Profesor: Luis Felipe Giraldo
Autores:
Isabella Torres
Andrés Mugnier
Alejandro Arias

Universidad de los Andes
Facultad de Ingeniería

Análisis Inteligente de Señales y Sistemas: Caso de Estudio 1 Solución

Punto 1

- **Enunciado:** Considere la imagen en escala de grises `paisaje.tiff`. Para obtener puntos en este ejercicio tiene que resolver todos los enunciados en este problema.

  a) Del libro Digital Image Processing, de Gonzalez, lea y entienda el procedimiento para realizar la equalización de una imagen en escala de grises. Cualquier edición del libro le sirve. En este informe, explique brevemente cada paso para ecualizar el histograma de una imagen en escala de grises.

  Antes de explicar el procedimiento para realizar una equalización es necesario entender qué es la equalización. Este procedimiento se realiza cuando la intensidad de los pixeles de una imágen se concentra e un rango o unos rangos específicos, lo que genera que no sea una imágen muy nítida. Lo que hace la equalización es redistribuir estas intensidaddes, ampliando el rango de intensidades y mejorando su nitidez ya que hace que los colores sean más distinguibles.

Figure 1: Equalización de una imagen

Como se observa en la figura 1, se ve cómo se reordenan las intensidades de forma más uniforme, y la función de distribución acumulada tiene una pendiente menor y mejor distribuida. Para implementar matemáticamente el filtro se parte de una función:

$$p(r_k) = \frac{n_k}{MN}$$

Donde n_k es el número de pixeles de la imagen que tienen una intensidad r_k y N y M son las dimensiones de la imágen, por lo que $p(r_k)$ se podría interpretar como la probabilidad de que esa intensidad se presente en la imagen. Adicionalmente, se considera esta función dentro de un rango de $[0, L-1]$, donde L es la intensidad más alta presente en la imagen. Tomando en cuenta lo anterior, se busca obtener una transformación $T(r_k)$ en la que a cada uno de los pixeles se le asigne un nuevo valor de intensidad s_k. Matemáticamente, se puede expresar como:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

Finalmente, el valor de s_k obtenido, se aproxima al entero más cercano, es decir, a la intensidad con el valor que mejor represente el resultado.

b) Equalice la imagen utilizando Matlab o Python. Puede utilizar la función de Matlab histeq o su equivalente en Python. Grafique la imagen original y la imagen resultante con sus respectivos histogramas y funciones de distribución acumulada, y analice los resultados del procedimiento de equalización. Asegúrese que la imagen leida esté en escala de grises. NO puede utilizar las funciones de Matlab o Python que hacen estos procedimientos para calcular las funciones de distribución acumulada e histograma. Es decir, tiene que hacer la rutina que cuenta la cantidad de píxeles por cada valor de nivel de gris para el histograma y la suma para la distribución acumulada.

La imagen no equalizada se presenta a continuación.

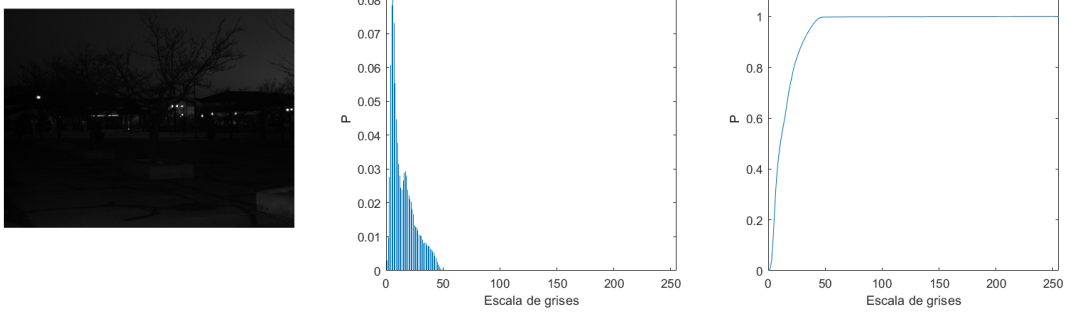


Figure 2: Imagen del paisaje no equalizada

Luego de la equalización, por medio de la función de MATLAB histeq, se obtiene la siguiente imagen, en la que se puede observar una mejora notoria en la nitidez, y también un histograma y una función de densidad de probabilidad mucho más uniforme.

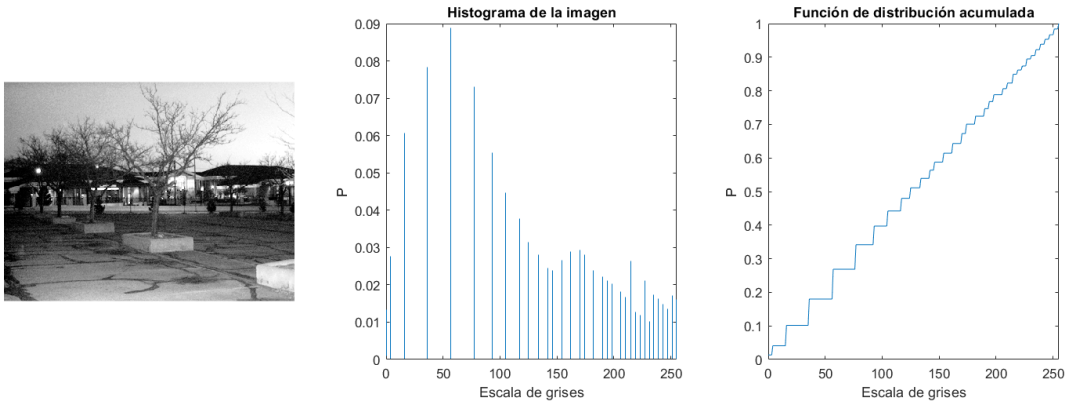


Figure 3: Imagen del paisaje equalizada

Punto 2

- **Enunciado:** El archivo `comprimida.txt` contiene los primeros 50x100 coeficientes de la transformada del coseno de una imagen en escala de grises de tamaño 630x945. Para obtener puntos en este ejercicio tiene que resolver todos los enunciados en este problema.

  a) Utilizando alguno de los comandos `imagesc` o `bar3` visualice los coeficientes de la transformada.

  . ***Solución*** **:**
  Utilizando la función `imagesc` de Matlab se visualizaron los coeficientes de la transformada del coseno discreta. Los resultados se muestran en la siguiente imagen:

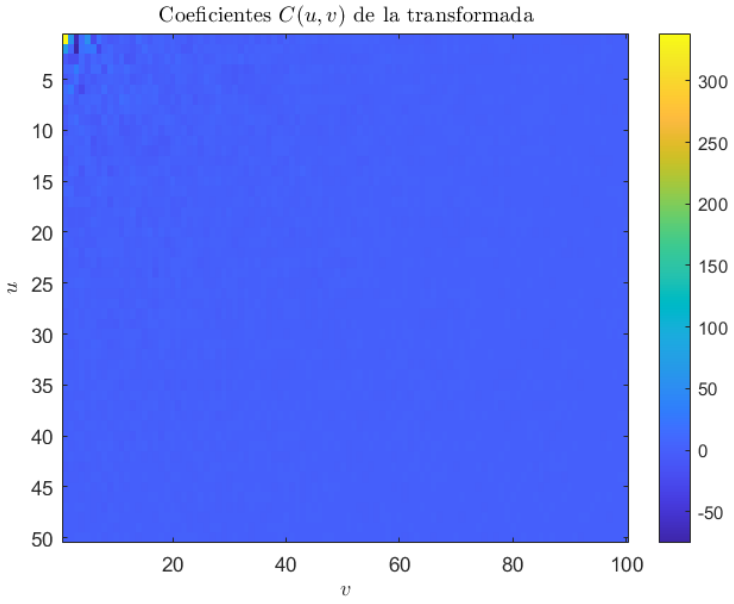


Figure 4: Coeficientes de la transformada

Como se puede ver en la figura 7 los coeficientes con valor más alto "Con más peso" están muy cercanos al origen, mientras que la gran mayoria están en el rango $[0,100]$.Es decir, las frecuencias bajas predominan en los coeficientes $C(u,v)$

b) Escriba una rutina en Matlab o Python para reconstruir la imagen original a partir de estos coeficientes, y muestre la imagen resultante (por ejemplo, puede utilizar la función `imshow` de Matlab). Lo más probable es que esta rutina tarde varios minutos en correr. Trate de identificar lo que contiene la imagen. NO puede utilizar funciones preestablecidas de Matlab o Python para hacer esta reconstrucción. Aquí tiene que ser una implementación clara de las ecuaciones de reconstrucción de la imagen a través de transformada del coseno.

. ***Solución*:**
Luego se implementó una rutina para reconstruir la señal original en la escala de grises, es decir se implementó la DCT inversa. Para recuperar la imagen original se implementó la siguiente ecuación:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha_u \alpha_v C(u,v) cos(\frac{\pi u(2x+1)}{2M^*}) cos(\frac{\pi v(2y+1)}{2N^*}) \tag{1}$$

Donde:

$$\alpha_u = \begin{cases} \frac{1}{\sqrt{M^*}} & \text{si u=0} \\ \sqrt{\frac{2}{M^*}} & \text{otro caso} \end{cases} \tag{2}$$

y,

$$\alpha_v = \begin{cases} \frac{1}{\sqrt{N^*}} & \text{si v=0} \\ \sqrt{\frac{2}{N^*}} & \text{otro caso} \end{cases} \tag{3}$$

Para este caso, se quería reconstruir una imagen de tamaño $630x945$. Por lo que tendríamos $x = 0,...,629 = M^* - 1$ y $y = 0,...,944 = N^* - 1$ y $M = 50$ y $N = 100$. Es decir, queremos reconstruir la imagen original con menos información.
La implementación de esta ecuación se encuentra en el código anexo.
Luego de obtener los coeficientes en escala de grises, es decir la matriz con entradas $f(x,y)$, se utilizó la función `imshow()` de Matlab para visualizar el resultado. La imagen resultante se muestra en la siguiente figura:

Figure 5: Reconstrucción de la imagen

Como se puede ver en la figura 5, la imagen corresponde a la estatua de San Alberto Magno de la universidad de los Andes, mejor conocido como *El bobo*.
Al ser una reconstrucción con menos coeficientes que píxeles en la imagen original, se pierde

información al reconstruirla, es por esto que la imagen se ve con muy baja calidad.
A su vez, como en la figura 7 se observa que los coeficientes predominantes corresponden a frecuencias más bajas, es por esto que en la reconstrucción de la imagen no se ven tan bien definidos los bordes que representan altas frecuencias en terminos de escala de grises.

Punto 3

- **Enunciado:** Considere la imagen en escala de grises ladron.jpg. Para obtener puntos en este ejercicio tiene que resolver todos los enunciados en este problema.

a) Implemente en Maltab o Python una rutina que realice la convolución en imágenes vista en clase dado un filtro o kernel de tamaño nŒn. NO puede utilizar funciones preestablecidas de Matlab o Python para filtrar imágenes.

Para realizar el filtro, se implemetó un filtro por medio de una convolución, en el que el valor de un pixel filtrado se obtiene mediante un promedio de ese pixel y los pixeles cercanos. En este caso, el kernel del filtro es de 3, lo que indica que el filtro funciona como una ventana que toma una matriz de 3X3. Este filtro se define como:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

En el que w es el filtro implementado, f es la imagen que se desea filtrar y b y a son los limites inferiores y superiores del filtro, al rededor del pixel x, y.

b) Pruebe su rutina con la imagen ladron.jpg y dos filtros de tamaño 3 Œ 3 cada uno que realicen las siguientes operaciones: borroseado (blurring) y detección de bordes (edge detección). Muestre los filtros elegidos y las imágenes resultantes después del filtrado.

Después de implementar esta función que recibe por parámetro una matriz de un filtro con sus coeficientes, se puede pasar a especificar el filtro deseado para dos escenarios. El primero es un filtro que vuelve la imagen mas borrosa, por lo que se toma un promedio ponderado de los pixeles cercanos, en la que cada valor de la matriz equivale a $\frac{1}{n^2}$, y ya que $n = 3$:

$$W_b = \begin{bmatrix} 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \end{bmatrix}$$

Por otro lado, para implementar el filtro de detección de bordes se uso la matriz:

$$W_{bd} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 24 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

Con el filtro que hace la imagen mas borrosa, se obtuvo la imagen:

Figure 6: Imagen original vs. filtro borroso

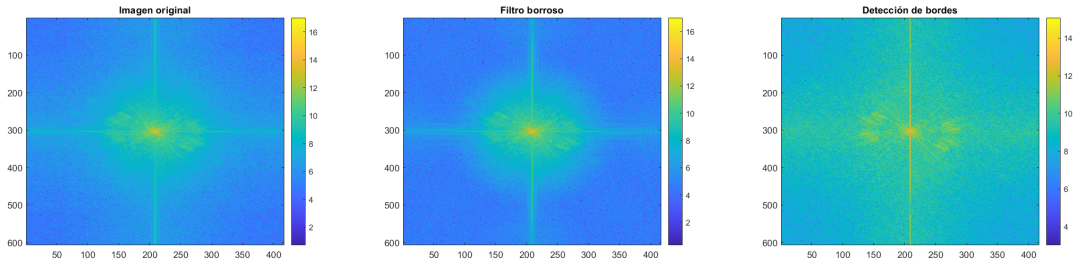
Y para el filtro de deteccion de bordes se obtuvo la siguiente figura, en la que se logran identificar los bordes o grandes contrastes en la imagen.

Figure 7: Imagen original vs. filtro detector de bordes

c) Utilice las funciones fft2, fftshift, y imagesc de Matlab, o sus equivalentes en Python, para graficar la imagen de la magnitud de la transformada de Fourier de las tres imágenes: original, borroseada, y con bordes detectados. Asegúrese de que la frecuencia 0 de la transformada esté en el centro de la imagen. Analice los resultados teniendo en cuenta el impacto que hace cada filtro sobre la imagen desde una perspectiva frecuencial (por ejemplo, si atenúa o amplifica frecuencias altas o bajas o

bandas de frecuencia).

Por medio del código implementado, se obtuvo la gráfica de las mmagnitudes en frecuencia de la imágen, tal como se presenta a continuación.



(a) Original (b) Borroso (c) Bordes

Figure 8: The same cup of coffee. Two times.

Antes de analizar la imágn en frecuencia, cabe aclarar que cuando en una imágen hay grandes contrastes en las intensidades de la imágen, se requieren magnitudes de frecuencia más grandes para su representación. Por lo anterior, vemos que cuando a la imágen original se le aplica un filtro borroso y se difuminan las intensidades, se requieren de menores magnitudes de frecuencia, por lo que la gráfica toma valores de azul más oscuros. Por otro lado, cuando se aplica el filtro de detección de bordes, aumentan drásticamente los contrastes en la imagen, por lo que se requieren más frecuencias de mayor magnitud para su representación.

Punto 4

- **Enunciado:** Considere las imagenes en escala de grises `ladron.jpg` y `ladronborroso.jpg`. Resulta que la segunda imagen es el resultado de aplicar un filtrado lineal (a través de la convolución) utilizando un kernel (es decir, un filtro) de 3x3 desconocido. Lo único que se conoce de este kernel es que los elementos de la segunda fila son cero. Para obtener puntos en este ejercicios tiene que resolver todos los enunciados en este problema.

(a) ladron.jpg (b) ladronborroso.jpg

Figure 9: Imágenes ladrón

a) Intente explicar intuitivamente qué tipo de filtro se utilizó, y el efecto que está generando.

   Se utilizó un filtro cuyo kernel posee unas características que pueden intuirse teniendo en cuenta el resultado de la imagen: primero, es posible notar una imagen en la cual el color negro es predominante, por lo que sus componentes sumados no deben dar 1 como resultado (si fuera así, el color predominante sería similar al gris de la imagen original 9a), por lo tanto deben sumar un valor menor que 1; segundo, como se observa en las líneas diagonales sufren cambios distintos según su orientación (en el caso del sentido /estas no desaparecen, en el contrario sí se pierde claridad); tercero, las líneas verticales (como por ejemplo, en la orientación del cuchillo) tienden a desaparecer, por lo que es probable que los componentes horizontales del kernel sean cercanos a 0.

b) Formule matemáticamente, implemente, y resuelva un problema de optimización que encuentre los coeficientes del kernel de tal manera que el resultado de filtrar la imagen `ladron.jpg` con ese filtro corresponde a la imagen `ladronborroso.jpg`. Vea la ayuda abajo para recomendaciones de implementación.

   Sea U una matriz asociada a la imagen `ladron.jpg`, V una matriz asociada a la imagen `ladronborroso.jpg`, y sea k un kernel de 3×3. Denotamos la operación de filtrado entre la imagen U y el kernel k como $K\{U\}$. El problema de optimización se podría formular como encontrar los parámetros de h tal que la función $\|V - K\{U\}\|^2$ es minimizada, sujeto a que los elementos de la segunda fila de k son cero. Para resolver este problema, se recomienda utilizar alguna

de las funciones de Matlab `fminsearc`, `fminunc`, o `fmincon` o su equivalente en Python. Debido al tiempo de cómputo que este procedimiento de optimización puede tomar, también se recomienda utilizar la función `imfilter` en Matlab (o su equivalente en Python), la cual recibe una imagen y un kernel, y realiza el filtrado de forma eficiente. Vean la rutina ejemploFiltrarImagen.m como ejemplo a seguir. (este planteamiento corresponde al sugerido en la ayuda del enunciado).

Formalmente, se plantea así

$$\min_{h} \qquad f(h) = \|V - K\{U\}\|_F^2 \tag{4a}$$

$$Donde \qquad h = [h_1, h_2, h_3, h_4, h_5, h_6]^T. \tag{4b}$$

Teniendo en cuenta que el kernel final será de la forma:

$$k = \begin{bmatrix} h_1 & h_2 & h_3 \\ 0 & 0 & 0 \\ h_4 & h_5 & h_6 \end{bmatrix} \tag{5}$$

Y la operación $K\{U\}$ corresponde a la convolución en 2D entre la imagen U y el kernel k. Y la función objetivo obtiene la norma Frobenius de la diferencia entre el resultado de esta convolución y la imagen borrosa V.

c) Muestre el kernel que resulta del proceso de optimización y analice el porqué este kernel produce ese efecto de filtrado.

Al tener una condición inicial en 0, se obtiene un filtro con un kernel como se muestra en la ecuación 6:

$$\begin{bmatrix} 0.6989 & 0.3081 & 0.5803 \\ 0 & 0 & 0 \\ -0.7163 & -0.2344 & -0.5772 \end{bmatrix} \tag{6}$$

Teniendo en cuenta este resultado, es posible observar que los componentes sumados dan un valor de 0.0594, es decir, un valor bastante menor que 1, confirmando una de las hipótesis planteadas en el literal a).

Además, se produce un efecto de oscurecimiento/desaparición de las componentes de la imagen e sentido vertical. Esto debido a que el kernel posee componentes positivos en le primera fila, 0 en la segunda, y negativos en la tercera. Por lo que si se aplica el filtro sobre una zona cuyos componentes son uniformes, estos desaparecerán (tendrán valores cercanos a 0).

d) Grafique la imagen que resulta de filtrar `ladron.jpg` con el kernel estimado, compárela con `ladronborroso.jpg`. £Existen diferencias?

En la figura 10 se pueden observar las gráficas correspondientes a: la imagen `ladron.jpg`, `ladronborroso.jpg` y el resultado de filtrar la imagen original usando el kernel mostrado en la ecuación 6.

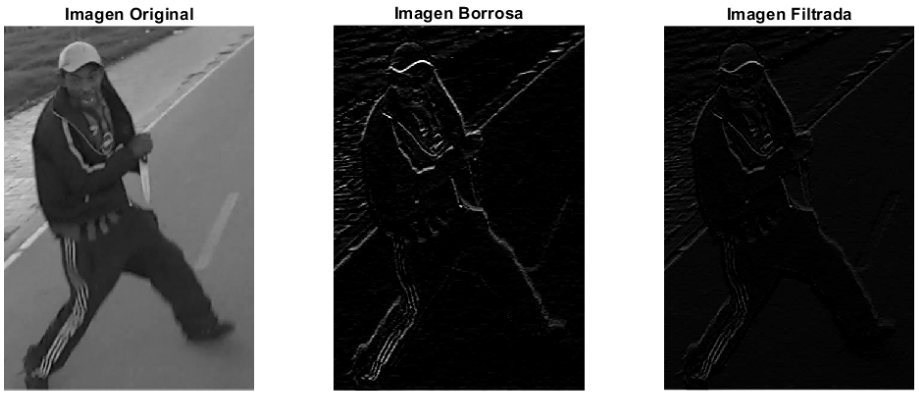


Figure 10: Resultados

Se pueden notar ciertas diferencias, principalmente debido a que el problema de optimización posee un gran número de minimizadores locales. Esto se puede comprobar al utilizar un punto de inicio aleatorio, observando que se converge a un punto distinto en cada caso.