Trabajo práctico 1

Compresión de Imágenes

1. Introducción

1.1. Representación de una imagen digital

Una imagen digital es esencialmente una matriz de números en 2D, donde cada elemento representa un píxel. El valor numérico del elemento cuantifica la *intensidad lumínica del píxel.* Por ejemplo, las imágenes más simples tienen píxeles con solo dos niveles, blanco o negro (1 o 0) como se ve en el siguiente ejemplo:

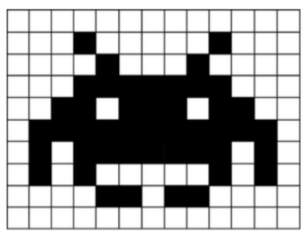

Figura 1: Imagen monocromática

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

Figura 2: Matriz de la imagen de la figura 1.

Para representar una mayor cantidad de niveles de brillo, los elementos pueden tomar una mayor cantidad de niveles de intensidad (por ejemplo 256 si se codifica cada píxel con 8 bits) como se ve en la siguiente imagen:

Figura 3: Imagen en escala de grises.

$$
\begin{bmatrix}
128 & 128 & 160 & 128 & 128 & 96 & 128 & 128 & 128 & 128 \\
128 & 160 & 128 & 64 & 32 & 0 & 0 & 192 & 160 & 128 \\
160 & 160 & 32 & 32 & 0 & 0 & 32 & 64 & 128 & 160 \\
192 & 64 & 128 & 255 & 255 & 255 & 192 & 96 & 64 & 192 \\
160 & 96 & 192 & 255 & 192 & 192 & 255 & 96 & 64 & 192 \\
160 & 64 & 160 & 160 & 192 & 192 & 128 & 64 & 0 & 160 \\
160 & 64 & 160 & 0 & 160 & 128 & 0 & 32 & 0 & 128 \\
192 & 160 & 128 & 255 & 192 & 128 & 160 & 64 & 64 & 255 \\
160 & 192 & 96 & 255 & 160 & 64 & 128 & 32 & 192 & 255 \\
160 & 192 & 128 & 192 & 192 & 96 & 96 & 64 & 255 & 192 \\
160 & 192 & 160 & 64 & 128 & 64 & 32 & 96 & 255 & 192 \\
160 & 192 & 255 & 0 & 0 & 0 & 96 & 128 & 255 & 25
\end{bmatrix}
$$

Figura 4: Matriz de la imagen de la figura 3.

1.2. Compresión de imágenes

La mayoría de las imágenes no son colecciones de transiciones de intensidad arbitrarias. Cada imagen que vemos contiene algún tipo de estructura. Esto implica que en general existe cierta correlación entre los píxeles vecinos. El objetivo de comprimir una imagen es almacenarla de una forma más compacta, es decir, que se requiera menos bits para su codificación, y que a su vez no se afecte apreciablemente su calidad. Esto es posible si se tiene en cuenta que una imagen en crudo, en su formato "sin procesar", contiene un alto grado de información redundante. De este modo, las técnicas de compresión buscan eliminar esas redundancias de modo tal que se mantenga la mayor parte de la información en una menor cantidad de bits. Existen muchas técnicas para cumplir con este objetivo. En nuestro caso, el trabajo se enfocará en una técnica basada en la transformada de Karhunen-Loève (KLT) que utiliza el concepto de análisis de componentes principales (PCA) [1] [2]. Las ventajas del método PCA es que tiene un resultado óptimo en cuanto a la separación de las redundancias, pero con la desventaja de un costo computacional mucho mayor que otras técnicas usadas normalmente (como la DCT en el formato JPEG).

2. Método de compresión KLT

El método de compresión basado en KLT permite hallar una transformación reversible que elimine la redundancia descorrelacionando los datos y almacenándolos de forma más eficiente, de manera tal de eliminar los elementos de menor varianza, que serán los píxeles vecinos que presenten cambios menos perceptibles.

2.1. Matriz de covarianza de la imagen

Para aplicar la compresión con este método, primero es necesario estimar la matriz de covarianza de la imagen, donde el vector aleatorio asociado represente una vecindad de cierta cantidad de píxeles. Suponiendo que una imagen se segmenta bloques más chicos de $N \times N$, típicamente de 8×8, como se muestra en la figura 5 (a). Cada bloque representa un conjunto de píxeles vecinos que permiten definir un vector aleatorio $\mathbf{x}$ desenrollando el bloque, como se muestra en la figura 5 (b). Con esta definición, podemos suponer que todos los bloques que conforman una imagen son realizaciones particulares $\mathbf{x}_i$ del vector aleatorio $\mathbf{x}$. De esta forma, la covarianza $C_{\mathbf{x}} = E\left[(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^T\right]$ da una idea de **cuánta variabilidad tiene la imagen en relación a píxeles vecinos para un área del tamaño de un bloque**. Por lo tanto, se podrán estimar la media $\widehat{\boldsymbol{\mu}}_{\mathbf{x}}$ y la matriz $\widehat{C}_{\mathbf{x}}$ de acuerdo a las ecuaciones 1 y 2 respectivamente (donde L sería la cantidad de bloques totales en la que se segmenta la imagen).

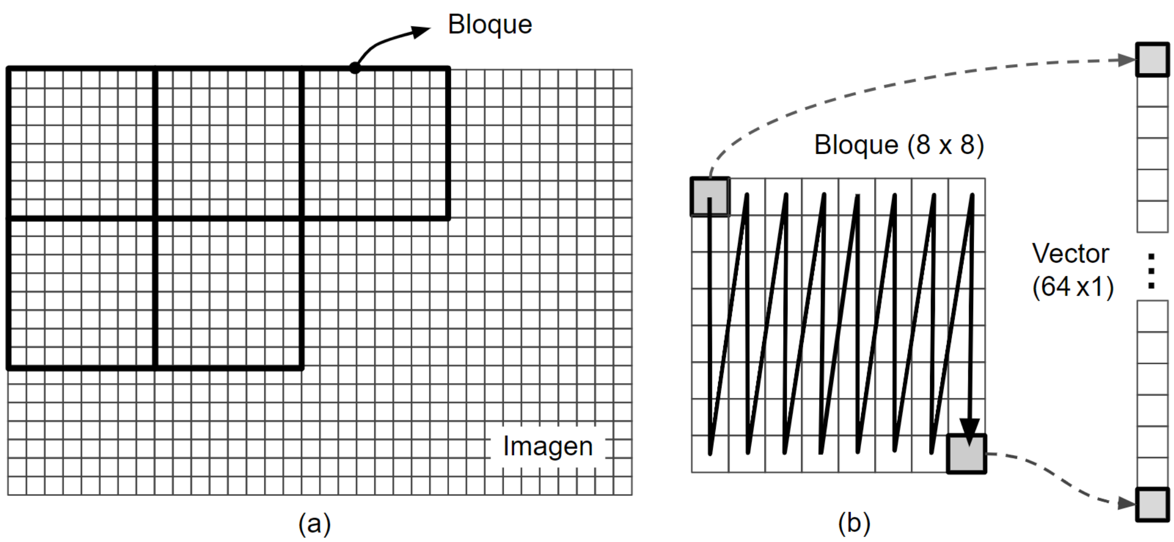

Figura 5: (a) Segmentación de la matriz de una imagen en bloques (submatrices) de 8×8. (b) Armado de una realización $\mathbf{x}_i$ del vector aleatorio $\mathbf{x}$.

$$\widehat{\boldsymbol{\mu}}_{\mathbf{x}} = \frac{1}{L}\sum_{i=0}^{L-1} \mathbf{x}_i \tag{1}$$

$$\widehat{C}_{\mathbf{x}} = \frac{1}{L}\sum_{i=0}^{L-1} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{x}})(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{x}})^T \tag{2}$$

2.2. Compresión mediante PCA

El método PCA consiste proyectar el vector aleatorio $\mathbf{x}$ en un espacio de menor dimensión conformado por una parte de las bases ortogonales extraídas de la propia matriz de covarianza. De este modo se busca conservar la mayor parte de la información de la imagen pero con un tamaño de datos mucho menor. Las bases utilizadas son los autovectores de la matriz de covarianza $C_{\mathbf{x}}$, cuya diagonalización se expresa en la ecuación 3, donde D es una matriz diagonal de autovalores y V la matriz de autovectores asociados. Se puede demostrar que si se proyecta el vector $\mathbf{x}$ en el espacio de autovectores, se obtiene un nuevo vector $\mathbf{y} = V^T\mathbf{x}$, cuya matriz de covarianza es $C_{\mathbf{y}} = D$. Esto implica que las componentes del vector $\mathbf{x}$ están descorrelacionadas, ya que su covarianza es diagonal, siendo las varianzas de cada componente $\sigma^2_{yi} = \lambda_i$. De este modo, descartando aquellas componentes con menor varianza (las que aportan menos información) puede reducirse la cantidad de datos necesarios para conservar la mayor parte de la información de la imagen. Si se define una matriz U a partir de los K autovectores principales, asociados a los autovalores de mayor peso, la proyección $\hat{\mathbf{y}} = U^T\mathbf{x}$ permite conservar casi la misma infromación pero con un tamaño menor al que requiere $\mathbf{x}$ (dado que $\dim(\mathbf{y}) = K < 64$ y $\dim(\mathbf{x}) = 64$). Luego, el proceso puede revertirse para recuperar el vector original (asumiendo cierto error al ser una compresión con pérdidas) aplicando la transformación inversa $\hat{\mathbf{x}} = U\hat{\mathbf{y}}$. En caso que el vector tenga una media no nula, debe restarse la media antes de todo el proceso y volver a sumarla en la reconstrucción.

De esta forma, para almacenar la imagen, es decir todas las realizaciones $\mathbf{x}_i$, podemos almacenar las proyecciones $\hat{\mathbf{x}}_i$ utilizando un menor espacio en memoria y posteriormente reconstruirla con la operación inversa.

$$C_{\mathbf{x}} = VDV^T \tag{3}$$

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & ... & 0 \\ 0 & \lambda_2 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & ... & \vdots \\ 0 & 0 & 0 & ... & \lambda_{64} \end{bmatrix} \tag{4}$$

En resumen, el procedimiento para la compresión y descompresión de imágenes con esta técnica, sigue los siguientes pasos:

Compresión

1. Segmentar la imagen en bloques de 8×8 y convertirlos en vectores $\mathbf{x}_i$ de 64×1.
2. Estimar la media $\boldsymbol{\mu}_{\mathbf{x}}$ del vector aleatorio de acuerdo a la ecuación 1.
3. Estimar la matriz de covarianza $C_{\mathbf{x}}$ de la imagen de acuerdo a la ecuación 2.
4. Diagonalizar la matriz de covarianza $C_{\mathbf{x}}$ para obtener las matrices V y D.
5. Definir una matriz U que contenga únicamente las K autovectores (columnas) de V asociados a los autovalores principales.
6. Obtener $\hat{\mathbf{y}}_i = U^T\mathbf{x}_i$ proyectando todos los vectores $\mathbf{x}_i$ en el espacio de bases de U.
7. Guardar: la colección de vectores $\hat{\mathbf{y}}_i$, el vector de media estimada $\hat{\boldsymbol{\mu}}_{\mathbf{x}}$ y la matriz U.

Descompresión

1. Abrir la colección de vectores $\hat{\mathbf{y}}_i$, el vector de media estimada $\hat{\boldsymbol{\mu}}_{\mathbf{x}}$ y la matriz U.
2. Volver a transformar cada vector reducido al espacio original aplicando la transformación inversa agregando la media (que fue restada en la compresión) $\hat{\mathbf{y}}_i = U\hat{\mathbf{x}}_i + \hat{\boldsymbol{\mu}}_{\mathbf{x}}$.
3. Reconstruir la imagen volviendo a convertir los vectores $\hat{\mathbf{x}}_i$ a bloques de 8×8, combinando todos los bloques para formar nuevamente la matriz de una imagen.

Métricas Algunas métricas que se pueden utilizar para medir el desempeño de la compresión son el error cuadrático medio (MSE), ecuación 5, donde p_{ij} y $\hat{p}_{ij}$ rerpesentan el valor del píxel en la posición ij original y reconstruido, N_w y N_h la cantidad de píxeles de ancho y de alto respectivamente. Por otro lado, también es útil la tasa de compresión (CR) definida como la relación entre la cantidad de datos almacenados en relación a la cantidad de datos de la imagen completa, ecuación 6.

$$MSE = \frac{1}{N_w\, N_h} \sum_{i=0}^{N_w-1} \sum_{j=0}^{N_h-1} (p_{ij} - \hat{p}_{ij})^2 \tag{5}$$

$$CR = \frac{\text{datos comprimidos}}{\text{datos totales}} \times 100\,\% \tag{6}$$

3. Ejercicios

Ejercicio 1: Correlación

El objetivo de este ejercicio es descomponer la matiz de imagen en bloques de solo dos píxeles con el propósito de poder ver gráficamente la correlación entre píxeles vecinos.

(a) Cargar en Matlab una imagen (convertirla a escala de grises y a tipo `double`). Formar bloques de 2×1 para definir el vector $\mathbf{x} = [x_0\ x_1]^T$ (con x_0 y x_1 dos píxeles contiguos).

(b) Hacer un gráfico de dispersión para ver gráficamente cuánta correlación existe entre los dos píxeles vecinos para cada imagen. Probar con las imágenes `img_01.jpg` y `img_02.jpg`.

(c) Calcular coeficiente de correlación de cada imagen (se puede utilizar la función `corrcoef()`).

Ejercicio 2: Compresión

Para realizar el proceso de compresión será necesario segmentar la imagen en bloques de tamaño estándar para luego aplicar la reducción de dimensionalidad mediante PCA.

(a) Desarmar la imagen `img_03.jpg` en bloques de 8×8 y obtener los vectores $\mathbf{x}_i$. Estimar la matriz de covarianza $C_{\mathbf{x}}$ y la media $\widehat{\boldsymbol{\mu}}_{\mathbf{x}}$.

(b) Aplicar el método de KLT: diagonalizar la matriz $\widehat{C}_{\mathbf{x}}$. Asumiendo $CR = 20\,\%$ quedarse con los autovectores necesarios para definir U y cumplir con esa tasa de compresión. Obtener los vectores proyectados $\widehat{\mathbf{y}}_i$ en el espacio de la compresión.

(c) Calcular la cantidad de elementos almacenados (incluyendo matriz de transformación, la media, y los vectores reducidos). Comparar el tamaño reducido con el original (en términos de cantidad de elementos almacenados).

Ejercicio 3: Descompresión

(a) Implementar el proceso inverso para decodificar la imagen comprimida. Regenerar la imagen a reconstruir mediante la transformación y el vector de medias.

(b) Graficar con `imshow()` la imagen y compararla con la original (*Ayuda: convertir a 8 bit para graficar la imagen*).

Ejercicio 4: Error cuadrático medio

(a) Abrir la imagen `img_04.jpg`, calcular y graficar MSE en función de $CR[\%] = \{5 : 5 : 95\}$.

(b) Graficar la imagen original y todas las versiones comprimidas para los casos $CR[\%] = \{5,\ 10,\ 15,\ 20,\ 25\}$.

4. Conclusiones

Como conclusiones, elabore un resumen breve y conciso comentando características que considere relevantes del método propuesto en este trabajo y los resultados obtenidos, así como dificultades encontradas y cómo fueron abordadas.

5. Herramientas de utilidad

A continuación se describe una lista de las funciones proporcionadas por las herramientas de software utilizadas en la materia (Maltlab u Octave) que resultarán útiles para la realización de este trabajo práctico. Ante la duda se recomienda explorar el `help`.

- `image_rgb = imread('img_01.bmp') % cargar imagen. Retorna una matriz RGB`
- `image_gray = rgb2gray(image_rgb) % convierte imagen RGB a grises`
- `x = double(n) % Convierte a punto flotante`
- `n = uint8(x) % Convierte a entero de 8 bits`
- `imshow(A) % Muestra la imagen de la matriz A (A debe ser uint8)`
- `A = reshape(v, [3 5]) % Convierte v (de 1x3*5) en una matriz de 3x5`
- `v = A(:) % Convierte la matriz A en un vector`
- `B = A(1:5, 1:8) % indexar la matriz A para elegir submatriz entre 1:5 y 1:8`

6. Normas y material entregable

- **Informe**: debe ser conciso y mostrar los resultados solicitados con los comentarios y suposiciones hechas para cada ejercicio. El informe debe entregarse en formato PDF (**no se aceptarán otros formatos**) y con nombre: `TP1_GXX.PDF` (donde `XX` es el número de grupo). No incluir desarrollos teóricos adicionales que no se pidan explícitamente en el enunciado. Tampoco agregar líneas de código en el mismo PDF.

- **Código**: Los archivos de código utilizados deben ser en formato `.m` de Matlab/Octave (o alternativamente `.py` si usara lenguaje Python). El código debe incluirse junto al informe en un archivo ZIP (con mismo nombre que el informe) que deberá subirse al campus.
- Los conceptos y desarrollos involucrados en el TP podrán ser evaluados en cualquiera de las instancias de examen que establezca el docente.

Referencias

[1] Steven M. KAy. Intuitive probability random processes using MATLAB. Springer 1951.

[2] Ed. K. R. Rao and P.C. Yip. Boca Raton. The Transform and Data Compression Handbook, CRC Press LLC, 2001.

[3] CCITT. Information technology - Digital compression and coding of continuous-tone still images - requirements and guidelines. The international telegraph and telephone consultative committee. T.81, 1992.