el aspecto del *blob* que se mostrará será de una elipse color rojo sin relleno, pero conteniendo píxeles también de color rojo, indicando que son los puntos que se clasificaron como pertenecientes a la cabeza de la persona.

- **Abrir Video:** Este botón se utiliza para seleccionar el video que se desea procesar

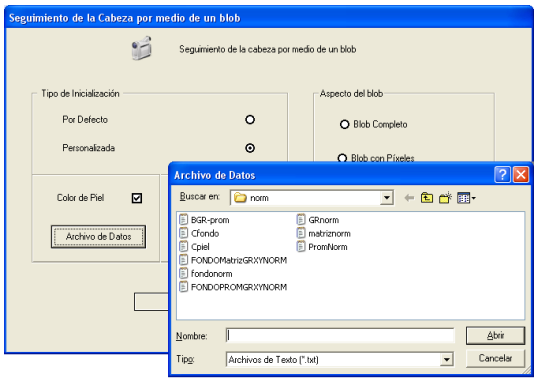


Figura C.7: Selección del archivo que define a la clase (la cual puede ser color de piel o fondo).

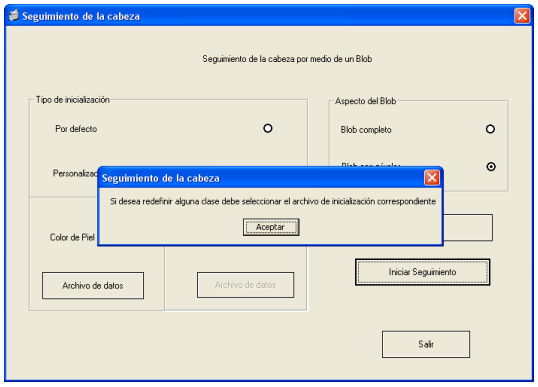


Figura C.8: Mensaje de error al no seleccionar un archivo de inicialización correspondiente a la casilla.

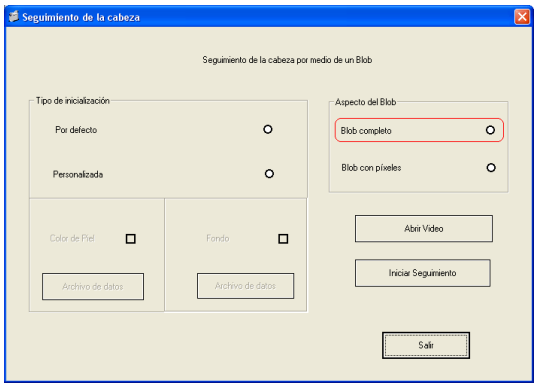


Figura C.9: Aspecto del *blob* seleccionado: *Blob* completo.

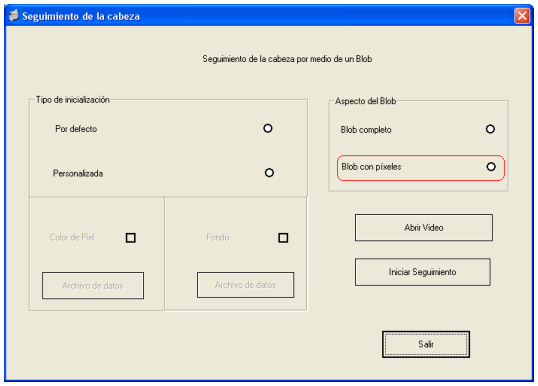


Figura C.10: Aspecto del *blob* seleccionado: *Blob* con píxeles.

con el sistema. Al seleccionar esta función aparecerá un cuadro de diálogo (Figura C.11) donde se podrá escoger el video, por *default* la extensión de los archivos a mostrar es ".avi", ya que son el tipo de archivos que soporta el sistema.

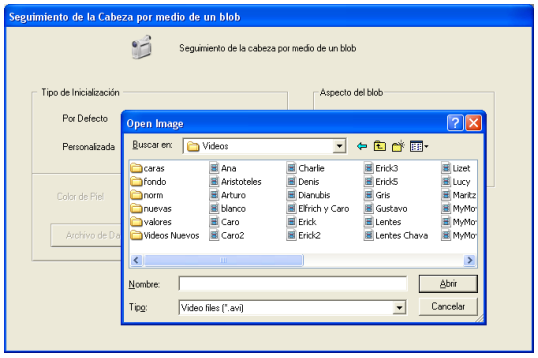


Figura C.11: Cuadro de diálogo utilizado para escoger el video a procesar.

- **Iniciar Seguimiento:** Antes de seleccionar esta opción es necesario seleccionar el tipo de inicialización, de lo contrario la aplicación mostrará un mensaje de error (Figura C.12), después se debe escoger algún archivo de video, de lo contrario aparecerá un mensaje de error (Figura C.13), también se tiene que elegir el aspecto del *blob*, de lo contrario, también el sistema mostrará una ventana de aviso (Figura C.14) que indica este error.

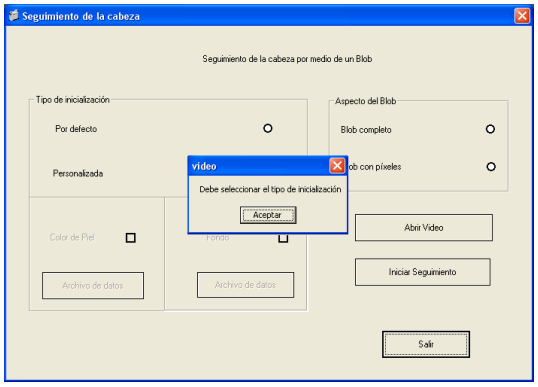


Figura C.12: Ventana de error al no seleccionar el tipo de inicialización.

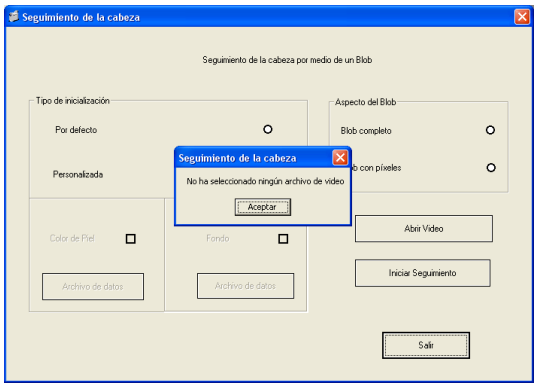


Figura C.13: Ventana de error al no seleccionar algún archivo de video.

Al seleccionar esta función se podrá observar que una nueva ventana aparecerá, en la cual se mostrará el video antes seleccionado (Figura C.15). Para iniciar con la localización y seguimiento de la cabeza de la persona es necesario hacer un *click* con el botón izquierdo de *mouse* sobre la ventana del video. Una vez realizada

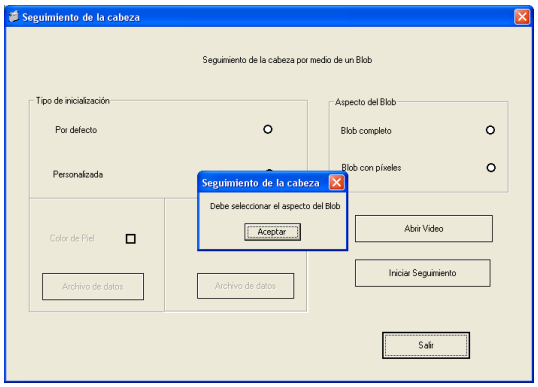


Figura C.14: Ventana de error al no seleccionar el aspecto del *blob*.

esta acción se podrá observar el *blob* (que dependerá del aspecto del *blob* que se haya seleccionado anteriormente, y que se puede cambiar en el transcurso de la duración del video) y la ventana de búsqueda alrededor de la cabeza de la persona (Figuras C.16 y C.17).

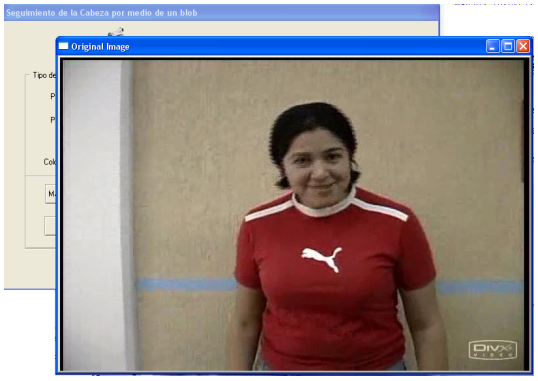


Figura C.15: Ventana que muestra el video seleccionado.

- **Salir:** Este botón se utiliza para salir del sistema. Al seleccionar esta opción la ventana principal se cerrará.

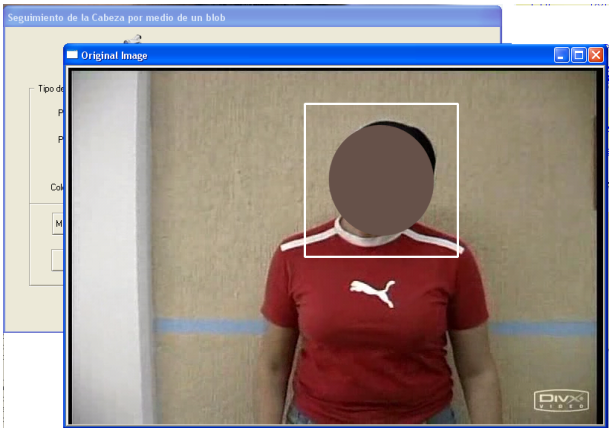


Figura C.16: Inicio del seguimiento de la persona, con la opción *Blob completo* seleccionada.

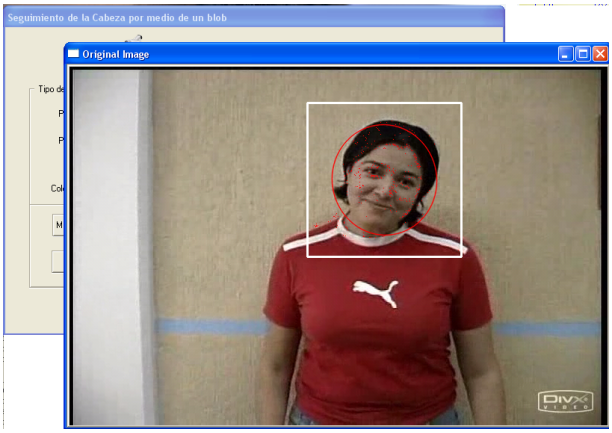


Figura C.17: Inicio del seguimiento de la persona, con la opción *Blob con píxeles* seleccionada.

Apéndice D

Aplicación para obtener los archivos de inicialización

El siguiente manual de usuario, está desarrollado con la finalidad de ayudar a quienes quieran utilizar la aplicación de apoyo que se ha construido para generar los archivos que contienen la información de los tonos de piel en el espacio de color RGB normalizados. La aplicación está dividida en tres partes, la primera se utiliza para generar archivos de inicialización de la clase de *color de piel* utilizando para ello la segmentación por crecimiento de regiones. La segunda produce los archivos de inicialización de la clase $fondo$. La tercera permite crear un archivo de inicialización apartir de datos r y g de cualquier clase, recuerde que el archivo de inicialización de una clase contiene el vector medio y la matriz de covarianza de los píxeles que la definen, y que para efectos de este proyecto sólo se consideraron los valores r y g de dichos píxeles.

Para poder trabajar con la aplicación se requiere su instalación en la computadora, para iniciar este proceso se debe ejecutar el archivo "Segmentación de Imágenes.exe", que se encuentra en el CD-ROM de la tesis (Figura D.1).

Al ejecutar el archivo aparecerá una ventana que indica el inicio de la instalación (Figura D.2(a)), para continuar se debe presionar el botón "Siguiente", después se tiene que escoger la ruta de la aplicación (Figura D.2(b)), puede elegir la ruta por $default$ o

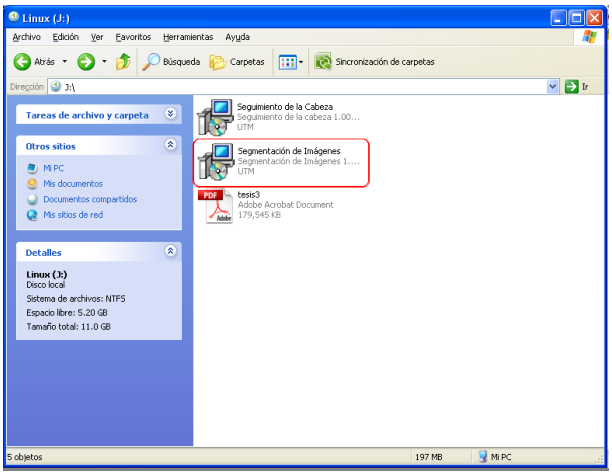


Figura D.1: Instalador de la aplicación.

bien, una ruta personalizada presionando el botón "Examinar". Una vez escogida la ruta se debe presionar "Siguiente". En la siguiente ventana se decide si se desea un acceso directo en el escritorio (Figura D.2(c)), se recomienda crear el acceso directo, para continuar se presiona "Siguiente". Ahora aparecerá una ventana indicando las opciones que se han elegido hasta el momento (Figura D.2(d)), para iniciar el proceso de instalación se debe presionar el botón "Siguiente", al realizar esta acción se abrirá una ventana que indica el avance de la instalación (Figura D.2(e)). Al terminar se mostrará un mensaje dando la opción de ejecutar la aplicación (Figura D.2(f)).

Se puede ejecutar la aplicación de dos maneras distintas: Desde Visual C++ 6.0, abriendo el espacio de trabajo "main.dsw" que se encuentra ubicado en la carpeta donde se ha instalado la aplicación previamente (Figura D.3), para mayor detalle lea el archivo "Instrucciones.txt" que se encuentra en el mismo directorio. La segunda, utilizando el acceso directo que se encuentra en el escritorio, este acceso directo fue generado durante la instalación. Cuando se inicia la aplicación se muestra una ventana principal, la cual se puede observar en la Figura D.4, esta ventana contiene cuatro botones, los cuales se describen brevemente a continuación:

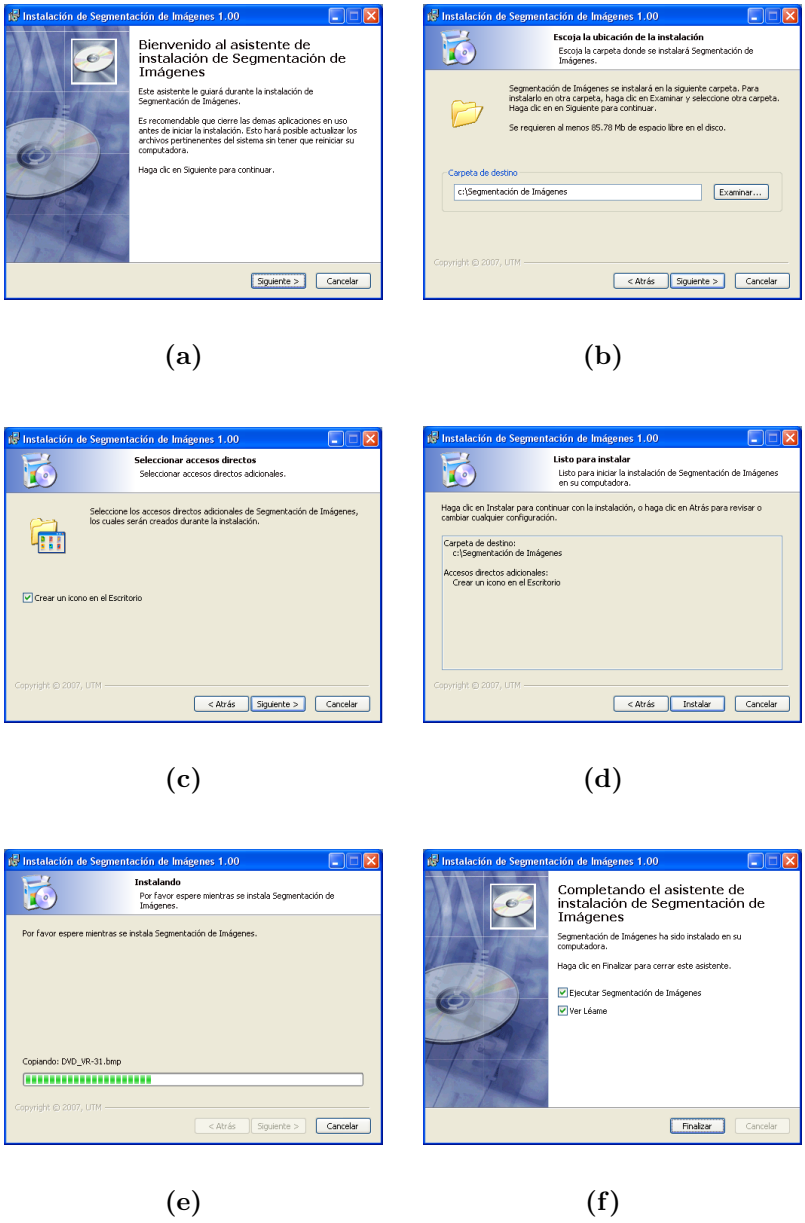


Figura D.2: Instalación de la aplicación "Segmentación de Imágenes".

- **Segmentación de Piel:** Se utiliza para generar los archivos de inicialización de la clase de *color de piel*.

- **Fondo:** Abre una ventana con herramientas para poder producir los archivos de inicialización para la clase $fondo$.

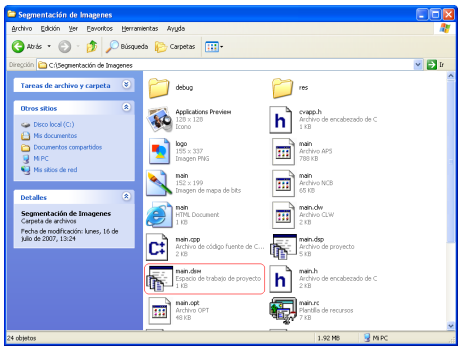


Figura D.3: Espacio de trabajo para editar el programa en Visual C++ 6.0.

- **Generar archivo de inicialización:** Se utiliza para crear un archivo de inicialización de cualquier clase a partir de datos r y g.

- **Salir:** Se utiliza para salir de la aplicación.

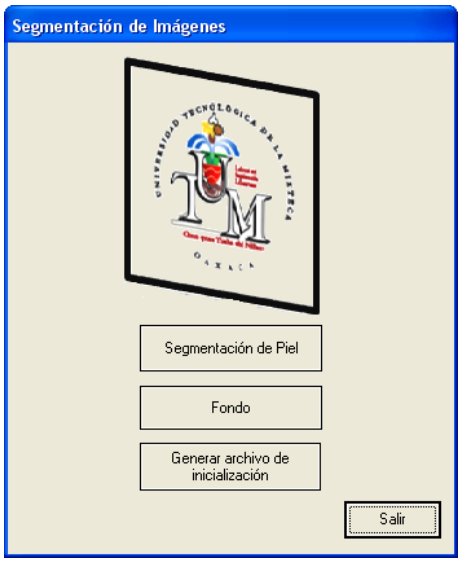


Figura D.4: Ventana principal del sistema.

En seguida se detalla cada una de las ventanas que aparecen al presionar los primeros tres botones de la ventana principal.

D.1. Segmentación de Piel

Esta opción ayuda a crear archivos de inicialización de la clase de *color de piel*, los cuales se pueden ocupar para la aplicación "Seguimiento de la cabeza por medio de un *blob*". La aplicación recibe de entrada una imagen en en formato *PNG* (Portable Network Graphics), JPG (Joint Photographic Experts Group) o BMP (BitMaP) que contiene la imagen de una persona con vista frontal. Cuando se inicia, se muestra la ventana de la Figura D.5, a continuación se describen los elementos que contiene:

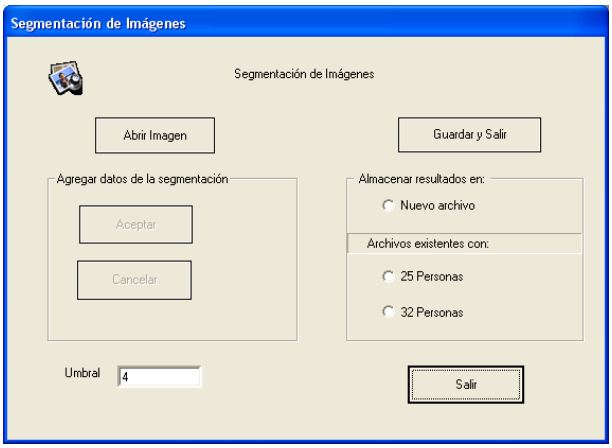


Figura D.5: Ventana de la opción "Segmentación de la piel".

- **Abrir imagen:** Este botón se utiliza para seleccionar una imagen de entrada al sistema. Cuando se presiona aparece un cuadro de diálogo (Figura D.6) donde se puede buscar alguna imagen, en formato PNG, JPG o BMP para ser segmentada.

  Al escoger alguna imagen y presionar el boton "Abrir", aparecerá una ventana nueva, la cual muestra la imagen seleccionada (Figura D.7).

  Una vez realizado esto es posible iniciar el proceso de segmentación por crecimiento de regiones, para ello se debe especificar manualmente un punto semilla sobre el área a segmentar. Cada punto semilla se elige haciendo un *click* con el botón izquierdo del ratón. A partir del punto semilla se inicia el crecimiento de la región

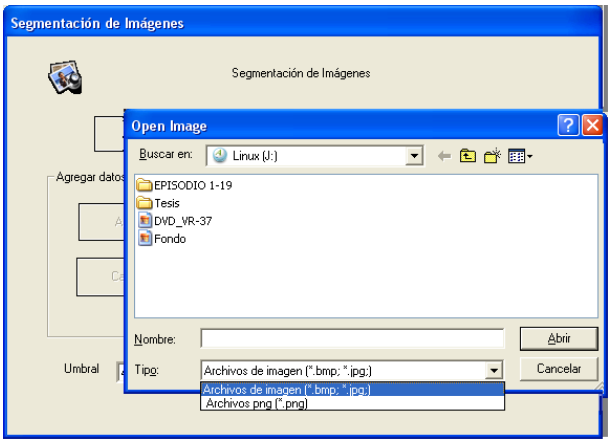


Figura D.6: Cuadro de diálogo para seleccionar alguna imagen en formato *PNG*, *JPG* o *BMP*.

Figura D.7: Ventana que muestra la imagen seleccionada.

homogénea, la cual se pintará de color verde (Figura D.8). Se puede repetir el proceso de segmentación en diferentes áreas, tantas como se requiera, utilizando el mismo proceso.

- Antes de pasar a otra imagen para realizar la segmentación, primero se le debe indicar al programa si los datos que se generaron con el proceso de la segmentación se agregan o no al archivo. Para indicarle esto existe el frame **Agregar datos de la segmentación**. En este *frame* se encuentran dos botones: El botón **Aceptar** determina que los datos que se han generado con la imagen se deben agregar al

Figura D.8: Crecimiento de la región homogénea pintada de color verde.

archivo que se obtendrá. El botón **Cancelar** borrará todos los datos generados y no se agregará nada al archivo. Al seleccionar cualquiera de las dos opciones la aplicación se encuentra lista para segmentar otra imagen o para terminar.

Se puede aplicar el mismo procedimiento antes explicado a cualquier número de imágenes, abriendo otra imagen y repitiendo los pasos mencionados anteriormente.

- Antes de salir de la aplicación, se debe especificar si los datos deben ser guardados. En el frame **Almacenar resultados en** se establece como deben ser guardados los datos generados hasta el momento, la opción **Nuevo archivo**, almacena un archivo únicamente con los datos producidos con la aplicación. Las opciones **25 personas** y **32 personas** guardan los datos obtenidos con la aplicación, agregándole los datos de los archivos de inicialización que viene con la tesis, con datos de 25 ó de 32 personas respectivamente.

- **Guardar y Salir:** Una vez especificada la forma de almacenamiento en la sección **Almacenar resultados en**, con este botón se puede guardar el archivo con los datos generados. Al presionar este botón la aplicación mostrará una ventana para seleccionar la ubicación donde se almacenará el archivo de inicialización (Figura

D.9). Debe indicar una extensión para el nombre del archivo. Al mismo tiempo se genera otro archivo con el mismo nombre dado por el usuario al archivo de inicialización, pero al final del nombre del archivo se le agregará "_P" para indicar que el archivo pertenece a la clase de color de piel. Este nuevo archivo contiene los datos de r y g de los píxeles que generaron al vector medio y a la matriz de covarianza, por lo que mediante los datos contenidos en este archivo es posible generar un archivo de inicialización como se explicará en la sección D.3. Este nuevo archivo se almacena en el mismo directorio que fue especificado por el usuario para almacenar el archivo de inicialización.

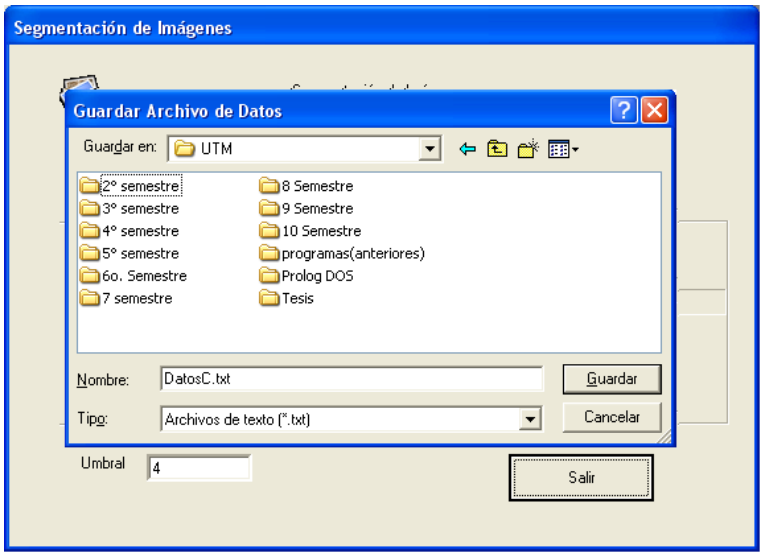


Figura D.9: Cuadro de diálogo para guardar el archivo de inicialización que contiene el vector medio y la matriz de covarianza.

Si no se ha seleccionado ninguna opción en la sección **Almacenar resultados en** antes de presionar este botón, el sistema mostrará un mensaje de error (Figura D.10).

- **Umbral:** Esta opción se incluye para permitir al usuario poder variar el umbral que se utiliza al realizar el algoritmo. Por $Default$ el valor es de 4, que es la

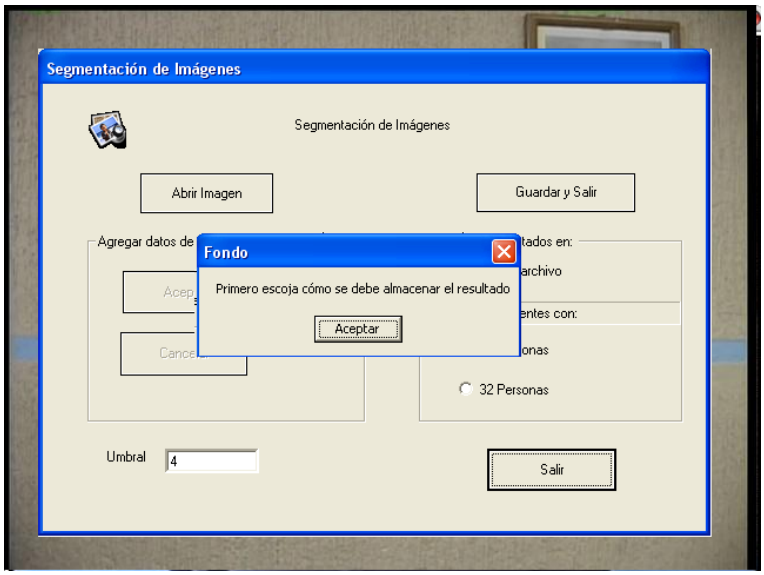


Figura D.10: Mensaje de error debido a que no se ha seleccionado cómo se guardará el resultado de la segmentación.

cantidad con la que se generaron los archivos de inicialización del sistema de seguimiento. Para indicar un nuevo valor se requiere posicionarse en el cuadro de texto de umbral que aparece en la ventana e ingresar algún número positivo (ver figura D.11).

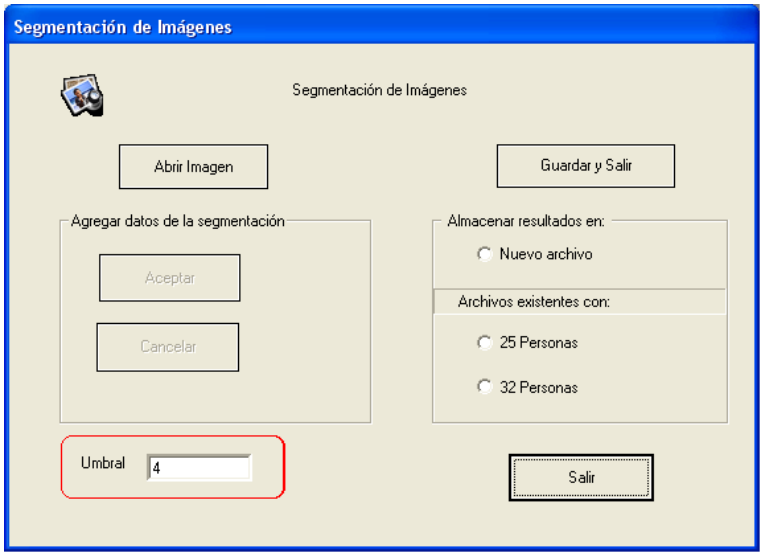


Figura D.11: Opción para modificar el umbral del algoritmo.

Si por error se ha introducido un número negativo o inválido, se mostrará un mensaje de error (figura D.12).

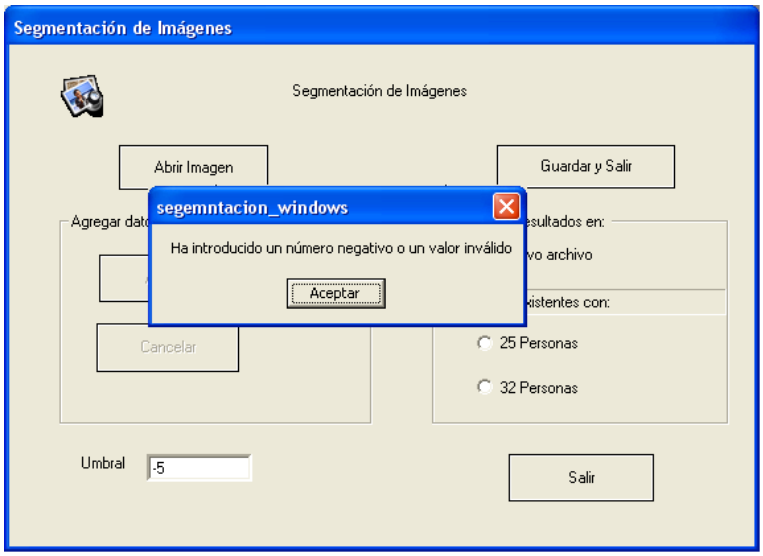


Figura D.12: Mensaje de error al introducir un número negativo o inválido en la opción umbral.

- **Salir:** Este bóton se utiliza para salir de la aplicación sin guardar los datos.

D.2. Fondo

Esta opción está desarrollada con el fin de ayudar a crear archivos de inicialización de la clase $fondo$. Esta aplicación recibe de entrada una imagen en formato PNG (Portable Network Graphics), JPG (Joint Photographic Experts Group) o BMP (BitMaP) que contiene alguna escena sin ninguna persona contenida en ella. Esta aplicación no genera ninguna segmentación, ya que toma en cuenta todos los píxeles de la imagen para generar el archivo de salida.

Cuando se inicia, se muestra la ventana de la Figura D.13, a continuación se describen los elementos que contiene:

- **Abrir imagen:** Este botón se utiliza para seleccionar una imagen de entrada al

Figura D.13: Ventana de la opción "fondo".

sistema. Cuando se presiona aparece un cuadro de diálogo (Figura D.14) donde se puede buscar alguna imagen, en formato PNG, JPG o BMP.

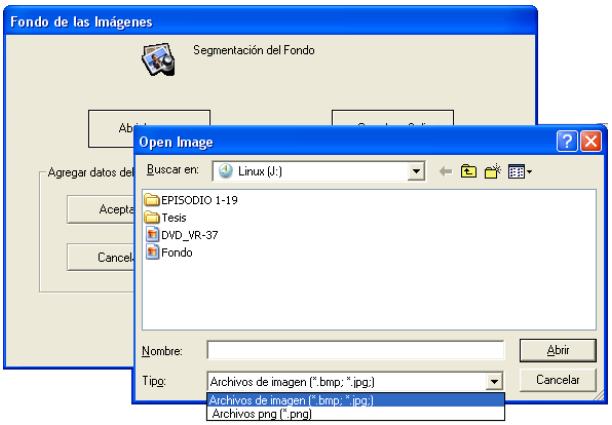


Figura D.14: Ventana para seleccionar alguna imagen en formato *PNG*, *JPG* y *BMP*.

Al escoger alguna imagen y presionar el botón "Abrir", aparecerá una ventana nueva, la cual muestra la imagen elegida (Figura D.15). No es necesario realizar alguna operación, la aplicación tomará todos los píxeles de la imagen para agregarlos al archivo.

- Antes de pasar a otra imagen, se debe especificar si se agregarán los datos de la imagen actual, para realizar esto existe el frame **Agregar datos del fondo**. En esta sección se encuentran dos botones: El botón **Aceptar** indica que todos los

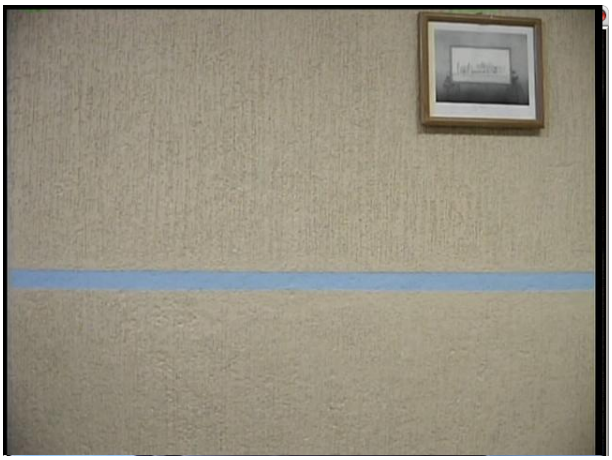


Figura D.15: Ventana que muestra la imagen seleccionada.

datos se deben añadir al archivo, al terminar de incorporar los datos al archivo, se mostrará una ventana que indica que el proceso ha terminado (Figura D.16).

Figura D.16: Mensaje indicando que ha terminado de agregar los datos al archivo.

El botón **Cancelar** descarta todos los datos de la imagen actual. Al seleccionar cualquiera de las dos opciones, la aplicación se encuentra lista para agregar datos de otra imagen o para terminar.

- El frame **Almacenar resultados en** permite especificar como deben ser guardados los datos generados hasta el momento, la opción **Nuevo archivo**, guarda un

archivo únicamente con los datos obtenidos con la aplicación. La opción **Archivo existente** almacena los datos producidos por la aplicación, agregándole los datos del archivo de inicialización que viene con la tesis y que contiene la información de 32 fondos.

- **Guardar y Salir:** Una vez especificada la forma de almacenamiento en el frame **Almacenar resultados en**, con este botón se puede guardar el archivo con los datos obtenidos. Al presionar este botón, la aplicación mostrará una ventana para seleccionar la ubicación donde se almacenará el archivo de inicialización (Figura D.17). Debe indicar una extensión para el nombre del achivo. Al mismo tiempo se genera otro archivo con el mismo nombre dado por el usuario al archivo de inicialización del fondo, pero al final del nombre del archivo se le agregará "_F" para indicar que el archivo pertenece a la clase $fondo$. Este nuevo archivo contiene los datos de r y g de los píxeles que generaron al vector medio y a la matriz de covarianza, por lo que mediante los datos contenidos en este archivo es posible crear un archivo de inicialización como se explicará en la sección D.3. Este nuevo archivo se almacena en el mismo directorio que fue especificado por el usuario para almacenar el archivo de inicialización.

  Al terminar de normalizar y generar los archivos se mostrará un mensaje indicando que el proceso ha terminado y que la ventana se va a cerrar (Figura D.18).

  Si no se ha seleccionado ninguna opción en la sección **Almacenar resultados en** antes de presionar este botón, el sistema mostrará un mensaje de error (Figura D.19).

- **Salir:** Este bóton se utiliza para salir de la aplicación sin guardar los datos.

Figura D.17: Cuadro de diálogo para guardar el archivo de inicialización que contiene el vector medio y la matriz de covarianza, del fondo.

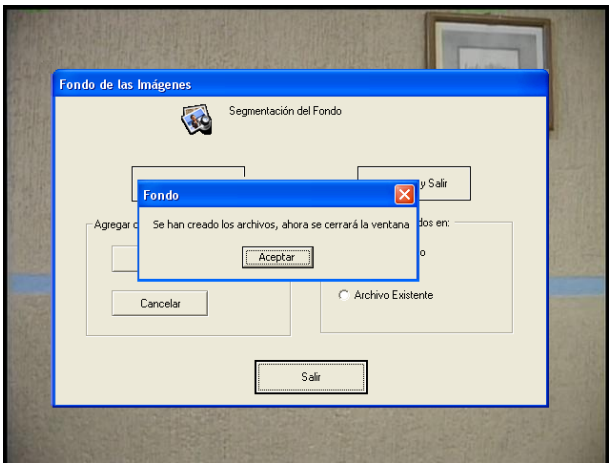


Figura D.18: Mensaje indicando que han terminado todos los procesos y se cerrará la ventana.

D.3. Generar archivo de inicialización

Este botón se utiliza para generar un archivo que contenga el vector medio y la matriz de covarianza, que se usa para la inicialización de la aplicación "Seguimiento de la cabeza por medio de un blob". El archivo de entrada es un archivo de texto que debe contener dos columnas de datos decimales, los cuales representan valores r y g del espacio de color RGB normalizado. Este archivo, como ya se explicó, se obtiene al utilizar cualquiera de las dos opciones anteriores: "Segmentación de imágenes" o "Fondo".

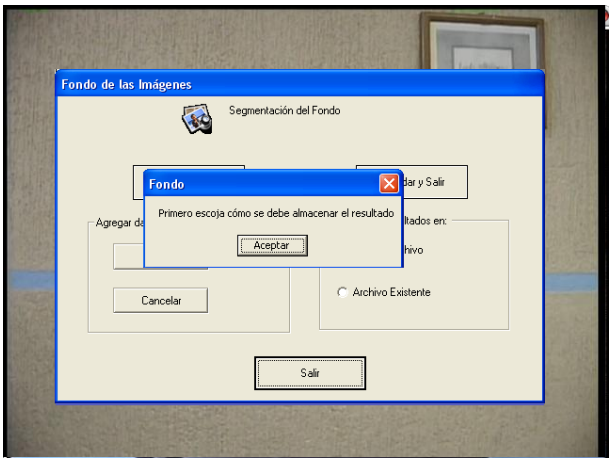


Figura D.19: Mensaje de error debido a que no se ha seleccionado cómo se guardará el resultado de la segmentación.

Al presionar el botón aparecerá una ventana donde se debe elegir al archivo de entrada (Figura D.20), una vez que se ha seleccionado el archivo, se debe presionar "Abrir". La aplicación realizará los cálculos correspondientes y al terminar mostrará una ventana para especificar el nombre y la ubicación del archivo de salida (Figura D.21).

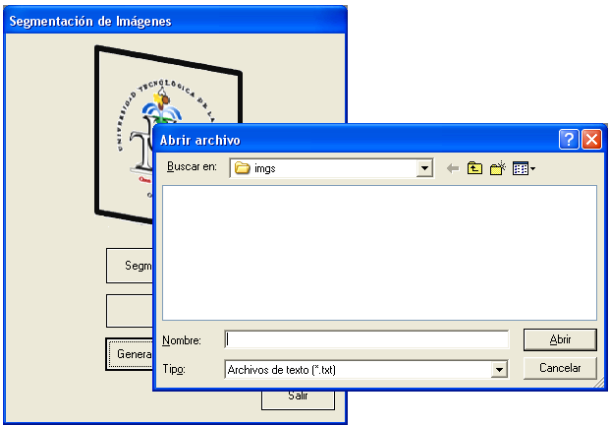


Figura D.20: Ventana para seleccionar el archivo de entrada.

Para finalizar se debe presionar el botón "Guardar", al hacerlo se mostrará un cuadro de diálogo que indicará que el archivo se ha creado exitosamente (Figura D.22).

Como puede verse en esta sección, es posible generar diferentes archivos de inicialización para una clase determinada (ya sea la clase de color de piel o la clase fondo,

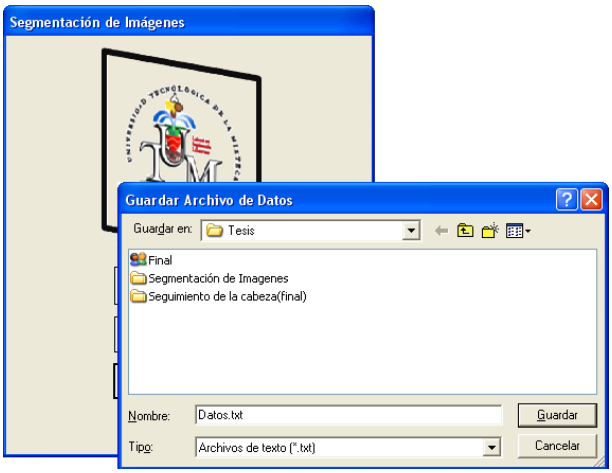


Figura D.21: Ventana para indicar el nombre y la ubicación del archivo que contendrá al vector medio y la matriz de covarianza.

Figura D.22: Mensaje indicando que se ha creado el archivo de inicialización.

según se requiera) a partir de los archivos que contienen los valores r y g de los píxeles que la definieron, como se recordará estos archivos se producen al mismo tiempo que se crean los archivos de inicialización (secciones D.1 y D.2). Si se desea se pueden combinar los valores r y g de varios de estos archivos utilizando cualquier editor de texto.

Bibliografía

[1] BAÉZ ROJAS J.J., GUERRERO M.L., CONDE ACEVEDO J., URCID SERRANO G., PADILLA VIVANCO A., Segmentación de imágenes de color, Revista Mexicana de Física, 50(6):579-587, 2004.

[2] BROOKS R. A., The Intelligent room project, In Proceedings of the Second International Cognitive Technology , August 1997. **Disponible en:** http://people.csail.mit.edu/brooks/papers/aizu.pdf. [Último acceso 10/07/07]

[3] BURDEA G.C., COIFFET P., Virtual reality technology, Wiley-Interscience, Second edition, New Jersey, 2003.

[4] CASTLEMAN K. R., Digital image processing, Prentice Hall, 1996.

[5] CHANG A., A help system for the intelligent room, Proc. Student Oxygen Workshop, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2001. **Disponible en:** http://sow.csail.mit.edu/2001/proceedings/changa.pdf. [Último acceso 10/07/07]

[6] ESSA I. A., Coding, analysis, interpretation, and recognition of facial expressions, Technical Report GIT-GVU-98-23, College of Computing, Georgia Institute of Technology, Graphics, Visualization and Usability Center (GVU Center), Au-

gust 1998. **Disponible en:** http://smartech.gatech.edu/bitstream/1853/3468/1/98-23.pdf. [Último acceso 10/07/07]

[7] FREIXENET J., MUÑOZ X., RABA D., MARTÍ J., CUFÍ X., Yet another survey on image segmentation: Region and boundary information integration, Nielsen M., Johansen P., Heyden A., Sparr G., editor, Proceedings of the 7th European Conference on Computer Vision (ECCV 2002), Part III, 408-422, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002.

[8] FREY D., Mahalanobis taguchi system, Lecture Notes of the course "Robust System Design", Graduate Course for the Aeronautics and Astronautics Engineering, MIT's OpenCourseWare, 1998. **Disponible en:** http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-881Robust-System-DesignSummer1998/523AE239-A747-495F-8B7B-0AD80478E420/0/l16_mahalanobis2.pdf. [Último acceso 10/07/07]

[9] FUENTES L. M., VELASTIN S. A., Vigilancia avanzada: del *tracking* a la detección de sucesos, Revista Transactions IEEE América Latina, 2(3), September 2004. **Disponible en:** http://ieeexplore.ieee.org/iel5/9907/31503/01468654.pdf?arnumber=1468654. [Último acceso 10/07/07]

[10] GONZÁLEZ R. C., WOODS R. E., Tratamiento digital de imágenes, Addison-Wesley, USA, 1996.

[11] GUERRERO RAMÍREZ M. L. A., Segmentación de imágenes digitales en color modificado, Tesis de Maestría, Maestría en Electrónica y Computación, Universidad Tecnológica de la Mixteca, 2001.

[12] HAMPTON C., PERSONS T., WYATT C., ZHANG Y., Survey of image segmentation, 1998. **Disponible en:** http://citeseer.ist.psu.edu/cache/papers/cs/7550/http:zSzzSzwww.rad.bgsm.eduzSz~timzSzpaperszSzseg_intro.pdf/hampton98survey.pdf. [Último acceso 10/07/07]

[13] HAYKIN S., Neural networks: A comprehensive foundation, Second edition, Prentice Hall, 1999.

[14] HUANG T. S., TAO H., Visual face tracking and its application to 3D model-based video coding, Picture Coding Symposium 2001, 57-60, 2001. **Disponible en:** http://www.soe.ucsc.edu/~tao/pps/PCS01.pdf. [Último acceso 10/07/07]

[15] JAIN A. K., Fundamentals of digital image processing, Prenctice Hall, USA, 1989.

[16] JAIN R., KASTURI R., SCHUNCK B. G., Machine Vision, McGraw-Hill International Editions, Computer Science Series, Singapur, 1995.

[17] JEZEKIEL B., Project Report: Human activity scripts and queries for video Databases, award number: 0208300, HTML, 06/01/2002-05/31/2005. **Disponible en:** http://www.cs.dartmouth.edu/~idm2004/online/0208300/0208300.html. [Último acceso 10/07/07]

[18] LA CASCIA M., SCLAROFF S., ATHITSOS V., Fast, reliable head tracking under varying illumination: an approach based on registration of texture-mapped 3D models, IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI), 22(1), 322-336, April 2000. **Disponible en:** http://www.cs.bu.edu/groups/ivc/pubs/pami.22.4.pdf. [Último acceso 10/07/07]

[19] LIRA J., La percepción remota: Nuestros ojos desde el espacio (documento en línea), Cuarta impresión, 1995. **Disponible en:**

http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen1/ciencia2/33/htm/sec_4.html. [Último acceso 23/08/07]

[20] LU S., TSECHPENAKIS G., METAXAS D. N., JENSEN M. L., KRUSE J., Blob analysis of the head and hands: A method for deception detection. Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), 2005. **Disponible en:** http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/01/22680020c.pdf. [Último acceso 10/07/07]

[21] MANRESA-YEE C., VARONA J., RIBOT T., PERALES F. J., Non-verbal communication by means of head tracking, Brunet P., Correia N., Baranoski G, editor, Ibero-American Symposium on Computer Graphics (SIACG 2006), July 2006. **Disponible en:** http://dmi.uib.es/∼ugiv/papers/non_verbal_communication.pdf. [Último acceso 10/07/07]

[22] MARTÍNEZ GÓMEZ L. A., Sistema de visión para equipos de robots, Tesis de licenciatura en Ingeniería en Computación, Instituto Tecnológico Autónomo de México, 2004.

[23] MAXWELL B. A., SHAFER S. A., Physics-based segmentation: looking beyond color, Proceedings of Image Understanding Workshop, 1996. **Disponible en:** http://www.palantir.swarthmore.edu/maxwell/papers/pdfs/Maxwell-IUW-1995.ps.gz. [Último acceso 10/07/07]

[24] MORROW-TESCH J., DAILEY J. W., JIANG H., A video data base system for studying animal behavior, Journal of Animal Science, 76(10):2605-2608, 1998. **Disponible en:** http://jas.fass.org/cgi/reprint/76/10/2605.pdf. [Último acceso 10/07/07]

[25] PENTLAND A. P., Smart rooms: Machine understanding of human behavior. Pentland A. P. Cipolla R., editor, Computer Vision for Human-Machine Interaction, Cambridge University Press, United Kingdom, 1998.

[26] PRESS W. H., TEUKOLSY S. A., VETTERLING W. T., FLANNERY B. P., Numerical Recipes in C: The Art of Scientific Computing, Second edition, Cambridge University Press, 1992, Reprinted with corrections 2002.

[27] QUINTANA DUQUE J. C., Síntesis animada de voz visual a partir del video de un rostro, Tesis de licenciatura de Ingeniería en Electrónica, Pontificia Universidad Javeriana, Facultad de Ingeniería, Bogotá, 2005. **Disponible en:** http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis108.pdf. [Último acceso 10/07/07]

[28] RICHARDS J. A., Remote sensing digital image analysis: An introduction, Springer-Verlag New York, Inc., Second edition, 1993.

[29] SCHOLD A., HARO A., ESSA I. A., Head tracking using a textured polygonal model, Technical Report GIT-GVU-98-24, College of computing, Georgia Institute of Technology, Graphics, Visualization and Usability Center (GVU Center), August 1998. **Disponible en:** ftp://ftp.cc.gatech.edu/pub/gvu/tr/1998/98-24.pdf. [Último acceso 10/07/07]

[30] SPARACINO F., (Some) computer vision based interfaces for interactive art and entertainment installations, INTER_FACE Body Boundaries, issue editor Emanuele Quinz, Anomalie, Anomos, (2), 2001. **Disponible en:** http://alumni.media.mit.edu/∼flavia/Papers/Flavia_isea2000.pdf. [Último acceso 10/07/07]

[31] SUETENS P., FUA P., HANSON A. J., Computational strategies for object recognition, Association for Computing Machinery (ACM) Computing Surveys, 24(1):5-62, New York, NY, USA, 1992.

[32] TANAWONGSUWAN R., STOYTCHEV A., ESSA I. : Robust tracking of people by a mobile robotic agent, Technical Report GIT-GVU99-19, Georgia Institute of Technology, 1999. **Disponible en:** http://www.cs.iastate.edu/∼alex/papers/GIT-GVU-99-19.pdf. [Último acceso 10/07/07]

[33] VARONA X., BUADES J. M., PERALES F. J., Hands and face tracking for VR applications, International Journal of Systems & Applications in Computer & Graphics, 29(2):179-187, 2005. **Disponible en:** http://dmi.uib.es/∼ugiv/papers/CGForum04.pdf. [Último acceso 10/07/07]

[34] WANG Y., VAN DYCK R. E., DOHERTY J. F., Tracking moving objects in video sequences, Proc. Conference on Information Sciences and Systems (CISS), 2000, **Disponible en:** http://w3.antd.nist.gov/pubs/ciss00.pdf. [Último acceso 10/07/07]

[35] WREN C. R., Pfinder: Real-time tracking of the human body, Master's degree thesis, Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, 1996.

[36] WREN C. R., AZARBAYEJANI A., DARRELL T, PENTLAND A. P., Pfinder: Real-time tracking of the human body, IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 19(7): 780-785, 1997.

[37] YANG J., LU W., WAIBEL A., Skin-color modeling and adaptation, Technical Report CMUCS-97-146, Carnegie Mellon University, 1997. **Disponible en:** http://citeseer.ist.psu.edu/cache/papers/cs/3409/

http:zSzzSzreports-archive.adm.cs.cmu.eduzSzanonzSz1997zSzCMU-CS-97-146.pdf/yang97skincolor.pdf. [Último acceso 10/07/07]