UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS

RECONOCIMIENTO FACIAL
USANDO EL ANÁLISIS DE LOS COMPONENTES PRINCIPALES

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO DE SISTEMAS

RUIZ ADARMES, MIGUEL ANGEL
Y
VILCAPUMA DE LA CRUZ, JHON PUBLIO

LIMA PERU
2002

INDICE

DESCRIPTORES TEMATICOS

1 Reconocimiento Facial.

2 Segmentación usando la Transformada de Hough.

3 Extracción usando el Análisis de los Componentes Principales.

4 Toma de Decisión para : Identificación, Reconocimiento y Clasificación.

5 Uso de algoritmos *LVQ* y del *Error de Reconstrucción* para el problema de Clasificación de Sexo.

RESUMEN

Existen dos tipos de aplicaciones que pueden beneficiarse de las técnicas de recnocimiento facial, las cuales son:

- Aplicaciones con input de imágenes estáticas
- Aplicaciones con input de imágenes dinámicas en tiempo real.

Historia de la Transformada de Karhunen-Loève

El núcleo fundamental del Análisis de Componentes Principales (ACP) ó transformada de Karhunen-Loève, y en general del Análisis Factorial, es el problema de la obtención de los vectores y valores propios de un operador vectorial, que en el campo del cálculo matricial se da bajo el problema de la diagonalización de una matriz cuadrada.

Para el problema de análisis de datos, la transformada de Karhunen-Loève es el método lineal óptimo, debido al mínimo error cuadrado medio, para reducir la redundancia en el conjunto de datos (Fukunaga, [28]).

Reconocimiento usando el Espacio Eigen

La idea básica detrás del método del espacio eigen es la de representar un gran número de imágenes para “entrenamiento" en un subespacio de bajas dimensiones. Cuando una imagen nueva es vista, puede ser clasificada como muy similar o muy diferente de las imágenes de entrenamiento con sólo unas cuantas operaciones en el subespacio calculado con anterioridad. Una vez que estas imágenes de entrenamiento son separadas en sus componentes usando la transformada de Karhunen-Loève, cualquier imagen nueva puede ser clasificada.

Turk y Pentland [33], [34] consideran la metodología para tratar el reconocimiento facial como un problema de reconocimiento en dos dimensiones, debido al hecho de que los rostros están normalmente en un plano vertical y de esta manera pueden ser descritas por un pequeño conjunto de características visibles 2-D.

Definición General del Problema de Reconocimiento Facial

Una definición general del problema puede formularse como sigue:

Dada una imagen o el video de una escena, identificar uno o más personas dentro de la imagen o la escena usando una base de datos de imágenes almacenadas.

La solución del problema general puede dividirse en tres etapas diferentes:

1. Segmentación de rostros dentro de un conjunto de rostros.

2. Extracción de las características de una región de la cara.
3. Toma de Decisión.

La Segmentación: Es usualmente lograda por la construcción de un mapa de los entornos, luego se concectan las orillas usando algoritmos heurísticos, y las orillas son acomodadas dentro de figuras elípticas usando la *Transformada de Hough*.

La Extracción: Es la etapa crítica. Existen dos tipos de rasgos: Holísticos, donde cada rasgo es una característica global de la cara y Parciales (pelo, nariz, boca, ojos, etc.). Las técnicas de rasgos parciales toman algunas medidas dentro de muchos puntos cruciales de la cara, mientras que las técnicas de rasgos holísticos manejan las características de la cara como un todo. ACP es una técnica de rasgos holísticos.

La Toma de Decisión: Se hace partiendo de la información obtenida en la etapa previa. Tres tipos de decisión pueden ser tomadas dependiendo de la aplicación:

1. **Identificación**: En el cual los datos de una persona en particular tienen que ser obtenidas (1:N).
2. **Reconocimiento**: De una persona, el cuál puede ser ejecutado, si es que esta persona en particular antes ya ha sido vista o enrolada. Este concepto esta asociado con el proceso de Reconstrucción de un rostro a partir de una imagen parcial.
3. **Toma de Decisión**: En el cuál el rostro tiene que ser asignado a cierta categoría.

Resultados Obtenidos por Experimentación

Usando Imágenes Estáticas: Se analizo el comportamiento del ACP cuando es enfrentado a los dos mayores problemas del reconocimiento facial – identificación y clasificación. La técnica *Jack-Knife* se uso para construir los conjuntos de entrenamiento y prueba, dado que los conjuntos originales de entrenamiento y prueba de la base de datos facial no son apropiados para el ACP.

La prueba de identificación usa la regla de la distancia mínima simple desde un vector almacenado, y los resultados observados son buenos (91%).

La clasificación se probó usando dos algoritmos. El método LVQ produce resultados pobres (81%), mientras que el método basado en el error de reconstrucción produce resultados muy buenos (98%) la cuál también se generaliza para individuos desconocidos (88%) a pesar de que la base de datos facial no es una buena base de datos para el ACP.

Usando Imágenes Dinámicas: Una vez analizado y demostrado la ventaja del ACP para el reconocimiento facial sobre imágenes estáticas, se realizo un nuevo experimento usando imágenes de rostros obtenidos en forma dinámica (en vivo). Los resultados generales son aceptables obteniendo porcentajes de certeza superiores al 93.11%. Como el ACP es un proceso 2D, el cuál toma las características faciales como un todo, se puede decir que el porcentaje mayor de identificación se obtiene cuando se presenta al proceso de identificación un

individuo bajo las mismas condiciones de luminosidad y de fondo con las que fue entrenado el sistema. Por lo tanto, si bien las conclusiones estadísticas son concluyentes, se puede decir que dependen grandemente de los algoritmos de ingreso de imágenes faciales hacia el sistema. Como se dijo se hace evidente una cierta normalización de las imágenes faciales con ayuda de un operador. El límite establecido de 93.11% de exactitud al identificar individuos se estableció en forma experimental ya que si bien límites inferiores presumen una identificación correcta, fueron observados errores en el método con seguridad motivados por el fondo o condiciones de luminosidad distintas a las imágenes faciales usadas para el entrenamiento.

Conclusiones

EL Análisis de lo Componentes Principales hace uso extensivo de gran cantidad de productos de matrices grandes, lo cuál representa gran tiempo de calculos de computadora. Esto explica porque el proceso de Reconocimiento Facial esta aún muy lejos de realizarse en tiempo real para grandes poblaciones.

CAPITULO I

INTRODUCCION

Este capítulo introductorio trata de las diferentes aplicaciones, sus ventajas y desventajas que podrían beneficiarse mediante el uso de las técnicas de reconocimiento facial. Se presentarán además las diferentes fases que son realizadas por un sistema de reconocimiento facial.

1.1 Aplicaciones, Ventajas y Desventajas

Las *Aplicaciones de Técnicas de Reconocimiento Facial* (TRF) se pueden subdividir en varias dentro de comerciales y legales. Una lista detallando algunas aplicaciones posibles con sus ventajas y desventajas se muestra en la tabla 1.1.

	Aplicaciones	Ventajas	Desventajas
1	Tarjetas de Crédito. Documentos de Identidad.	Control de imágenes. Segmentación controlada. Imágenes de buena calidad.	Necesidad de DB potentes. Algoritmos búsqueda compleja.
2	Búsqueda en conglomerados de imágenes.	Imágenes de calidad variada. Más de una imagen disponible.	Necesidad de DB potentes. Algoritmos búsqueda compleja
3	Seguridad en Bancos y Comercios.	Búsqueda geográficamente localizada.	Segmentación no controlada. Baja calidad de imagen.
4	Vigilancia de multitudes.	Tamaño de archivo pequeño. Disponibilidad de imágenes de video.	Segmentación no controlada. Baja calidad de imagen. Control en tiempo real.
5	Identificación policial.	Se posibilita el aumento de la precisión en la identificación.	Baja calidad de imagen. Certificación legal requerida.
6	Reconstrucción Facial por testigos.	Identificación rápida de individuos por testigos.	Similitudes desconocidas.

Estas aplicaciones se pueden clasificar en dos grandes grupos:

- Aplicaciones con input de imágenes estáticas
- Aplicaciones con input de imágenes dinámicas en tiempo real.

Entre estos dos grupos, existen diferencias significantes dependiendo de la aplicación específica. Las diferencias se encuentran en: la calidad de las imágenes, tamaño de imágenes del fondo y la naturaleza, tipo y cantidad de datos de una persona (puntos 5 y 6).

Los puntos del 1 al 4 involucran el matching de una imagen con otra imagen del rostro.

Los puntos del 5 y 6 involucran la búsqueda y la reconstrucción de una imagen del rostro, a partir de datos, el cual es similar a los datos que se tienen de un rostro humano de origen.

Cada una de estas aplicaciones necesitan diferentes requerimientos para el proceso de reconocimiento. El matching requiere que la imagen del rostro origen esté en algunos de los conjuntos de imágenes de rostros seleccionadas por el sistema.

Operaciones similares requieren, adicionalmente al matching, que las imágenes de los rostros pertenezcan a un grupo similar de ciertas características comunes y en donde necesariamente debe de existir el rostro de origen (parametrización); esto requiere que las clasificaciones usadas por el sistema de reconocimiento usen las mismas que los humanos.

Este es el caso de imágenes que podrían provenir de la foto de un pasaporte (aplicación 1).

Las aplicaciones listadas arriba son algunos ejemplos de problemas existentes que pueden resolverse con la ayuda de la ingeniería. Aunque, los ingenieros no son los únicos quienes investigan sobre soluciones que usen la TRF. En efecto, los psicólogos y los neuro científicos también conducen investigaciones dentro de este campo. Su propósito es entender cómo el cerebro humano discrimina e identifica las imágenes. Una de estas técnicas importantes, conocida como el *Análisi de los Componentes Principales* (ACP), también conocida como la *Transformada de Karhunen-Loeve* (KL), es la parte principal de este estudio.

1.2 Historia de KL:

El núcleo fundamental del Análisis de Componentes Principales (ACP) ó procedimiento Karhunen-Loève, y en general del Análisis Factorial, es el problema de la obtención de los vectores y valores propios (principales) de un operador vectorial, que en el campo del cálculo matricial se da bajo el problema de la diagonalización de una matriz cuadrada. Este problema algebraico, que inicialmente impulsó el desarrollo del Análisis Factorial en el estudio de la regresión lineal entre múltiples variables en los trabajos que Pearson (1901, 1904) realizó en aplicaciones biológicas y psicométricas, se ha convertido, a lo largo de nuestro siglo, en el uno de los instrumentos más extendidos en todas las ramas científicas. No sólo es una técnica de análisis empírico de la varianza, sino que puede jugar un papel decisivo en la formulación teórica, tal y

como lo demuestra su papel protagonista en la formulación de la teoría de la Mecánica Cuántica moderna.

El procedimiento Karhunen-Loève fue propuesto independientemente por Karhunen [1] (1946), Loève [2] (1955). Método conocido bajo una variedad de diferentes nombres en campos diferentes:

- Análisis de los Componentes Principales ó Análisis Hotelling (Hotelling, 1953, [4]).
- Análisis del Componente Empírico (Lorenz, 1956, [3]).
- Modos Quasi Harmónicos (Brooks el at., 1988, [5]).
- Descomposición Ortogonal Propia (Lumley, 1967, [6]).
- Descomposición del Valor Singular (Golub y Van Loan, 1983, [7]).
- Descomposición Empírica de la Función Eigen (Sirovich, 1987, [10]).
- Etc.

Muy cercana con esta técnica es el *análisis del factor*, la cuál es usada en sicología y economía (Harman, 1960, [8]).

Desde el punto de vista matemático, la transformada KL [2] no tiene mayor significado, la cuál es una transformación que diagonaliza una matriz R dada y proporciona una forma canónica R=ULV, donde L es una matriz diagonal. Por consiguiente los principios de KL se encuentran por la mitad del siglo ante pasado. Un repaso de la historia más cercana de la transformada KL se puede encontrar en [16]. El contenido matemático del procedimiento KL es por consiguiente clásico y está contenido en un paper de Schmidt [9] (1907).

Debido a la gran cantidad de cálculos requeridos para encontrar los vectores eigen, la técnica KL fue virtualmente no usada hasta la mitad de la centuria pasada. Cambios radicales ocurrieron con la aparición de poderosos computadores y el desarrollo de algoritmos eficientes para calcular las *funciones eigen*[1] (*método de snapshots*[2], [10]). Ahora la transformada KL es usada extensivamente en los campos de detección, estimación, reconocimiento de patrones, y procesamiento de imágenes como una herramienta eficiente para almacenar procesos randon, en sistemas de control.

La transformada KL, conocida también como la Descomposición Ortogonal Propia (DOP ó POD por sus siglas en inglés) es usada en conexión con los problemas estocásticos turbulentos (Lumley, 1967, [6]). En aquel contexto, las *funciones eigen*[3] asociadas pueden ser identificadas con los *remolinos característicos* del campo turbulento. También la transformada KL se encuentra inmersa en algunos procesos biológicos, como las configuraciones de algunos estímulos particulares definidos por la forma, posición y orientación de segmentos de bordes entre áreas de diferente brillo (los cuáles son llamados los segmentos *borde*[4]) que son la fuente para una respuesta efectiva en las células. Los segmentos de tipo borde están entre las primeras características extraídas en la corteza visual primaria (Hubel & Wiesel [29]).

[1] Anexo 7
[2] El Método de *Snapshots*, es conocido también como el *Análisis de los Componentes Principales (ACP)*.
[3] Anexo 7
[4] *Borde* (edge) definido como una conexión entre dos vértices de un gráfico.

1.2.1 Análisis de Señales y Reconocimiento de Patrones

Para el problema de análisis de datos, la transformación KL es el método lineal óptimo (debido al mínimo error cuadrado medio) para reducir la redundancia en el conjunto de datos (Fukunaga, [28]).

La idea básica detrás de la transformada KL es:

"Transformar posibles variables correlacionadas en un conjunto de datos dentro de un mínimo número de variables no correlacionadas".

1.2.1.1 Reconocimiento usando el Espacio Eigen

La idea básica detrás del método del espacio eigen es la de representar un gran número de imágenes para "entrenamiento" en un subespacio de bajas dimensiones. Cuando una imagen nueva es vista, puede ser clasificada como muy similar o muy diferente de las imágenes de entrenamiento con sólo unas cuantas operaciones en el subespacio calculado con anterioridad. Una vez que estas imágenes de entrenamiento son separadas en sus componentes usando la transformada de KL, cualquier imagen nueva puede ser clasificada.

Una de las primeras aplicaciones de la transformada KL para reconocimiento de patrones fue hecha por Watanabe en 1965 [25]. Lo aplico para *reconocimiento del hablante*.

La metodología general de la representación KL sobre cualquier gran base de datos se discute en [24]. Lo óptimo del método KL se ilustra con una gran variedad de ejemplos (reconocimiento de patrones, flujo de turbulencias, sicología y flujo oceanográfico).

Turk y Pentland [33], [34] consideran la metodología para tratar el reconocimiento facial como un problema de reconocimiento en dos dimensiones, tomando ventaja del hecho de que los rostros están normalmente en un plano vertical y de esta manera pueden ser descritas por un pequeño conjunto de características visibles 2-D. Las imágenes del rostro fueron proyectadas obre un espacio característico ("espacio facial") que codifican mejor las variacione entre imágenes de rostro conocidas. El espacio facial fue definido luego por los eigenfaces, los cuáles son los eigenvectors del conjunto de rostros; los cuales no necesariamente correspondían a características aisladas tales como ojos, oídos y narices. Este entorno de trabajo proveyó la habilidad de aprender a reconocer nuevos rostros de una manera automática.

La Identificación del contenido de formas de data digital puede ser hecha de un modo más confiable, pero el gran volumen de datos disponible en la actualidad hace necesario el uso de técnicas automáticas. Muy pocas técnicas han sido aplicadas al problema de extraer y almacenar información tomada de video clips. Una nueva técnica para manejar información relativa en video clips, basada sobre la transformada KL fue propuesta por Griffioen et al [35]. La idea clave para la eficiencia de la metodología es el explotar el hecho de que los videos clips son almacenados digitalmente en un formato comprimido, transformado.

En el campo del *reconocimiento de objetos*, Murase y Nayar [11] direccionaron el problema de modelos de objetos de aprendizaje automático para reconocimiento y estimación de la posición. En contraste con la técnica tradicional, el problema de reconocimiento fue reformulado como uno de correspondencia con la apariencia en vez de la forma. Para cada objeto de interés, se obtuvo un gran conjunto de imágenes mediante la variación

automática de la posición e iluminación. Este conjunto de imágenes se comprimió para obtener un subespacio dimensional con bajas dimensiones, llamado *eigenspace*[5] (espacio eigen), en el cuál el objeto fue representado como un colector. Dada una imagen desconocida como entrada, el sistema de reconocimiento proyecta la imagen en el *eigenspace*. El objeto es reconocido basado sobre el colector en el que cae. La posición exacta de la proyección sobre el colector determina la posición del objeto en la imagen. Se ha desarrollado un sistema de reconocimiento en tiempo real con 20 objetos complejos en la base de datos. El *paper* ha sido concluido con una discusión sobre los varios temas relativos de la metodología propuesta de aprendizaje y reconocimiento.

Gorecki [26] aplico la transformada KL para clasificar superficies rugosas analizando el número reducido de coeficientes de Fourier de la superficie de la imagen. También los métodos KL reducen la dimensionalidad de la data de muestra y comprime la data de input.

1.2.2 Comprensión de Datos

La transformada KL es reconocida como una de las herramientas más efectivas para compresión de información.

Uno de los primero usos de la transformada KL en codificación y compresión de imágenes fue desarrollado en [17].

[5] Si A es una matriz cuadrada de $n \times n$ y λ es un eigenvalue de A, luego la reunión (U) con el vector cero O y el conjunto de todos los $eigenvectors$ correspondientes a los $eigenvalues$ λ es un subespacio de R^n conocido como $eigenspace$ de λ.

Abbas y Fahmy ([20]) aplicaron las redes neuronales artificiales, basados sobre el procedimiento KL, en el área del procesamiento de señales digitales y en la compresión de datos para los sistemas de codificación de imágenes. La implementación de redes neuronales eliminan muchos problemas relativos a la transformada KL como los grandes cálculos. La red neuronal fue entrenada usando sólo parte de la imagen mientras que el resto de la imagen fue codificada con el mismo conjunto de *eigenvectors*. El algoritmo mostró un buen rendimiento relativo a la generalización de las capacidades del modelo cuando nuevas imágenes han sido representadas por los *eigenvectors* producidos usando otras imágenes. Yamashita y Ogawa [13] aplicaron satisfactoriamente una técnica KL modificada para compresión de datos en presencia de ruido.

Hilay y Rubinstein [19] desarrollaron un algoritmo de comprensión basado sobre el procedimiento KL, el cuál es invariante bajo ciertas transformaciones (rotación en dos dimensiones). Permite reconocer imágenes rotadas comprimidas.

El método KL fue también usado en compresión de data del habla (Chen, Huo [22]). La señal del habla fue descompuesta usando la *Transformada Fourier-Bessel*[6] con la transformada KL de los coeficientes obtenidos. Esto permite reducir la velocidad de la data del habla sintetizada a 11 Kbit/seg con el error cuadrado medio menor al 3%.

[6] Anexo 13.

1.2.3 Sistemas de Control

El hecho de que la transformación KL sea tan buena motivo su uso en el área de sistemas de control. En esta área la transformada KL trabaja principalmente como una técnica de reducción de data, permitiendo analizar rápidamente y más eficientemente la data experimental. Los resultados del análisis son generalmente un parámetro de input para el lazo de control.

Una de las primeras aplicaciones de KL en sistemas de control fue presentada en [31] para la identificación de *Sistemas de Parámetros Distribuidos*[7] lineales (DPS por sus siglas en inglés) con sólo información limitada a priori acerca del proceso físico. Información básica a partir de inputs y outputs del sistema fue usada para identificar aproximadamente la característica espacial del DPS. El procedimiento de identificación resulta en un seudo modelo de input/output para el sistema. Aproximaciones para las *funciones eigen*[8] fueron determinadas numéricamente a partir de data de input/output usando el *Método de Galerkin*[9] y la descomposición KL. Se demostró la aplicación de este procedimiento para DPS descrita tanto por un modelo parabólico auto adjunto y no-auto adjunto (modelo parabólico *self-adjoint* y *non-self-adjoint*)[10]. Se mostró la utilidad del modelo identificado para el diseño de sistema de control usado como ejemplo.

[7] La teoría clásica de sistemas de control fue desarrollada para sistemas gobernados por *ecuaciones diferenciales ordinarias* (ODE) de dimensión finita. Cuando las ecuaciones que gobiernan el sistema son ecuaciones diferenciales parciales (PDE) donde el estado del sistema permanece en alguna función espacial de dimensión finita; a este sistema se le conoce también como Sistema de Parámetros Distribuidos, en donde los inputs de control y estado están distribuidos espacialmente (Lions (1971)). Brian G. Allan, en *"Optimal Control Using a Reduced Order Model of the Incompresible Navier – Stockes Equations"*, 3rd ASME/JSME Joint Fluids Engineering Conference, Julio 1999

[8] Anexo 7

[9] Anexo 4

[10] Anexo 5

Krischer et al [27] usó KL para analizar las inestabilidades en los sistemas reactivos químicos. Aquí KL permitió la identificación de estructuras espaciales coherentes a partir de la data experimental y la determinación de los grados de libertad activos (estructuras espaciales importantes). Proyección sobre estos "modos" redujo la data a un pequeño número de series de tiempo. Continuando estas series de tiempo fueron procesados por una Red Neural Artificial para obtener un modelo dinámico no lineal de baja dimensionalidad. Predijo correctamente el comportamiento espacio temporal en el corto plazo y la dinámica del sistema en el largo plazo (el atractor[11], attractor en inglés). El modelo basado en KL demostró ser una buena herramienta de procesamiento post data, el cuál puede liderar los modelos de input/output de baja dimensionalidad.

1.2.4 Resumen

La capacidad de la transformada KL, para encontrar un mínimo número de parámetros independientes con el objetivo de describir un conjunto o base de datos estadísticos, es ampliamente usada en varios campos teóricos y de aplicación de la ciencia. La aparición de computadoras más poderosas y más rápidas ayudo en la solución de uno de los principales problemas para las aplicaciones KL, a saber el costo computacional de resolver ecuaciones matriciales muy grandes. Para algunos casos importantes se han desarrollado algoritmos eficientes como el método de la toma instantánea.

[11] Attractor se define como un conjunto de estados, invariantes bajo dinámicas; cuyos estados de sus límites en un dado, conjunto de puntos en el espacio de variables del sistema tales que las condiciones iniciales seleccionadas en este conjunto dinámicamente lo envuelven, asintóticamente se aproximan en el curso de la evolución dinámica. (http://mathworld.wolfram.com/Attractor.html)

La optimalidad del método KL permite reducir la cantidad de información acerca de la señal o proceso en un razonable número de *funciones eigen*[12] independientes, quienes representan las más importantes características de la señal. Por consiguiente, KL es una herramienta muy adecuada para el posterior análisis de los procesos.

1.3 Tres Etapas

Una definición general del problema puede formularse como sigue:

Dada una imagen o el video de una escena, identificar uno o más personas dentro de la imagen o la escena usando una base de datos de imágenes almacenadas.

La solución del problema general puede dividirse en tres etapas diferentes:

1. Segmentación de rostros dentro de un conjunto de rostros.
2. Extracción de las características de una región de la cara.
3. Toma de Decisión.

La Segmentación: Es usualmente lograda por el siguiente algoritmo: Se construye un mapa de los entornos, luego las orillas son conectadas usando varios algoritmos heurísticos, y las orillas son acomodadas dentro de figuras elípticas usando la *Transformada de Hough*. Se debe de hacer notar que si el input de datos esta compuesto por imágenes de video, se puede lograr la segmentación usando el movimiento como una variable.

La Extracción: La etapa crítica es la extracción de los rasgos. Hay esencialmente dos tipos de rasgos: rasgos holísticos (donde cada rasgo es una

[12] Anexo 7

característica global de la cara) y rasgos parciales (pelo, nariz, boca, ojos, etc.). Las técnicas de rasgos parciales toman algunas medidas dentro de muchos puntos cruciales de la cara, mientras que las técnicas de rasgos holísticos maneja las características de la cara como un todo. ACP es una técnica de rasgos holísticos.

La Toma de Decisión: Se hace partiendo de la información obtenida en la etapa previa. Tres tipos de decisión pueden ser tomadas dependiendo de la aplicación:

1. **Identificación**: En el cual los datos de una persona en particular tienen que ser obtenidas (1:N).
2. **Reconocimiento**: De una persona, el cuál puede ser ejecutado, si es que esta persona en particular antes ya ha sido vista o enrolada. Este concepto esta asociado con el proceso de Reconstrucción de un rostro a partir de una imagen parcial.
3. **Clasificación**: En el cuál el rostro tiene que ser asignado a cierta categoría.

El segundo capítulo esta referido a la *Segmentación* de rostros, detallándose para ello los fundamentos teóricos de la *Transformada de Hough*.

El tercer capítulo, la parte central de este documento, detalla la teoría del ACP usada para la *Extracción*.

En el cuarto capítulo se exponen los resultados experimentales obtenidos usando el ACP.

En el quinto capítulo se expone la teoría usada para la *Toma de Decisión* con respecto al problema de Clasificación de Sexo, explicándose los fundamentos teóricos de los algoritmos *LVQ* y del *Error de Reconstrucción*.

En el sexto capítulo se describe brevemente algunos aspectos sobre el programa desarrollado para este proyecto.

Los Anexos contienen el *Manual del Usuario* y notas aclaratorias sobre conceptos importantes usados en este documento.

CAPITULO II

SEGMENTACIÓN USANDO LA TRANSFORMADA DE HOUGH

2.1 Motivación

Si disponemos de la forma de las regiones o de los objetos de una imagen, la segmentación se puede considerar como el problema de localizar dichos objetos. Ejemplos clásicos son la búsqueda en aplicaciones industriales de objetos circulares, o la búsqueda de objetos de formas específicas en imágenes de satélite.

La clave del éxito en estos métodos es la unión de la información de los puntos de discontinuidad de la imagen con el conocimiento a priori que se tiene sobre la forma a localizar.

El hablar de la localización de objetos puede indicar que nuestro objetivo será la localización de una curva cerrada que delimita la frontera de una región en la imagen. Sin embargo, nuestro objetivo será más general ya que intentaremos

extraer aquellas curvas, de forma conocida, que determinan la separación entre objetos.

Un método muy eficaz para resolver dicho problema es la *Transformada de Hough*, que se comporta con una gran robustez incluso sobre imágenes con graves problemas de solapamiento y ruido.

2.1.1 Problema

El problema que queremos resolver consiste en diseñar un algoritmo para la localización de curvas en una imagen. Para ello, disponemos de:

- Una imagen I de niveles de gris y de dimensiones NxN. A partir de ella, podemos aplicar un algoritmo de extracción de discontinuidades dando lugar a un conjunto de m puntos frontera

  $$P_I = \{(x,y) / (x,y) \text{ frontera en } I\} \quad (2.1)$$

  Este conjunto de puntos, que será el que usemos como entrada al algoritmo, nos informa sobre los lugares más probables por donde puede pasar una curva que separa dos objetos o regiones de la imagen. Así, nosotros los denominaremos el conjunto de *evidencias*.

- Un modelo de la curva o familia de curvas que queremos localizar. Este conjunto de curvas vendrá descrito en forma de una expresión analítica. La expresión general de una expresión analítica podemos escribirla como

  $$f(x,v) = 0 \quad (2.2)$$

donde $v = (a_1,\ldots,a_n)$ es el vector de parámetros de la curva, que corresponde a un punto del *espacio de parámetros*.

Como ejemplos, podemos considerar las rectas, círculos, elipses, etc. La extracción de este tipo de curvas será de gran utilidad para ciertas aplicaciones en las que la localización de estas *características* en la imagen puede servir como entrada para etapas posteriores del procesamiento[13].

Obviamente, para muchas aplicaciones, las curvas que determinan los bordes de objetos no pueden expresarse (al menos de una forma simple) de forma analítica. A pesar de ello, será posible obtener una generalización de la *Transformada de Hough* para estos casos.

2.1.2 Soluciones simples

Algunas soluciones simples y a la vez de difícil aplicación en la práctica debida especialmente a problemas de eficiencia son por ejemplo:

1. Supongamos que queremos localizar las rectas que aparecen en la imagen y que el número de puntos frontera obtenidos es n, es decir, $Cardinal(P_I) = n$. Una posible solución es determinar una recta por cada par de puntos obtenidos y calcular los puntos que caen "cerca" de ella. Este algoritmo implicaría el cálculo de $\frac{n(n-1)}{2}$ líneas y para cada una de ellas del orden de n comprobaciones (para cada uno de los puntos restantes). En definitiva resultaría un algoritmo poco eficiente.

[13] Estas etapas posteriores podrían incluso corresponder a la aplicación de la *transformada de Hough* usando características de más alto nivel semántico que los puntos de frontera(véase Suchendra [9])

2. Supongamos que se quiere localizar un círculo de radio r en la imagen. Se puede construir una máscara de tamaño $2r \times 2r$ que contenga un 1 en cada uno de los puntos que corresponden al círculo y un 0 en el resto. Una posible solución es "pasear" la máscara por la imagen y calcular la correlación. Los valores más altos en el resultado indicarán posibles localizaciones del círculo. Desdichadamente, este algoritmo sólo es posible aplicarlo para una determinada máscara (una sola curva). El plantearlo para una familia de curvas implicaría realizar una operación de correlación para cada uno de los elementos de C, lo que conllevaría a un algoritmo demasiado ineficiente.

Por supuesto, estos dos algoritmos simples, se han particularizado para los casos de las rectas y los círculos, aunque pueden ser generalizados para otras curvas, dando lugar a resultados igualmente poco satisfactorios.

2.1.3 Objetivo

Como se puede observar, en las soluciones simples que acabamos de exponer, nuestra intención ha sido la de evaluar qué número de puntos de la imagen frontera, evidencian un determinado punto del espacio de parámetros. Como resultado del algoritmo, daremos el punto de este espacio con un mayor valor.

Nuestro objetivo será encontrar un método que a partir de la información de puntos frontera, obtenga un valor de acumulación de evidencias (votos) para cada posible t-upla de valores del espacio de parámetros. La solución que buscamos vendrá dada por el máximo obtenido.

En la figura 2.1 podemos ver de forma gráfica esta idea, considerando los casos concretos de rectas y círculos. En la parte izquierda aparecen dos imágenes, que evidencian la existencia de una recta y un círculo, respectivamente. En la parte de la derecha se representan sus respectivos espacios de parámetros, m-c para rectas y (a,b,r) para círculos.

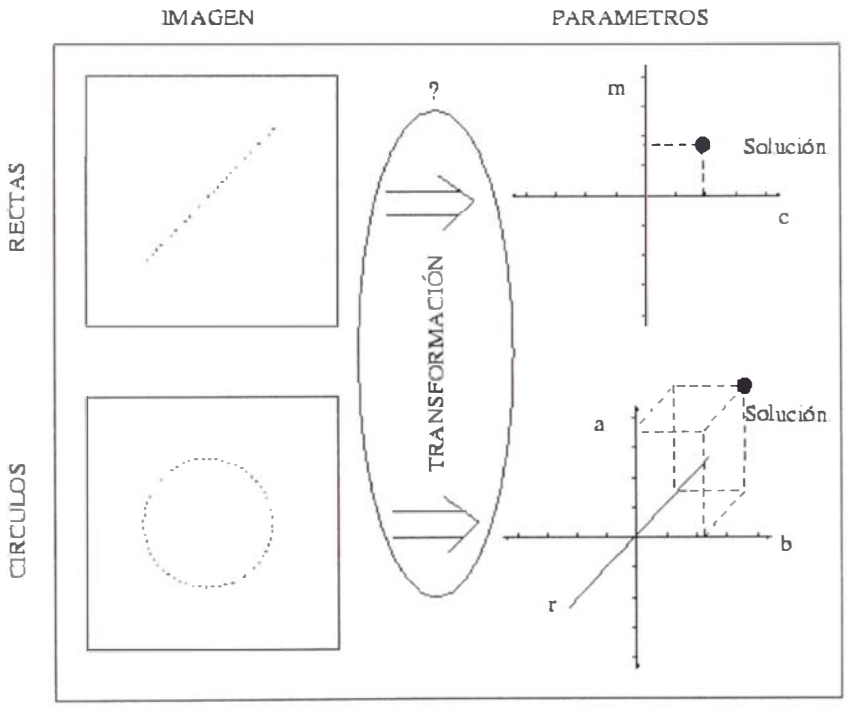


Fig. 2.1: Motivación de la transformada de Hough

Nuestro objetivo es determinar un algoritmo que en un tiempo razonable nos permita transformar, la información sobre localización de fronteras, al espacio de votaciones sobre los parámetros.

A continuación presentamos una solución a este problema, la *transformada de Hough*. La idea intuitiva que podemos tener en cuenta y que es base de la transformada de Hough es:

Recorreremos todos los puntos frontera, y para cada uno de ellos, acumularemos un voto en aquellas posiciones del espacio de parámetros que correspondan a una curva que pasa por dicho punto frontera.

2.2 Transformada de Hough

Como hemos visto en los ejemplos simples del apartado anterior, la localización directa en el *espacio de la imagen* del conjunto de puntos que determina la localización de la curva en dicha imagen no es simple. Este problema se puede reformular a través de la *Transformada de Hough*, la cual establece un método por el que se transforma el conjunto de evidencias que aparecen en el *espacio de la imagen* al *espacio de los parámetros*. De esta manera, el problema de localizar un conjunto de puntos dispersos en el primer espacio se traduce en determinar un único punto en el segundo. Veamos a continuación cómo se puede llevar a cabo dicha transformación para distintas familias de curvas.

2.2.1 Detección de líneas

Las fronteras en forma de líneas rectas son muy comunes en muchas aplicaciones de visión artificial. Considérese por ejemplo la gran cantidad de componentes industriales con líneas rectas, o la gran cantidad de líneas que aparecen en nuestro entorno (edificios, habitaciones, muebles, etc.). Por este

motivo, la detección de este tipo de fronteras ha sido de especial importancia en muchos problemas.

La robustez y eficacia de la transformada de Hough la ha convertido prácticamente en el método más utilizado para la detección de estas líneas rectas desde que Hough, en 1962, patentara el dispositivo electrónico para detectar los caminos que seguían las partículas de alta energía (Hough [6]).

2.2.1.1 Ecuación explícita: espacio m-c

La formulación inicial del método que se propone se realizó sobre el espacio (m,c) que corresponde a la pendiente y el desplazamiento de la recta

$$y = mx + c \tag{2.3}$$

Como se ha indicado anteriormente, nuestro objetivo será realizar una transformación de las evidencias del espacio de la imagen (espacio x-y) al espacio de las rectas (espacio m-c). El concepto básico que se utiliza para la detección de líneas en la transformada de Hough es la dualidad punto-línea.

Por supuesto, un sólo punto en el espacio m-c corresponde, como sabemos, a una recta en el espacio x-y. Por otro lado, un punto en el espacio x-y se puede definir con el punto donde cortan las infinitas líneas (para todos los valores de m) que pasan a través de él. Para un punto (x_0 , y_0), las rectas que pasan a través de él son las rectas que cumplen $y_0 = mx_0 + c$, es decir, los puntos (m,c) que corresponden a la recta $c = -mx_0 + y_0$ del espacio m-c.

Si se consideran un conjunto de n puntos alineados P_I, sólo existe una recta $y = m_0x + c_0$ que pasa a través de todos ellos. Por lo tanto, si dibujamos las n rectas

del espacio m-c que corresponden a cada uno de los puntos obtendremos que todas ellas tienen en común el punto (m_0, c_0).

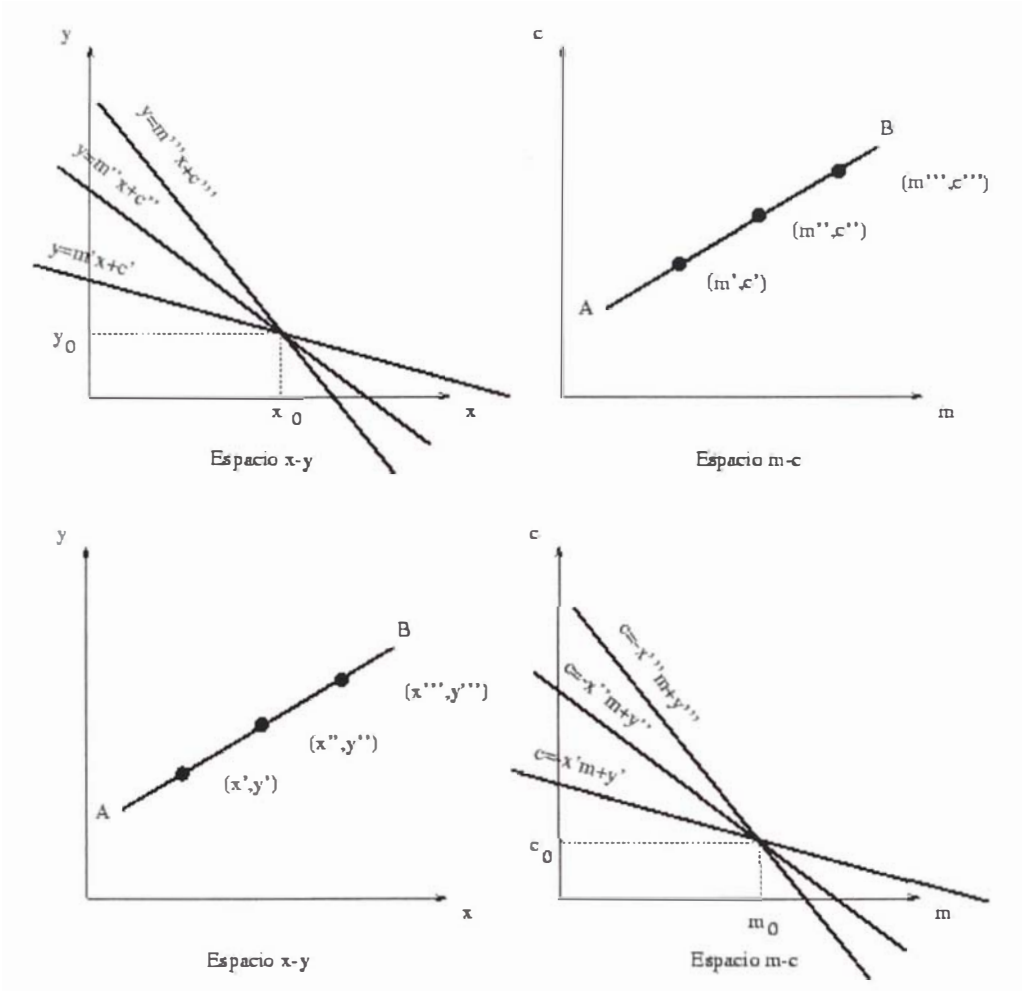


Fig. 2.2: Dualidad punto-línea.

En la figura 2.2 podemos observar gráficamente esta dualidad. Vemos como el haz de líneas que pasa en un punto del espacio x-y corresponde a una línea del espacio m-c. De igual forma, vemos en la parte inferior cómo tres puntos alineados del espacio x-y corresponden a tres líneas que se cortan en un punto

común, es decir, los puntos en una misma línea $y = m_0x + c_0$ corresponden a un haz de líneas en el espacio m-c sobre el punto (m_0, c_0).

Esta situación nos sugiere un método para detectar líneas a partir de un conjunto de puntos P_I. Para ello, interpretamos el espacio m-c como un espacio de votación, es decir, en el que para cada punto se acumulará un valor entero correspondiente al número de votos que indican la presencia de esa recta en la imagen. Para calcular el número de votos en cada posición, hacemos que para cada punto de P_I se incremente en uno todos los puntos que corresponden a la recta del espacio m-c (parte inferior de la figura 2.2).

Como resultado de este algoritmo, en cada punto (m,c) obtenemos un número de votos igual al número de puntos situados sobre la recta $y = mx + c$. Es decir, la búsqueda de un conjunto de puntos alineados en el espacio x-y se convierte en la búsqueda de un punto con un alto valor de acumulación en el espacio m-c (parte superior de la figura 2.2).

Obviamente, el algoritmo se aplica a un conjunto de puntos P_I que corresponden a píxeles en la imagen I. Este algoritmo transforma estos puntos al espacio de parámetros A que resulta de una *discretización* del espacio continuo m-c. Para llevar a cabo esta discretización, será necesario establecer:

- Un rango de valores válidos para cada uno de los parámetros del espacio. En el caso de las rectas tendremos que determinar los intervalos.

$$[m_{min}, m_{max}] \; y \; [c_{min}, c_{max}] \tag{2.4}$$

- Un incremento o precisión en cada parámetro. En el caso de las rectas tendremos que determinar los valores Δm y Δc.

Estos valores dan lugar a un espacio de parámetros A de dimensiones

$$n_m \times n_c = \left(\frac{m_{\max} - m_{\min}}{\Delta m} + 1 \right) \times \left(\frac{c_{\max} - c_{\min}}{\Delta c} + 1 \right) \qquad (2.5)$$

tal como muestra la figura 2.3, de forma que la posición A[m,c] contiene el número de votos de la recta con parámetros (m,c).

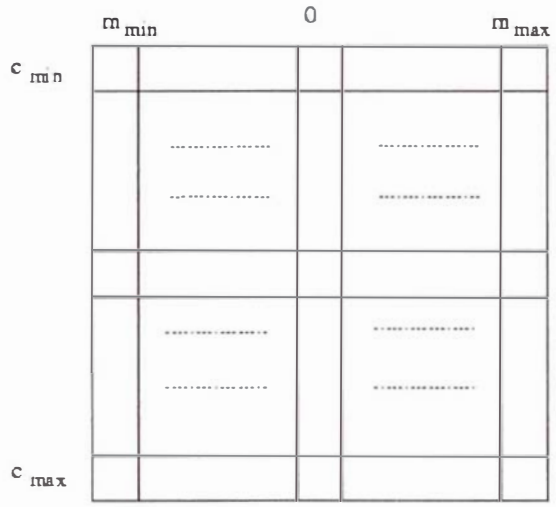


Fig. 2.3: Discretización del espacio m-c.

El algoritmo sería:

Algoritmo de detección de rectas (espacio m-c) Algoritmo 1

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en I.
2. Inicializar el acumulador $A[m,c]$ a cero.

3. Para todo $(x,y) \in P_I$ hacer
    a. Para $m = m_{min}$ hasta $m = m_{max}$ hacer
        i. $c = -xm + y$ (redondeando al valor más cercano de la discretización)
        ii. $A[m,c] = A[m,c] + 1$
4. Los máximos del acumulador $A[m,c]$ corresponden a los puntos frontera que se encuentran alineados en la imagen I.

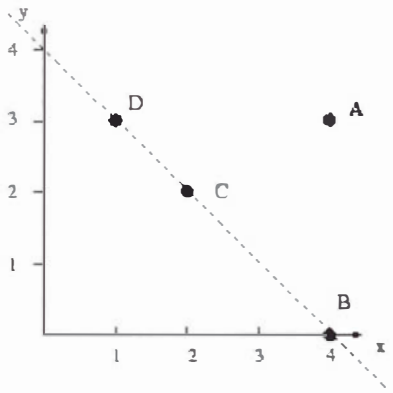
Un ejemplo de la aplicación de este método se presenta en la figura 2.4. En esta figura podemos ver:

1. Espacio de la imagen y mejor solución. Aquí se representan los cuatro puntos (A,B,C,D) de entrada al algoritmo. Como se puede observar, la mejor solución es la recta que pasa por los puntos $\{B,C,D\}$.

2. Puntos y rectas correspondientes. Considerando el haz de rectas que pasa por cada uno de los puntos, se puede representar la recta que corresponde a ese haz en el espacio de los parámetros. Aquí se muestran los puntos en el espacio de la imagen y su recta correspondiente en el de parámetros.

3. Espacio de parámetros. Aquí mostramos la representación gráfica en el espacio continuo m-c de las cuatro rectas que se han obtenido. Se ve claramente que en el punto (-1,4) se cruzan 3 rectas y por tanto estos valores corresponderán a la pendiente y desplazamiento de la recta de mejor solución.

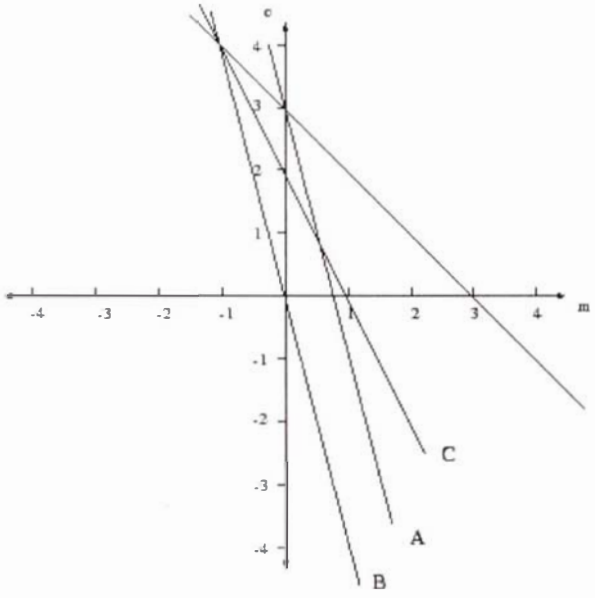
4. Finalmente representamos una matriz que corresponde a la discretización del espacio m-c, es decir, al acumulador usado en el algoritmo. En este hemos tomado como valores de m los del rango [-1,3] con saltos de 0.5 y como valores de c los del rango [0,4] con saltos igualmente de valores 0.5. Se ha representado un círculo negro por cada voto. Como podemos ver, se han obtenido un máximo en la casilla (-1,4)



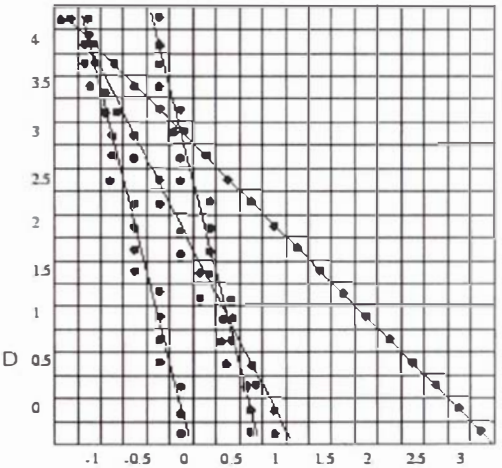
Espacio de la imagen y mejor solución

	Puntos		Rectas
	X	Y	c=-xm+y
A	4	3	c=-4m+3
B	4	0	c=-4m+0
C	2	2	c=-2m+2
D	1	3	c=-1m+3

Puntos y rectas correspondientes



Espacio de parámetros



Espacio de parámetros discretizado

Fig. 2.4: Transformada de Hough usando el espacio m-c.

2.2.1.2 Fuentes de error en la Transformada de Hough.

Aunque desde un punto de vista teórico hemos podido observar la eficacia del método de la transformada de Hough, desde un punto de vista práctico, tendremos que tener especial cuidado en su implementación debido a diferentes fuentes de error. Podemos distinguir las siguientes:

1. *Votación uniforme en el espacio de parámetros*. Consideremos el algoritmo expuesto en el apartado anterior y el ejemplo de la figura 2.4. En el algoritmo recorremos el rango del parámetro m para obtener los valores correspondientes a c y realizar la votación. Una primera implementación de este método podría llevarnos a recorrer cada uno de los valores de m representado en el acumulador y para cada uno de ellos obtener el punto de votación. Si se observa el espacio de parámetros discretizado de la figura vemos que esto nos llevaría a un grave error ya que algunas de las rectas[14] discretizadas no podría representarse de forma apropiada. El resultado sería una recta "a saltos" pues para un valor concreto de m le corresponden varios de c. Esto puede provocar errores debido a que no se ha votado en algunos puntos válidos del espacio de parámetros y puede llevarnos a eliminar votos de la posición donde deberíamos obtener el máximo que buscamos.

   La solución a este problema es una correcta representación de la curva en el espacio de parámetros. Esta se puede obtener aplicando

[14] Las rectas que generarían problemas serían aquellas cuya pendiente fuese mayor que 1 o menor que -1

algoritmos de dibujo de curvas en un espacio discreto, tales como los usados en informática gráfica. Así, el bucle interior del algoritmo presentado debería reescribirse de acuerdo a alguno de ellos. Por ejemplo, usar el algoritmo de *Bresenham (Profesor de Ciencias de la Computación, PhD Stanford University, 1964)*[15] para obtener los puntos de la recta $c = -mx + y$ en el espacio discreto $A[m,c]$.

2. *Di cretización del espacio de parámetros*. Como podemos observar en la figura 2.4, Existen tres puntos destacados en el espacio continuo m-c, correspondientes a un corte de 3 rectas y dos de 2 rectas. Sin embargo, en el espacio de parámetros discretizado existen más de tres. Este error se debe a la gruesa discretización que se ha llevado a cabo. Si los parámetros no se muestrean suficientemente, en algunas celdas podremos acumular votos que nos lleven erróneamente a la detección de máximos.

   Podemos plantear una solución mediante el aumento de la precisión del espacio de parámetros a cambio de recursos en tiempo y espacio. Efectivamente, esta solución provocaría una mayor dispersión de votos en varias casillas evitando la acumulación excesiva que comentamos. Sin embargo, debemos tener en cuenta que

   - Existe un error de discretización en el espacio de parámetros. Consideremos por ejemplo, que los parámetros de la curva a localizar corresponden a un valor intermedio a dos celdas. Debido

[15] Anexo 17

a la aproximación de las curvas teóricas cuando las dibujamos podemos obtener una distribución de votos en varias casillas distintas. Esta situación dificultaría la obtención de un máximo en ese punto. Nótese que esta dispersión de votos puede aumentarse si incrementamos la precisión del muestreo de los parámetros.

- Existe un error de discretización en el espacio de la imagen. La situación de la figura 2.4 es ideal en el sentido de que los tres puntos alineados están situados exactamente en la misma recta. Además, esta recta se representa justo en una celda del espacio de parámetros. En la práctica, el conjunto de puntos de la imagen se obtendrá probablemente tras la aplicación de un detector de fronteras que unido a los errores de discretización dará lugar a un conjunto de puntos lejos de ser ideal. Este ruido en los datos de entrada provoca igualmente una dispersión de los votos que se ve incrementada si aumentamos la precisión del muestreo del espacio de parámetros.

En definitiva, los errores de discretización nos llevan por un lado a una acumulación excesiva de votos y, por otro, a una dispersión de éstos.

En la práctica, la discretización del espacio de parámetros se realiza hasta un nivel de precisión suficientemente alto como para evitar la acumulación excesiva de votos y obtener una cota de error aceptable para la aplicación. La localización de los máximos puede verse afectada por la *dispersión de máximos*, por lo que dicha fase se acompaña de algún

procedimiento de postprocesado del espacio de parámetros como un alisado. Finalmente, se obtienen como solución los puntos de valores altos que corresponden a máximos locales (dado un máximo, se eliminan como candidatos los puntos de su entorno).

3. *Expresión analítica de la curva a localizar*. Para una misma curva, podemos escoger entre varias expresiones analíticas para representarla. Obviamente, esta elección afecta directamente a la forma y significado del espacio de parámetros. Dos características deseables para dichos parámetros son que estén acotados y que tengan una precisión uniforme a lo largo de los distintos valores de rotación de la curva.

   Si volvemos sobre el ejemplo de la figura 2.4, vemos que en el espacio de parámetros continuo se observan otros dos cortes en los que sólo intervienen dos rectas. Estos corresponden a las rectas A,D y A,C. Es interesante destacar que no aparece la recta A,B ya que esta tendría que estar representada en un valor de pendiente infinita. Este es un ejemplo claro en el que el rango de valores de m no está acotado (la recta A,B no es representable) y la precisión no es uniforme.

   Este problema sobre la representación que se ha usado nos lleva a realizar una nueva propuesta, en la que podremos representar fácilmente cualquier recta de la imagen en un espacio de parámetros acotado.

2.2.1.3 Ecuación "normal": espacio θ - p

Hemos podido ver la forma en que podríamos aplicar la transformada de Hough usando el espacio m-c. En este apartado vamos a plantear una nueva representación para obtener un espacio de parámetros más adecuado para la detección de rectas.

En primer lugar podríamos proponer una segunda representación basada en el espacio m-c que nos evitara el problema del rango no acotado. Esta solución consiste en establecer dos conjuntos de puntos, aquellos que tienen pendientes (en valor absoluto) menores que 1 y los que tienen mayores o iguales a 1. Para poder representar los valores altos en este segundo caso, la ecuación 2.3 se reemplaza por

$$x = \tilde{m}y + \tilde{c} \tag{2.6}$$

donde

$$\tilde{m} = \frac{1}{m} \tag{2.7}$$

y por tanto los valores de la pendiente vuelven a estar acotados en el rango [-1.0,1.0].

Sin embargo, no se va a adoptar esta solución sino la propuesta por Duda y Hart [5] en la que se propone usar la ecuación "normal" definida sobre los parámetros (θ,p)

$$p = x\cos\theta + y\sin\theta \tag{2.8}$$

En la figura 2.5 se puede observar la interpretación de los parámetros de la recta y la discretización del plano θ - p en celdas. Como se puede observar, p es la distancia del origen a la recta (distancia del origen al punto más cercano) y θ corresponde al ángulo formado por la recta, que une el origen con el punto más cercano, y el eje de abcisas.

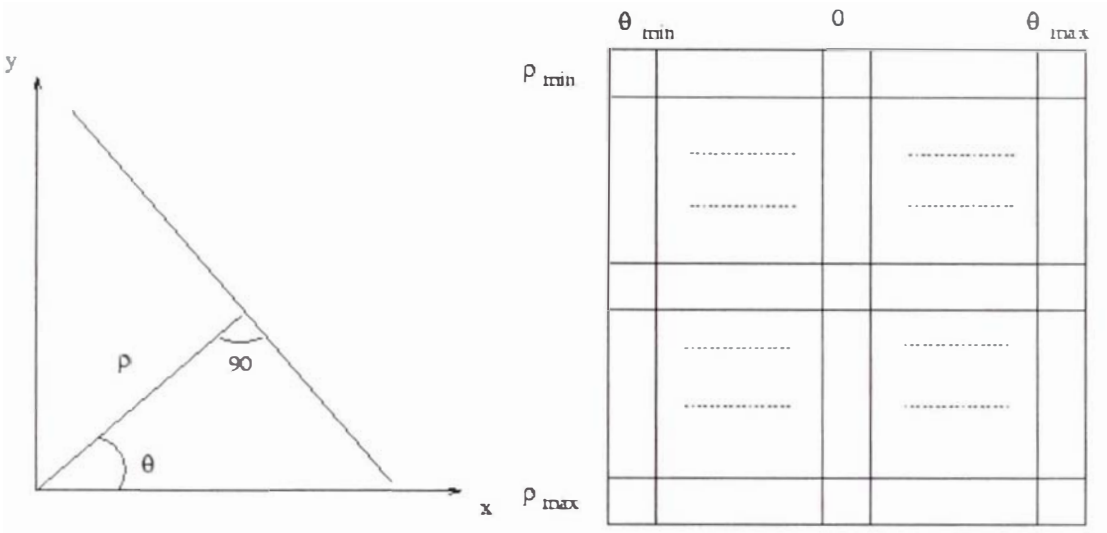


Fig. 2.5: Representación "normal" de la recta y discretización del espacio θ, p

En el caso anterior (espacio m-c), un haz de líneas en el espacio de la imagen se convertía en una línea en el espacio de los parámetros. En esta formulación, un haz de líneas se transforma en una curva senoidal.

$$(x_0, y_0) \Rightarrow \left\{ \begin{array}{ll} c = -mx_0 + y_0 & Si \text{ usamos el espacio m - c} \\ p = x_0 \cos\theta + y_0 sen\theta & \text{Si usamos el espacio } \theta \text{ - p} \end{array} \right\} \qquad (2.9)$$

En la figura 2.6 se muestra un ejemplo de la transformada de Hough para localizar las rectas usando el espacio θ - p. En la parte superior izquierda se han representado los cinco puntos de entrada al algoritmo. En la parte superior

derecha se han pintado con trazo discontinuo las dos rectas principales, es decir, las que acumulan más votos (en este ejemplo simple, sólo tres votos). En la parte inferior se muestra el espacio de parámetros.

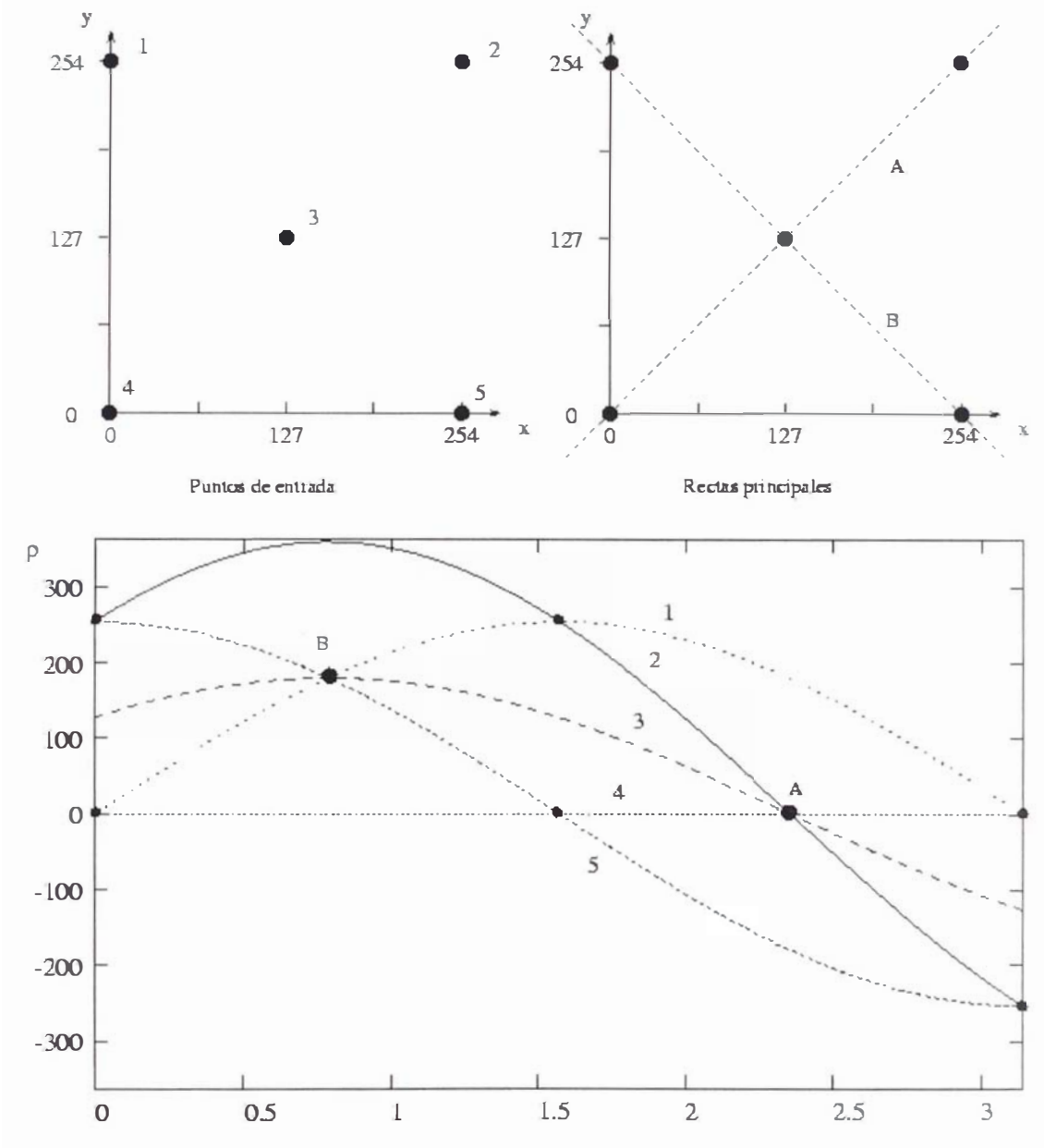


Fig. 2.6: Transformada de Hough usando el espacio θ- p.

Como se puede observar, aparecen dos puntos destacados (marcados con dos círculos y las letras A, B), correspondientes a las rectas A y B que se pretendían buscar. En cada uno de estos puntos se cruzan tres curvas que provienen de cada uno de los puntos que pertenecen a cada recta. Las dos curvas principales por tanto son las correspondientes a parámetros $(179, \frac{\pi}{4})$, es decir,

$$179 = \sqrt{127^2 + 127^2} = x\cos\tfrac{\pi}{4} + y\sin\tfrac{\pi}{4} \tag{2.10}$$

y la de parámetros $(0, \frac{3\pi}{4})$, es decir,

$$0 = x\cos\tfrac{3\pi}{4} + y\sin\tfrac{3\pi}{4} \tag{2.11}$$

Por otro lado hemos marcado con círculos de menor radio otros cruces de las curvas, en este caso de sólo dos de ellas. Estos puntos coinciden con las cuatro rectas de dos votos que pueden obtenerse de los datos de entrada (dos verticales y dos horizontales). Es de destacar que los puntos marcados en (0,0) y (π,0) se refieren a la misma recta así como los puntos (0,254) y (π,-254). Nótese que este efecto se debe a la forma en que los parámetros p y θ cambian en los límites 0y π.

2.2.2 Ejemplos

2.2.2.1 Ejemplo 1: 4 líneas.

En primer lugar vamos a estudiar un ejemplo muy simple. Es el caso de una imagen en la que sólo aparecen 4 líneas rectas. El ejemplo se muestra en la figura 2.7. Como podemos ver, en la parte izquierda se muestra una imagen de

tamaño 200 x 200 con cuatro líneas numeradas. En la parte derecha se muestra el espacio de parámetros correspondiente, donde se ha señalizado la localización de los 4 máximos de cada una de las rectas de entrada. También se puede observar un posible máximo (no señalizado) en coordenadas (π,-100) que corresponde a la recta número 1 y que es consecuencia de la "continuidad reflejada" del espacio de parámetros (corresponde al máximo (0,100)).

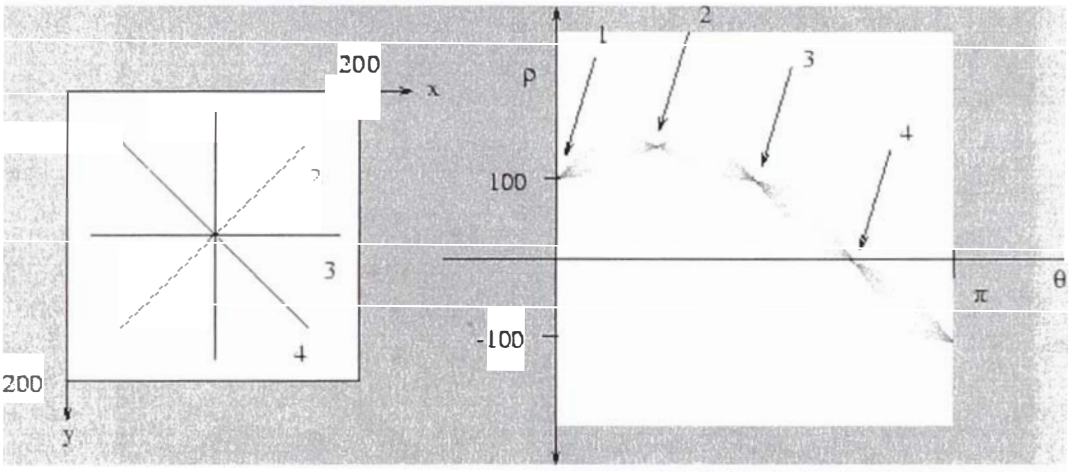


Fig. 2.7: Ejemplo sencillo de 4 líneas rectas.

2.2.2.2 Ejemplo 2: imagen de una casa.

El segundo ejemplo (figura 2.8) corresponde a la imagen de una casa. En este caso, se han localizado las fronteras de la imagen original, sobre las que se ha aplicado el algoritmo de detección de líneas con la condición de que aparezcan cualquiera líneas que acumulen más de 10 puntos. Las imágenes que se muestran son:

A. Imagen original de dimensiones 256 x 256 con 256 niveles de gris.

B. Fronteras obtenidas desde la imagen A.

C. Espacio de parámetros resultante de la imagen B.

D. Líneas detectadas.

E. Segmentos de línea originales detectados sobre la imagen D.

F. Segmentos de línea determinados por la existencia de puntos frontera de la imagen B.

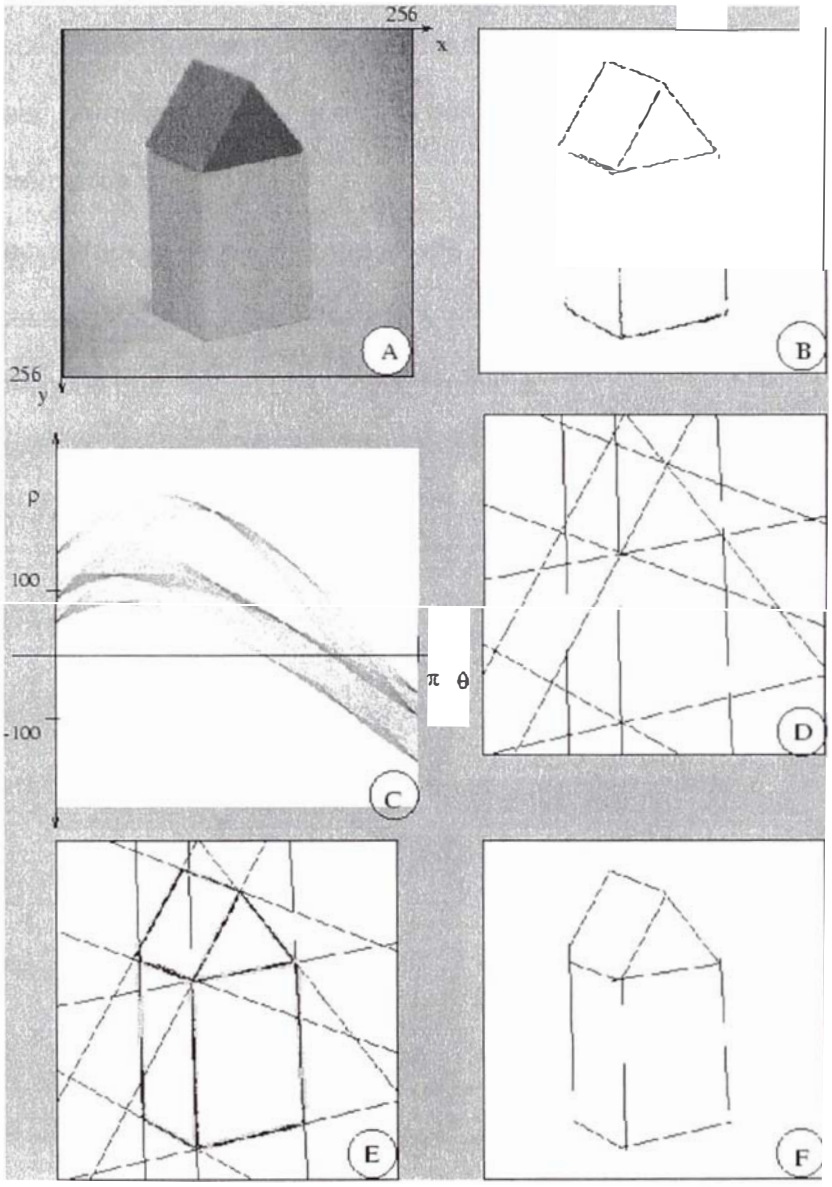


Fig. 2.8: Detección de líneas de una imagen real.

2.2.2.3 Ejemplo 3: imagen ruidosa

En este último ejemplo (figura 2.9), se muestra el resultado a partir de una imagen que contiene 6 líneas y que ha sido contaminada por un 10% de ruido. Las imágenes que se muestran son:

A. Imagen original de 6 líneas con ruido.
B. Espacio de parámetros obtenido desde la imagen A.
C. Seis primeras líneas que se han obtenido desde el espacio de parámetros B.
D. Recuperación de segmentos desde las imágenes A y C. El algoritmo aplicado sigue la línea obtenida y dibuja todos los segmentos de línea continuos que tengan una longitud mínima.

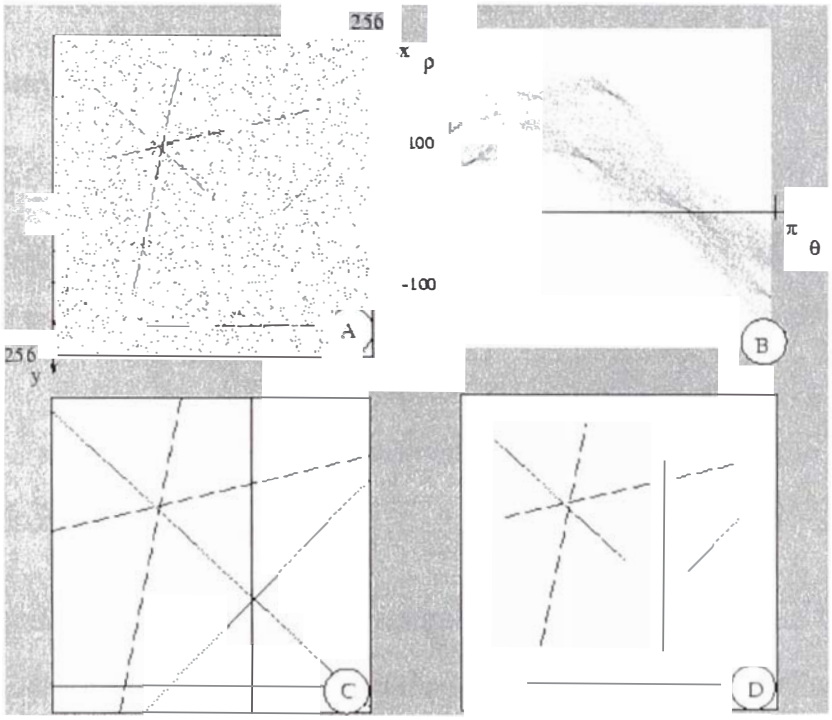


Fig. 2.9: Detección de líneas en una imagen ruidosa.

Como se ha indicado, la condición de extracción de máximos para este caso ha sido que hemos establecido a priori que queríamos obtener 6 líneas. Obviamente, en una aplicación real, es difícil establecer una condición general que sea válida para todas las entradas. Nótese que en la imagen ruidosa pueden existir un gran número de rectas que, conteniendo un número suficiente de puntos, sólo estén compuestas por ruido y por tanto no sean válidas al no tener continuidad.

En la práctica se puede establecer un número mínimo de puntos para obtener una respuesta positiva en la detección de la línea, y posteriormente aplicar algún algoritmo de extracción de los puntos que componen el segmento en la imagen de fronteras para confirmar su existencia (los resultados de esta extracción se muestran en la figura 2.8-F y figura 2.9-D).

2.2.3 Generalización a una curva cualquiera.

Es sencillo llevar a cabo una generalización del algoritmo que se ha presentado para curvas generales definidas por una expresión analítica de la forma

$$f(x,y,v) = 0 \qquad (2.12)$$

donde $v = (v_1,\ldots,v_p)$ es el vector de parámetros de la curva y (x,y) es un punto en el espacio de la imagen.

Al igual que en el caso de la detección de líneas, tendremos que definir un acumulador A que resulte de la discretización del espacio continuo de los parámetros. Para ello será necesario

- Discretizar el espacio p-dimensional de parámetros de la curva, determinando valores mínimos, máximos e incrementos para cada parámetro V_i

$$v_{i,min}, v_{i,max}, \Delta v_i \tag{2.13}$$

dando lugar a un conjunto discreto de n_{V_i} valores para el parámetro i-ésimo.

- Definir el espacio discreto p-dimensional $A[V_1,...,V_p]$ de dimensiones $n_{V1} \times ... \times n_{Vp}$.

Además, supongamos una ecuación $V_p = F(V_1,...,V_{p-1},x,y)$ que relaciona el punto frontera (x,y) con la curva de parámetros $(V_1,...,V_p)$ a partir de la ecuación 2.12.

Así, la generalización del algoritmo 1 sería:

Algoritmo de detección de curvas con expresión analítica Algoritmo 2

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se Inicializa el acumulador A a cero.
3. Para todo $(x,y) \in P_I$ hacer
    a. Para toda dupla $(V_1,...,V_{p-1})$ del acumulador hacer
        i. $V_p = F(V_1,...,V_{p-1},x,y)$
        ii. $A[V_1,...,V_p] = A[V_1,...,V_p] + 1$
4. Los máximos del acumulador $A[V_1,...,V_p]$ corresponden a las curvas en la imagen I.

Sin embargo, en la práctica puede ser más interesante usar una representación paramétrica de la curva que facilite el cálculo de las posiciones a incrementar en el acumulador. Además, estas representaciones pueden facilitar el uso de algoritmos de informática gráfica para asegurar la votación uniforme y mejorar la eficiencia (véase la sección "Fuentes de error en la Transformada de Hough").

Para ello, se definen dos funciones introduciendo una variable independiente t

$$V_p = f_1(V_1,\ldots,V_{p-2},x,y,t) \tag{2.14}$$

$$V_{p-1} = f_2(V_1,\ldots,V_{p-2},x,y,t) \tag{2.15}$$

para valores de t en el rango $[0,2\pi]$.

Una vez que se han fijado los valores de $(V_1,\ldots,V_{p-2},x,y,t)$, (V_p,V_{p-1}) es un plano 2-dimensional y las ecuaciones 2.14 y 2.15 definen una curva que se puede dibujar en dicho plano usando algoritmos de informática gráfica (optimizados por los investigadores en este campo).

Es interesante destacar la elección de los parámetros (V_p,V_{p-1}) como el desplazamiento de la curva en el plano x-y. En este caso, la precisión en estos parámetros podemos relacionarla directamente con la de la imagen y podemos establecer $(\Delta V_p, \Delta V_{p-1}) = (\Delta x, \Delta y)$.

Si renombramos estos dos parámetros como (a,b) y reformulamos las ecuaciones de la curva como

$$(x,y) = (a,b) + (f_x(V_1,\ldots,V_{p-2},t), fy(V_1,\ldots,V_{p-2},t)) \Rightarrow 0 < t < 2\pi \tag{2.16}$$

podemos proponer el siguiente algoritmo

Algoritmo de detección de curvas (paramétrica) Algoritmo 3

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se Inicializa el acumulador A a cero.
3. Para todo $(x,y) \in P_I$ hacer
    a. Para toda dupla $(V_1,...,V_{p-2})$ del acumulador hacer
        i. Para todo t tal que $0 < t < 2\pi$ hacer
            1. $a = Cuantizar(x - f_x(V_1,...,V_{p-2},t))$
            2. $b = Cuantizar(x - f_y(V_1,...,V_{p-2},t))$
            3. $A[V_1,...,V_{p-2},a,b] = A[V_1,...,V_{p-2},a,b] + 1$
4. Los máximos del acumulador $A[V_1,...,V_p]$ corresponden a las curvas en la imagen I.

Si estudiamos detenidamente los pasos que realiza el bucle más interno, nos podemos dar cuenta que no es más que el dibujo de la curva

$$(-f_x(V_1,...,V_{p-2},t), -f_y(V_1,...,V_{p-2},t)) \Rightarrow 0 < t < 2\pi \qquad (2.17)$$

centrada en el punto (x,y), en el plano $A[V_1,...,V_{p-2}]$.

2.2.3.1 Detección de círculos.

En este apartado vamos a ver como la transformada de Hough para líneas puede formularse para la detección de círculos en una imagen siguiendo la generalización que acabamos de exponer.

La ecuación no paramétrica de la curva a localizar se puede escribir como

$$f(x,y,a,b,r) = (x-a)^2 + (y-b)^2 - r^2 = 0 \quad (2.18)$$

donde (a,b) determina el centro del círculo (el rango de valores de este punto central puede ser cualquier valor (x,y) de la imagen) y r es el radio del círculo (en un rango $[1,n_r]$).

Como vemos, el espacio de parámetros es ahora 3-dimensional y por tanto será necesario definir un acumulador de la forma A[a,b,r] de dimensiones $N \times N \times n_r$.

Esta ecuación puede reescribirse para obtener un parámetro en función de los demás de la siguiente forma

$$r = +\sqrt{(x-a)^2 + (y-b)^2} \quad (2.19)$$

Por lo tanto, el algoritmo queda definitivamente como

Algoritmo de detección de círculos (no paramétrica) Algoritmo 4

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.

2. Para todo (a,b,r) tal que $a \in \{0,...,N-1\}$, $b \in \{0,...,N-1\}$, $r \in \{1,...,n_r-1\}$ hacer
    a. $A[a,b,r] = 0$
3. Para todo $(x,y) \in P_I$ hacer
    b. Para todo (a,b) tal que $a \in \{0,...,N-1\}$, $b \in \{0,...,N-1\}$ hacer
        i. $r = Cuantizar\left(+\sqrt{(x-a)^2 + (y-b)^2}\right)$
        ii. $A[a,b,r] = A[a,b,r] + 1$
4. Los máximos del acumulador $A[a,b,r]$ corresponden a las curvas en la imagen I.

Sin embargo, también podemos usar las ecuaciones paramétricas de la curva. En este caso, el círculo puede expresarse como

$$a = x - r\cos t \tag{2.20}$$

$$b = y - r\,sen\,t \tag{2.21}$$

para valores de t en el rango $[0,2\pi]$.

Por tanto podemos construir un algoritmo usando estas expresiones. Este se podría escribir como

Algoritmo de detección de círculos (paramétrica) Algoritmo 5

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Para todo (a,b,r) tal que $a \in \{0,...,N-1\}$, $b \in \{0,...,N-1\}$, $r \in \{1,...,n_r-1\}$ hacer
    a. $A[a,b,r] = 0$

3. Para todo $(x,y) \in P_I$ hacer
    a. Para todo (r,t) tal que $r \in \{1,...,n_r - 1\}$, t $\in \{0,...,2\pi\}$ hacer
        i. $a = Cuantizar(x - r \cos t)$
        ii. $b = Cuantizar(y - r \operatorname{sen} t)$
        iii. $A[a,b,r] = A[a,b,r] + 1$
4. Los máximos del acumulador $A[a,b,r]$ corresponden a las curvas en la imagen I.

Nótese que el bucle interno no es más que el dibujo de los puntos de una circunferencia con centro en (x,y) y radio r en el plano (a,b). Por tanto podremos usar un algoritmo de dibujo rápido para localizar las celdas que tienen que modificarse.

Un ejemplo de la aplicación del algoritmo a una imagen con 6 puntos se muestra en la figura 2.10. En esta figura se muestran los 6 círculos a localizar, la circunferencia con mayor acumulación de votos (centro (100,100) y radio 50), y finalmente el espacio de parámetros con valores de desplazamiento del centro en el rango [0,200] y valores de radio {40,50,60}. Se puede observar como el máximo del espacio de parámetros se encuentra en la posición $(a,b,r)=(100,100,50)$.

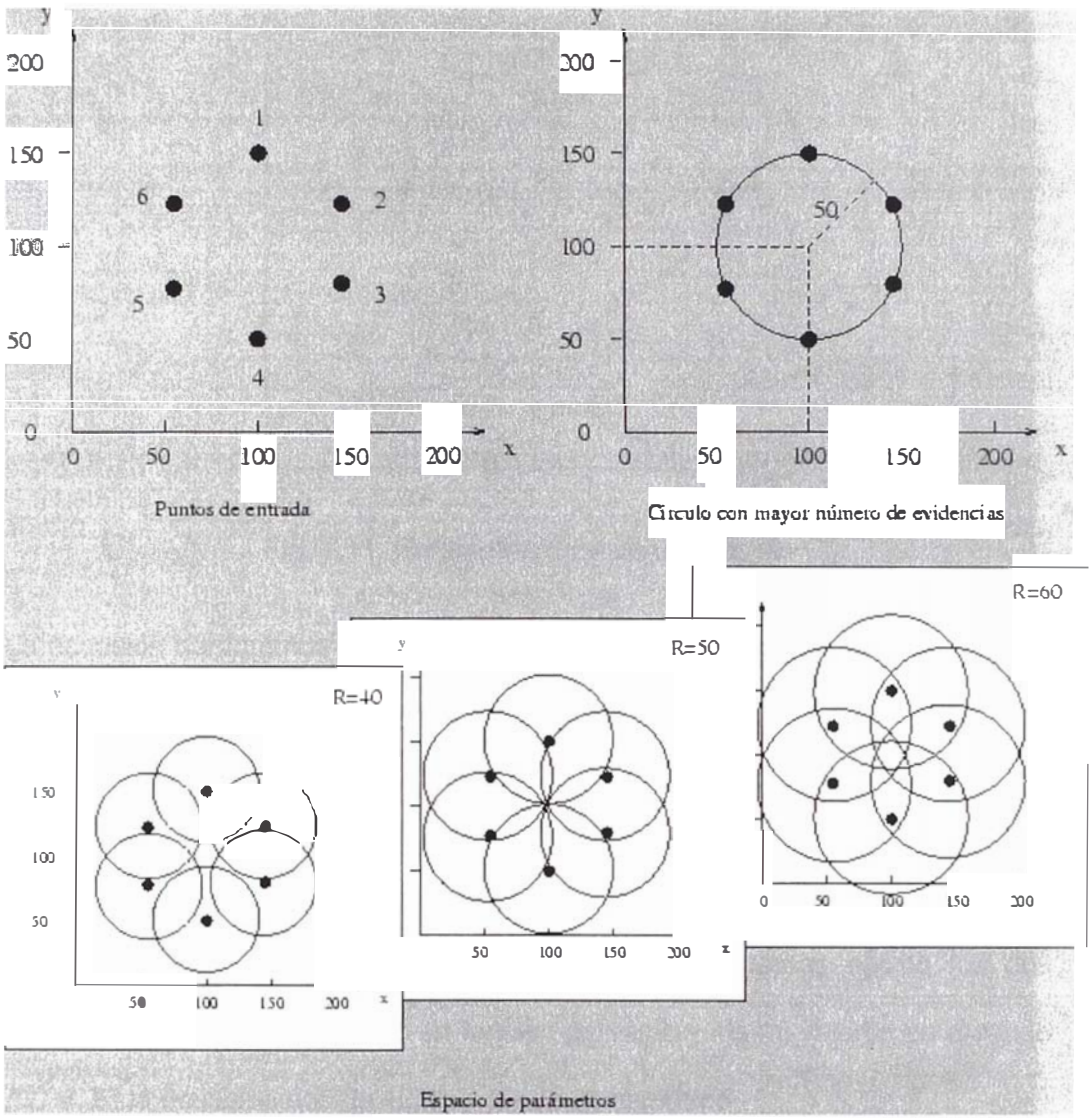


Fig. 2.10: Transformada de Hough para la localización de círculos.

2.2.3.2 Detección de elipses.

Para representar una elipse, podemos utilizar los parámetros (a,b,c,d,θ), donde (a,b) indica el centro de la elipse, c es el semieje mayor, y d es el semieje menor. En la figura 2.11 se muestra gráficamente el significado de estos parámetros.

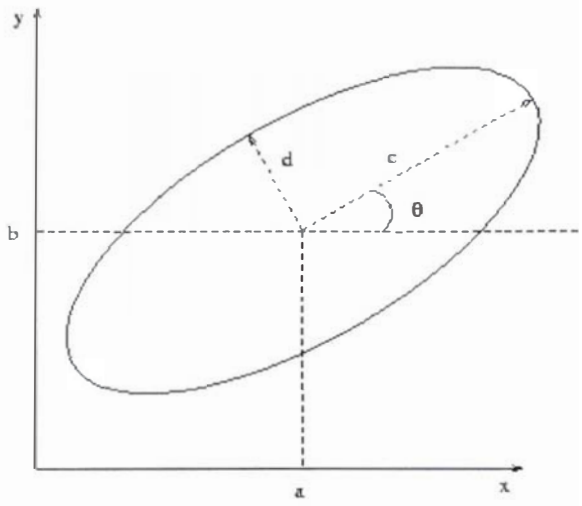


Fig. 2.11: Parametrización de una elipse.

La ecuación paramétrica en forma matricial que podemos usar para construir el algoritmo es

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \cos t \\ \sin t \end{pmatrix} \tag{2.22}$$

donde, de igual forma que en el caso del círculo, hemos escrito los dos parámetros de desplazamiento en función del resto y t toma valores en el rango $[0,2\pi]$. Esta ecuación nos da lugar al siguiente algoritmo

Algoritmo de detección de elipses Algoritmo 6

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Para todo (a,b,c,d,θ) del espacio paramétrico discretizado hacer
   a. $A[a,b,c,d,\theta] = 0$
3. Para todo $(x,y) \in P_I$ hacer
   a. Para todo (c,d,θ,t) hacer

i. $a = Cuantizar(x - (c\cos\theta\cos t - d\,sen\theta\,sen\,t))$

ii. $b = Cuantizar(y - (c\,sen\theta\cos t - d\cos\theta\,sen\,t))$

iii. $A[a,b,c,d,\theta] = A[a,b,c,d,\theta] + 1$

4. Los máximos del acumulador $A[a,b,c,d,\theta]$ corresponden a las curvas en la imagen I.

Nótese de nuevo, que el bucle interior corresponde al dibujo de una elipse de centro (x,y), semieje mayor c, semieje menor d y ángulo de inclinación θ. Por tanto, los cálculos que hay que realizar no son necesarios si sustituimos este bucle por un algoritmo de dibujo de elipses en el plano (a,b) para cada t-upla de valores (c,d,θ) considerada.

2.2.3.3 Generalización a una curva dibujada.

Hasta ahora hemos estudiado la aplicación de la transformada de Hough para curvas con expresión analítica. En este apartado vamos a considerar el problema de buscar un objeto que no tiene una forma analítica simple, sino un determinado contorno[16].

Consideremos una curva C en el plano. Podemos seleccionar un punto de referencia en el plano (a,b) (por ejemplo, el centroide) para definir la siguiente ecuación paramétrica de la curva:

$$(x,y) = (a,b) + (f_x(t),f_y(t)) \qquad 0 < t < 2\pi \tag{2.23}$$

[16] Nótese que vamos a hablar de curvas cerradas por su relación con el contorno de un objeto aunque el algoritmo será válido igualmente para cualquier curva.

La aplicación del algoritmo a la expresión de la curva de la ecuación 2.23 es directa. Si definimos un acumulador $A[a,b]$ para todos los posibles desplazamientos con una precisión igual a la de la imagen, el algoritmo quedaría:

Algoritmo de desplazamiento de una curva cualquiera Algoritmo 7

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se inicializa el acumulador A a cero.
3. Para todo $(x,y) \in P_I$ hacer
    a. Para todo t tal que $0 < t < 2\pi$ hacer
        i. $a = Cuantizar(x - f_x(t))$
        ii. $b = Cuantizar(y - f_y(t))$
        iii. $A[a,b] = A[a,b] + 1$
4. Los máximos del acumulador $A[a,b]$ corresponden a las curvas en la imagen I.

Si estudiamos la operación que se lleva a cabo el bucle más interior, veremos que sólo se trata del dibujo de una curva centrada en el punto (x,y) sobre el plano A. Esta curva es $(-f_x(t),-f_y(t))$, exactamente la misma que la de la ecuación 2.23 reflejada con respecto a su punto central o de referencia, como era de esperar. Nótese que sólo hemos reescrito el algoritmo que habíamos propuesto anteriormente usando la ecuación 2.23. De hecho, para los casos de círculo y elipse que hemos estudiado, se comentó que el bucle interior correspondía al

dibujo de la circunferencia o elipse correspondiente aunque sin referirnos a la necesidad de reflejarlo con respecto al punto central ya que el resultado era la misma curva.

Supongamos que disponemos de una curva C dibujada sobre una imagen. Es posible, por tanto, proponer el siguiente algoritmo:

Algoritmo de desplazamiento de una curva dibujada Algoritmo 8

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se inicializa el acumulador A a cero.
3. Para todo $(x,y) \in P_I$ hacer
    a. Dibujar C reflejada sobre (x,y) en el plano A.
4. Los máximos del acumulador $A[a,b]$ corresponden a las curvas en la imagen I.

En la figura 2.12 se presenta un ejemplo de aplicación de este algoritmo. En la parte superior izquierda tenemos la curva a localizar, a la que se le ha añadido una cruz con línea discontinua indicando el punto de referencia que hemos escogido. En la parte superior derecha, se presentan los puntos que serán entrada al algoritmo junto con la curva sobre ellos para señalar la posición $(100,100)$ como la mejor localización. En la parte inferior izquierda se muestra la curva reflejada con respecto a su punto de referencia, es decir, la que dibujaremos centrada en cada uno de los puntos de evidencia. Finalmente, en

la parte inferior derecha se muestra el espacio de parámetros resultante, en el que se puede ver como el conjunto de curvas coinciden en señalar el punto *(100,100)* como solución. En este punto se encontrará el valor máximo (11) y será la solución final, como era de esperar.

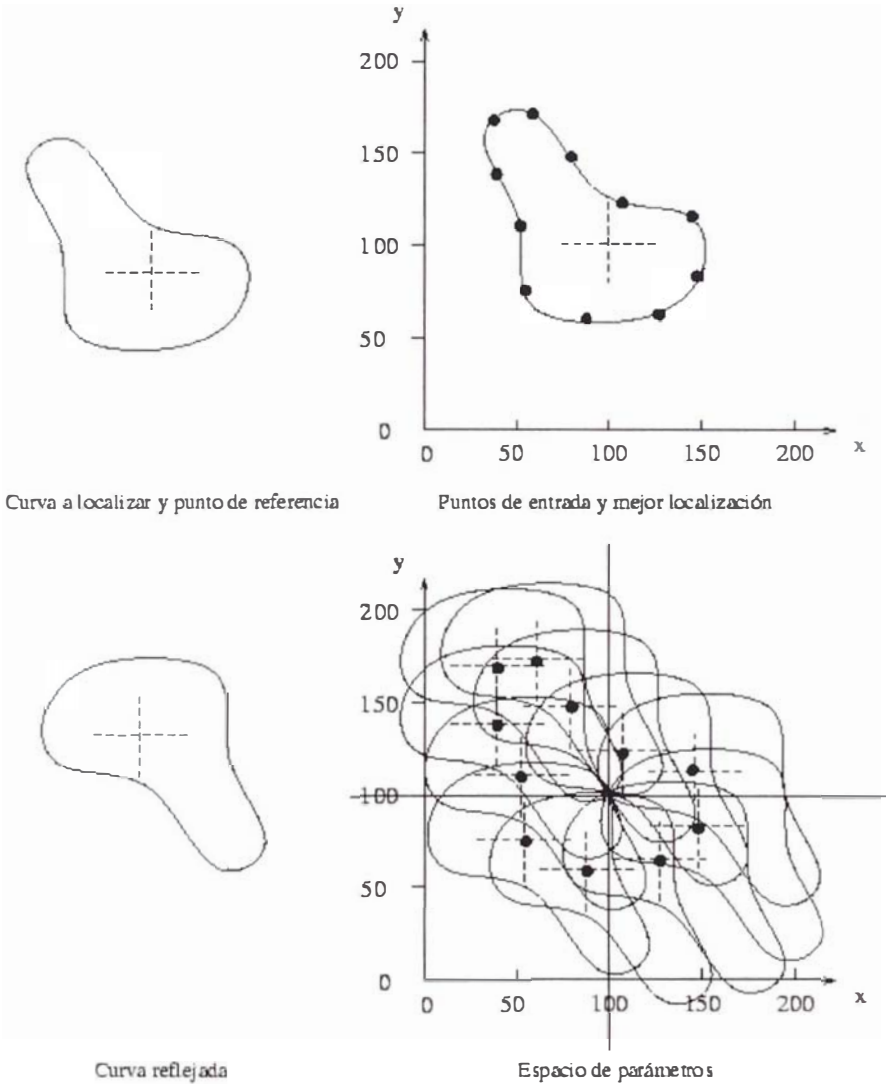


Fig. 2.12: Transformada de Hough para la localización de una curva cualquiera.

2.2.3.4 Introducción de escala y rotación.

En la práctica, la localización de una curva cualquiera no suele aparecer simplificada a la determinación de los parámetros de desplazamiento (a,b) sino que puede venir afectada por una transformación de escala y rotación.

La introducción de estos dos parámetros en los cálculos es trivial ya que sólo tenemos que considerar la forma en que se incluían un número variable de parámetros en las secciones anteriores. Definamos el espacio de parámetros $A[e,r,a,b]$ de parámetros escala, rotación desplazamiento en x y desplazamiento en y. Si disponemos de una curva C dibujada sobre una imagen, el algoritmo para su localización teniendo en cuenta escala y rotación es:

Algoritmo de escala, rotación y desplazamiento de una curva Algoritmo 9

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se inicializa el acumulador A a cero.
3. Para todo $(x,y) \in P_I$ hacer
    b. Para todo (e,r) hacer
        i. Dibujar C escalada con e, rotada r y reflejada sobre (x,y) en el plano $A[e,r]$
4. Los máximos del acumulador $A[e,r,a,b]$ corresponden a las curvas en la imagen I.

Por supuesto, la introducción de nuevos parámetros de transformación de la curva se tendrá en cuenta introduciendo cambios similares.

2.2.4 Ejemplos.

En la figura 2.13 se incluyen tres ejemplos de la generalización de la transformada de Hough:

a. Ejemplo de detección de círculos: la imagen de la izquierda corresponde a las fronteras extraídas. A continuación se presenta el espacio de parámetros para un radio fijo y finalmente la solución obtenida sobre la imagen original.

b. Ejemplo de detección de una curva dibujada: Se presenta la forma dibujada a localizar (izquierda) y el contorno (punteado) sobre una imagen aérea en la que se ha localizado.

c. Una imagen de células, donde se aplica la transformada de Hough para la detección de elipses. La imagen original se presenta en la imagen izquierda mientras que las elipses correspondientes se presentan en la parte derecha.

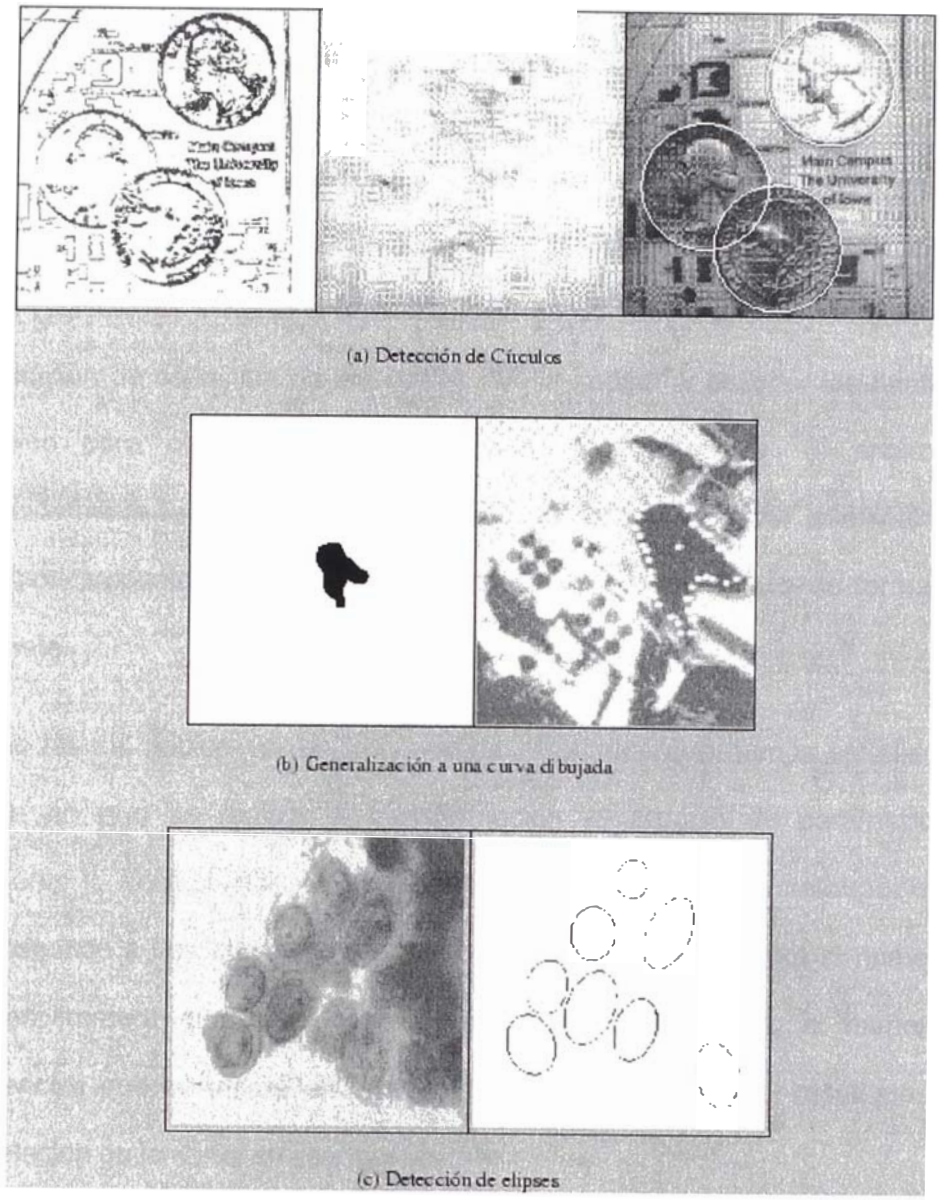


Fig. 2.13: Ejemplos de generalización.

2.3 Eficiencia de la Transformada de Hough

2.3.1 Correlación vs. Transformada de Hough

En la segunda de las soluciones que hemos visto en el primer apartado del tema (soluciones simples a la localización de curvas) proponíamos calcular la correlación de cada una de las curvas con la imagen y escoger los puntos de máximo para obtener las localizaciones más probables. Si analizamos detenidamente la transformada de Hough, podemos observar que realmente estamos proponiendo un algoritmo para realizar dicho cálculo de forma más eficiente.

Como hemos podido ver en los diversos algoritmos que hemos estudiado, la forma en que se realiza la transformación al espacio de parámetros es mediante la acumulación de evidencias. En cada punto de este espacio, que corresponde a una curva C del espacio de la imagen, aparecerá el número de puntos frontera que se sitúan en dicha curva. Por tanto, el número de evidencias que se tendrá en una localización coincide con el resultado de la correlación de la curva en esa localización.

Como indicábamos, el uso de la operación de correlación sobre todas las posibles curvas requeriría una cantidad de tiempo demasiado alta.

Para una determinada localización de la curva, el conjunto de operaciones a realizar para obtener el valor de correlación consiste en la multiplicación punto a punto de los valores de la máscara por los correspondientes de la imagen. Ahora bien, La mayoría de los puntos de la máscara son cero, al igual que

ocurre para la mayoría de los puntos de la imagen. Por tanto, sólo un pequeño porcentaje de las operaciones que se llevan a cabo son realmente útiles en el sentido de que afectan al valor o conjunto de votos de esa posición. La eficacia de la transformada de Hough surge del hecho de que considera sólo estas operaciones, es decir, recorre cada uno de los puntos frontera y cada uno de los puntos que forman la curva de manera que en cada paso realiza una de las operaciones válidas, es decir, que provocan un incremento de un voto en el resultado.

2.3.2 Requisitos de espacio y tiempo.

Uno de los grandes inconvenientes que hacen difícil la utilización de la transformada de Hough en la práctica es la gran cantidad de espacio y tiempo que requiere.

Los recursos en espacio vienen directamente determinados por el tamaño del acumulador que se utiliza. Dado que se utiliza una matriz p-dimensional $A[V_1,...,V_p]$ de dimensiones $n_{V_1} \times ... \times n_{V_p}$ (véase la sección *Generalización a una curva cualquiera*).

En definitiva, el tiempo requerido para esta fase dependerá también de forma exponencial del número de parámetros.

Resumiendo, los factores que afectan directamente a la eficiencia del algoritmo son:

- El número de parámetros.
- El número de muestras de cada parámetro.

- El número de puntos frontera obtenidos.

A fin de aliviar estos requerimientos y hacer más recomendable la utilización de la transformada de Hough, se han propuesto multitud de modificaciones y algoritmos en la literatura. En la siguiente sección vamos a ver un método para mejorar la eficiencia.

2.3.3 Mejora de eficiencia: Uso del gradiente .

Hasta ahora sólo se ha usado como evidencia la situación de los puntos frontera en el espacio de la imagen. Una información que hemos obviado y que resulta muy interesante al mejorar la eficiencia de los algoritmos es el uso del *gradiente*[17]

Cuando se procesa un punto frontera, se realiza una votación en todas aquellas posiciones del espacio de parámetros que pudieran dar lugar a una curva que pasa por ese punto. Si tenemos en cuenta la dirección del *gradiente*, dado un punto y una dirección, sólo votaremos en las posiciones que dan lugar a una curva que pasa por ese punto **y además tienen ese mismo** *gradiente*.

Desde otro punto de vista, la localización de un punto da lugar a una condición (ecuación) que nos fija un parámetro a partir de los demás. El incluir el *gradiente* permite tener en cuenta una nueva condición por lo que podremos fijar dos parámetros a partir de lo demás. En términos de eficiencia el uso de esta información implica, por tanto, disminuir en uno las dimensiones a tener en cuenta durante la votación.

[17] Anexo 14

2.3.3.1 Gradiente aplicado a la detección de líneas rectas.

El algoritmo de votación para la detección de líneas rectas necesitaba recorrer todos los valores de θ y para cada uno de ellos, calcular p para acumular un voto. Dado que podemos considerar que la dirección del *gradiente* es perpendicular a la recta que buscamos, si calculamos el ángulo que forma el vector *gradiente* y el eje de abcisas, obtenemos directamente el parámetro θ de dicha recta. Este valor se puede usar en la ecuación 2.8 y obtener directamente el valor de p.

El algoritmo quedaría

Algoritmo de detección de rectas usando gradiente Algoritmo 10

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en I.
2. Inicializar el acumulador $A[\theta,p]$ a cero.
3. Para todo $(x,y) \in P_I$ hacer
    a. $\theta = Angulo(grad_x(x,y),grad_y(x,y))$
    b. $p = x\cos\theta + y\sin\theta$
    c. $A[\theta,p] = A[\theta,p] + 1$
4. Los máximos del acumulador $A[\theta,p]$ corresponden a los puntos frontera que se encuentran lineados en la imagen I.

Por lo tanto sólo es necesario un voto por cada punto frontera.

2.3.3.2 Gradiente aplicado a la detección de círculos.

Para realizar la votación en la detección de círculos, para cada punto frontera recorremos todos los valores del radio y todos los valores del parámetro t que recorre la curva. Al añadir el *gradiente*, y por tanto una condición más, sólo necesitamos recorrer los distintos valores del radio y determinar para cada uno de ellos el centro del círculo correspondiente[18].

El algoritmo por tanto quedaría como sigue:

Algoritmo de detección de círculos usando gradiente Algoritmo 11

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Para todo (a,b,r) tal que $a \in \{0,...,N-1\}$, $b \in \{0,...,N-1\}$, $r \in \{0,...,n_r-1\}$ hacer
   a. $A[a,b,r] = 0$
3. Para todo $(x,y) \in P_I$ hacer
   a. Para todo (r) tal que $r \in \{1,...,n_r-1\}$ hacer
      i. $t = Angulo(grad_x(x,y), grad_y(x,y))$
      ii. $a = x - r \cos t$
      iii. $b = y - r \operatorname{sen} t$
      iv. $A[a,b,r] = A[a,b,r] + 1$
4. Los máximos del acumulador $A[a,b,r]$ corresponden a las curvas en la imagen I.

[18] Nótese que aquí usamos no sólo la dirección sino también el sentido del gradiente. Si el sentido no nos determina la posición del objeto tendríamos que votar en dos posiciones distintas

2.3.3.3 Gradiente aplicado a la detección de una curva cualquiera.

Como se ha indicado anteriormente, podemos construir un algoritmo para localizar una curva de p parámetros usando las ecuaciones paramétricas. Este algoritmo nos permitía definir directamente la forma en que podíamos localizar una curva dibujada pues básicamente el bucle interior sólo tenía que dibujar la curva (véase la sección *Generalización a una curva dibujada*).

El uso del *gradiente* afecta a este algoritmo eliminando la necesidad de dibujar la curva completa. Cuando no se tenía información de *gradiente*, hacía falta recorrer cualquier punto de la curva ya que la evidencia podría corresponder a cualquiera de esos puntos. Ahora el *gradiente* determina que la evidencia sólo puede corresponder a aquellos puntos de la curva cuya tangente sea perpendicular a ese *gradiente*. Por tanto, el bucle interno que dibujaba la curva, ahora sólo tendrá que representar esos puntos compatibles. Por ejemplo, en el caso de un círculo, para un radio determinado, un punto y su *gradiente* determinan una única circunferencia (como se ha visto en la sección anterior).

Para una curva cualquiera (considérese por ejemplo la de la figura 2.12) la inclusión del *gradiente* se puede realizar mediante la inclusión de una *R - tabla*[19]. La idea consiste en la partición del conjunto de puntos de la curva clasificándolos según su *gradiente* o la tangente al contorno. Cada uno de estos conjuntos será una entrada en la *R - tabla*, y para cada una de estas entradas se

[19] El algoritmo propuesto por Ballard inicialmente como la *Transformada de Hough Generalizada* describía el algoritmo en términos de esta *R-tabla* y contenía los pasos que se describen en esta sección.

almacenan un conjunto de valores *(r, α)* que indican la correspondiente localización del punto de referencia.

La forma de la *R - tabla* se muestra en la tabla 2.1. Si localizamos un punto frontera con una tangente de valor ϕ_i, esta tabla nos lleva al conjunto de puntos que hay que votar, es decir, al primer punto de radio r_i^1 y dirección α_i^1, al segundo punto de radio r_i^2 y dirección α_i^2, hasta el último punto n_i.

$$\begin{array}{cccccc}
\phi_1 & \rightarrow & (r_1^1,\alpha_1^1) & (r_1^2,\alpha_1^2) & \cdots & (r_1^{n_1},\alpha_1^{n_1}) \\
\phi_2 & \rightarrow & (r_2^1,\alpha_2^1) & (r_2^2,\alpha_2^2) & \cdots & (r_2^{n_2},\alpha_2^{n_2}) \\
\phi_3 & \rightarrow & (r_3^1,\alpha_3^1) & (r_3^2,\alpha_3^2) & \cdots & (r_3^{n_3},\alpha_3^{n_3}) \\
\vdots & \rightarrow & \vdots & \vdots & \ddots & \vdots \\
\phi_k & \rightarrow & (r_k^1,\alpha_k^1) & (r_k^2,\alpha_k^2) & \cdots & (r_k^{n_k},\alpha_k^{n_k})
\end{array}$$

Tabla 2.1: R – Tabla

En la figura 2.14 se muestra gráficamente el sentido de estos valores. Se puede observar como dos puntos de la curva tienen una misma tangente ϕ y sus dos parejas de valores *(r_i, α_i)* que indican la posición del centro.

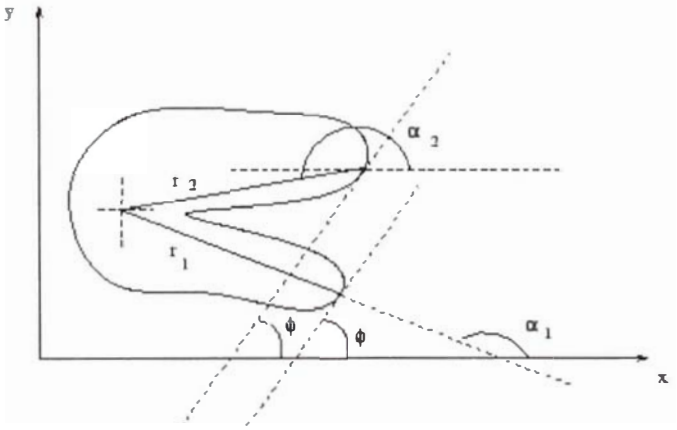


Fig. 2.14: Geometría de la R-Tabla en la Transformada de Hough Generalizada.

En una aplicación que busque esta curva, si localizamos un punto que corresponda a una tangente ϕ, tendremos que votar en dos posiciones de desplazamiento, que podrán ser localizadas con esas dos parejas de valores. El algoritmo para obtener el desplazamiento se muestra a continuación:

Algoritmo de desplazamiento de una curva dibujada Algoritmo 12

1. Obtener P_I, el conjunto de puntos que marcan discontinuidades en la imagen original I.
2. Se inicializa el acumulador A a cero
3. Para todo $(x,y) \in P_I$ hacer
    a. Determinar el valor ϕ de tangente en el punto (x,y)
    b. Para todo (r,α) de la entrada ϕ de la R – Tabla hacer
        i. $a = x + r\cos\alpha$
        ii. $b = y + r\,sen\,\alpha$
        iii. $A[a,b] = A[a,b] + 1$
4. Los máximos del acumulador $A[a,b]$ corresponden a las curvas en la imagen I.

Nótese en este algoritmo como la nueva aportación es que el bucle interior no dibuja toda la curva, sino que utiliza la partición determinada por la R-tabla para dibujar únicamente aquellos puntos que tienen una pendiente determinada.

2.3.3.4 Errores en el gradiente.

El uso de la información de *gradiente* para determinar un subconjunto de posiciones en donde votar puede llevarnos a resultados incorrectos si no tenemos en cuenta el posible error en la estimación de la dirección. Un pequeño error en el valor del ángulo puede implicar un desplazamiento en la posición del voto de varios píxeles con respecto a la posición buscada.

Consideremos el caso de la estimación de θ para aplicarla a la ecuación normal de la recta:

$$p = x \cos \theta + y \sin \theta \tag{2.24}$$

Si el ángulo es cercano a $\pi/2$ y el valor de x es muy alto o el ángulo es cercano a 0 y el valor de y es alto, el error en la estimación de p puede ser del orden de varias unidades.

Si consideramos por otro lado el caso del círculo, si el radio es suficientemente grande, un error pequeño en la estimación del ángulo del *gradiente* puede desplazar varias posiciones el centro del círculo.

En la figura 2.15 se muestran gráficamente estas dos situaciones.

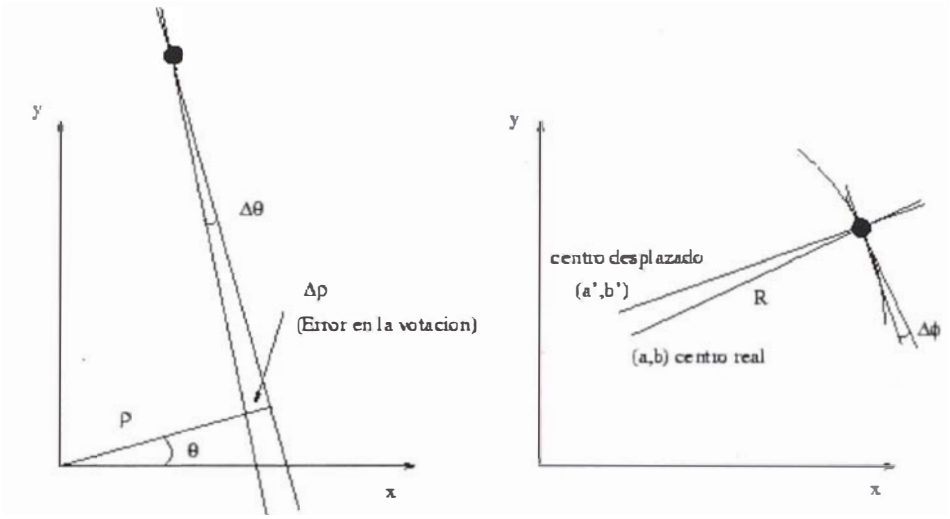


Fig. 2.15: Errores en el *gradiente*.

Por lo tanto, si el error en el cálculo del *gradiente* puede ser bastante alto, tendremos que evitar el desplazamiento que sufren los votos. En la práctica, la solución a este problema puede ser el votar no sólo en el ángulo detectado sino en un entorno de éste. Por ejemplo, para evitar el problema en la detección del círculo, no sólo se vota en el centro *(a,b)* sino en un pequeño arco centrado en ese punto.

CAPÍTULO III

USANDO EL ANÁLISIS DE LOS COMPONENTES PRINCIPALES

3.1 El Rostro visto como un Vector

El rostro , visto como una imagen, puede ser visualizado como un vector. Si el ancho y altura de la cara son **w** y **h** píxeles respectivamente, el número de componentes de este vector será **w*h**. Cada píxel esta codificado dentro de un vector componente. La construcción de este vector partiendo de una imagen se hace por concatenación simple – las filas que componen la imagen son puestas una al lado de otra, como se muestra en la figura 3.1.

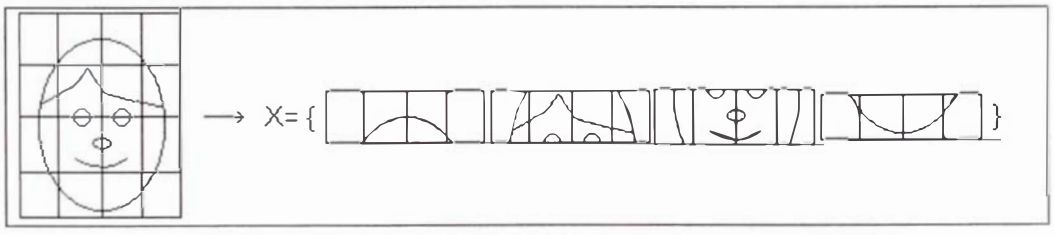


Fig. 3.1: Formación del vector del rostro tomado como origen una imagen del rostro

3.2 Espacio de la Imagen

El vector del rostro descrita en la sección previa pertenece a un espacio. Este espacio es el espacio de las imágenes, el espacio de todas las imágenes cuyas dimensiones son w por h píxeles. El espacio de las imágenes, básicamente, esta compuesta por los siguientes vectores:

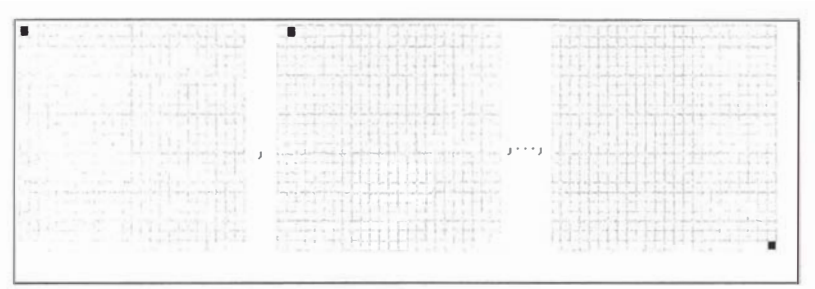


Fig. 3.2: Principio fundamental del espacio de la imagen

Todos los rostros se parecen. Tienen dos ojos, una boca, una nariz, etc. ubicados en el mismo sitio. Por consiguiente, todos los vectores del rostro están localizados en una pequeña zona del espacio de la imagen, como se muestra en la figura 3.3.

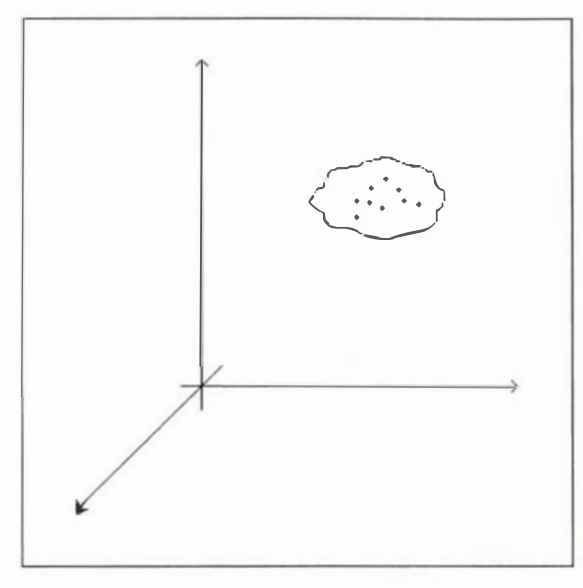


Fig. 3.3: Espacio de la Imagen y el conjunto de los rostros

Por lo tanto, el espacio completo de todas las imágenes no es óptimo para la descripción de los rostros. La tarea presentada aquí tiene como finalidad la construcción de un espacio de rostros que describa mejor a estos mismos. Los vectores básicos de este espacio de rostros son llamados los *Componentes Principales*.

La dimensión del espacio de imágenes es **w*h**. Por supuesto, no todos los píxeles de un rostro son relevantes, y cada píxel depende del entorno en que se encuentre. Así, la dimensión del espacio de rostros es menor que la dimensión del espacio de imágenes. La dimensión del espacio de rostros no puede ser determinada con anticipación, pero por lo antedicho se entiende claramente que este es un subconjunto del espacio de imágenes.

El objetivo del método presentado aquí, el *Análisis de los Componentes Principales*, es reducir la dimensión de un conjunto o espacio de tal manera que el nuevo entorno describa mejor los "modelos" típicos del conjunto. En nuestro caso los

"modelos" son el conjunto de rostros. El entorno nuevo será construido mediante combinación lineal. Los componentes en este espacio de rostros básicos serán *no correlacionados* y *maximizarán la varianza de sus variables originales*. Estas propiedades serán más fácilmente entendidas cuando consideremos el ejemplo final de este capítulo.

El *Análisis de los Componentes Principales* apunta a obtener la variación total en el conjunto de los rostros modelos, y explicar esta variación mediante pocas variables. El hecho de que este procedimiento reduce el número de las variables es importante. En efecto, una observación descrita por pocas variables es más fácil de manejar y más fácil de entender que si esta fuera definida por un gran número de variables. Y cuando muchas observaciones o muchos rostros, en nuestro caso, tengan que ser procesados la reducción del número de variables es de gran importancia.

3.3 Punto de Vista Teórico

El Análisis de los Componentes Principales fue inicialmente desarrollado por los estadísticos. Luego fue reformulado desde un punto de vista de una red neuronal artificial. Así que existen dos maneras para explicar sus principios. Para lograr entender el ACP, se tienen que considerar ambos puntos de vista. Como son complementarios el ACP tiene que ser primeramente explicado desde el punto de vista de una red neuronal artificial (RNA) porque es más intuitiva, mientras que el punto de vista estadístico es mucho más riguroso matemáticamente.

3.4 Memoria Lineal Auto Asociativa

3.4.1 Memoria Lineal Auto Asociativa

Una memoria auto asociativa (MAA) es una clase de memoria direccionable. Esta tiene como objetivo obtener una respuesta la cuál es la clave o llave o identificación memorizada más cercana a la clave o llave o identificación que se está ingresando. En nuestro caso, el término "clave o llave o identificación" puede ser considerado como un rostro.

La memoria lineal auto asociativa (MLAA) está formada por una red de redes neuronales. Cada neurona de está red de redes está asociada con un componente del vector del rostro. Así está red de redes contiene **w*h** neuronas. Adicionalmente, cada neurona se interconecta con todas las otras neuronas, como se muestra en la figura 3.4.

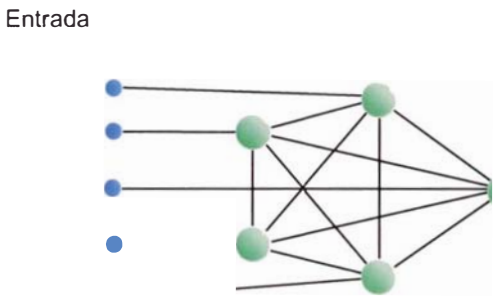


Fig. 3.4: Arquitectura de la Memoria Lineal Auto - Asociativa

De esta manera la MLAA está construida computando los pesos de $(\mathbf{w*h})^2$ redes neurales. Las cargas se calculan durante el proceso de aprendizaje. Durante este proceso, varios rostros de prueba se presentan a la MLAA, la cuál tiene que memorizarlos.

3.4.2 Regla de Aprendizaje de Widrow-Hoff

Durante el proceso de aprendizaje los pesos tienen que cambiar con el fin de minimizar errores. Esto puede hacerse usando muchas reglas. Una de ellas, la cuál se usará aquí, es la regla de aprendizaje de Widrow-Hoff [20].

Definamos:

- x_i son los i^{esimos} componentes del rostro de input.
- o_j es la salida de la neurona j^{esima}.
- w_{ij} es la intensidad de la conexión entre la i^{esima} y la j^{esima} neurona.
- η es la constante de aprendizaje.
- t es el índice de iteración.

En el caso de una memoria auto asociativa, la regla de aprendizaje de Widrow-Hoff para una neurona puede ser formulada como sigue:

$$\mathbf{w}_{ij}^{(t+1)} = \mathbf{w}_{ij}^{(t)} + \eta * (\mathbf{x}_j - \mathbf{o}_j) * \mathbf{x}_i \qquad (3.1)$$

Para la MAA, la salida esperada para una neurona, $\mathbf{o_j}$, es la entrada, $\mathbf{x_j}$. Esta regla dice que si el peso produce un resultado el cuál es diferente que la salida esperada, luego tiene que ser cambiada. La medida de este cambio es proporcional a la excitación de la neurona $\mathbf{x_j}$ y a la constante de aprendizaje $\boldsymbol{\eta}$.

La regla de aprendizaje es una regla iterativa. Este proceso se realiza varias veces para cada rostro del conjunto de aprendizaje. Cuando el proceso se completa para un solo rostro, se llama iteración, cuando se ha realizado sobre

---

[20] Anexo 9

el total del conjunto de rostros de entrenamiento, se llama un estado. Cuando muchos estados son realizados sucesivamente, es obvio que el error, $\mathbf{e}_j = \mathbf{o}_j - \mathbf{x}_j$, es mínimo, si es que se selecciona η correctamente.

Esta regla se puede aplicar al conjunto total de K rostros de entrenamiento usando notación matricial.

Definamos:

- **l = w*h** es el número de píxeles del rostro o el número de componentes del vector del rostro,
- **K** es el número de rostros en el conjunto de entrenamiento,
- **X = [$\mathbf{x}_{ik}$]** es la matriz de dimensión l*K que define el conjunto de entrenamiento. El k^{esimo} vector columna de esta matriz, $\mathbf{x}_k$, corresponde al k^{esimo} rostro del conjunto de entrenamiento.
- **W = [$\mathbf{w}_{ij}$]** es la matriz peso de dimensión l*l.

Desde ahora, se asume que el vector $\mathbf{x}_k$ esta normalizado, lo que significa que $|\mathbf{x}_k| = \sqrt{x_k^T \cdot x_k}$, donde el superíndice T indica que la matriz esta transpuesta.

Luego, la regla de aprendizaje se puede escribir como:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \eta * (\mathbf{X} - \mathbf{W}^{(t)} * \mathbf{X}) * \mathbf{X}^T \qquad (3.2)$$

Esta es la formula de un estado, y aquí el índice t es el índice de un estado.

Antes de continuar, es necesario hacer algunas explicaciones adicionales sobre las matrices. Una matriz rectangular real, X, puede ser descompuesta en *valores singulares*[21]:

[21] Anexo 19

Definamos:

- **P** es la matriz de los *eigenvectors*[22] de la matriz $\mathbf{X}*\mathbf{X}^T$,
- **Q** es la matriz de los *eigenvectors* de la matriz $\mathbf{X}^T*\mathbf{X}$,
- Δ es la matriz de los valores singulares (esto es $\Delta = \sqrt{\Lambda}$, donde Λ es la matriz diagonal conteniendo los *valores eigen*[23] de la matriz $\mathbf{X}*\mathbf{X}^T$ la cual es igual a los *valores eigen*[24] de la matriz $\mathbf{X}^T*\mathbf{X}$)

La descomposición en valores singulares es:

$$\mathbf{X} = \mathbf{P} * \Delta * \mathbf{Q}^T \qquad (3.3)$$

Debe notarse de estas definiciones, que $\mathbf{P}*\mathbf{P}^T = \mathbf{I}$ y $\mathbf{Q}*\mathbf{Q}^T = \mathbf{I}$, donde I es la matriz identidad.

Ahora, es posible retornar a la regla de aprendizaje. Esta dada como una regla de iteraciones. Así, que se podría preguntar si esta regla converge. Podemos demostrar esta convergencia ahora:

$$W^{(0)} = 0$$
$$W^{(1)} = \eta * \mathbf{X} * \mathbf{X}^T = \eta * \mathbf{P}*\Delta*\mathbf{Q}^T * \mathbf{Q}*\Delta*\mathbf{P}^T = \eta * \mathbf{P} * \Lambda * \mathbf{P}^T$$
$$W^{(2)} = \eta * \mathbf{P} * \Lambda * \mathbf{P}^T + \eta(\mathbf{P}*\Delta*\mathbf{Q}^T - \eta*\mathbf{P}*\Lambda*\mathbf{P}^T * \mathbf{P}*\Delta*\mathbf{Q}^T) * \mathbf{Q}*\Delta*\mathbf{P}^T$$
$$W^{(2)} = \eta * \mathbf{P} * \Lambda * \mathbf{P}^T + \eta*\mathbf{P}*\Delta*\mathbf{Q}^T * \mathbf{Q}*\Delta*\mathbf{P}^T - \eta^2*\mathbf{P}*\Lambda*\Delta*\mathbf{Q}^T * \mathbf{Q}*\Delta*\mathbf{P}^T$$
$$W^{(2)} = \eta*\mathbf{P}*\Lambda*\mathbf{P}^T + \eta*\mathbf{P}*\Lambda*\mathbf{P}^T - \eta^2*\mathbf{P}*\Lambda^2*\mathbf{P}^T$$
$$W^{(2)} = \mathbf{P} * (\eta*\Lambda + \eta*\Lambda - \eta^2*\Lambda^2) * \mathbf{P}^T$$
$$W^{(2)} = \mathbf{P} * (\mathbf{I} - (\mathbf{I} - \eta*\Lambda)^2) *\mathbf{P}^T$$

Así, en el estado t:

$W^{(t)} = \mathbf{P} * (\mathbf{I} - (\mathbf{I} - \eta*\Lambda)^t) *\mathbf{P}^T$

Y la MAA converge sólo si:

$$\lim_{t\to\infty}(I - \eta\cdot\Lambda)^t = 0 \qquad (3.4)$$

---

[22] Anexo 12
[23] Anexo 6
[24] Anexo 6

Esto es si $0 < \eta < \frac{2}{\lambda_{max}}$, donde λ_{max} es el mayor valor eigen de $\mathbf{X}^*\mathbf{X}^T$.

Como una consecuencia, cuando se ha realizado el proceso de aprendizaje, la matriz de pesos tiene la siguiente forma: $W = \mathbf{P}^*\mathbf{P}^T$

Como ya se ha dicho, el objetivo de una MAA es la reconstrucción de los rostros memorizados. Entonces ahora que la matriz de pesos ha sido calculada, la salida de la MAA es:

$$\begin{aligned} \mathbf{O} &= W * X \\ &= P*P^T * X \\ &= P*P^T * P*\Delta*Q^T \\ &= P * \Delta * Q^T \\ &= X \end{aligned}$$

y la reconstrucción es perfecta.

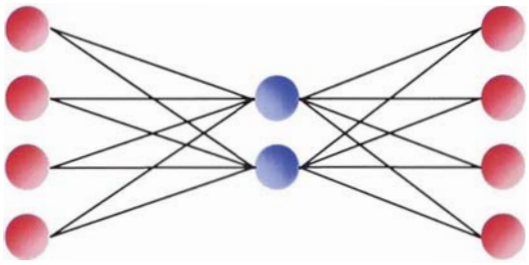
3.4.3 Componentes Principales

Como se ha visto, un rostro que ha sido memorizado por la MAA, cuando se le presenta como input, con ruido adicionado (ejemplo: fondo distinto), puede ser perfectamente reconstruido. Este proceso de reconstrucción es llevado a cabo linealmente por la matriz W. Se debe notar que esta matriz esta formada por dos partes: $W = \mathbf{P}^*\mathbf{P}^T$. Así, el proceso de reconstrucción se completa en dos partes:

$$\mathbf{y} = \mathbf{P}^T * \mathbf{x} \quad \text{y} \quad \mathbf{o} = \mathbf{P} * \mathbf{y} \qquad (3.5)$$

Aquí, **x** es un rostro (vector columna), **o** es el rostro reconstruido e **y** es la forma intermedia.

La arquitectura de la red neural que provee esta forma intermedia se muestra en la figura 3.5



Capa Entrada	**Capa Invisible**	**Capa Salida**
I neuronas	grado($\mathbf{X}^*\mathbf{X}^T$) neuronas	I neuronas

Fig. 3.5: Arquitectura de red Neural representando la forma intermedia

La matriz **P** transforma un rostro representado en el espacio de imagen **x**, en un rostro descrito en el espacio de rostros **y**. Los vectores columna de **P** son las bases de este espacio de rostro, y son llamados los *componentes principales*. Debido al método de construcción de la matriz **P**, sus vectores son ortogonales y pueden ser normalizados. Así el espacio del rostro esta descrito por bases ortonormales. Estas bases proveen un buen conjunto de características para el conjunto de rostros.

En este escenario, algunas consideraciones deben de ser dadas sobre las dimensiones de los vectores y las matrices envueltas en este proceso:

dim(**x**) = dim(**o**) = (**w***__h__) x 1.

dim(**P**) = (**w***__h__) x grado($\mathbf{X}^*\mathbf{X}^T$), y grado($\mathbf{X}^*\mathbf{X}^T$) = min(**w***__h__, **K**), **K** es el número de rostros memorizados.

$$\dim(\mathbf{y}) = \text{grado}(\mathbf{X}^*\mathbf{X}^T) \times 1.$$

Frecuentemente, como sucede en el experimento presente, **w*h** es más grande que **K**. La Base de Datos para el conjunto de entrenamiento, usada para este trabajo, contiene 300 rostros, cada rostro mide 64x64 píxeles. En esta estructura, un rostro codificado mediante los componentes principales tiene menos componentes que la de su forma original descrita en píxeles. Esto es debido a una reducción dimensional. El factor de reducción es grande; en este ejemplo, el rostro original esta compuesta por 4096 píxeles y la forma intermedia, **y**, esta compuesta por 300 componentes. Este factor de reducción puede ser favorablemente incrementado como se mostrará más abajo.

Esta reducción dimensional puede ser ejecutada debido a que el rostro original ha sido proyectada sobre una nueva base. Esta base esta dada por los vectores columna de la matriz **P**. Esta es la mejor base que puede encontrarse para describir los rostros para entrenamiento. Está base forma un nuevo espacio llamado espacio rostro.

Como ejemplo, la figura 3.6 muestra los primeros componentes principales.

Fig. 3.6: Primeros Componentes Principales

3.4.4 Conclusión

Encontramos que la descripción del ACP en el paradigma de las redes neurales y la memoria lineal autoasociativa es interesante por muchas razones. Provee un sentido intuitivo para los componentes principales, y muestra claramente como podría implementarse el ACP. Sin embargo, el paradigma estadístico es más riguroso matemáticamente y nos permite obtener una mayor reducción dimensional, como se mostrará en la sección subsiguiente.

3.5 ACP Estadísticamente

3.5.1 Matriz de Transformadas

El espacio de imagen es altamente redundante cuando se usa para describir rostros. Así, como ha sido previamente demostrado, se debería de construir un espacio de rostros. Esta redundancia es debida a la característica de que cada píxel en un rostro está altamente correlacionada a los otros píxeles. De hecho, la matriz de covarianzas, Σ, para un conjunto de rostros es altamente no diagonal. Tomando lo previo obtenemos:

$$\Sigma_{\mathbf{X}} = \mathbf{X}^{*}\mathbf{X}^{T} = \begin{vmatrix} \sigma^{X}_{11} & \sigma^{X}_{12} & \cdots & \sigma^{X}_{1,w*h} \\ \sigma^{X}_{21} & \sigma^{X}_{22} & \cdots & \sigma^{X}_{2,w*h} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma^{X}_{w*h,1} & \sigma^{X}_{w*h,2} & \cdots & \sigma^{X}_{w*h,w*h} \end{vmatrix} \qquad (3.6)$$

σ_{ij} representa la covarianza entre el píxel i y el píxel j. Existe una relación entre el coeficiente de covarianza y el coeficiente de correlación:

$$r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii} \cdot \sigma_{jj}}} \qquad (3.7)$$

El coeficiente de correlación es un coeficiente de covarianza normalizado.

El objetivo es la construcción de un espacio del rostro, donde cada componente este no correlacionado con cualquier otro componente. Esto significa que la matriz de covarianza de los nuevos componentes debería ser diagonal.

$$\Sigma_Y = \mathbf{Y}^*\mathbf{Y}^T = \begin{bmatrix} \sigma_{11}^Y & 0 & \dots & 0 \\ 0 & \sigma_{22}^Y & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma_{w^*h,w^*h}^Y \end{bmatrix} \quad (3.8)$$

donde:

- $\mathbf{y}_i$ es el vector columna que describe el rostro $\mathbf{x}_i$ en el espacio básico del rostro, los principales componentes.
- **X** es la matriz conteniendo los rostros(en el espacio básico de imágenes), $\mathbf{x}_i$, y
- **Y** es la matriz conteniendo los vectores $\mathbf{y}_i$.

Esta forma diagonal de la matriz de covarianza implica que la varianza de una variable con ella misma sea maximizada mientras que la covarianza de la variable con cualquier otra variable se hará cero. Las variables no volverán ha ser correlacionadas. Así, esta construcción encuentra aquellas direcciones que maximizan la varianza.

Como ya se vio, los componentes principales son calculados linealmente. Sea **P** la matriz de transformadas:

$$\mathbf{Y} = \mathbf{P}^T * \mathbf{X} \text{ y } \mathbf{X} = \mathbf{P} * \mathbf{Y}$$

De hecho $\mathbf{P} = \mathbf{P}^{-1}$, porque las **P**'s columnas son ortonormales entre sí: $\mathbf{P}^T\mathbf{*P} = \mathbf{I}$

Ahora, la pregunta es, que debería ser **P**; dada la condición que Σ_Y tiene que ser una matriz diagonal.

$$\Sigma_Y = \mathbf{Y} * \mathbf{Y}^T = \mathbf{P}^T * \mathbf{X} * \mathbf{X}^T * \mathbf{P}$$
$$= \mathbf{P}^T * \Sigma_X * \mathbf{P}$$

Así Σ_Y es la rotación de Σ_X en **P**.

Definamos **P** como la matriz que contiene los *eigenvectors* de Σ_X

$$\Sigma_X * \mathbf{P} = \Lambda * \mathbf{P}$$

donde Λ es la matriz diagonal que contiene los *valores eigen*[25] de Σ_X. Así,

$$\Sigma_Y = \mathbf{P}^T * \Lambda * \mathbf{P} = \Lambda * \mathbf{P}^T * \mathbf{P} = \Lambda$$

y Σ_Y es la matriz diagonal que contiene los *valores eigen*[26] de Σ_X. Desde que los elementos de la diagonal de Σ_Y son la varianza de los componentes de los rostros de aprendizaje en el espacio de rostros, los *valores eigen*[27] de Σ_X son aquellas varianzas. El entendimiento completo del significado de los elementos de Λ es importante para comprender la reducción de los componentes que será discutida a continuación.

3.5.2 Reducción Dimensional

El propósito de ACP es reducir la dimensión del espacio de trabajo. El número máximo de los componentes principales es el número de variables en el espacio original. No obstante, con el fin de reducir la dimensión, algunos componentes principales se deben de omitir.

Obviamente, la dimensionalidad del espacio de rostros es menor que el espacio de imágenes:

---

[25] Anexo 6
[26] Anexo 6
[27] Anexo 6

$$\dim(\mathbf{Y}) = \text{col}(\mathbf{P}) \text{ x } \text{col}(\mathbf{X}) = \text{rango}(\mathbf{X}*\mathbf{X}^T) \text{ x } \mathbf{K},$$
$$\dim(\mathbf{y}_i) = \text{rango}(\mathbf{X}*\mathbf{X}^T) \text{ x } 1$$

Como se menciono previamente, el rango($\mathbf{X}*\mathbf{X}^T$) es generalmente igual a K. Por lo tanto se ha realizado una reducción de dimensión, y la información almacenada por $\mathbf{y}_i$ es absolutamente la misma que la información almacenada en el rostro original $\mathbf{x}_i$. Se puede hacer una reducción adicional de dimensión. La pregunta es, cual es la dimensión apropiada para el espacio de rostros. Ciertamente, este no puede perfectamente ser conocido. Sin embargo, no hay nada que sugiera que la dimensión del espacio del rostro tiene que ser el número de los rostros memorizados, como ha sido descrita hasta aquí.

El cerebro humano es muy bueno para el reconocimiento facial. Aunque la tarea sea muy difícil, el cerebro puede identificar un rostro muy rápidamente. Esta velocidad podría implicar que la dimensionalidad del espacio del rostro no es más que unas pocas decenas. Esto se confirma en la práctica, como se demostrara en el tercer capítulo.

Ahora, supongamos que la dimensión del espacio del rostro es pequeña, ¿Qué vectores básicos tienen que estar presentes y cuáles tienen que ser desechados?.

La reconstrucción de un rostro $\mathbf{x}_i$ mediante los componentes $\mathbf{y}_i$ está dada por la siguiente formula:

$$\mathbf{x}_i = \mathbf{P} * \mathbf{y}_i \qquad (3.9)$$

Si los $\mathbf{p}_i$ son los vectores columna de $\mathbf{P}$, esto es, los vectores básicos, la reconstrucción de un rostro, usando sólo algunos de los componentes $\mathbf{y}_{ij}$ es:

$$\overline{x_i} = \sum_{j=1}^{M} y_{ij} \cdot p_j + \sum_{j=M+1}^{K} y_{ij} \cdot p_j$$

$$= \sum_{j=1}^{M} y_{ij} \cdot p_j + \sum_{j=M+1}^{K} a_j \cdot p_j$$ donde $\mathbf{a}_j$ es una constante cuyo valor será determinado

Esto introduce un error. El vector error es:

$$e = x_i - \overline{x_i}$$

$$= \sum_{j=M+1}^{K} (y_{ij} - a_j) \cdot p_j$$

El vector error es un vector random y su efectividad es calculada por el valor de su mínimo cuadrado:

$$\langle e^2 \rangle = \mu(e^T * e)$$

$$= \mu\left(\sum_{j=M+1}^{K} \sum_{l=M+1}^{K} (y_{ij} - a_j) \cdot (y_{il} - a_l) \right) \cdot p_i^T \cdot p_l$$

$$= \sum_{i=M+1}^{K} \mu\left[(y_i - a_i)^2 \right]$$

El objetivo es minimizar el error. Esto se hace, resolviendo:

$$\frac{\partial}{\partial a_i} \langle \mathbf{e}^2 \rangle = 0$$ esto es

$$-2 * \mu((\mathbf{y}_i - \mathbf{a}_i)) = 0$$ así,

$$\boldsymbol{a}_i = \mu(\mathbf{y}_i)$$ para $M+1 \leq i \leq K$

Esto significa que los valores omitidos de $\mathbf{y}_i$ tienen que ser reemplazados por sus valores mínimos para poder minimizar el error de reconstrucción. El vector error se transforma en:

$$\langle e^2 \rangle = \sum_{i=M+1}^{K} \mu\left[\left(y_i - \mu(y_i)\right)^2\right] = \text{suma de la varianza de los componentes omitidos}$$

$$= \sum_{i=M+1} \lambda_i$$

Luego, si los *valores eigen*[28] son clasificados en orden decreciente, los últimos valores eigen (y sus *eigenvectors*) podrían ser ignorados. *Este es la real reducción dimensional del ACP.*

En efecto esto significa que algunos componentes principales pueden ser ignorados porque ellos sólo explican una baja cantidad de información, mientras que la mayor cantidad de información esta contenida en los otros componentes principales. La cantidad de información que el i^{esimo} componente principal tiene esta dada por su *valor eigen*, λ_i. Usualmente los componentes principales (*eigenvectors*) están ordenados en el orden decreciente de sus *valores eigen*[29] y los últimos son ignorados. Este hecho es particularmente importante ya que cuando tomamos conocimiento de más rostros, el decrecimiento de los *valores eigen*[30] es exponencial.

3.5.3 Número de Dimensiones Requerida

Como se ha dicho previamente, el número de dimensiones del espacio de rostros no puede ser determinada. Sin embargo, una prueba simple puede darnos una buena indicación del número de dimensiones relevantes o importantes. Se ha venido resaltando que los *valores eigen*[31] decrecen de una

---

[28] Anexo 6
[29] Anexo 6
[30] Anexo 6
[31] Anexo 6

manera exponencial. Cuando se considera el final de este proceso de decrecimiento, todos los *valores eigen*[32] son iguales:

$$\lambda_N = \lambda_{N+1} = \lambda_{N+2} = \ldots = \lambda_K, \text{ para } 1 < N < K \qquad (3.10)$$

Se puede decir que los principales componentes asociados con estos *valores eigen*[33] son arbitrarios, y por consiguiente deberían de ser descartados. Así, N será la dimensión del espacio de rostros.

3.5.4 Ejemplo Gráfico

Quizás sea más fácil entender el ACP mediante un ejemplo gráfico que mediante ecuaciones matemáticas. Como ejemplo tenemos un proceso random que produce un resultado bi dimensional (x1,x2). Se ha realizado un gran número de experimentos de este proceso, y los resultados obtenidos se muestran en la figura 3.7:

[32] Anexo 6
[33] Anexo 6

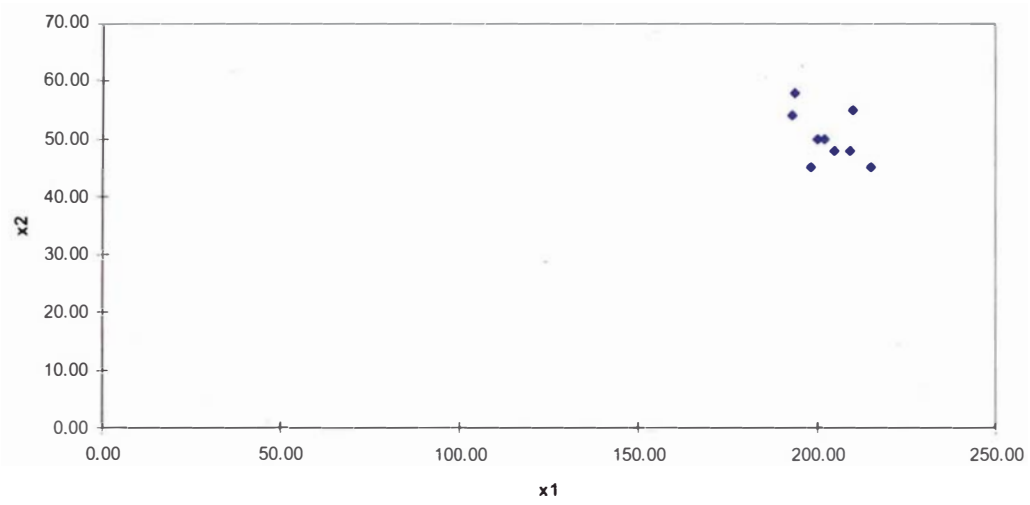


Fig. 3.7: Proceso random en sus coordenadas originales, x_1 y x_2

Es claro, que en este proceso random, x_1 está correlacionado con x_2. De la imagen se deduce que ejes diferentes a x_1 y x_2 serían más convenientes para describir este proceso. El propósito del ACP es buscar los ejes que maximicen la varianza de la data. Estos ejes se muestran en la figura 3.8. Ellos son una característica del proceso, y por consiguiente, son llamados *ejes característicos*.

Se debe de notar que, como se dijo en la parte teórica, los ejes característicos son ortogonales.

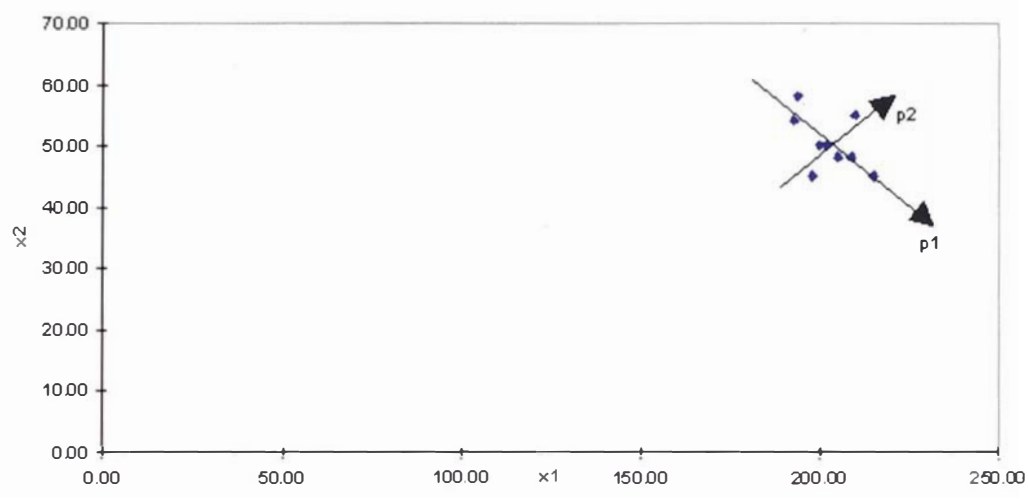


Fig. 3.8: Proceso random y sus ejes característicos

Es obvio, a partir del gráfico, que la varianza de la data es máxima en la dirección p1. La dirección p1 maximiza la variación de la proyección de los puntos. La dirección que produce la siguiente mayor varianza de la data, a condición de que sea ortogonal a p1, es p2. Esto puede apreciarse mejor cuando observamos la cantidad de data (los pesos) en cada una de las direcciones p1 y p2. La dispersión de la data es la más extensa en la dirección de p1, y la siguiente dispersión más extensa se encuentra en la dirección de p2.

λ_1, el valor eigen del *eigenvector* p1 es 55, y λ_2, el respectivo de p2 es 7. Así, el 89% de la variación de la data está explicada por p1, y sólo el 11% está explicada por p2. Esto significa que p1 captura la mayor parte de la variación (esto es, las distancias entre las observaciones) en el espacio bi dimensional original. Por lo tanto, dependiendo de la siguiente etapa del proceso, la dimensión menos importante, p2, podría ser suprimida. Desde que esta

reducción de dimensión suprime información, está sólo debería de hacerse si la información no es relevante para la siguiente etapa del proceso.

Los componentes principales son combinaciones lineales de las variables originales:

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0.94 & 0.33 \\ -0.33 & 0.94 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

La cuál es, efectivamente, una rotación en 20º de los ejes originales.

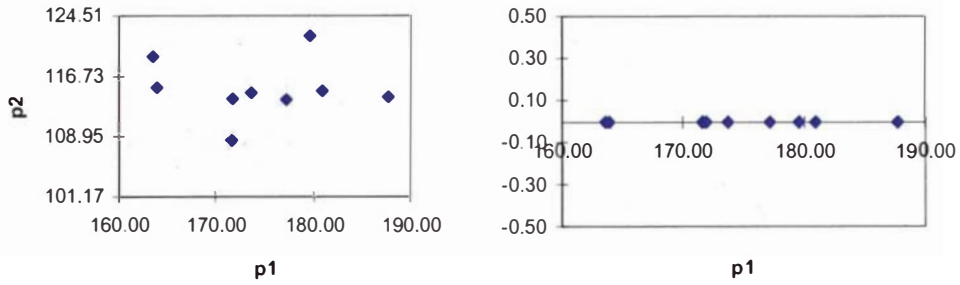


Fig. 3.9: En la izquierda, la data random en el espacio característico compuesto por dos características. En la derecha la data esta representada sólo por la primera característica

Otra propiedad del primer componente principal, p1, es que él minimiza la suma de los cuadrados de las distancias, de las observaciones, de sus proyecciones perpendiculares sobre él más extenso componente principal. Por esta razón, el primer componente principal es frecuentemente referenciado o llamado como *la línea de tendencia más cercana*.

Debería también notarse que la variación de los puntos en la dirección del segundo componente principal, p2, es menor que la variación de los puntos en la dirección del mayor componente principal, p1.

CAPITULO IV

EXPERIMENTOS USANDO EL ANALISIS DE LOS COMPONENTES PRINCIPALES

4.1 Introducción

El objetivo en este capítulo es experimentar con el Análisis de los Principales Componentes (ACP) como un medio de reconocimiento facial. Para facilitar estos experimentos, se ha desarrollado un programa que ejecute el ACP. Este programa será descrito en el sexto capítulo.

En la introducción se hizo notar que el llamado reconocimiento facial es un término extenso el cuál será ampliado en los siguientes términos:

- **Identificación**, donde se deben obtener datos individuales,
- **Reconocimiento** de una persona, donde se debe de decidir si los datos individuales ya han sido obtenidos con anterioridad y,
- **Toma de Decisión**, donde el rostro tiene que ser asignado ha cierta categoría o grupo. Aquí, se examinarán dos de estas tareas,

identificación y categorización. En este caso la categorización que se estudiara es la de sexo.

Obviamente, para probar el sistema se requieren algunos rostros. Todos los experimentos descritos aquí han sido ejecutados sobre los rostros proveídos por una base de datos facial. La siguiente sección describe esta base de datos.

4.2 Base de Datos Facial

Se uso imágenes estáticas de rostros de treinta individuos diferentes. Se dividieron estás imágenes en dos grupos: un grupo de entrenamiento y otro grupo de prueba. 10 imágenes de cada persona se han incluido en cada grupo, por lo tanto existen 300 imágenes en el grupo de entrenamiento y 300 imágenes en el grupo de prueba. La mayor parte de las personas de está base de datos facial se encuentran entre 25 y 35 años. Con la excepción de una mujer joven de raza negra y un hombre chino de edad mayor, la base de datos esta compuesta de caucasianos adultos.

Hay sólo 7 mujeres contra 23 hombres. Esto planteará el problema de clasificación de sexo, porque la probabilidad de que una persona sea hombre es muy diferente de la obtenida para una mujer bajo el universo actual. Por consiguiente, en la clasificación de sexo 16 hombres serán eliminados de la base de datos facial con la finalidad de igualar estas probabilidades.

Una representación gráfica de la base de datos permitirá entenderla mejor. Un sub conjunto de rostros de cada una de los grupos de la base de datos se muestra en las figuras 4.1 y 4.2.

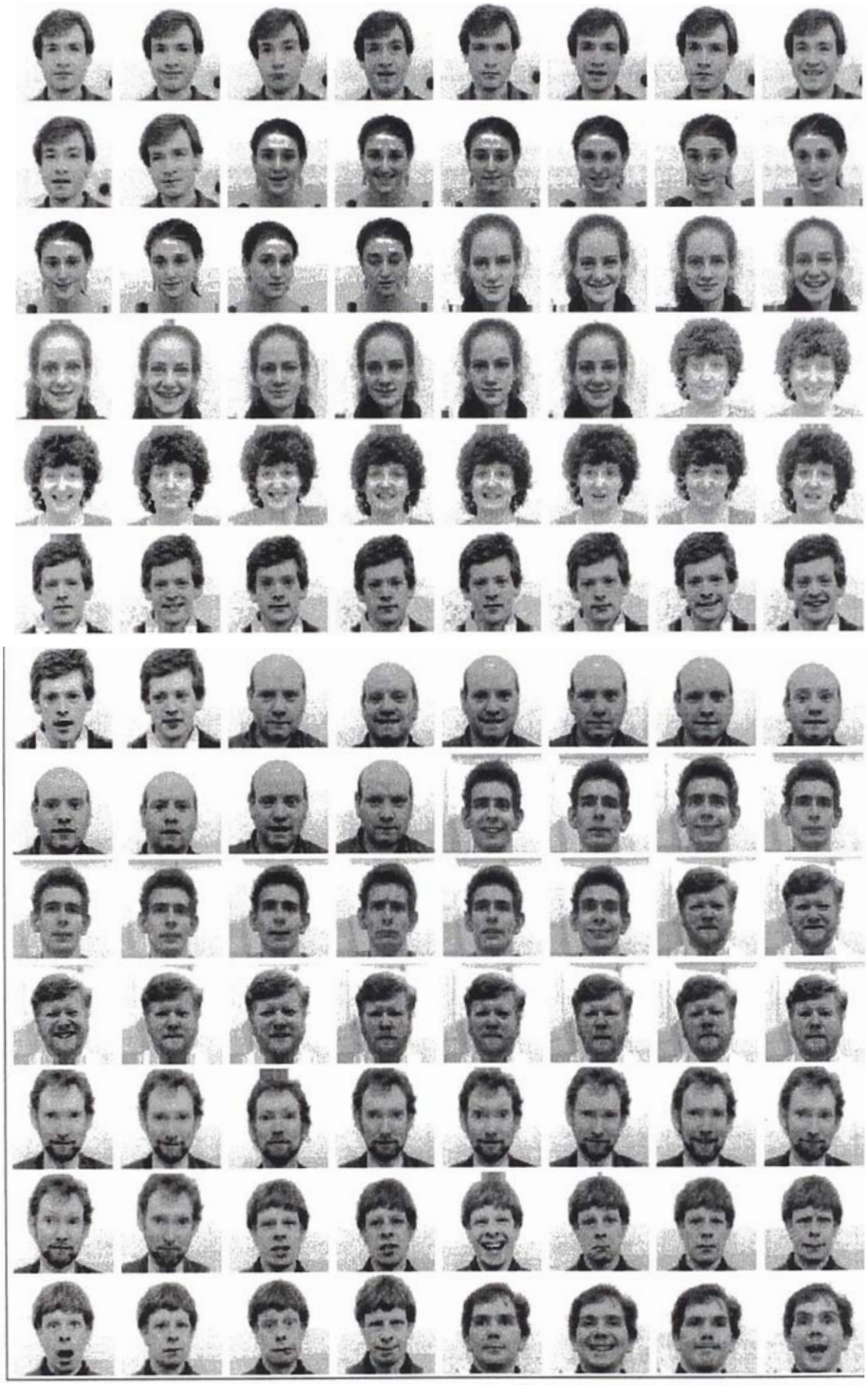


Fig. 4.1: Parte del conjunto de entrenamiento de la base de datos facial

Fig. 4.2: Parte del conjunto de prueba de la base de datos facial

Teniendo en cuenta la figura 4.1 (que representa el conjunto de entrenamiento de la base de datos facial) y la figura 4.2 (que representa el conjunto de prueba), se observa una diferencia la cuál es importante para los experimentos. Los rostros en el conjunto de entrenamiento están de alguna manera normalizados. De hecho, están aproximadamente centrados, todos a la misma escala, con un fondo, aproximadamente, equivalente para cada figura. Cada expresión de los individuos es aproximadamente la misma, con el cabello de las damas recogido hacia atrás (cuando tienen cabello largo).

En el conjunto de prueba los rostros no están normalizados del todo. No están centrados, exhiben una amplia variedad de escalas e incluyen una gran variación en el fondo. Las expresiones faciales son variadas (a veces sonrientes, a veces tristes, etc.), el cabello de las damas no están necesariamente arreglados y algunas imágenes faciales de las personas presentan anteojos.

Sin embargo, si el conjunto de entrenamiento es usado para entrenar al sistema ACP y el conjunto de prueba para probarlo, los resultados obtenidos son muy pobres, se obtuvo sólo el 40% de identificación correcta.

Como podría esperarse, el ACP produce muy malos resultados usando esta configuración.

De hecho el ACP usa el conjunto de entrenamiento para construir una representación del espacio facial. Sin embargo está representación no caracteriza el espacio facial completo. Sólo representa la parte de ella que tiene los rostros normalizados. Además, el conjunto de prueba representa otra

región, ortogonal, del espacio facial. Por lo tanto, esta configuración no es buena para el ACP. Se hace necesaria la selección de otra configuración distinta para el conjunto de entrenamiento y el conjunto de prueba.

Cada imagen facial es de 64 por 64 píxeles y cada píxel está codificado en 8 bits (256 niveles de grises).

4.3 Eigenfaces

Antes de considerar los resultados de los experimentos, primero se debe de entender el proceso por los cuales han sido obtenidos. Con esta finalidad, las siguientes secciones cubrirán los aspectos más prácticos de la forma en que se usa el ACP.

4.3.1 Introducción

Como se ha mencionado, el ACP calcula las bases de un espacio el cuál esta representado por sus vectores de entrenamiento. Los vectores básicos calculados por el ACP se encuentran en la dirección de la mayor varianza de los vectores de entrenamiento. Estos vectores básicos están calculados mediante la solución de un *problema eigen*, y por consiguiente los vectores básicos son *eigenvectors*. Estos *eigenvectors* están definidos en el espacio de imágenes. Pueden ser vistos como imágenes y ciertamente son rostros y por lo tanto son usualmente referidos como *eigenfaces*.

Fig. 4.3: eigenfaces iniciales

El primer *eigenface* es el llamado rostro promedio, mientras que el resto de los *eigenface* representan variaciones a partir de este rostro promedio.

El primer *eigenface* es un buen filtro: cada rostro existente en el espacio facial multiplicado píxel por píxel (producto interno) con el rostro promedio produce un número cercano a uno – con imágenes diferentes a rostros el producto interno es mucho menor que uno. La siguiente dirección de la mayor variación (partiendo del promedio) de los vectores de entrenamiento está descrita por el segundo *eigenface*. La dirección de la siguiente mayor variación (partiendo del promedio) de los vectores de entrenamiento está descrita por el tercer *eigenface*, y así sucesivamente.

Cada *eigenface* puede ser visto como una característica. Cuando un rostro en particular es proyectado dentro del espacio facial, su vector, compensado por sus pesos con respecto a cada *eigenface*, en el espacio facial, muestran la importancia de cada una de aquellas características en el rostro. La figura 4.4 describe este proceso gráficamente.

Fig. 4.4: Un rostro desarrollado en el espacio facial

En este ejemplo, vemos el desarrollo de un rostro dentro del espacio facial. El rostro está descrito en el espacio facial mediante sus coeficientes (o pesos) de sus *eigenface* . Como la imagen desarrollada en el espacio facial es un rostro, el peso del primer *eigenface* es muy alto, casi igual a la unidad. (Está propiedad puede ser usada para identificar rápidamente una imagen como un rostro). El valor de los pesos decrece al crecer el número de *eigenfaces*, por la definición de *eigenface*. En efecto, el ACP, encuentra las direcciones de las variaciones mayores. El primer *eigenface* explica la mayor variación, el segundo explica la segunda mayor variación, etc.

4.3.2 Proceso de Generación

La figura 4.5 muestra esquemáticamente el proceso del ACP. Toma los rostros de entrenamiento como datos de entrada y produce los *eigenfaces* en la salida. Luego de esto se pueden ejecutar los procesos de identificación o categorización.

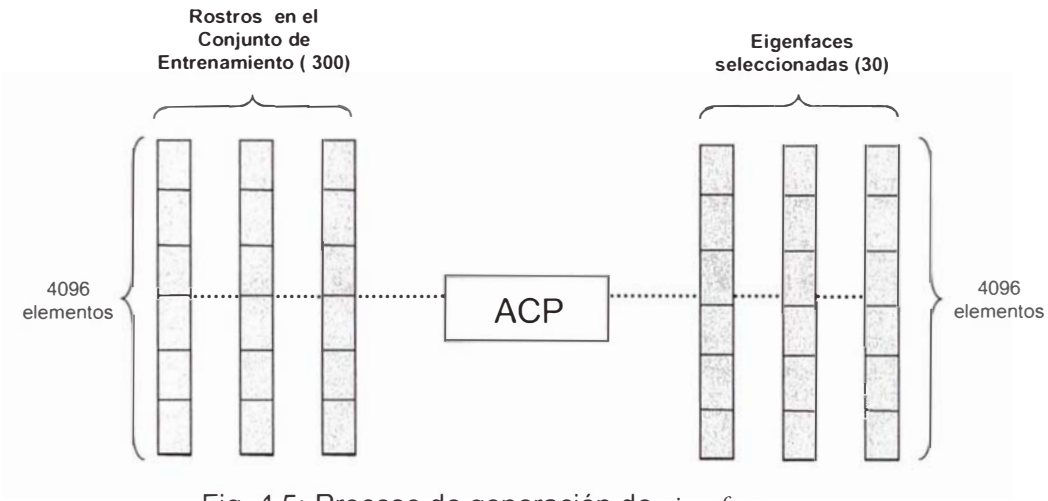


Fig. 4.5: Proceso de generación de *eigenfaces*

4.4 Reconstrucción

Una vez que los *eigenfaces* han sido calculados, cada rostro en el espacio de imágenes puede ser visto en el espacio facial. La transformación desde el espacio de imágenes hacia el espacio facial es bastante simple.

Definamos:

- Sea *E* la matriz de los primeros *eigenfaces*, donde la primera columna es el primer *eigenface* y así sucesivamente,
- Sea f_I un rostro en el espacio de imágenes, y
- Sea f_F el mismo rostro en el espacio facial.

$$\mathbf{f}_F = \mathbf{f}_I * \mathbf{E}^T \qquad (4.1)$$

Esta es una transformación de muchos a uno, desde que la dimensionalidad del espacio de imágenes es de lejos mucho mayor que la dimensionalidad del espacio facial. De esta manera, la transformación introduce un error el cuál se puede ver observando el rostro reconstruido. La reconstrucción se realiza tomando la transformada inversa de 4.1:

$$\mathbf{f}_I = \mathbf{f}_F * \mathbf{E} \qquad (4.2)$$

Como un ejemplo, la figura 4.6 muestra la reconstrucción de un rostro.

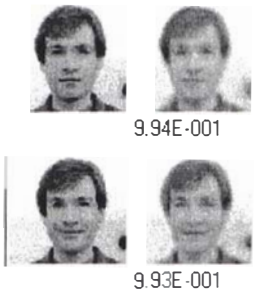


Fig. 4.6: Rostro original junto con su reconstrucción, el error de reconstrucción se muestra debajo del rostro reconstruido.

Nótese lo borroso de la imagen reconstruida. Esto puede ser explicado de una manera simple. El ACP coloca cada rostro del conjunto de prueba sobre otro, y observa las variaciones. Si el rostro no esta centrado o no se encuentra en la misma escala, la representación de los rostros será borrosa. En la base de datos facial, usada para estos experimentos, los rostros no están centrados o no están en la misma escala.

Además la mitad de la imagen pertenece al fondo, y para el ACP, un píxel que pertenece al fondo tiene la misma importancia que un píxel que pertenece a la parte útil de la imagen, o sea el rostro.

La magnitud de la diferencia entre el rostro original y su reconstrucción, llamado error de reconstrucción, se calcula fácilmente. Sea f'_l el rostro reconstruido del rostro original f_l , y ε el error de reconstrucción:

$$\mathbf{f'}_l = \mathbf{E}^T * \mathbf{E} * \mathbf{f}_l \text{ y } \varepsilon = |\mathbf{f}_l - \mathbf{f'}_l| \qquad (4.3)$$

Como los vectores están normalizados, podemos usar la distancia del coseno en vez de la distancia Euclidiana:

$$\varepsilon = 1 - \cos(\mathbf{f'}_l , \mathbf{f}^T_l)$$

$$\varepsilon = 1 - \mathbf{f'}_l * \mathbf{f}^T_l \qquad \text{porque } \mathbf{f'}_l \text{ y } \mathbf{f}^T_l \textit{ están normalizados}$$

4.5 Conjunto de Entrenamiento / Prueba

La base de datos facial usada no es la más apropiada para ser usada con el ACP debido a dos factores principales:

- Primero como se considero en este capítulo, ninguno de los rostros están centrados y no se encuentran en la misma escala dentro del conjunto de prueba, causando un efecto de desenfoque.
- Segundo, el conjunto de entrenamiento no representa el espacio facial completo; representa sólo a los rostros que están normalizados – como ya se ha dicho en este capítulo.

Parece no haber una solución consistente para el primer problema. Un programa podría traducir los rostros (con la ayuda del usuario) de tal manera que *la nariz* quede en el centro de la imagen. Sin embargo el problema de la escala permanecería sin solución.

Podría ser necesaria la reconversión de la imagen antes de ejecutar el ACP. Se debe de colocar cada rostro sobre el límite de una superficie elíptica. Los resultados obtenidos por medio de este método son muy buenos y el efecto de desenfoque es más o menos evitado. *Poggio*[34] usa un algoritmo de flujo óptico

[34] *Dr. Tomaso Poggio*, profesor de Ciencias de la Visión y Biofísica en el Departamento de Ciencia Cognoscitivas y de la Mente en el MIT desde 1993, http://www.ai.mit.edu./people/poggio.

para igualar los rostros gradualmente. Sin embargo este algoritmo requiere las vistas frontal y de lado de un rostro en particular.

Sin embargo, como se podrá ver en los resultados finales mostrados más tarde, el efecto de este problema en la base de datos facial, aunque perceptible, no es lo suficientemente serio para invalidar el ejercicio completo.

El segundo problema podría ser manejado con la ayuda de la técnica *Jack-knife*[35]. En la configuración experimental se obtuvo que el 90% de las imágenes deben de conformar el conjunto de entrenamiento y el 10% restante el conjunto de prueba. Las imágenes del conjunto de prueba deben de ser seleccionadas aleatoriamente. El experimento se repitió varias veces, con la finalidad de verificar que el método sea consistente.

[35] Anexo 10

CAPITULO V

TOMA DE DECISION

5.1 Identificación

5.1.1 Proceso de Identificación

Una vez que los *eigenfaces* han sido calculados, el espacio facial debe ser poblado con rostros conocidos. Estos rostros se toman del conjunto de entrenamiento. Cada rostro conocido es transformado dentro del espacio facial y sus componentes son almacenados.

Luego podemos iniciar el proceso de identificación. Un rostro desconocido es presentado al sistema. El sistema lo proyecta dentro del espacio facial y calcula su distancia a partir de todos los rostros almacenados. El rostro se identifica como si fuera el individuo con el rostro que tenga la menor distancia a él (el más cercano) en el espacio facial. Existen muchos métodos para calcular la distancia entre vectores multidimensionales. Aquí se usa una forma de la distancia Euclidiana: el coseno del ángulo entre los dos rostros que son

calculados. Como los rostros están normalizados (magnitud del vector igual a 1), las distancias coseno y euclidiana son idénticas. La ventaja de las distancias del coseno es que pueden ser calculadas fácilmente.

El proceso de identificación se resume en la figura 5.1.

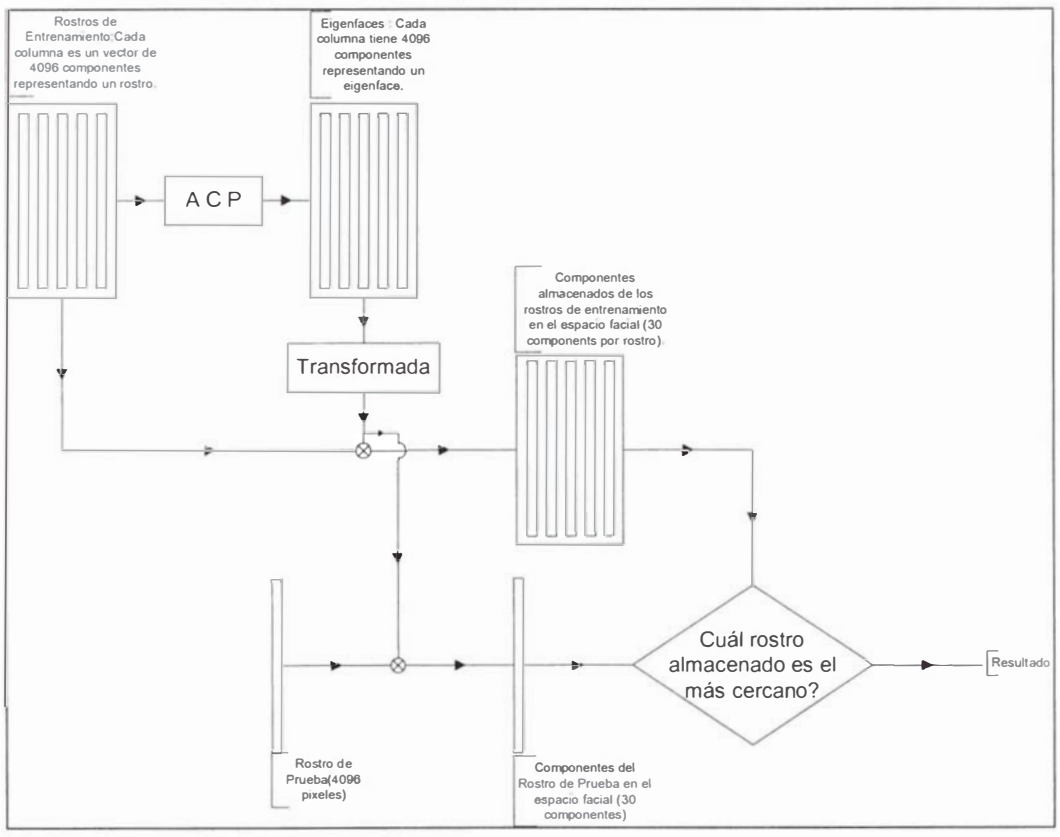


Fig. 5.1: Proceso de Identificación

El algoritmo descrito aquí no es genérico para realizar la identificación facial, porque omite lo siguiente:

1. ¿Qué sucede si la imagen presentada al sistema no es un rostro?. Como la proyección sobre el espacio facial es de varias imágenes sobre una,

puede ocurrir que muchas imágenes que no son rostros se proyecten sobre un vector facial almacenado.

2. ¿Qué sucede si el rostro presentado al sistema aún no ha sido memorizado (esto es no ha sido almacenada como un rostro conocido)?.

El primer defecto puede ser fácilmente evitado. Ya se ha determinado que el primer *eigenface* es un buen filtro facial. Cada imagen que esta altamente correlacionada con ella puede ser definida como un rostro mientras que las imágenes con una bajo factor de correlación pueden ser rechazadas.

El segundo defecto puede ser manejado mediante la adición del algoritmo desarrollado por Turk y Pentland en 1991.

5.1.2 Algoritmo de Reconocimiento Facial de Turk y Pentland

Como se representa un rostro mejor en el espacio facial (por lo explicado anteriormente), la reconstrucción de este debería ser muy similar al original, y el error de reconstrucción debería de ser muy pequeño. Por otro lado, imágenes distintas a rostros deberían de tener un error de reconstrucción grande. Esto implica que el error de reconstrucción de rostros debería estar dentro un cierto umbral, θ.

Cuando un rostro es proyectado sobre el espacio facial, puede encontrarse en cuatro diferentes regiones:

1. Cerca del espacio facial y cerca de un rostro memorizado.
2. Cerca del espacio facial y lejos de un rostro memorizado.
3. Lejos del espacio facial y cerca de un rostro memorizado.

4. Lejos del espacio facial y lejos de un rostro memorizado.

Estas posibilidades están representados en la figura 5.2 y en la tabla 5.1

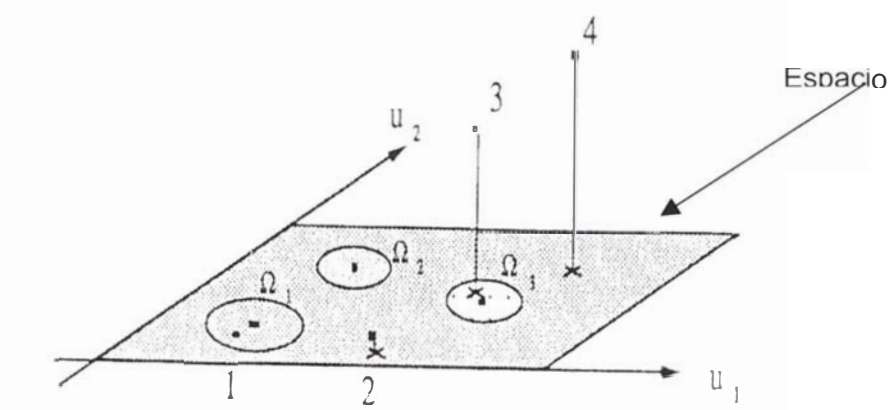


Fig. 5.2: Versión simplificada del espacio facial ilustrando los cuatro resultados de la proyección de una imagen sobre el espacio facial. En este caso hay dos *eigenfaces*, u1 y u2.

	Sobre el Espacio Facial	Rostro Memorizados	Resultado
1	Cerca	Cerca	Reconocido como Ω_1
2	Cerca	Lejos	Rostro desconocido
3	Lejos	Cerca	No es un rostro
4	Lejos	Lejos	No es un rostro

Tabla 5.1: Ilustra los cuatro resultados de la proyección de una imagen sobre el espacio facial

5.2 Clasificación de Sexo

5.2.1 Introducción

Los sicólogos piensan que el espacio facial esta ordenado. La clasificación del género es un buen ejemplo de un problema cuya solución incide sobre los sorteos del orden en que deberían de aparecer en el espacio facial.

Efectivamente, la clasificación del sexo es uno de los procesos biológicos más importantes y probablemente el más fácil y el más rápido de realizar por los humanos. Bruce (1993) escribió que los observadores humanos tienen la habilidad de clasificar fotografías de rostros no familiares con su respectivo sexo con un 96% de precisión. Sin embargo, este problema esta muy lejos de ser resuelto por programas de computador, puesto que aún no es entendida que información visual usan los humanos para decidir si un rostro es masculino o femenino.

Existen dos categorías de representación de la información visual:

- Basada en la medida
- Basada en la codificación de los píxeles de los rostros.

Los experimentos presentados aquí pertenecen a la segunda categoría.

El proceso de clasificación puede ser subdividido en dos partes:

1. Primero, los píxeles son transformados hacia un espacio facial.
2. Segundo, se aplica un algoritmo de clasificación.

Para el experimento de clasificación de sexo, estos dos procesos serán ejecutados. El objetivo es probar el ACP, y la pregunta que intentaremos

responder es la siguiente: ¿Es el espacio facial una buena representación para los rostros?. Por lo tanto, si fuera así, sólo la primera parte del proceso de clasificación (la representación de la información visual) debería ser probada, ya que la segunda sería una consecuencia.

Con el fin de entender mejor la primera parte (la representación de la información visual), se realizarán dos conjuntos de experimentos que difieren en sus procesos de clasificación. El primer conjunto de experimentos usará el algoritmo *LVQ*[36], y el segundo usará el *Error de Reconstrucción*.

Los mismos conjuntos de entrenamiento / prueba usados en los experimentos de identificación serán usados aquí para el proceso de clasificación de sexo. Sin embargo, estos conjuntos contienen imágenes de 7 mujeres y 23 hombres. Con el objetivo de obtener una probabilidad igual de hombres y mujeres, 16 hombres han sido eliminados de los conjuntos. Esto *significa*[37] que el conjunto de entrenamiento contiene (7+7)*10*2 = 280*0.9 = 252 rostros, y el conjunto de prueba contiene (7+7)*10*2 = 280*0.1 = 28 rostros

5.2.2 Primer Experimento, usando LVQ

5.2.2.1 Método

El algoritmo de Cuantificación del Vector Lineal (LVQ por sus siglas en inglés), el cuál se deriva del Mapa Auto Organizado de Kohonen, ha sido usado extensivamente en los procesos de clasificación de sexo de los individuos.

[36] Anexo 3
[37] 7m + 7h = 14 sujetos * 10 imágenes / cada uno = 140 rostros para el conjunto de entrenamiento y lo mismo para el conjunto de prueba, es decir 140 rostros más; hacen un total de 280 rostros; por lo definido después de aplicar la técnica *Jack-knife* se tiene una relación de 90% para el conjunto de entrenamiento y el 10% para el conjunto de prueba

El Mapa Auto Organizado de Kohonen está compuesto de celdas neuronales dispuestas en un plano 2D. Estas celdas, llamadas *vectores de códigos*, están específicamente reajustadas para varias clases de patrones ha través de un proceso de aprendizaje supervisado. Las ubicaciones de las respuestas intentan ser ordenadas como si se hubiera creado sobre la red un sistema coordenado para diferentes características de entrada.

El proceso básico de aprendizaje es bastante simple, en cada iteración, un vector de códigos, m_c, se actualiza. Este vector de códigos es el *vector de códigos* más coincidente con el vector de input x:

$$|x - m_c| = \min_i (|x - m_i|) \qquad (5.1)$$

donde m_i son los vectores de códigos.

La actualización de m_c es conseguida mediante la aplicación de una *regla delta*[38] usando tiempos discretos (t=0,1,2,...); los valores óptimos son alcanzados de manera asintótica:

$$m_c(t+1) = m_c(t) + \alpha(t)(x(t) - m_c(t)) \quad \text{y,}$$

$$m_i(t+1) = m_i(t) \quad \text{para } i \neq c$$

siendo $\alpha(t)$ congruente, decreciente para incrementos del coeficiente escalar, $0 < \alpha(t) < 1$. Cerca del valor inicial de $\alpha(t)$, se encuentran algunos de los parámetros LVQ. Pueden ser decrementos directos o inversos en el tiempo.

[38] Anexo 9

Sin embargo, con el objetivo de obtener un mapa ordenado que tenga la misma estructura de los patrones de entrada, un vecindario, N_c, alrededor de la mejor coincidencia del vector de códigos, m_c, tiene que ser actualizado en cada iteración.

El tipo de este vecindario es otro parámetro del algoritmo. Puede tener forma de burbuja o gausiana.

El enfoque de este método es descubrir cuál combinación de parámetros producen el mejor resultado. Este ajuste sólo puede hacerse mediante la experimentación. Los parámetros se listan aquí en orden de importancia:

1. Número de vectores de código usados,
2. Tipo de decremento de la función alfa.
3. Tipo de vecindario.
4. Tipo de topología, estructura de los patrones de entrada.
5. Número de vecinos más cercanos usados en el proceso de clasificación.

Con el conjunto de estos parámetros se tiene que encontrar el mejor proceso de aprendizaje. Aquí también, existen diversas opciones para la inicialización y para el aprendizaje de los vectores de códigos:

1. **Inicialización:** Los vectores de código pueden inicializarse uniformemente (igual número de vectores de código por clase) o proporcionalmente a la densidad de la probabilidad de los vectores de aprendizaje (el número de vectores de código asociados a una clase en particular depende del número de vectores que pertenecen a esa clase).

2. **Aprendizaje:** El aprendizaje puede ser llevado a cabo con el algoritmo *lvq1*[39], con el algoritmo optimizado *lvq1 (olvq1)*[40], con el algoritmo *lvq2*[41], o con el algoritmo *lvq3*[42].

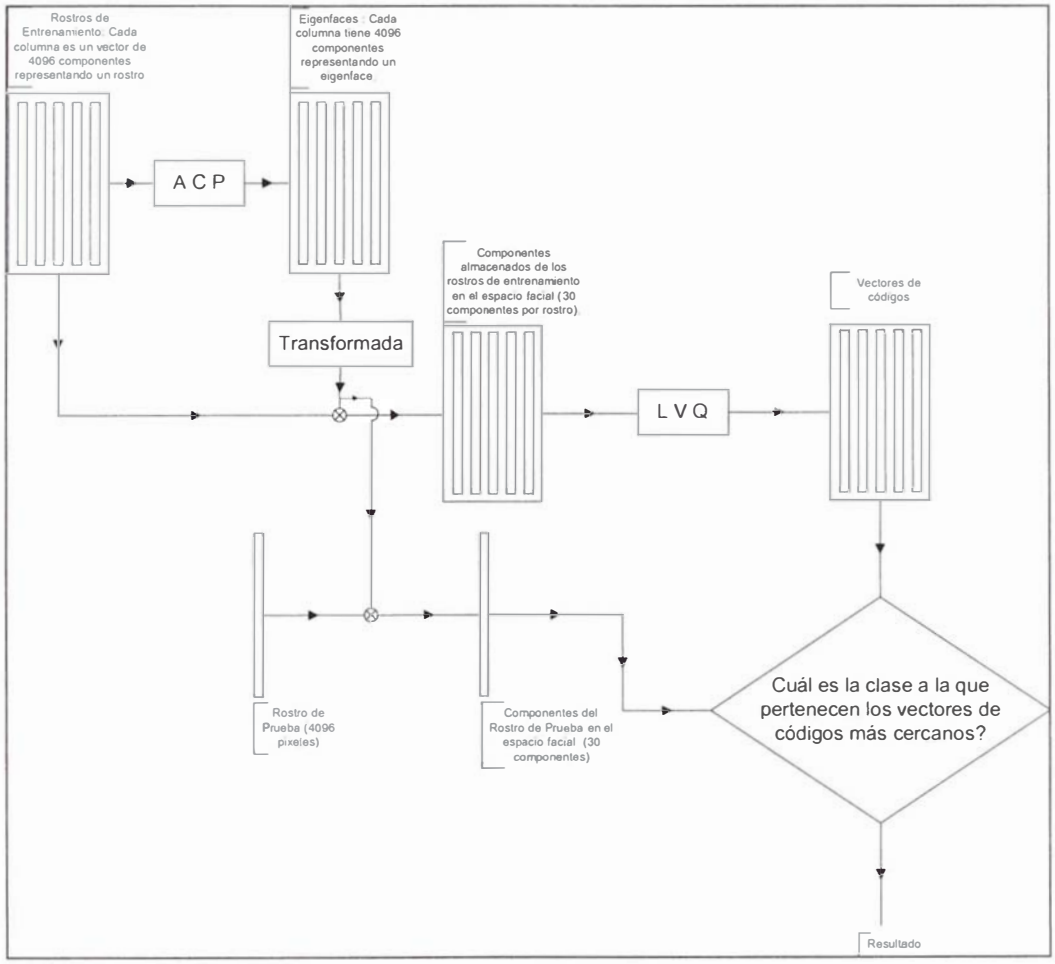


Fig. 5.3: Clasificación con LVQ

[39] Anexo 3
[40] Anexo 3
[41] Anexo 3
[42] Anexo 3

5.2.2.2 Resultados

La combinación de los parámetros que producen los mejores resultados son los siguientes:

- Número de vectores de código usados 50
- Tipo de la función decremento alfa Lineal
- Tipo de vecindario gausiano
- Tipo de topología Hexagonal
- Vecinos más cercanos usados en el proceso de clasificación 5

La inicialización de los vectores de código se ha hecho uniformemente, y el proceso de aprendizaje se realizo así:

1. olvq1 con 20,000 iteraciones.
2. lvq3 durante 150,000 iteraciones con una *longitud de ventana* de 0.1 y un epsilon de 0.3

Esto produce un promedio (para los 10 experimentos) de 81.4% (extremos de 88% y 71%) de clasificación masculina correcta y un promedio de 80.6% de clasificación femenina correcta (extremos de 87% y 69%).

Esto es paradójico, ya que la exactitud de la clasificación es menor que la exactitud de identificación. Esta paradoja se discutirá en la conclusiones.

Un mapeo no lineal, conocido como el *Mapeo Sammon*, se uso para ver si la clase masculina y femenina forman dos grupos linealmente separables. Este mapeo transforma vectores N dimensiónales en superficies 2D con el fin de ver la estructura de la data. Este mapeo no lineal (NLM) se aplico a los vectores de

código. Intenta minimizar el error de mapeo, el cual es igual a la diferencia de la distancia entre los vectores del plano 2D y la distancia entre los vectores en el espacio facial N dimensional. El error de mapeo esta dado por:

$$\varepsilon = \frac{1}{\sum_{i<j} d_{ij}^{*}} \cdot \sum_{i<j}^{L} \frac{\left(d_{ij}^{*} - d_{ij}\right)^{2}}{d_{ij}^{*}} \qquad (5.2)$$

donde,

- d_{ij}^{*} es la distancia entre los vectores i y j en el espacio facial N dimensional,
- d_{ij} es la distancia entre los vectores nuevos sobre el plano 2D, y
- L es el número de vectores que deben ser mapeados.

El mapa obtenido de vectores de código se muestra en la figura 5.4.

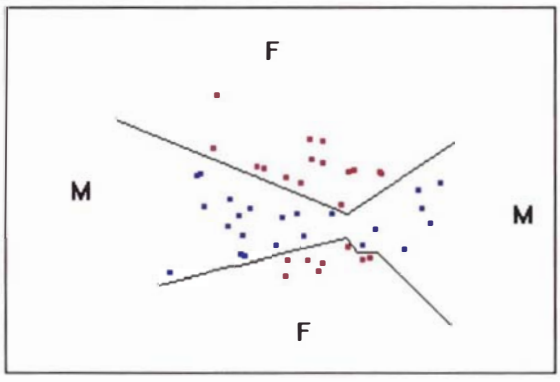


Fig. 5.4: Vector de Código en el plano 2-D, el error de mapeo es de 0.14

Sobre la primera consideración de esta figura, se podría asumir que la clase femenina esta dividida en dos partes la cuál esta separada por la clase masculina (o viceversa). Los vectores de código masculino y femenino efectivamente forman dos clases. Esté gráfico es también evidencia que la clasificación de sexos no puede ser realizada con precisión con la regla simple

de la “distancia mínima desde el vector promedio”, porque el vector de código promedio masculino es casi idéntico que el vector de código promedio femenino.

5.2.3 Segundo Experimento, usando el Error de Reconstrucción

5.2.3.1 Método

Este es un buen método de ingeniería pero no psicológico. No existen evidencias reales de que el cerebro humano trabaje de esta manera. En lugar de usar un solo espacio facial, este método usa un espacio por clase: esto es un espacio facial distinto masculino y femenino.

La generación de los *eigenfaces* masculino y femenino es el mismo empleado en la generación de los *eigenfaces*. Obviamente, la única diferencia es que para construir el espacio facial masculino, se usaron solamente rostros masculinos para el conjunto de entrenamiento, y para el espacio facial femenino, sólo rostros femeninos fueron usados para el conjunto de entrenamiento.

El proceso de reconstrucción ya fue descrito con anterioridad en este documento. La presunción es que el error de reconstrucción, para un rostro masculino en particular reconstruido a partir del espacio facial masculino, será menor que el error de reconstrucción de este rostro masculino reconstruido a partir del espacio facial femenino.

La regla de clasificación se hace simple. Un rostro pertenece a la clase que tiene el error mínimo de reconstrucción. Este método se resume en la figura 5.5.

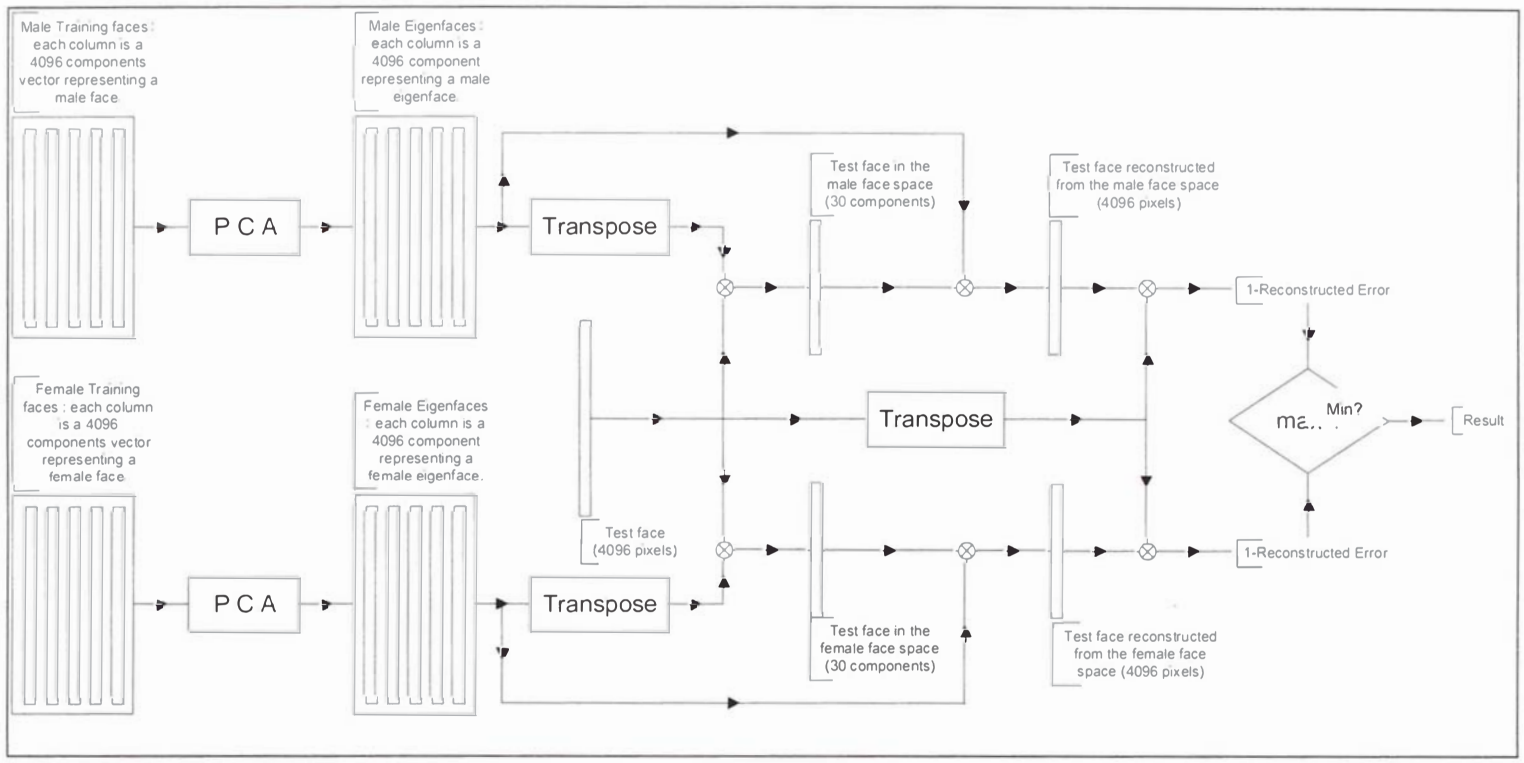


Fig. 5.5: Clasificación basado en el error de reconstrucción

5.2.3.2 Resultados

Los resultados de este método son muy buenos. Se obtiene un porcentaje del 98% de clasificación correcta (con extremos del 100% al 96%). Es interesante notar, como se muestra en la figura 5.6, que los rostros masculinos reconstruidos a partir de un espacio facial femenino se parecen a rostros femeninos y viceversa.

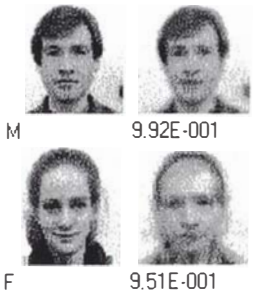


Fig. 5.6: En la parte superior, un rostro masculino con su reconstrucción a partir del espacio facial masculino y, en la parte inferior, un rostro femenino con su reconstrucción a partir del espacio facial masculino

Los resultados satisfactorios de este método anticipan buenos resultados en las pruebas con nuevos individuos. Los resultados anteriores fueron logrados usando imágenes de individuos que pertenecen al conjunto de entrenamiento.

Un nuevo experimento fue ejecutado usando el método *Jack-knife*[43]. El espacio facial femenino fue construido usando 20 imágenes de 6 mujeres e igualmente el espacio facial masculino se construyó usando 20 imágenes de 6 hombres. El séptimo hombre y la séptima mujer (random) fueron los nuevos sujetos para la

[43] Anexo 10

clasificación. El experimento se repitió siete veces. El resultado promedio de este proceso fue del 79%, a pesar de que uno de los experimentos produjo 53%. Sin considerar este experimento el promedio de clasificación correcta fue del 88%. El detalle de lo observado en la data es la imagen de un hombre quien casi siempre fue clasificado como una mujer. Este hombre es calvo, y por consiguiente su imagen no es del todo femenina. La razón de este comportamiento posiblemente proviene del fondo de la imagen. En la base de datos facial la mitad de los píxeles forman el fondo. Se presume que la persona quien fue erróneamente clasificada por sexo, intervino el fondo en su proceso de clasificación.

5.2.4 Conclusión

Existe una paradoja en el primer método de clasificación, en que los resultados producidos son inferiores al proceso de identificación. El proceso de clasificación debería proporcionar mejores resultados que el de identificación. Esto es verdad para el segundo método, el cuál sólo usa la información contenida en el espacio facial que representa a los rostros. El primer método usa un algoritmo complejo para intentar diferenciar las imágenes de los hombres y las mujeres con un número pequeño de vectores de código.

5.3 Comentarios de resultados obtenidos usando imágenes estáticas

Se analizo en este capítulo el comportamiento del ACP cuando es enfrentado a los dos mayores problemas del reconocimiento facial – identificación y clasificación. La técnica *Jack-Knife* se uso para construir los conjuntos de entrenamiento y prueba, dado que los conjuntos originales de entrenamiento y prueba de la base de datos facial no son apropiados para el ACP.

La prueba de identificación usa la regla de la distancia mínima simple desde un vector almacenado, y los resultados observados son buenos (91%).

La clasificación se probo usando dos algoritmos. El método LVQ produce resultados pobres (81%), mientras que el método basado en el error de reconstrucción produce resultados muy buenos (98%) la cuál también se generaliza para individuos desconocidos (88%) a pesar de que la base de datos facial no es una buena base de datos para el ACP.

5.4 Comentarios de resultados obtenidos usando imágenes dinámicas

Una vez analizado y demostrado la ventaja del ACP para el reconocimiento facial sobre imágenes estáticas, se realizo un nuevo experimento usando imágenes de rostros obtenidos en forma dinámica (en vivo). Los resultados generales son aceptables obteniendo porcentajes de certeza superiores al 93.11%. Como el ACP es un proceso 2D, el cuál toma las características

faciales como un todo, se puede decir que el porcentaje mayor de identificación se obtiene cuando se presenta al proceso de identificación un individuo bajo las mismas condiciones de luminosidad y de fondo con las que fue entrenado el sistema. Por lo tanto, si bien las conclusiones estadísticas son concluyentes, se puede decir que dependen grandemente de los algoritmos de ingreso de imágenes faciales hacia el sistema. Como se dijo se hace evidente una cierta normalización de las imágenes faciales con ayuda de un operador. El límite establecido de 93.11% de exactitud al identificar individuos se estableció en forma experimental ya que si bien límites inferiores presumen una identificación correcta, fueron observados errores en el método con seguridad motivados por el fondo o condiciones de luminosidad distintas a las imágenes faciales usadas para el entrenamiento.

CAPITULO VI

IMPLEMENTACION

6.1 Introducción

La primera versión de este programa fue desarrollado por el equipo de investigación del Dr. Sami Romdhani, en 1996, quien es actual profesor en el Instituto de Informática de la Universidad de Freiburg en Alemania. El desarrollo se baso en la identificación de individuos partiendo de imágenes planas hasta cierto punto ideales. Esta primera versión se hizo con la finalidad de demostrar la eficacia de los algoritmos descritos en estas páginas. Los algoritmos LVQ usados en el presente proyecto fueron desarrollados en 1995 por el Equipo de Programación LVQ del Laboratorio de Computación y Ciencias de la Información de la Universidad de Tecnología de Helsinki – Finlandia. La versión presentada aquí tiene la característica que permite identificar individuos en tiempo real y fue desarrollada durante el 2002 por los autores del presente documento .

Esta versión está subdividida en tres partes. El archivo ejecutable (PCA.EXE), y dos DLLs (PCADLL.DLL y LVQ.DLL). Se uso MS Visual C++ 6.0.

6.2 PCADLL.DLL

Como se ha dicho en la discusión teórica sostenida anteriormente, el ACP hace uso extensivo de las matrices. Un rostro puede ser visto como una imagen o un vector columna, y un conjunto de rostros por consiguiente se consideran un conjunto de imágenes o una matriz. Por lo tanto PCADLL.DLL esta subdividida en una sección que maneja vectores y matrices y otra que maneja imágenes. Existe además un puente de clases y funciones que conectan ambas partes.

6.3 PCA.EXE

La funcionalidad principal de esta aplicación esta contenida en la clase *CAAM* (por *Class Auto Associative Memory*). La implementación de esta clase se encuentra en los archivos *CAAM.CPP* y *LOAD.CPP*.

EL único propósito del archivo *LOAD.CPP* es cargar el archivo de comandos del proyecto, interpretar su código y pasar los comandos interpretados a *CAAM.CPP*. La definición del archivo de comandos del proyecto se encuentra en el *Manual del Usuario*[44]. Este archivo proyecto es escrito automáticamente por la aplicación. El motor de este programa esta contenido en el archivo *CAAM.CPP*.

[44] Anexo 1

6.4 LVQ.DLL

El código de LVQD.DLL fue escrito por el equipo de programación LVQ de la Helsinki University of Technology. Este código puede ser bajado del servidor ftp *cochlea.hut.fi (130.233.168.48)* del directorio */pub/lvq_pak/.* Ha sido modificado con la finalidad de ser ensamblado con el resto del sistema ACP:

6.5 Usando Imágenes Faciales Dinámicas

En este punto detallaremos a aplicación propuesta de uso del ACP para el reconocimiento facial usando imágenes faciales dinámicas (captadas en vivo). Como se explico líneas arriba, los algoritmos originales desarrollados por el Instituto de Informática de la Universidad de Freiburg en Alemania en 1996 y por el Equipo de Programación LVQ del Laboratorio de Computación y Ciencias de la Información de la Universidad de Tecnología de Helsinki – Finlandia en 1995; fueron modificados e integrados de tal manera que conformen un solo sistema gráfico y amigable.

6.5.1 Hardware y Software Usado

Hardware:

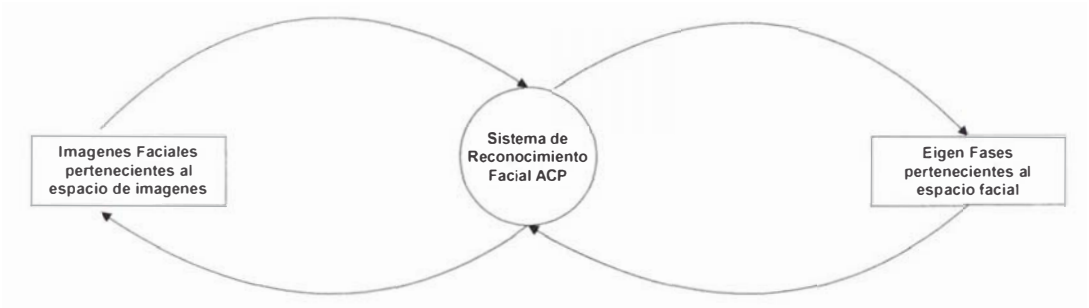
- Una computadora portátil, marca IBM modelo 600X, con 64 Mb de RAM y 600hrz de velocidad de procesamiento.
- Una web cam de Creative Labs modelo CT6840 con entrada USB.

Software:

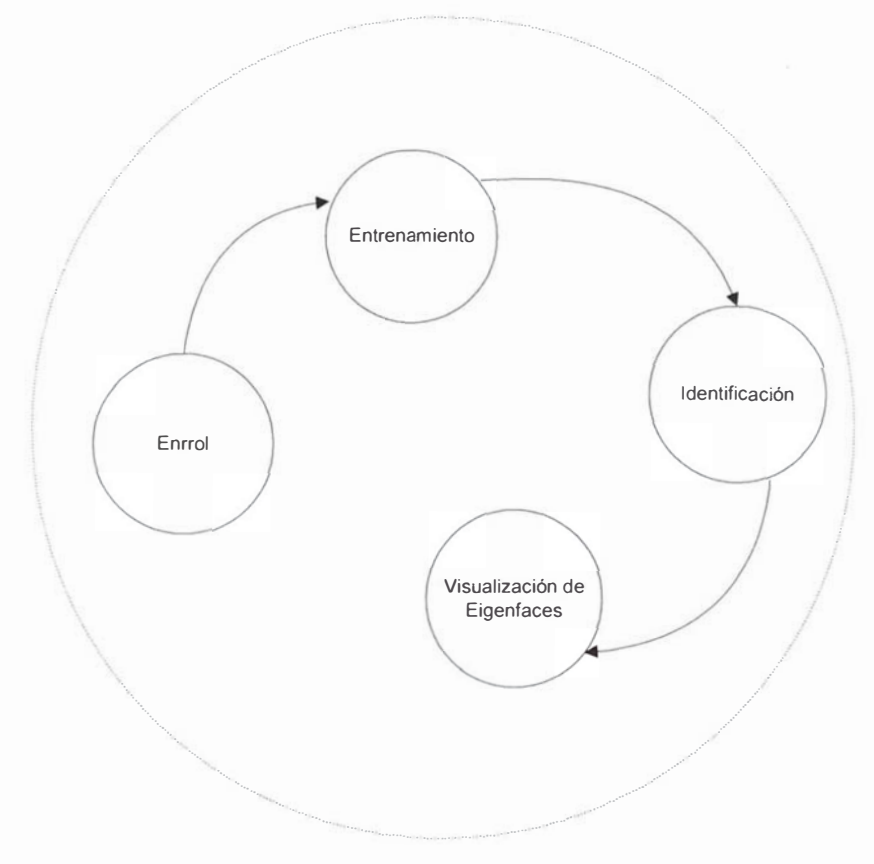
- Sistema Operativo Windows 2000

- Paquete de Desarrollo MS Vision, conjunto de DLLs para realizar tareas visuales usando cámaras.
- MS Visual C++ 6.0, para el desarrollo del sistema.

6.5.2 Diagrama de Contexto



6.5.3 Diagrama de Módulos del Sistema



6.5.4 Diagrama Gráfico del Sistema

Aquí se muestran los componentes necesarios para hacer funcionar el sistema ACP. Como se menciono sólo es necesario definir las variables que intervienen en la generación de la imagen facial como son: luminosidad, fondo y distancia. Es necesario mencionar que la luminosidad puede ser afectada por el entorno el cuál debe ser controlado. Para el funcionamiento del sistema es necesario la presencia desde el inicio de la cámara de video ya que de lo contrario la aplicación producirá un error. La imágenes faciales deben de estar

normalizadas, es decir ubicarse dentro de una posición de espacio previamente limitado, lo cuál se muestra gráficamente en la figura 6.1.

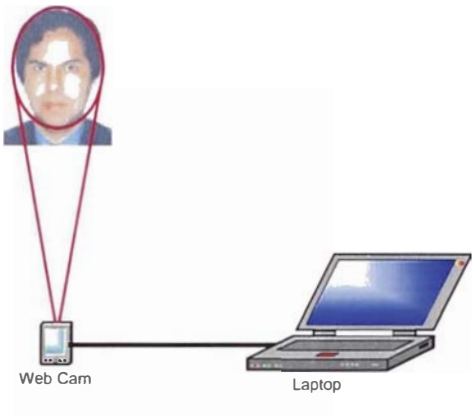


Fig. 6.1: En la parte superior, un rostro masculino ubicado dentro de la posición espacial delimitada por la elipse y en la parte inferior, una cámara de video conectada a un computador portátil

6.5.5 Diagrama en Bloques del Proceso de Enrrol

El proceso de enrrol consiste en la toma de tres imágenes dinámicas del individuo, luego son almacenadas en formato PPM en la Base de Datos de Imágenes tal como muestra la figura 6.2.

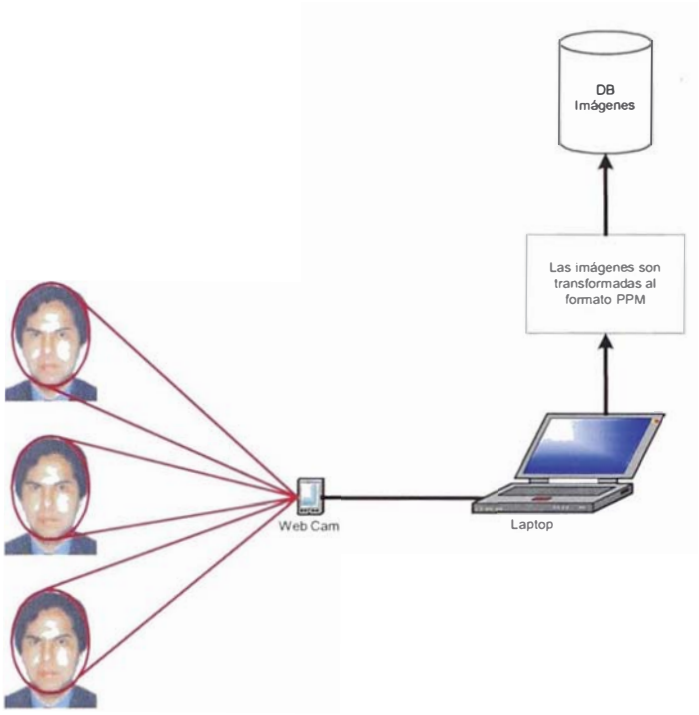


Fig. 6.2: Se toman tres imágenes normalizadas del individuo guardándose estas en formato PPM en la Base de Datos de Imágenes

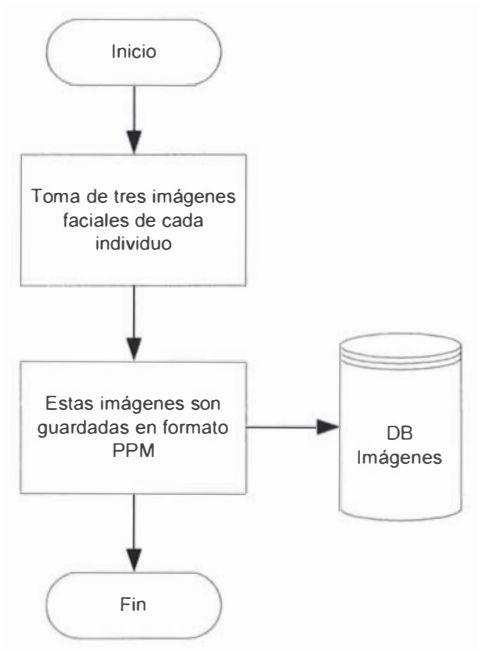


Fig. 6.3: Diagrama en Bloque del Proceso de Enrrol

6.5.6 Diagrama en Bloques del Proceso de Entrenamiento

El proceso de entrenamiento consiste en la aplicación de la *Transformada de Karhunen-Loève* al universo contenido en el Espacio de Imágenes obteniéndose el Espacio Facial compuesta por los *eigenfaces* , almacenándose sólo los componentes relevantes como muestra la figura 6.3.

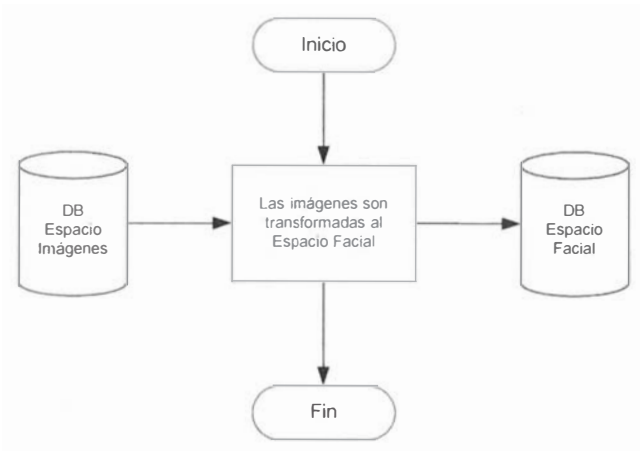


Fig. 6.4: Las imágenes del Espacio de Imágenes son transformadas al Espacio Facial mediante la *Transformada de Karhunen-Loève*

6.5.7 Diagrama en Bloques del Proceso de Identificación

En el proceso de identificación se presenta la imagen facial de un individuo al sistema, este lo transforma al Espacio Facial, luego calcula el eigenface más coincidente, si la precisión de coincidencia es mayor a 0.9311 el sistema ACP lo tomas como una identificación correcta, de lo contrario produce un error de identificación como se muestra en la figura 6.4.

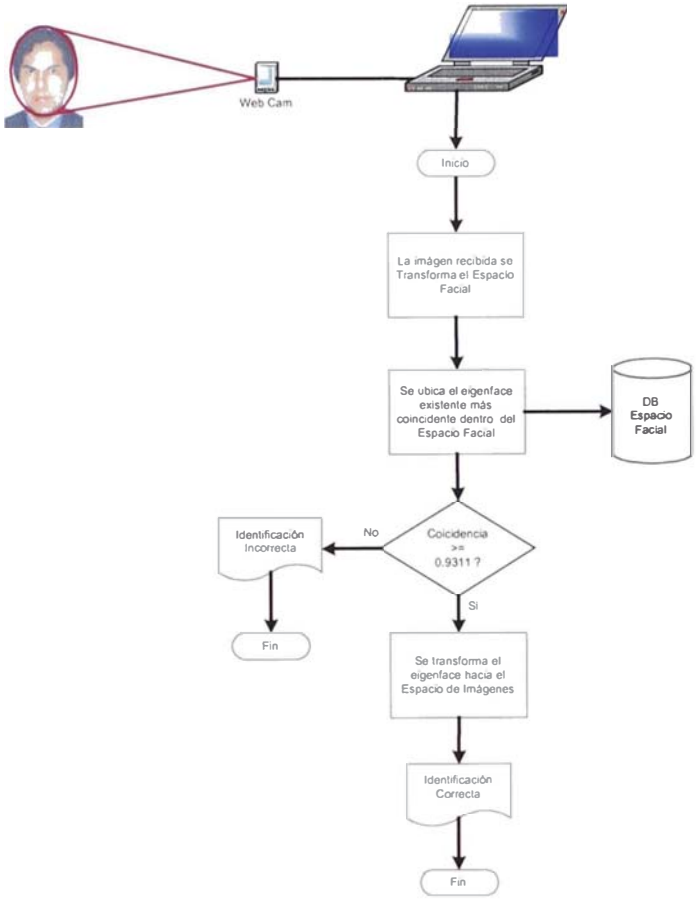


Fig. 6.5: Una imagen facial es presentada al sistema la cuál es transformada al Espacio Facial

6.5.8 Pantallas del sistema "ACP para Reconocimiento Facial 2.0"

6.5.8.1 Presentación:

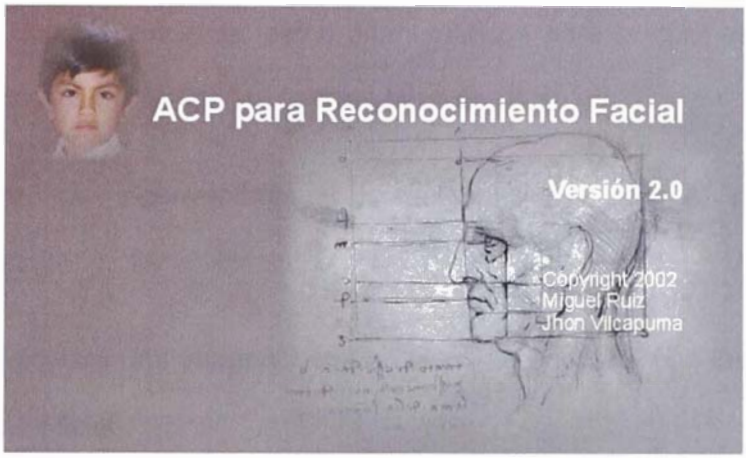


Fig. 6.6: Pantalla inicial de presentación durante la instalación del sistema ACP

6.5.8.2 Pantalla Inicial:

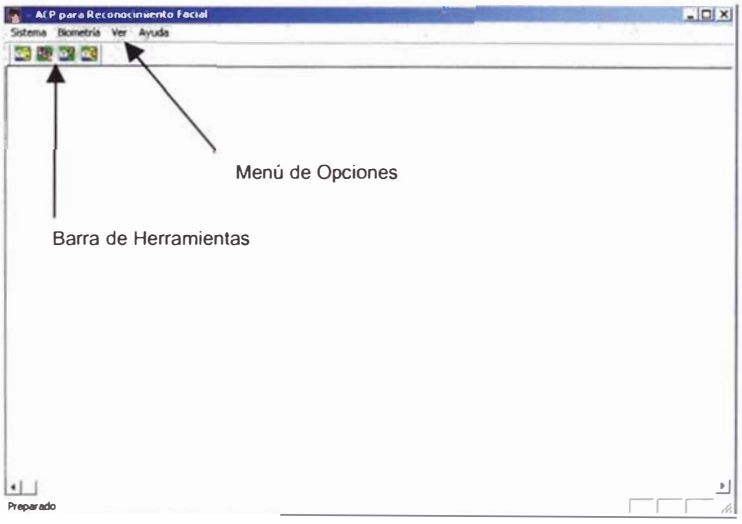


Fig. 6.7: Barras del menú de opciones y de herramientas del sistema ACP

6.5.8.3 Menú Biometría:

Permite acceder a las distintas opciones necesarias para realizar la tarea de identificar a un individuo ha través de su imagen facial, sus opciones son:

Enrrol: Adquiere imágenes faciales por primera vez ó actualiza imágenes existentes. Permite actualizar datos alfanuméricos para lo cuál es necesario actualizar con nuevas imágenes faciales también.

Entrenar: Crea los eigenvectores y los eigenvalues para el proceso de identificación.

Identificación: Adquiere una imagen facial en vivo y la compara con las existentes en el Espacio Facial.

Mantenimiento: Permite el mantenimiento de datos alfanuméricos así como la eliminación de registros existentes. El orden actual es el físico.

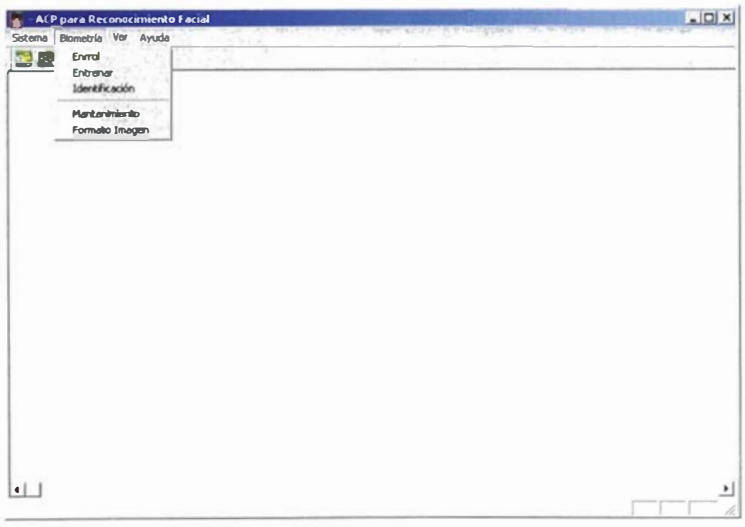


Fig. 6.8: Opciones biométricas del sistema ACP

6.5.8.4 Menú Ver:

Permite acceder a los resultados del proceso de identificación por medio de imágenes faciales, sus opciones son:

Rostros de Entrenamiento: Permite la visualización de la base de datos usada para entrenar el sistema.

Eigen Faces: Permite la visualización de los eigenfaces calculados.

Rostros Reconstruidos: Permite la visualización de la asociación más cercana obtenida por el sistema de un rostro existente en la base de datos facial de la imagen presentada de un rostro parcialmente eliminado.

Rostros Reconocidos: Permite la visualización del proceso de identificación.

Proyección Sammon: Permite la visualización del proceso de clasificación de sexo.

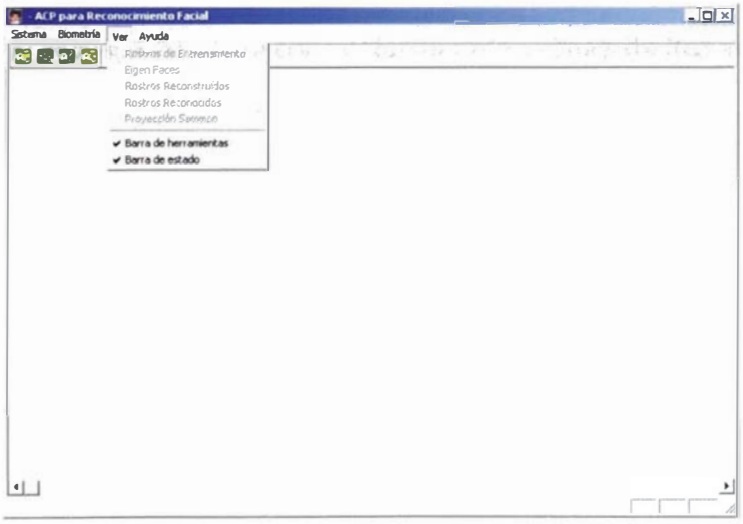


Fig. 6.9: Opciones para visualizar resultados obtenidos mediante el ACP

6.5.8.5 Pantalla Enrrol:

En la parte superior izquierda se displaya una ventana permanente de toma en vivo, se trata de tomar las mejores imágenes faciales del individuo tratando de que estén lo más normalizadas posibles, las tres imágenes del individuo se usarán para entrenar al sistema.

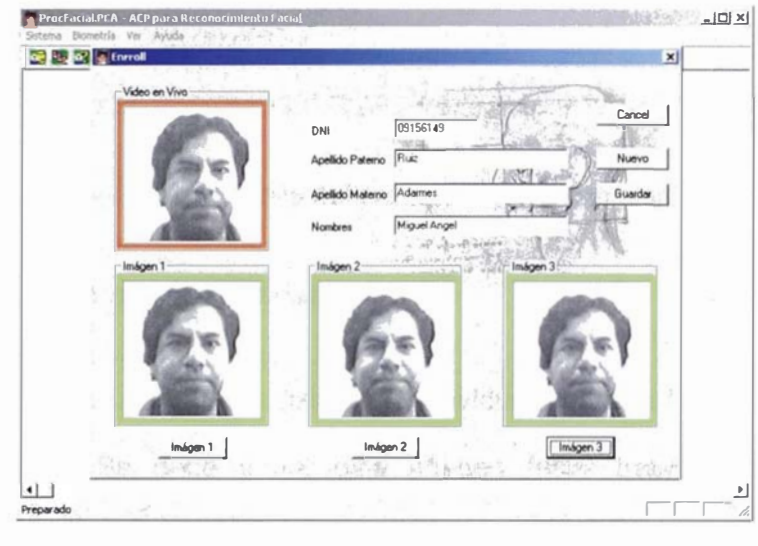


Fig. 6.10: Pantalla de enrrol con ventanas para captura de tres imágenes faciales del individuo

6.5.8.6 Pantalla Entrenamiento:

Permite el cálculo de Espacio Facial que mejor representan a las imágenes del Espacio de Imágenes. Se ha definido como default el cálculo de 6 eigenfaces.

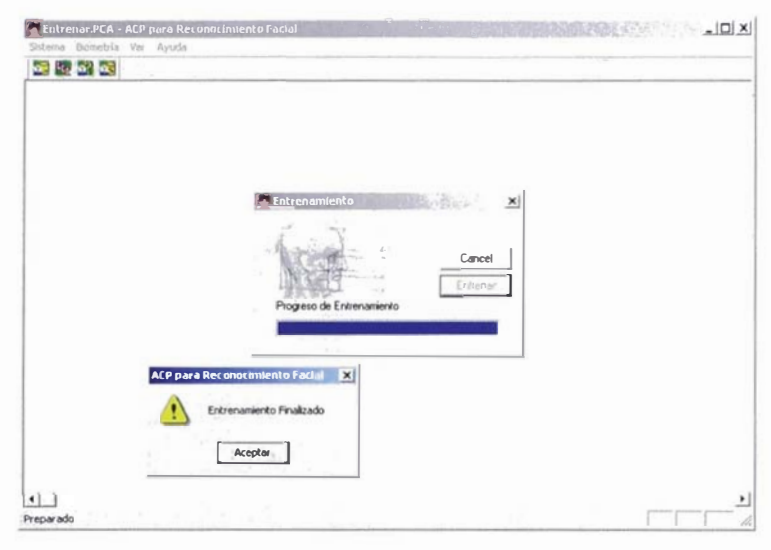


Fig. 6.11: Pantalla del proceso de entrenamiento

6.5.8.7 Pantalla Identificación:

Permite identificar la imagen facial de un individuo mediante la búsqueda en el Espacio Facial. Se debe tomar esta imagen facial tratando de que las condiciones de Enrrol se repitan lo más cercanamente posible. Si el porcentaje de exactitud supera el 93.11%, muestra los datos alfanuméricos del individuo ubicado, de lo contrario produce un error de identificación.

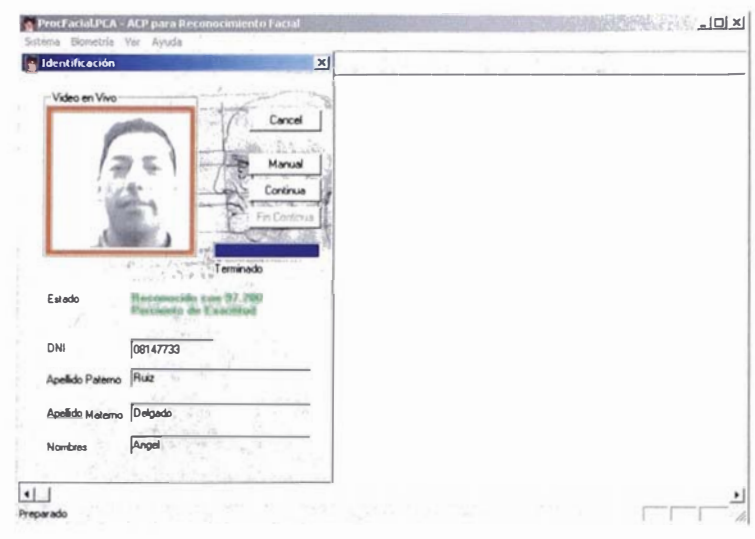


Fig. 6.12: Pantalla para identificación ha través de una imagen facial presentada

6.5.8.8 Pantalla Mantenimiento:

Permite ubicar un registro ha través de su número de su DNI. En esta versión no existe búsqueda por otros datos y el orden mostrado es el físico; pero estimo que no son necesarias estas opciones para la finalidad de demostrar el ACP. Estas opciones pueden ser fácilmente implementadas en una versión posterior. En esta pantalla también es posible eliminar un registro existente.

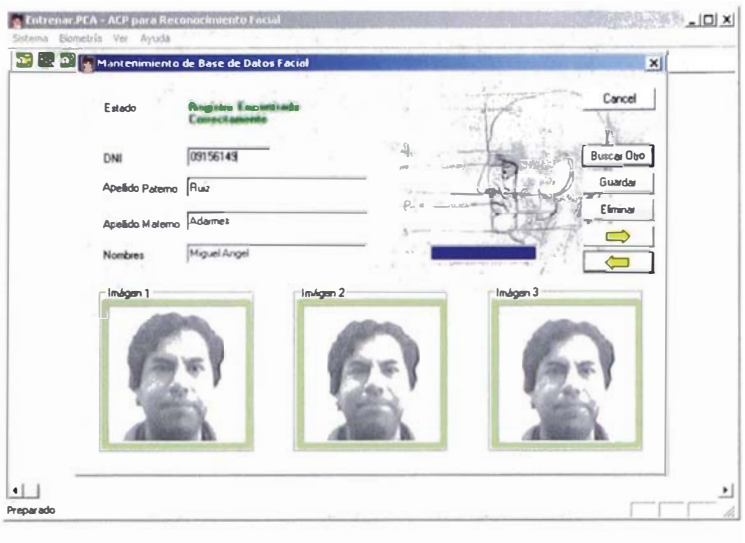


Fig. 6.13: Pantalla para mantenimiento y navegación sobre la base de datos facial

6.5.8.9 Pantalla Ver Eigen Faces:

Permite ver los eigenfaces calculados. Los números mostrados debajo de cada eigenface son los eigenvalues, los cuáles muestran la importancia de cada uno. Como se dijo con anterioridad en la discusión teórica, el primer eigenface es conocido como el rostro promedio y es el más importante.

Fig. 6.14: Eigenfaces obtenidos con sus pesos asociados

6.5.8.10 Pantalla Ver Rostros de Entrenamiento:

Permite visualizar el universo de rostros pertenecientes a la base de datos facial.

Fig. 6.15: Base de datos facial para entrenamiento

6.5.8.11 Pantalla Ver Rostros Reconocidos:

Permite visualizar el resultado del proceso de identificación. El número inferior es la precisión obtenida en la reconstrucción.

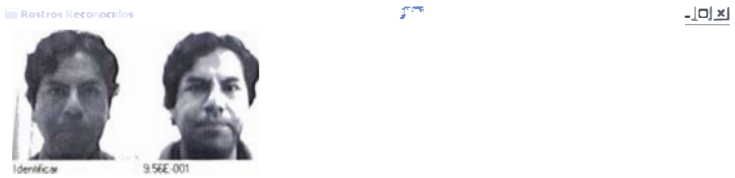


Fig. 6.16: Resultado del proceso de identificación. La imagen izquierda es la previamente enrrolada

CONCLUSIONES Y RECOMENDACIONES

Aquí se describen las conclusiones finales y recomendaciones que se deden de tener en cuenta como resultado de la investigación realizada en el presente documento en lo referente al Reconocimiento Facial.

CONCLUSIONES

1. Resumen del Proceso de Reconocimiento Facial:

   Los componentes principales L son los primeros *eigenvectors* de la matriz de covarianzas W. Así que el objetivo es calcular los *eigenvectors* de W.

   W es la matriz de covarianzas de X la cuál es la matriz del conjunto de entrenamiento. El producto externo de una matriz es el producto de esta matriz con su forma transpuesta.

   La matriz del conjunto de entrenamiento es la matriz donde los vectores entrenamiento son colocados, cada vector entrenamiento forma una columna de X.

   Se define a P como la matriz de los primeros L *eigenvectors,* cada *eigenvector* es una columna de P. Así que el objetivo es calcular P.

Para calcular W y luego P es necesario mucho tiempo de computación. Efectivamente, X es una matriz de N por K (donde N es el número de elementos (píxeles) en un vector (rostro)). Por consiguiente W es una matriz N por N. Pero N es mucho más grande que K. Por ejemplo, si se usará 300 rostros con cada rostro teniendo una resolución de 64 por 64 píxeles, entonces K = 300 y N = 64*64 = 4096.

En cambio es mucho mejor calcular V, que es el producto interno de X. El producto interno de una matriz es el producto de la forma transpuesta de esta matriz por ella misma. Así V es una matriz K por K.

Se puede mostrar que P = X * Q * E donde:

- Q es la matriz de *eigenvectors*[45] de V.
- E es la matriz invertida diagonal cuadrada de los *valores eigen*[46] de V.

La prueba detallada de este resultado es mostrado en muchos libros especializados sobre álgebra lineal.

Esta es la manera que el ACP calcula los componentes principales.

Sea F la matriz de los componentes de los rostros almacenados en la matriz X (la iesima columna de F forma los componentes de la iesima columna de X, el cuál es un rostro). ACP calcula F como sigue:

- $F = P^T * X$

[45] Anexo 12
[46] Anexo 6

Sea Y la matriz de reconstrucción de los componentes almacenados en la matriz F. ACP calcula Y como sigue:

- $Y = P * F$

Note que $Y = P * F = P * P^T * X = W * X$

2. EL ACP usa gran cantidad de productos de grandes matrices, lo cuál representa gran tiempo de computación. Esto explica porque el proceso de Reconocimiento Facial esta aún muy lejos de realizarse en tiempo real para grandes poblaciones.

3. El Reconocimiento Facial puede ser aplicado con gran éxito en distintos sistemas que requieran de su aplicación siempre que las condicieones del hardware y del software lo permitan.

RECOMENDACIONES

1. Esta biométria puede ser aplicada en procesos de identificación policiaca, civil y sobre multitudes.

2. Para usar efectivamente los algoritmos descritos aquí son necesarios computadores potentes.

3. Para un efectivo Reconocimiento Facial es necesario una base de datos facial centralizada.

BIBLIOGRAFIA

1. Karhunen, K. “Zur Spektraktheorie Stochasticher”, *Prozessa Ann. Sci. Fennicae*, (1946), 37
2. Loève, M. M., “Probabilty Theory”, Princeton, N. J.: Van Nostrand, 1955
3. Lorenz, E. N., “Empirical Orthogonal Functions and Statistical Weather Prediction”, Cambridge: MIT, Department of Meteorology, Statistical Forecasting Project, 1956.
4. H. Hotelling, “Analisys of a complex of statistical variables into principal components”, Journal of Educational Psychology, (1933) 24, pp. 417 – 441, 488 – 520
5. Brooks, C. L., Karplus, M., and Pettitt, B.M., “Proteins: a Theoretical Perspective of Dynamics, Structure and Thermodynamics”, New York: Wiley, 1988
6. Lumley, J.L., “The Structure of Inhomogeneous Turbulent Flows” In *Atmospheric turbulence and radio propagation*, editado por A.M. Yaglom and V.I. Tatarski, Moscow: Nauka, 1967, pp.166-178
7. Golub, G.H., Van Loan, C.F., “Matrix Computations”, Oxford: North Oxford Academic, 1983
8. Harman, H., “Modern Factor Analysis”, Chicago: University of Chicago Press, 1960
9. Schmidt, E., “Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Teil: Entwicklung willkülicher Funktion nach

Systemen vorgeschriebener", *Mathematische Annalen*, (1907), 63, pp. 433-476

10. Sirovich ,L, "Turbulence and the Dynamics of Coherent Structure: I, II and III", *Quarterly Applied Mathematics*, (1987), 45, pp. 561
11. H. Murase y S. Nayar, "Visual learning and recognition of 3-D objects from appearance", *International Journal of Computer Vision*, (1995), 14, pp.5-24
12. Graham, M.D., Kevrekidis, I.G., "Alternative approaches to the Karhunen-Loève decomposition for model reduction and data analysis", *Computers and chemical engineering*, (1996), 20, p. 495
13. Yamashita, Y., Ogawa, H., "Relative Karhunen-Loève Transform", *IEEE transactions on signal processing*, (1996), 44 , p. 371
14. Regalia, P.A., "An Adaptive Unit Norm Filter with Applications to Signal Analysis and Karhunen-Loeve Transformations", *IEEE transactions on circuits and systems*, (1990), 37, p. 646
15. Qin, F., E.E. Wolf, H.-C. Chang and C. Hwang, "Controlling thermal front propagation on a catalityc water through video feedback", In *Proceedings of the American Control Conference*, (1993), 2, p.1302
16. Stewart, G.W., "On the Early History of the Singular Value Decomposition", *SIAM Review*, (1993), 35, pp.551-566
17. Tasto, M., Wintz, P., "Image Coding by Adaptive Subimage Quantization", *IEEE Transactions on Communications*, (1971), COM-19 , pp. 957-972

18. Everson, R., Sirovich, L., “Karhunen-Loève procedure for gappy data”, *Journal of the optical society of America. A, Optics and image science*, (1995), 12, p. 1657

19. Hilai, R., Rubinstein, J., “Recognition of rotated images by invariant Karhunen-Loève expansion”, *Journal of the optical society of America. A, Optics and image science*, (1994), 11, p. 1610

20. Abbas, H.M., Fahmy, M. M., “Neural model for Karhunen-Loève transform with application to adaptive image”, *IEEE proceedings. I, Communications, speech, and vision*, (1993), 140, p. 135

21. Burl, J.B., “Reduced-Order System Identification using the Karhunen-Loève Transform”, *International journal of modeling and simulation*, (1993), 13 , p. 183

22. Chen, C.S., Huo, K.-S., “Karhunen-Loève method for data compression and speech synthesis”, *IEE proceedings. I, Communications, speech and vision*, (1991), 138, p. 377

23. Sirovich, L., Everson, R., “Management and Analysis of Large Scientific Datasets”, *International journal of supercomputer applications*, (1992), 6 , p.50

24. Kirby, M., Sirovich, S., “Application of the Karhunen-Loève Procedure for the characterization of human faces”, *IEEE transactions on pattern analysis and machine intelligence*, (1990), 12, pp. 103-108

25. S. Watanabe, “Karhunen-Loève expansion and factor analysis theoretical remarks and applications”, In *Transactions of the 4th Prague Conference on Information Theory*, (1965)

26. Gorecki, C., “Surface classification by an optoelectronic implementation of the Karhunen-Loève expansion”, *Applied optics*, (19), 30 , pp.4548-4553

27. Krischer K., R. Rico-Martinez, J.G. Kevrekidis, H.H Rotenmund, G. Ertl and J.L Hudson, "Model Identification of a Spatiotemporal Varying Catalytic Reaction", *American Institute of Chemical Engineers Journal*, (1993), 39, pp.89-98

28. K. Fukunaga, "Introduction to Statistical Recognition", Academic Press, 1990

29. Hubel, D., Wiesel, T., "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex", *Journal of Physiology (London)*, (1962), 160 , pp.106-154

30. Dony, R., Haykin, S., "Neural network approaches to ımage compression", *Proceedings of the IEEE*, (1995), 83(2), pp.288-303

31. Gay, D.H., Ray, W.H., "Identification and control of linear distributed parameter systems through the use of experimentally determined singular functions", *IFAC Proceddings Series. Control of Distributed Parameter Systems, Los Angeles, 30 June-2 July 1986*, ed. H.E. Rauch, pp. 173-179. Oxford/New York:Pergamon

32. V.R Algazi and D.J. Sakrison, "On the Optimality of the Karhunen-Loève Expansion", *IEEE Trans. Inform. Theory*, (1969), 15, pp.319-321

33. Turk, M. A., Pentland, A. P., "Face recognition using eigenfaces", (1991), *Proceedings of 1991 IEEE Computer Society Conference on Computer Vision Pattern Recognition*, pp.586-591

34. Turk, M. A., Pentland, A. P., "Recognition in face space", *Proceedings of SPIE - The International Society for Optical Engineering*, 1381, pp.43-54: Published by Int. Soc. for Optical Engineering, Bellingham, WA, USA.

35. James Griffioen, Brent Seales, Raj Yavatkar, "Content-based Multimedia Data Management and Efficient Remote Access", URL: http://www.uky.edu/kiernan/DL /brent.html

36. D.H. Ballard y C.M. Brown, en *Computer Vision*, Prentice Hall. 1982.

37. E. Davies, en *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press. 1990.

38. R. Haralick y L. Shapiro., en *Computer and Robot Vision Vol I*, Addison-Wesley. 1992.

39. M. Sonka, V. Hlavac y R.Boyle, en *Image Processing, Analysis and Machine Vision*, Chapman & Hall. 1993.

40. R.O. Duda y P.E. Hart, en *Use of the Hough transform to detect Lines and Curves in Pictures*, Communications of the ACM, vol. 15, no. 1, pp. 11-15, 1972.

41. P.V.C., en *A Method and Means for Recognizing Complex Patterns*, U.S. Patent N. 3,069,654, 1962.

42. J. Illingworth and J. Kittler, en *A survey of the Hough Transform*, CVGIP, vol. 44, pp. 87-116, 1988.

43. Kalviainen, H., Hirvonen, P., Xu, L., y Oja, E., en *Probabilistic and Non-probabilistic Hough Transforms: Overview and Comparisons*, Image and Vision Computing, vol. 13, no. 4, pp. 239-252, 1995.

44. Suchendra M. Bhandarkar, Minsoo Suk, en *Qualitative features and the Generalized Hough Transform*, Pattern Recognition, Vol. 25, N. 9, pp. 987-1006, 1992.

45. J. W. Sammon, Jr, en *A Nonlinear Mapping for Data Structure Analysis*, IEEE Transactions on Computers, Vol. C-18, No. 5, (Mayo 1969).

46. T. Kohonen, en *Correlation Matrix Memories*, IEEE Transactions on Computers, Vol. C-21, No. 4, (Abril 1972).

47. Duda y P.E. Hart, en *Pattern classification and scene analysis*, Wiley, New-York (1973).

48. Anderson, J. W. Silverstein, S. A. Ritz and R.S. Jones, en *Distinctive features, categorical perception, and probability learning : Some applications of a neural model*, Psychol. Rev. 84, 413-451 (1977).

49. Kohonen, en *Associative memory : A system theoretic approach*. Springer-Verlag, Berlin (1977).

50. McClelland and D.E. Rumelhart, en *Parallel distributed processing : Explorations in the microstructure of cognition*, MIT Press/Bradsford Books, Cambridge (MA) (1986).

51. Abdi, *A generalized approach for connectionist auto-associative memories : interpretation, implication & illustration for face processing*, en J. Demongeot, et al. *Artificial intelligence and cognitive sciences*, Manchester university press, pp 149-165, (1988).

52. G. H. Dunteman, en *Principal components analysis*, Sage publications, (1989).

53. T. Kohonen, en *Self-organition and associative memory*, Third Ed., Springer-Verlag, (1989).

54. A. M. Burton, V. Bruce and R. A. Johnston, en *Understanding face recognition with an interactive activation model*, British Journal of Psychology, 81, pg 361-380, (1990).

55. M. Kirby, and L. Sirovich, en *Application of the Karhunen-Loève procedure for the characterization of human faces*, IEEE transactions on pattern analysis and machine intelligence, Vol. 12, No. 1, (1990).

56. T. Kohonen, *Statistical pattern recognition revisited*, en *Advanced Neural Computers*, R. Eckmiller, Elsevier, (1990).

57. T. Kohonen, en *The self-organizing map*, Proc. of the IEEE, Vol. 78, No. 9, (1990).

58. M. A. Turk y A. P. Pentland, en *Face recognition using eigenfaces*, Proc. of the IEEE on Computer Soc. Conf. (1991).

59. M. A. Turk y A. P. Pentland, en *Recognition in face space*, Int. Soc. for Optical Engineering Bellingham, WA, USA, (1991).

60. T. Kohonen, J. Hynninen, J. Kangas, y J. Laaksonen, en *LVQ_PAK : A program package for the correct application of Linear Vector Quantization algorithms*, (1992).

61. H. Abdi, en *Les réseaux de neurones*, Presses universitaires de Grenoble (1993).

62. A. M. Burton y V. Bruce, en *Naming faces and naming names : exploring an interactive activation model of person recognition*, Memory, 1 (4), 457-480, (1993).

63. S. A. Sirohey, en *Human face segmentation and identification*, (1993).

64. H. Abdi, y D. Valentin, en *Modèles neuronaux, connexionistes et numeriques pour la mémoire des visages*, Psychologie Francaise No. 39, (1994).

65. M. Bichsel and A. P. Pentland, en *Human face recognition and the face image et's topology*, CVGIP : Image Understanding, Vol. 59, No. 2, March, pp 254-261, (1994).

66. A. M. Burton, en *Learning new faces in an interactive activation and competition model*, Visual cognition, 1 (2/3), 313-348, (1994).

67. B. Moghaddam, y A. Pentland, en *Face recognition using view-based and modular eigenspaces*, Automatic systems for the identification and inspection of humans, SPIE vol. 2277, (Julio 1994).

68. A. Pentland, R. W. Picard, y S. Scarloff, en *Photobook : content-based manipulation of image databases*, SPIE Storage and retrieval image and video databases II, No. 2185, (Feb. 1994).

69. D. Valentin, H. Abdi, A. J. O'Toole, G. W. Cottrell, en *Connectionist models of face processing : a survey, Pattern Recognition*, Vol. 27, pp.1209-1230, (1994).

70. D. Valentin, H. Abdi, A. J. O'Toole, en *Categorization and identification of human face images by neural networks : a review of the linear autoassociative and principal component approaches*, Journal of Biological Systems (2), pp 413-429, (1994).

71. Y. Yacoob y L. S. Davis, en *Recognizing human facial expression*, (Mayo 1994).

72. H. Abdi, D. Valentin, B. Edelman, A.J. O'Toole, en *More about the difference between men and women : evidence from linear networks and principal components approach*, Perception, vol. 24, pg. 539-562, (1995).

73. R. Chellappa, C. L. Wilson, y S. Sirohey, en *Human and machine recognition of faces : a survey*, Proc. of the IEEE, (Mayo 1995).

74. B. Moghaddam, y A. Pentland, en *An automatic system for model-based coding of faces*, IEEE data compression conference, Snowbird, Utah, (Marzo 1995).

75. T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, en *SOM_PAK, the self-organizing map program package ver 3.1*, ftp anómino en cochlea.hut.fi (130.233.168.48) dir. /pub/som_pak (1995).

76. P. J.B. Hancock, A. M. Burton y V. Bruce, en *Face processing : human perception and principal component analysis*, Memory and Cognition (1996).

77. D. Valentin, y H. Abdi, en *Can a linear autoassociator recognize faces from new orientations ?*, J. Opt. Soc. Am. A., Vol 13, (1996).

78. D. Valentin, H. Abdi y A. J. O'Toole, en *Principal component and neural network analyses of face images : Explorations into the nature of information available for classifying faces by sex*, en Progress en mathematical psychology, New-York : Elsevier.

79. W.H. Press et al., en *Numerical recipes in C - The art of scientific computing*, segunda ed.

80. I. Craw, D. Tock, y A. Bennett, en *Finding face features*.

81. T. Kohonen, en *Representation and processing of associations using vector space operations*.