La Columna de Matemática Computacional

Sección a cargo de

Tomás Recio

En este número... se describe una experiencia docente desarrollada en un primer curso universitario cuyo objetivo era presentar a los alumnos aplicaciones reales y útiles de las Matemáticas. Con esto perseguimos atraer la atención del alumnado hacia nuestras asignaturas. Así veremos cómo cifrar una imagen digital, cómo se puede comprimir una imagen, cómo ocultar un mensaje secreto en una imagen sin que nadie sospeche, cómo compartir una imagen secreta entre varias personas de modo que sólo cuando se junten un número determinado de personas se pueda recuperar la imagen secreta o, por último, cómo funciona el reconocimiento automático de rostros o caras de personas. Todas estas aplicaciones usan imágenes digitales. Las imágenes digitales se convierten, por tanto, en una excelente herramienta docente en clase de Matemáticas.

Trabajando con imágenes digitales en clase de Matemáticas

por

Ángela Rojas Matas y Alberto Cano Rojas

1. Introducción

Los profesores de Matemáticas, sobre todo de titulaciones de Ciencias e Ingeniería, debemos presentar a nuestros alumnos aplicaciones de las Matemáticas en áreas de interés para sus estudios. De esta forma conseguiremos que nuestros alumnos aprendan Matemáticas sabiendo para qué les pueden servir y una mayor motivación de los mismos hacia la asignatura. En este trabajo se muestran unas propuestas desarrolladas con alumnos de primer curso de Ingeniería Técnica Informática. Veremos cómo las imágenes digitales pueden ser una magnífica herramienta en clase de Matemáticas.

Las imágenes digitales son matrices y por eso existe una relación evidente entre imágenes y Matemáticas. Podemos utilizar imágenes para trabajar con conceptos matemáticos desarrollados en clase. En este artículo se van a exponer algunos de los ejemplos que hemos propuesto a nuestros alumnos. Algunos de estos ejemplos eran

realmente sencillos y se podían resolver sin ninguna dificultad en las clases prácticas desarrolladas en nuestro Laboratorio de Matemáticas, que dispone de ordenadores. Otros requerían algo más de esfuerzo y fueron resueltos como actividades académicas dirigidas y presentados a final de curso como trabajo de la asignatura. Se puede emplear cualquier software para llevar a cabo la experiencia. En nuestro caso utilizamos Mathematica pero también se puede utilizar Matlab o su análogo libre Octave.

Esta experiencia se llevó a cabo con alumnos de la asignatura Matemáticas II (Matemática Discreta y Álgebra Lineal) de primer curso. Las aplicaciones se escogieron teniendo en cuenta tres factores:

- Debían estar relacionadas con temas del área de la Computación ya que nuestros alumnos eran de Ingeniería Técnica Informática.
- Debían ser atractivas y actuales para así motivar más a los alumnos.
- El grado de dificultad para abordarlas no debía ser excesivo para que pudiesen ser trabajadas por alumnos de primer curso. Además debían aparecer conceptos matemáticos tratados en la asignatura Matemáticas II.

El principal problema con el que nos encontramos fue que una parte del alumnado de primero abandonó la asignatura de Tecnología y Metodología de la Programación y tenían, por lo tanto, problemas para hacer los códigos de las prácticas de Matemáticas II (aunque éstos eran en su mayoría muy sencillos), ya que no sabían prácticamente nada de programación. Sin embargo, el resultado de la experiencia ha sido satisfactorio porque una parte importante de los alumnos mostró una actitud positiva.

La organización del presente trabajo es como sigue: en la sección 2 se verá cómo cifrar imágenes digitales con sencillas técnicas matriciales, en la sección 3 se estudiará la compresión de imágenes digitales, la sección 4 está dedicada a la ocultación de secretos en una imagen digital, en la sección 5 se estudiará cómo compartir un secreto entre varias personas y, por último, en la sección 6 estudiaremos el reconocimiento automático de rostros o caras de personas.

2. Cifrado de imágenes digitales

Lester Hill propuso en 1929 un método matricial para cifrar un mensaje de texto [6]. No sólo es interesante el cifrado de mensajes de texto sino que también es interesante el cifrado de imágenes digitales, ficheros de audio, vídeo, etc. La idea de Hill es bastante sencilla, como vamos a exponer a continuación directamente adaptada al caso de imágenes digitales.

Una imagen digital en escala de grises no es más que una matriz de números donde cada número indica el nivel de gris de un píxel. Por ejemplo, la imagen de la izquierda de la figura 1 es una matriz de tamaño 256×256 donde los niveles de gris de la imagen varían desde 0 correspondiente al negro hasta 255 correspondiente al blanco. Los números comprendidos entre 0 y 255 se escriben en binario con 8 bits, por eso esta imagen necesita 1 byte por píxel.

Usaremos una matriz secreta K sólo conocida por emisor y receptor, por ejemplo de tamaño 2×2. Cogeremos los niveles de gris de la imagen A de dos en dos,

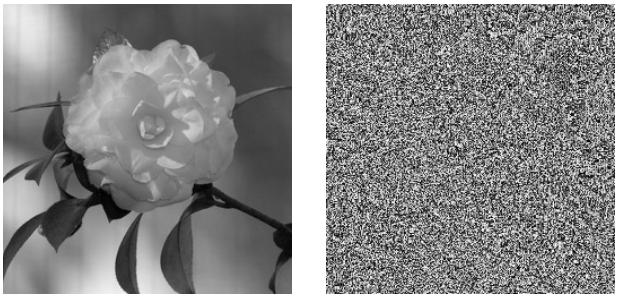

Figura 1: Imagen original e imagen cifrada con el método de Hill.

empezando en la esquina superior izquierda y moviéndonos de izquierda a derecha y de arriba a abajo. Así el primer bloque será a_{11}, a_{12}, el segundo bloque será a_{13}, a_{14}, etc. Supongamos que los dos niveles de gris que nos toca coger los indicamos por g_1, g_2. Los transformamos en otros dos diferentes haciendo la operación

$$K \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} g_1' \\ g_2' \end{pmatrix} \pmod{256}.$$

Podríamos tomar como K la matriz

$$K = \begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix}$$

y si los dos primeros niveles de gris fueran, por ejemplo, 125 y 137, se obtendría

$$\begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix} \begin{pmatrix} 125 \\ 137 \end{pmatrix} = \begin{pmatrix} 7420 \\ 13073 \end{pmatrix} = \begin{pmatrix} 252 \\ 17 \end{pmatrix} \pmod{256}.$$

Es necesario hacer la congruencia módulo 256 para obtener siempre un nivel de gris válido, es decir, un número entre 0 y 255. De esta forma, los dos niveles de gris originales, que eran 125 y 137, se transformarán en 252 y 17 respectivamente. A la izquierda de la figura 1 podemos ver la imagen original y a la derecha la imagen cifrada usando la clave anterior.

Hay que hacer una observación importante: no vale cualquier matriz clave K, ya que si K no es inversible módulo 256 no podremos descifrar. Por ejemplo, si se usa la matriz

$$K = \begin{pmatrix} 20 & 8 \\ 15 & 7 \end{pmatrix},$$

el emisor podrá cifrar la imagen, pero el receptor no podrá descifrarla, por lo tanto no sirve para nada. Veamos por qué. Si $K = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ y $|K| = ad - bc \neq 0 \mod 256$, entonces $K^{-1} = \frac{1}{ad-bc} \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$ siempre que $|K|$ sea un número inversible cuando se trabaja módulo 256, lo cual es equivalente a que $|K|$ sea un número primo relativo con 256. Por esta razón, $K = \left(\begin{smallmatrix} 20 & 8 \\ 15 & 7 \end{smallmatrix}\right)$ no es una matriz de cifrado válida ya que $|K| = 20$ y este número no es primo relativo con 256. Sin embargo, si $K = \left(\begin{smallmatrix} 21 & 35 \\ 18 & 79 \end{smallmatrix}\right)$

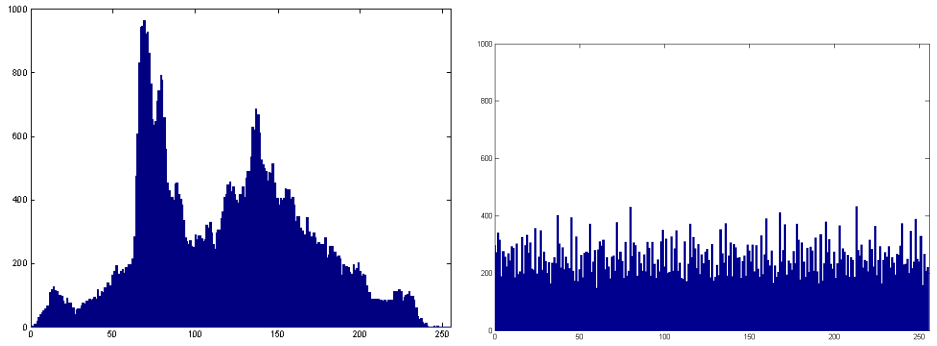

Figura 2: Histogramas de la imagen original y de la imagen cifrada.

resulta que $|K| = 1029 = 5 \pmod{256}$. Como 5 tiene inverso módulo 256, ya que ambos números son primos relativos, podemos usar esta matriz como clave en el proceso. El inverso de 5 módulo 256, que se puede encontrar usando el algoritmo extendido de Euclides, resulta ser 205. Entonces la matriz de descifrado será

$$K^{-1} = 205 \begin{pmatrix} 79 & -35 \\ -18 & 21 \end{pmatrix} = \begin{pmatrix} 67 & 249 \\ 150 & 209 \end{pmatrix} \pmod{256}.$$

El receptor usará la matriz de descifrado anterior y podrá recuperar la imagen original. Como se puede comprobar en [7], se sigue trabajando en este tipo de cifrado.

El histograma de la imagen original y el de la imagen cifrada serán distintos, como se muestra en la figura 2. Podemos ver que los niveles de gris entre 0 y 255 se distribuyen de una forma más uniforme en la imagen cifrada que en la imagen original.

A nuestro alumnado se le proporcionaba la imagen cifrada de la derecha de la figura 1 y la clave K usada, y en clases prácticas su trabajo consistía en descifrarla usando para ello Mathematica.

Vamos a ver a continuación otro método de cifrado también matricial parecido al anterior, pero con una idea algo diferente. En este caso, lo que haremos será cambiar los píxeles de posición. Supongamos, por ejemplo, que tenemos una imagen en color, aunque el proceso que vamos a seguir también puede aplicarse a imágenes en escala de grises. Una imagen en color en formato RGB (Red Green Blue), se compone de tres capas o planos de color: rojo, verde y azul. Cada capa es una imagen del mismo tamaño que la original donde cada dato se interpreta como la cantidad de rojo, verde o azul presente en el color del píxel. Los datos de cada capa de color ocupan un byte: desde el 0 hasta el 255.

Supongamos que tenemos una imagen $N \times N$. Indicamos por (x, y) a las coordenadas de los píxeles, siendo x la fila e y la columna, variando tanto x como y entre 0 y $N-1$. Ahora vamos a llevar el píxel situado en la posición (x, y) a otra posición (x', y') de la forma

$$K \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} \pmod{N}.$$

Figura 3: Imagen original e imagen cifrada.

Hemos de tener en cuenta que la matriz K debe ser de nuevo inversible, para que podamos después descifrar, por lo que el determinante de K debe ser primo relativo con el módulo N. Al hacer esta operación con todos los píxeles de la imagen, lo que estamos haciendo es una permutación de los mismos.

En la figura 3 podemos ver a la izquierda la imagen original en color y a la derecha la imagen cifrada. Al igual que antes, en clases prácticas se proporcionaba a los alumnos la imagen cifrada y su trabajo consistía en descifrarla.

Este segundo método se incluye sólo con fines docentes ya que no sería seguro. El histograma de la imagen original y el de la imagen cifrada son exactamente el mismo lo que posibilita un ataque estadístico. En este método no se cifra la información sino que sólo se reordena. Sin embargo, sí que puede ser útil como paso previo para un algoritmo de cifrado: primero se cambian los píxeles de posición (confusión) y segundo se cambian los niveles de gris o el color de los píxeles por otros distintos (difusión).

Hemos trabajado también con otros métodos de cifrado de imágenes, pero por falta de espacio no los hemos incluido. Entre ellos hemos visto cómo cifrar una imagen usando el teorema chino de los restos, usando secuencias caóticas, usando aritmética del cuerpo finito $GF(256)$, etc. Se puede ver en [5, 8, 16] cómo se está investigando precisamente en estos métodos de cifrado de imágenes digitales.

3. Compresión de imágenes digitales

La compresión de una imagen digital permite que dicha imagen ocupe menos espacio. Esto provocará que la imagen viaje más rápido por Internet, que ocupe menos memoria en nuestros discos duros, etc. En clase hemos estudiado dos técnicas de compresión de una imagen digital. Una basada en la descomposición en valores singulares de una matriz, y otra es la conocida compresión JPEG que se puede implementar con técnicas del Álgebra Lineal. En esta sección describiremos sólo la primera ya que la descomposición en valores singulares será también usada en otra sección posterior. Por falta de espacio no abordaremos la compresión JPEG.

La descomposición en valores singulares es una de las herramientas más útiles del Álgebra Lineal. Esta aplicación puede verse publicada en los artículos [10, 15]

ya que, de hecho, es una de las aplicaciones más conocidas de esta descomposición matricial.

Dada A una matriz real de dimensiones $m \times n$, existe una matriz U ortogonal de orden m, una matriz S «diagonal» de dimensiones $m \times n$ y una matriz V ortogonal de orden n de forma que

$$A = USV^t.$$

Los elementos no nulos de la diagonal de S se denominan *valores singulares* de A y son números no negativos, pues son la raíz cuadrada de los autovalores no nulos de la matriz A^tA o de AA^t, ya que ambos coinciden (salvo el autovalor nulo, si es que lo tienen). Las columnas de U y V se denominan vectores singulares por la izquierda y por la derecha respectivamente. Esta descomposición matricial se conoce como **SVD** (**S**ingular **V**alue **D**ecomposition).

Una propiedad muy interesante de la descomposición SVD de una matriz A es que el número de valores singulares no nulos de A coincide con el rango de dicha matriz. Supongamos que una matriz A es de rango r y que los valores singulares no nulos están ordenados en orden decreciente:

$$s_1 \geq s_2 \geq \cdots \geq s_r > 0.$$

Se puede comprobar que

$$A = USV^t = s_1u_1v_1^t + s_2u_2v_2^t + \cdots + s_ru_rv_r^t, \tag{1}$$

donde s_j es un escalar (el valor singular correspondiente), mientras que u_j es la columna j-ésima de la matriz U y v_j es la columna j-ésima de la matriz V.

Si una matriz tiene rango r, entonces, según (1), se podrá expresar como la suma de r matrices de rango unidad. Es lógico pensar que si los valores singulares son muy pequeños, entonces la suma (1) se podrá truncar en el momento en que la magnitud de estos valores singulares sea suficientemente pequeña, obteniendo de esta forma aproximaciones de la matriz A original. En efecto esto es así, ya que si llamamos

$$A_1 = s_1u_1v_1^t,$$

esta matriz es de rango 1 y es la matriz de rango 1 más parecida a A. Si llamamos

$$A_2 = s_1u_1v_1^t + s_2u_2v_2^t,$$

esta matriz es de rango 2 y también resulta ser la matriz de rango 2 más parecida a A. Y así sucesivamente, de modo que la matriz de rango k más parecida a la original es la matriz

$$A_k = s_1u_1v_1^t + s_2u_2v_2^t + \cdots + s_ku_kv_k^t,$$

siendo $1 \leq k \leq r$. Una definición precisa de lo que entendemos por matriz más «parecida» a la original puede encontrarse en Noble [11].

En la figura 4 se muestra una imagen 512×512 y su aproximación de rango 50. En la imagen original tenemos $512 \times 512 = 262\,144$ datos, mientras que en la

Figura 4: Imagen original 512×512 y aproximación de rango 50.

aproximación de rango 50 tenemos 50 vectores singulares por la izquierda (cada uno con 512 componentes), 50 vectores singulares por la derecha (cada uno con 512 componentes) y 50 valores singulares, lo que hace un total de 51 250 datos. Queda patente la reducción conseguida en el número de datos.

El trabajo realizado por nuestros alumnos en clase consistía en hallar aproximaciones de bajo rango de una imagen dada usando Mathematica. En [12, 13, 14] se puede comprobar cómo se sigue investigando en el uso de SVD para comprimir una imagen.

4. Ocultación de secretos en una imagen

Existen ocasiones donde se desea mantener en secreto alguna información: transacciones bancarias, secretos militares, patentes industriales, comercio electrónico, etc. Esta información viajará probablemente por canales inseguros como Internet y podrá ser interceptada por algún intruso mal intencionado. Por lo tanto, es muy importante estudiar formas de intercambiar información de forma segura entre dos usuarios.

La criptografía se ocupa del cifrado de la información de modo que resulte ininteligible para un usuario no autorizado. Por ejemplo:

```
-----BEGIN PGP MESSAGE-----
qANQR1DBwU4DxkriL8wrACgQB/4nWbELJMR/Rt8RkkL1qkwZJWg2nyWges
LTwIAFZ+ZKMVfhyxEmTvQuIPCL7DsX2c01HhlPk6hgAWA9EN+G7sduA4AD
-----END PGP MESSAGE-----
```

es un mensaje de texto cifrado con PGP (Preety Good Privacy), una herramienta muy popular que nos permite, entre otras cosas, el intercambio de correos cifrados entre dos usuarios (la versión gratuita del programa PGP permite esto, además de algunas otras cuestiones).

Sin embargo puede resultar mucho más interesante que la información que se desea mantener en secreto viaje oculta en un fichero digital de cobertura totalmente inocente como una imagen de unas vacaciones familiares en la playa, de modo que no levante sospechas. Si un intruso intercepta una imagen de este tipo probablemente

Figura 5: Imagen con marca visible.

no sospeche nada y, a pesar de eso, oculta en la imagen puede haber información secreta. De esto se ocupa la *esteganografía digital*.

La palabra esteganografía procede de un tratado del monje alemán Johannes Trithemius llamado Steganographia (publicado en el año 1500). La esteganografía digital ha adquirido mayor relevancia en la comunidad científica en los últimos años, al correr el rumor por Internet de que Al Qaeda había utilizado este método para cometer los atentados de las Torres Gemelas de 2001. En la actualidad, los vídeos de terroristas preparados para su difusión por televisión o por Internet son minuciosamente analizados para detectar si contienen información oculta.

Una de las más importantes aplicaciones de la esteganografía digital son *las marcas de agua*, conocidas en inglés por *watermarking*. Las marcas de agua se añaden a un objeto con intención de identificar al propietario del mismo. Sirve para proteger los derechos de autor. Pueden ser visibles o invisibles. En la figura 5 se muestra una imagen digital marcada con una marca visible para proteger los derechos de autor. Las invisibles están íntimamente ligadas a la esteganografía.

Un ejemplo curioso: se está investigando en cómo incorporar pequeñas mutaciones en el ADN de organismos a modo de marca de agua para identificar «copias no autorizadas» de organismos genéticamente manipulados. Se escoge una célula y se le hace la prueba de ADN: «Patentado por XYZ». Esto suena a ciencia ficción.

La esteganografía digital se ocupa de la ocultación de información que se desea mantener en secreto en un objeto de cobertura «inocente». La información a ocultar puede ser de distinto tipo: un mensaje de texto, una imagen digital o un fichero de audio. En los tres casos, la información secreta a ocultar se convertirá en una secuencia de bits: ceros y unos. Esta secuencia de bits será almacenada en un fichero digital, una imagen por ejemplo, que nos va a servir de cobertura.

Para ocultar la información secreta, la imagen original será ligeramente modificada por el algoritmo de ocultación utilizado obteniendo una nueva imagen que se conoce con el nombre de *estego-imagen*. Comenzamos comentando en qué consiste el método esteganográfico más usado: el método LSB (**L**east **S**ignificant **B**it), o método del bit menos significativo.

Vamos a trabajar con una imagen con 256 niveles de gris, que varían desde el 0 (negro) al 255 (blanco). Para almacenar un nivel de gris necesitamos 8 bits, es decir, 1 byte. Si el nivel de gris de un píxel es 127, es decir 01111111 en binario, y cambiamos el último bit obtenemos 01111110, es decir 126. El ojo humano no distingue esta pequeña alteración en el nivel de gris de un píxel. Ésta es la base

del método LSB, usar este último bit de cada byte para ocultar en ellos bits de información secreta.

Una imagen, como la que vamos a considerar en nuestro primer ejemplo, de tamaño 256×256, se compone de 65 536 bytes con niveles de gris. Si usamos solamente el último bit, es decir, el bit menos significativo, el número máximo de bits que se pueden ocultar en dicha imagen sería 65 536 bits.

Supongamos que la información secreta que deseamos ocultar es un mensaje de texto: «LA CRIPTOGRAFÍA ES LA CIENCIA...» (es más largo pero sólo hemos escrito los primeros caracteres del mensaje). Supongamos que usamos los códigos ANSI donde al letra L se convierte en 76, la letra A se convierte en 65, el espacio en blanco es 32, etc. El código ANSI asocia un número entre 0 y 255 a cualquier letra o carácter de nuestro mensaje. Entonces el mensaje de texto dará lugar a una secuencia de números: $\{76, 65, 32, \ldots\}$. Convertiremos a binario con una cadena de 8 bits para cada número, obteniendo $\{01001100, 01000001, 00100000, \ldots\}$. En definitiva, la información secreta a ocultar es una secuencia de bits.

Supongamos que deseamos almacenar un bit secreto en el bit menos significativo del nivel de gris de un píxel. La forma de proceder sería como sigue. Supongamos que el nivel de gris del píxel es 112, que en binario se escribe como 01110000; en este caso:

- si el bit a ocultar es 0 entonces no hacer nada, es decir:
  nivel de gris modificado $= 01110000_2 = 112$
- si el bit a ocultar es 1, entonces modificar el bit menos significativo, es decir:
  nivel de gris modificado $= 01110001_2 = 113$

De forma similar se razona si el nivel de gris es impar, por ejemplo 127 que en binario es 01111111; ahora:

- si el bit a ocultar es 0 entonces modificar el bit menos significativo, es decir:
  nivel de gris modificado $= 01111110_2 = 126$
- si el bit a ocultar es 1 entonces no hacer nada, es decir:
  nivel de gris modificado $= 01111111_2 = 127$

Usando el método LSB se obtiene el resultado mostrado en la figura 6. La imagen de la izquierda es la imagen original que nos ha servido de cobertura para ocultar un mensaje de más de 8 000 caracteres. En la imagen de la derecha se muestra la estego-imagen resultante.

Si en cambio utilizamos los dos últimos bits (los dos bits menos significativos) para ocultar información, tampoco será apreciable la diferencia para el ojo humano. El número máximo de bits que podremos ocultar si usamos los dos bits menos significativos será $256 \times 256 \times 2 = 131\,072$. Como vemos aumenta considerablemente la capacidad de ocultación. Pero si usamos una imagen en color todavía tendremos mayor capacidad de ocultación, de hecho tendremos el triple de capacidad para ocultar información que una imagen en blanco y negro. Así, por ejemplo, para una imagen en color de tamaño 256×256, si usamos los dos bits menos significativos podríamos ocultar hasta $256 \times 256 \times 3 \times 2 = 393\,216$ bits. Para hacernos una idea, el primer capítulo del Quijote contiene 10 350 caracteres (incluidos los espacios en

Figura 6: Imagen original y estego-imagen.

blanco), y el código ANSI de cada carácter se convierte en 8 bits, por lo tanto el número total de bits será $10\,350 \times 8 = 82\,800$. De manera que podríamos almacenar cuatro veces el primer capítulo del Quijote en una imagen en color de tamaño 256×256 usando los dos bits menos significativos.

Esta técnica es muy utilizada en la práctica. Sin embargo, no es resistente a ataques de ningún tipo. El formato de imagen que se emplea es el de una imagen en mapa de bits, por ejemplo BMP. Una simple compresión JPEG, formato muy habitual que usamos para que una imagen ocupe menos espacio, destruye por completo el mensaje oculto. Este ataque puede ser no intencionado, pero también puede que la estego-imagen sea sometida a ataques malintencionados con el objetivo de destruir el mensaje oculto. Por ese motivo, se buscan métodos más robustos capaces de resistir los ataques más frecuentes a los que son sometidos las estego-imágenes.

A continuación vamos a describir un método que usa la descomposición en valores singulares SVD que ha sido explicada en la sección anterior. La imagen se divide en submatrices de tamaño, por ejemplo, 2×2. A cada una de estas submatrices se le calcula la descomposición SVD. Supongamos que A es una de estas submatrices, entonces se calcula

$$A = USV^t.$$

Vamos a ocultar 1 bit secreto en el primer valor singular (el mayor) de la siguiente forma: se divide el valor singular más grande s_1 entre una constante secreta q, por ejemplo $q = 10$, y nos quedamos con el cociente entero c. Entonces, razonamos como sigue:

- Si c es par y el bit que toca ocultar es 0, no hacer nada, es decir, $c^* = c$.
- Si c es par y el bit que toca ocultar es 1, entonces incrementar c en una unidad, es decir $c^* = c + 1$.
- Si c es impar y el bit que toca ocultar es 1, no hacer nada, es decir $c^* = c$.
- Si c es impar y el bit que toca ocultar es 0, entonces decrementar c en una unidad, es decir $c^* = c - 1$.

Observar que lo que se hace es emplear el método LSB pero en lugar de hacerlo con el nivel de gris original se hace con el coeficiente c anterior.

Figura 7: Imagen original y estego-imagen con SVD.

A continuación calculamos el valor singular modificado:

$$s_1^* = c^* q + \frac{q}{2}$$

y después calculamos la matriz modificada:

$$A^* = U \begin{pmatrix} s_1^* & 0 \\ 0 & s_2 \end{pmatrix} V^t.$$

Entonces la matriz A^* ocupará la posición de A en la imagen. En la figura 7 se muestra a la izquierda la imagen original y a la derecha la estego-imagen.

Si la estego-imagen se comprime con JPEG a un 90 por ciento de calidad se destruyen sólo una pequeña cantidad de bits y el mensaje extraído se puede leer. Esto no ocurre con el método LSB.

A nuestros alumnos se les proporcionaba la estego-imagen anterior y una descripción minuciosa del proceso seguido. Su trabajo consistía en extraer el mensaje oculto.

Como dijimos al comienzo de esta sección, uno de los principales usos de la esteganografía es el marcado digital para proteger los derechos de autor. Podemos comprobar en [2, 4, 9, 19] cómo se está investigando en el uso de SVD para marcar imágenes digitales.

5. Reparto de secretos

Existen ocasiones donde una información secreta no es deseable que esté en manos de una sola persona. Puede interesar que varias personas posean parte de dicha información y que sólo se consiga recuperar la información secreta completa si juntamos a varias de estas personas. Por ejemplo, a un banco puede que le interese que ninguno de sus empleados posea la clave que abre la caja fuerte. Por el contrario, puede repartir entre 6 empleados, por ejemplo, parte de la información, de forma que para conseguir la clave de la caja fuerte tengan que juntarse al menos 3 de los 6 empleados.

Los esquemas de reparto de secretos, conocidos también como *esquemas umbrales*, son protocolos criptográficos para compartir un secreto entre un conjunto de participantes de modo que sólo cuando se junten un número especificado de participantes sea posible recuperar el secreto. Fueron introducidos simultáneamente y de manera independiente por A. Shamir [17], con una técnica que describiremos a continuación, y por G. R. Blakey [3].

Vamos a desarrollar un esquema umbral (n, k), es decir, hay un total de n participantes y sólo al juntar como mínimo a k de ellos será posible recuperar el secreto. Lógicamente $k \leq n$. Supongamos que el secreto es un número S que el dueño del banco quiere proporcionar a sus empleados y que es la clave de la caja blindada. Para ello, formará un polinomio $P(x)$ de grado $k-1$ donde los coeficientes se eligen al azar salvo el término independiente que se hará coincidir con el número secreto S. Entonces, el dueño del banco calcula

$$P(1), P(2), \ldots, P(n).$$

Al empleado número 1 se le proporciona el par $(1, P(1))$, al empleado número 2 se le proporciona el par $(2, P(2))$ y así sucesivamente, hasta el empleado n al que se le proporciona el par $(n, P(n))$. Hemos elegido $1, 2, \ldots, n$ por simplicidad, pero se pueden elegir evidentemente otros valores $x_1, x_2, \ldots, x_n$ para evaluar el polinomio en ellos.

Cuando se junten al menos k de ellos tendremos datos suficientes para averiguar el polinomio $P(x)$, es decir, sus k coeficientes. Bastará con evaluar el polinomio en cero para obtener el secreto: $S = P(0)$. Si se juntan menos de k participantes no tendremos posibilidad de calcular el secreto.

Por ejemplo, supongamos que $S = 1234$ y que vamos a hacer un esquema umbral $(4, 3)$. El dueño elige el polinomio

$$P(x) = S + a_1 x + a_2 x^2 = 1234 + 166x + 94x^2,$$

calcula $P(1) = 1494$, $P(2) = 1942$, $P(3) = 2578$, $P(4) = 3402$. Cuando se junten al menos tres de los participantes será posible recuperar el secreto. Por ejemplo, si se juntan los empleados 1, 3 y 4, se obtendrá el sistema lineal

$$\begin{aligned} S + a_1 + a_2 &= 1494 \\ S + 3a_1 + 9a_2 &= 2578 \\ S + 4a_1 + 16a_2 &= 3402 \end{aligned}$$

cuya solución nos permitirá obtener los tres coeficientes del polinomio y, por lo tanto, nos permitirá recuperar el secreto.

En el artículo original de Shamir [17] se propone elegir un número primo p, siendo el secreto S un número entre 0 y $p-1$. El dueño del banco, que desea llevar a cabo un esquema umbral (n, k), construiría un polinomio de grado $k-1$ donde el término independiente se hace coincidir con el secreto y el resto de coeficientes del polinomio se elegirán aleatoriamente con números entre 0 y $p-1$. A continuación se razona como antes, sólo que ahora se trabaja con congruencias módulo p.

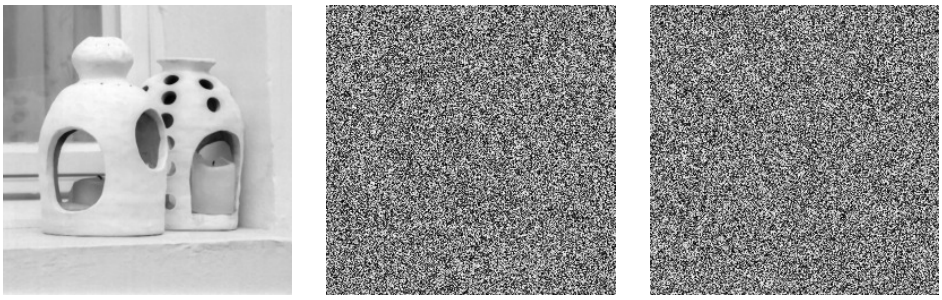

Figura 8: Imagen original y las dos sombras.

Vamos a aplicar esta idea para compartir una imagen digital tal y como se describe en [18]. Supongamos que tenemos una imagen digital en escala de grises con 256 niveles de gris, variando entre 0 y 255, que queremos compartir con el esquema umbral (n, k). Escogemos un número primo p. Sea a_0 el nivel de gris de un píxel. El dueño de la imagen crea el polinomio de grado $k-1$ de la forma

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1},$$

y después calcula $S_1 = P(1) \pmod p$, $S_2 = P(2) \pmod p$,..., $S_n = P(n) \pmod p$.

Cuando se junten al menos k de los empleados se podrá recuperar el nivel de gris original a través de la resolución del sistema correspondiente. La elección del número p primo garantiza que el sistema siempre tenga solución. Escogemos como $p = 251$ que es el mayor número primo menor o igual que 256.

Para aplicar este esquema los niveles de gris de la imagen tienen que variar entre 0 y 250. Los niveles de gris de la imagen mostrada a la izquierda de la figura 8 están dentro de este rango. Esto no es infrecuente. De todos modos, si no fuera así, se puede modificar el algoritmo como se indica en [18].

Cada S_i será un número entre 0 y 250 que se puede interpretar como un nivel de gris. Para cada nivel de gris a_0 de la imagen original se calculará un polinomio $P(x)$ diferente eligiendo aleatoriamente los coeficientes entre 0 y 250. Razonando de esta forma, y píxel a píxel, construiremos para cada participante una imagen con aspecto aleatorio que se suele llamar *sombra*. En la figura 8 se muestra el resultado de un esquema umbral $(2, 2)$, con la imagen original a la izquierda y las dos sombras a su derecha.

En este caso se proporcionaba a nuestros alumnos las sombras y debían recuperar la imagen secreta. El tema de reparto de imágenes digitales atrae también la atención de investigadores de todo el mundo, como puede comprobarse en [1, 23].

6. Reconocimiento automático de caras

Esta última sección se dedica a una aplicación más compleja. Sólo fue abordada por un grupo de alumnos como actividades académicas dirigidas para que estudiaran una aplicación real de la diagonalización de matrices.

Se está investigando mucho actualmente en el tema de reconocimiento automático. Hay mucho interés comercial en ello. Concretamente se estudia cómo reconocer automáticamente a una persona a través de sus huellas dactilares, del iris de los ojos, de la cara, de la voz, etc. El reconocimiento automático de caras interesa a las empresas por distintas razones:

- Como mecanismo para permitir el acceso a un sistema restringido: por ejemplo, el acceso a un ordenador se puede conseguir, en lugar de con claves complejas, con un mecanismo de reconocimiento de caras con una webcam.
- Como método policial para el reconocimiento automático de delincuentes: imaginemos su uso en un aeropuerto o en una estación de tren, donde pasan cientos de personas cada día.

El método que vamos a describir en primer lugar se publicó en 1991 [20] y fue patentado por el MIT. Es todo un clásico en el tema de reconocimiento automático de caras. A raíz de su publicación han surgido, a través de nuevas publicaciones, multitud de variaciones de la idea original, como [21, 22], entre otros, que han logrado, generalmente, mejorar las tasas de aciertos y los tiempos de computación. En el trabajo descrito en [20] se introdujo por primera vez el término *autocaras*, que tiene que ver con el término *autovectores* del Álgebra Lineal.

En esta sección se va a usar una base de datos de caras de personas conocida como ORL, de los laboratorios ATT de la Universidad de Cambridge, que está disponible libremente para su uso por investigadores de todo el mundo. La base de datos ORL contiene 400 imágenes, donde cada imagen es de tamaño 112×92, de un total de 40 personas distintas, con 10 imágenes por persona. Las imágenes de cada persona varían en posición (de frente, de lado, etc.), de actitud (sonriente, neutral, etc.), con gafas o sin ellas. En la figura 9 se muestran algunas de las imágenes disponibles de cuatro individuos.

En un sistema de reconocimiento automático, lo que se hace en primer lugar es *entrenar* al sistema para que pueda reconocer a un conjunto determinado de personas. Después viene la fase *test* donde se debe poner en práctica lo aprendido. En nuestro caso, se escogieron aleatoriamente 5 imágenes de cada individuo para la fase de entrenamiento, y las 5 restantes para la fase test. De manera que el conjunto de entrenamiento estuvo formado por 200 caras y el conjunto test por las otras 200.

En la fase test el sistema se enfrentará a imágenes con identidad desconocida, no incluidas en la fase de entrenamiento, como las mostradas en la figura 10, deberá compararlas con las almacenadas en su conjunto de datos de entrenamiento y nos dirá cuál es la identidad de cada imagen test.

Para resolver este problema, el sistema podría buscar la imagen más parecida de su base de datos a la imagen test proporcionada. Esta idea se conoce con el nombre de *búsqueda del vecino más cercano*.

Cada imagen test T es una matriz 112×92 y cada una de las imágenes E de entrenamiento es también 112×92. La distancia entre ambas se puede medir, por ejemplo, como la suma de los cuadrados de las diferencias entre los elementos de

Figura 9: Algunas de las imágenes de entrenamiento de ORL.

Figura 10: Algunas de las imágenes test de ORL.

ambas matrices:

$$\text{distancia}(T, E) = \sum_{i=1}^{112} \sum_{j=1}^{92} (T_{ij} - E_{ij})^2.$$

La estrategia del *vecino más cercano* es muy sencilla, pero en la práctica requiere mucho tiempo de computación cuando la base de datos es muy grande, con miles de imágenes. Por esta razón, se buscan métodos que puedan simplificar estas comparaciones para conseguir hacer el sistema mucho más rápido, es decir, *aligerar* el

problema. Hemos utilizado lo que se llama una *reducción de dimensionalidad*. La idea es sencilla: manejamos imágenes de $112 \times 92 = 10\,304$ datos cada una; si consiguiéramos reducir el número de datos de cada imagen a 8 solamente, por ejemplo, el problema se aliviaría significativamente.

En el artículo [20], se convierten las matrices 112×92 en vectores unidimensionales de 10 304 componentes. Se buscará un subespacio vectorial de dimensión reducida k, por ejemplo $k = 8$, donde proyectar los vectores de entrenamiento y los vectores test de manera que en lugar de trabajar con las imágenes originales se trabajarán con las coordenadas de dichas proyecciones. De esta forma pasamos de trabajar con vectores de 10 304 componentes a trabajar con vectores con sólo 8 componentes. La búsqueda del vecino más cercano será ahora mucho más rápida.

Ahora bien, ¿cómo calculamos el subespacio óptimo donde proyectar las imágenes? Turk et. al. [20] buscaban el subespacio de forma que las proyecciones estuvieran lo más separadas posible unas de otras, es decir, se buscaba maximizar la varianza de las proyecciones. La búsqueda de esta base óptima de proyección se conoce como *Análisis de Componentes Principales*, en inglés **P**rincipal **C**omponent **A**nalysis, simplificadamente PCA.

La matriz de proyección W se formará con los vectores de una base ortonormal del subespacio óptimo anterior escritos por columnas. Supongamos que decidimos quedarnos con una base formada por $k = 8$ vectores. Cada vector de la base tendrá 10 304 componentes. Formamos la matriz W escribiendo en cada columna de esta matriz un vector de la base, de modo que W tendrá dimensión $10\,304 \times 8$. Las coordenadas de la proyección de un vector E de 10 304 componentes sobre este subespacio vendrán dadas por

$$Y = W^t E,$$

y el vector Y resultante tendrá sólo 8 componentes en nuestro caso.

En el artículo [20] se demostró que la forma de obtener esta matriz de proyección óptima es la siguiente:

- Cada imagen del conjunto de entrenamiento se transforma en un vector unidimensional Γ_i, con $i = 1, 2, \ldots, M$, siendo M el número de caras del conjunto de entrenamiento. En nuestro caso $M = 200$.
- Se calcula la media Ψ de todas los vectores de entrada mediante

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i.$$

- A cada vector Γ_i se le resta la media Ψ, obteniendo el vector X_i de la forma

$$X_i = \Gamma_i - \Psi,$$

  para $i = 1, 2, \ldots, M$.
- Formamos la matriz A escribiendo los vectores X_i anteriores por columnas, es decir

$$A = (X_1 \;\; X_2 \;\; \ldots \;\; X_M).$$

  La matriz A tiene dimensión $10\,304 \times M$.

- Calculamos la matriz C de la forma

$$C = \frac{1}{M} A A^t.$$

Esta matriz se conoce en Estadística como *matriz de covarianza*. Se puede reescribir como

$$C = \left(\frac{1}{\sqrt{M}} A\right)\left(\frac{1}{\sqrt{M}} A\right)^t = H H^t,$$

siendo

$$H = \frac{1}{\sqrt{M}} A$$

una matriz de tamaño $10\,304 \times 200$ en nuestro caso.

- Los autovalores de C son números reales no negativos. Calculamos los $k = 8$, por ejemplo, autovalores de mayor tamaño de la matriz C y la base buscada estará formada por los autovectores asociados. En nuestro caso, la matriz C es de tamaño $10\,304 \times 10\,304$, y esto hace muy costoso calcular los autovalores y autovectores de C.

  La descomposición en valores singulares de H puede trabajar, en principio, con HH^t de tamaño $10\,304 \times 10\,304$ o con H^tH de tamaño 200×200, según convenga. En este caso lo hará con la matriz más pequeña, es decir, con H^tH.

  Resulta que los autovectores de $C = HH^t$ coinciden con los vectores singulares por la izquierda de H.

  Resumiendo, la base óptima de proyección se obtendrá con los autovectores asociados a los $k = 8$ autovalores más grandes de $C = HH^t$. Para calcularlos haremos la descomposición en valores singulares de H, esto es, $H = USV^t$, y nos quedaremos con las $k = 8$, por ejemplo, primeras columnas de la matriz U. En definitiva, la descomposición SVD hace más manejable el problema.

Turk et al. en [20] introdujeron un término nuevo: *autocara*. Vamos a ver a continuación a qué se refieren con esta palabra. Si nos quedamos con el primer vector de la base óptima anterior, que será un vector de 10 304 componentes, y lo volvemos a escribir como una matriz de tamaño 112×92, obtendremos la imagen de la izquierda de la figura 11, que se conoce como *autocara*. En dicha figura se muestran las 4 primeras autocaras. Como podemos observar en la figura 11, las autocaras tienen un aspecto como fantasmal. Las autocaras asociadas a los autovalores más grandes van a ser las que nos proporcionan la mejor base de proyección y las que nos van a permitir reducir la dimensión del problema.

Una vez conseguida esta base óptima de proyección, formaremos la matriz de proyección W escribiendo los vectores de la base por columnas. Después proyectaremos todas las caras del conjunto de entrenamiento en dicho subespacio:

$$Y_i = W^t E_i,$$

con $i = 1, 2, \ldots, M$.

Figura 11: Las cuatro primeras autocaras.

De esta forma cada imagen del conjunto de entrenamiento vendrá descrita por sus 8 coordenadas Y_i. Así queda patente la reducción de la dimensionalidad del problema: pasamos de 10 304 componentes a sólo 8.

Para clasificar una imagen test, lo que haremos será lo siguiente:

- Escribiremos la imagen test en formato unidimensional.
- Le restaremos la media Ψ, al igual que hicimos con las imágenes de entrenamiento.
- Proyectaremos el vector anterior en la base óptima, obteniendo las 8 componentes del vector proyectado que indicaremos por Z.
- A continuación averiguamos cuál es el vecino más cercano de Z buscando entre todas las proyecciones Y_i del conjunto de entrenamiento. A la imagen test le adjudicaremos la identidad del individuo más cercano.

En las pruebas realizadas con la base ORL, cogiendo 5 imágenes de cada individuo como imágenes de entrenamiento y las restantes 5 como imágenes test, hemos probado a aplicar el método anterior (cogiendo los 8 primeros autovectores de la matriz de covarianza) y el resultado obtenido en la fase test es de un 86 % de aciertos.

El método de Turk et al. de [20] ha tenido una gran repercusión. Este método se conoce como PCA o 1D-PCA. Se han publicado muchísimas variaciones del mismo como [21, 22]. Una generalización es la técnica propuesta en [22] que adapta el método anterior a dos dimensiones y se conoce como 2D-PCA. Este nuevo método consigue mejores tasas de aciertos que el original 1D-PCA, alrededor del 94 % en la base de datos ORL.

Referencias

[1] G. Álvarez, L. Hernández-Encinas y A. Martín del Rey, A multisecret sharing scheme for color images based on cellular automata, *Information Sciences* **178** (2008), 4382–4395.

[2] A. Basso, F. Bergadano, D. Cavagnino, V. Pomponiu y A. Vernone, A Novel Block-based Watermarking Scheme Using the SVD Transform, *Algorithms* **2** (2009), 46–75.

[3] G. R. Blakley, Safeguarding cryptographic keys, *AFIPS Conference Proceedings* **48** (1979), 313–317.

[4] B. Chandra Mohan y S. Srinivas Kumar, A Robust Image Watermarking Scheme using Singular Value Decomposition, *Journal of Multimedia* **3** (2008), 7–15.

[5] C. Fu, Z. C. Zhang, Y. Chen y X. Wang, An Improved Chaos-Based Image Encryption Scheme, *Lectures and Notes in Computer Sciences* (2007), 575–582.

[6] L. S. Hill, Cryptography in an algebraic alphabet, *The American Mathematical Monthly* **38** (1929), 135–154.

[7] I. A. Ismail, M. Amin y H. Diab, An efficient modified Hill Cipher adapted to image encryption, *ICGST-CNIR Journal* **5** (2006), 53–62.

[8] V. Jagannathan, A. Mahadevan, R. Hariharan y S. Srinivasan, Number Theory Based Image Compression Encryption and Application to Image Multiplexing, *International Conference on Signal Processing, Communications and Networking* (2007), 59–64.

[9] C. C. Lai, H. C Huang y C. C. Tsai, Image Watermarking Scheme Using Singular Value Decomposition and Micro-genetic Algorithm, *IIHMSP'08 International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2008), 469–472.

[10] J. J. Martínez Fernández de la Heras, La descomposición en valores singulares (SVD) y algunas de sus aplicaciones, *La Gaceta de la Real Sociedad Matemática Española* **8** (2005), 796–810.

[11] B. Noble y J. W. Daniel, *Álgebra Lineal Aplicada*, 3.ª ed., Prentice-Hall Hispanoamericana, 1989.

[12] T. J. Peters, R. Smolikova-Wachowiak y M. P. Wachowiak, Microarray Image Compression Using a Variation of Singular Value Decomposition, *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2007), 1176–1179.

[13] H. S. Prasantha, H. L. Shashidhara y K. N. Balasubramanya, Image Compression Using SVD, *Conference International on Computational Intelligence and Multimedia Applications* **3** (2007), 143–145.

[14] A. Ranadea, S. S. Mahabalarao y S. Kale, A variation on SVD based image compression, *Image and Vision Computing* **25** (2007), 771–777.

[15] A. Rojas y M. Torralbo, Aproximaciones de bajo rango de una matriz, *Suma* **30** (1999), 47–51.

[16] L. Seripeariu y M. D. Frunza, A new image encryption algorithm based on inversable functions defined on Galois fields, *International Symposium on Signals, Circuits and Systems* (2005), 243–246.

[17] A. Shamir, How to share a secret, *Communications of the ACM* **22** (1979), 357–390.

[18] C. C. Thien y J. C. Lin, Secret Image Sharing, *Computers and Graphics* **26** (2002), 765–770.

[19] C. F. Tsai y W. Y. Yang, Real-Time Color Image Watermarking Based on D-SVD Scheme, *Lecture Notes in Computer Science, Advances in Image and Video Technology* (2007), 289–297.

[20] M. A. Turk y A. P. Pentland, Face recognition using eigenfaces, *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), 586–591.

[21] C. Wang, B. Yin, X. Bai y Y. Sun, Color face recognition based on 2DPCA, *Proceedings 19th International Conference on Pattern Recognition* (2008), 1–4.

[22] J. Yang, D. Zhang, A. F. Frangi y J. Y. Yang, Two-dimensional PCA: a new approach to appearance-based face representation and recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** (2004), 131–137.

[23] R. Zhaoa, J. J. Zhaob, F. Daia y F. Q. Zhaoa, A new image secret sharing scheme to identify cheaters, *Computer Standards and Interfaces* **31** (2009), 252–257.

Ángela Rojas Matas, Dpto. de Matemáticas, Universidad de Córdoba
Correo electrónico: ma1romaa@uco.es

Alberto Cano Rojas, Alumno de Ingeniería Informática, Universidad de Córdoba
Correo electrónico: i52caroa@uco.es