UNIVERSIDAD POLITECNICA SALESIANA
SEDE CUENCA
CARRERA DE INGENIERIA DE SISTEMAS

Tema:

"EXTRACCIÓN DE CARACTERÍSTICAS PARA LA CLASIFICACIÓN DE IMÁGENES A TRAVÉS DE LA REDUCCIÓN DE DIMENSIONALIDAD APLICANDO ANÁLISIS DE COMPONENTES INDEPENDIENTES (ICA)"

Tesis previa a la obtención del

Título de Ingeniero de Sistemas

AUTOR:

Esteban Andrés López Pacheco

DIRECTOR:

Ing. Vladimir Robles Bykbaev

CUENCA - ECUADOR

2013

Breve reseña del autor e información de contacto

Esteban Andrés López Pacheco
Egresado de la Carrera de Ingeniería de Sistemas
Universidad Politécnica Salesiana
lopez.eandres@hotmail.com

Ing. Vladimir Robles B.

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, realizada por el Sr. Andrés López Pacheco, y por cumplir los requisitos autorizo su presentación.

Cuenca, Junio del 2013

Ing. Vladimir Robles B.

Director de Tesis

DECLARACIÓN DE RESPONSABILIDAD

Yo, Esteban Andrés López Pacheco, portador de la cédula de ciudadanía 0105610539, estudiante de la Ingeniería en Sistemas, certifico que los conceptos desarrollados, análisis realizados, así como los criterios vertidos en la totalidad del presente trabajo son de exclusiva responsabilidad del autor y autorizo a la Universidad Politécnica Salesiana el uso de la misma con fines académicos.

Cuenca, Junio del 2013

Andrés López Pacheco

DEDICATORIA

Por haber estado una vez más presente en esta batalla, la más difícil en lo que lleva mi vida, este gran triunfo que representa el terminar la universidad te lo dedico a ti. Aún recuerdo claramente aquellos días en los que me enseñabas los más ejemplares valores del día a día, días en los que empecé a convivir contigo, aún recuerdo aquel triste día en el que inesperadamente te fuiste, día en el que te prometí salir adelante y afrontar valientemente cualquier reto que el camino me tenga preparado. Desde aquel día muchas batallas se me han presentado en mi camino, en cada una de ellas te he sentido a mi lado ya que has causado en mi la suficiente inspiración para no dar mi brazo a torcer. Todo este trabajo va dedicado a tu memoria abuelita Leti.

A ti abuelito Leonidas, que a pesar de todos tus años te sigues levantando firmemente día tras día causándome una gran admiración por la fuerza con la que vives, mostrándome que las adversidades de la vida son superables si las enfrentamos con una gran actitud y muchas ganas de vivir.

A ustedes, mis amados padres, sin ustedes es seguro que nada de lo que ahora estoy viviendo hubiese sido posible, su apoyo incondicional ha sido vital en los éxitos que ahora cosecho. Padres como ustedes no los hay en todos lados, nuestra relación no siempre ha sido la mejor, pero así mismo, la forma de mostrarles cuanto aprecio todo lo que me han instruido es logrando superar difíciles pruebas de la vida, que al final son victorias en las que ustedes sin falta están. Reitero, sin su amor y su apoyo, difícilmente hubiese llegado a ser quien ahora soy.

Por último, sin eso significar menos importante, a mis hermanos, Johanna y Mateo, y como no, a mi sobrino Isaac, fueron siempre una inspiración que me motivaba a superarme todos los días, ya sea por seguir su ejemplo, o también ser yo quien lo ponga, para ti Mateo, te motivo a que me superes con creces, sé que tienes un gran potencial para lograr grandes cosas, y que así mismo representes el mejor ejemplo para nuestro sobrino Isaac.

Andrés

AGRADECIMIENTOS

Con este documento, se cierra una de las más hermosas etapas de mi vida hasta ahora, con la mano en el corazón digo que la universidad cambio mi vida para bien, que las personas que durante estos más de 5 años conocí han ido formando poco a poco mi carácter y mi personalidad, aquellos docentes que año tras año me retaban a dar lo mejor de mí y mucho más, principalmente aquellos que empezaron siendo profesores, y los he terminado considerando amigos. Empiezo con el Ing. Vladimir Robles, a quien conocí como docente estando aún en segundo ciclo, desde aquel momento supe que tenerlo como docente marcaría un antes y después en mi vida académica, prueba de aquello son los concursos que he ganado gracias a las sabias enseñanzas que fueron impartidas en sus clases, pero más allá de ser docente, su personalidad es la que considero ideal para un docente, ya que al final se mostró como una persona llena de valores, totalmente amigable. Me considero totalmente afortunado de que esta tesis haya sido dirigida por usted, aprendiendo al final mucho sobre este tema y además sobre ortografía, gracias por ser mi docente y amigo durante casi toda mi vida universitaria.

Además de Vladimir, docentes como la Ing. Tacuri, Ing. Ingavelez, Ing. Quinde, Ing. Quintuña quienes en su debido momento confiaron en mí asignándome responsabilidades que trascendían de la vida estudiantil, más que responsabilidades, llegaron a ser oportunidades que muy agradecido las tome.

¿Qué habría sido de mi vida estudiantil sin amigos?, de seguro no sería la misma persona que ahora soy. Siempre he considerado que en la universidad se conoce a las personas que permanecerán para toda la vida, una de aquellas personas eres tu Fátima, querida "primita", realmente no tengo idea de lo mucho que te he llegado a querer, aunque con mucha gracia recuerdo que no siempre fuimos los mejores amigos que ahora somos, eres de las personas que deseo que estén para toda mi vida, siempre te deseare lo mejor, me causa mucha nostalgia el recordar aquellos días en los que coincidíamos en cada hora, aquellos largos fines de semana de tareas, y en algunas ocasiones, amanecidas por proyectos, si que formamos un gran equipo querida prima, "López & López", deberíamos pensar en patentar una marca. Ojala que la vida nos reúna de nuevo más adelante.

Una gran amiga eres tú también Priscila "picher", tantas historias que asimismo recuerdo desde el día en que te conocí, hiciste de mi vida universitaria bastante mejor, si que adoro matarte de iras, hablar por horas y horas, coincidir en proyectos, apoyarnos mutuamente en cualquier necesidad que se nos presentaba, te echo de menos pichita.

Este documento crecería demasiado si nombrara a todos aquellos amigos que aportaron con su granito de arena para que mi vida universitaria sea la mejor, gracias Daniel, Joshe, Pedro, Tavo, Gato, Tuto, Gordo, Punka, Eva, Marco, Edisson, Geovanny, Blanco, Libio, Davicho, Mocho, Jenner, Lore, Dunia, Jessy, Ardilla, Kelo, Diabla, Brujita, Javier, Diego, Jacky, Geovy, Jorge, etc.

Trascendiendo las tareas universitarias, tuve la grandiosa oportunidad de ser representante estudiantil, lo que ha dejado en mí un muy grato recuerdo, un muy cordial agradecimiento al Eco. Luis Tobar, Vicerrector de la UPS sede Cuenca, por todas aquellas horas de planificación y negociación de eventos. Al Padre Javier Herrán, Rector de la UPS, por brindarme su vital apoyo en varias ideas. A diversas autoridades con las que trace una buena amistad, gracias a Nancita, Vilmita, Sole, Paolita, Carmita, Manolin, Blas, Andrés, Xavier, María, Ing. Díaz, Vicky, Eco. Vázquez, etc.

De parte de mi familia, a mis padres por confiar plenamente en mi al permitirme estudiar los 5 años que duro mi carrera universitaria.

No podría olvidar el agradecer también a una persona que durante el transcurso de mi tesis ha estado pendiente, y siempre apoyándome a que siga adelante, me refiero a ti Anahi, han sido muy motivadoras todas aquellas largas jornadas en las que me has acompañado, subiéndome los ánimos cuando parecía que todo estaba perdido. Gracias amor, sin ti todo hubiese sido mucho más difícil.

A mis nuevas grandes amistades que han estado también pendientes del desarrollar de este trabajo, pendientes por mi bienestar en mi nueva vida, gracias Tania, Lenin, Xavier y Livania.

Mi grandiosa experiencia universitaria ha sido gracias a cada uno de ustedes. Los aprecio demasiado.

Andrés

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Antecedentes

Una de las más apasionantes y extensas ramas de la inteligencia artificial, y de la computación en general, es la visión por computador. Por ello, podemos anotar que, la visión por computador abarca a cada técnica que permite el uso de algoritmos computacionales para la extracción, análisis y procesamiento de información útil, partiendo de representaciones gráficas de datos, como imágenes o vídeos. Una de las motivaciones principales para la existencia de esta rama es la de tratar de emular la visión humana, y proyectándose más allá, mejorarla, ya que gracias al poder computacional de la actualidad podríamos procesar mucha más información que la que un humano podría.

El procesamiento de imágenes es una extenuante tarea para el computador, ya que se dispone de gran cantidad de información a ser analizada. Una simple imagen de un paisaje está compuesta por una gran cantidad de datos, los que saltan a primera vista son los colores. Cada pixel que podemos apreciar en una imagen es el resultado de una combinación de una serie de valores, cada uno de estos tiene su propia función. Al juntar todos los pixeles de una imagen se puede suponer que la información desde un principio es grande, entonces, algo sumamente eficiente es la reducción de dimensiones.

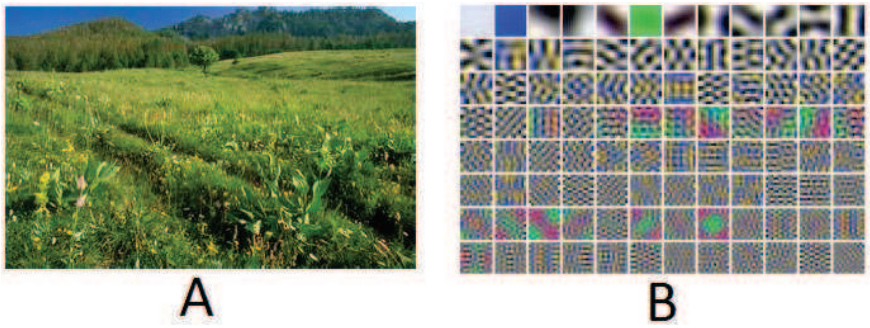


Figura 1.1: En A podemos apreciar una imagen de un paisaje cualquiera, mientras que en B, encontramos a la misma imagen luego de aplicarle algoritmos de reducción de dimensionalidad.

Otra de las amplias áreas en las que está presente el procesamiento de imágenes es la reducción de dimensiones, interpretando lo dicho en párrafos anteriores, podemos deducir que la información almacenada en una imagen es sumamente amplia, ocupando sobre todo recursos de almacenamiento. Existe una diversidad bastante grande de procesos que logran disminuir el tamaño de los elementos multimedia, logrando así eficiencia, tanto en el almacenamiento como en el futuro procesamiento, ya que se procesará una menor cantidad de información por cada imagen.

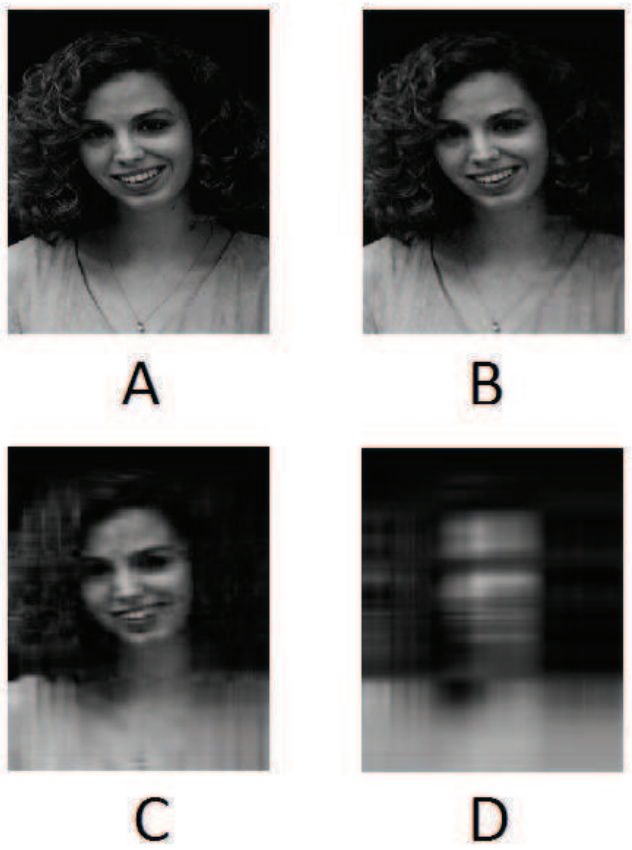


Figura 1.2: Imágenes en las que se evidencia la compresión mediante PCA, la imagen A es la original, la B usa 20 componentes principales, la C usa 10, y la D solamente 2.

Imagen A	Imagen B	Imagen C	Imagen D
78.3 KB	67.7 KB	47.6 KB	33.5 KB

Cuadro 1.1: Comparación de tamaños de una imagen aplicados procesos de reducción de tamaño.

En definitiva, cualquier estudio que involucre un procesamiento de imágenes, así como su tratamiento está dentro del análisis de imágenes, la definición que [22] plantea es:

"Análisis de Imágenes: Proceso mediante el cual a partir de una imagen se obtiene una medición, interpretación o decisión."

Tal como lo cita el mismo autor, el análisis de imágenes está ampliamente

presente en varios campos, es usado por ejemplo para la detección de fallos en productos, puede detectar mal formaciones mínimas que fácilmente se escaparían al ojo humano, o con un simple análisis del color se puede saber la calidad de un determinado alimento.

Años atrás, a la visión artificial se la consideraba como algo utópico, propio de la ciencia ficción. Conforme pasa el tiempo, las técnicas que emulan a la visión van mejorando, cada vez se aleja de la ciencia ficción para convertirse en realidad, estas técnicas ayudan enormemente a labores cotidianas, grandes proyectos existen en la actualidad que involucran directamente a la visión por computador, grandes investigaciones están en curso, este es un campo en el que queda aún mucho por descubrir.

1.2. Objetivos

1.2.1. Objetivo general

Aplicar la técnica de reducción de dimensionalidad ICA a la clasificación de imágenes y medir su efectividad para dicha tarea.

1.2.2. Objetivos específicos

- Estudiar las técnicas más importantes para la reducción de dimensionalidad.
- Analizar los campos de aplicación de las técnicas para la reducción de dimensionalidad en imágenes.
- Exponer detalladamente la técnica de Análisis de Componentes Independientes (ICA).
- Desarrollar un sistema que permita la aplicación de la técnica ICA a un proceso de clasificación de imágenes.
- Comparar los resultados obtenidos con ICA respecto al estado del arte.

1.3. Alcance

El proyecto de tesis apunta a desarrollar una serie de pruebas para exponer a detalle, primero el campo de uso en la reducción de dimensionalidad en una imagen, y luego, las bondades de ICA frente a otros procedimientos con el mismo fin. Para lograr unos resultados más novedosos, el resultado de esta tesis será la clasificación en categorías partiendo de la extracción de características de cada imagen perteneciente a un numeroso corpus.

Gran parte de las referencias que se han usado para este trabajo están escritas en un nivel altamente técnico, lo que dificultó en cierto nivel la investigación, por lo que se decidió que para cada técnica aquí expuesta se realizara un detallado estudio explicando paso a paso cada cálculo llevado a cabo.

El resultado final de esta tesis es la publicación de un artículo en el que se proponga una nueva técnica para la clasificación de imágenes, persiguiendo el objetivo de tener una alta tasa de precisión.

Capítulo 2

Técnicas de reducción de dimensionalidad

2.1. Álgebra presente reducción de dimensiones

2.1.1. Vectores

A un vector se lo representa con una letra minúscula, de la siguiente manera:

$$\overrightarrow{x} = [x_0, x_1, \cdots, x_n]$$

2.1.1.1. Tamaño de un vector

$$|\overrightarrow{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Por ejemplo, si se desea extraer el tamaño del siguiente vector:

$$\overrightarrow{a} = [1, 2, 3, 4, 5]$$

Entonces:

$$|\overrightarrow{a}| = \sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2}$$

$$|\overrightarrow{a}| = \sqrt{55}$$

$$|\overrightarrow{a}| = 7{,}42$$

2.1.1.2. Suma de vectores

La suma de vectores se la puede realizar entre 2 vectores que tienen el mismo número de términos. Por ejemplo, sumar los vectores $\overrightarrow{a}$ y $\overrightarrow{b}$ teniendo que:

$$\overrightarrow{a} = [3, 2, 1, -2] \text{ y } \overrightarrow{b} = [2, -1, 4, 1]$$

Entonces:

$$\overrightarrow{a} + \overrightarrow{b} = [3, 2, 1, -2] + [2, -1, 4, 1]$$
$$\overrightarrow{a} + \overrightarrow{b} = [(3+2), (2-1), (1+4), (-2+1)]$$
$$\overrightarrow{a} + \overrightarrow{b} = [5, 1, 5, -1]$$

2.1.1.3. Multiplicación de un escalar por un vector

$$\overrightarrow{a} = [1, 2, 3, 4]$$
$$\overrightarrow{a} \cdot 2 = [1, 2, 3, 4] \cdot 2$$
$$\overrightarrow{a} \cdot 2 = [1 \cdot 2, 2 \cdot 2, 3 \cdot 2, 4 \cdot 2]$$
$$\overrightarrow{a} \cdot 2 = [2, 4, 6, 8]$$

2.1.1.4. Producto escalar

El producto escalar, también conocido como "dot product" o "producto punto" es un valor que se obtiene al multiplicar 2 vectores realizando la siguiente operación:

$$\overrightarrow{a} \cdot \overrightarrow{b} = \sum_{i=1}^{n} a_i \cdot b_i$$

Por ejemplo, multiplicar los vectores $\overrightarrow{a}$ y $\overrightarrow{b}$ teniendo que:

$$\overrightarrow{a} = [1, 6, 7, 4] \text{ y } \overrightarrow{b} = [3, 2, 8, 3]$$

Entonces:

$$\overrightarrow{a} \cdot \overrightarrow{b} = [1, 6, 7, 4] \cdot [3, 2, 8, 3]$$
$$\overrightarrow{a} \cdot \overrightarrow{b} = 1 \cdot 3 + 6 \cdot 2 + 7 \cdot 8 + 4 \cdot 3$$
$$\overrightarrow{a} \cdot \overrightarrow{b} = 83$$

2.1.1.5. Vector Normal

Un vector normal es aquel cuyo tamaño es 1, es decir:

$$|\overrightarrow{x}| = 1$$

El siguiente vector es normal:

$$\overrightarrow{a} = [\tfrac{2}{5}, \tfrac{4}{5}, \tfrac{2}{5}, \tfrac{1}{5}]$$

Ya que:

$$|\overrightarrow{a}| = \sqrt{(\tfrac{2}{5})^2 + (\tfrac{4}{5})^2 + (\tfrac{2}{5})^2 + (\tfrac{1}{5})^2}$$
$$|\overrightarrow{a}| = \sqrt{\tfrac{25}{25}}$$
$$|\overrightarrow{a}| = 1$$

2.1.1.6. Vectores ortogonales

Dos vectores son ortogonales cuando su producto escalar es 0, su principal propiedad es que entre ambos existe perpendicularidad, por ejemplo, los siguientes vectores son ortogonales:

$$\overrightarrow{a} = [2, 1, -2, 4] \text{ y } \overrightarrow{b} = [3, -6, 4, 2]$$

ya que

$$\overrightarrow{a} \cdot \overrightarrow{b} = [2, 1, -2, 4] \cdot [3, -6, 4, 2]$$

$$\overrightarrow{a} \cdot \overrightarrow{b} = [(2 \cdot 3) + (1 \cdot (-6)) + (-2 \cdot 4) + (4 \cdot 2)]$$

$$\overrightarrow{a} \cdot \overrightarrow{b} = 0$$

2.1.1.7. Vectores ortonormales

Dos vectores son ortonormales si:

- Son ortogonales entre sí.
- Si cada uno de ellos es normal.

Por ejemplo, los siguientes vectores son ortonormales:

$$\overrightarrow{a} = [\tfrac{2}{5}, \tfrac{1}{5}, -\tfrac{2}{5}, \tfrac{4}{5}] \text{ y } \overrightarrow{b} = [\tfrac{1}{5}, \tfrac{2}{5}, \tfrac{4}{5}, \tfrac{2}{5}]$$

ya que:

$$|\overrightarrow{a}| = 1 \text{ y } \left|\overrightarrow{b}\right| = 1$$

y:

$$\overrightarrow{a} \cdot \overrightarrow{b} = 0$$

2.1.2. Matrices

Una matriz es un conjunto bidimensional de datos, cada uno de ellos identificado por una posición (x, y) o también (m, n) representando la fila y columna, respectivamente. A diferencia de un vector, una matriz es representada por una letra mayúscula, por ejemplo:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,0} & \cdots & \cdots & a_{m,n} \end{bmatrix}$$

2.1.2.1. Matriz cuadrada

Una matriz cuadrada es aquella que tiene igual número de filas y de columnas.

2.1.2.2. Transpuesta

La transpuesta de una matriz, representada como A^T que se lee "la transpuesta de A" es otra matriz que se obtiene al intercambiar las filas por las columnas de la primera matriz. Por ejemplo, de la matriz:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Su transpuesta sería:

$$A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 5 & 9 \end{bmatrix}$$

La transpuesta se rige a las siguientes propiedades:

- $(A^T)^T = A$
- $A^T + B^T = (A + B)^T$
- $(\overrightarrow{a} \cdot A)^T = \overrightarrow{a} \cdot A^T$
- $(A \cdot B)^T = B^T \cdot A^T$

2.1.2.3. Matriz identidad

Una matriz identidad, representada por la letra I es una matriz cuadrada de tamaño n, en la que solamente los elementos de su diagonal son 1, mientras que el resto de elementos es 0. Por ejemplo:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.1.2.4. Matriz Ortogonal

Es aquella que multiplicada por su correspondiente transpuesta da como resultado una matriz identidad, es decir:

$$A \cdot A^T = I$$

Al obtener la determinante de una matriz ortogonal podemos obtener como resultado:

- 1, lo que significa que es una matriz ortogonal directa.
- -1 indicando que es una matriz ortogonal inversa.

Además, se tiene que tanto la transpuesta como la inversa de una matriz ortogonal son iguales.

2.1.2.5. Matriz Diagonal

Es aquella que cuyos elementos por encima y por debajo de la diagonal principal son 0, por ejemplo:

$$A = \begin{bmatrix} a_{0,0} & 0 & 0 & 0 & 0 \\ 0 & a_{1,1} & 0 & 0 & 0 \\ 0 & 0 & a_{2,2} & 0 & 0 \\ 0 & 0 & 0 & a_{3,3} & 0 \\ 0 & 0 & 0 & 0 & a_{4,4} \end{bmatrix}$$

2.1.2.6. Eigenvectores y Eigenvalores

En español conocidos como vectores y valores propios respectivamente, un eigenvalue de una matriz cuadrada es un escalar representado por la letra griega λ, pronunciada lambda.

Un eigenvector es un vector representado comúnmente por la letra x. La principal característica de este vector es que debe ser distinto de 0.

Todos los eigenvalues y eigenvectors satisfacen la siguiente ecuación:

$A \cdot x = \lambda x$

Para una matriz cuadrada A, siendo el eigenvalue un escalar, no solamente puede adoptar números reales, sino que puede ser representado por números complejos.

Planteando una definición más formal:

Considerando la matriz cuadrada A, decimos que λ es un eigenvalue de A si existe un vector x distinto de 0 tal que satisface la ecuación $A \cdot x = \lambda x$. En este caso, a x se lo llama eigenvector. Al conjunto λ y x se lo conoce como eigenpair de A.

Por ejemplo: Se desea saber si

$$x = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

es un eigenvector correspondiente al eigenvalue $\lambda = 0$,de la matriz

$$A = \begin{bmatrix} 6 & 3 \\ -2 & -1 \end{bmatrix}$$

Empezamos reemplazando los valores en la fórmula:

$$A \cdot x = \lambda x$$

$$\begin{bmatrix} 6 & 3 \\ -2 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} = 0 \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} 6 \cdot 1 + 3 \cdot (-2) \\ (-2) \cdot 1 + (-1) \cdot (-2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Como acotación, ¿Por qué x debe ser un vector distinto de 0?, si tenemos un vector de 0, el resultado de la ecuación representaría un caso impráctico.

A la ecuación $A \cdot x = \lambda x$ se la puede escribir también de la forma $(A - \lambda \cdot I) \cdot x = 0$, en donde I es una matriz identidad del mismo tamaño que A. Se debe cumplir que $(A - \lambda \cdot I)$ sea no invertible.

Llamamos a $p(\lambda) = det(A - \lambda \cdot I)$ el polinomio característico. A partir de este polinomio podemos obtener los eigenvalues.

En el siguiente ejemplo vamos a obtener los eigenvalues de la siguiente matriz:

$$A = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}$$

Entonces, empezamos reemplazando en la fórmula:

$$p = det(A - \lambda \cdot I)$$

$$p(\lambda) = det(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - \lambda \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix})$$

$$p(\lambda) = det(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix})$$

$$p(\lambda) = det(\begin{bmatrix} (2 - \lambda) & -4 \\ -1 & (-1 - \lambda) \end{bmatrix})$$

$$p(\lambda) = (2 - \lambda) \cdot (-1 - \lambda) - (-4) \cdot (-1)$$

$$p(\lambda) = (-2 - 2\lambda + \lambda + \lambda^2) - 4$$

$$p(\lambda) = (\lambda^2 - \lambda - 6)$$

$$p(\lambda) = (\lambda - 3)(\lambda + 2)$$

$$\lambda_1 = 3 \quad \lambda_2 = -2$$

Se encontraron 2 eigenvalues, para cada uno de estos debemos obtener su correspondiente eigenvector, para lo cual usamos la ecuación $(A - \lambda \cdot I) \cdot x = 0$, entonces:

para $\lambda_1 = 3$:

$$(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - 3 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}) \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}) \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} (2-3) & -4 \\ -1 & (-1-3) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -4 \\ -1 & -4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -x_1 & -4x_2 \\ -x_1 & -4x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Tenemos 2 ecuaciones similares:

$$-x_1 - 4x_2 = 0 \text{ y } -x_1 - 4x_2 = 0$$

En la primera ecuación reemplazamos x_2 por t, entonces:

$$-x_1 - 4t = 0$$

$$x_1 = -4t$$

Y reemplazamos este valor en la segunda ecuación:

$$4t - 4x_2 = 0$$

$$4x_2 = 4t$$

$$x_2 = t$$

Entonces, para $\lambda_1 = 3$ encontramos que su eigenvalue es:

$$x = \begin{bmatrix} -4 \\ 1 \end{bmatrix}$$

ahora, para $\lambda_1 = -2$:

$$(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - (-2) \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}) \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix} - \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}) \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} (2+2) & -4 \\ -1 & (-1+2) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

8	2	5	7	2	6	7

Cuadro 2.1: Datos ficticios para extraer la media y la desviación típica

$$\begin{bmatrix} 4 & -4 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4x_1 & -4x_2 \\ -x_1 & x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Tenemos 2 ecuaciones :

$$4x_1 - 4x_2 = 0 \text{ y } -x_1 + x_2 = 0$$

En la primera ecuación reemplazamos x_2 por t, entonces:

$$4x_1 - 4t = 0$$

$$4x_1 = 4t$$

$$x_1 = t$$

Y reemplazamos este valor en la segunda ecuación:

$$-t + x_2 = 0$$

$$x_2 = t$$

Entonces, para $\lambda_1 = 2$ encontramos que su eigenvalue es:

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

2.1.3. Técnicas estadísticas

2.1.3.1. Desviación típica

Partiendo de un concepto sencillo, la desviación típica, o también llamada desviación estándar (representada con σ), mide cuanto se separan los datos de la media de los mismos. Para obtener la desviación típica, extraemos la raíz cuadrada de la varianza.

La varianza representada por σ^2 es la media de las diferencias con la media elevadas al cuadrado. Con el siguiente ejemplo explicaremos la obtención de estos valores.

Vamos a hallar la varianza y la desviación típica de los siguientes datos:

Partimos de la obtención de la media:

$$media = \frac{8+1+5+7+8+6+7}{7}$$

$$media = 6$$

Ahora obtenemos la varianza:

$$\sigma^2 = \frac{(8-6)^2+(1-6)^2+(5-6)^2+(7-6)^2+(8-6)^2+(6-6)^2+(7-6)^2}{7}$$

$$\sigma^2 = \frac{4+25+1+1+4+0+1}{7}$$

$$\sigma^2 = 5{,}14$$

Y por último, obtenemos la desviación típica:

$$\sigma = \sqrt{5{,}14}$$

$$\sigma = 2{,}267786$$

2.1.3.2. Covarianza

Es representada con σ_{xy}

Un dato interesante de la covarianza es que esta indica el sentido de correlación entre las variables estudiadas, y obedece a las siguientes propiedades:

- Si $\sigma_{xy} < 0$ entonces la correlaciones es inversa.
- Si $\sigma_{xy} > 0$ entonces la correlaciones es directa.
- Si $\sigma_{xy} = 0$ entonces se sabe que no existe una relación entre las variables involucradas.

2.1.3.3. Correlación

La correlación representa al nivel de dependencia existente entre 2 o más variables situadas en un plano bidimensional. En otros términos, permite determinar si el cambio en una variable afecta a otra dentro del mismo plano, si es este el caso se sabe que hay correlación entre ellas.

Existen 3 grados de correlación:

- Directa, en la que el aumento en una variable representa un incremento en las demás variables. Este tipo de correlación está representado como una recta creciente, de la siguiente manera:

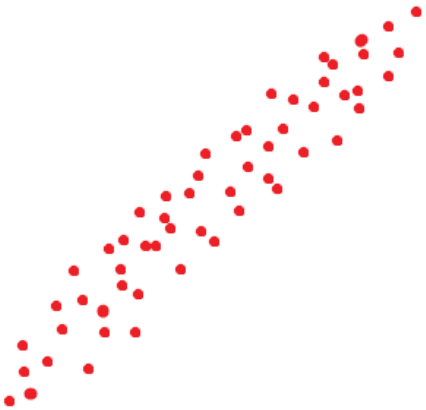


Figura 2.1: Correlación directa.

- Inversa, cuando el aumento de una variable desencadena el decremento de la otra, es representada como una recta decreciente:

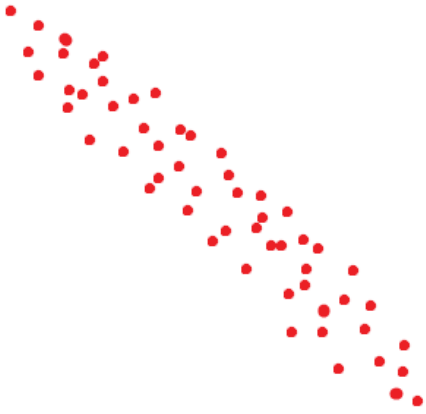


Figura 2.2: Correlación inversa.

- Nula: Se da cuando el cambio en una variable no representa cambio alguno en el resto de variables. Se la representa como una nube de puntos.

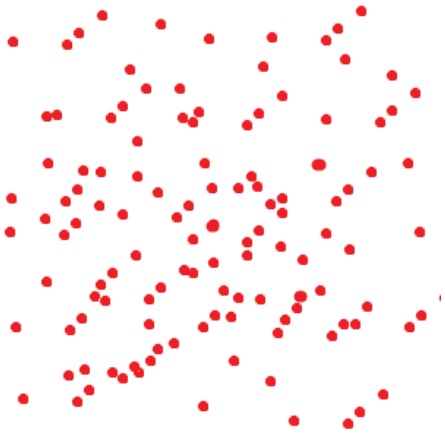


Figura 2.3: Correlación nula.

Además, existen 3 diferentes grados de correlación:

- Fuerte: Se da mientras las variables están más cerca de la recta.

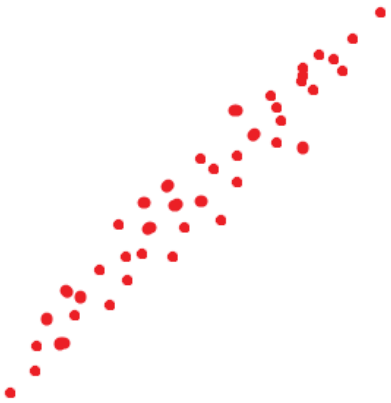


Figura 2.4: Correlación Fuerte.

- Débil: Contraria a la anterior, existen cuando los puntos están alejados de la recta.

Figura 2.5: Correlación débil.

Cabe indicar que en las imágenes anteriores, los puntos son valores estadísticos representados en un sistema de coordenadas.

Al coeficiente de correlación se lo expresa con la letra r, y la fórmula a usarse para calcularla es la siguiente:

$$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

Procederemos con un claro ejemplo, tomado de [28]. Hallar el coeficiente de correlación teniendo que las notas de 12 alumnos en 2 asignaturas son las siguientes:

Matemáticas	2	3	4	4	5	6	6	7	7	8	10	10
Física	1	3	2	4	4	4	6	4	6	7	9	10

Cuadro 2.2: Matriz de notas, usadas para un ejemplo.

Partimos realizando algunos cálculos que nos servirán más adelante, con la letra x representaremos a Matemáticas, y con la letra y a Física para una facilitar la comprensión:

x_i	y_i	$x_i \cdot y_i$	x_i^2	y_i^2
2	1	2	4	1
3	3	9	9	9
4	2	8	16	4
4	4	16	16	16
5	4	20	25	16
6	4	24	36	16
6	6	36	36	36
7	4	28	49	16
7	6	42	49	36
8	7	56	64	49
10	9	90	100	81
10	10	100	100	100
72	60	431	504	380

Cuadro 2.3: Matriz de resultados en cálculos previos.

La última fila corresponde a los totales de cada columna. Ahora, calculamos las medias aritméticas:

$$\bar{x} = \frac{72}{12} \quad \bar{y} = \frac{60}{12}$$

$$\bar{x} = 6 \quad \bar{y} = 5$$

Procedemos a calcular la covarianza:

$$\sigma_{xy} = \frac{431}{12} - 6 \cdot 5$$

$$\sigma_{xy} = 5.92$$

Encontramos las desviaciones típicas:

$$\sigma_x = \sqrt{\frac{504}{12} - 6^2} \quad \sigma_y = \sqrt{\frac{380}{12} - 5^2}$$

$$\sigma_x = 2{,}45 \quad \sigma_y = 2{,}58$$

Y por último, obtenemos el coeficiente de correlación:

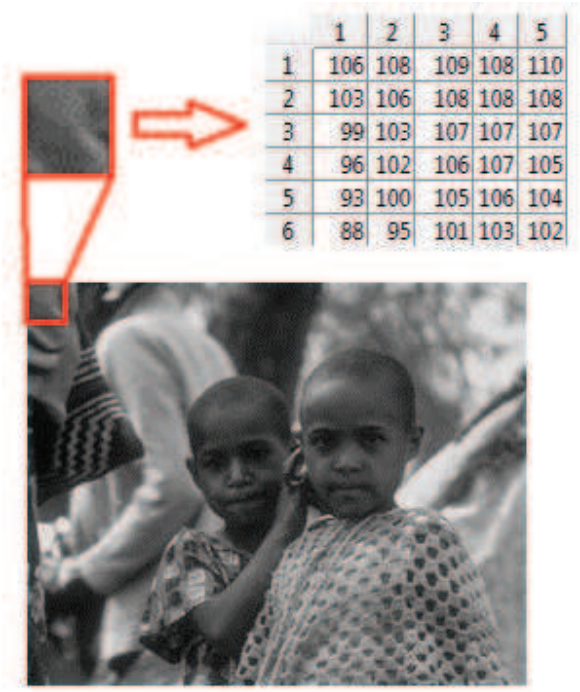
$$r = \frac{5{,}92}{2{,}45 \cdot 2{,}58}$$

$$r = 0{,}94$$

Tomando este coeficiente de correlación, podemos decir que:

- Debido a que es muy cercano a 1, la correlación es muy fuerte.
- Debido a que es positivo, la correlación es directa.

Un ejemplo más cercano a nuestras necesidades, es decir, al análisis de imágenes se expone a continuación, una imagen al final es una matriz bidimensional, es decir, una matriz compuesta de valores como:



	1	2	3	4	5
1	106	108	109	108	110
2	103	106	108	108	108
3	99	103	107	107	107
4	96	102	106	107	105
5	93	100	105	106	104
6	88	95	101	103	102

Figura 2.6: Comúnmente una imagen está representada de esta forma. En la figura vemos una imagen en escala de grises, cada pixel posee un valor entre 0 y 255, como se ha dicho antes, una imagen es una matriz de pixeles.

Para el ejemplo, unos valores que se encuentran en la posición x, y son los siguientes, nótese que x_i y y_i son las coordenadas de un pixel, mientras que f_i es el valor que dicho pixel tiene.

x_i	y_i	f_i
0	1	2
0	2	1
0	3	2
2	1	1
2	2	4
2	3	5
4	1	3
4	2	2

Cuadro 2.4: Matriz parecida a la que comúnmente puede representar a una imagen.

Empezamos realizando cálculos previos:

x_i	y_i	f_i	$x_i \cdot f_i$	$x_i^2 \cdot f_i$	$y_i \cdot f_i$	$y_i^2 \cdot f_i$	$x_i \cdot y_i \cdot f_i$
0	1	2	0	0	2	2	0
0	2	1	0	0	2	4	0
0	3	2	0	0	6	18	0
2	1	1	2	4	1	1	2
2	2	4	8	16	8	16	16
2	3	5	10	20	15	45	30
4	1	3	12	48	3	3	12
4	2	2	8	32	4	8	16
		20	40	120	41	97	76

Cuadro 2.5: Cálculos realizados en la matriz anterior.

Calculamos las medias aritméticas:

$$\bar{x} = \frac{40}{20} \quad \bar{y} = \frac{41}{20}$$

$$\bar{x} = 2 \quad \bar{y} = 2{,}05$$

Calculamos las desviaciones típicas:

$$\sigma_x^2 = \frac{120}{20} - 2^2 \quad \sigma_y^2 = \frac{97}{20} - 2{,}05^2$$

$$\sigma_x^2 = 2 \quad \sigma_y^2 = 0{,}65$$

$$\sigma_x = \sqrt{2} \quad \sigma_y = \sqrt{0{,}65}$$

$$\sigma_x = 1{,}41 \quad \sigma_y = 0{,}81$$

Calculamos la covarianza:

$$\sigma_{xy} = \frac{76}{20} - 2 * 2{,}05$$

$$\sigma_{xy} = -0{,}3$$

Por último, obtenemos el coeficiente de correlación:

$$r = \frac{-0{,}3}{1{,}41 \cdot 0{,}81}$$

$$r = -0{,}26$$

Con lo que concluimos que:

- Siendo negativo el coeficiente de correlación, esta es inversa.
- Al ser cercano a 0, la correlación es débil.

2.2. Descomposición en Valores Singulares

2.2.1. Definición

Entre las técnicas para el procesamiento de imágenes digitales encontramos a la descomposición en valores singulares, a la que llamaremos SVD por sus siglas en inglés. Específicamente en la compresión de imágenes es en donde se usa SVD, siendo el objetivo transmitir una imagen reduciendo su tamaño pero al mismo tiempo perdiendo la menor cantidad de información.

[25] plantea una definición mas formal: de una matriz A de n filas por m columnas, a partir de sus eigenvalues pertenecientes a la matriz obtenida de $A^T \cdot A$ podemos obtener sus valores singulares representados por σ calculando la raíz cuadrada de los mismos. Estos eigenvalues siempre serán positivos para poder obtener su raíz cuadrada. Por cada eigenvalue se obtiene un valor singular, a los que ordenaremos de forma descendente, es decir:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$$

La matriz A puede ser escrita de la forma:

$$A = U \cdot \Sigma \cdot V^T$$

En donde U es una matriz ortogonal de n filas y n columnas, V es una matriz ortogonal de m filas y m columnas, y Σ es una matriz de n filas por m columnas cuya diagonal es formada por los valores singulares de A y sus otros valores son 0. Explicado todo esto, a la expresión $U \cdot \Sigma \cdot V^T$ se la llama la descomposición en valores singulares de A.

2.2.2. Ejemplo

Obtener la descomposición en valores singulares de la siguiente matriz:

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ \sqrt{3} & 0 \end{bmatrix}$$

Empezamos multiplicando la transpuesta de la A por A, recordamos que si cambiamos este orden, el resultado obtenido será diferente:

$$A^T \cdot A = \begin{bmatrix} 1 & 1 & \sqrt{3} \\ -1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ \sqrt{3} & 0 \end{bmatrix}$$

$$A^T \cdot A = \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix}$$

De esta matriz obtenida, calculamos sus eigenvalues:

$$det\left(\begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$det\left(\begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = 0$$

$$det\left(\begin{bmatrix} 5-\lambda & -2 \\ -2 & 2-\lambda \end{bmatrix}\right) = 0$$

$$(5-\lambda)(2-\lambda) - (-2 \cdot 2) = 0$$

$$10 - 5\lambda - 2\lambda + \lambda^2 - 4 = 0$$

$$\lambda^2 - 7\lambda + 6 = 0$$

$$(\lambda - 6)(\lambda - 1) = 0$$

$$\lambda_1 = 1 \quad \lambda_2 = 6$$

Al tener los eigenvalues, podemos inmediatamente obtener los valores singulares:

$$\sigma = \sqrt{\lambda}$$

$$\sigma_1 = \sqrt{\lambda_1} \quad \sigma_2 = \sqrt{\lambda_2}$$

$$\sigma_1 = 1 \quad \sigma_2 = \sqrt{6}$$

Ahora podemos formar a la matriz Σ:

$$\Sigma = \begin{bmatrix} \sqrt{6} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Ponemos en ese orden a los valores singulares por lo previamente explicado.

2.2.3. Aplicaciones de SVD

Como se mencionó en la definición, en el tratamiento de imágenes digitales se puede encontrar el uso de SVD, específicamente en la compresión de imágenes.

Sabemos que toda imagen puede ser representada por una matriz de valores, por lo que dada una imagen en escala de grises, tenemos que cada posición de la matriz indica el nivel de gris del correspondiente pixel, este nivel está comprendido entre 0 y 255.

Al aplicar SVD en esta matriz, podemos definir cuál es la cantidad de información de la matriz que podríamos obviar, sin que este signifique la perdida de información importante, como la nitidez en la imagen [10].

2.3. Análisis de Componentes Principales

2.3.1. Definición

PCA (por sus siglas en inglés, Principal Component Analysis) está entre los métodos que más frecuentemente se usan para la extracción de características. Se trata básicamente de una transformación lineal.

Su objetivo principal [14] es tomar un espacio de representación P y transformarlo en un nuevo espacio P' en el que sus datos no estén correlacionados. La matriz de covarianza presente en P' será diagonal. Se encontrará también un nuevo conjunto de ejes ortogonales cuya varianza será la máxima en alguna dirección.

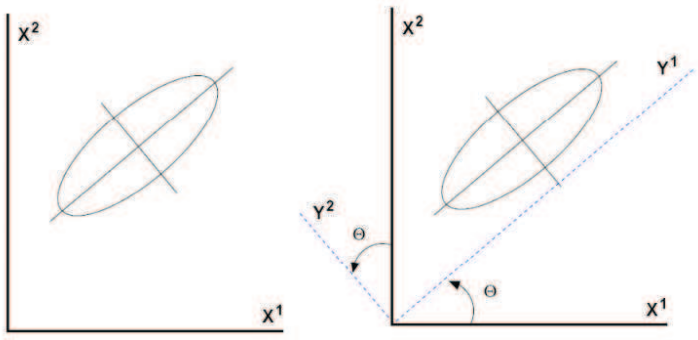


Figura 2.7: Rotación de ejes para PCA. Fuente: [14].

Se sabe que gran parte de la información está presente en el eje principal Y_1, esto permitiría una selección de características

Los nuevos ejes guardan una relación con los antiguos, que es la siguiente:

$$Y_1 = W_{11}X_1 + W_{12}X_2$$

$$Y_2 = W_{21}X_1 + W_{22}X_2$$

Los nuevos ejes se encuentran uno a continuación de otro, el primero, Y_1, es el que representa la mayor varianza entre todos los nuevos ejes posibles, y el segundo, Y_2, es perpendicular al primero. Estos ejes son ortogonales entre sí y definen a P', y todos los datos en P' están no correlacionados. El primer objetivo para encontrar P' será definir a W_{ij}.

Para empezar con el análisis de PCA, supongamos que tenemos una matriz X de N columnas a las que llamaremos vectores y de M filas las que representarán al tamaño de cada vector:

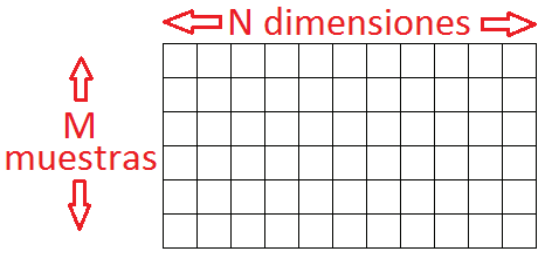


Figura 2.8: Definición de una matriz.

La proyección del vector x_j sobre un vector u se la calcula con:

$$p_j = \overrightarrow{u}^T \cdot \overrightarrow{x_j}$$

$$p_j = \sum_{i=1}^{M} u_i \cdot x_{ij}$$

Lo que requerimos es obtener una función para poder maximizarla:

$$J^{PCA}(\vec{u}) = \sigma^2(p_j)$$

$$J^{PCA}(\vec{u}) = \frac{1}{N}\sum_{j=1}^{N}(p_j - \overline{p})^2$$

$$J^{PCA}(\vec{u}) = \vec{u}^T C \vec{u}$$

En donde u es un vector unitario, y C es la matriz de covarianzas de la matriz X con las que originalmente estamos trabajando.

Siendo u un vector normal, lo que se desea encontrar es la dirección u, la misma que permite maximizar la varianza de las proyecciones de cada vector x_j sobre la dirección.

En la función obtenida, para continuar con la maximización procedemos a usar los multiplicadores de Lagrange. Ahora calcularemos los eigenvalues y eigenvectors de la matriz de covarianzas C, los que presentan la solución al problema de maximización.

Aquí necesitamos diagonalizar la matriz C, para esto usamos la técnica SVD, que vimos anteriormente, aplicamos SVD a la matriz de covarianza C:

$$C = U\Lambda U^T$$

En donde U esta compuesta por los Eigenvectores $u_1, u_2, \cdots, u_N$ como columnas, y la matriz diagonal Λ formada por los eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_N$.

Para clasificarlos, se toma a los eigenvalues y se los ordena decrecientemente: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$

Logrando que la mayor variabilidad de las entradas quede contenida en los primeros eigenvectors. A este conjunto de eigenvectors se los llama Vectores Principales.

2.3.2. Aplicaciones de PCA

Entre las técnicas para la extracción de características, PCA es una de las más populares, siendo ampliamente usada en el reconocimiento de rostros. Para esta práctica, se realizará el reconocimiento basado en la apariencia.

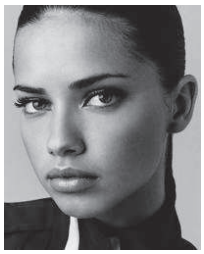


Figura 2.9: PCA tiene alto potencial en la detección de rostros. Fuente: Wikipedia, Adriana Lima.

Al aplicar técnicas a imágenes, básicamente estamos trabajando con vectores bidimensionales de datos, siendo el valor de cada pixel de la imagen el valor de su correspondiente posición en el vector. Una imagen tiene f filas y c columnas, creando un espacio de m-dimensiones, teniendo que $m = f \cdot c$. Dado este caso, la imagen se convierte en un vector de alta dimensión requiriendo un alto esfuerzo para su procesamiento. En lugar de trabajar con el vector completo, podemos usar un clasificador, el más recomendable en este caso el de el vecino más cercano.

Procediendo con el ejemplo, tenemos un rostro y queremos saber a cuál se parece más de un grupo de n rostros que previamente se han establecido:

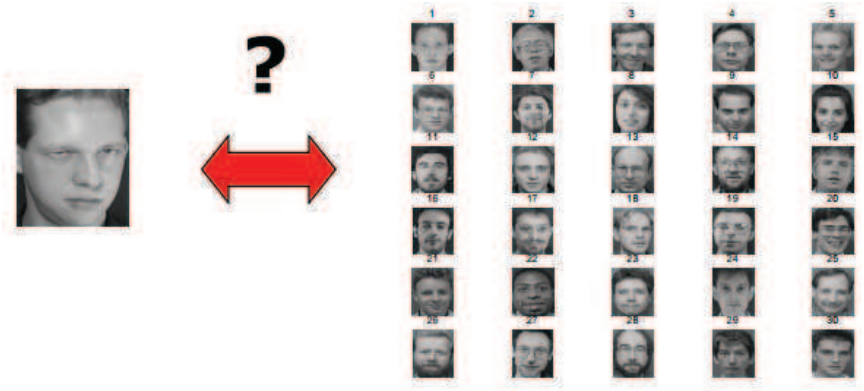


Figura 2.10: De entre un set de rostros, se desea descubrir a cuál se asemeja más un nuevo rostro. Fuente: [14].

Con PCA lo que hacemos es obtener las características principales del dato de entrada, en este caso la imagen, y mediante la técnica del vecino más cercano podremos reconocer a quién pertenece ese rostro:

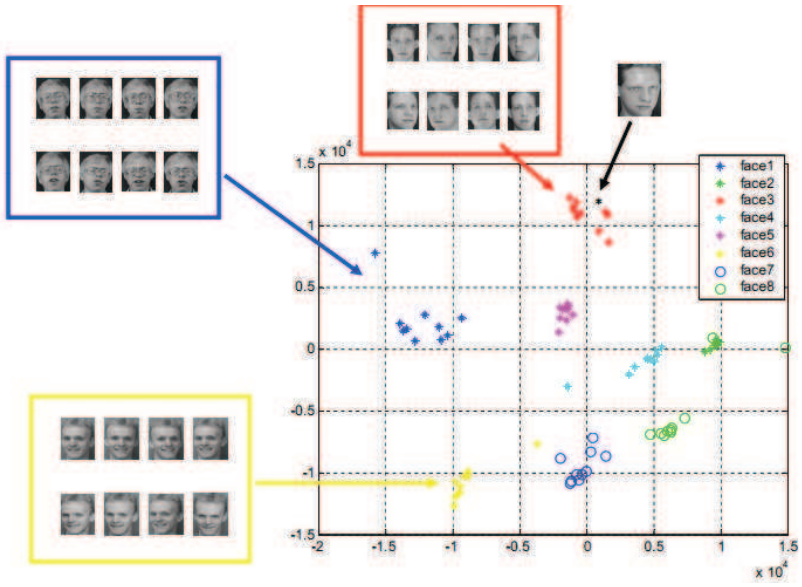


Figura 2.11: Mediante la técnica de los K-vecinos podemos agrupar a los rostros pertenecientes a una persona, y cuando necesitemos comprobar un nuevo rostro, lo ubicamos en el plano y vemos cuál es el grupo más cercano Fuente: [14].

2.4. Análisis de Componentes Independientes

2.4.1. Definición

Al igual que las otras técnicas estudiadas en esta investigación, el análisis de componentes independientes (ICA por sus siglas en inglés) posee un gran campo de aplicación, por citar algunos de ellos:

- Reducción de dimensionalidad [11].
- Separación de señales [4] [3].
- Análisis de imágenes médicas [27].
- Reconocimiento de patrones y extracción de características [13] [15].

Una de las ramas más ampliamente investigadas, y precisamente la que más nos puede interesar es la aplicación de ICA en el tratamiento de imágenes. ICA está altamente relacionado con la técnica llamada "Separación Ciega de fuentes" o BSS por sus siglas en inglés, la cual explicaremos en la siguiente sección.

2.4.2. Separación ciega de fuentes

Un ejemplo clásico para la explicación de la BSS es el "Cocktail Party" [6], en donde se cuenta con varias personas hablando en una habitación, al mismo tiempo que existen varios sensores que captarán las voces de estas personas. Las señales captadas por estos sensores se consideran la mezcla entre las señales originales o fuentes, y el ruido ajeno a estas fuentes. Vamos a denotar a la señal captada por un sensor con la letra x, para nuestro ejemplo supondremos que tenemos 3 sensores, por lo tanto, las señales serian $x_1(t)$, $x_2(t)$ y $x_3(t)$, mientras que cada señal original es representada con la letra s, por lo tanto tenemos $s_1(t)$, $s_2(t)$ y $s_3(t)$. Teniendo la anterior, podemos formar el siguiente sistema:

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t)$$

$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t)$$

$$x_3(t) = a_{31}s_1(t) + a_{32}s_2(t) + a_{33}s_3(t)$$

O de forma más general:

$$x = A \cdot s$$

En donde cada coeficiente $a_{i\,j}$pertenece a la matriz A, que a su vez es considerada como la matriz de mezcla, en otras palabras, son los valores que representan a todo el ruido que mezclado con la señal original dan como resultado la salida x_i captada por los sensores.

De todas las variables mencionadas, lo único que conocemos son las señales entregadas por los sensores, no sabemos cuáles son las fuentes ni mucho menos

los valores de mezcla, el objetivo es el de encontrar las señales originales, de aquí el nombre "Ciega", debido a que es muy poco lo que conocemos para proceder con el cálculo.

Continuando con el procedimiento para BSS, según [6], la matriz A está compuesta por coeficientes que la hacen invertible. Con esto creamos una matriz W que es la inversa de A y esta compuesta por coeficientes w_{ij}, esta matriz nos permitirá separar a las señales $s_i(t)$ de la siguiente manera:

$$s_1(t) = w_{11}x_1(t) + w_{12}x_2(t) + w_{13}x_3(t)$$

$$s_2(t) = w_{21}s_1(t) + w_{22}x_2(t) + w_{23}x_3(t)$$

$$s_3(t) = w_{31}x_1(t) + w_{32}x_2(t) + w_{33}x_3(t)$$

Representable también de la forma:

$$s = W \cdot x$$

Esta representación es similar a la que teníamos en un inicio. A cada señal $x_i(t)$, $t = 1, 2, \cdots, T$ se considera como una variable aleatoria x_i, el valor de esta variable está dado por la amplitud de su correspondiente señal en un determinado momento.

Básicamente el modelo que se sigue en la BSS es el siguiente:

Figura 2.12: Proceso que siguen las fuentes, empezando con su obtención, alterándose con la matriz de mezcla, el proceso de observación, el proceso para separación y una obtención estimada de los orígenes. Fuente: [6].

El objetivo es obtener a los coeficientes de la matriz W, la misma que nos permitirá obtener una aproximación bastante cercana de las señales originales. La solución a este problema es simplemente plantearnos el escenario en el que estimamos que existe independencia estadística entre las señales de origen. Se sabe que si las señales son no Gaussianas, nos bastará para obtener los coeficientes w_{ij}, planteando:

$$y_1(t) = w_{11}s_1(t) + w_{12}s_2(t) + w_{13}s_3(t)$$

$$y_2(t) = w_{21}s_1(t) + w_{22}s_2(t) + w_{23}s_3(t)$$

$$y_3(t) = w_{31}s_1(t) + w_{32}s_2(t) + w_{33}s_3(t)$$

Que es lo mismo que:

$$y = W \cdot x$$

Así que si $y_1(t)$, $y_2(t)$ y $y(t)$ son independientes entre sí entonces son iguales a las señales de origen $s_1(t)$, $s_2(t)$ y $s_3(t)$. Con esta información podemos obtener a la matriz W, y posteriormente podremos estimar a las señales originales, cumpliendo el objetivo de BSS.

2.5. Comparación entre técnicas planteadas

Cada una de las técnicas planteadas en la sección anterior tienen fortaleza en ciertos campos, y debilidades en otros.

Existe una gran cantidad de implementaciones de las técnicas antes mencionadas, cada una defiende el uso de las mismas, por lo que a continuación exponemos sus funciones específicas:

- SVD:
  - Tanto en SVD como en PCA, la matriz de entrada es una imagen [26].
- PCA:
  - Disminuir el rango de error en los datos comprimidos [2].
  - Elimina los datos que resultan redundantes [7].
  - Trabaja solamente con datos estadísticos de primer y segundo orden [7].
  - Busca proyectar los datos en un nuevo sistema de ejes, con el fin de que los componentes presentes en el nuevo sistema no estén correlacionados [7].
- ICA
  - Disminuye la dependencia estadística entre los vectores de entrada [2].
  - Usa a PCA en una etapa de preprocesamiento [2].
  - El procesamiento para ICA es mucho más exigente que las otras 2 técnicas [2].
  - Obtiene uno a uno los componentes independientes [7].

En algunos estudios distintos [12] [26] se trabaja en una comparación más específica, usan ICA, PCA y otros algoritmos para la detección de rostros y expresiones faciales, y la precisión obtenida principalmente entre ICA y PCA es bastante parecida, es decir, la diferencia no es mayor como para optar por una técnica o la otra. Los autores de dichos trabajos resaltan que la interpretación de dichos resultados es compleja, ya que existen una diversidad de factores que dan a una técnica mas ventaja sobre la otra, factores como luminosidad, cercanía, etc.

Durante la investigación nos encontramos con una interesante implementación [5], se trata de un trabajo en Matlab que realiza 4 experimentos para comparar la efectividad de PCA o de ICA, cada experimento trabaja con distintos datos de entrada, este experimento concluye en la fortaleza que tienen las técnicas en diversos escenarios, por lo que dependiendo de la investigación en la que se trabaje, se puede partir de estos estudios para saber por cuál técnica optar para obtención de resultados.

Capítulo 3

Análisis y diseño del prototipo

3.1. Diseño del plan de experimentación

En el transcurso de la realización de este proyecto, se investigaron varias técnicas que involucran a los componentes independientes en la clasificación de imágenes. Algunas de las que relatamos a continuación fueron pensadas por iniciativa del autor y del director de tesis, otras fueron implementadas a partir de artículos relacionados con el tema y unas últimas pruebas fueron una mezcla entre técnicas de artículos e iniciativa. No todas las pruebas citadas a continuación entregaron los resultados esperados, la prueba final es la que mejores resultados produjo.

Algo bastante importante de mencionar es que el algoritmo más usado para la extracción de ICs es FastICA. Este algoritmo lo usamos basándonos en:

- [23] menciona que las iteraciones producidas por FastICA es entre 10 y 100 veces más rápido que los demás algoritmos para la extracción de ICs.
- Existe ya una robusta implementación de FastICA para Matlab, de la misma que comprobamos su fácil uso y perfecta adecuación para los experimentos.

A continuación, las pruebas realizadas:

3.1.1. Cálculo de la distancia Euclidea entre componentes independientes de cada imagen

Hemos partido de lo más sencillo, extraer los ICs de cada imagen y posterior a esto comparar cada imagen de entrenamiento contra cada imagen de prueba.

Para esta prueba, trabajamos con la base de datos Corel , como citamos posteriormente, esta base de datos tiene 10 categorías, cada una compuesta por 100

imágenes. Lo primero que necesitamos es obtener un conjunto de entrenamiento y otro de prueba. Basados en trabajos anteriores, esta división se hizo tomando el 30 % de imágenes para entrenamiento y el sobrante 70 % para prueba, teniendo en total 300 y 700 imágenes respectivamente, vale la pena acotar que de cada categoría fueron 30 las imágenes para entrenamiento y 70 para pruebas. Una vez hecha esta división se procedió a obtener los componentes independientes de cada imagen de cada conjunto.

En capítulos anteriores se explicaba el concepto de dimensiones y número de muestras que eran representadas por los componentes independientes, recordando, tenemos que cada columna representa a una dimensión, mientras que el total de filas representan al número de muestras por dimensión.

Una vez obtenida la matriz de cada imagen de entrenamiento y de prueba, se procedió a comparar cada imagen de prueba contra cada imagen de entrenamiento, es decir, que cada imagen de prueba se comparaba 300 veces extrayendo la distancia Euclidea en cada comparación. En cada iteración se seleccionaba la menor distancia obtenida, teniendo así la imagen de entrenamiento más parecida a la imagen de prueba usada en esa comparación.

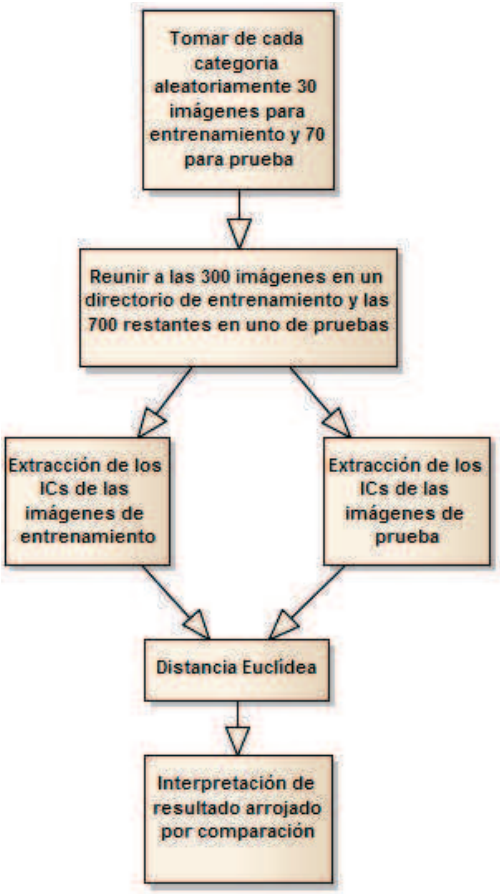


Figura 3.1: Proceso para la realización del experimento 1.

3.1.2. Extracción de componentes independientes de imágenes de texturas y posterior comparación con imágenes naturales.

Este experimento está inspirado en lo encontrado en [23], comparado con el experimento anterior, lo aquí citado resulta bastante más complejo.

Partimos del uso de varias imágenes del álbum de Brodatz explicado en una sección posterior. Resumiendo este experimento, se basa en la extracción de ICs de determinadas texturas, y posteriormente, en base a estos con cada imagen de texturas formar lo que [23] define como vector de características, al cual llamaremos VC a partir de este punto. El mismo proceso se lo hace con las imágenes de prueba, en este caso las imágenes de la base de datos Corel. Los VCs de las imágenes de textura forman al conjunto de entrenamiento, mientras

que los VCs de las imágenes de la base de datos Corel formarán al conjunto de pruebas. El último paso es extraer la distancia Euclidea de cada imagen y en base a los resultados saber a qué categoría pertenece cada imagen de prueba.

A continuación se procede a explicar detalladamente este experimento. Partimos de usar Q imágenes del álbum de Brodatz, representadas en la figura 3.6, se sabe que cada imagen es tiene 640 dimensiones con 640 muestras cada una, es decir $N\,x\,N$. En un paso previo a la extracción de ICs usando FastICA, se usa PCA para la reducción de dimensiones, en cada imagen obtenemos M dimensiones y mantenemos las N muestras. Posterior a esto se calculan los ICs de la matriz obtenida en el paso anterior. Para finalizar la parte de la extracción de ICs, de las N muestras tomaremos solamente los O primeros valores por cada dimensión, formando así la matriz a la que llamaremos IC_Q en la que Q es la imagen de la que obtuvimos los ICs, que nos servirá en los siguientes pasos. El porqué de esto quedará aclarado más adelante. Resumiendo, hasta aquí tenemos por cada imagen de textura una matriz de M dimensiones con O muestras cada dimensión. Todo este primer procedimiento se ve reflejado en la figura 3.2.

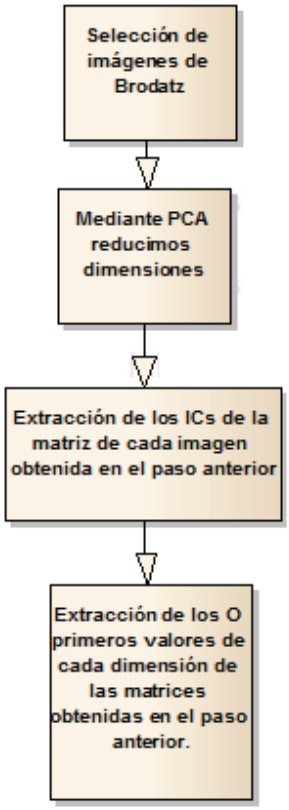


Figura 3.2: Proceso para la extracción de los ICs para cada textura en el experimento 2.

El siguiente paso se lo realiza usando las mismas Q imágenes del paso anterior, de cada una de estas imágenes de entrenamiento, obtenemos una subimagen de $P\,x\,P$ pixeles, siendo esta tomada de cualquier sección de la imagen. A esta matriz obtenida, la convertimos en un vector columna, que es lo mismo que una matriz de P^2 filas y 1 columna. El artículo seguido no menciona que tipo de barrido se usa para formar este vector, en este experimento se tomó la libertad de usar un barrido vertical para formar el vector, es decir una columna a continuación de la otra, esto quedará más claro si se observa la figura 3.3.

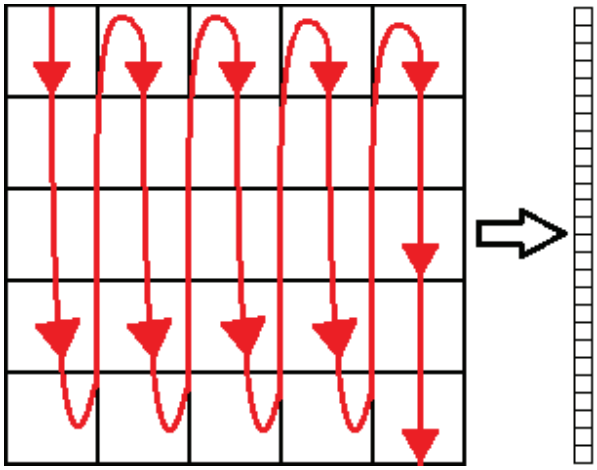


Figura 3.3: Barrido vertical de una imagen.

Con este vector realizamos un producto punto, visto en el capítulo 2 sección 1, con cada dimensión obtenida de cada imagen en el paso anterior, lo que nos dará un total de $M\,x\,Q$ valores, de los cuales obtenemos su valor absoluto, y por último obtenemos los 3 valores más altos. Con estos 3 valores formamos un vector, siendo este el VC de la imagen procesada. El diagrama en la figura 3.4 ilustra al proceso para la obtención del VC de una imagen.

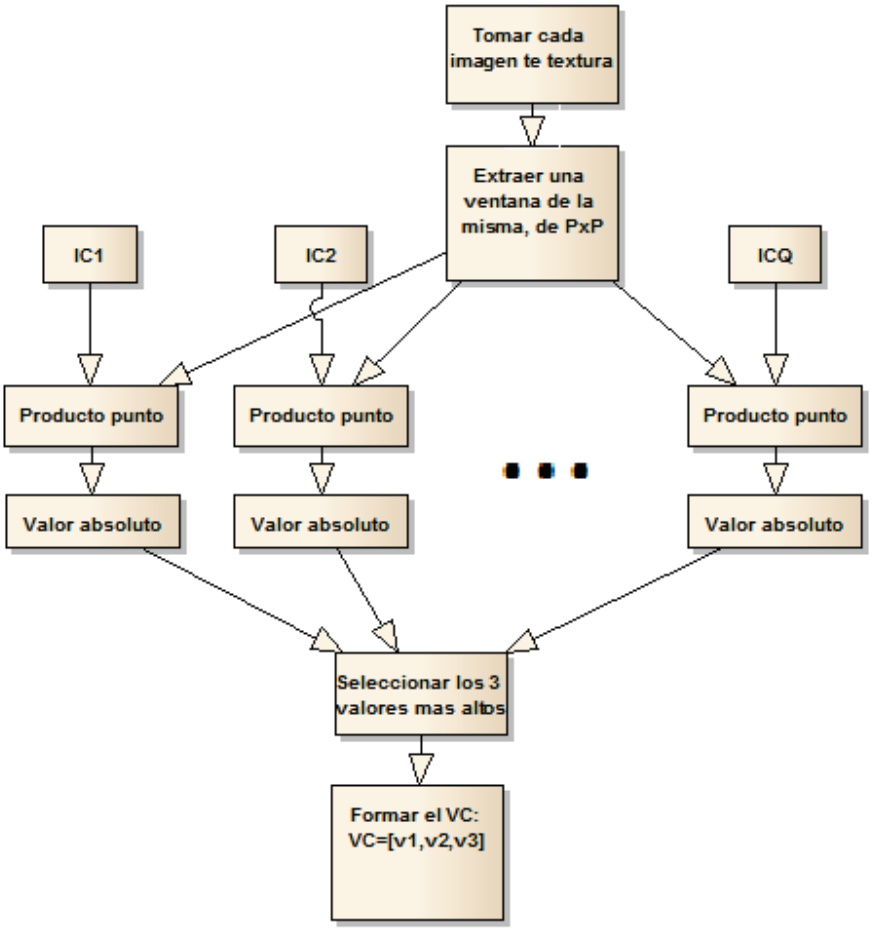


Figura 3.4: Obtención de los VCs de cada imagen en el experimento 2.

Lo siguiente será realizar lo mismo del paso anterior, pero ahora con las imágenes de prueba, en este caso las de la base de datos Corel . Entonces, hasta aquí tenemos un VC por cada imagen tanto de entrenamiento como de prueba.

Lo último será comparar cada VC de prueba contra cada VC de entrenamiento mediante distancia Euclidea, y buscar las imágenes más cercanas. El objetivo de todo esto es el buscar qué textura está más presente en cada imagen, por ejemplo, una imagen de una playa tendrá muchas similitudes con la textura arena, pero al mismo tiempo estarán presentes varias otras texturas, al realizar la comprobación, se apreciará que hay más arena en la imagen, por lo que se podrá suponer que dicha imagen es de una playa.

3.1.3. Extracción de componentes independientes de imágenes de texturas y posterior comparación con imágenes de las mismas texturas.

Partiendo del experimento anterior, se planteó una variante al mismo, que consiste en no comparar contra las imágenes de la base de datos de Corel, sino contra las mismas texturas. El procedimiento es similar en la extracción de VCs de cada imagen tanto de entrenamiento como de prueba, si recordamos en la sección anterior, de cada imagen se extrae una subimagen de $P\,x\,P$ de una sección cualquiera de la imagen, en otras palabras, desde el pixel (i, j) de la imagen extraemos una subimagen de tamaño $P\,x\,P$, todo esto con cada imagen de entrenamiento. Con las imágenes de prueba hacemos lo mismo, solamente que trabajamos extrayendo subimágenes desde un pixel (i, j) distinto al usado en la imagen de entrenamiento.

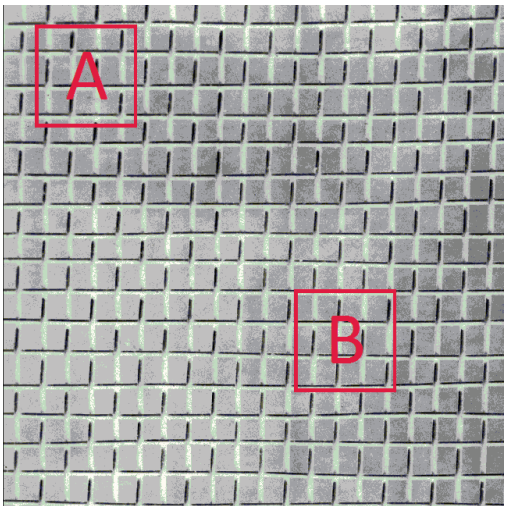


Figura 3.5: A es la subimagen de la que se extraerá el VC de entrenamiento, mientras que B servirá para el VC de prueba, se debe tener en cuenta que A y B son del mismo tamaño.

Finalizando esta prueba, se obtiene asimismo las imágenes más cercanas usando la distancia Euclidea. El resultado esperado de este experimento es una perfecta comparación. Explicando de otra manera, los VCs de entrenamiento representan a cada textura, si contra todos ellos se compara un VC de la imagen de entrenamiento 1 el resultado debería emparejarlo con el VC de entrenamiento de la figura 3.5, a pesar de que ambos VCs trabajaron con subimagenes tomadas de distintas posiciones.

3.1.4. Creación de vectores característicos de imágenes y posterior comparación con imágenes de la misma y de distintas categorías.

Este último experimento está inspirado en los 2 anteriores, en lugar de trabajar con texturas, trabajamos con imágenes de Corel.

- Empezamos seleccionando una de las 10 categorías que componen a Corel.
- Seleccionamos cierta cantidad de imágenes para crear el conjunto de entrenamiento.
- Al igual que en el procesamiento de texturas, trabajamos solamente con 40 dimensiones por cada imagen, de las cuales extraemos sus ICs mediante FastICA.
- Formamos los conjuntos de pruebas, para esto lo que hacemos es crear 10 subconjuntos, estos subconjuntos contienen una imagen de cada categoría del Corel, es decir, cada subconjunto tendría 10 imágenes.
- Estos subconjuntos deberán cumplir las siguientes propiedades:
  - Ningún subconjunto tendrá una imagen que esté presente en otro subconjunto.
  - Ningún subconjunto tendrá una imagen presente en el conjunto de imágenes usado para entrenamiento.
- De cada imagen de los subconjuntos extraemos sus respectivos VCs.
- Mediante distancia Euclidea los comparamos con los VCs de las imágenes de entrenamiento, obteniendo las distancias más cortas.

El objetivo de este test es el de realizar 10 pruebas, una con cada subconjunto. Lo que se espera es que el algoritmo planteado indique cuál de esas 10 imágenes pertenece a la categoría usada para el entrenamiento, por ejemplo, si se usa la categoría 2 para llenar el conjunto de entrenamiento, al comparar sus VCs con cada imagen de las 10 que conforman cada subconjunto de pruebas, el resultado debería ser una imagen de la misma categoría 2.

3.2. Preparación del Corpus

La gran cantidad de estudios e implementaciones de sistemas que procesan imágenes demandan en ciertos casos etapas de entrenamiento con imágenes que representan escenarios o elementos específicos. Un claro ejemplo de lo anterior puede ser un sistema de detección de rostros, el mismo para cumplir su objetivo deberá aprender de una gran cantidad de imágenes que sean solamente rostros. Los trabajos que requieren bases de datos de imágenes muy específicas cuentan con repositorios ya existentes de dichas imágenes, es decir, existen conjuntos

de imágenes que son destinadas a ser usadas en implementaciones bastante particulares. Apegándonos al ejemplo que citamos, existen varias bases de datos de imágenes compuestas solamente por rostros, existen las que los representan desde distintos ángulos, hay otras que representan distintas expresiones faciales, etc.

Para nuestro caso, debido a que son varios los experimentos realizados, siendo cada uno distinto a los demás, necesitamos bases de datos específicas para cada implementación.

Para las distintas pruebas realizadas bastó emplear 2 conjuntos de imágenes:

- Brodatz Álbum [18], una base de datos de compuesta por 100 imágenes que representan a distintas texturas, como por ejemplo arena, césped, tela, etc.

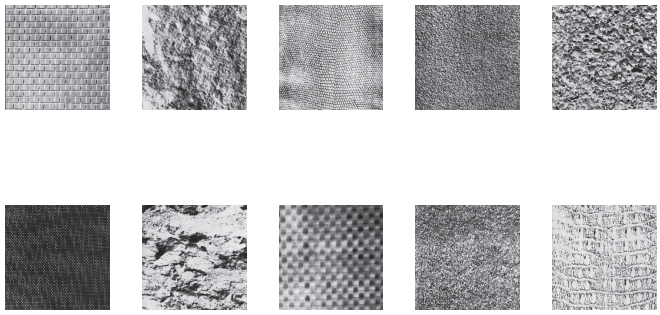


Figura 3.6: Algunas imágenes extraídas del Álbum de Brodatz.

- Corel database [20], compuesta por 1000 imágenes divididas en grupos de 100 formando 10 categorías que son:
  - Categoría 0 formada por imágenes de África.
  - Categoría 1 formada por imágenes de playa.
  - Categoría 2 formada por imágenes de Roma.
  - Categoría 3 formada por imágenes de buses.
  - Categoría 4 formada por imágenes de dinosaurios.
  - Categoría 5 formada por imágenes de elefantes.
  - Categoría 6 formada por imágenes de flores.
  - Categoría 7 formada por imágenes de caballos.
  - Categoría 8 formada por imágenes de montañas.

- Categoría 9 formada por imágenes de platos de comida.

Figura 3.7: Algunas imágenes extraídas de la base de datos Corel.

3.3. Selección de herramientas a usar

Existe una gran cantidad de herramientas que facilitan en gran medida el procesamiento de imágenes.

3.3.1. OpenCV

Una poderosa herramienta sin duda es Open Computer Vision, más conocida como OpenCV, que es una librería disponible para varios lenguajes de programación, como C++, Phyton, Android, etc. El manejo de esta herramienta es sencillo ya que trae consigo una gran cantidad de implementaciones de algoritmos para el tratamiento de imágenes.

3.3.2. ImageMagick

ImageMagick es una compilación de herramientas para realizar procesamiento de imágenes, estas herramientas por lo general son manejadas desde línea de

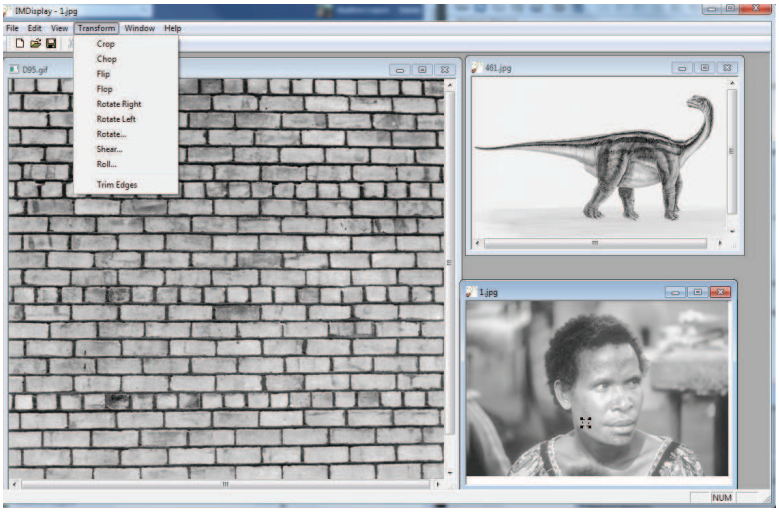


Figura 3.8: Interfaz de ImageMagick.

comandos, siendo mucho más recomendable manejarle desde ahí, aunque ImageMagick trae consigo una interfaz para los procesos.

3.3.3. Matlab

Sin duda, Matlab, será el pilar fundamental de esta tesis, y ha sido escogido por sobre otras herramientas debido a la gran cantidad de implementaciones que tiene para el tema de esta investigación.

Para el análisis de componentes independientes, existen una serie de paquetes que facilitarán el cálculo de los componentes independientes de cada imagen, algunos de los usados son:

- FastICA (http://research.ics.aalto.fi/ica/fastica/).

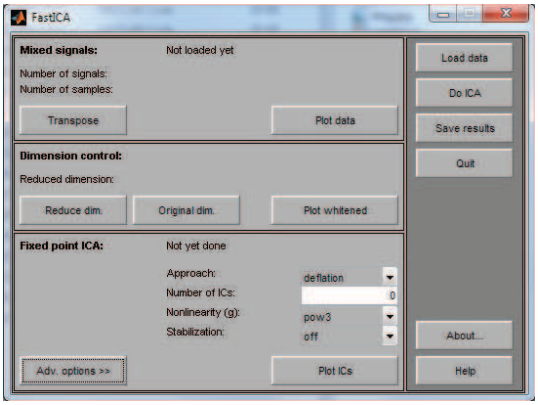


Figura 3.9: Pantalla principal de FastICA

- ICALAB (http://www.bsp.brain.riken.go.jp/ICALAB/).

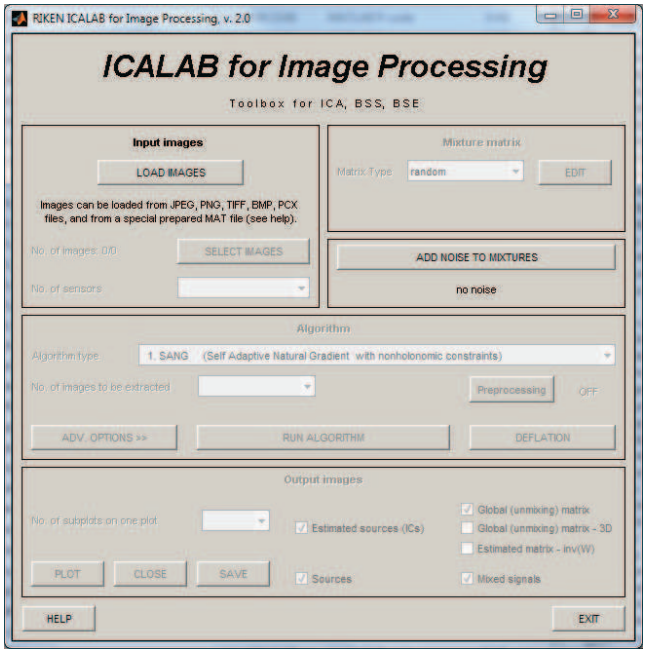


Figura 3.10: Pantalla principal de ICALAB

El uso de estas 2 herramientas será explicado en la sección de anexos.

3.4. Pruebas con las herramientas seleccionadas

Durante todo el proceso que cada imagen recibe, sea esta de entrenamiento o de prueba, deberá atravesar una serie de pasos o conversiones.

La mayoría de imágenes que serán usadas, están en el espacio de color RGB lo que representa una mayor cantidad de dimensiones. Para nuestro análisis, requerimos que las imágenes estén en escala de grises, cualquier editor de imágenes puede realizar esta conversión, pero al tratarse conjuntos grandes de imágenes que bordean las 1000 unidades, un procesamiento de cierta forma "manual" con estos editores sería muy poco eficiente. Aquí es en donde entra una herramienta previamente expuesta, ImageMagick, la misma que permite procesar directorios enteros de imágenes. Procederemos a analizar los tiempos obtenidos con diferentes cantidades de imágenes:

Imágenes	Tiempo
100	1.73s
500	4.83s
1000	8.49s

Cuadro 3.1: Análisis de tiempos que toma a ImageMagick convertir un set de imágenes, de formato PPM a JPG en escala de grises.

La siguiente prueba que realizaremos será la extracción de ICs de determinado número de imágenes usando el paquete FastICA. Cabe indicar que no se realiza ninguna reducción de dimensionalidad en las imágenes por lo que el proceso puede tornarse bastante demorado para cada imagen, las imágenes usadas han sido tomadas de Corel y los resultados obtenidos son:

Imágenes	Tiempo
1	3.65s
10	1m, 2.803s
50	4m, 39.23s

Cuadro 3.2: Tiempo que tarda extraer los ICs de un determinado número de imágenes usando FastICA.

Los resultados obtenidos dependen en gran medida del poder computacional, para estas pruebas se usó un computador con procesador Core I5 de 3.20 GHz, con 4 GB en RAM, corriendo un sistema operativo Windows 7 Pro de 32 bits.

Capítulo 4

Ejecución del plan de experimentación y análisis de resultados

4.1. Ejecución del plan de experimentación

Vamos a proceder a detallar cada experimento que se explicó en la sección 1 del capítulo anterior.

4.1.1. Experimento 1

Este primer experimento es sencillo de implementar. Usa en su totalidad imágenes pertenecientes a Corel.

Partimos de crear los sets de entrenamiento y de prueba de cada categoría, para esto tomamos un 30 % y un 70 % respectivamente, inicialmente las imágenes estaban agrupadas en directorios por cada categoría, ahora solamente tenemos un directorio para las imágenes de entrenamiento y uno para las imágenes de prueba, conteniendo estas 300 y 700 imágenes respectivamente.

En Corel encontramos solamente imágenes de 2 tamaños, las verticales son de $384x254$ pixeles y las horizontales son al contrario, de $254x384$, para este experimento trabajaremos con imágenes horizontales, es decir, imágenes en las que el número de columnas sea mayor al de las filas, por lo tanto si encontramos una imagen vertical, mediante transpuesta la convertimos en horizontal, como la figura 4.1.

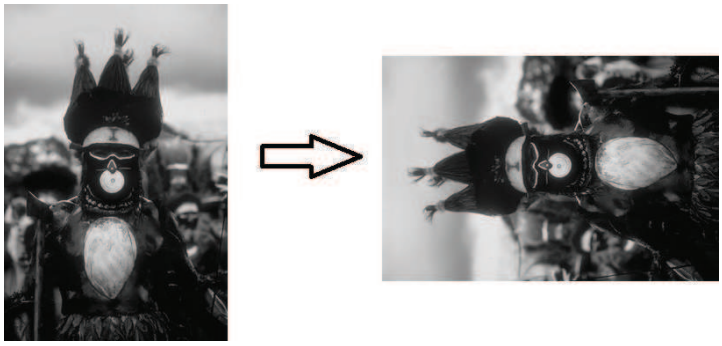


Figura 4.1: Transformación de una imagen vertical en una horizontal mediante transpuesta.

Mediante el uso del paquete FastICA explicado anteriormente, de cada imagen se extraen sus componentes independientes. Empezamos extrayendo de las 300 imágenes de entrenamiento en las cuales aplicamos una reducción de dimensiones[1], usando solamente 75 dimensiones en cada imagen, esto interfiere directamente en tiempo que toma la extracción de los ICs por cada imagen, reduciéndolo notablemente.

Hasta este punto tenemos cada imagen transformada a una matriz de $256x75$, lo que quiere decir que tenemos 256 dimensiones, cada una de 75 muestras, estos son los ICs de la imagen trabajada; Para proseguir con la experimentación, tenemos que convertir dicha matriz en una vector, es decir, formar una sola columna que contenga todos los valores presentes en la matriz, aquí usamos lo visto en la figura 3.2 para formar este vector. Aquí finalizamos el tratamiento a las imágenes de entrenamiento, siendo cada uno de estos vectores la representación de cada imagen que forma el set de entrenamiento, con lo que tendríamos 300 vectores.

Ahora, con las imágenes de prueba, realizamos el mismo proceso, después del cual tendríamos 700 vectores para componer al set de pruebas.

El paso final es el extraer la mínima distancia Euclidea entre cada uno de los 700 vectores de prueba y cada uno de los 300 vectores de entrenamiento. Para medir los aciertos y errores de este experimento tomamos en cuenta las siguientes consideraciones:

- En el set de entrenamiento, los vectores comprendidos entre el 1 y el 30 representan a la categoría 1, entre el 31 y el 60 a la categoría 2, siguiendo esta definición, sabremos que los vectores entre el 271 y 300 serán los correspondientes a la categoría 10.

- En el set de pruebas, los vectores comprendidos entre el 1 y el 70 representan a la categoría 1, entre el 71 y el 140 a la categoría 2, siguiendo

[1] A la reducción de dimensiones se le dedicará un apartado en la sección de anexos

esta definición, sabremos que los vectores entre el 631 y 700 serán los correspondientes a la categoría 10.

Basándonos en lo anterior, podemos usar como ejemplos los siguientes casos para explicar qué será considerado un acierto y que un error:

- Al comparar el vector 10 del set de pruebas contra todos los vectores de entrenamiento, la mínima distancia Euclidea obtenida es con el vector 25, entonces es un acierto, ya que el vector 10 del set de pruebas se encuentra entre 1 y 70, correspondientes a la categoría 1, lo mismo para el vector 25, que está entre 1 y 30, siendo los que representan a la categoría 1.
- Al comparar el vector 11 del set de pruebas (categoría 1) con el set de entrenamiento, y se obtiene la mínima distancia Euclidea con el vector 31 (categoría 2), lo que es un error.

Los resultados de este experimento son aproximadamente del 30 % de aciertos, resultando muy por debajo de lo esperado, hay varios parámetros que fueron configurados de distintas maneras para tratar de conseguir mejores resultados, pero la precisión no mejoró. Al revisar los valores contenidos por la matriz de ICs de cada imagen, encontramos que 2 matrices de ICs obtenidas de imágenes de la misma categoría tenían valores totalmente distintos, nada parecidos entre sí, por lo tanto, al obtener la distancia Euclidea entre los vectores que representan a las imágenes, tendríamos como resultado valores bastante altos, es decir que 2 imágenes de la misma categoría estarían bastante lejanas según este experimento. Por lo tanto, la conclusión es descartar este experimento ya que los ICs de 2 imágenes de una misma categoría no tienen relación alguna.

4.1.2. Experimento 2

Para este experimento usamos 2 bases de datos de imágenes, la de texturas llamada Brodatz, y la utilizada en el experimento anterior, Corel.

Básicamente el objetivo de este experimento es determinar qué texturas existen en una imagen, en este caso, una imagen de la base de datos Corel, y en base a esto determinar a qué categoría representa dicha imagen. Por ejemplo, si encontramos que en una imagen, varias secciones de la misma coinciden con arena, entonces podemos suponer que la imagen pertenece a la categoría de imágenes de playa.

Empezamos por determinar un numero de imágenes de texturas que nos servirán para llenar al set de entrenamiento, en este caso seleccionamos 10 en total, representadas en la figura 4.2.

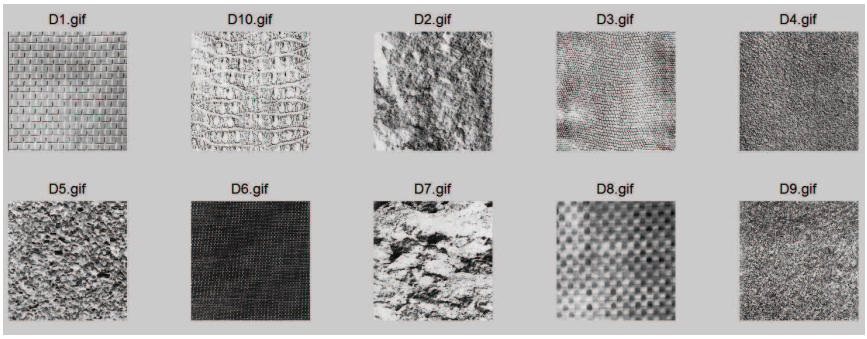


Figura 4.2: Imagen de texturas usadas en este experimento.

Ahora, mediante FastICA extraemos los componentes de cada imagen, cabe recalcar que entre el procedimiento de extracción de ICs, esta una etapa en la que mediante PCA se reducen dimensiones, para este experimento reducimos a 40. Al finalizar la aplicación de FastICA, como resultado tendremos una matriz de 640x40, lo que viene a ser 40 dimensiones con 640 muestras cada una. De cada dimensión nos quedamos solamente con 64 muestras, esto debido a que más adelante tendremos que obtener el producto punto entre cada dimensión y un vector de tamaño 64, por lo que este número de muestras debe coincidir. Hasta este punto, tenemos una matriz de $64x40$ por cada imagen de entrenamiento. Continuando con los pasos para formar el set de entrenamiento, lo siguiente será obtener los VCs de cada textura usada, este proceso se logra seleccionando una porción de la imagen de textura, esta porción será de 8x8 pixeles, dicha porción se llama ventana. Posterior a esto se forma un vector siguiendo lo indicado en la figura 3.2, teniendo un vector de tamaño 64, para construir el VC de cada imagen, tomamos cada matriz de ICs, y extraemos el producto punto entre cada una de sus dimensiones y el vector obtenido en el paso previo, y de este producto punto obtenemos su valor absoluto. Ahora tendremos 40 valores, de los cuales solo obtenemos los 3 más altos. El objetivo es extraer por cada imagen su VC, el mismo que tendrá 3 valores, entonces luego de este paso tendremos 10 vectores de tamaño 3 cada uno, es decir, 10 VCs.

El siguiente paso es bastante interesante, se trata de extraer varias ventanas de una imagen de Corel, estas ventanas serán de posiciones aleatorias y tendrán la misma dimensión usada en la etapa de entrenamiento, es decir, $8x8$. De cada una de estas ventanas obtenemos su respectivo VC. Luego obtenemos la distancia Euclidea entre este VC y cada uno de los 10 VCs de las imágenes de entrenamiento, de estos 10 valores obtenemos solamente el más bajo, y este será el que represente a la textura.

Para dejar más claro, por ejemplo si se extrae el VC de una ventana de la imagen, y al comparar con los VCs de entrenamiento, la distancia mínima es con el VC 4, siendo el correspondiente a la textura césped, esto quiere decir que en esa ventana de esta imagen de prueba existe césped. Esta idea se detalla en la figura 4.3.

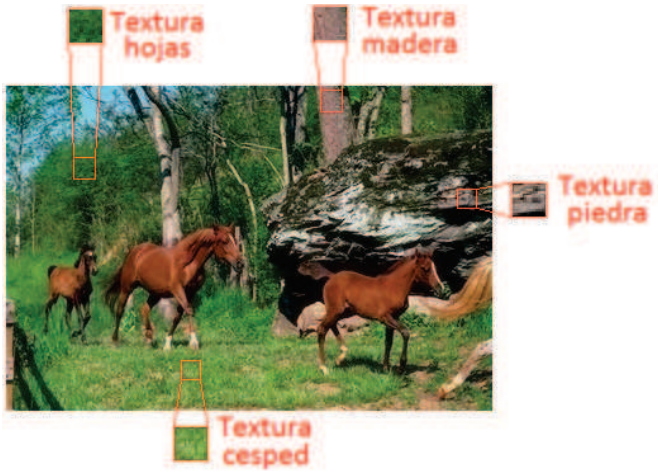


Figura 4.3: Diferentes texturas presentes en una imagen.

El resultado de este experimento fue extremadamente bajo, como un experimento previo al experimento real, se extrajo a propósito de una imagen de la categoría playa una ventana que tenía solamente arena, y su VC se comparó con los 10 VCs, el resultado esperado era obviamente que se obtenga como resultado a la textura de arena, pero en muy pocos casos esto se consiguió. Se intentó comparando otras ventanas que así mismo contenían arena, pero el resultado no mejoro, en algunos casos la precisión obtenía estaba por debajo del 10 % de aciertos, lo que nos hizo descartar esta metodología.

4.1.3. Experimento 3

Este experimento es bastante parecido al visto anteriormente, la diferencia está en que esta implementación trabaja totalmente con el álbum de Brodatz. Esta idea parte de la baja precisión que el experimento anterior arrojó, entonces lo aquí planteado intenta explorar un nivel más bajo de comparación, ya que la idea básica es comparar cada textura con las mismas texturas, con la diferencia de que se trabajarán con porciones extraídas de distintos puntos de una imagen.

La idea es similar, formar un conjunto de pruebas con 10 imágenes de entrenamiento, para posteriormente extraer el VC de cada una de ellas, recordando, los VCs de las imágenes de entrenamiento se extraen de determinada sección de imagen, la llamada ventana que tiene dimensiones cuadradas. Para este experimento, al ventana es extraída de la posición $8, 8$ teniendo un tamaño de $8x8$.

Posterior a esto, se tomó una imagen de las mismas 10, se extrajo una ventana de la posición $20, 20$, con tamaño $8x8$ también, y asimismo extraemos su VC. Este proceso observa en la figura 4.4.

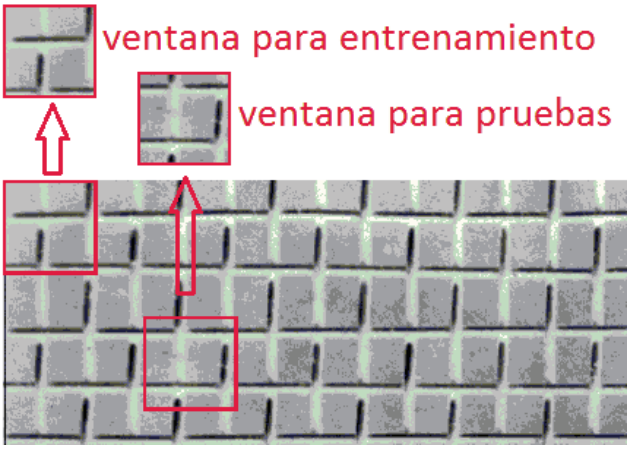


Figura 4.4: Porciones de imagen de las que se extrae los VCs de las texturas.

Resumiendo, lo que se ha hecho hasta este paso es, primero extraer una ventana de cada imagen, desde la posición $8x8$, para de las mismas obtener su VC, es decir, en total 10 para tener el set de entrenamiento. Luego se extrajo de las mismas imágenes 10 VCs más, pero ahora con ventanas tomadas desde la posición $20, 20$.

El último paso de este experimento es similar al que hemos hecho en experimentos anteriores, obtener la distancia Euclidea entre cada uno de los 10 VCs de prueba y los 10 de entrenamiento, de dicha operación por cada VC de prueba obtenemos 10 valores, de los cuales tomamos el menor, correspondiente a la imagen que más se parece a la imagen de prueba usada.

Los resultados obtenidos tampoco cumplieron las expectativas que para este experimento se tenían, de las 10 imágenes de texturas que fueron usadas, se registró solamente 2 aciertos. Luego de esto se probó con distintas posiciones para las ventanas, y el resultado no variaba mucho, no siempre las mismas imágenes acertaban en cada experimento, por lo que se descubrió que este experimento no es apto para realizar clasificación de imágenes, la precisión promedio obtenida es de 20 %.

4.1.4. Experimento 4

Como se detalló en la explicación del este experimento en el capítulo anterior, esta implementación se basa en los 2 experimentos anteriores. Empezamos recordando que se trabaja solamente con imágenes presentes en Corel. Lo primero es elegir una de las 10 categorías, para nuestro experimento probamos con varias categorías escogiendo a la categoría 4 ya que fue con la que mejores resultados obtuvimos. De las 100 imágenes que componen a esta categoría, escogemos aleatoriamente a 30 para formar el set de entrenamiento. Las imágenes escogidas son:

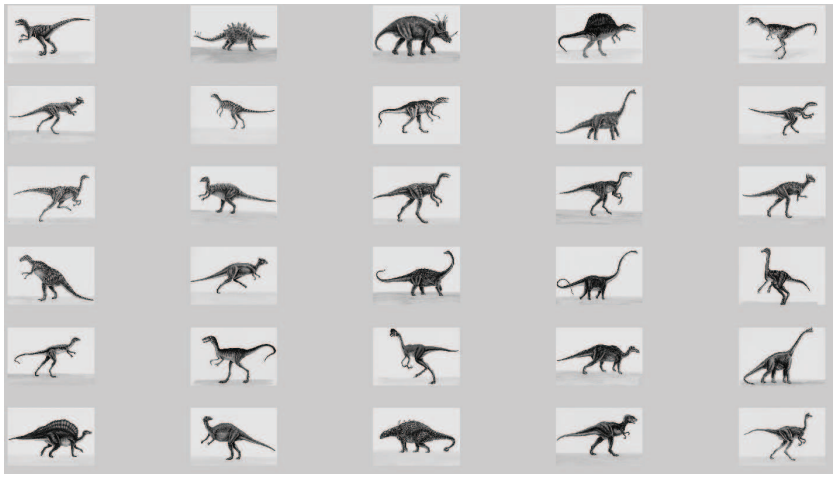


Figura 4.5: Imágenes que forman el set de entrenamiento, pertenecientes a la categoría 4.

Mediante PCA se disminuye a 40 las dimensiones usadas de cada imagen, para posteriormente extraer los ICs. Por cada imagen se obtendrá una matriz de 640 filas por 40 columnas, es decir 40 dimensiones de 640 muestras cada una, detallando un poco más, cada imagen tendrá 40 dimensiones. Para continuar con esta prueba, de cada dimensión se usara solamente los 64 primeros valores. La razón de que sea 64 valores se explicará en el siguiente paso. Ahora, procedemos a extraer el VC de cada imagen de entrenamiento, para esto debemos trabajar con solamente una porción de cada imagen, en este caso se la llama "ventana", la misma que según [23] es de $8x8$. A cada ventana la transformamos en un vector columna usando un recorrido vertical, como se observa en la figura 3.2, obteniendo un vector de tamaño 64, coincidiendo este valor con el de cada dimensión de la matriz de ICs de cada imagen; Es por esto que solamente se tomaban 64 de las 640 muestras que la matriz tiene por cada dimensión. Lo siguiente es el obtener el producto punto entre este vector columna y cada dimensión de cada imagen, y de este producto punto extraemos su valor absoluto. Al final de este paso tendremos nuevamente un vector formado por 1200 resultados de la operación producto punto, ¿Por qué 1200?, porque por cada imagen tenemos 40 dimensiones, si son 30 imágenes tenemos 1200 dimensiones a ser procesadas con el vector columna resultando de esta operación solo un valor del cual se extrae su valor absoluto.

El último paso es ya formar el VC, y para esto simplemente de los 1200 valores, tomamos solamente los 3 más altos [23], obteniendo así el VC de una imagen. Este proceso se lo hace con cada una de las imágenes de entrenamiento.

Tenemos 30 vectores, cada uno que tiene 3 valores. Estos vectores forman al set de entrenamiento.

La prueba que se planteó es la siguiente, crear varios subconjuntos formados

por 1 imagen de cada categoría, es decir, en total 10 imágenes cada subconjunto, de los cuales seguimos los mismos pasos para obtener los VCs de cada una de estas 10 imágenes, es decir, tendríamos 10 vectores de 3 valores cada uno. El objetivo es determinar cuál de las 10 imágenes pertenece a la categoría usada para formar el set de entrenamiento.

Los resultados obtenidos fueron interesantes, procedemos a indicar cuales son las imágenes de cada subconjunto y cuál es la imagen que el algoritmo obtuvo como la perteneciente a la categoría, dicho en otras palabras, al haber creado un conjunto de entrenamiento con la categoría de dinosaurios, la imagen que se espera al comparar con las 10 imágenes de pruebas es la perteneciente a la categoría dinosaurios. Las 10 imágenes en las primeras 2 filas son las usadas para formar el conjunto de pruebas, mientras que la única imagen en la tercera fila indica el resultado de aplicar el algoritmo en ese conjunto de pruebas.

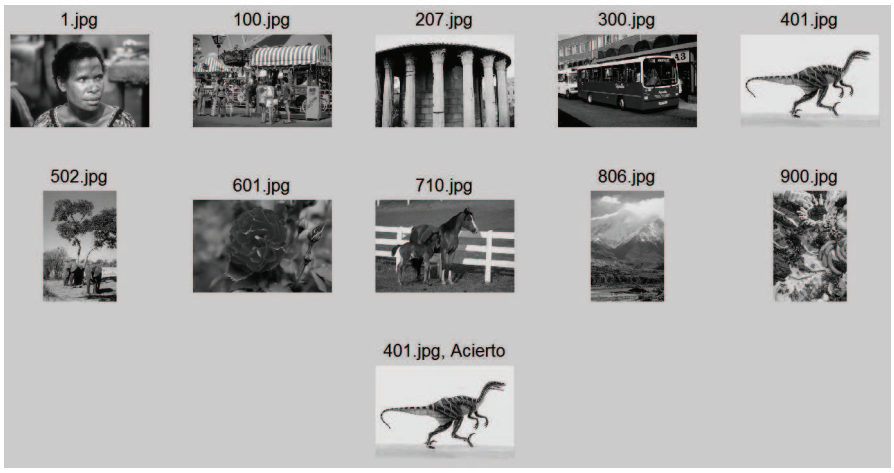


Figura 4.6: Subconjunto 1, acierto.

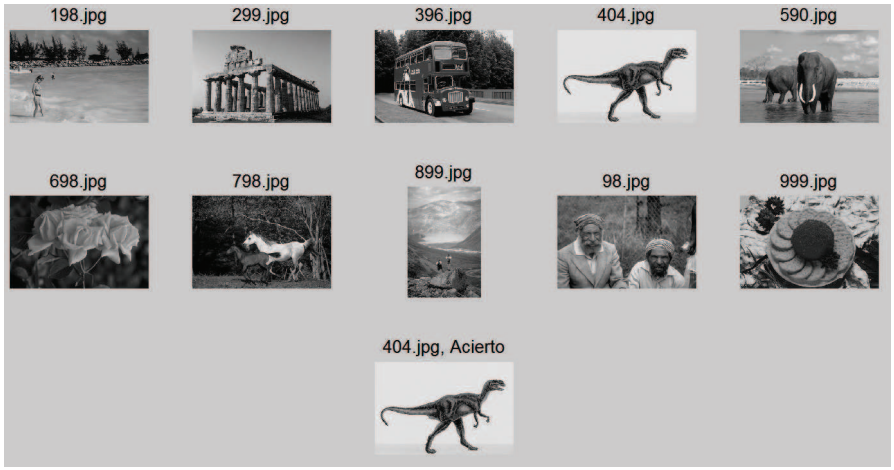


Figura 4.7: Subconjunto 2, acierto.

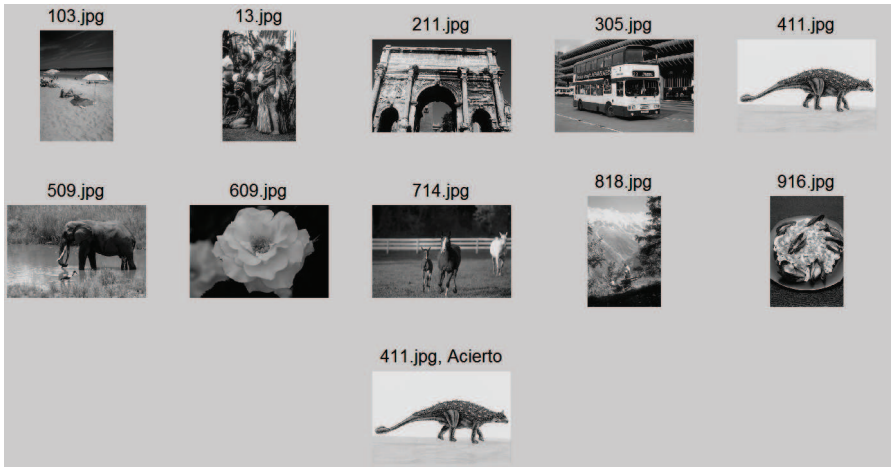


Figura 4.8: Subconjunto 3, acierto.

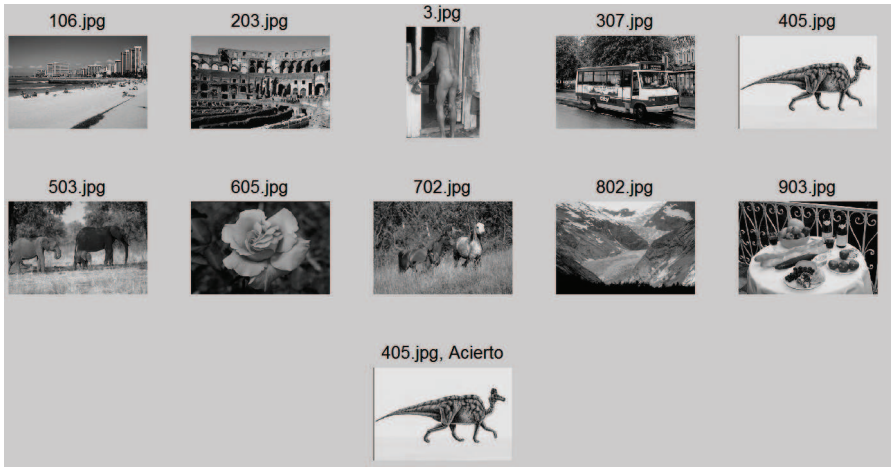


Figura 4.9: Subconjunto 4, acierto.

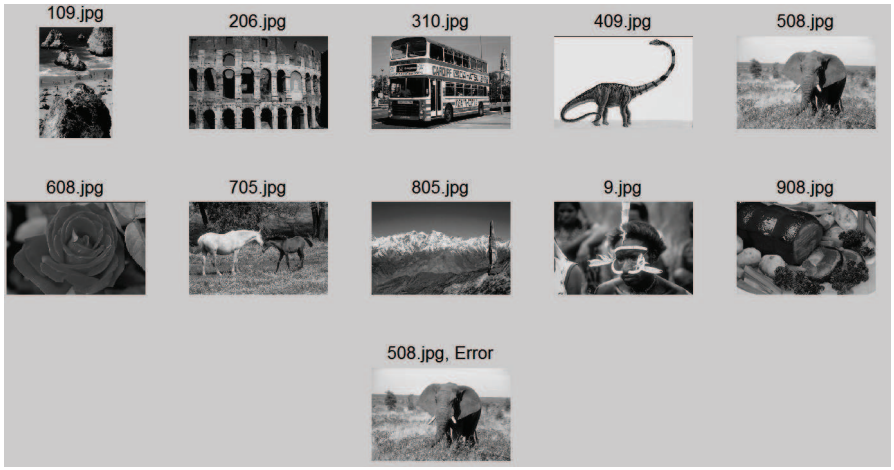


Figura 4.10: Subconjunto 5, error.

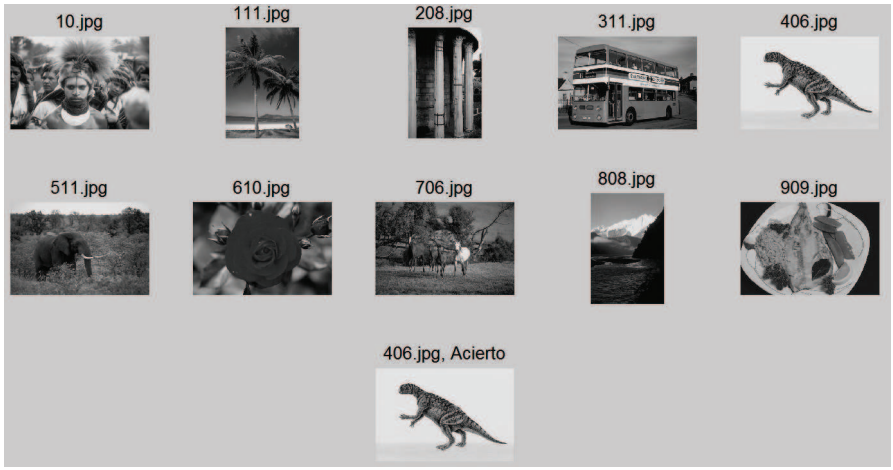


Figura 4.11: Subconjunto 6, acierto.

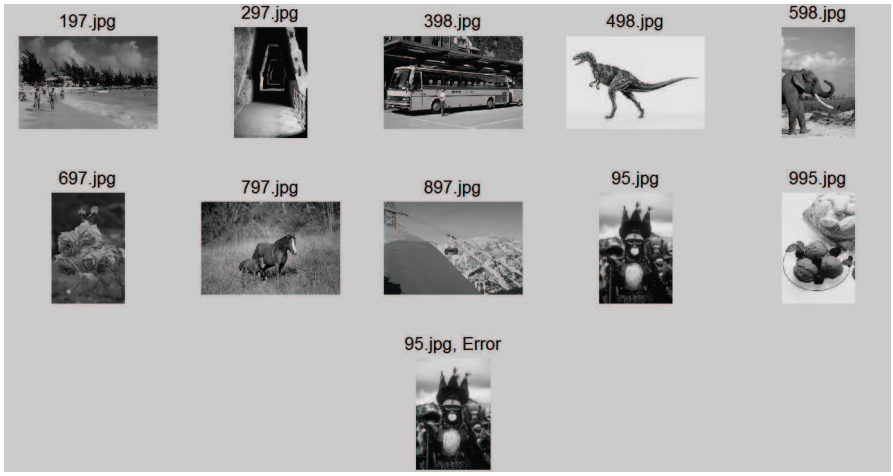


Figura 4.12: Subconjunto 7, error.

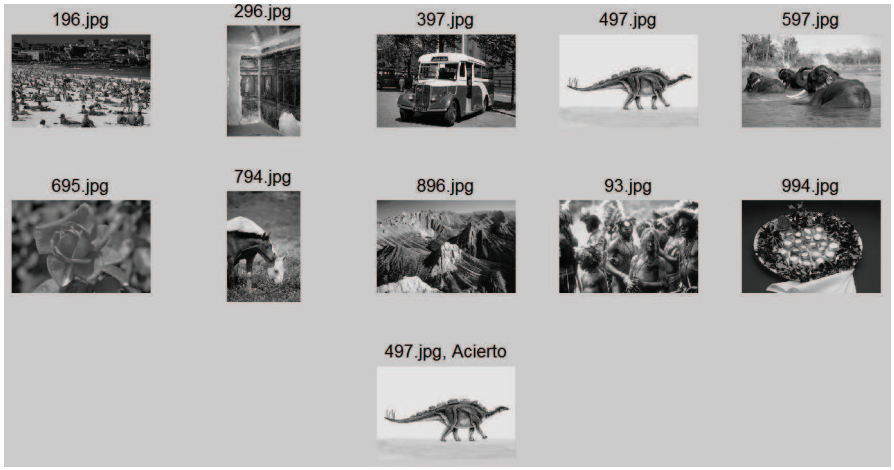


Figura 4.13: Subconjunto 8, acierto.

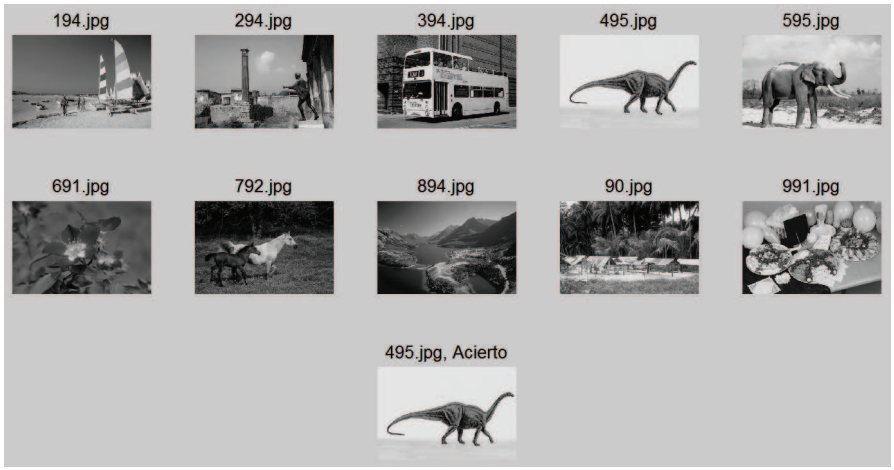


Figura 4.14: Subconjunto 1, acierto.

Con estas pruebas iniciales, obtenemos un 80 % de precisión, superando ampliamente a los experimentos anteriores. Si aumentamos la cantidad de conjuntos de pruebas, es bastante probable que esta precisión suba, en la sección 4 de este capítulo se plantean varias mejoras a este experimento que se estima conseguirán resultados con mayor nivel de precisión.

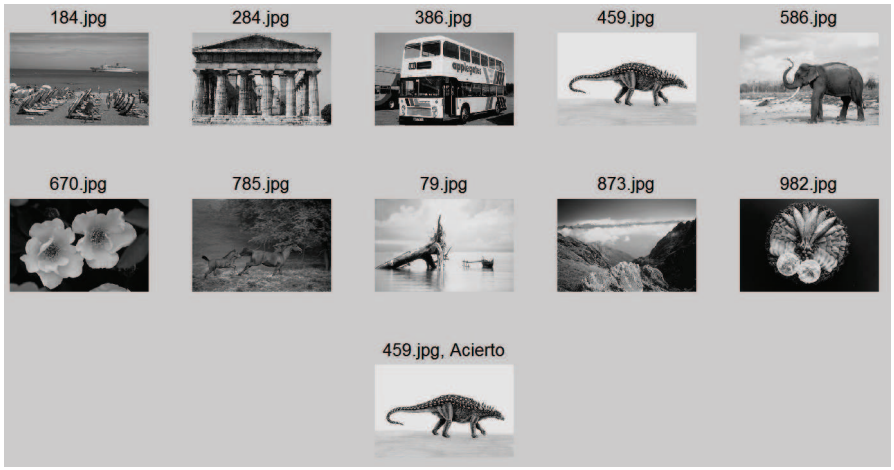


Figura 4.15: Subconjunto 1, acierto.

4.2. Análisis de precisión, cobertura y F-measure obtenidos

Partiendo de los resultados obtenidos en el experimento 4, podemos definir a la precisión como la cantidad de de pruebas en las que el algoritmo emparejó correctamente la categoría usada para el entrenamiento con la imagen entre las 10 usadas para pruebas. Luego de las pruebas realizadas, se obtuvo una precisión equivalente al 80 %.

En este trabajo, al tratarse de una técnica que usa todos los documentos (imágenes), la cobertura es del 100 %

Con lo datos anteriores podemos calcular el F-Measure, usando la siguiente formula:

$$F - Measure = 2 * \frac{Precisión * Cobertura}{Precisión + Cobertura}$$

$$F - Measure = 2 * \frac{80 * 100}{80 + 100}$$

$$F - Measure = 88{,}9$$

El resultado para el F-Measure nos indica que los resultados obtenidos alcanzan un nivel aceptable.

4.3. Comparativa con el estado del arte

Existen una gran variedad de implementaciones que buscan clasificar imágenes usando ICs, citando algunas de estas, empezamos tomando la que usamos para los experimentos 2, 3 y 4 [23], en esta implementación se usó la técnica de

VCs para la comparación entre imágenes. Los resultados expuestos en aquella investigación son los apreciados en el cuadro 4.1:

Tamaño de ventana	Precisión alcanzada
$3x3$	74.56 %
$5x5$	82.06 %
$8x8$	97.54 %
$12x12$	93.24 %
$17x17$	89.69 %

Cuadro 4.1: Precisión obtenida por [23], en ese trabajo se plantean distintos tamaños de ventanas.

Observando la tabla anterior notamos que se logra una alta precisión, que alcanza en el mejor caso un 97.54 %.

En otro trabajo [16], se plantea el uso de filtros ICA para clasificar las imágenes, aplicando a 105 imágenes de entrenamiento y 540 para pruebas. La precisión obtenida por este trabajo alcanza el 85 %.

Comparando la tasa de precisión de trabajos ya planteados, con la mejor precisión obtenida en el experimento 4, sabemos que estamos en un rango aceptable para concluir que la técnica presentada es factible.

4.4. Mejoras planteadas

Principalmente, en el experimento 4 se pueden plantear mejoras ya que fue la implementación con mejores resultado. Por ejemplo, para intentar mejorar la precisión de cada imagen de prueba se pueden extraer no solamente una ventana, sino más, quizá unas 5 o 10, y determinar a qué categoría pertenece la mayoría de estas ventanas para en base a esto tomar un decisión.

El trabajo del último experimento puede plantearse como plataforma para una nueva implementación que a continuación detallamos.

Se trata de expandir el set de entrenamiento, no solo tendríamos los VCs de las 30 imágenes de solamente una categoría, sino incluimos también 30 VCs por cada categoría, es decir, en total tendríamos 300 VCs que formarían al set de entrenamiento. Contra todo este set podríamos comparar cualquiera de las 700 imágenes restantes y analizaríamos la precisión que esa implementación alcanzaría.

Para extender aún más el alcance de los experimentos, se puede también probar lo que sucede si usáramos más dimensiones, o quizá ventanas de diferentes tamaños, esto sin duda variaría la precisión.

Existen otros artículos [17], [24], [27], etc., que usan otras implementaciones para extraer los ICs de las imágenes, sería interesante realizar los experimentos aquí planteados con diferentes técnicas, así podríamos analizar a más de la precisión, el tiempo tomado, y podríamos también sacar una conclusión sobre el mejor algoritmo para el tratamiento de imágenes.

Conclusiones

El trabajo aquí estudiado ha concluido en la presentación de una técnica válida, que consiguió los objetivos de clasificación planteados desde un principio.

Al iniciar cada experimento, de ninguna manera se tenía la certeza de que los mismos tendrían una cuota de precisión aceptable, quedando evidenciado precisamente eso en los resultados que los 3 primeros experimentos arrojaron, pero escenarios como aquellos son necesarios para cada vez plantear mejores técnicas, lo que se vio representado en el experimento 4, que empezó siendo una mejora y recopilación de los 3 anteriores, para terminar siendo el experimento que mejores resultados consiguió.

Este trabajo viene a ser el fin de una larga cadena de investigación apegada a la inteligencia artificial, pues fue en dicha asignatura y los diferentes proyectos propuestos por el docente, lo que creo en el autor fascinación por el estudio de esta área.

Todo lo aquí estudiado representa a largas horas de investigación y búsqueda de información en artículos y libros relacionados con el tema, además de propuestas planteadas entre el director y el autor que al final lograron el cometido del trabajo, este es un campo muy amplio en el que se ha incursionado con esta tesis, aún hay mucho por descubrir y proponer.

La conclusión que todo este trabajo entrega es que lo aquí planteado es totalmente viable, implementable y seguramente mejorable, el escenario ideal será de que principalmente el último experimento sirva como plataforma para futuros trabajos de investigación que estén dentro del área, y que sobre todo se consiga mejorar la precisión aquí obtenida.

La visión por computador es una área tan fascinante, y por ello también quedó claro de lo grande que puede tornarse los componentes independientes, que al final son solamente una mínima sección de todo lo que abarca la visión artificial.

Recomendaciones

Algo que quedó completamente evidenciado en este trabajo es que los resultados no siempre serán los que esperamos, pueden ser bastante más bajos de lo deseado, pero estos no dejan de ser resultados, el objetivo de trabajos como este es el de probar la efectividad de experimentos planteados, y mientras más sean las técnicas propuestas, mejor será el trabajo realizado.

En futuras implementaciones que se basen en los experimentos aquí expuestos se recomienda usar muchas más bases de datos de imágenes, lo que de seguro permitirá descubrir que posiblemente un experimento tenga resultados bastante más precisos con un conjunto de imágenes, mientras que con otro la precisión sea demasiado baja.

En futuras investigaciones que estén centradas en la clasificación de imágenes, se recomienda partir del experimento 4 principalmente, ya que se pudo comprobar que es eficiente, mientras que los demás experimentos no fueron analizados a fondo, y, basándose en los resultados preliminares lo más probable es que estos no mejoren.

Al ser una rama de la visión por computador el análisis mediante componentes independientes, se recomienda fuertemente proponer muchas más investigaciones en este campo, aún hay mucho por investigar, mucho por descubrir, pero sobre todo, mucho por crear y mejorar.

Glosario

det Determinante.

ICA Independent Component Analysis o Análisis de Componentes Independientes.

ICs Independent Components o Componentes Independientes.

BSS Blind Source Separation o Separación Ciega de Fuentes.

JPG Joint Photographic Experts Group o Grupo Conjunto de Expertos en Fotografía.

OpenCV Open Computer Vision.

PCA Principal Component Analysis o Análisis de Componentes Principales.

PPM Portable Pixmap o Mapa de Pixeles Portable.

RAM Random Access Memory o Memoria de Acceso Aleatorio.

RGB Red, Green and Blue, o Rojo, Verde y Azul, son 3 colores que al mezclarlos permiten obtener cualquier otro color.

SVD Singular Value Decomposition o Descomposición en Valores Singulares.

VCs Vectores Característicos.

Bibliografía

[1] Department of Information Aalto University and Computer Science. Independent Component Analysis (ICA) and Blind Source Separation (BSS).

[2] K. Baek, B. A. Draper, J. R. Beveridge, and K. She. PCA vs. ICA: A comparison on the FERET data set. In *Joint Conference on Information Sciences, Durham, NC*, pages 824 – 827, 2002.

[3] H. Le Borgne, N. Guyader, A. Guerin-Dugue, and J. Hérault. Classification of images: ICA filters vs human perception. In *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, volume 2, pages 251 – 254. IEEE, 2003.

[4] S. Chindaro, K. Sirlantzis, and M. C. Fairhurst. ICA-based multi-colour space texture classification system. *Electron. Lett*, 42(21):1208 – 1210, 2006.

[5] Laboratorio de Control y Sistemas Inteligentes. ICA VS. PCA COMPARISON.

[6] Biblioteca de ingeniería Universidad de Sevilla. Fundamentos básicos del Análisis de Componentes Independientes.

[7] Biblioteca de Ingeniería Universidad de Sevilla. INTRODUCCIÓN A ICA Y PCA .

[8] S. Deegalla and H. Bostrom. Reducing high-dimensional data by principal component analysis vs. Random projection for nearest neighbor classification. In *Machine Learning and Applications, 2006. ICMLA'06. 5Th International Conference on*, pages 245 – 250. IEEE, 2006.

[9] K. Delac, M. Grgic, and S. Grgic. Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, 15(5):252 – 260, 2005.

[10] Facultad de ciencias exactas UNLP Departamento de Matemática. Descomposicion en Valores singulares(SVD). 2010.

[11] L. J. P. Van der Maaten. An introduction to dimensionality reduction using matlab. *Report*, 1201:07 – 07, 2007.

[12] B. A. Draper, K. Baek, M. S. Bartlett, and J. R. Beveridge. Recognizing faces with PCA and ICA. *Computer vision and image understanding*, 91(1):115 – 137, 2003.

[13] Oscar Reinoso García. Inteligencia Artificial y Reconocimiento de Patrones. pages 26–32.

[14] Oscar Reinoso García. Selección y Extracción de características.

[15] P. O. Hoyer and A. Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191 – 210, 2000.

[16] B. Huang, J. Li, and S. Hu. Texture feature extraction using ICA filters. In *Intelligent Control and Automation, 2008. WCICA 2008. 7Th World Congress on*, pages 7631 – 7634. IEEE, 2008.

[17] J. Hurri. Independent component analysis of image data. *Master's thesis, Dept. of Computer Science and Engineering, Helsinki University of Technology, Espoo, Finland*, 1997.

[18] Universitetet i Stavanger. Brodatz Textures.

[19] M. Journée. Matlab project Independent component analysis. 2008.

[20] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1075 – 1088, 2003.

[21] I. E. López. *Descripción de texturas: Aplicaciones a su comprensión y clasificación.* Universitat de Valčncia, Facultat de Cičncies Matemātiques, 2002.

[22] D. Mery. Visión por computador. *Santiago de Chile. Universidad Católica de Chile*, 2004.

[23] A. Nadi, D. Abu, Mansour, and A. M. Independent component analysis (ICA) for texture classification. In *Systems, Signals and Devices, 2008. IEEE SSD 2008. 5Th International Multi-Conference on*, pages 1 – 5. IEEE, 2008.

[24] O. G. Sezer, A. Ertiizun, and A. Erçil. Independent component analysis for texture defect detection. *PATTERN RECOGNITION AND IMAGE ANALYSIS C/C OF RASPOZNAVANIYE OBRAZOV I ANALIZ IZOBRAZHENII.*, 14(2):303 – 307, 2004.

[25] Paul Skoufranis. Singular Value Decomposition Example. March 2010.

[26] M. A. Vicente, C. Fernández, A. Gil, and L. Payá. Equivalencia entre ICA y PCA como métodos de extracción de características en reconocimiento visual basado en apariencia.

[27] R. N. Vigário. Extraction of ocular artefacts from EEG using independent component analysis. *Electroencephalography and clinical neurophysiology*, 103(3):395 – 404, 1997.

[28] Vitutor. *Coeficiente de correlación*.

Anexos

Anexo 1: Uso de ImageMagick.

Como lo mencionamos en la sección de herramientas a ser usadas en esta tesis, ImageMagick es un potente utilitario para el procesamiento de imágenes. En esta tesis representó uno de los primeros pasos a ser realizados, específicamente en los experimentos que usaban imágenes de la base de datos Corel, cada una de las imágenes que la componen están en formato PPM, nativamente el sistema operativo Windows no incluye un codec para la lectura de este formato.

Para los experimentos, se decidió es trabajar con las imágenes en formato JPG. En el procesamiento de grandes lotes de imágenes, ImageMagick es un gran aliado ya que permite el aplicar una diversidad de técnicas a grandes conjuntos de imágenes con solo un comando. Recordando lo citado anteriormente, es mucho más eficiente trabajar mediante línea de comandos que con la interfaz de usuario que trae consigo ImageMagick.

Un directorio de una categoría del Álbum Corel contiene originalmente 100 imágenes, como se indica en la figura 4.16.

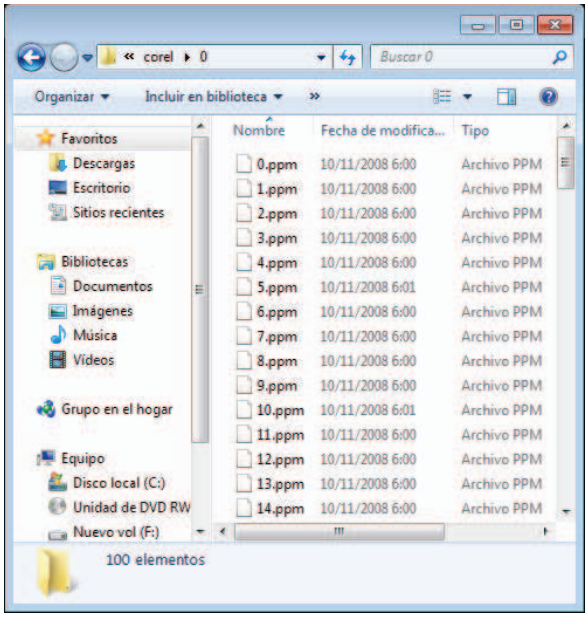


Figura 4.16: Cada uno de las 10 carpetas que conforman la base de datos de imágenes Corel, contiene 100 elementos, todos ellos en formato PPM.

Si queremos convertir cada una de estas imágenes a formato JPG, basta con ejecutar el siguiente comando:

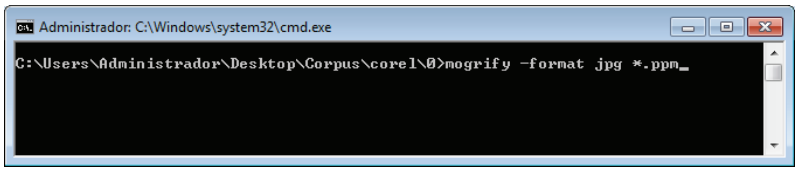


Figura 4.17: Este comando en particular sirve para ubicar todos los archivos de formato PPM en el presente directorio y convertirlos a JPG, lo clave aquí es la palabra "mogrify", que es la que permite trabajar con directorios.

Al ejecutar el comando de la figura 4.17, se obtiene como resultado cada una de las imágenes ya convertidas en el mismo directorio:

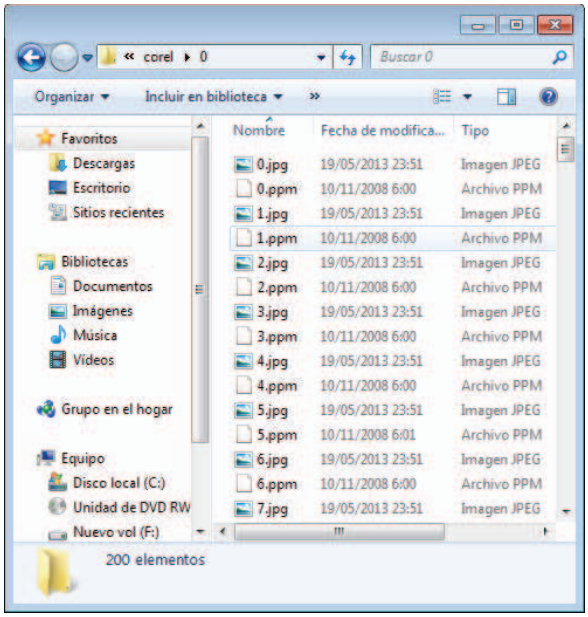


Figura 4.18: El mismo directorio de la figura 4.16, pero con las imágenes ya convertidas.

En el capítulo 3, en el cuadro 3.1 se incluyen los tiempos que este proceso toma.

Anexo 2: Uso de FastICA mediante interfaz gráfica.

Esta herramienta facilitó considerablemente el trabajo propuesto en cada uno de los experimentos, por el hecho de que FastICA es una implementación bastante amplia de la extracción de ICs de matrices de datos, caso contrario hubiésemos tenido que dedicar un considerable tiempo a la creación de un algoritmo lo bastante robusto para la obtención de los ICs.

Primero procedemos a descargar el paquete de FastICA, que lo encontramos en la dirección mencionada en [22], el cual es un conjunto de archivos, cada uno cumpliendo una determinada función. Al examinar detalladamente estos archivos encontramos que están minuciosamente documentados, es decir si quisiéramos modificarlos o acoplarlos para que cumplan determinada función nos fuera bastante sencillo debido a que están eficientemente estructurados.

La interfaz gráfica es bastante sencilla de entender, para poder acceder a ella, en Matlab nos situamos en el directorio que contiene al paquete descargado de FastICA, y aquí escribimos "Fasticag" lo que abre una ventana como la mostrada en la figura 3.9 del capítulo 3.

Los pasos para extraer los ICs de una imagen son los siguientes:

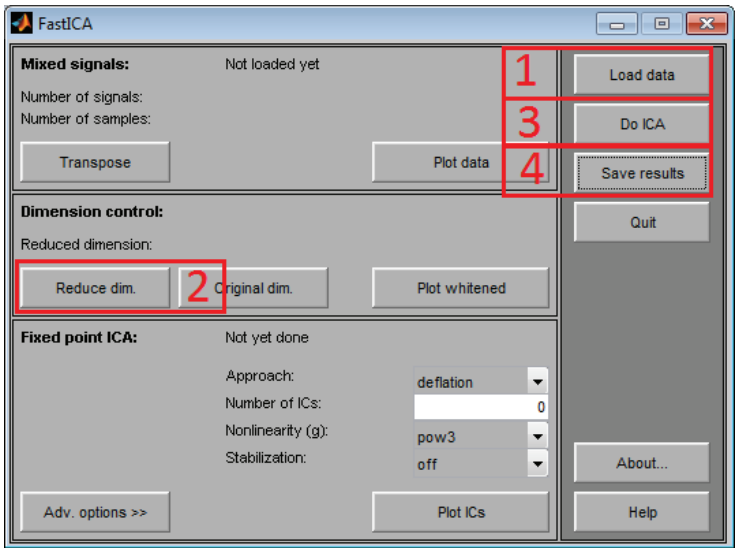


Figura 4.19: Pasos para extraer los ICs de una imagen usando la interfaz gráfica de FastICA

- Cargar una imagen pulsando en el botón "Load data", lo que abrirá el recuadro ilustrado en la figura 4.20, aquí tenemos que ingresar la variable en la que fue cargada una imagen en Matlab, este proceso debió realizarse mediante el comando "Imread", por ejemplo:

```
texture = imread('images/image.jpg');
```

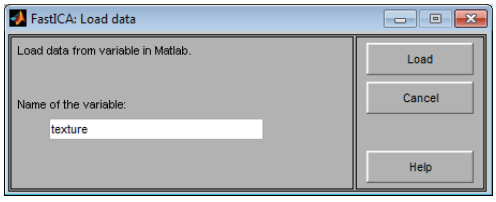


Figura 4.20: Recuadro para la carga de una variable que almacena una imagen en FastICA.

- Luego, podremos aplicar una reducción de dimensiones, para eso pulsamos "Reduce dim", que en la figura 4.19 se lo aprecia con el número 2, al hacerlo se mostrará una nueva ventana como la que se aprecia en la figura 4.21, lo encerrado en el recuadro rojo nos permite indicar el rango entre el que se tomarán las dimensiones.

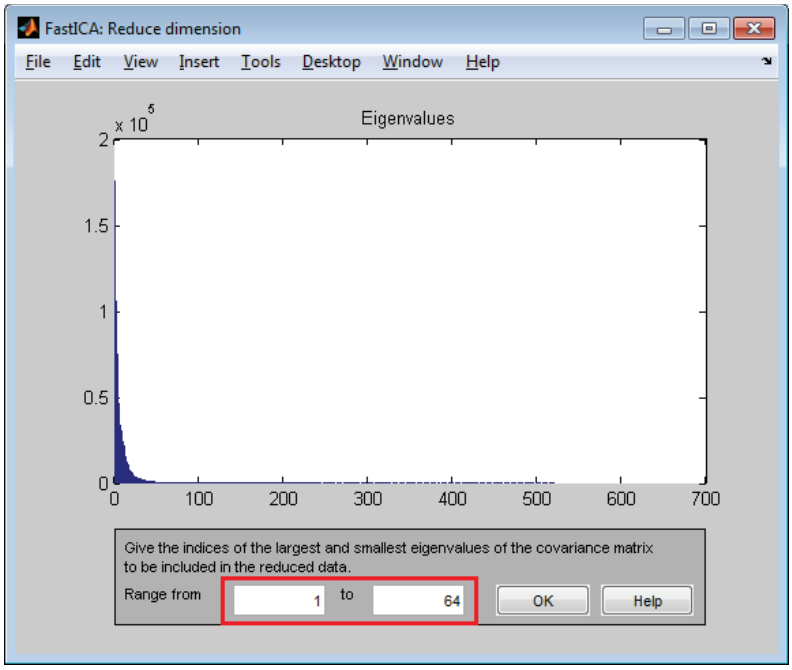


Figura 4.21: Ventana para aplicar reducción de dimensiones.

- Ahora procedemos a pulsar el botón "Do ICA", el que iniciará el proceso de extracción de los ICs, el avance de estos cálculos se lo ve en la consola de Matlab.
- Lo último que haremos es almacenar los datos, para eso pulsamos el botón "Save", lo que abrirá un recuadro en el que debemos especificar un prefijo con el que se guardarán los datos calculados, lo que aquí ingresemos como texto, estará precedido por todas las siguientes variables:
  - W: La matriz estimada de separación, es decir, la que devuelve a un estado normal a los datos que han sido mezclados por la matriz de mezcla.
  - A: La matriz estimada de mezcla, contraria a la anterior, permite mezclar a los datos de entrada.
  - IC: La matriz que contiene a los componentes independientes estimados.
  - D: Contiene a los EigenVectors.
  - E: Contiene a los EigenValues.
  - WhiteningMatrix: Matriz de blanqueo de datos, usada en la reducción de dimensiones.

- DewhiteningMatrix: Inversa a la matriz anterior, permite recuperar los datos a su estado original.

Los pasos que se acabaron de redactar, son para una extracción bastante básica de ICs, quedará como pendiente el investigar más a fondo los demás componentes que son totalmente parametrizables en la interfaz gráfica del paquete de FastICA, asimismo, es mucho más recomendado trabajar directamente desde la ventana de comandos, debido a que será más manejable el tema de procesamiento de varias imágenes al mismo tiempo, así como la aplicación de algoritmos en mitad del proceso. Los experimentos trabajados en esta tesis no hubiesen podido ser implementados solamente desde de la interfaz gráfica, ya que como procesos intermedio se requería la toma de determinado número de muestras, por lo que la personalización de las diferentes clases que componen al paquete FastICA fue necesaria.

En el siguiente anexo se incluye como es el procesamiento de una imagen solamente mediante comandos.

Anexo 3: Uso de FastICA mediante línea de comandos.

Al usar la línea de comandos contamos con mucha más libertad para personalizar cualquier etapa de todo el proceso que conlleva la extracción de ICs.

Apegándonos a los pasos descritos en la figura 4, el código para la carga de imágenes es el siguiente:

```
texture = imread('image.jpg');
texture = double(texture);
```

La segunda línea convierte cada elemento de la matriz a tipo double, ya que FastICA trabaja con este tipo de datos, si no hacemos este paso previo, internamente FastICA se encarga de hacerlo indicándonos que dicho proceso fue realizado.

El paso número 2, la reducción de dimensiones se lo realiza de la siguiente manera:

```
[E,D] = pcamat(texture,1,dimensions);
D = rot90(D,2);
E = flipdim(E,2);
[nv, wm, dwm] = whitenv(texture, E, D);
```

En donde, la variable "dimensions" indica con cuántas dimensiones trabajaremos, explicando este caso, la primera línea obtiene los EigenValues y EigenVectors aplicando PCA para reducir dimensiones de la variable "texture", tomando las que estan en el rango entre 1 y "dimensions. Las siguientes 2 líneas sirven para ordenar de mayor a menor los EigenValues y EigenVectors, ya que los propios paquetes indican que no en todos los casos estos valores están ordenados de mayor a menor, al comprobarlo notamos que están invertidas estas

matrices, es decir, encontramos los valores de menor a mayor. La cuarta línea calcula la matriz de blanqueo y su inversa, necesarias para el cálculo de los ICs.

El paso número 3 se lo realiza de la siguiente manera:

```
[A, W] = fpica(nv, wm, dwm, 'defl', dimensions, 'pow3', '
    off', 1, 1, 0, 'on', 0.0001, 1000, 100, 'guess', guess
    , 1, 'off', 1, 'on');
icasig = W * texture;
```

Como se aprecia en el código, interviene una parte parámetros obtenidos en pasos anteriores, mientras que el resto son parámetros para configurar el cálculo de los ICs, todos excepto la variabe "guess" tienen valores por defecto, tomados de las clases que componen al paquete FastICA; La variable "guess" es una matriz aleatoria inicial de la que se partirá el cálculo de la matriz de ICs de la imagen con la que se está trabajando. Lo último que se hace es calcular la matriz de ICs, multiplicando la matriz de entrada por la matriz de separación "W", entendiéndole de otra forma, la matriz "texture" contiene a los datos originales, es decir, incluye a los datos que no nos interesa en nuestro cálculo, si la multiplicamos por la matriz de separación "W", obtendremos solamente los componentes independientes.

Anexo 4: Código fuente usado en el experimento 4.

```
%%Obtención de los componentes independientes de las
    imagenes de %%entrenamiento
clear;
categoria           = 'images/din/c4/';
dimensions          = 40;
windowSize          = 8;
dir1                = dir(strcat(categoria,'*.jpg'));
Ics                 = cell(1,numel(dir1)*dimensions );
featVectorsTrain    = cell(1,numel(dir1));
guess = orth (randn (256, dimensions));

for i=1:numel(dir1)
        texture=double(imread(strcat(categoria,dir1(i).
            name)));            [E,D]=pcamat(texture,1,
            dimensions);
        D=rot90(D,2);
        E=flipdim(E,2);
        [nv, wm, dwm] = whitenv(texture, E, D);
        [A, W] = fpica(nv, wm, dwm, 'defl', dimensions, '
            pow3', 'off', 1, 1, 0, 'on', 0.0001, 1000,
            100, 'guess', guess, 1, 'off', 1, 'on');
```

```
    icasig = W * texture;       icasig=icasig';
        for j=1:dimensions
        vectorAux=icasig(:,j);
            meanAux=mean(vectorAux);
        for k=1 : length(vectorAux)
                if(vectorAux(k)<meanAux)
                        vectorAux(k)=0;
                end
                end
        icasig(:,j)=vectorAux;
        Ics{(i-1)*40+j}=icasig(1:64,j);
        end
end

%% Obtención de vectores características de entrenamiento
xRand = 170;
yRand = 128;
for i=1:numel(dir1)
        featVectorsTrain{i}=cell(1,400);
        texture=double(imread(strcat(categoria,dir1(i).
            name)));                    subTexture=texture(
            xRand:xRand+windowSize-1,yRand:yRand+
            windowSize-1);                      subTexture=
            reshape(subTexture,windowSize*windowSize,1);
        for j=1:numel(dir1)*dimensions
                featVectorsTrain{i}{j}=dot(subTexture,Ics
                    {j});
    end
        featVectorsTrain{i}=featVectorsTrain{i}';
end
menor=0;
texto='';
for i=1:numel(dir1)
        column=abs(cell2mat(featVectorsTrain{i}));
        column=sort(column,'descend');
        column=column(1:3,1);
        featVectorsTrain{i}=column;
end

%% Obtención de vectores características de prueba y su
    comparación
%% mediante distancia Euclidea
dir2                = dir('images/din/otra10/*.jpg');
xRand = 170;
yRand = 128;
superMayor = 999999999;
```

```
for i=1:numel(dir2)
        texture=double(imread(strcat('images/din/otra10/'
            ,dir2(i).name)));        subTexture=texture(
            xRand:xRand+windowSize-1,yRand:yRand+
            windowSize-1);
    subTexture=reshape(subTexture,windowSize*windowSize
        ,1);
        featVectorTest(1)=0;
    for j=1:400
        featVectorTest(j,1)=dot(subTexture,Ics{j});
    end
        featVectorTest=abs(featVectorTest);
        featVectorTest=sort(featVectorTest,'descend');
                    featVectorTest=featVectorTest(1:3,1);
    mayor=999999999999999;
    for j=1:30
        distance=norm(featVectorTest - featVectorsTrain{i
            });
        if(distance<mayor)
            mayor=distance;
        end
        end
        if(superMayor>mayor)
                superMayor = mayor;
                texto = dir2(i).name;
        end
end
texto
```