UNIVERSIDAD CENTROCCIDENTAL LISANDRO ALVARADO
DECANATO DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE SISTEMAS

SISTEMA DE RECONOCIMIENTO DE ROSTROS PARA MAGGIE

DRA. MARITZA BRACHO DE RODRÍGUEZ

Barquisimeto, Noviembre 2008

UNIVERSIDAD CENTROCCIDENTAL LISANDRO ALVARADO
DECANATO DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE SISTEMAS

SISTEMA DE RECONOCIMIENTO DE ROSTROS PARA MAGGIE

por

Dra. Maritza Bracho de Rodríguez

Trabajo de Ascenso para Optar
a la Categoría de Asociado
en el Escalafón del Personal Docente y de Investigación

Barquisimeto, Noviembre 2008

DEDICATORIA

A Dios.

A Mi Padre, Jesús Bracho Barreto.

A Mi Esposo, Lucio Antonio Rodríguez Verde.

AGRADECIMIENTOS

Al Dr. **Miguel Ángel Salichs Sánchez-Caballero**, quien me ha permitido incorporarme a su equipo de trabajo durante el desarrollo de este proyecto, brindándome adicionalmente, su amistad y confianza.

A mi Amiga, MSc. **Ana del Valle Corrales Paredes**, por su alegría, compañía y fortaleza.

A mis Amigos del Robotics Lab: **Álvaro Castro, Ramón Barber, Javier Gorostiza, Fernando Alonso, María Malfaz, Enrique Tomas y David García**.

A la MSc. **Briceida Salinas**, Presidenta de FUNDACITE Lara, por su apoyo incondicional.

A la **Universidad Centroccidental Lisandro Alvarado**, por haberme permitido el logro de esta meta.

A Todos Ellos, Muchas Gracias.

Este trabajo ha sido desarrollado con apoyo financiero
de las siguientes Instituciones:

Universidad Centroccidental Lisandro Alvarado

Fondo Nacional de Ciencia, Tecnología e Innovación
adscrito al Ministerio de Ciencias y Tecnología
República Bolivariana de Venezuela

Fundación para el Desarrollo de la Ciencia y
la Tecnología del Estado Lara

Robotics Lab
Universidad Carlos III de Madrid

ÍNDICE GENERAL

Capítulo Página

LISTA DE FIGURAS

Figura Página

LISTA DE TABLAS

Tabla Página

RESUMEN

Este trabajo ha sido realizado en el contexto de la Robótica, particularmente en el campo de Visión Artificial. Su aporte fundamental es el desarrollo de un sistema de reconocimiento de rostros para Maggie, robot autónomo personal diseñado y construido en el Robotics Lab de la Universidad Carlos III de Madrid. El sistema ha sido integrado a la arquitectura de control, denominada AD, como un conjunto de habilidades que le permiten al robot ejecutar en tiempo real y en forma autónoma las siguientes acciones: detectar los rostros de la personas en escenas visuales, tomar imágenes de los rostros segmentados, construir modelos de los rostros a partir de estas imágenes, almacenar en bases de datos tanto las imágenes tomadas, como los modelos obtenidos de los rostros y usar la información de los rostros capturados en las imágenes para reconocer personas a través de la comparación de determinados características, propiedades y rasgos de la imagen facial, con los almacenados en la base de datos. Con estas habilidades Maggie está en capacidad de tomar 124 imágenes del rostro de una persona y almacenarlas en memoria a largo plazo en 1,43 minutos, con una efectividad del 73,20%. Además, puede hacer hasta 162 reconocimientos de una persona en 1,50 minutos, siempre y cuando la imagen del rostro esté almacenada en la base de datos correspondiente.

Capítulo 1

INTRODUCCIÓN

1.1 Motivación

Uno de los mecanismos de percepción más poderosos que pueden ser aplicados a la robótica autónoma es el de visión artificial. Generalmente los sistemas de visión artificial intentan reproducir ciertas funciones hasta ahora atribuibles a organismos biológicos. Su propósito fundamental es el de programar computadores de tal forma que puedan entender escenas visuales o reconocer y comprender las características de una imagen.

Entre las funciones que se han intentado reproducir en los sistemas de visión artificial se encuentra la detección, segmentación, localización y reconocimiento de objetos en imágenes, particularmente la capacidad de reconocimiento de rostros presente en ciertos organismos biológicos. El rostro humano es la parte del cuerpo que nos permite reconocer a una persona y detectar sus emociones. A pesar de que la mayoría de las caras tienen una composición y estructura similar, existe una variación considerable de un individuo a otro.

El reconocimiento de rostros es una habilidad innata de los seres humanos que les permite distinguir caras tanto familiares como desconocidas y clasificarlas de acuerdo al sexo, edad y estado emocional. La investigación en esta área ha sido conducida por más de 40 años, en consecuencia existe una gran cantidad de estudios en psicología y neurobiología para explicar como funciona este proceso. En 1888, [4], Francis Galton propuso por primera vez un método formal para la clasificación de rostros, basado en la colección de perfiles faciales como curvas, a los cuales encontraba su norma y luego clasificaba otros perfiles a partir de sus desviaciones con respecto a esta norma. La clasificación era multimodal y se conformaba un vector resultante que podía ser comparado con otos vectores almacenados en una base de datos.

El reconocimiento facial es un tema muy estudiado por la visión computacional. A partir de Bledsoe, [6], Sakai, Nagao, Kanade, [31], Harmon, [13] y Kanade, [15], han

surgido gran número de estrategias, tanto automáticas como semiautomáticas para desarrollar sistemas de reconocimiento facial. En los últimos años este problema ha atraído la atención de muchos investigadores, sin embargo continúa abierto a la investigación debido a que, en el reconocimiento de rostros, influyen factores como: el estado de ánimo de la persona, la edad, la variación en el peso, la etnia, lo cual complica dicho proceso.

Un sistema de reconocimiento facial debe permitir la identificación en forma automática de una persona en una imagen digital. El problema puede ser formulado de la manera siguiente, [39]: Dado un video o un conjunto de imágenes, identificar o verificar la presencia de una o más persona en la escena usando una base de datos con imágenes de rostros almacenadas. Información colateral como etnia, edad, género, expresión facial y forma de hablar puede ser suministrada para limitar la búsqueda y aumentar el reconocimiento. La solución del problema requiere de la segmentación o detección de la cara, en escenas abarrotadas de objetos, la extracción de características de la región facial y finalmente el reconocimiento o verificación. En problemas de identificación, la entrada al sistema es una cara desconocida y el sistema reporta la identidad desde una base de datos de individuos conocidos, mientras que en los problemas de verificación, el sistema requiere confirmar o rechazar la identidad reclamada.

Centrándose en los aspectos funcionales, un sistema de visión artificial tradicional está constituido por las siguientes unidades: adquisición, procesamiento, segmentación, extracción de características, representación y clasificación o reconocimiento, [10]. Cada etapa o módulo cumple con una función específica dentro del proceso. En [4], Barret señala que el problema del reconocimiento facial puede dividirse en etapas como: 1) La identificación de la imagen del rostro en la escena, proceso denominado segmentación o detección de caras. 2) El reconocimiento facial: comparación del rostro con una base de datos en la cual se almacenan imágenes de caras existentes. En [39] se establece, que sin importar cual sea la formulación del problema, tres tareas principales deben ser ejecutadas en el reconocimiento de rostros: 1) detección del rostro; 2) extracción de características; 3) reconocimiento de la cara.

1.1.1 Detección de Caras en Imágenes

Consiste en localizar la cara o rostro de la persona dentro de una escena. Sin embargo, la segmentación no es necesariamente una tarea simple: las imágenes de muchos objetos comunes se parecen a las imágenes de los rostros y una vez segmentados pueden ser rechazados posteriormente, en la etapa de reconocimiento. Esta tarea requiere de un modelo para discriminar las caras de otros objetos.

Numerosos enfoques han sido aplicados en la ejecución de esta tarea, tales como, el uso de plantillas de rostros, patrones basados en características de la cara, color de la piel, redes neuronales, entre otros. Durante la última década métodos confiables han sido desarrollados para manejar el problema de la segmentación de múltiples caras en fondos complejos, donde del rostro puede estar parcialmente oculto, girado en el plano o rotado en profundidad. Los enfoques más conocidos están basados en redes neuronales, [30], aprendizaje basado en ejemplos, [34], autocaras o eigenfaces, árboles de decisión, clasificadores Bayesianos y máquinas de soporte vectorial, [28]. En [38], se encuentra un estudio actualizado en el cual se analizan diferentes métodos que se han desarrollado para la detección de caras y se encuentran numerosas referencias a la literatura especializada en el tema.

1.1.2 Extracción de Características

Para simplificar la comparación y/o búsqueda en una base de datos, una imagen debe ser transformada en un sistema de coordenadas de baja dimensión que preserve la calidad de percepción de la imagen, con frecuencia se seleccionan las características más representativas. Por lo tanto, esta etapa consiste en hacer los ajustes necesarios para el procesamiento de la imagen de la cara: extracción de información relevante, detección de componentes y su codificación, transformaciones lineales, rotación, escalamiento. Si los ojos y la boca pueden ser localizados, estos puntos de referencia pueden ser usados para manejar la normalización y producir una imagen facial estandarizada

Según Valentín, Abdi, O´toole y Cotrell, [36], los sistemas de codificación de caras pueden clasificarse en sistemas basados en componentes, no-conexionistas, basados en representación de las imágenes faciales, conexionistas. Los primeros representan las caras en términos de distancias, ángulos y áreas determinadas, como el de Kanade, [15], quien utilizó componentes geométricos para extraer la posición relativa y

parámetros distintivos de ojos, boca, nariz y barbilla. Los modelos conexionistas utilizan la intensidad de los píxeles. Una ventaja de utilizar modelos conexionistas es que la información es obtenida automáticamente de la estructura estadística de las caras y se evita el problema de seleccionar componentes individuales. Los métodos más usados están basados en el análisis de componentes principales, PCA o autocaras o eigenfaces, [35], análisis de componentes independientes, ICA, [3], métodos basados en redes neuronales, [12], [16], [17], modelos ocultos de Markov, [33], comparación de patrones, [7], y análisis bayesiano, [21].

1.1.3 Reconocimiento de Caras

El reconocimiento de imágenes faciales, también denominado reconocimiento de caras, permite determinar la identidad de una persona, al comparar una imagen de su cara con imágenes de referencia almacenadas en una base de datos, en la que también se almacena la identidad de las personas asociadas a cada imagen de referencia. Esta comparación se realiza analizando elementos estructurales presentes en las caras. Esto es, se extrae la información relevante de una cara, codificándola lo más eficientemente posible y se compara dicha codificación con una base de datos de otros modelos codificados de forma similar.

El reconocimiento facial es una correspondencia de 1 a muchos donde la verificación facial es una correspondencia más simple 1 a 1. Existen principalmente cuatro técnicas de reconocimiento facial: autocaras, caras propias o eigenfaces, análisis de características, redes neuronales y procesamiento facial. Una amplia gama de referencias a estas técnicas pueden encontrarse en [20] y en el estudio publicado en [39].

En el método de las autocaras, básicamente, una base de datos contiene un gran número de patrones faciales. Cuando una cara es leída se crea una correspondencia con los patrones faciales de manera que el programa pueda mezclar los diferentes patrones faciales para reproducir una cara. Entonces esto se usa a posteriori para realizar reconocimientos y verificaciones. Las técnicas de autocaras se aplican en situaciones suficientemente controladas como una fotografía de un carné, no a imágenes de baja calidad tomadas por una cámara de seguridad. El método de redes neuronales, usa un patrón facial, el cual es introducido a una red neuronal que intenta identificarlo contra su

base de datos, o verificarlo contra el patrón que el usuario tiene registrado. Si se produce un falso positivo o un rechazo la red neuronal modifica sus pesos para ayudar a mejorar el reconocimiento en los siguientes intentos.

1.2 Objetivos

A partir de las consideraciones escritas en los párrafos anteriores puede definirse entonces, como objetivo general de este proyecto, el de realizar un sistema de reconocimiento de facial para ser implementado en el robot personal Maggie, de tal forma que le permita ejecutar en forma autónoma las siguientes acciones: detectar los rostros de la personas en escenas visuales adquiridas en tiempo real, tomar imágenes de los rostros detectados, almacenar estas imágenes en bases de datos y luego usar estos datos para reconocer personas a través de la comparación de determinados características, propiedades y rasgos faciales de la imagen facial, con los almacenados en la base de datos.

Este objetivo general requiere del cumplimiento de los siguientes objetivos específicos:

- Estudiar y analizar la arquitectura de hardware y software actualmente instalado en el robot, con el fin de determinar los requerimientos para el sistema de reconocimiento facial.

- Partiendo del análisis de requerimientos, examinar, comparar y evaluar los diferentes kits de desarrollo de software, para reconocimiento de rostros que existen en el mercado para seleccionar el que pueda ser implementado en Maggie, en correspondencia con su estructura, sus componentes de hardware y a su sistema operativo.

- Diseñar la arquitectura de software más apropiada para implementar en el robot el sistema para reconocimiento de rostros. Desarrollar las aplicaciones necesarias para incorporar estas capacidades a la arquitectura de control del robot.

- Implementar físicamente en el robot y en tiempo real, el sistema para el reconocimiento de rostros.

1.3 Kits de Software para el Desarrollo de Sistemas de Reconocimiento Facial

Aunque la visión es un sensor muy importante para la robótica autónoma, su aplicabilidad es bastante complicada debido a los altos requerimientos de procesamiento: es necesario revisar un largo flujo de datos de entrada y ejecutar complejos algoritmos para generar información sensorial de alto nivel en el sistema.

Un gran número de compañías se han dedicado al desarrollo de software para visión artificial, mientras que un grupo pequeño se ha especializado en el diseño e implementación de sistemas para el reconocimiento de rostros. Un resumen de éstas puede encontrase en [39], [1] y en [11].

Para cumplir con el objetivo propuesto en este proyecto se efectuó una recopilación de datos sobre aplicaciones y conjuntos de herramientas para el desarrollo de sistemas de reconocimiento facial ofrecidos en el mercado. Posteriormente, esta información fue clasificada considerando el sistema operativo sobre el cual funcionan. El resultado es mostrado en las Tablas No 1.1, 1.2, 1.3 y 1.4, denominadas Software para Reconocimiento Facial. Estas tablas contienen una fila para cada proveedor y dos columnas, en las cuales se encuentra el nombre de la compañía, el URL correspondiente y una breve descripción de las características del software.

Para seleccionar el software de soporte para el desarrollo de las habilidades para el reconocimiento de rostros fueron considerados cinco atributos principales:

- Compatibilidad del kit con el sistema operativo Linux instalado en el robot.

- Factibilidad de instalación y viabilidad para poner en funcionamiento el kit en la arquitectura de hardware y software del robot.

- Capacidad de operación y respuesta en tiempo real.

Tabla No 1.1

Software para Reconocimiento Facial que Funciona sobre Sistema Operativo No Identificado

Nombre y URL de la Herramienta	Características
Animetrics http://www.animetrics.com/	• Reconocimiento facial incorporado al hardware • Conjunto de aplicaciones diseñadas para analizar y manipular bases de datos de rostros e imágenes faciales tomadas por cámaras en cualquier instante del tiempo. • Incluye un SDK para el reconocimiento facial, análisis y visualización desde cualquier fuente disponible
Bioscript http://www.bioscrypt.com/	• Identificación mediante huellas dactilares • Lector de caras 3D • Integrado en el hardware
L1 Identity Solutions http://www.l1id.com/	• Incluye hardware y software para el reconocimiento de rostros
Organix IT http://www.organixit.com/svms_en/index.html	• Captura y reconocimiento de caras (Funcionalidades extras) • Sistema de monitorización y vigilancia inteligente por video

Tabla No 1.2

Software para Reconocimiento Facial que Funciona sobre Sistema Operativo Windows

Nombre y URL de la Herramienta	Características
Active Face http://www.activeface.net/	• Sistema para reconocimiento de rostros. • URL no está activo en Septiembre 2008
Ayonix http://www.ayonix.com/aboutus.php	• Ofrece algoritmos y software para reconocimiento de rostros para compañías de seguridad, integradores de sistema y fabricantes de hardware y software • Incluye SDK
Betaface http://www.betaface.com/ info@betaface.com	• Sistema para detección de caras, reconocimiento de rostros, medidas biométricas, entre otros. Incluye reconocimiento de texto y de habla • Será liberada una Open Web API • Incluye SDK binario sólo con licencia para proyectos de tiempo limitado como campañas promocionales o productos basados en servidores

Nombre y URL de la Herramienta	Características
	• Desarrollado para Microsoft con Wunderman (http://www.wunderman.com)
BioID http://www.bioid.com/	• Tecnología biométrica que usa reconocimiento de rostros, voces y movimiento de labios para identificar una persona • Propósito principal es aplicaciones de seguridad o vigilancia • SDK para reconocimiento de cara, voz y movimientos
Cross Match Tecnology http://www.crossmatch.com/	• Ofrece un conjunto de aplicaciones para la detección y el reconocimiento de rostros basados en métodos neuronales • Incluye registro, verificación contra una base de datos, identificación y toma de fotografías • SDK para reconocimiento de rostros no disponible aún
Dr. Hu http://www.drhu.org/index.php	• Website de reconocimiento de caras • http://digface.t35.com/ • En este website se encuentra la explicación de algunos métodos algoritmos y programas de código abierto que pueden ser aplicados en la búsqueda, detección y reconocimiento de caras.
Ex-Sight http://www.ex-sight.com/index1.htm	• Sistema de reconocimiento de rostros para aplicaciones biométricas que usan cámaras analógicas o digitales.
Fraunhofer IIS http://www.iis.fraunhofer.de/EN/	• Detección de rostros, ojos, nariz, labios. • Estima estado de ánimo de persona a partir de características anteriores.
idfend Security Shield http://www.tcc.us.com/products.htm	• Verificación de usuario para entrada y salida al sistema Windows • Diseñado para ambiente de múltiple dominios y permite la autenticación usando reconocimiento facial en conjunto con el *login* de Windows
Luxand http://www.luxand.com/products/	• Ofrece productos para el reconocimiento de caras, animación, reconocimiento de música, procesamiento y análisis de imágenes y sonido • Incluye SDK para reconocimiento facial
Sensible Vision http://www.sensiblevision.com/company/about.htm	• Reconocimiento de caras para seguridad • Aplicación biométrica para reconocimiento de caras que permite el acceso a ambientes computacionales eliminando el uso de *login* y contraseñas
Smarti http://www.tab-systems.com/index.php	• Reconocimiento de Caras integrado en hardware para puertas y teléfonos • Usa dos tipos de equipos DIADEM y ALCOR
Visiphor http://www.visiphor.com/solutions/integration_technologies.html	• Reconocimiento Facial • Ofrece un modulo de búsqueda de fotografías para ser usado en ambientes policiales y en el sector judicial.

Tabla No 1.3

Software para Reconocimiento Facial que Funciona sobre Sistema Operativo Linux

Nombre y URL de la Herramienta	Características
Fdlib http://www.kyb.mpg.de/bs/people/kienzle/fdlib/fdlib.htm	• Librería de detección de caras programada sobre lenguaje C y MatLab
Torch 3 Vision http://torch3vision.idiap.ch/introduction.php	• Librería de visión programada en C++ sobre la librería de aprendizaje Torch
Colorado State University http://www.cs.colostate.edu/evalfacerec/index.html	• Conjunto de programas para evaluar algoritmos de reconocimiento de rostros
VXL http://vxl.sourceforge.net/index.html#intro	• Conjunto de librerías en C++ diseñadas para la investigación en visión artificial.
XVision http://www.cs.jhu.edu/CIPS/xvision/	• Conjunto de herramientas independientes usadas para hacer seguimientos a través de visión: 1) control de manipuladores; 2) seguimiento de ojos y boca en el rostro de una persona para animar la cara virtual de un payaso; 3) interacción con objetos virtuales haciendo uso de los gestos y las manos.

Tabla No 1.4

Software para Reconocimiento Facial que Funciona sobre Sistema Operativo Windows y Linux

Nombre y URL de la Herramienta	Características
Cognitec http://www.cognitec-systems.de/	• Identificación de rostros en fotografías y videos • Incluye SDK para registrar rostros y para detectar y reconocer caras en imágenes y videos • Liberada versión 7.1, para Linux, Agosto 2008

Nombre y URL de la Herramienta	Características
Cybula http://www.cybula.com/	• Reconocimiento de caras en 3 dimensiones. • Incluye un SDK denominada AURA que permite trabajar con texto, imágenes de rostros y otros tipos de datos • Versión para Linux y Windows
HMM http://www-robotics.cs.umass.edu/Documentation/FaceRecognition#HMM	• Conjunto de programas desarrollados sobre OpenCV para el reconocimiento de rostros haciendo uso de Modelos Ocultos de Markov Embebidos • Versión para Linux y Windows
Oki http://www.oki.com/jp/FSC/ics/en/index.html	• Motor de reconocimiento de caras. Ofrece un conjunto de librerías que funcionan como middleware, denominado FSE Solution Libraries. (Librería de Funciones para reconocimiento de rostros) • Liberada versión que trabaja sobre cualquier sistema operativo en Agosto 2008
Omni Perception http://www.omniperception.com/products/affinity/sdk	• Software de soluciones biométricas que permite la identificación de personas en verificaciones uno a uno, o en verificaciones de uno a muchos • Incluye SDK para reconocimiento de caras • Versión para Linux y Windows
VeriLook: Neurotechnology http://www.neurotechnology.com/index.html	• Sistema para el reconocimiento facial. • Ofrece un SDK para el reconocimiento de caras • Versión para Linux y Windows

- Eficiencia de los algoritmos y métodos empleados para la detección de caras, extracción de características o modelado del rostro y para el reconocimiento de la persona a partir del rostro modelado.

- Compatibilidad de la licencia del SDK con la arquitectura de software del robot y grado de accesibilidad al código fuente del conjunto de herramientas.

Evaluadas las características anteriores y una vez realizadas las pruebas de operación y funcionamiento correspondientes, se tomó la decisión de desarrollar las habilidades para el reconocimiento de rostros haciendo uso del conjunto de herramientas de software desarrolladas por Ou, [14], para el reconocimiento de rostros, el cual emplea la biblioteca de visión OpenCV y aplica Modelos Ocultos de Markov Embebidos en el modelado y reconocimiento de los rostros, de acuerdo a lo propuesto por Nefian, [24], [25].

- La detección de rostros será lograda a través del método de localización de caras propuesto por Viola y Jones, [37], mejorado por Lienhart, Liang y Kuranov, [18], el cual se encuentra integrado en la biblioteca OpenCV. El clasificador denominado cascada de clasificadores aumentados trabaja con características tipo haar, y ha sido entrenado y utilizado en diferentes aplicaciones de visión para la detección de rostros en imágenes. Este método ha demostrado su capacidad para segmentar caras en imágenes en tiempo real de forma confiable, - 15 cuadros por segundo en imágenes de 320x240, ejecutado en procesadores personales -, [38].

- Como en este proyecto el sistema debe funcionar en tiempo real, dos métodos son propuestos para formar los vectores de observación de los rostros: la transformada de Karhunen Loève, - KLT -, y la transformada discreta del coseno en 2 dimensiones, - 2D DCT -. Como el uso del KLT para obtener los vectores de observación tiene la desventaja de que se requiere calcular las funciones bases de un gran número de imágenes se aplicará el cálculo del DCT en 2 dimensiones, de acuerdo a lo establecido por Nefian en [23].

La DCT es una de las transformadas más populares en el mundo de la compresión de imágenes y video. La DCT es utilizada en normas de codificación como

JPEG, H.261, H.263, MPEG-1, MPEG-2 y MPEG-4, entre otras, lo que prueba sus elevadas potencialidades.

Para el cálculo de los coeficientes 2D DCT se emplea las funciones integradas a la biblioteca OpenCV, de acuerdo a lo establecido en [14].

- Para el reconocimiento de rostros es utilizado el método propuesto por [23] basado en modelos ocultos de Markov embebidos. Este método se encuentra implementado en [14] y hace uso de la biblioteca OpenCV.

- OpenCV, [26], es la librería de visión artificial desarrollada por Intel y publicada bajo licencia de Distribución de Software Berkeley, - BSD -. Por lo tanto, puede ser usada libremente para propósitos comerciales y de investigación, con las condiciones en ella expresadas. Es una librería multiplataforma y está orientada al tratamiento de imágenes en tiempo real.

Capítulo 2

MODELOS OCULTOS DE MARKOV EMBEBIDOS

2.1 Modelo de Markov

Un modelo de Markov es un proceso probabilístico sobre un conjunto finito de N estados, $S = \{S_1, S_2, ..., S_N\}$. Cada transición de estado genera un carácter de un alfabeto del proceso. El objeto de estudio se relaciona con la probabilidad de que el próximo estado sea un estado dado, $P\{q_t = S_i\}$ y esto puede depender de la historia anterior, $t-1$.

En este tipo de problemas, el objetivo es reconocer patrones que aparecen sobre espacios finitos de tiempo. Estos patrones ocurren en muchas áreas: pautas usadas por personas para dar instrucciones a un computador, secuencias de palabras en oraciones, secuencia de fonemas en palabras habladas, cualquier área donde una secuencia de ocurrencia de eventos puede producir patrones útiles.

Un modelo de Markov es un proceso que se mueve de un estado a otro dependiendo solamente de los n estados previos. Se le denomina un modelo de *orden n*, donde n es el número de estados que afectan la selección del próximo estado. El proceso de Markov más simple es de primer orden, donde la selección del estado es hecha basándose únicamente en el estado previo. Esta selección es probabilística.

Un modelo de Markov de orden *1*, o de primer orden, tiene una memoria de tamaño 1 y está definido por una matriz de probabilidades $P(q_t = S_j \mid q_{t-1} = S_i)$ para $i=1, \ldots, N$, $j=1, \ldots, N$. Un modelo de Markov de orden cero no tiene memoria. Para todos los puntos t de una secuencia, $P(q_t = S_i) = P(q_{t+k} = S_i)$. Un modelo de Markov de orden cero es equivalente a una distribución probabilística multinomial.

En el diagrama de la Figura No 2.1, se representa un proceso con M estados y M^2 transiciones que pueden ocurrir entre los estados, puesto que es posible que cualquier estado suceda a otro. Asociada a cada transición de estado existe una

probabilidad, la probabilidad de moverse de un estado al otro. Estas probabilidades pueden reunirse en un matriz de transición de estados, $A = \{ a_{ij} \}$, $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$, $i=1, \ldots, N$, $j=1, \ldots, N$, $0 \le a_{ij} \le 1$, con la restricción $\sum_{j=1}^{N} a_{ij} = 1$, $i=1, \ldots, N$. Se considera que estas probabilidades no varían en el tiempo.

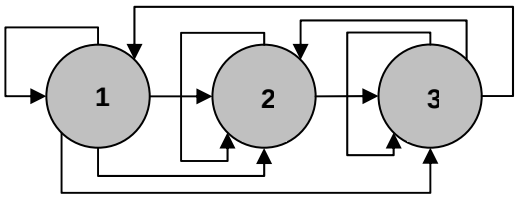

Figura No 2.1 Proceso de Markov con 3 Estados

Un ejemplo de la matriz, para el proceso presentado en la figura es la siguiente:

Estados en t-1	*Estados en t* 1	2	3
1	0,500	0,375	0,125
2	0,250	0,125	0,625
3	0,250	0,375	0375

Si en el tiempo t-1 el sistema estaba en el estado 1, la probabilidad de que en el tiempo t esté en el estado 2 es de 0,375, en el estado 1 es de 0,500 y en el estado 3 es de 0,125.

Para inicializar el sistema se debe conocer el vector de las probabilidades iniciales de cada estado, o la distribución de probabilidades del estado inicial, $\prod = \{ \pi_i \}$.

Estados en t_1	1	2	3
$\prod =$	0,500	0,250	0,250

$\pi_i = P(q_1 = S_i)$, $i=1, \ldots, N$

En resumen, un modelo de Markov de primer orden consiste de:

- De un conjunto de N estados, $S = \{S_1, S_2, ..., S_N\}$, con el estado al tiempo t denotado por $q_t \in S$.

- Matriz de transición de estados, $\boldsymbol{A} = \{ a_{ij} \}$, donde $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$, $i=1, \ldots, N,\ j=1, \ldots, N,\ 0 \le a_{ij} \le 1$, con la restricción $\sum_{j=1}^{N} a_{ij} = 1$, $i=1, \ldots, N,$ la cual indica la probabilidad de que el sistema se encuentre en un estado, dado el estado anterior.

- La distribución inicial de probabilidades $\prod = \{ \pi_i \}$, donde $\pi_i = P(q_1 = S_i)$, $i=1, \ldots, N$.

2.2 Modelo Oculto de Markov

En algunos casos los patrones que se desean encontrar, no describen suficientemente el proceso de Markov y se tienen dos conjuntos de estados: los observables y los ocultos. Por ejemplo, en un sistema de reconocimiento de habla, el sonido que se escucha es el producto de las cuerdas vocales, tamaño de la garganta, posición de la lengua y otras cosas más. Cada uno de estos factores interactúa para producir el sonido de una palabra y los sonidos detectados por el sistema de reconocimiento de habla son generados por los cambios físicos internos de la persona que está hablando. Algunos dispositivos de reconocimiento de habla trabajan considerando que la producción interna del habla es una secuencia de estados ocultos y que el sonido resultante es una secuencia de estados observables generada por el proceso de habla que aproxima mejor los verdaderos estados ocultos.

Un modelo oculto de Markov, HMM, es un modelo estadístico en el cual se asume que el sistema modelado es un proceso de Markov con parámetros desconocidos y el reto consiste en determinar los parámetros ocultos, a partir de los parámetros observables. Un modelo oculto de Markov puede ser considerado como una red Bayesiana dinámica muy simple. En el diagrama de la Figura No 2.2, se representa un proceso con estados observables y estados ocultos.

Los estados ocultos están modelados por un proceso de Markov de primer orden y están conectados unos a otros. Las conexiones entre los estados ocultos y los estados observables representan la probabilidad de generar un estado observable en particular dado que el proceso de Markov se encuentra en un estado oculto particular, $b_j(\boldsymbol{O}_t) = P(\boldsymbol{O}_t \mid q_t = S_j)$. La matriz de distribución de probabilidades para las

observaciones, $\boldsymbol{B}=\{b_j(\boldsymbol{O}_t)\}$, donde $b_j(\boldsymbol{O}_t)$ es la probabilidad de observar $\boldsymbol{O}_t$ en el tiempo t dado que el estado es $q_t = S_j$.

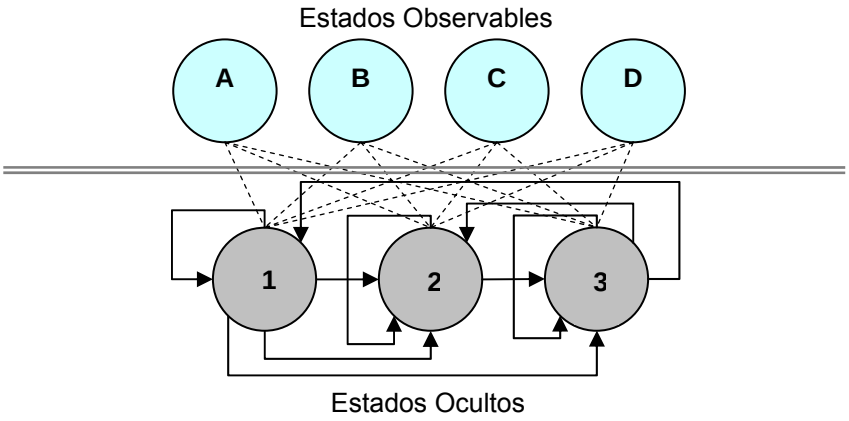

Figura No 2.2 Proceso de Markov con Estados Ocultos y Estados Observables

Estados Observables t

Estados Ocultos	*A*	*B*	*C*	*D*
1	*0,60*	*0,20*	*0,15*	*0,05*
2	*0,25*	*0,25*	*0,25*	*0,25*
3	*0,05*	*0,10*	*0,35*	*0,50*

La matriz de distribución de probabilidades indica que cuando el proceso se encuentra en el estado oculto 3, la probabilidad de observar el estado A es 0,05; la de observar el estado B es de 0,10; la de observar el estado C es de 0,35 y la de observar el estado D es de 0,50.

Un modelo oculto de Markov, HMM, es un modelo de Markov estándar aumentado por un conjunto de estados observables y una relación probabilística entre éstos y los estados ocultos.

En un modelo oculto de Markov discreto, [23], la probabilidad de observación de un símbolo está definida por $b_j(\boldsymbol{O}_t) = P(\boldsymbol{O}_t = v_k \mid q_t = S_j)$, $i = 1,\ldots,N$, donde $V = \{v_1, v_2, ..., v_M\}$ es el conjunto de todos los símbolos posibles de observación y M es el número de posibles símbolos de observación diferentes.

En un modelo oculto de Markov de densidad continua las observaciones están caracterizadas por funciones de probabilidad continuas. Una representación general de la función de densidad de probabilidad, [23], es una mixtura finita de la forma,

$b_i(\boldsymbol{O}_t) = \sum_{k=1}^{M_i} c_{ik} N(\boldsymbol{O}_t, \boldsymbol{\mu}_{ik}, \boldsymbol{U}_{ik})$, $i = 1,...,N$, donde c_{ik} es el coeficiente de mixtura para la *kth* mezcla en el estado i. Usualmente, $N(\boldsymbol{O}_t, \boldsymbol{\mu}_{ik}, \boldsymbol{U}_{ik})$, es una función de densidad Gaussiana, con el vector media $\boldsymbol{\mu}_{ik}$ y la matriz de covarianza $\boldsymbol{U}_{ik}$. Si el modelo es de mixtura atada significa que los componentes gaussianos son almacenados en un pool y todas las distribuciones de salida de los estados comparten este pool, de tal forma que la distribución de salida para el estado i es definida como $b_i(\boldsymbol{O}_t) = \sum_{k=1}^{M} c_{ik} N(\boldsymbol{O}_t, \boldsymbol{\mu}_k, \boldsymbol{U}_k)$, $i = 1,...,N$ y en este caso los parámetros de la componente Gaussiana y de la mixtura son independientes del estado.

Un modelo oculto de Markov es un tripla $\lambda = (\prod, \boldsymbol{A}, \boldsymbol{B})$, con los siguientes elementos, [23]:

- $\prod = \{\pi_i\}$, es la distribución inicial de probabilidades, donde $\pi_i = P(q_1 = S_i)$, *i=1, …, N.*

- $\boldsymbol{A} = \{a_{ij}\}$, es la matriz de transición de estados, donde $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$, *i=1, …, N, j=1, …, N*

- $\boldsymbol{B} = \{b_j(\boldsymbol{O}_t)\}$, es la matriz de distribución de probabilidades para las observaciones, donde $b_j(\boldsymbol{O}_t) = P(\boldsymbol{O}_t \mid q_t = S_j)$, $b_j(\boldsymbol{O}_t)$ es la probabilidad de observar $\boldsymbol{O}_t$ en el tiempo t dado que el estado es $q_t = S_j$.

- Cada probabilidad en la matriz de transición de estados y en la matriz de observaciones es independiente del tiempo, esto es no cambian sus valores con la dinámica del sistema.

2.2.1 Usos Asociados con los Modelos Ocultos de Markov

Una vez que un sistema ha sido descrito como un HMM, tres problemas diferentes pueden ser resueltos, [5], [22]: a) Estimación: generar el HMM dada la secuencia de observaciones; b) Decodificación: encontrar la secuencia de estados ocultos que más

probablemente generó la secuencia observada; c) Evaluación: encontrar la probabilidad de una secuencia observada dado un HMM.

2.2.1.1 Estimación

Dada la secuencia de observaciones $\boldsymbol{O} = \boldsymbol{O}_1, \boldsymbol{O}_2, ..., \boldsymbol{O}_t$, encontrar los parámetros del modelo HMMM $\lambda = (\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B})$ que maximicen $P(\boldsymbol{O} \mid \lambda)$, En este caso se usa el algoritmo iterativo de *Baum Welsh,* el cual resuelve este problema mediante un proceso de maximización de la esperanza.

2.2.1.2 Decodificación

Encontrar la secuencia de estados ocultos más probable dada una secuencia de observaciones. Esto es encontrar los estados ocultos que generaron la salida observada. En muchos casos, el interés está en los estados ocultos del modelo porque representan algo de valor que no es directamente observable. Dado el modelo $\lambda = (\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B})$ y la secuencia de observaciones $\boldsymbol{O} = \boldsymbol{O}_1, \boldsymbol{O}_2, ..., \boldsymbol{O}_t$ encontrar la secuencia de estados $q_1, q_2, ..., q_t$ que maximicen $P(\boldsymbol{O}, \boldsymbol{q} \mid \lambda)$. En este caso se usa el algoritmo de *Viterbi* para determinar la secuencia de estados ocultos más probables dada una secuencia de observaciones y un HMM.

2.2.1.3 Evaluación

Dado el modelo $\lambda = (\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B})$ y la secuencia de observaciones $\boldsymbol{O} = \boldsymbol{O}_1, \boldsymbol{O}_2, ..., \boldsymbol{O}_t$ calcular la probabilidad de la secuencia de observaciones $\boldsymbol{O}$ dado λ, $P(\boldsymbol{O} \mid \lambda)$. En este caso se usa el algoritmo *hacia delante (forward)* complementado con el algoritmo *hacia atrás (backward)* cuando las matrices $\boldsymbol{A}$ y $\boldsymbol{B}$ no pueden se medidas directamente.

2.3 Modelos Ocultos de Markov para Modelar Rostros

El uso de los HMM en el modelado de las caras se debe a la estructura del rostro y por la invarianza parcial al escalamiento. Para imágenes frontales, las regiones significantes, -cabello, frente, ojos, nariz, labios -, aparecen en un orden natural, desde el tope hasta la

base aún si las imágenes son tomadas bajo pequeñas rotaciones del plano de la imagen o en rotaciones del plano perpendicular al plano de la imagen.

En [23], cada una de estas regiones faciales, de izquierda a derecha, es asignada a un estado en un HMM, unidimensional continuo con matriz de covarianza diagonal. Antes de que un modelo oculto de Markov pueda ser usado para el reconocimiento de la cara, se debe definir la estructura del HMM, la cual incluye el número de estados, el conjunto de transiciones permitidas y las observaciones que son producidas por cada estado. El modelo usado en este sistema es un HMM de 6 estados de izquierda-a-derecha, donde los estados son usados para representar cabello, frente, ojos, nariz, labios y quijada, - una versión de 5 estados de este modelo es mostrado en la Figura No 2.3.

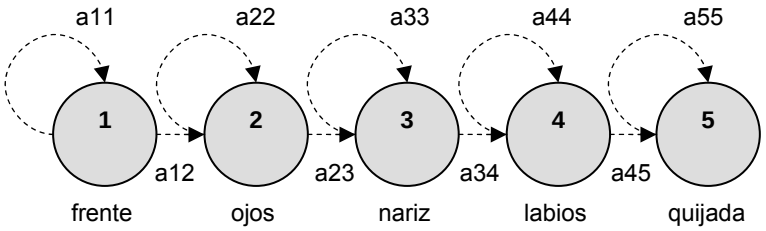

Figura No 2.3 HMM con 5 Estados para Modelar Caras

2.3.1 Vectores de Observación

En [23] se describe este método para la extracción de los vectores de observaciones: Dada una imagen de W píxeles de ancho y de H píxeles de alto que contienen una cara, bloques de las imágenes de L filas pueden ser extraídos y usados para formar las observaciones, Figura No 2.4. Los bloques adyacentes pueden ser superpuestos por P filas. Este solapamiento permite que las características sean capturadas de una forma que es independiente de la posición vertical, mientras que el particionamiento disjunto de la imagen podría resultar en el truncamiento de características que ocurren a través de las fronteras de los bloques. Si se usa un valor pequeño para L se puede traer insuficiente información discriminante al vector de observaciones, mientras que si se usan valores grandes de L se incrementa la probabilidad de corte a través de ciertas características. Sin embargo, la rata de reconocimiento del sistema no es muy sensible a las variaciones en L, mientras el valor de P sea grande ($P \leq L-1$) y $L \approx H/10$. en lugar de usar

intensidades de los píxeles en cada bloque de $L \times W$, se forman vectores de observación con los coeficientes de la transformada de Karhunen Loève (KLT) o transformada discreta del coseno (2D DCT) para cada bloque de la imagen. En [33] los vectores de observación consisten de todos los valores de píxeles de cada de bloque y por lo tanto la dimensión del vector de observación es $L \times W$, $(L=10, W=92)$. El uso de los valores de los píxeles como vectores de observación tiene dos desventajas importantes: primero los valores de los píxeles no representa una característica robusta, puesto que tienden a ser muy sensibles al ruido de la imagen, como también, a la rotación de la imagen o cambios en iluminación. Segundo, las grandes dimensiones del vector de observaciones conducen a una complejidad elevada en el cómputo del sistema en el entrenamiento, detección y reconocimiento. Esto puede ser crítico para el sistema de reconocimiento de caras que opera con grandes bases de datos o cuando el sistema es usado para aplicaciones en tiempo real.

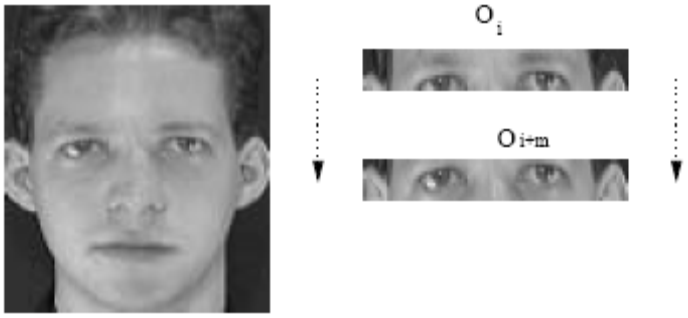

Figura No 2.4 Parametrización de Imágenes y Extracción de Bloques

Tomada de Nefian, A. (1999). A Hidden Markov Model Based Approach for Face Detection and Recognition. Thesis of Doctor of Philosophy in Electrical Engineering. Georgia Institute of Technology. Atlanta, USA, página 42.

Las propiedades de comprensión óptima del KLT, así como sus propiedades de correlación, la hacen una transformada atractiva par ser usada en los vectores de observación. Las funciones bases de KLT, las cuales son los vectores propios de la matriz de covarianza de todas las muestras de entrada, son obtenidos de la forma siguiente:

- Los bloques de la imagen extraídos desde todas las imágenes en el conjunto de entrenamiento son organizados como vectores por concatenamiento de las columnas. Sea $\boldsymbol{x}_i^{(r)}$ el vector obtenido del bloque *i*-ésimo de la *r*-ésima imagen.

- La media muestral de los vectores es calculada de acuerdo a: $\mu = \frac{1}{\sum_{r=1}^{R} T^{(r)}} \sum_{r=1}^{R} \sum_{i=1}^{T^{(r)}} \boldsymbol{x}_i^{(r)}$, donde R es el número de imágenes en el conjunto de entrenamiento y $T^{(r)}$ es el número de bloques extraídos desde la imagen r-ésima. Un nuevo conjunto de vectores es obtenido sustrayendo la media de todos los $\boldsymbol{x}_i^{(r)}$: $\bar{\boldsymbol{x}}_i^{(r)} = \boldsymbol{x}_i^{(r)} - \mu$. Como los vectores $\boldsymbol{x}_i^{(r)}$ están caracterizados por medias estadísticas de valor cero, pueden ser usados para determinar el conjunto de funciones básicas para el KLT.

- La matriz de covarianza de los vectores $\boldsymbol{x}_i^{(r)}$ es calculada de acuerdo: $\boldsymbol{C} = \frac{1}{\sum_{r=1}^{R} T^{(r)}} \sum_{r=1}^{R} \sum_{i=1}^{T^{(r)}} \bar{\boldsymbol{x}}_i^{(r)} (\bar{\boldsymbol{x}}_i^{(r)})^T$ y los vectores propios normalizados $\boldsymbol{u}_k$ de la matriz de covarianza son determinados por $\boldsymbol{C}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k$ donde λ_k es el valor propio de los vectores propios de la matriz de covarianza $\boldsymbol{C}$.

El uso del KLT para obtener los vectores de observación tiene la desventaja de que requiere del cálculo de las funciones bases de un gran número de imágenes. Una alternativa es calcular un DCT de 2D, (2D DCT), para formar los vectores de observaciones, partiendo de las razones siguientes: 1) muchos de los coeficientes DCT tienden a ser pequeños y aquellos que son grandes generalmente se concentran alrededor de bajas frecuencias. 2) Los coeficientes DCT, a diferencia de la intensidad de los píxeles, tienden a ser menos sensitivos al ruido, rotaciones de la imagen o cambios en la iluminación.

2.3.2 Entrenamiento del Modelo Oculto de Markov del Rostro

Para la detección de la cara las imágenes en el conjunto de entrenamiento representan caras frontales de diferentes personas tomadas bajo diferentes condiciones de iluminación. Todas las imágenes del conjunto de entrenamiento son usadas para entrenar un HMM. Para el reconocimiento de caras, cada individuo en la base de datos está representado por un modelo HMM. Un conjunto de imágenes representando

diferentes instancias de la misma persona son usadas para entrenar cada HMM, después de extraer los bloques de cada imagen en el conjunto de entrenamiento y obtener los vectores de observaciones (coeficientes DCT o KLT).

Dado un conjunto de caras que forman un conjunto de entrenamiento HMM, el procedimiento usado para entrenar el modelo oculto de Markov, mostrado en la Figura No 2.5, es el siguiente, [23]:

- Los datos de entrenamiento son segmentados de manera uniforme, desde la parte superior hasta la parte inferior de la imagen, de acuerdo al número de estados del HMM y los vectores de observaciones asociados con cada estado son generados y usados para obtener los estimados iniciales de la matriz de probabilidades de las observaciones $\boldsymbol{B}$.

- Una buena estimación inicial de los parámetros es esencial para la convergencia rápida al máximo global de la función de similitud. Los valores para $\boldsymbol{A}$ son colocados de tal manera que son consistentes con la estructura arriba - abajo del modelo, esto es $a_{ij} = 0$ para $j < i$ y $j > i + 1$, lo cual significa que solo se pueden hacer transiciones desde el estado i hacia sí mismo o al próximo estado. Para la distribución probabilística del estado inicial se coloca $\pi_1 = 1$ y $\pi_i = 0$ para $i \neq 1$, es decir que el HMM comienza en el primer estado.

- En la próxima iteración la segmentación uniforme es reemplazada por una segmentación de Viterbi. Para cada uno de los N estados, el resultado de cada segmentación de la secuencia de entrenamiento es un estimado de máxima similitud del conjunto de observaciones, que ocurre con cada estado de acuerdo al modelo actual. Las iteraciones continúan hasta que la similitud de la segmentación de Viterbi, en una iteración consecutiva, se hace menor al valor de algún umbral previamente definido. A este punto los parámetros del HMM están inicializados.

- Siguiendo el modelo de inicialización, los parámetros del modelo son re-estimados usando el procedimiento de reestimación de *Baum Welsh*. Este procedimiento ajusta los parámetros del modelo de tal forma que maximizan la probabilidad de observación de los datos de entrenamiento, dado cada modelo.

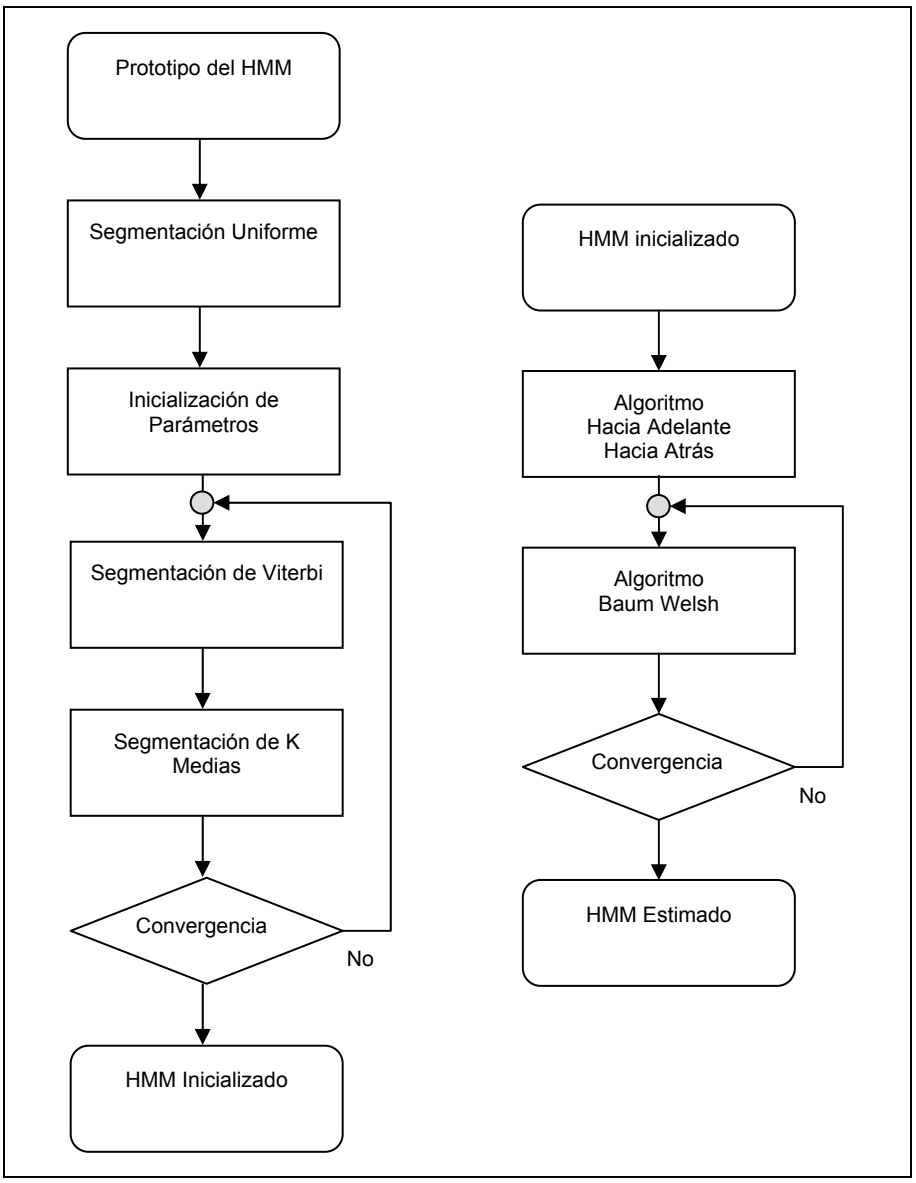

Figura No 2.5 Entrenamiento del HMM

- El modelo resultante es comparado entonces con el modelo previo, -calculando la distancia que refleja el resultado de la similitud estadística del HMMs -. Si la distancia del modelo excede a un umbral, entonces el modelo λ es reemplazado por el nuevo modelo $\tilde{\lambda}$ y el lazo de entrenamiento es repetido. Si la distancia del modelo está por debajo del umbral, entonces se asume la convergencia del modelo y los parámetros finales son guardados.

2.3.3 Reconocimiento de Rostros Usando Modelos Ocultos de Markov

Para usar el HMM en el reconocimiento de una cara en particular, partiendo de una imagen del rostro, se emplea el procedimiento mostrado en la Figura No 2.6, [23]. Primero, es generada una secuencia de observaciones mediante el escaneo de la imagen, de arriba hacia abajo, formando bloques de L filas y extrayendo los vectores de observación de cada bloque. Luego la probabilidad de la secuencia de observación, dado el modelo HMM para cada rostro, es encontrada usando el algoritmo de *Viterbi*. El modelo con la máxima similitud es seleccionado de tal forma que el modelo revela la identidad de la cara desconocida.

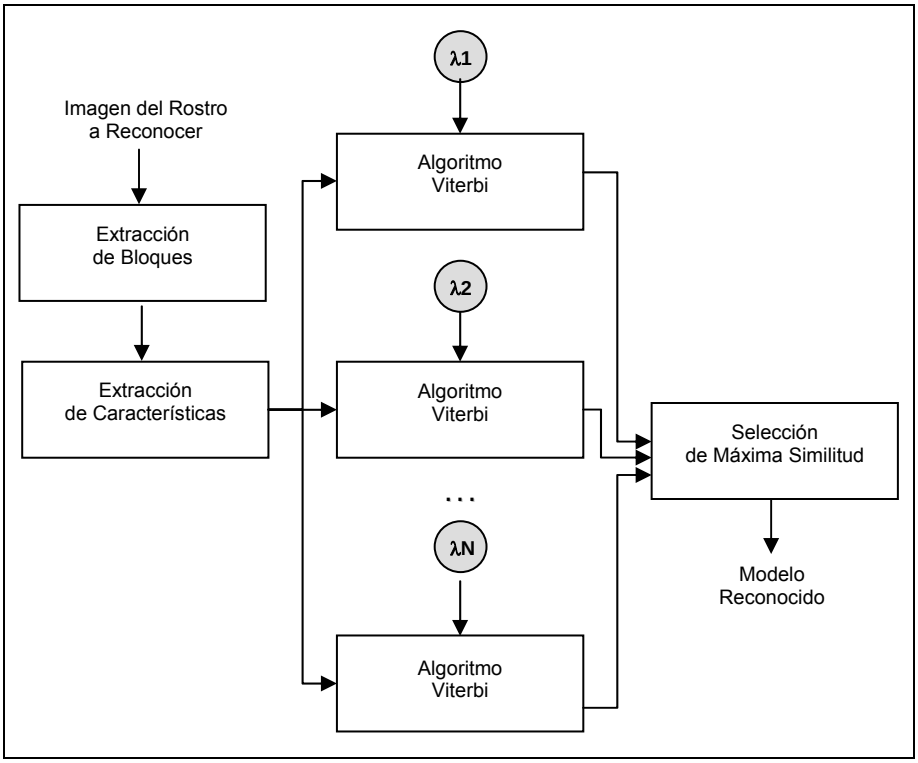

Figura No 2.6 Reconocimiento del Rostro Usando HMM

2.4 Modelos Ocultos de Markov Embebidos

El modelo estándar del HMM, de una dimensión, tiene limitaciones en el modelado de datos bidimensionales, como los de las imágenes de los rostros. Para caracterizar este tipo de datos, en [24] se introduce el uso de modelos ocultos de Markov embebidos en el modelado de este tipo de datos.

Un modelo oculto de Markov embebido, EHMM, es una generalización de modelo oculto de Markov, donde cada estado del HMM unidimensional, es por sí mismo un HMM. Un modelo oculto de Markov embebido consiste de un conjunto de superestados con un conjunto de estados embebidos. Los superestados modelan datos bidimensionales a lo largo de una dirección, mientras que los modelos ocultos de Markov embebidos modelan los datos a lo largo de la otra dirección. Sin embargo, un HMM doblemente embebido no es un HMM bidimensional verdadero, puesto que las transiciones entre los estados de diferentes superestados no son permitidas.

Un EHMM consiste de los siguientes elementos, [23]:

- De un conjunto de N_0 estados, $S_0 = \{S_{01}, S_{02}, ..., S_{0N0}\}$.

- La distribución de probabilidades para un superestado inicial, $\prod_0 = \{ \pi_{0,i} \}$, donde $\pi_{0,i}$ es la probabilidad de estar en el superestado i en el tiempo cero.

- La matriz de transición entre los superestados, $A_0 = \{ a_{0,ij} \}$, donde $a_{0,ij}$, es la probabilidad de hacer una transición desde el superestado i hasta el superestado j.

- En un HMM embebido, cada superestado es por sí mismo un HMM y su estructura es la misma que la de un HMM unidimensional. Sin embargo, a diferencia del HMM de una dimensión, el número de estados, las probabilidades del estado inicial y la matriz de transición de estado dependerán del superestado en cual se encuentra el HMM. Para cada HMM embebido se tiene lo siguiente:

a. El número de estados embebidos, en el superestado k-ésimo, N_1^k, y el conjunto de estados embebidos, $S_1^k = \{S_{1,i}^k, i = 1,2,...N_1^k\}$,.

b. La distribución de probabilidades del estado inicial para los estado embebidos, $\prod_1^k = \{ \pi_{1,i}^k \}$, donde $\pi_{1,i}^k$, es la probabilidad de estar en el estado i del superestado k en el tiempo cero.

c. La matriz de transición de estados para los estados embebidos, $\boldsymbol{A}_1^k = \{ a_{1,ij}^k \}$, donde $a_{1,ij}^k$ especifica la probabilidad de hacer una transición del estado *i* al estado *j*, en el superestado *k*.

d. La matriz de la distribución de probabilidades para las observaciones, $\boldsymbol{B}^k = \{ b_j^k(\boldsymbol{O}_{t_0,t_1}) \}$, donde $b_j^k(\boldsymbol{O}_{t_0,t_1})$ es la probabilidad de la observación $\boldsymbol{O}_{t_0,t_1}$, dado el estado embebido *j* en el superestado *k*.

e. Para un HMM embebido de tipo discreto se asume que $\boldsymbol{O}_{t_0,t_1}$ puede tomar un número finito de símbolos de observación. Sea P el número de símbolos de observación diferentes y sea V el conjunto de todos los símbolos de observación posibles, $V = \{ v_1, v_2, ..., v_p \}$. En este caso $\boldsymbol{B}$ es la matriz de probabilidad de observación del símbolo, por ejemplo $\boldsymbol{B}^k = \{ b_j^k(p) \}$, donde $b_i^k(p) = P(\boldsymbol{O}_{t_0,t_1} = v_p \mid S_{0,k}, S_{1,i}^k, \lambda)$.

f. Para un HMM embebido con densidad continua, las observaciones están caracterizadas por una función de probabilidad de densidad continua, la cual es tomada de la mezcla de Gaussianas finitas de la forma $b_i^k(\boldsymbol{O}_{t_0,t_1}) = \sum_{m=1}^{M_i^k} c_{im}^k N(\boldsymbol{O}_{t_0,t_1}, \boldsymbol{\mu}_{im}^k, \boldsymbol{U}_{im}^k)$, para $1 \le i \le N_1^k$, donde c_{im}^k es el coeficiente de mezcla para la *m*-ésina mixtura en el estado *i* del superestado *k* y $N(\boldsymbol{O}_{t_0,t_1}, \boldsymbol{\mu}_{im}^k, \boldsymbol{U}_{im}^k)$ es la función de Gaussiana con vector media $\boldsymbol{\mu}_{im}^k$ y matriz de covarianza $\boldsymbol{U}_{im}^k$.

g. Un HMM embebido de mezcla atada es uno en el cual los componentes Gaussianos son almacenados en un pool y todos los estados de las distribuciones de salida comparte este pool, de forma tal que la distribución de salida del estado *i* está definido como $b_i^k(\boldsymbol{O}_{t_0,t_1}) = \sum_{m=1}^{M} c_{im}^k N(\boldsymbol{O}_{t_0,t_1}, \boldsymbol{\mu}_m^k, \boldsymbol{U}_m^k)$.

Esta ecuación difiere de la anterior en que los parámetros de la componente Gaussiana y el número de componentes de la mixtura son independientes de los estados.

Sea $\Lambda^k = \{ \Pi_1^k, A_1^k, B^k \}$ el conjunto de parámetros que definen el k-ésimo superestado, entonces el HMM embebido está definido por la tripla $\lambda = (\Pi_0 , A_0 , \Lambda)$, donde $\Lambda = \{ \Lambda^1, \Lambda^2, ... \Lambda^{N_0} \}$.

2.4.1 Usos Asociados con los Modelos Ocultos de Markov Embebidos

Al igual que en el caso de los HMM, una vez que un sistema ha sido descrito como un EHMM, tres problemas pueden ser resueltos: a) Evaluación: encontrar la probabilidad de una secuencia observada dado un HMM. Para el cálculo de la probabilidad $P(O|\lambda)$ de la secuencia de observaciones O dado el modelo λ se aplica el algoritmo *hacia delante* y el algoritmo *hacia atrás* para HMM embebidos; b) Decodificación: dada una secuencia de observaciones O, la meta del algoritmo de decodificación es recuperar el conjunto más probable q que maximiza $P(O,q|\lambda)$. Para encontrar la secuencia de estados ocultos que más probablemente generó la secuencia observada, se aplica el algoritmo de *Viterbi*; c) Estimación: para estimar los parámetros del HMM embebido se usa la iteración para ajustar los parámetros del modelo con respecto a cierto criterio de optimización. En este caso se usa un algoritmo similar al de *Baum Welsh*, derivado del HMM unidimensional.

2.5 Modelos Ocultos de Markov Embebidos para Modelar Rostros

En esta sección se presenta un HMM embebido para el modelado de rostros, [23]. Un ejemplo de este tipo de modelos es mostrado en la Figura No 2.7, donde cada uno de los cinco estados presentados en el HMM unidimensional de la Figura No 2.3, se convierte en un superestado que contiene un HMM embebido. Los superestados modelan la imagen a lo largo de la dirección vertical, mientras que los HMM embebidos modelan los datos a lo largo de a dirección horizontal. En este modelo se tienen $N_0 = 5$ superestados que son usados para representar las 5 regiones faciales principales cuando una imagen es recorrida de arriba hacia abajo. El primer y último superestado tiene tres estados embebidos, mientras que los restantes tienen cinco. El número de estados embebidos fue seleccionado de acuerdo al número de regiones que se espera encontrar en una cara, en la medida que se produce el escaneo de izquierda a derecha en un superestado. Por ejemplo en el segundo superestado, ojos, se tiene un estado para cada una de las regiones laterales entre los ojos y el cabello, un estado para cada uno de los ojos y dos

estados para la región entre los ojos. Cada estado fue modelado por una mezcla de tres Gaussianas y la matriz de covarianza se seleccionó diagonal.

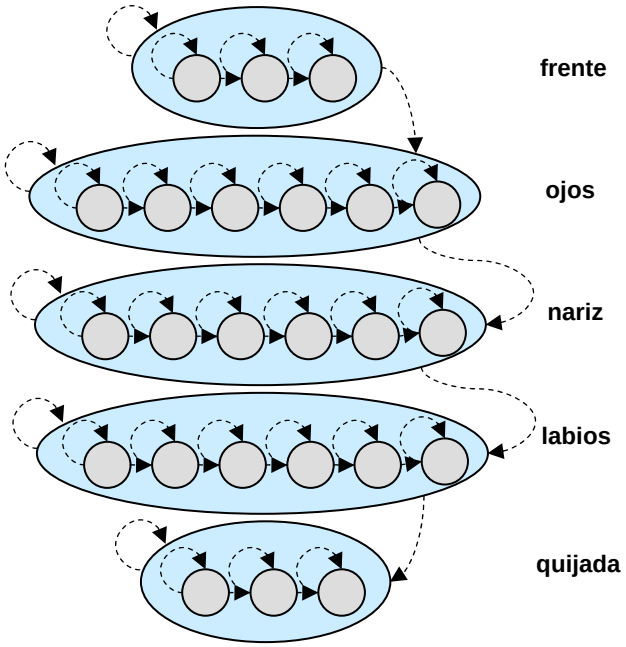

Figura No 2.7 HMM Embebido para Detección y Reconocimiento de Rostros

2.5.1 Vectores de Observación

La secuencia de observación para las imágenes está formada por bloques de tamaño $L_x \, x \, L_y$ que son extraídos al recorrer la imagen de izquierda a derecha y de arriba abajo. Los bloques adyacentes se solapan por P_y filas en la dirección vertical y por P_x columnas en la dirección horizontal. Los valores seleccionados fueron los siguientes: $L_x = 8$, $L_y = 10$, $P_x = 6$, $P_y = 8$. Los vectores de observaciones estuvieron formados por 6 coeficientes bidimensionales DCT o por 4 coeficientes KLT, correspondientes a los valores propios más grandes. Esta selección reduce la dimensión de los vectores de observación por un factor de cerca de 13 a 20. Además la reducción de la dimensionalidad de los vectores de observación usando los coeficientes DCT también reduce la sensibilidad del HMM al ruido, a las rotaciones de las imágenes y a los cambios de iluminación. Por otra parte los cambios de iluminación, lo cual constituye uno de los principales problemas en sistemas de reconocimiento de caras, pueden ser considerados

como insignificantes cuando los bloques son pequeños en relación al tamaño de la imagen, como los usados en este enfoque de 8x10 píxeles. Por lo tanto las variaciones de iluminación afectarán primordialmente a los componentes DC del 2D DCT y dejando al resto de los coeficientes invariables. Si las intensidades de los píxeles son usadas como vectores de observación, las variaciones en iluminación afecta todos los elementos de estos vectores y consecuentemente reduce la robustez del sistema. Finalmente, el uso los coeficientes DCT para las observaciones permite la detección y reconocimiento de las caras en imágenes que están en formatos comprimidos, como por ejemplo JPEG.

2.5.2 Entrenamiento del Modelo Oculto de Markov Embebido

El entrenamiento del EHMM usa el criterio de máxima verosimilitud. El entrenamiento es similar al empleado en el entrenamiento de un HMM estándar, excepto que en este caso la segmentación de *Viterbi*, el procedimiento *hacia adelante* y *hacia atrás* y el algoritmo de re-estimación de *Baum Welsh* son reemplazados por los correspondientes algoritmos de decodificación, evaluación y estimación. Para entrenar un HMM embebido el procedimiento, mostrado en la Figura No 2.8, es el siguiente, [23]:

- Primero los vectores de observaciones son extraídos de cada individuo en el conjunto de entrenamiento. Sean T_1 observaciones de izquierda a derecha y T_0 observaciones de arriba hacia abajo. Existe entonces un arreglo de $T_0 T_1$ observaciones. Los valores de T_0 y T_1 son determinados según el tamaño de la imagen, el tamaño de los bloques de las imágenes (L_x y L_y) y de la cantidad de solapamiento (P_x y P_y).Estas observaciones son usadas para el entrenamiento.

- En la primera iteración, los vectores de observaciones son segmentados de manera uniforme en N_0 superestados verticales y los vectores con cada superestado k son uniformemente segmentados de izquierda a derecha en N_1^k estados, como se muestra en la Figura No 2.9.

- En la próxima iteración, la segmentación uniforme es reemplazada por una *segmentación de Viterbi* doblemente embebida.

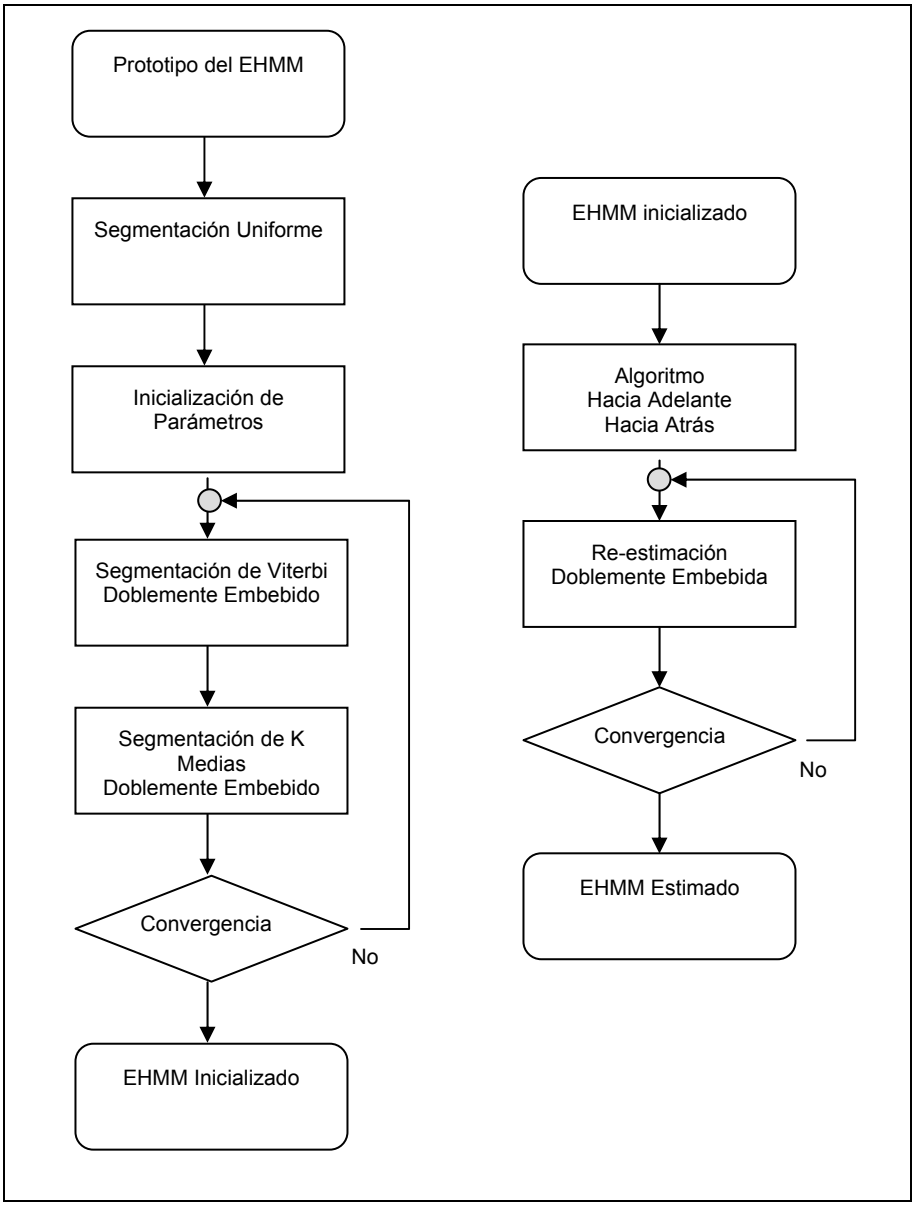

Figura No 2.8 Entrenamiento del HMM Embebido

- Los parámetros del modelo son estimados usando una extensión del algoritmo de *segmentación k-medias*, en dos dimensiones. Específicamente, las probabilidades de transición estimadas entre los superestados $\tilde{a}_{0,ij}$ y las probabilidades de transición estimadas entre los estados embebidos $\tilde{a}^{i}_{1,il}$ son obtenidas de la forma siguiente:

$$\tilde{a}_{1,jl}^{i} = \frac{número\ de\ transiciones\ desde\ S_{1,j}^{i}\ hasta\ S_{1,l}^{i}}{número\ total\ de\ transiciones\ desde\ S_{1,j}^{i}}$$

$$\tilde{a}_{0,ij} = \frac{número\ de\ transiciones\ desde\ S_{0,i}\ hasta\ S_{0,j}}{número\ total\ de\ transiciones\ desde\ S_{0,i}}$$

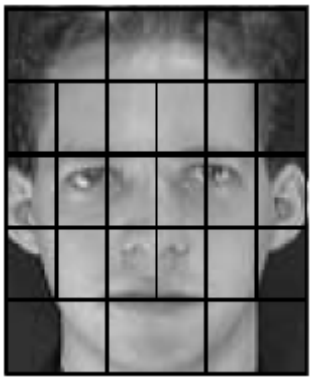

Figura 2.9 Segmentación de la Imagen del Rostro para HMM Embebido

Tomada de Nefian, A. (1999). A Hidden Markov Model Based Approach for Face Detection and Recognition. Thesis of Doctor of Philosophy in Electrical Engineering. Georgia Institute of Technology. Atlanta, USA, página 93

- Adicionalmente, la media estimada $\tilde{\mu}_{jm}^{i}$, la covarianza $\tilde{\boldsymbol{U}}_{jm}^{i}$ de la mixtura Gaussiana y los coeficientes de mixtura $\tilde{c}_{jm}^{i}$ para lo mixtura m del estado j en el superestado i es obtenida de acuerdo:

$$\tilde{\mu}_{jm}^{i} = \frac{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1=1}^{T_1}\psi_{jm}^{i,r}(t_0,t_1)\boldsymbol{O}(t_0,t_1)}{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1=1}^{T_1}\psi_{jm}^{i,r}(t_0,t_1)}$$

$$\tilde{\boldsymbol{U}}_{jm}^{i} \frac{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1}^{T_1}\psi_{jm}^{i,r}(t_0,t_1)(\boldsymbol{O}(t_0,t_1)-\tilde{\mu}_{jm}^{i})(\boldsymbol{O}(t_0,t_1)-\tilde{\mu}_{jm}^{i})^{T}}{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1}^{T_1}\psi_{jm}^{i,r}(t_0,t_1)}$$

$$\tilde{c}_{jm}^{i} = \frac{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1=1}^{T_1}\psi_{jm}^{i,r}(t_0,t_1)}{\sum_{r=1}^{R}\sum_{t_0=1}^{T_0}\sum_{t_1=1}^{T_1}\sum_{m=1}^{M}\psi_{jm}^{i,r}(t_0,t_1)}$$

Donde $\psi_{jm}^{i,r}(t_0,t_1)$ es igual a uno si $\boldsymbol{O}(t_0,t_1)$ está asociado con el componente de mixtura m del estado j en el superestado i y es cero en cualquier otro caso.

- La iteración se detiene y el HMM embebido es inicializado cuando la puntuación alcanzada por el *Viterbi* doblemente embebido en una iteración consecutiva es menor al valor de un umbral especificado.

- Los parámetros de los HMM embebidos son re-estimados usando el procedimiento *hacia adelante hacia atrás* y el procedimiento de re-estimación para maximizar la probabilidad de las observaciones, dado el modelo de la cara,

- La iteración se detiene y los parámetros del HMM embebido están entrenados, cuando la probabilidad de las observaciones dado el modelo, es menor al valor de un umbral especificado, en iteraciones consecutivas

2.5.3 Reconocimiento de Rostros Usando Modelos Ocultos de Markov Embebidos

Una vez que el HMM embebido ha sido entrenado con un conjunto de imágenes, puede ser usado en el reconocimiento de caras. El procedimiento, mostrado en la Figura No 2.10, es el siguiente, [23]:

- Primero los vectores de observaciones son extraídos del rostro desconocido.

- La probabilidad de la secuencia de observación dado el HMM embebido del modelo del rostro de cada persona es calculado usando el algoritmo de *Viterbi doblemente embebido*.

- El modelo con la máxima verosimilitud es seleccionado. Este modelo seleccionado debe revelar la identidad del rostro desconocido.

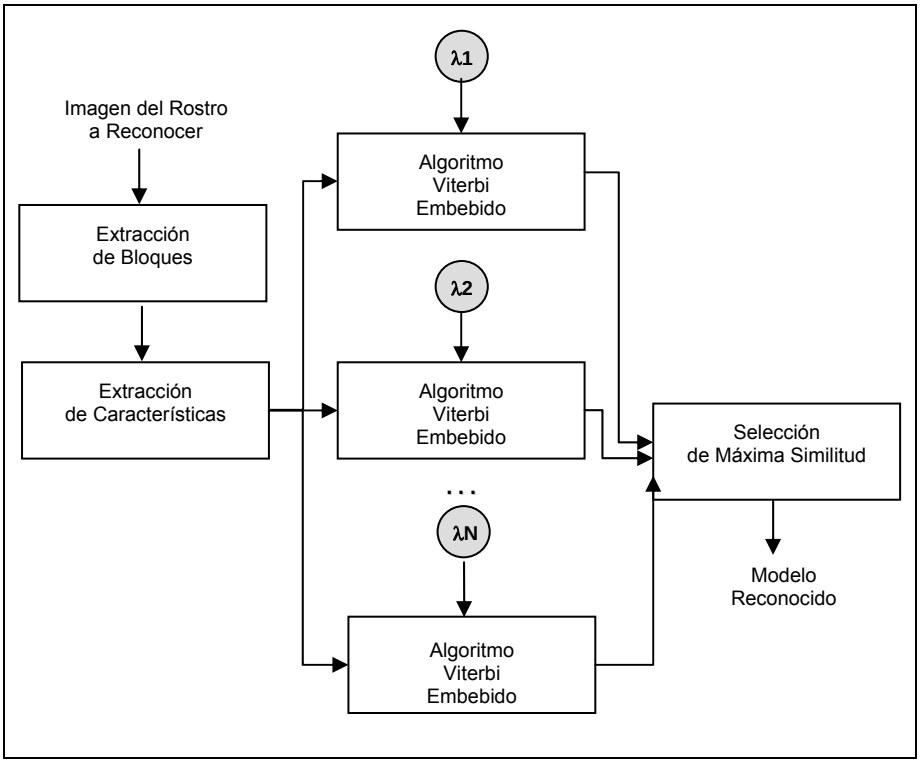

Figura No 2.10 Reconocimiento del Rostro Usando HMM Embebido

Capítulo 3

HABILIDADES PARA RECONOCIMIENTO DE ROSTROS

3.1 Arquitectura de Hardware del Robot

La plataforma experimental es el robot autónomo personal Maggie, [32], desarrollado, a nivel de hardware y de software, en el Robotics Lab de la Universidad Carlos III de Madrid, Figura No 3.1 . Su misión principal es servir de soporte a la investigación sobre la interacción entre humanos y robots, la inteligencia robótica y la autonomía en los robots.

Figura No 3.1 Robot Personal Maggie

Maggie tiene una altura de 1,4m. En la parte inferior el robot posee una base móvil, en la cual están instalados 12 sensores de contacto, 12 sensores infrarrojos, 12 sonares, 1 láser Sick LMS 200 y 2 ruedas diferenciales que le permiten desplazarse por su entorno. Para las tareas de interacción Maggie tiene sensores de tacto distribuidos a lo largo del cuerpo, lo que le dan la sensación de estar cubierta por una “piel sensible”, [8]. Otras articulaciones, además de la base móvil, del robot están situadas en el cuello, brazos y párpados. Está equipado con dos antenas de radio frecuencia, una ubicada a la altura de la nariz, en la cabeza, y la otra en la base y que le permiten identificar objetos por radio frecuencia, [9].

Posee una cámara, Logitech QuickCam Pro 5000, situada en la boca la cual es usada para visión artificial. También tiene un sintetizador de voz que le permite hablar a través de los altavoces situados en el cuello. Presenta una serie de diodos LED en la boca que se iluminan cuando habla. Finalmente, un emisor infrarrojos situado tras la esfera negra ubicada en la parte frontal inferior del cuerpo le permite intercambiar información con otros dispositivos, como por ejemplo un aparato de TV o un equipo de música.

Maggie cuenta con dos procesadores: vismaggie, un Acer Travel Mate 3043 WTMI encargado del control de todo del hardware y un Tablet PC Acer Travel Mate 370, situado en la parte frontal media del cuerpo y usado como interfaz con el usuario. Estos procesadores están interconectados a través de un HUB Ethernet Belkin y con el mundo exterior a través de una red WiFi 802.11.

3.2 Arquitectura de Control para el Robot

La arquitectura de control del robot fue modelada por Barber y Salichs, [2], considerando la forma como los seres humanos organizan los procesos mentales. En el hombre se diferencian dos niveles de actividad mental: En el nivel deliberativo se realizan actividades de forma consciente, procesos que requieren de la capacidad de razonamiento o decisión. En el nivel automático se controla la ejecución de movimientos instintivos, procesos mentales que se ejecutan sin necesidad de tener conciencia de los mismos. Partiendo de estas ideas, en esta arquitectura, denominada AD, se establecen sólo dos niveles de control: Deliberativo y Automático, Figura No 3.2.

En el nivel deliberativo se encuentran procesos que requieren tiempo para la ejecución como consecuencia de un razonamiento: planificador de trayectorias, modelado del entorno, supervisor de tareas. En el automático están los módulos que interactúan con los sensores y actuadores y que requieren un tiempo mínimo para procesar la información: percepción sensorial y módulos de actuación.

Ambos niveles presentan una característica común: están formados por habilidades. Cada habilidad consiste en la capacidad de realizar un razonamiento o llevar a cabo una acción. Las habilidades son activadas por órdenes de ejecución de otras habilidades o de un secuenciador. Devuelven datos y eventos a las habilidades o secuenciadores que las han activado. Esas habilidades son la base de la arquitectura AD.

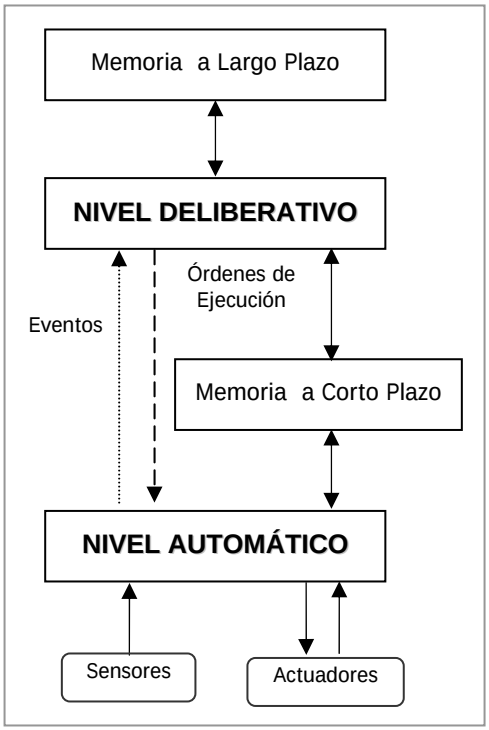

Figura No 3.2 Arquitectura AD

En el nivel deliberativo, [2], Figura No 3.3, se encuentran los módulos que requieren razonamiento o una capacidad de decisión. Estos módulos no producen respuestas inmediatas, y necesitan procesar la información con la que trabajan para tomar decisiones. Estos módulos son los que forman las habilidades deliberativas y son activados por un secuenciador, que será el encargado del correcto funcionamiento de estas habilidades. A diferencia del nivel automático, las actividades del nivel deliberativo

se llevan a cabo secuencialmente y no es posible realizar más de una actividad deliberativa al mismo tiempo.

En la memoria a largo plazo se encuentra toda la información de tipo permanente o casi permanente, sobre la que se va a razonar o tomar decisiones. Esta información puede ser definida a priori, o puede ser producto de un aprendizaje, o el resultado de procesar la información interna.

El secuenciador principal se encarga de gestionar las habilidades deliberativas dando la orden de ejecución de cada una de ellas en el momento oportuno. Otros secuenciadores, incorporados a otras habilidades deliberativas se encargan de gestionar las habilidades del nivel automático.

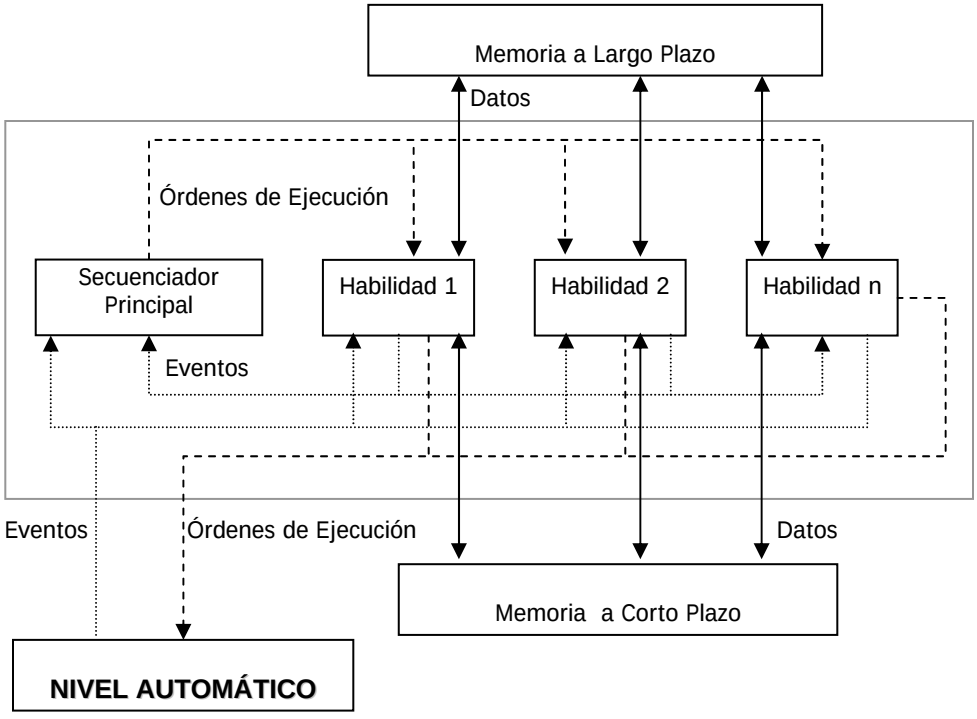

Figura No 3.3 Nivel Deliberativo de la Arquitectura AD
Tomada de Barber, R. (2000). *Desarrollo de una Arquitectura para Robots Móviles Autónomos, Aplicación a un Sistema de Navegación Topológica.* Tesis Doctoral. Universidad Carlos III de Madrid. Leganés, España, página: 55

La comunicación entre los niveles deliberativo y automático es bidireccional y el intercambio de datos se efectúa a través de la memoria a corto plazo. En esta memoria se almacena y mantiene información acerca del estado del robot y de dos flujos de información. 1) Flujo de eventos: las habilidades situadas en el nivel automático notifican

a las habilidades del nivel deliberativo los eventos que generan. 2) Flujo de órdenes de ejecución: las habilidades automáticas reciben órdenes de ejecución procedentes del nivel deliberativo. Las habilidades deliberativas deciden qué habilidades del nivel automático tienen que ser activadas o desactivadas en cada momento.

El nivel Automático, [2], Figura No 3.4, se encarga del control de los dispositivos del robot: en él se encuentran las habilidades que interactúan directamente con los sensores y actuadores. Este nivel permite al robot tener la capacidad de reacción necesaria para responder rápidamente a cambios producidos en su entorno: es aquí donde aparece la reactividad del robot. Todos los elementos definidos en este estrato deben ser capaces de ejecutarse en paralelo y de interactuar unos con otros.

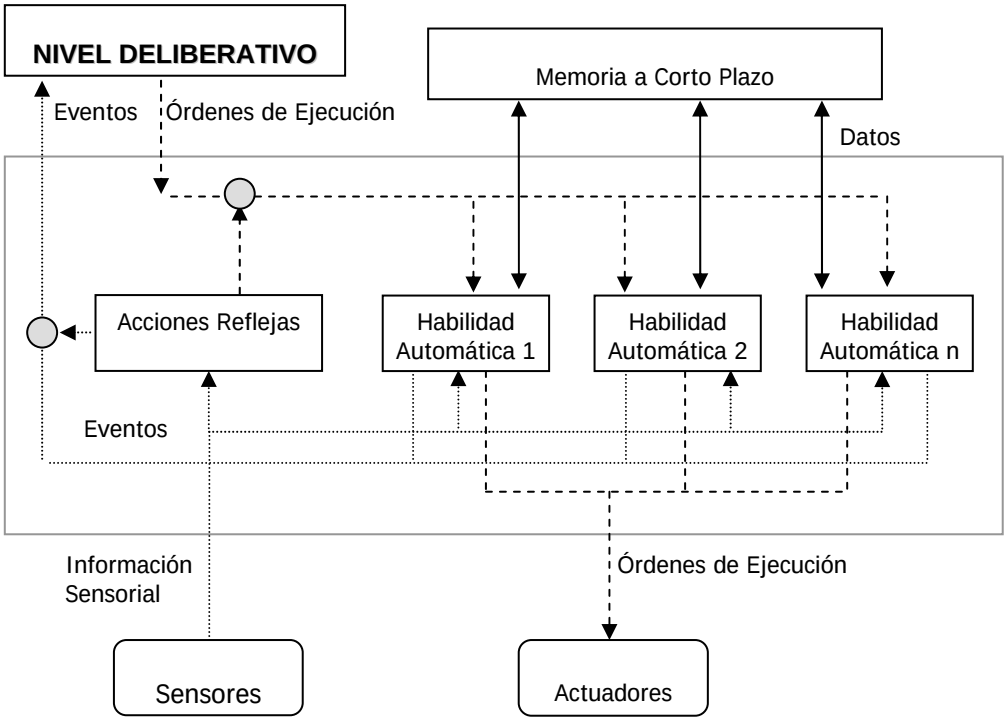

Figura No 3.4 Nivel Automático de la Arquitectura AD
Tomada de Barber, R. (2000). *Desarrollo de una Arquitectura para Robots Móviles Autónomos, Aplicación a un Sistema de Navegación Topológica.* Tesis Doctoral. Universidad Carlos III de Madrid. Leganés, España, página: 107

Entre los diferentes elementos que componen este nivel se encuentran las acciones reflejas las cuales proceden directamente de los sensores y contienen información prioritaria. Éstas son tratadas como interrupciones, deshabilitan las órdenes del nivel deliberativo y tienen atención preferente.

Las habilidades automáticas, el otro componente de este nivel, son las capacidades sensoriales y motoras del sistema. Estas habilidades pueden tomar información directamente de los sensores y ejecutar órdenes directamente sobre los actuadores o pueden ser actividades mediante los secuenciadotes de las habilidades deliberativas.

La arquitectura AD ha sido implementada, [29], en GNU/Linux, sobre un sistema basado en tres componentes principales, desacoplados y distribuidos: 1) Una jerarquía compuesta por una superclase abstracta y conjuntos de subclases que sirven para la definición e implementación de las habilidades. 2) Un manejador de eventos que permite definir y ejecutar los comportamientos requeridos por las habilidades. 3) Una memoria compartida y distribuida que almacena en tiempo de ejecución los datos e información producida y disponible para todos los elementos de la arquitectura. La administración de los eventos esta gobernado por un servidor principal y un conjunto de servidores secundarios, distribuidos sobre diferentes plataformas, diseñados todos bajo el paradigma editor - subscriptor.

3.3 Habilidades para el Reconocimiento de Rostros

Uno de los aspectos fundamentales en el diseño de cualquier sistema, de cierta complejidad, es su arquitectura, representada por un conjunto de elementos computacionales o componentes y una serie de conexiones o interacciones entre estos componentes. Las arquitecturas muestran la estructura del sistema, sus interrelaciones, los principios y reglas que gobiernan su diseño y ejecución a lo largo del tiempo. Los detalles estructurales incluyen la organización general y la estructura de control global, los protocolos de comunicación, sincronización y acceso a datos, la asignación de funcionalidad a los elementos de diseño, la distribución física, la composición de los elementos de diseño, su escalabilidad.

En el diseño de estas habilidades para el reconocimiento de rostros es utilizada una arquitectura de tres capas. En el nivel más externo, el nivel de las habilidades, se encuentra la interfaz con la arquitectura de control del robot. Es aquí donde se define e implementa el software que recibe y provee los datos dinámicos al robot. En la capa intermedia se encuentran los controladores, responsables de recibir y enviar los eventos

desde la interfaz y de comunicarlos al nivel más profundo de la arquitectura. Finalmente, en la última capa se definen elementos y componentes necesarios para la implementación del modelo oculto de Markov embebido para el modelado de los rostros y las rutinas para el reconocimiento de las caras, [23]. Los componentes de esta capa han sido extraídos del conjunto de herramientas de software desarrolladas por Ou, [14].

3.3.1 Nivel Habilidades

En este diseño las habilidades componen el nivel vista o la interfaz y se encargan de presentar el modelo y los datos en el formato adecuado para interactuar con el resto de la arquitectura de control AD. Es la capa más externa del sistema de reconocimiento de rostros y se encuentra definida en tres habilidades: 1) habilidad para tomar fotografías o imágenes de los rostros de las personas que pueden ser incorporadas, o no, en la Base de Datos; 2) habilidad para creación y actualización de la Base de Datos de los Rostros o Caras; 3) habilidad para reconocimiento de rostros.

Para la arquitectura AD, las habilidades son objetos capaces de encapsular comportamientos del robot, capacidades del robot para efectuar un razonamiento, llevar a cabo una acción o procesar información.

3.3.1.1 Habilidad para Tomar Imágenes de Rostros

Esta habilidad es la encargada de tomar las imágenes de los rostros de las personas. Estas imágenes pueden ser incorporadas posteriormente a la base de datos de rostros del robot. El Diagrama de Clases correspondiente a esta habilidad es mostrado en la Figura No 3.5, el Diagrama de Secuencia en las Figuras No 3.6 y 3.7 y el Diagrama de Componentes en la Figura No 3.8.

La habilidad tiene una interfaz que la comunica con la capa en la cual se encuentra definido e implementado el modelo oculto de Markov embebido y que se identifica en los diagramas con el nombre de *HMMFacesTomarImagenes*.

El funcionamiento de esta habilidad es el siguiente:

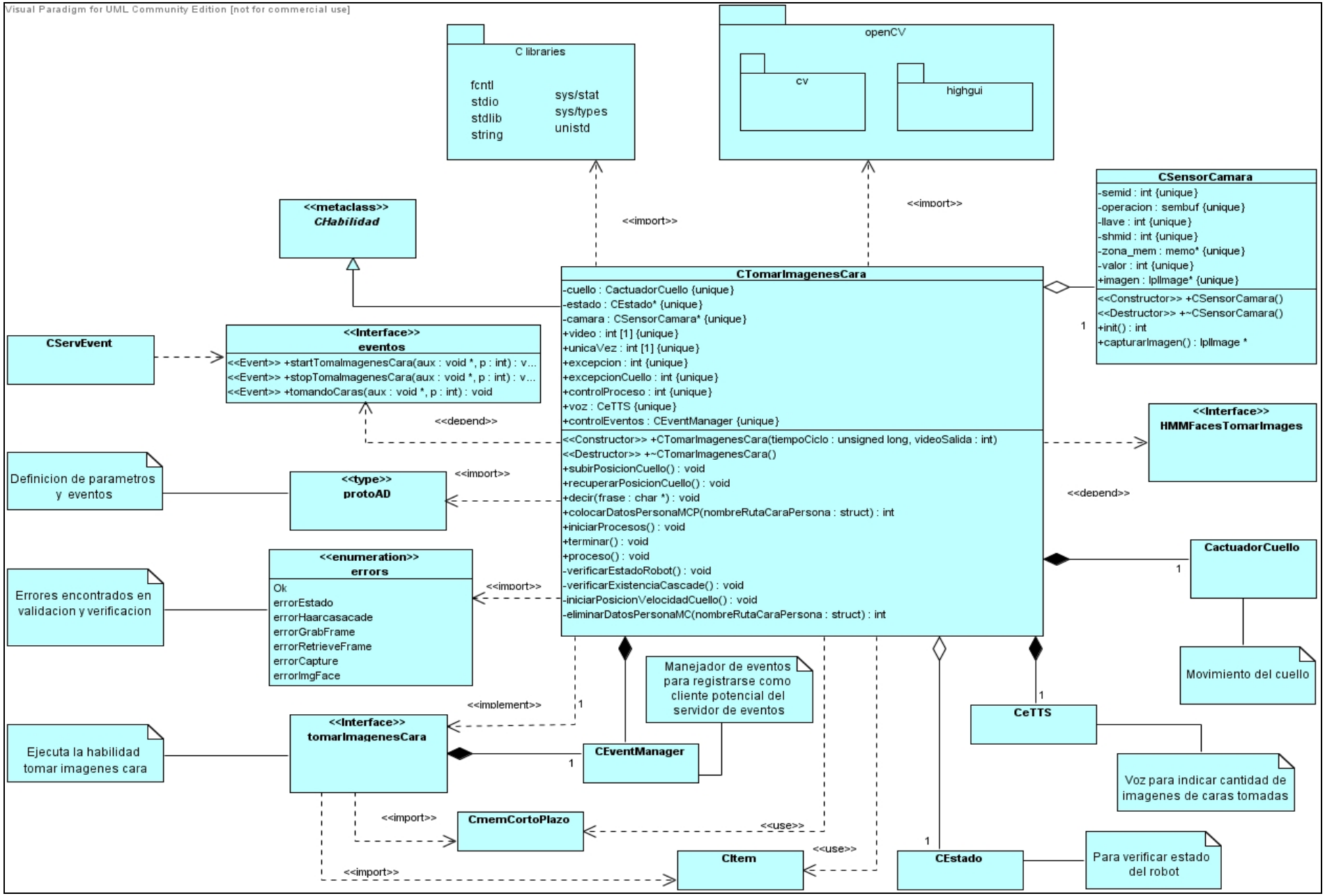

Figura No 3.5 Diagrama de Clases Habilidad Tomar Imágenes de Cara

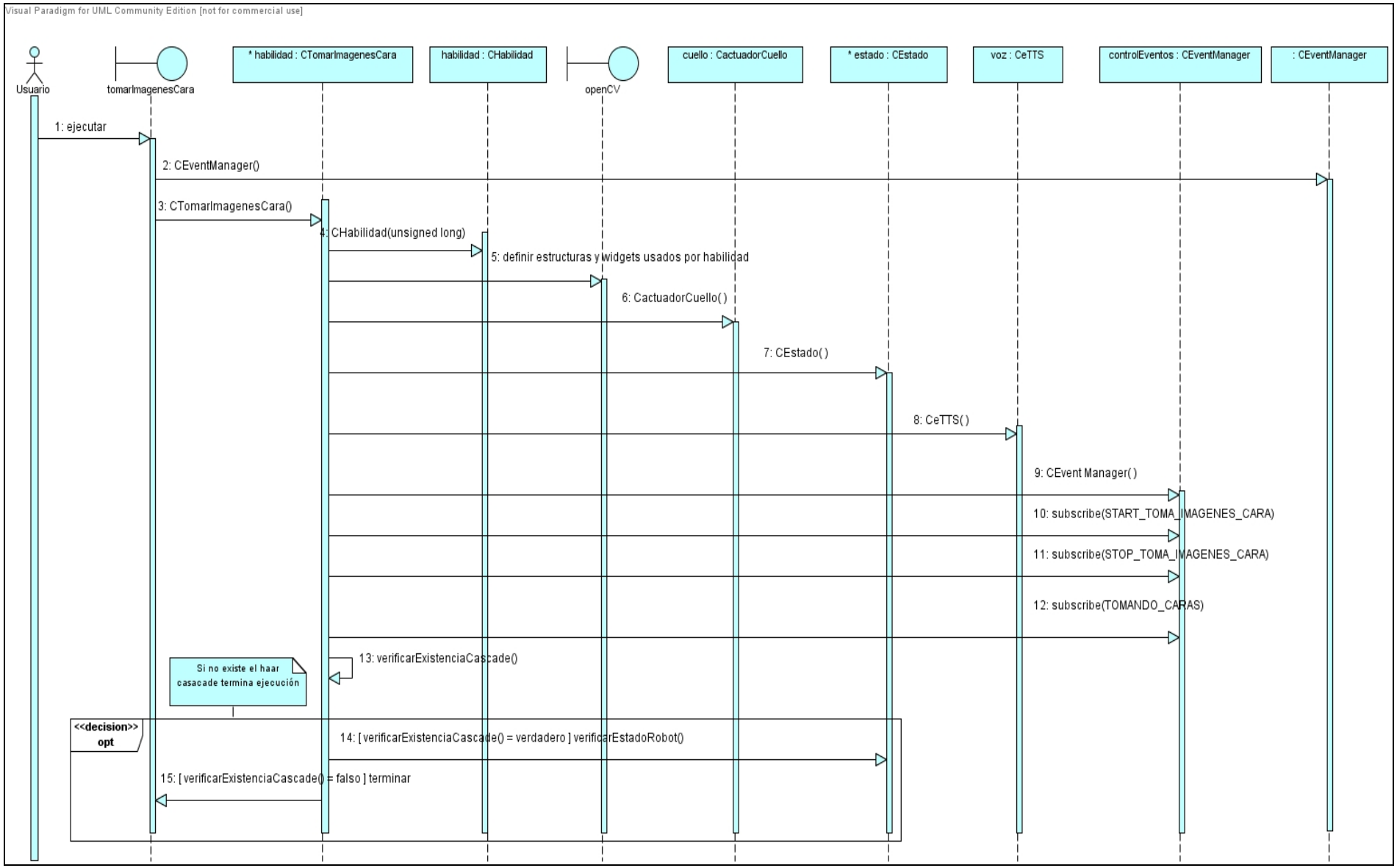

Figura No 3.6 Diagrama de Secuencia Habilidad Tomar Imágenes de Cara: Ejecución del Constructor de la Habilidad

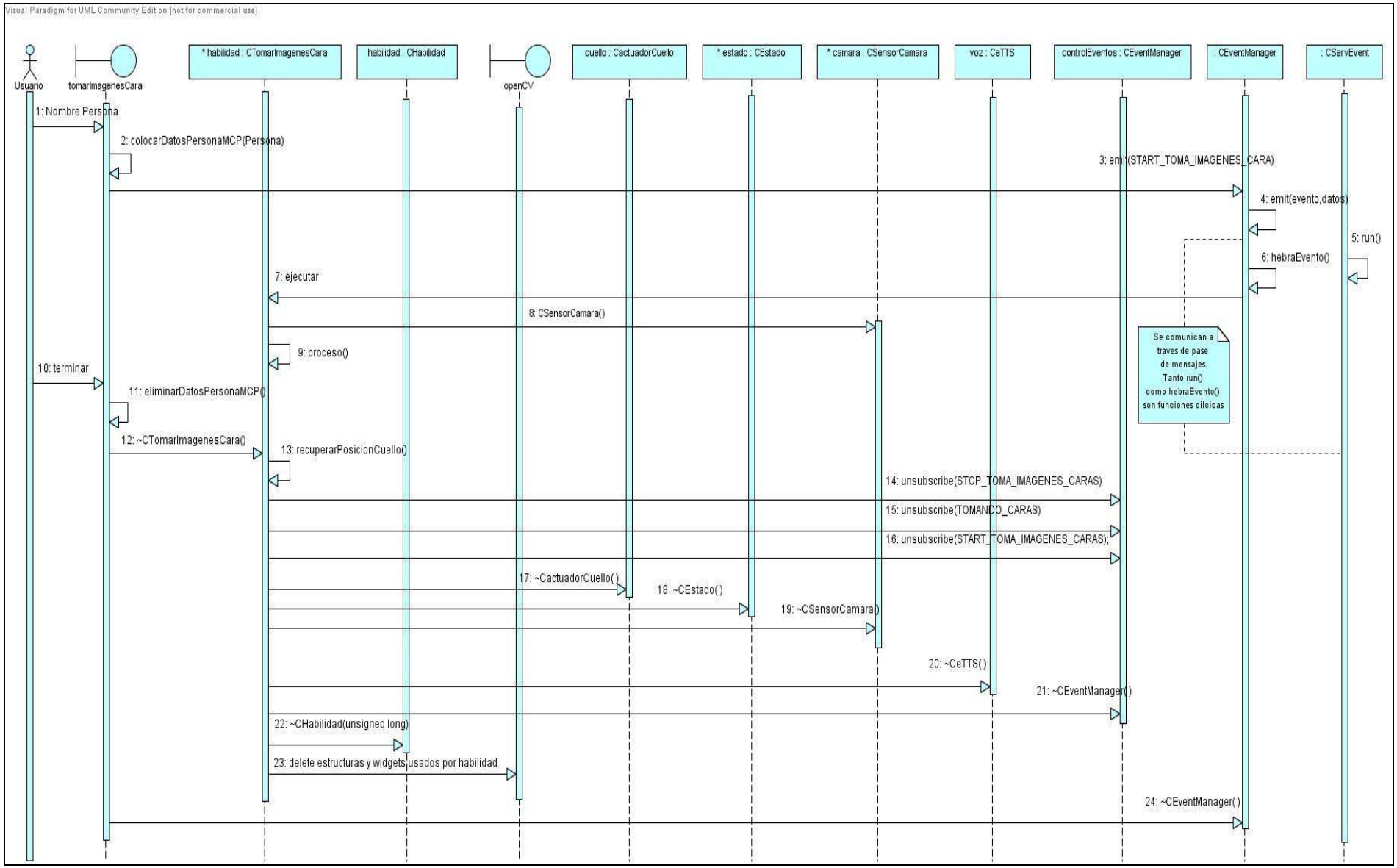

Figura No 3.7 Diagrama de Secuencia Habilidad Tomar Imágenes de Cara: Ejecución del Proceso y Destructor de la Habilidad

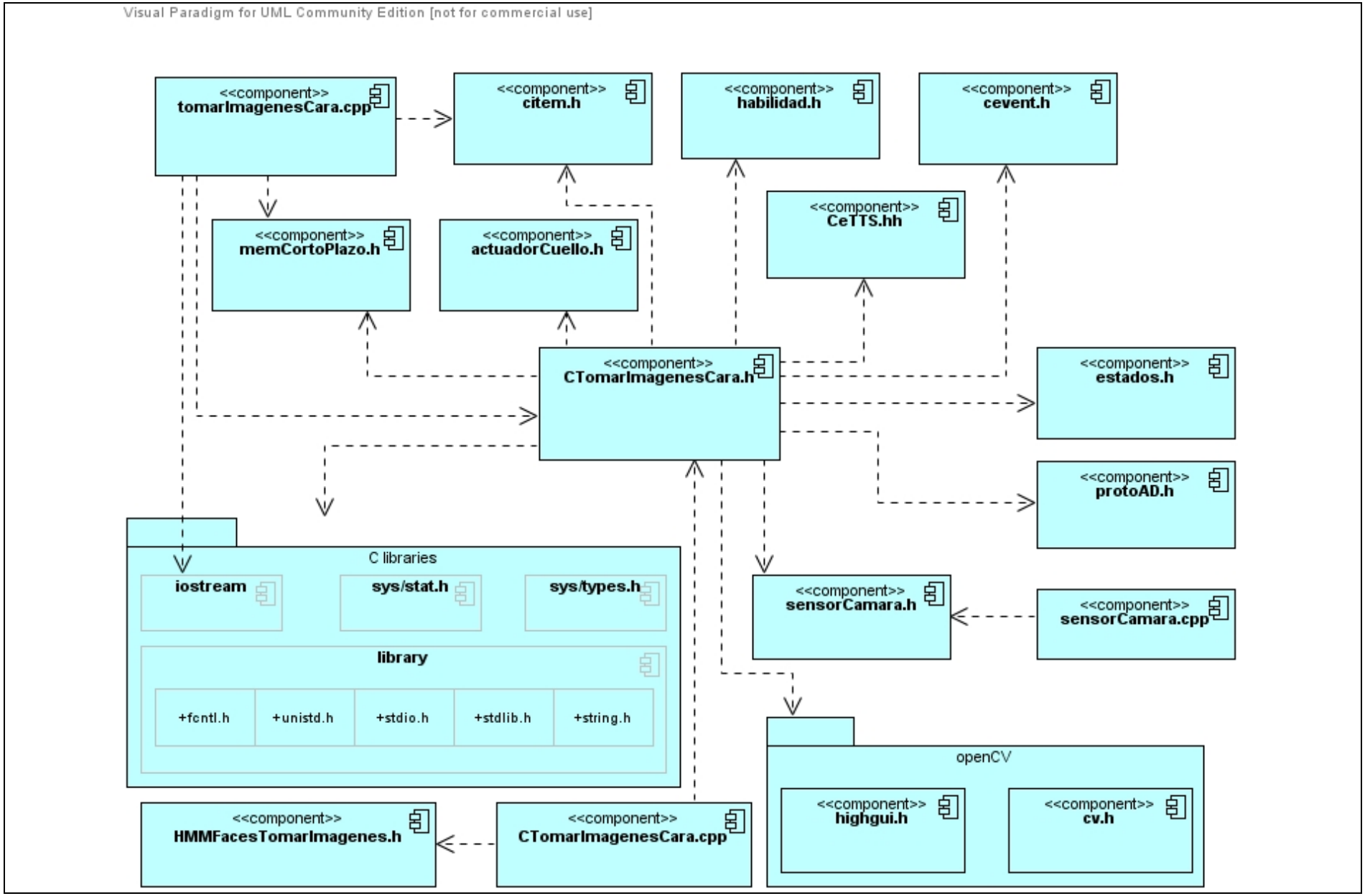

Figura No 3.8 Diagrama de Componentes Habilidad Tomar Imágenes de Cara

- Se ejecuta el constructor de la clase para generar la instancia de la habilidad correspondiente.

- Se suscriben ante el servidor de eventos y el administrador, los eventos y las funciones asociadas con la habilidad: *1) START_TOMA_IMAGENES_CARAS, startToma ImagenesCaras; 2) STOP_TOMA_IMAGENES_CARAS, stopTomaImagenesCara, 3) TOMANDO _CARAS, tomandoCaras.*.

- Se verifica la existencia del haarCascade, - *haarcascade_frontalface_alt.xml* -, archivo indispensable para que los módulos de las librerías del openCV puedan detectar caras. Si no se produce una excepción, se procede con la ejecución de la habilidad.

- Se verifica el estado del robot y si éste se encuentra en estado de emergencia o en algún estado no identificado se suspende la ejecución de la habilidad.

- Se efectúa el proceso cíclico de la habilidad toma de imágenes. Si es la primera vez, se ejecuta el inicio de los procesos:

  a. Se crea el sensor para capturar imágenes desde el servidor de la cámara.

  b. Se apertura y se lee de memoria a corto plazo, o de un archivo denominado *config.ini*, la ruta y el directorio en el cual se almacenarán las imágenes a ser tomadas y el nombre de la persona a quien corresponden las imágenes.

  c. Se verifica el estado del cuello del robot y se coloca la cabeza con la inclinación necesaria para capturar las imágenes.

  d. Se le otorga al controlador de la cámara unos segundos para que ajuste en forma automática los parámetros de la misma.

- En la ejecución del proceso cíclico de la habilidad, se captura la imagen del rostro y se envía al siguiente nivel, transfiriendo el control al módulo *HMMFacesTomarImagenes*, para su verificación y almacenamiento. Luego, la habilidad recibe nuevamente el control y notifica sobre la información procesada a través de la emisión de mensajes por consola y por medio de la voz del robot.

- Terminada la secuencia de toma de imágenes, la habilidad informa sobre la cantidad de fotos tomadas, colocando un mensaje en la memoria a corto plazo

- Se ejecuta el destructor de la habilidad.

- Al finalizar, debe existir un directorio con la ruta y nombre definidos y con tantas imágenes del rostro de la persona, como fotografías se hayan tomado. La primera imagen se almacena en colores y en formato *jpg*. Las demás imágenes se guardan en tonos de grises y en formato *pgm*.

3.3.1.2 Habilidad para Crear las Bases de Datos con los Rostros de las Personas

Esta habilidad es la encargada de crear y mantener actualizada la base de datos con los rostros de las personas que serán identificadas por el robot. El Diagrama de Clases correspondiente a esta habilidad es mostrado en la Figura No 3.9, el Diagrama de Secuencia en la Figura No 3.10 y el Diagrama de Componentes en la Figura No 3.11.

La habilidad tiene una interfaz que la comunica con la capa en la cual se encuentra definido e implementado el modelo oculto de Markov embebido y que se identifica en los diagramas con el nombre de *HMMFacesCrearDBF*.

Esta habilidad es muy importante porque maneja memoria a largo plazo y su ejecución debe garantizar la integridad, la independencia y la consistencia de los datos almacenados. A tal efecto deben tomarse en cuenta las siguientes consideraciones:

- Los archivos iniciales necesarios para la creación de la base de datos son las imágenes tomadas de los rostros de las personas: una imagen en colores y formato *jpg* y varias imágenes en tonos de grises y en formato *pgm*. Estos deben estar almacenados en un directorio denominado *FacesImgDB/Images* y ordenados en subdirectorios, cuyos nombres debe corresponder con el de la persona a quien corresponden las fotografías, Figura No 3.12.

- Debe existir un directorio denominado *FacesEhmmDB*.

- La habilidad puede ejecutarse tantas veces como se desee, sin embargo cada ejecución es única, no es cíclica y es exclusiva, sólo puede haber uno y solamente un

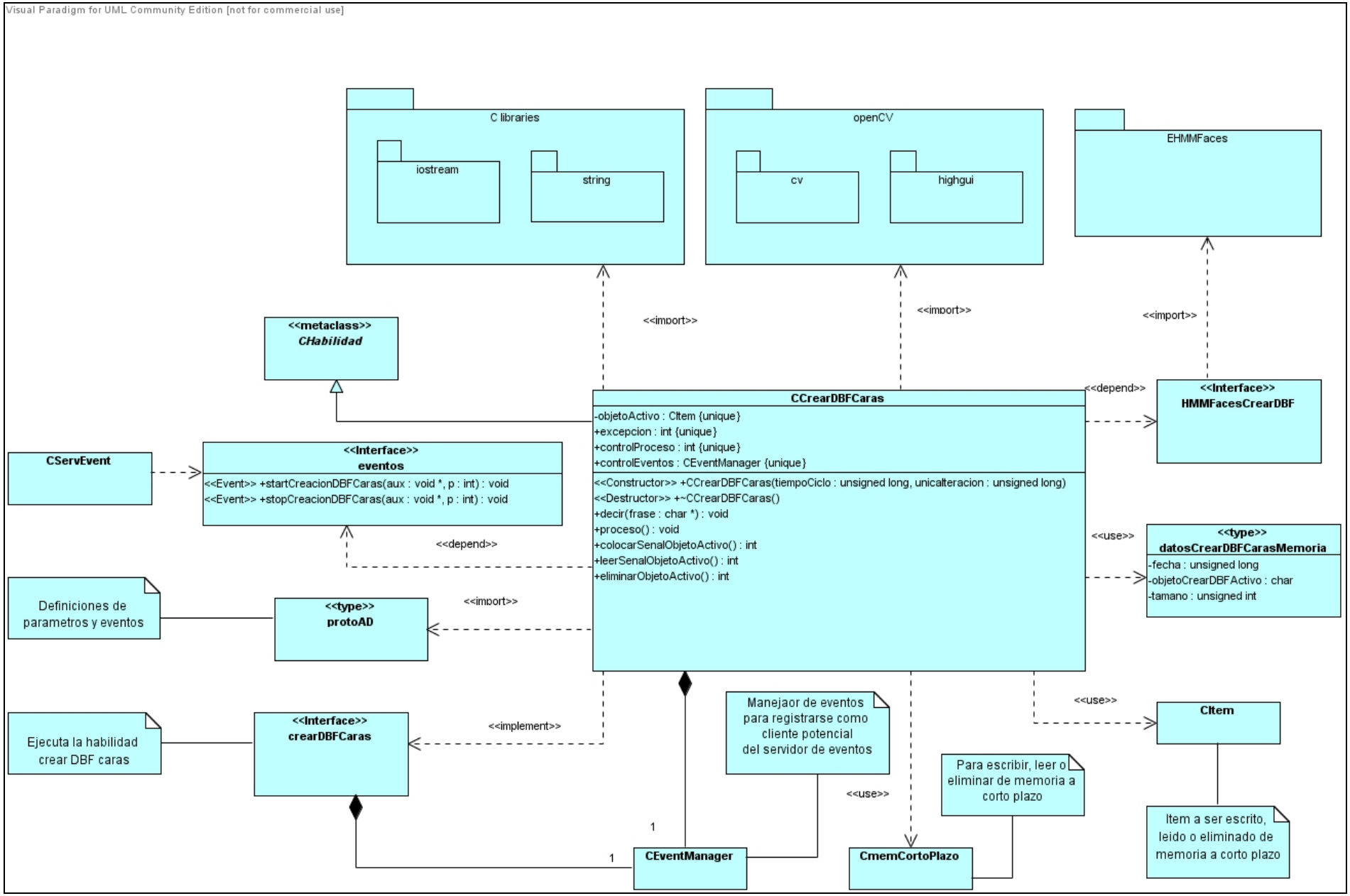

Figura No 3.9 Diagrama de Clases Habilidad Crear DBF Caras

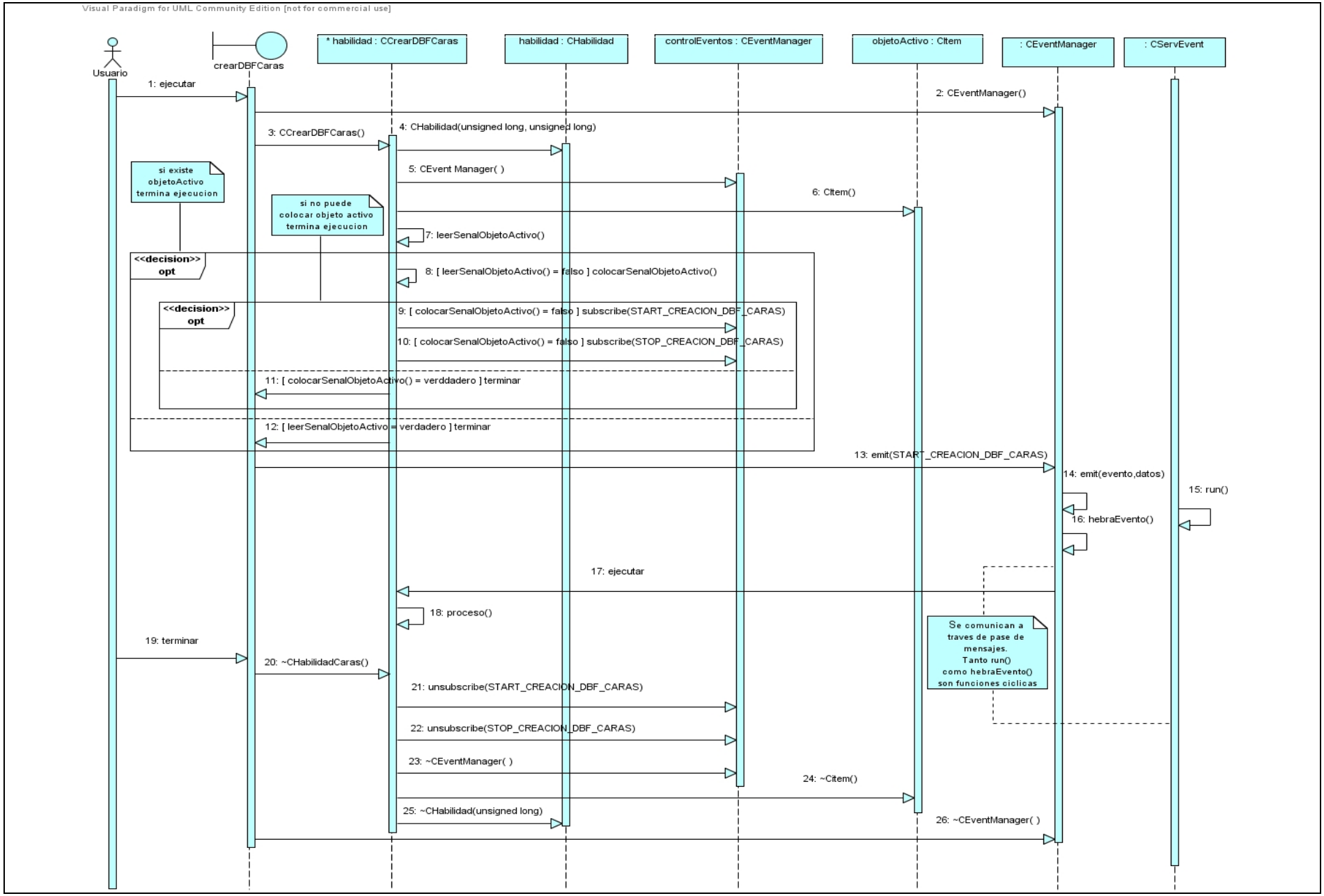

Figura No 3.10 Diagrama de Secuencia Habilidad Crear DBF Caras

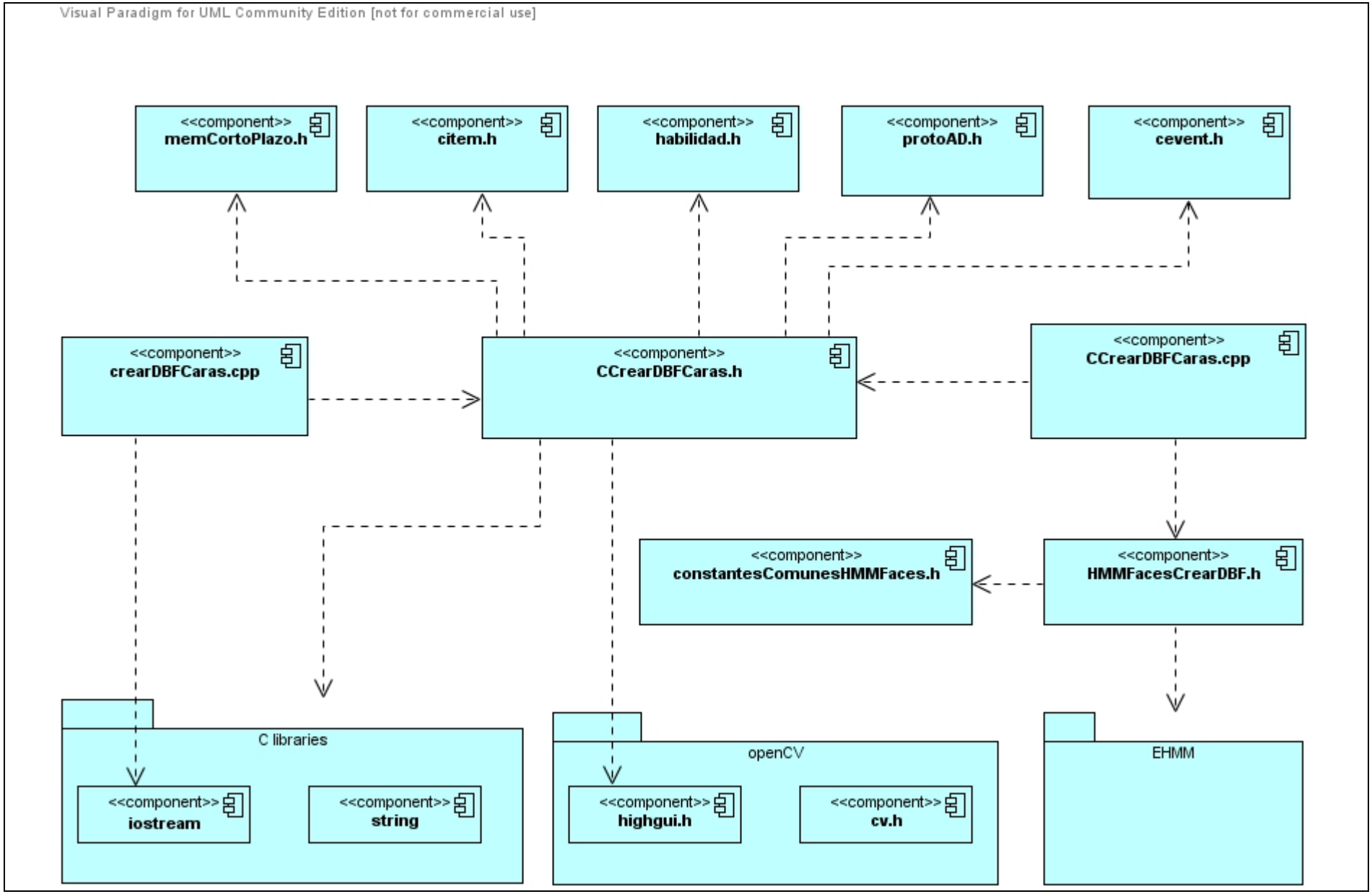

Figura No 3.11 Diagrama de Componentes Habilidad Crear DBF Caras

objeto actualizando la base de datos a la vez. Esto se debe garantizar a través de los siguientes mecanismos: 1) Colocando el parámetro *unicaIteracion* con valor 1 en el constructor de la habilidad *CCrearDBFCaras(unsigned long tiempoCiclo, unsigned long unicaIteracion).* 2) Definiendo funciones que no permitan la interrupción de la habilidad a través de las señales *SIGINT*, *SIGQUIT*. 3) Para asegurar que solamente una instancia de la habilidad esté actualizando la base de datos a la vez, el constructor de la habilidad coloca una señal, o cerrojo, en la memoria a corto plazo. 4) El proceso de la habilidad utiliza un semáforo para proteger su sección crítica.

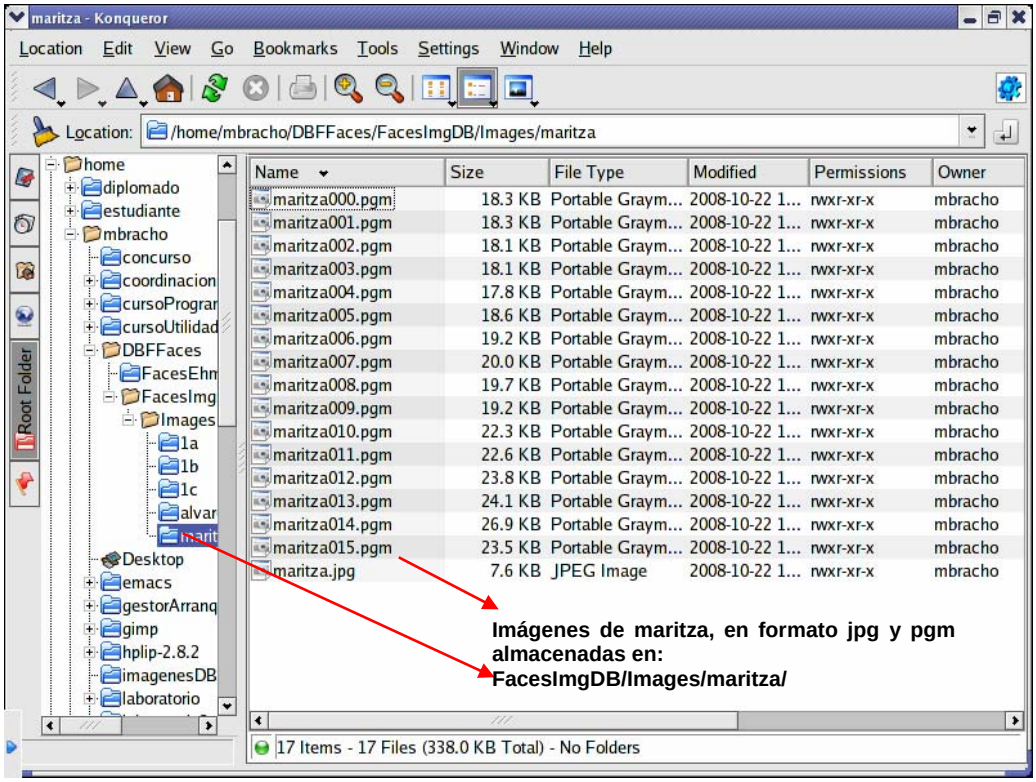

Figura No 3.12 Organización de las Imágenes Necesarias para Crear y Actualizar la Base de Datos de Rostros

El funcionamiento de esta habilidad es el siguiente:

- Se ejecuta el constructor de la clase para generar la instancia de la habilidad correspondiente.

- Para asegurar que solamente una instancia de la habilidad esté actualizando la base de datos a la vez, se coloca un cerrojo, o señal, en la memoria a corto plazo. Esto

es validado por el constructor y mientras esta señal exista, ninguna otra instancia de la habilidad podrá ser generada.

- Una vez que se valida el cumplimiento de la condiciones de unicidad y exclusividad, se suscriben ante el servidor de eventos y el administrador, los eventos y las funciones necesarias para ejecutar la habilidad: *1) START_CREACION_DBF_CARAS, startCreacionDBFCaras; 2)STOP_CREACION_DBF_CARAS, stopCreacionDBFCara.*

- Si hasta este momento no se ha producido un estado de excepción se procede con la ejecución de la habilidad. La creación y actualización de la base de datos con las imágenes de los rostros de las personas se ejecuta en el nivel más interno del sistema, identificado como el paquete *EHMMFaces*. El módulo *HMMFacesCrearDBF* contiene el nivel de control y de intercambio de información. La ejecución de los diferentes procesos de esta habilidad puede ser observada a través de una serie de mensajes emitidos por consola. Al finalizar, se elimina el cerrojo, o señal, colocado en la memoria a corto plazo al comienzo de la ejecución.

- Terminada la creación - actualización de la base de datos, se ejecuta el destructor de la habilidad.

- Al finalizar la ejecución, debe existir el directorio *FacesImgDB*, Figura No 3.13, con la base de datos *FACES.db* creada y un archivo de extensión *imobj* por cada una de las personas cuyos rostros han sido incluidos en la base de datos.

- Debe permanecer el directorio *FacesImgDB/Images* con los archivos iniciales usados en la creación de la base de datos.

- También deben existir en el directorio *FacesEhmmDB*, Figura No 3.14, la base de datos *FACES.db* creada y un archivo de extensión *ehmm* por cada una de las personas cuyos rostros han sido incluidos en la base de datos.

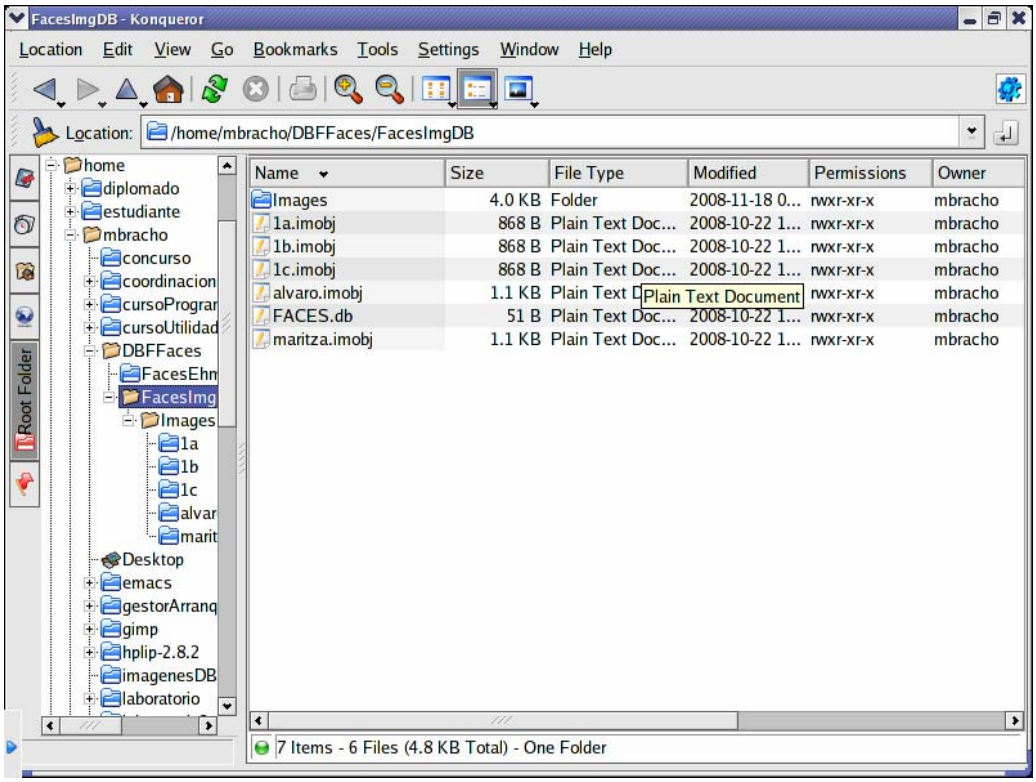

Figura No 3.13 Estado Final del Directorio FacesImgDB

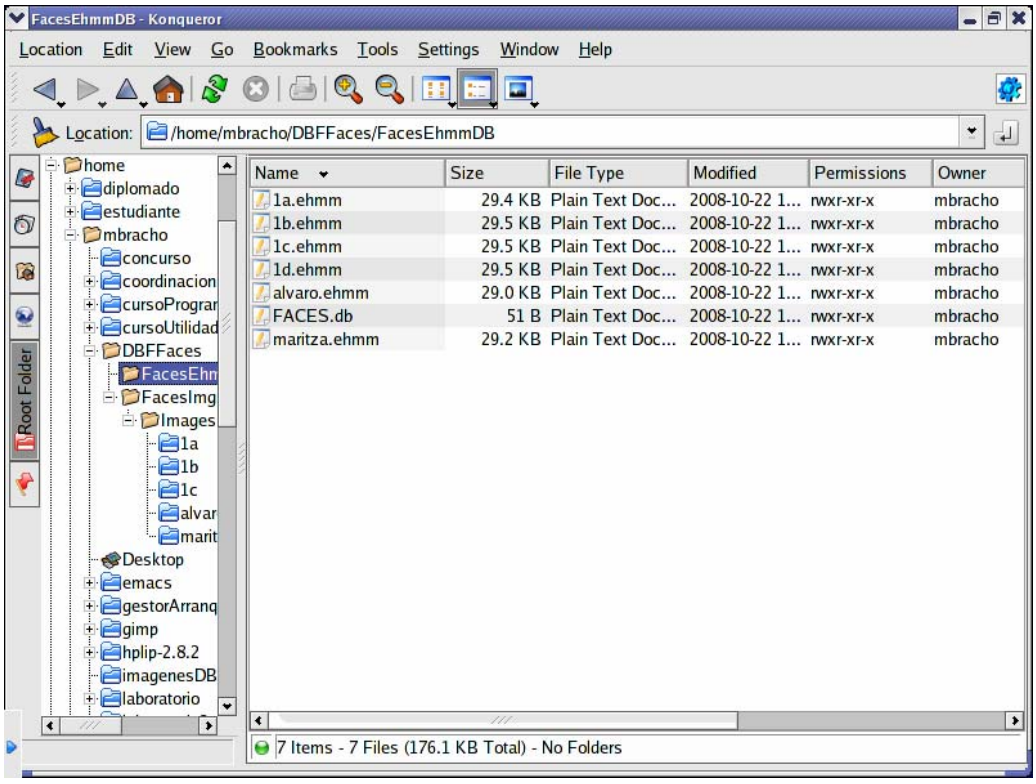

Figura No 3.14 Estado Final del Directorio FacesEhmmDB

3.3.1.3 Habilidad para Reconocimiento de Rostros

Esta habilidad es la encargada de reconocer a la persona, siempre y cuando su rostro pertenezca al dominio definido y la imagen de su cara se encuentre registrada en la base de datos. El Diagrama de Clases correspondiente a esta habilidad es mostrado en la Figura No 3.15, el Diagrama de Secuencia en la Figura No 3.16 y 3.17, el Diagrama de Componentes en la Figura No 3.18.

La habilidad tiene una interfaz, definida en el archivo *HMMFacesReconocer* que la comunica con la capa en la cual se encuentra implementado el modelo oculto de Markov embebido y que se identifica en los diagramas con el mismo nombre.

Su funcionamiento es el siguiente:

- Se ejecuta el constructor de la clase para generar la instancia de la habilidad correspondiente.

- Se verifica la existencia del haarCascade, - *haarcascade_frontalface_alt.xml* -, archivo indispensable para que los módulos de las librerías del openCV puedan detectar caras. Si existe una excepción, se suspende la ejecución de la habilidad

- Se verifica el estado del robot y si éste se encuentra en estado de emergencia o en algún estado no identificado se termina la ejecución.

- Se inician los procesos de la habilidad:

  a. Se crea el sensor para capturar imágenes desde el servidor de la cámara.

  b. Se verifica el estado del cuello del robot y se coloca la cabeza con la inclinación necesaria para capturar las imágenes.

  c. Se le otorga al controlador de la cámara unos segundos para que ajuste en forma automática los parámetros de la misma.

  d. Se verifica la existencia de la base de datos con los rostros de las personas.

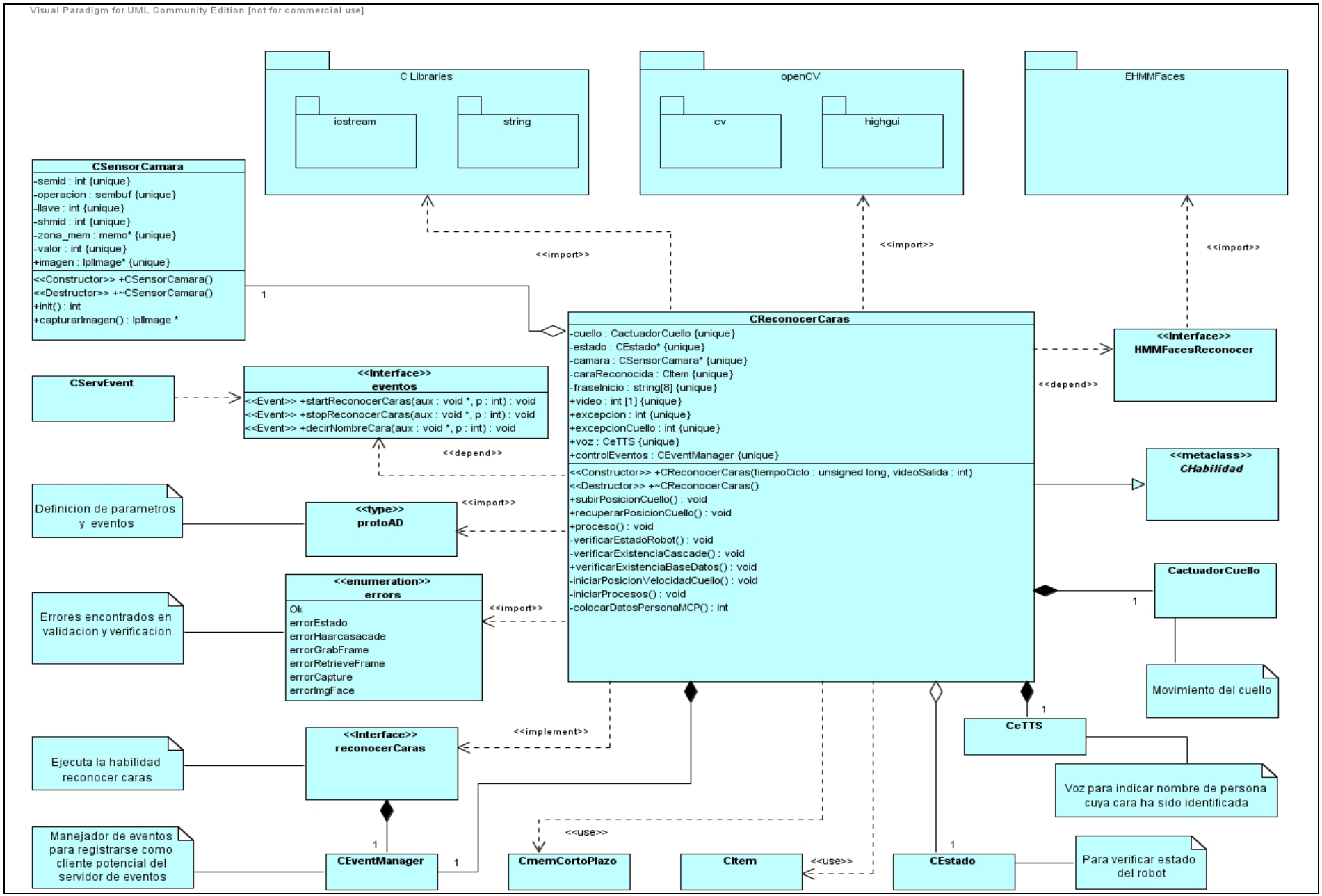

Figura No 3.15 Diagrama de Clases Habilidad Reconocer Caras

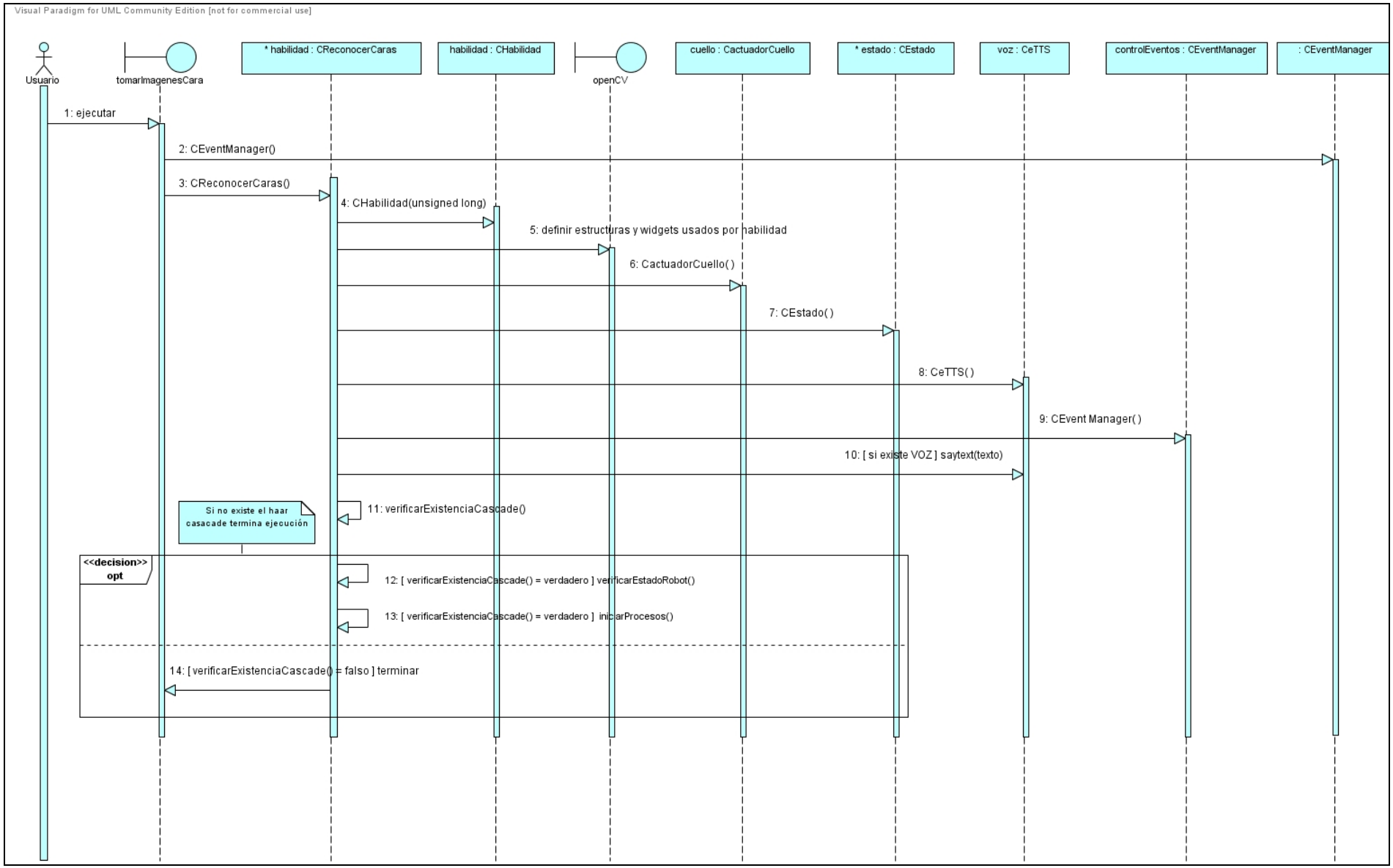

Figura No 3.16 Diagrama de Secuencia Habilidad Reconocer Caras: Ejecución del Constructor de la Habilidad

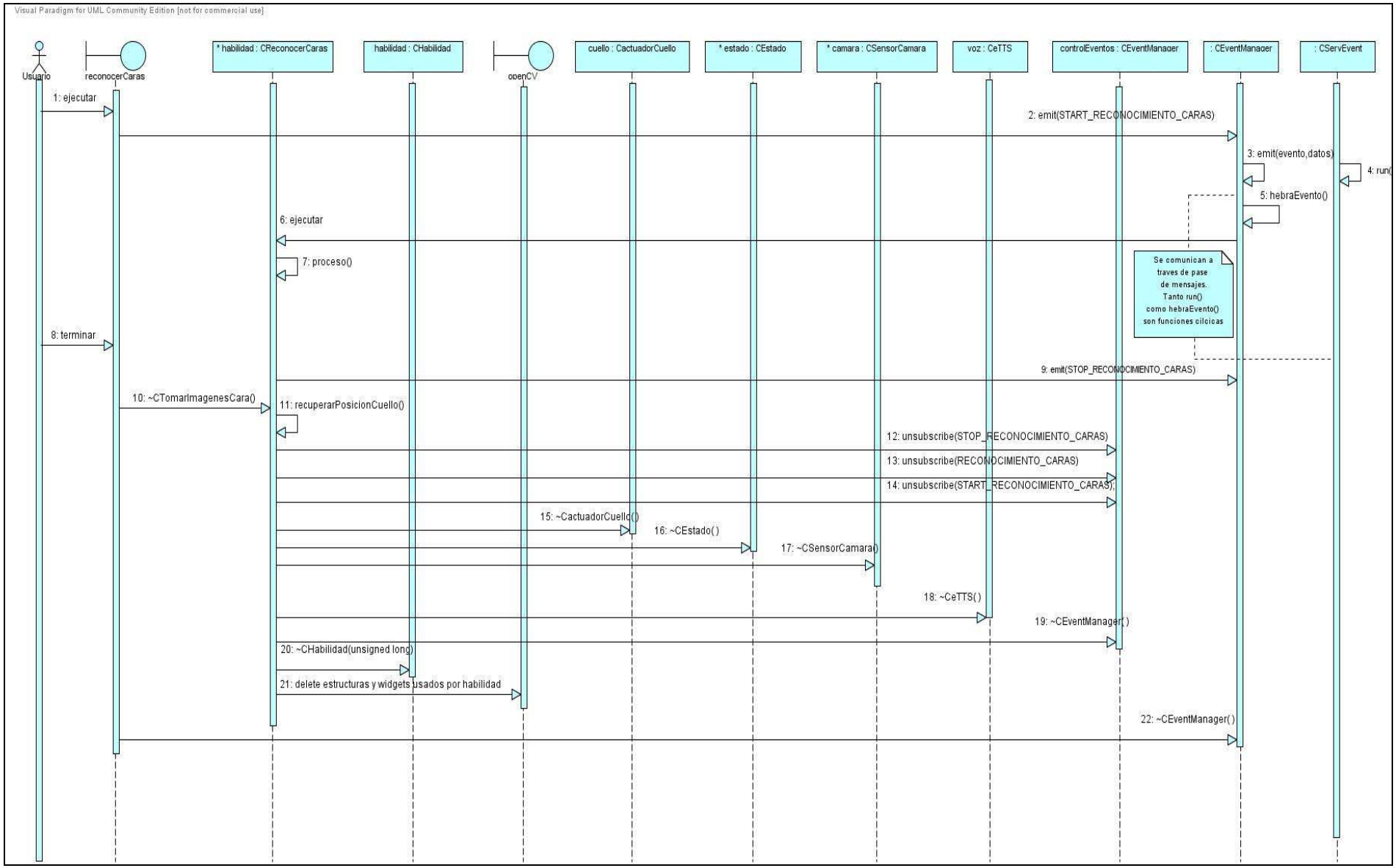

Figura No 3.17 Diagrama Secuencia Habilidad Reconocer Caras: Ejecución del Proceso y del Destructor

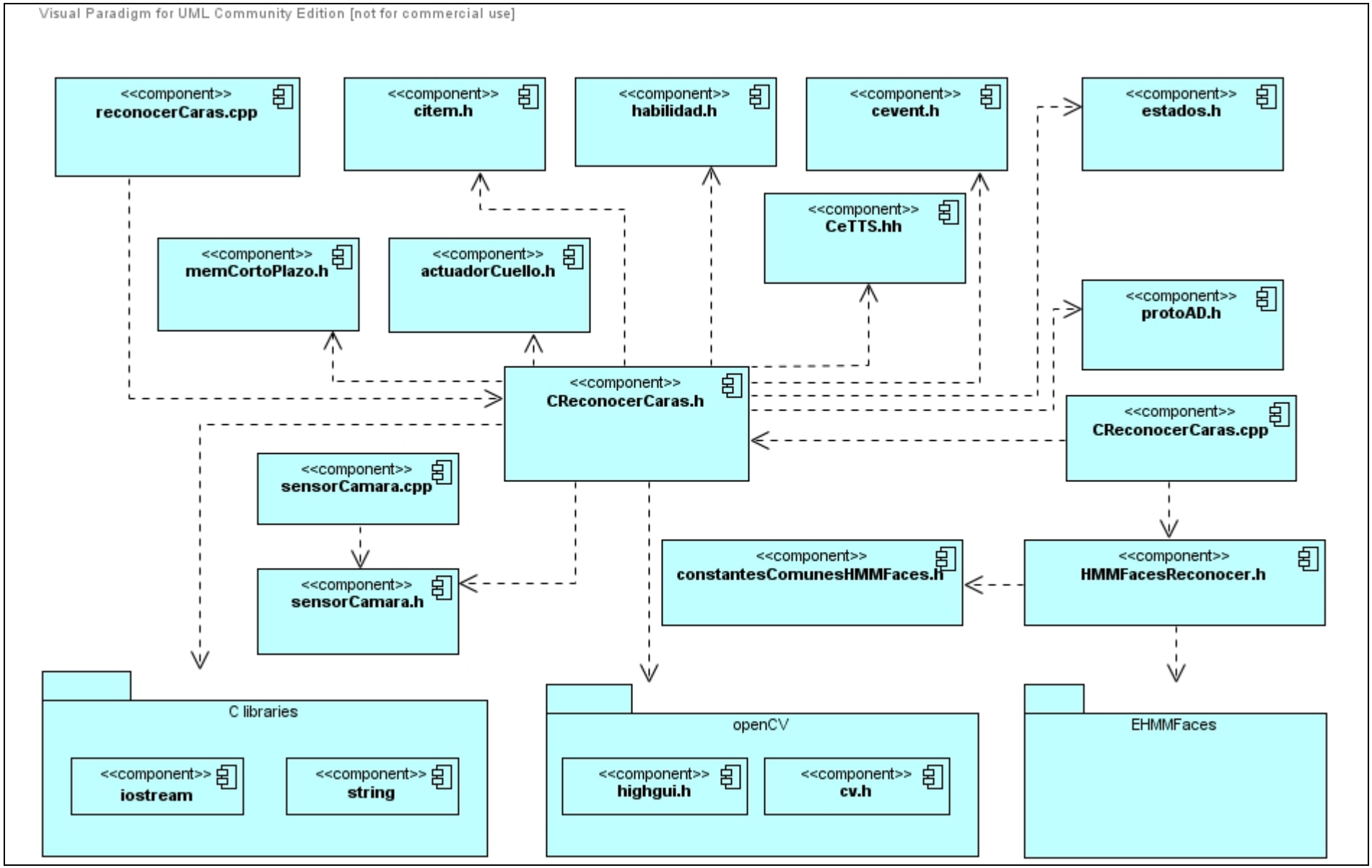

Figura No 3.18 Diagrama de Componentes Habilidad Reconocer Caras

Si ésta no existe, se produce una condición de excepción que no permite la ejecución de la habilidad.

e. Se suscriben ante el servidor de eventos y el administrador, los eventos y las funciones necesarias para ejecutar la habilidad: 1) *START_RECONOCIMIENTO_CARAS, startReconocerCaras*; 2) *RECONOCIMIENTO_CARAS, decirNombreCaras*; 3) *STOP_RECONOCIMIENTO_CARAS, stopReconocerCaras*.

- Se procede a la ejecución del proceso periódico de la habilidad: reconocimiento de rostros. Éste se ejecuta en el nivel más interno del sistema, en *EHMMFaces* y la interfaz *HMMFacesReconocer* contiene las instrucciones de control y de intercambio de información. Los resultados obtenidos durante la ejecución de esta habilidad, son notificados haciendo uso de tres medios: a través de la voz del robot, colocando mensajes en la memoria a corto plazo y en la consola de ejecución.

- Terminada la secuencia de reconocimiento de rostros se ejecuta el destructor de la habilidad.

3.3.2 Nivel Interfaces entre Habilidades y el Modelo EHMM

En este nivel se encuentran definidos los módulos que reciben los datos, comandos y eventos de las habilidades y los remiten al nivel modelo, para su debido procesamiento. Posteriormente reciben la información obtenida y procesada por el modelo y se la comunican a las habilidades para que puedan ser utilizadas por el resto de la arquitectura de control del robot. Está compuesto de tres módulos los cuales serán descritos a continuación.

3.3.2.1 HMMFacesTomarImagenes

Este módulo actúa de interfaz entre la habilidad para tomar imágenes de los rostros y el modelo oculto de Markov embebido. El módulo opera con funciones de la librería openCV para realizar la detección de la cara y el almacenamiento de las imágenes de rostros detectadas:

- La función *int FindFaces(IplImage* img)* es la encargada de detectar la cara haciendo uso de la función *cvHaarDetectObjects* de la librería openCV.

- El proceso *void SaveFace(IplImage* src, int x, int y, int w, int h)* es usado para almacenar las imágenes en colores, en formato *jpg.* Éste invoca a *void SavePgm(IplImage *gray, char *filename)* para guardar las imágenes en tonos de grises y formato *pgm*.

3.3.2.2 HMMFacesCrearDBF

Este módulo actúa de interfaz entre la habilidad para crear las bases de datos con los rostros de las personas y los objetos que implementan el modelo oculto de Markov embebido. Los métodos más importantes son los siguientes:

- Para controlar métodos asociados con la creación de la base datos con nombre de usuarios y rutas para las imágenes de los rostros, *void CreateFacesImgDb(const string &pathCvImgDb, const string &imgCvDbName)*

- Para controlar la carga imágenes de usuarios usando información almacenada en la base de datos, *void LoadImgDb(const string &pathCvImgDb, const string &imgDbName)*

- Para controlar métodos que se encargan de crear base de datos con objetos que implementan el modelo de Markov embebido, *void CreateFacesEHMMDb(const string &pathEhmmDb, const string &ehmmCvDbName)*

- Para controlar métodos de entrenamiento de la base de datos con objetos que implementan el modelo de Markov embebido, *void TrainEHMMDb(const string &pathImgDb, const string &imgDbName, const string &pathEhmmDb, const string &ehmmDbName)*

3.3.2.3 HMMFacesReconocer

Este módulo actúa de interfaz entre la habilidad para reconocer los rostros de las personas y el modelo oculto de Markov embebido. Sus componentes principales son los siguientes:

- La función *int FindFaces(IplImage* img)*, encargada de detectar la cara haciendo uso de la función *cvHaarDetectObjects* de la librería openCV.

- El método *void RecognizeEHMMDbFromCam(IplImage *imgCam, const string &pathEhmmDb, const string &ehmmDbName)*, encargado de controlar el reconocimiento de la imagen del rostro de la persona, capturada por la cámara, haciendo uso de la jerarquía objetos correspondiente en el paquete *EHMMFaces*.

3.3.3 Nivel Modelo Oculto de Markov Embebido

Es la capa más interna y en ella se encuentran definidas las jerarquías de clases, mostradas en las Figuras No 3.19 y No 3.21. Es en esta capa donde se define e implementa el modelo oculto de Markov embebido, representado por el paquete *EHMMFaces* en los diagramas hasta ahora presentados.

Los diagramas de clases y la definición de los elementos que componen este nivel han sido construidos luego de analizar, implementar y evaluar el conjunto de herramientas de software desarrolladas por Ou, [27], para el reconocimiento de rostros, en [14].

3.3.3.1 Jerarquías de Clases para Administrar Base de Datos con Imágenes de los Rostros

Estas jerarquías lógicas están formadas por las clases *Obj, ImgObj y ImgObjDatabase*. Su objetivo principal es el de permitir el acceso de una forma rápida, segura, ordenada y confiable a las imágenes de los rostros almacenadas en la base de datos, Figura No 3.19.

Asociado a esta jerarquía se encuentra, físicamente, el directorio *FacesImgDB*, Figura No 3.20, compuesto de los elementos siguientes: 1) los archivos con las imágenes de los rostros de las personas en formatos *pgm* y *jpg*, ordenados en carpetas con el nombre de la persona, almacenados en el subdirectorio *Images*. 2) Un archivo de extensión *imobj*, por cada una de estas personas. Éste contiene a su vez una línea por cada archivo de imágenes: en ésta se escribe la ruta completa y el nombre del fichero con la imagen. 3) El archivo con el nombre de la base de datos, de extensión *db*. Este archivo contiene registrado el nombre de cada una de las personas cuyos rostros se encuentran

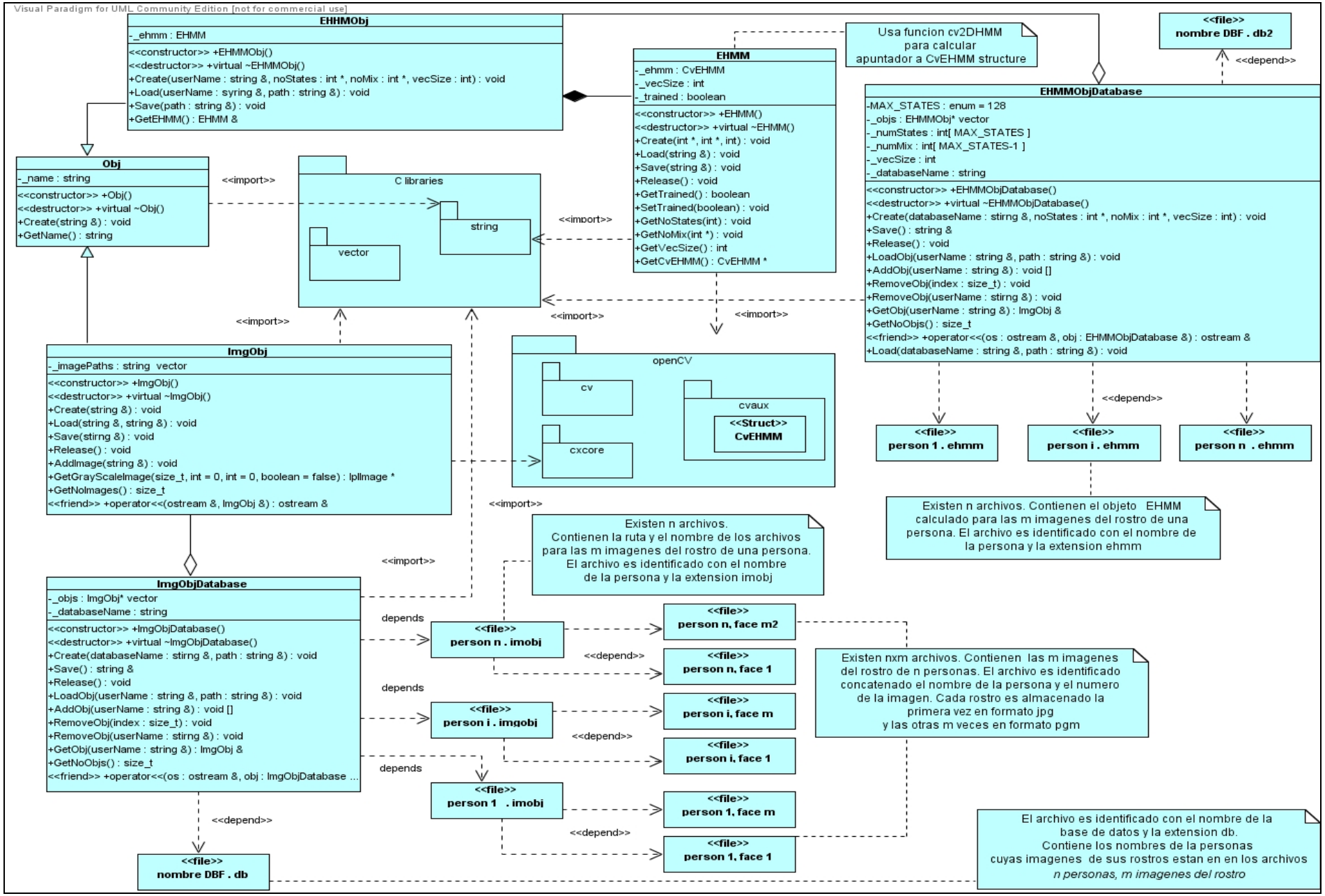

Figura No 3.19 Jerarquías de Clases EHMMFaces

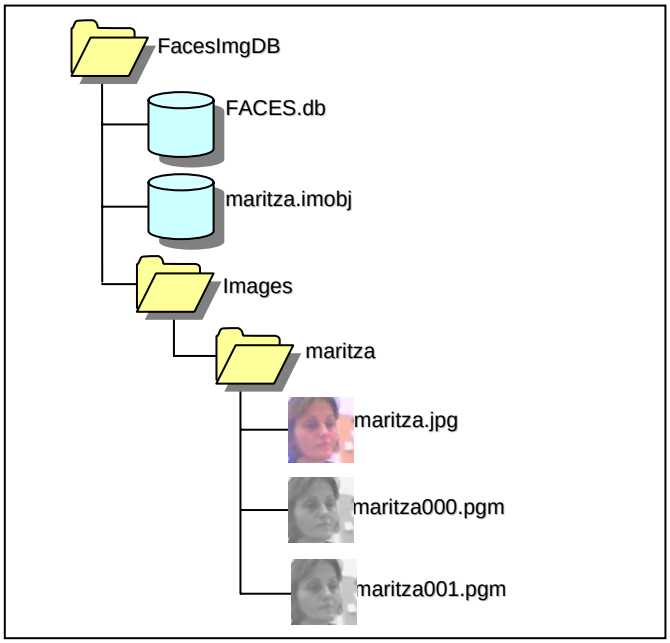

Figura No 3.20 Organización Física de Archivos de la Base de Datos de Imágenes

almacenados en la DBF.

Los métodos definidos para estas jerarquías se encargan de crear, cargar, almacenar, adicionar, liberar, remover, rutas y nombres de archivos con las imágenes de los rostros de las personas almacenadas en la base de datos.

Se hará mención especial del método *IplImage* ImgObj:: GetGrayScaleImage(size_t index, int imgWidth, int imgHeight, bool showImage)* en la clase *ImgObj*, porque haciendo uso de las librerías openCV, es el encargado de leer y cargar la imagen desde el archivo, convertirla en escala de grises y luego hacer un reescalamiento para colocarla en tamaño predefinido para todas las imágenes.

3.3.3.2 Jerarquías de Clases para Administrar Base de Datos con Objetos del Modelo Oculto de Markov Embebido

Estas jerarquías lógicas están formadas por las clases *Obj, EHMM, EHMMObj y EHMMObjDatabase*, Figura No 3.19. Su objetivo principal es el de crear y almacenar los objetos de tipo *EHMM* y permitir el acceso de una forma rápida, segura, ordenada y confiable a estos objetos almacenados en la base de datos.

Asociados a estas jerarquías lógicas, se encuentran, físicamente, los archivos que almacenan a los objetos *EHMM* para el conjunto de imágenes de los rostros de las personas. La extensión de estos archivos es *ehmm* y se encuentran almacenados el directorio *FacesEhmmDB*. El directorio incluye también al archivo de extensión *db*, con el nombre de la base de datos. Éste archivo contiene registrado el nombre de cada una de las personas, cuyos objetos y ficheros *EHMM* se encuentran almacenados en el directorio.

Los métodos definidos para las jerarquías lógicas formadas por estas clases se encargan de crear, cargar, almacenar, adicionar, liberar, remover, los archivos con la información de los objetos de tipo *EHMM* generados con las imágenes de los rostros de las personas almacenadas en la base de datos. Se hará especial mención de los siguientes métodos:

- El método *void EHMM::Create(int *noStates, int *noMix, int vecSize)* es el encargado de reservar el espacio en memoria y generar los objetos tipo *EHMM* haciendo uso de la función *CV_IMPL CvEHMM* cvCreate2DHMM(int *state_number, int *num_mix, int obs_size)*, definida en las librerías openCV.

- *void EHMM::Save(const std::string &fileName)* encargado de escribir los archivos de extensión *ehmm* con la información de los objetos tipo *EHMM* generados.

- *void EHMM::Load(const std::string &fileName)* encargado de leer los archivos de extensión *ehmm* con la información de los objetos tipo *EHMM* generados

3.3.3.3 Jerarquías de Clases para Entrenar los Objetos del Modelo Oculto de Markov Embebido

Estas jerarquías lógicas están formadas por las clases *ImgObj* e *ImgObjDatabase, EHMM*, *EHMMObj, EHMMObjDatabase,* y *EHMMObjRecognition*, Figura No 3.21. Su objetivo principal es entrenar a los objetos tipo *EHMMObj* y administrar los objetos de tipo *EHMMObjRecognition* permitiendo el acceso de una forma rápida, segura, ordenada y confiable a estos objetos. Los métodos definidos en la clase *EHMMObjRecognition* permiten crear, liberar y eliminar a los objetos de su tipo. Además, entrenan a los objetos tipo *EHMMObj* con las imágenes de los rostros de las personas almacenados en la base de datos.

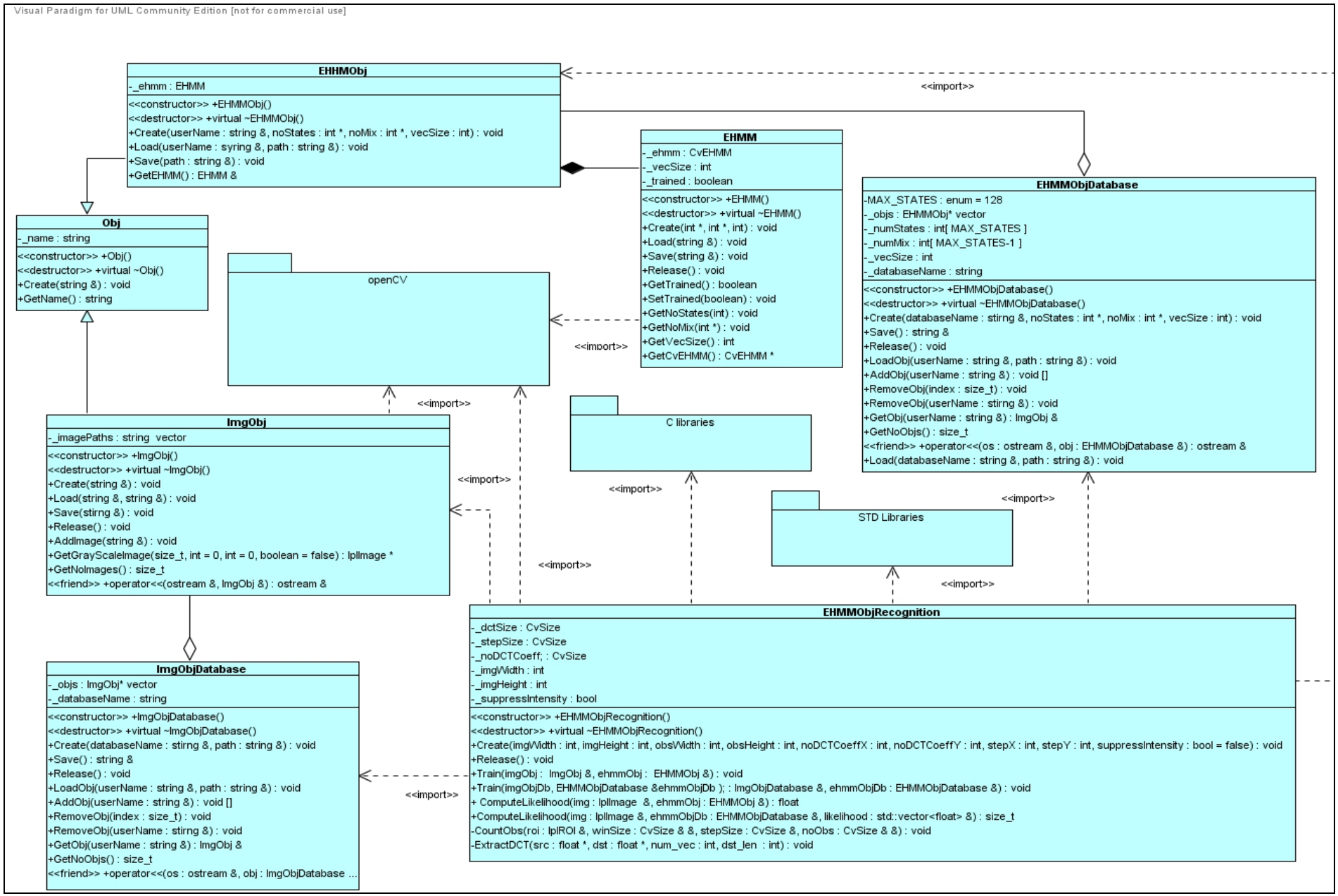

Figura No 3.21 Diagrama de Clases EHMMFaces para Entrenamiento y Reconocimiento de Rostros

Se hará especial mención de los siguientes métodos:

- Para contar el número de observaciones en un bloque en particular *void EHMMObjRecognition::CountObs(IplROI &roi, CvSize &winSize, CvSize &_stepSize, CvSize &_noObs)*

- Para extraer los coeficientes DCT *void EHMMObjRecognition::ExtractDCT(float* src, float* dst, int numVec, int dstLen)*

- Para entrenar cada uno de los objetos almacenados en *EHMMObjDatabase* con los objetos de *ImgObjDatabase*, *void EHMMObjRecognition::Train(ImgObj &imgObj, EHMMObj &ehmmObj*).

El entrenamiento de los *EHMMObj almacenados en EHMMObjDatabase* es realizado por el método *Train*. Para el entrenamiento se ejecutan las funciones y procesos descritos a continuación.

- Para cada imagen presente en la base de datos *ImgObjDatabase*:

a. Cargar imagen con *IplImage* ImgObj:: GetGrayScaleImage(size_t index, int imgWidth, int imgHeight, bool showImage)*.

b. Calcular el número de coeficientes bidimensionales DCT a generar usando el método *CountObs,* previamente definido.

c. Crear vector de observaciones de la imagen para cada las transformaciones con *CV_IMPL CvImgObsInfo* cvCreateObsInfo(CvSize num_obs, int obs_size).*

d. Generar los coeficientes DCT con *CV_IMPL void cvImgToObs_DCT(const void* arr, float *obs, CvSize dctSize, CvSize obsSize, CvSize delta)* . Toma como entrada una imagen y retorna la secuencia de observaciones a ser usada con el HMM embebido. El tipo de la matriz es especificado en la forma *CV_ <bit_depth> (S|U|F) C <number_of_channels>. Por ejemplo: CV_8UC1, 8 bits unsigned single channel; CV_32FC1, 32 bits float single channel*.

e. Suprimir el primer coeficiente DCT con el método *ExtractDCT* descrito anteriormente.

f. Ejecutar segmentación uniforme de la imagen *CV_IMPL void cvUniformImgSegm(CvImgObsInfo * obs_info, CvEHMM * hmm)* . (Esta función tiene algunos comentarios a nivel de la librería openCV).

g. Liberar la imagen.

- Una vez que se han procesado todas las imágenes relacionadas con un *EHMMObj* en particular, se realiza la mixtura de la segmentación de los estados del HMM embebido *CV_IMPL void cvInitMixSegm(CvImgObsInfo ** obs_info_array, int num_img, CvEHMM * hmm)*.

- Se ejecuta el siguiente procedimiento iterativo de entrenamiento hasta que el valor del estimador de similitud entre dos iteraciones sucesivas alcance un valor mínimo definido o se ejecuten el máximo de iteraciones preestablecido.

a. Estimación de los parámetros de estado del HMM con *CV_IMPL void cvEstimateHMMStateParams(CvImgObsInfo ** obs_info_array, int num_img, CvEHMM * hmm)*

b. Estimación de las probabilidades de transición con *CV_IMPL void cvEstimateTransProb(CvImgObsInfo ** obs_info_array, int num_img, CvEHMM * hmm)* Esta función calcula las probabilidades de transición del estado y del superestado del modelo, la segmentación del estado y los parámetros de entrada.

c. Para cada una de las imágenes del conjunto, por objeto se efectúan las siguientes funciones: 1) Cálculo de la probabilidad de observación en cada estado. *CV_IMPL void cvEstimateObsProb(CvImgObsInfo * obs_info, CvEHMM * hmm)* 2) Aplicación del algoritmo de Viterbi embebido a cada imagen *static float CV_STDCALL icvEViterbi(CvImgObsInfo* obs_info, CvEHMM* hmm)*.

d. Cálculo del estimador de similitud

e. Cálculo de la mixtura de la segmentación de los estados del HMM embebido, *static CvStatus CV_STDCALL icvMixSegmL2(CvImgObsInfo** obs_info_array, int num_img, CvEHMM* hmm)*

f. Evaluación de la convergencia para determinar si el *EHMMObj* está entrenado o si debe mantenerse el ciclo de entrenamiento.

- Una vez que finaliza este procedimiento el objeto es marcado como entrenado y es almacenado nuevamente en *EHMMObjDatabase*

3.3.3.4 Jerarquías de Clases Utilizadas para Reconocer los Rostros

Estas jerarquías lógicas están formadas por las clases *Obj, EHMM, EHMMObj, EHMMObjDatabase*, y *EHMMObjRecognition*, Figura No 3.21. Su objetivo principal es el de proporcionar los modelos de los rostros almacenados en la base de datos con el propósito de compararlos con la imagen presentada y determinar si el rostro de la persona puede ser reconocido.

El reconocimiento es efectuado por el método *size_t EHMMObjRecognition ::ComputeLikelihood(IplImage &img, EHMMObjDatabase &ehmmObjDb, vector< float > &likelihood).* Para ello se ejecutan las funciones y procesos descritos a continuación:

- Marcar al objeto como no entrenado aún.

- Para cada objeto presente en la *EHMMObjDatabase* invocar a la función *float EHMMObjRecognition::ComputeLikelihood(IplImage &img, EHMMObj &ehmmObj)* para calcular el estimador de similitud:

a. Calcular el número de coeficientes bidimensionales DCT a generar usando el método *void EHMMObjRecognition::CountObs(IplROI &roi, CvSize &winSize, CvSize &_stepSize, CvSize &_noObs).*

b. Crear vector de observaciones de la imagen para cada las transformaciones con *CV_IMPL CvImgObsInfo* cvCreateObsInfo (CvSize num_obs, int obs_size).*

c. Generar los coeficientes DCT con *CV_IMPL void cvImgToObs_DCT(const void* arr, float *obs, CvSize dctSize, CvSize obsSize, CvSize delta)* . Toma como entrada una imagen y retorna la secuencia de observaciones a ser usada con el HMM embebido. El tipo de la matriz es especificado en la forma *CV_ <bit_depth> (S|U|F) C <number_of_channels>. Por ejemplo: CV_8UC1, 8 bits unsigned single channel; CV_32FC1, 32 bits float single channel*.

d. Invocar al método *void EHMMObjRecognition::ExtractDCT(float* src, float* dst, int numVec, int dstLen)* para extraer los coeficientes DCT

e. Calcular de la probabilidad de observación en cada estado. *CV_IMPL void cvEstimateObsProb(CvImgObsInfo * obs_info, CvEHMM * hmm)*

f. Aplicar el algoritmo de Viterbi embebido *static float CV_STDCALL icvEViterbi(CvImgObsInfo* obs_info, CvEHMM* hmm).*

- Almacenar el estimador de similitud recién calculado. Verificar si este estimador presenta el máximo valor obtenido hasta el momento. Si este es el caso lo registra como valor máximo.

- Repetir hasta recorrer la totalidad de la base de datos *EHMMObjDatabase.*

- Para finalizar, el método de reconocimiento retorna el máximo valor de similitud calculado, así como el vector con todos lo estimadores calculados.

3.4 Servidor de la Cámara y Sensor Cámara

En las Figuras No 3.22 y 3.23 se muestran las clases y relaciones entre los artefactos instalados en el robot Maggie para proveerle de visión artificial: cámara, servidor y sensor cámara.

Al momento de incorporar estas habilidades a la arquitectura AD, la cámara es una Logitech QuickCam Pro 5000 y el sistema operativo es la distribución de Linux, Fedora Core 8. El controlador usado es el USB video Class, driver UVC, [19], el cual incluye el manejador del dispositivo V4L2 para el kernel, diferentes elementos y herramientas para el usuario.

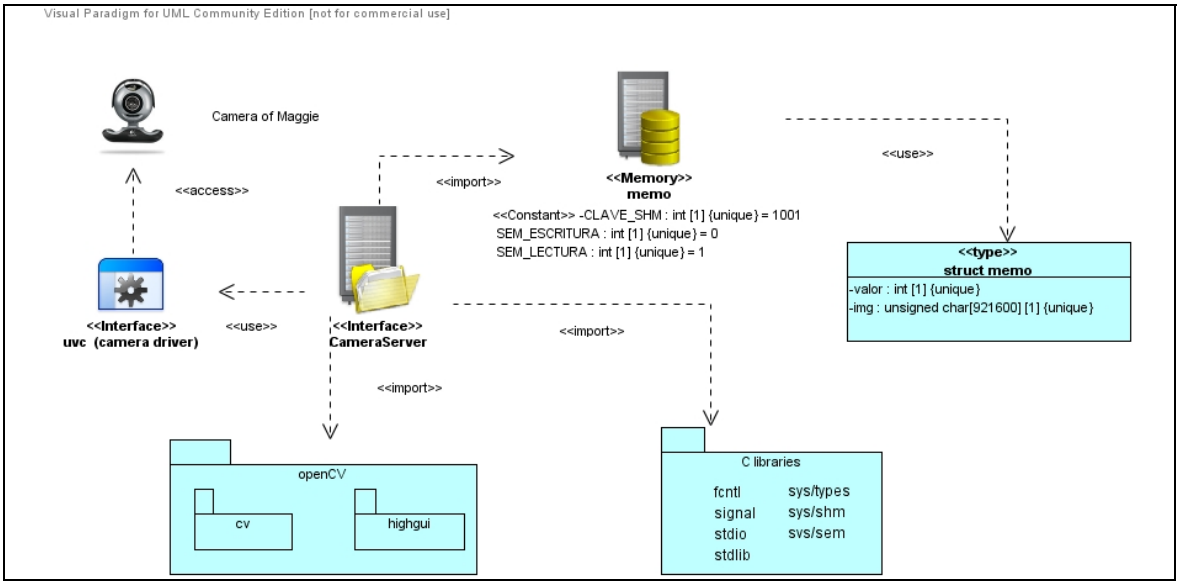

Figura No 3.22 Controlador y Servidor de la Cámara para Maggie

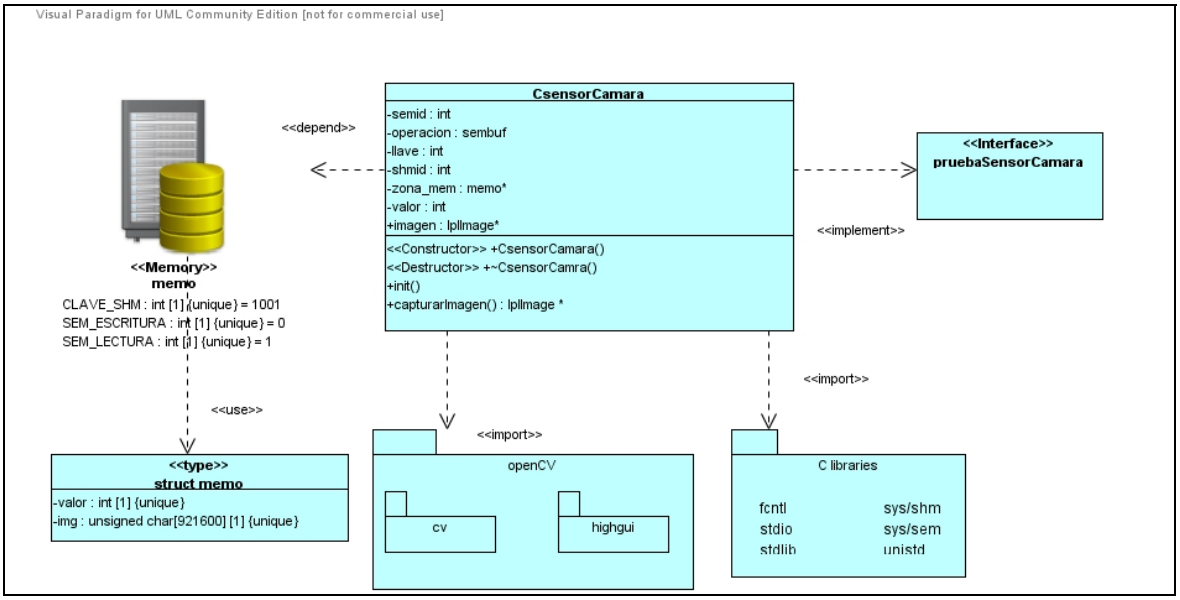

Figura No 3.23 Sensor Cámara para Maggie

La cámara es accedida, a través del controlador, por un servidor que almacena en memoria las imágenes capturadas por la cámara, en forma cíclica. Esta área de memoria puede a su vez ser accedida, a través de objetos instanciados de la clase CsensorCamara. Esta jerarquía de clases y componentes permite la distribución concurrente o paralela de la imagen capturada por la cámara, hacia los diferentes elementos componentes de la arquitectura AD.

Capítulo 4

ENSAYOS Y ANÁLISIS DE RESULTADOS

4.1 Diseño de los Experimentos

Con el fin de de observar el comportamiento del software de visión artificial desarrollado e implementado, evaluar los algoritmos y el rendimiento de la arquitectura propuesta para las habilidades de Reconocimiento de Rostros, se ejecutaron varias series de experimentos. Para cada uno de ellos se conformaron escenarios aleatorios, con condiciones de iluminación distintas, pero controladas, y posiciones diferentes tanto para el robot, como para el rostro de las personas a ser identificadas. Los resultados obtenidos son mostrados y discutidos en las secciones siguientes.

Maggie está explícitamente diseñada para la interacción hombre robot por lo tanto es importante señalar que en el funcionamiento de las habilidades, no solamente se considera el resultado lógico de la ejecución de las mismas. También se evalúa el tiempo en el cual se produce este resultado. Este tiempo de respuesta se mide usando la misma escala con la que se determina el tiempo en que ocurren los eventos para los seres humanos. Por esta razón los resultados de los experimentos son presentados y discutidos en términos de la complejidad temporal y el tiempo de ejecución.

Las habilidades en el momento de su ejecución no presentan complejidad espacial, por lo tanto no es de interés particular en esta etapa del proyecto, su análisis y evaluación. Una situación diferente se presenta con las bases de datos necesarias para el funcionamiento de las habilidades. Consideraciones sobre la complejidad espacial de las mismas son realizadas en el análisis de resultados.

Los ensayos se efectuaron sobre un Acer, Travel Mate 3043WTMI, con procesador Intel Core 2 Duo T5500, 1.66 GHz, 1GB de memoria, Intel Graphics Media Acelerator 950, sistema operativo Fedora Core 8, kernel 2.6.23.9 y disco duro de 120GB.

Todos los experimentos han sido ejecutados tomando como muestras a personas del Robotics Lab, para quienes las imágenes de sus rostros están o no están almacenadas en las bases de datos e imágenes estándares suministradas por Ou, [27] para este tipo de experimentación. Los datos requeridos por las habilidades han sido tomados de las imágenes capturadas, en tiempo real, por el sensor cámara del robot.

4.2 Habilidad para Tomar Imágenes de Rostros: Análisis de Tiempo de Ejecución y Tasa de Efectividad

El tiempo promedio empleado por esta habilidad para tomar la imagen del rostro de la persona y almacenarla en la memoria es de 688ms, con un máximo de 6.171ms y un mínimo de 585ms. En las gráficas, Figuras No 4.1 y No 4.2, se muestra el comportamiento de la habilidad. Nótese como de los 688ms, 384ms se invierten en la segmentación de la cara, con un rango de 282ms a 5.867ms, mientras que el proceso de captura de la imagen de la cara y almacenamiento en memoria a largo plazo necesita de 304ms, con un rango de 303ms a 305ms. El robot está en capacidad de tomar y almacenar 124 imágenes de un rostro en un tiempo de 1,43 minutos.

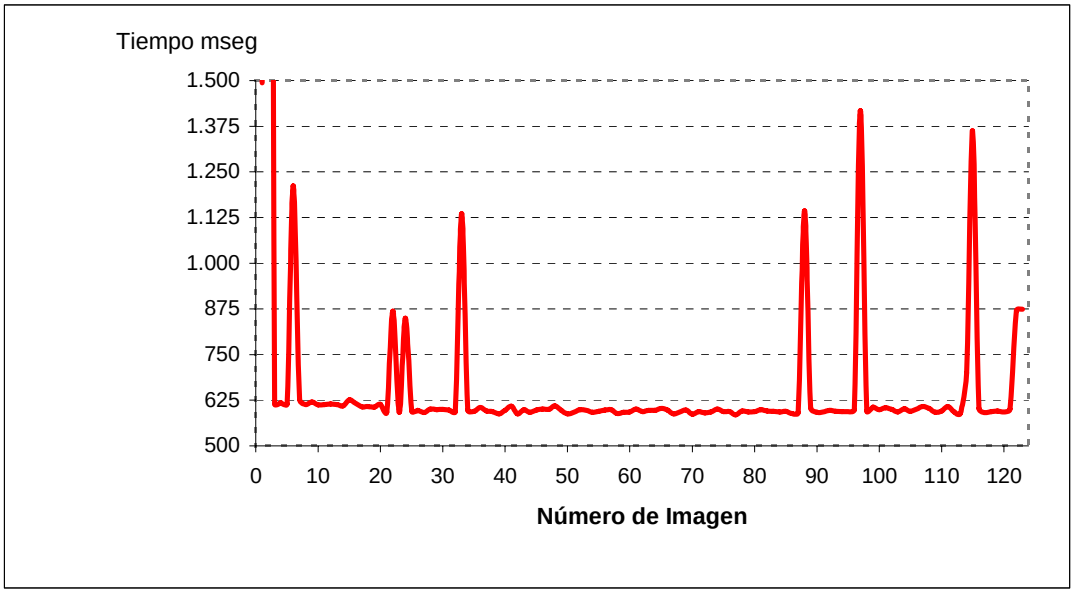

Figura No 4.1 Tiempo Promedio Total Empleado en Tomar Imagen del Rostro de la Persona

Los resultados mostrados indican que el proceso de detección de la cara puede requerir de varias imágenes en una secuencia para segmentar el rostro. Esta situación se presenta, generalmente, al comienzo de una secuencia y es un resultado esperado, puesto que es en este momento cuando el robot está observando escenas abarrotadas de objetos y debe segmentar el rostro de la persona en ellas. Una vez iniciada la secuencia pueden producirse retardos en la detección del rostro porque la persona realiza movimientos bruscos de la cabeza ocasionando la pérdida del enfoque de la cámara sobre el rostro. Por este motivo se recomienda que antes de la ejecución de la habilidad se coloque a la persona frente al robot, separada por una distancia de 1m, preferiblemente sentada en una silla, con la espalda recta, mirando la boca del robot, para facilitar el proceso de detección, captura y almacenamiento de la cara. La persona no debe estar inmóvil y es posible realizar movimientos lentos y suaves de la cabeza, con giros máximos de ±30º con respecto al plano horizontal y de ±30º con respecto al plano vertical.

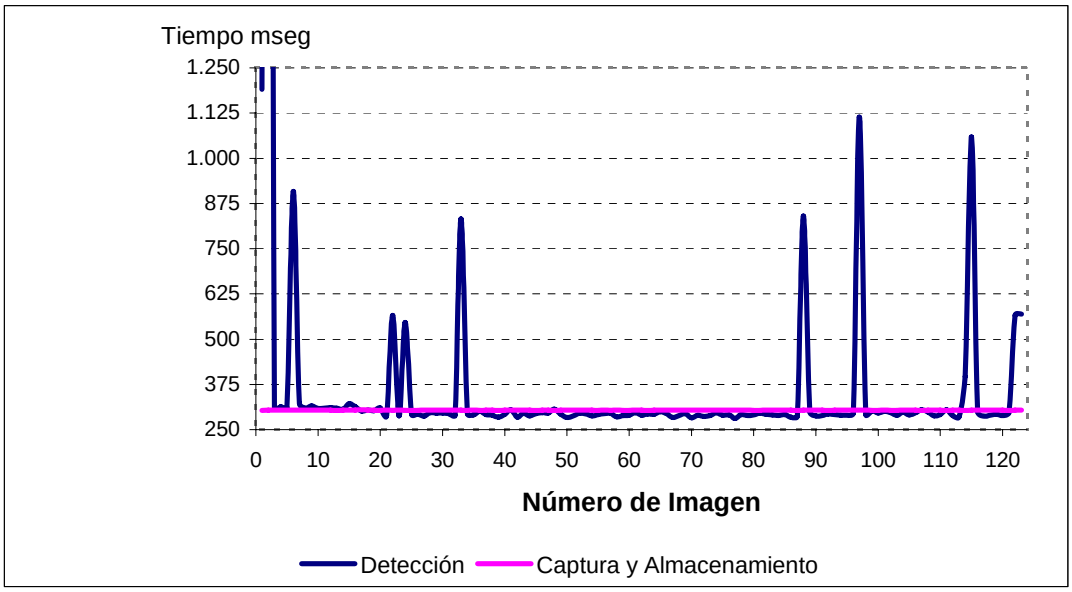

Figura No 4.2 Tiempo Promedio Empleado en Detección, Captura y Almacenamiento de la Imagen del Rostro de la Persona

En lo que se refiere a la calidad de las imágenes, en los experimentos se encontró una tasa de efectividad de 0,7302. Esto indica que el 73% de los frames capturados son utilizables. Esta tasa depende del comportamiento de la persona frente al

robot: debe mantener la posición donde ha sido localizada por el robot y no debe hacer movimientos bruscos porque la cámara pierde el encuadre del rostro.

4.3 Habilidad para Crear las Bases de Datos con los Rostros de las Personas: Análisis de Tiempo de Ejecución y Ocupación de Memoria a Largo Plazo

Para realizar el análisis y evaluación de los resultados experimentales de esta habilidad, tres etapas de la misma son consideradas: 1) Creación de la base de datos con imágenes de las caras. 2) Creación de la base de datos con los modelos ocultos de Markov embebidos para cada una de las caras, objetos EHMM. 3) Entrenamiento de los objetos EHMM.

Las mediciones de los tiempos de ejecución se realizaron sobre una serie de experimentos que incluyeron 28 imágenes del rostro por persona y un incremento progresivo de las personas en la bases de datos: 5, 10, 15, 20, y 25 personas. En la Figura No 4.3 se muestran los tiempos de ejecución promedio para cada una de estas etapas.

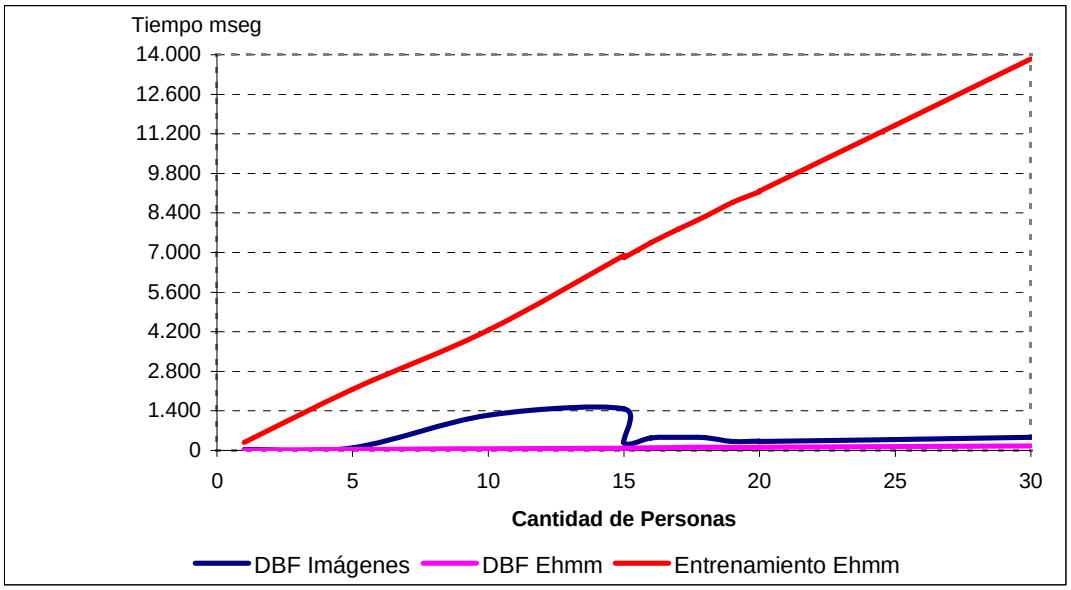

Figura No 4.3 Tiempo Promedio Empleado en Crear Bases de Datos de Rostros

El tiempo promedio de ejecución es el siguiente: 1) para la creación de la base de datos de imágenes de las caras es de 538ms, con un rango de variación de 33 a

1.473ms; 2) el de creación de la base de datos de objetos EHMM es de 89ms y un rango de 16 a 161ms; 3) el de entrenamiento es de 7.068ms y un rango de 287 a 13.852ms. Este último resultado está completamente justificado por la complejidad de los cálculos efectuados en esta etapa.

Dentro de la arquitectura AD implementada hasta ahora, las habilidades de los rostros son las primeras en generar información para ser almacenada en la Memoria a Largo Plazo. Estas habilidades tiene dos bases de datos: la primera almacena las imágenes de las personas y los registros necesarios para su localización e indexación. En la segunda se guardan los modelos de Markov embebidos generados y entrenados a partir de las imágenes de los rostros y los registros necesarios para su localización e indexación. Una proyección de la complejidad espacial de estas bases de datos es mostrada en la Figura No 4.4, para 28 imágenes de rostros por persona.

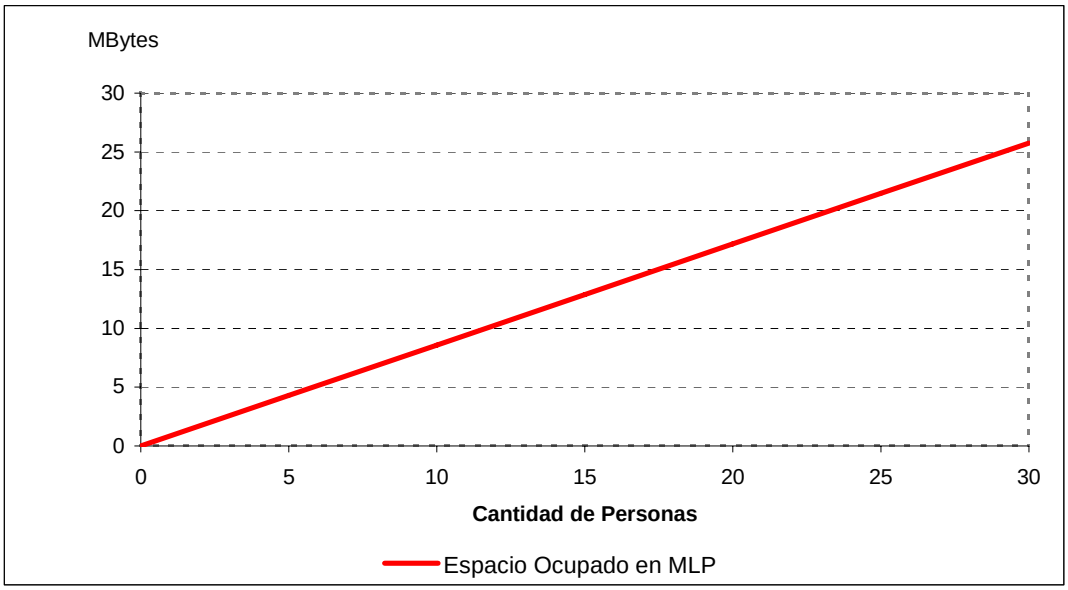

Figura No 4.4 Complejidad Espacial para DBF de Rostros

Estas bases de datos ocupan el siguiente espacio: 1) Por cada persona se guarda una imagen del rostro, en colores y en formato jpg. Esta imagen ocupa un máximo de 8KB. 2) Cada imagen del rostro es almacenada en escala de grises, en formato pgm, en un máximo de 30KB. 3) Por cada persona se genera un objeto tipo imobj de 1,2KB. 4) Existe un objeto tipo ehmm de 30KB, por cada persona. 5) Existen dos archivos de tipo

db. En uno de ellos se resume la información de las imágenes u objetos tipo imobj, en el otro la información de los objetos EHMM. Estos archivos tienen un tamaño de 50 bytes cada uno, (para 15 personas), pero este tamaño se incrementa en la medida que crece la lista de personas. Se estima un máximo de 1KB para estos archivos. El cálculo del tamaño del espacio en memoria a largo plazo ocupado por estas bases de datos es el siguiente: *(8KB + 30KB * cantidad de rostros almacenados + 1,2KB + 30KB)*cantidad de personas + 1KB.*

4.4 Habilidad para Reconocimiento de Rostros: Análisis de Tiempo de Ejecución y Nivel de Confiabilidad

El tiempo promedio empleado por esta habilidad para reconocer un rostro es de 434ms, con un mínimo de 352ms y un máximo de 1.235ms. En la gráfica de la Figura No 4.5 se muestra el comportamiento de esta habilidad en lo que se refiere a tiempo de ejecución.

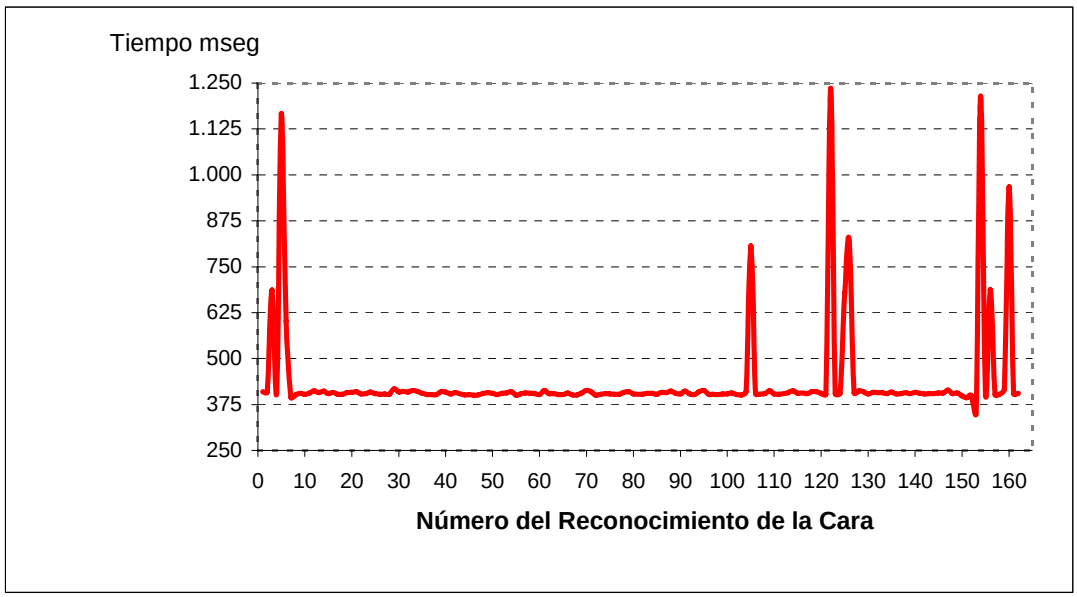

Figura No 4.5 Tiempo Promedio Empleado en Reconocer Rostros

Pueden observarse máximos locales en este tiempo de ejecución ocasionados por la función de detección de caras. Como se comentó previamente, en el numeral 4.2, estos retrasos se presentan cuando falla el proceso de detección de la cara o cuando se requieren varias imágenes en una secuencia para segmentar el rostro. Por este motivo,

nuevamente se recomienda que al ejecutar las habilidades de los rostros, se coloque a la persona frente al robot, separada por una distancia de 1m, preferiblemente sentada en una silla, con la espalda recta, mirando la boca del robot, para facilitar su proceso y para asegurar el resultado en la ejecución.

En la Figura No 4.6 se muestra como en series de reconocimientos consecutivas se comporta el nivel de confianza con el cual la habilidad está reconociendo el rostro. En este experimento se le ha pedido al robot que reconozca a personas cuyos rostros están efectivamente almacenados en las bases de datos.

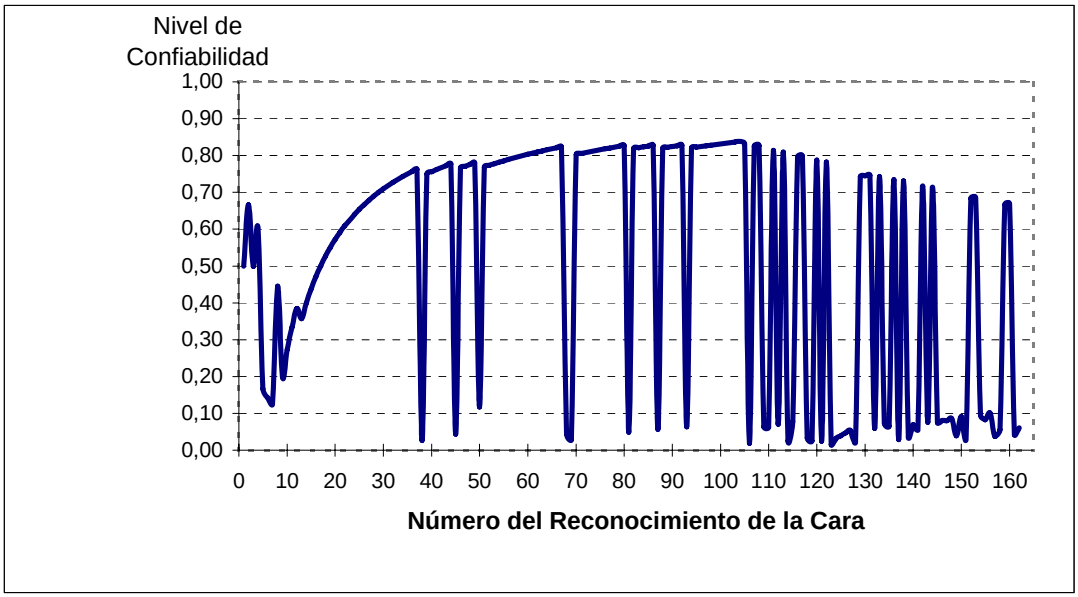

Figura No 4.6 Nivel de Confiabilidad para Reconocimiento de Rostros: Caras SI están Almacenadas en la Base de Datos

Los resultados mostrados se interpretan de la manera siguiente: por ejemplo, en el reconocimiento número 30, la habilidad identificó a una persona, - correctamente -, con una confiabilidad del 71%, mientras que en el reconocimiento número 38, identificó a otra persona, - incorrecta -, con una confiabilidad del 2,5%.

El análisis de esta serie de experimentos ha permitido definir las siguientes reglas de decisión: el robot habrá identificado de manera correcta a la persona cuando el nivel de confianza sea superior al 70%. El robot no habrá identificado a la persona cuando el nivel de confianza sea inferior al 25%. Cuando el nivel de confianza se encuentra entre el

70% y 50%, el robot habrá identificado a la persona con una probabilidad de error del 0,20. Finalmente, cuando el nivel de confianza se encuentra entre el 50% y 25%, es necesario suministrarle al robot mayor cantidad de información, para confirmar la identificación del rostro presentado.

La Figura 4.7, muestra el número de reconocimientos de rostros correctos o acertados en series de 10. El significado de este ensayo es el siguiente: se le presenta al robot un rostro, el cual se encuentra registrado en la base de datos, para su identificación. En la serie 1 sólo reconoció de manera correcta 3 de las 10 veces, mientras que en la serie 3 reconoció el rostro de manera correcta 10 de las 10 veces.

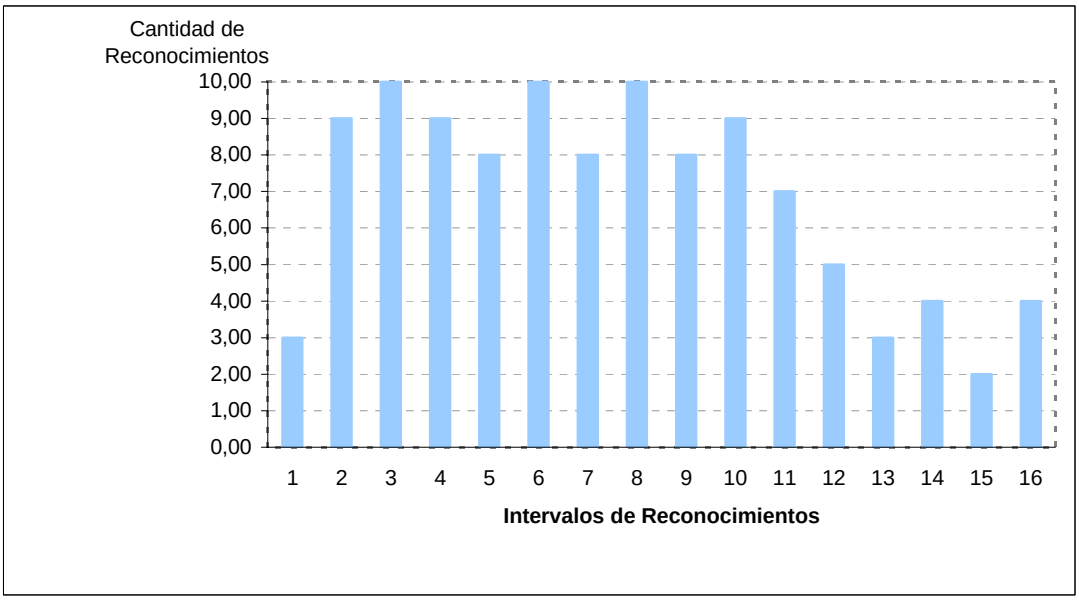

Figura No 4.7 Reconocimiento de Rostros: Cantidad de Acertados en Series de 10

Estos experimentos fueron realizados para verificar que efectivamente cuando el robot reconocía una cara con un nivel de confiabilidad, esta cara estaba bien identificada y permitieron confirman las reglas de decisión previamente definidas.

En la Figura No 4.8 se muestra como se comporta el nivel de confianza en series de reconocimientos consecutivas en las cuales la habilidad está tratando de reconocer un rostro: en estos ensayos se le ha pedido al robot que identifique a personas cuyos rostros no están almacenados en las bases de datos.

El análisis de esta serie de experimentos nos indica que la habilidad siempre tratará de asociar el rostro presentado con uno de los modelos almacenados en la base de datos: el más parecido o similar. Sin embargo será el nivel de confianza el indicador que permitirá decidir si efectivamente la cara identificada corresponde con la persona. El nivel de confiabilidad promedio para esta serie alcanza el 0,2203 y el 76% de los reconocimientos tienen un nivel de confiabilidad menor a 0,2500.

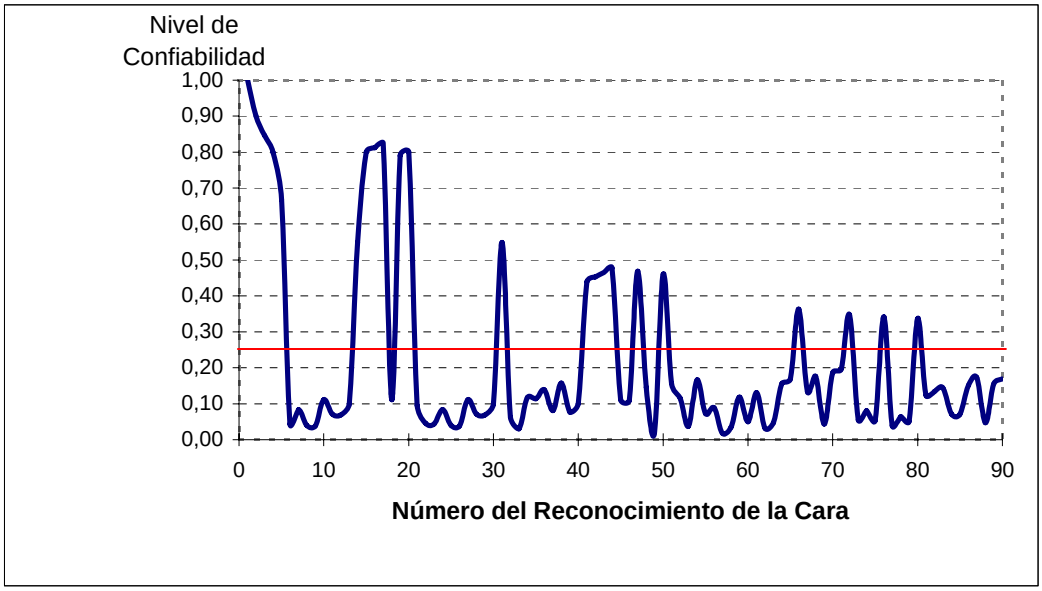

Figura No 4.8 Nivel de Confiabilidad para Reconocimiento de Rostros: Caras NO están Almacenadas en la Base de Datos

Como conclusión de los resultados obtenidos en los experimentos se propone incorporar al sistema en versiones posteriores, un módulo de toma de decisiones para la habilidad reconocer caras, de tal forma que pueda activar otras habilidades que permitan interactuar con el usuario y recolectar mayor información, cuando el nivel de confiabilidad del reconocimiento no es lo suficiente alto. También es necesario mejorar el módulo de detección de caras, a fin de evitar las fallas y retardos encontrados en los análisis experimentales.

CONCLUSIONES

Este trabajo ha sido realizado en el contexto de la Robótica, particularmente en el campo de Visión Artificial. Su aporte fundamental es el desarrollo de un sistema de reconocimiento de rostros para Maggie, robot autónomo personal diseñado y construido en el Robotics Lab de la Universidad Carlos III de Madrid, como plataforma de soporte a la investigación sobre la interacción entre humanos y robots, la inteligencia robótica y la autonomía en los robots.

El sistema de reconocimiento facial ha sido integrado a la arquitectura de control, denominada AD (Automática Deliberativa), como un conjunto de habilidades que le permiten al robot ejecutar en tiempo real y en forma autónoma las siguientes acciones: detectar los rostros de la personas en escenas visuales, tomar imágenes de los rostros detectados, construir modelos de los rostros a partir de estas imágenes, almacenar en bases de datos tanto las imágenes tomadas, como los modelos obtenidos de los rostros, y usar la información de los rostros capturados en las imágenes para reconocer personas a través de la comparación de determinados características, propiedades y rasgos de la imagen facial, con los almacenados en la base de datos.

Para la detección de los rostros en imágenes, las habilidades aplican métodos basados en los clasificadores aumentados en cascada que trabajan con características tipo haar, propuestos por Paul Viola y Michael Jones en el 2004. Para la extracción de características del rostro, construcción del modelo del rostro a ser almacenado en la bases de datos y para el reconocimiento de rostros se emplean modelos ocultos de Markov embebidos, aplicados de acuerdo al método propuesto por Aran Nefian, en 1999.

Las habilidades han sido diseñadas e implementadas en una arquitectura de tres capas, las cuales han sido denominadas: nivel habilidades, nivel de control y el nivel modelo de Markov. El nivel modelo de Markov ha sido construido usando como base el

software HMM Faces diseñado por Ou Shichao. Todas las habilidades han sido implementadas empleando la librería de visión artificial denominada OpenCV.

Los resultados observados en los diferentes ensayos y series de experimentos muestran cómo los algoritmos, aplicados en las diferentes etapas y módulos que conforman la arquitectura, son estables, no presentan complejidad espacial, tienen un comportamiento consistente y sus tiempos de ejecución son mínimos.

Maggie está explícitamente diseñada para la interacción hombre robot por lo tanto el funcionamiento de las habilidades se evalúa no solamente desde el punto de vista del resultado lógico de la ejecución de las mismas, sino que también se considera y evalúa el tiempo real que utiliza la habilidad para producir este resultado.

La habilidad para la toma de imágenes de rostros tiene una capacidad de procesamiento promedio de 1,45 imágenes por segundo. Esto significa que el robot puede tomar 124 imágenes de la cara de una persona y almacenarlas en memoria a largo plazo en 1,43 minutos. El tiempo promedio empleado por esta habilidad para tomar la imagen del rostro de la persona y almacenarla en la memoria es de 688ms, con un rango de 585ms a 6.171ms y una tasa de efectividad del 73,20%.

La habilidad para crear la bases de datos de las caras tiene el siguiente tiempo promedio de ejecución: 1) para la creación de la base de datos de imágenes de las caras 538ms, con un rango de variación de 33 a 1.473ms; 2) para la de creación de la base de datos de objetos EHMM, 89ms con un rango de 16 a 161ms; 3) para el entrenamiento de los objetos EHMM, 7.068ms con un rango de 287 a 13.852ms. Dentro de la arquitectura AD implementada hasta ahora, las habilidades de los rostros son las primeras en generar información para ser almacenada en la Memoria a Largo Plazo. El cálculo del tamaño del espacio en memoria a largo plazo ocupado por estas bases de datos es el siguiente: *(8KB + 30KB * cantidad de rostros almacenados + 1,2KB + 30KB)*cantidad de personas + 1KB.*

La habilidad para reconocer caras tiene una capacidad de procesamiento promedio de 2,30 imágenes por segundo. Esto significa que el robot puede hacer 162 reconocimientos de imágenes de una persona en 1,50 minutos. El tiempo promedio empleado por esta habilidad para reconocer a una persona a partir de la imagen del rostro es de 434ms, con un mínimo de 352ms y un máximo de 1.235ms. A partir de los

resultados obtenidos experimentalmente se pueden definir las siguientes reglas de decisión: el robot habrá identificado a la persona cuando el nivel de confianza sea superior al 70%. El robot no habrá identificado a la persona cuando el nivel de confianza sea inferior al 25%. Cuando el nivel de confianza se encuentra entre el 70% y 50%, el robot habrá identificado a la persona con una probabilidad de error del 0,20. Cuando el nivel de confianza se encuentra entre el 50% y 25%, es necesario suministrarle al robot mayor cantidad de información, para confirmar la identificación

Como conclusión de los resultados obtenidos en los experimentos se proponen las siguientes mejoras para que sean incluidas en versiones posteriores de las habilidades de reconocimiento de rostros:

- Diseñar un módulo de toma de decisiones para la habilidad reconocer caras, de tal forma que pueda activar otras habilidades para interactuar y recolectar mayor información del usuario, cuando el nivel de confiabilidad del reconocimiento no es lo suficiente alto.

- Optimizar el módulo de detección de caras, a fin de evitar las fallas y retardos encontrados en los análisis experimentales.

- Analizar y evaluar el comportamiento del sistema en condiciones de iluminación no controlados a objeto de efectuar el ajuste correspondiente en la programación.

Extensiones y Trabajo Futuro

Algunos de los aspectos surgidos en el curso de la investigación realizada y que emergen como posibles objetivos en el desarrollo de trabajos de investigación posteriores, son descritos a continuación:

- Las habilidades fueron desarrolladas tomando en consideración: 1) las características del hardware y software de Maggie; 2) la eficiencia de los métodos y algoritmos usados en el reconocimiento de los rostros; 3) la restricciones impuestas por

necesidad de operación y tiempos de respuestas en tiempo real; 4) las limitaciones causadas por el plazo de tiempo estipulado para su implementación.

Se plantea, entonces como una futura extensión de este trabajo, la comparación de los modelos, métodos y algoritmos empleados y de los resultados obtenidos con otros paradigmas empleados en el reconocimiento de rostros, tales como redes neuronales artificiales y máquinas de soporte vectorial.

- Los rostros humanos son superficies que pueden ser trazadas en espacios tridimensionales. Diferentes autores han propuesto modelos 3D para representar caras, métodos basados en modelos 3D para codificar forma y textura y algoritmos para recuperar estos parámetros desde la imagen de una cara o rostro. Además, el modelo 3D debería permitir representaciones para las caras más extensas y detalladas, especialmente para manejar variaciones faciales, tales como, pose, iluminación, entre otros.

En consecuencia, resulta completamente plausible proponer como objetivo de posibles investigaciones la aplicación de estos modelos, métodos y algoritmos en el diseño de sistemas de reconocimiento visual que puedan ser implementados en robots autónomos, como Maggie.

- Estudios realizados en el área de la neurociencia han arrojado evidencia acerca de cómo el proceso de reconocimiento de los humanos utiliza un amplio espectro de estímulos obtenidos de diferentes sentidos, tales como visual, auditivo, olfativo, táctil, entre otros. En muchas situaciones, hasta información contextual puede ser aplicada.

Bajo estas consideraciones resulta completamente factible plantear como posible objetivo de investigaciones futuras el desarrollo de sistemas biométricos para Maggie, que le permitan la identificación o el reconocimiento único de personas a partir de más un rasgo físico o de una o más características conductuales.

REFERENCIAS BIBLIOGRÁFICAS

[1] Armengot, M.; Ferri, F. (2006). *Análisis Comparativos de Métodos Basados en Subespacios Aplicados al Reconocimiento de Caras*. Reporte Técnico. Universidad de Valencia. Valencia, España.

[2] Barber, R. (2000). *Desarrollo de una Arquitectura para Robots Móviles Autónomos, Aplicación a un Sistema de Navegación Topológica.* Tesis Doctoral. Universidad Carlos III de Madrid. Leganés, España. Tutor: Salichs, M. A.

[3] Bartlett, M.; Hager, J.; Ekman, P.; Sejnowski, T. (1999). Measuring Facial Expressions by Computer Image Analysis. *Psychophysiology*, 36, 253-263

[4] Barret, W. (1998). A Survey of Face Recognition Algorithms and Testing Results. *Systems and Computers*, 1, 301-305.

[5] Black, P. (2008). Hidden Markov Model. En P. Black (Eds). Dictionary of Algorithms and Data Structures, *U.S. National Institute of Standards and Technology.* (Revisado en Septiembre 2008). Disponible en: http://www.nist.gov/dads/HTML/hiddenMarkovModel.html

[6] Bledsoe, W. (1964). The Model Method in Facial Recognition. *Technical Report PRI 15, Panoramic Research, Inc.*, Palo Alto, CA.

[7] Brunelli, R.; Poggio, T. (1993). Face Recognition: Features versus Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (10), 1042-1052.

[8] Castro, A. (2008). *Desde la Teleoperación hasta el Control por Tacto del Robot Maggie*. Tesis de Maestría. Universidad Carlos III de Madrid. Leganés, España. Tutor: Salichs, M. A.

[9] Corrales, A. (2008). *Sistema de Identificación de Objetos Mediante RFID para un Robot Personal*. Tesis de Maestría. Universidad Carlos III de Madrid. Leganés, España. Tutor: Salichs, M. A.

[10] Gonzalez, R.; Woods, R. (2008). *Digital Image Processing.* Upper Saddle River, NJ: Pearson, Prentice Hall.

[11] Face Recognition Homepage. (2008). (Revisado en Septiembre 2008). Disponible en: http://www.face-rec.org/

[12] Fogelman Soulie, F.; Viennet, E.; Lamy, B. (1993). Multi Modular Neural Network Architectures: Applications in optical Character and Human Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 721.

[13] Harmon, L. D. (1973). The Recognition of Faces. *Scientific American*, 229(5):71-82

[14] HMM Face Recognition. (2006). Laboratory for Perceptual Robotics, University of Massachusetts Amherst. (Revisado Septiembre 2008) Disponible en: http://www-robotics.cs.umass.edu/Documentation/FaceRecognition#HMM

[15] Kanade, T. (1973) *Picture Processing System by Computer Complex and Recognition of Human Faces*. Doctoral Dissertation. Kyoto University, Japan.

[16] Kang, S.; Young, K.; Park, R. (1998). Hybrid Approaches to Frontal View Face Recognition Using the Hidden Markov Model and Neural Network. *Pattern Recognition*, 31(3), 283-293.

[17] Kohonen, T. (1989). *Self Organization and Associative Memory*. Berlín: Springer Verlag.

[18] Lienhart, R.; Liang, L.; Kuranov, A. (2003). A Detector Tree of Boosted Classifiers for Real-time Object Detection and Tracking. *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo*, 1. (pp. 277-280). Washington, DC: IEEE Computer Society.

[19] Linux UVC Driver and Tools. Berlios. (Revisado en Septiembre 2008). Disponible en http://linux-uvc.berlios.de/

[20] Low, B.; Hjelmas, E. (2001). Face Detection: A Survey. Computer Vision and Image Understanding, 83(3), 236-274.

[21] Moghaddam, B.; Nastar, C.; Pentland, A. (1996). Bayesian Face Recognition Using Deformable Intensity Surfaces. *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, (pp. 638-645). Viena, Austria.

[22] Moisés, P. (2007). *Aportaciones al Reconocimiento Automático de Texto Manuscrito*. Tesis Doctoral. Universidad de Politécnica de Valencia. Valencia, España. Tutor: Vidal, E.

[23] Nefian, A. (1999) *A Hidden Markov Model Based Approach for Face Detection and Recognition.* Thesis of Doctor of Philosophy in Electrical Engineering. Georgia Institute of Technology. Atlanta, USA.

[24] Nefian, A.; Hayes, M. (1999). An embedded HMM based approach for face detection and recognition. *Proceedings of the 1999 IEEE International Conference on Acoustics Speech and Signal Processing,* 6, 3553-3556.

[25] Nefian, A.; Hayes, M. (2000). Maximum Likelihood Training of the Embedded HMM for Face Detection and Recognition. *Proceedings of the 2000 IEEE International Conference on Image Processing*. Vancouver, Canada, 1, 10-13.

[26] OpenCv: Experimental Functionality Reference. Intel Open Source Computer Vision Library. (Revisado en Septiembre 2008) Disponible en http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_Experimental.htm#facedetect_paper1

[27] Ou, S. Chaos Future: Latest in Robotics and Assistive Technology. (Revisado en Septiembre 2008). Disponible en http://www.chaosfuture.com/

[28] Phillips, P. J. (1998). Support Vector Machines Applied to Face Recognition. En Kearns, M.; Solla S.; Cohn, D. (Eds). *Advances in Neural Information Processing Systems II,* (pp. 803–809). Cambridge, MA: MIT Press.

[29] Rivas, R.; Corrales, A.; Barber, R; Salichs, M.A. (2007). Robot Skill Abstraction for AD Architecture. *6th IFAC Symposium on Intelligent Autonomous Vehicles*. Toulouse, France.

[30] Rowley, H.; Baluja, S.; Kanade, T. (1998). Neural Network Based Face Detection. *IEEE Transactions On Pattern Analysis and Machine intelligence*, 20(1), 23-38.

[31] Sakai, T.; Nagao, T.; Kanade, T. (1972). Computer Analysis and Classification of Photographs of Human Faces. *USA-Japan Computer Conference.* October 1972, (pp. 55-62). Tokyo, Japan.

[32] Salichs, M. A.; Barber, R.; Khamis, A.; Malfaz, M.; Gorostiza, J.; Pacheco, R.; Rivas, R.; Corrales, A.; Delgado, E.; García, D. (2006). Maggie: A Robotic Platform for Human-Robot Social Interaction. *IEEE International Conference on Robotics, Automation and Mechatronics*. Bangkok, Thailand.

[33] Samaria, F.; Young, S. (1994) HMM Based Architecture for Face Identification. *Image and Vision Computing*, 12(8), 537-543.

[34] Sung, K.; Poggio, T. (1997). Example Based Learning for View Based Human Face Detection. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 20,(1), 39–51.

[35] Turk, M.; Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1), 71-86.

[36] Valentín, D.; Abdi, H.; O´toole, A.; Cottrell, G. (1991). Connectionist Models of Face Processing: a Survey. *Pattern Recognition*, 27, 1209-1230.

[37] Viola, P.; Jones, M. (2004). Robust Real Time Face Detection. *International Journal of Computer Vision*, 57(2), 137–154.

[38] Yang, M-H. (2008). Face Detection. To appear in *Encyclopedia of Biometrics*. (Revisado en Septiembre 2008) Disponible en http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html

[39] Zhao, W.; Chellapa, R.; Rosenfeld, A.; Phillips, P. (2003). Face Recognition: A Literature Survey. *ACM Computing Survey CSUR*, 35(4), 399-458.