Isabella Salgado *201730418*
Johan Hernández *201729696*
Juan Pablo Naranjo *201730006*

Tarea 4

1. Punto 1

   a) A continuación se presenta brevemente el proceso que se debe llevar a cabo para realizar el proceso de ecualización de una imagen, de acuerdo a [1].
   En primera instancia, se define el *histograma no normalizado* de una imagen digital $f(x, y)$ de L bits como una función

   $$h(r_k) = n_k \tag{1}$$

   donde $k \in [0, L-1]$ y n_k es el número de píxeles de f que tienen una intensidad r_k.
   También se presenta la definición del *histograma normalizado* como

   $$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN} \tag{2}$$

   en donde M y N corresponden al número de filas y columnas de la imagen f (si se viera como una matriz). Por convención, cuando se usan los términos "histograma" o "histogramas de imágenes", usualmente se refiere al histograma normalizado, más que al histograma no normalizado. Es importante mencionar que el histograma está definido por $p(r_k)$, que se puede ver como la función de densidad de probabilidad para los valores de intensidad r_k, lo que quiere decir que la suma de $p(r_k)$ para todo k es igual 1.

   De manera general, este tipo de histogramas representan la cantidad de veces que se repite cierta intensidad definida por r_k en una imagen de MxN. Para una escala de intensidad definida entre 0 y $L-1$, usualmente se toma el 0 como el color negro y $L-1$ como el color blanco. Un histograma $p(r_k)$ que tenga un valor muy elevado para valores de r_k muy cercanos a $L-1$ usualmente implica que hay más píxeles claros que oscuros en la imagen. Similarmente, para valores de r_k muy cercados a 0, se puede inferir que hay más píxeles oscuros que claros en la imagen.

   La ecualización de un histograma es una transformación lineal que se le hace a una imagen digital de entrada para obtener a la salida otra imagen digital ecualizada. Esta transformación lineal se puede definir como

   $$s = T(r) \tag{3}$$

   con $0 \leq r \leq L-1$, en donde la transformación lineal representa un mapeo de intensidades.

   De manera general, la transformación $T(r)$ debe cumplir con dos propiedades importantes para garantizar que las intensidades de salida nunca sean menores que sus intensidades de entrada correspondientes y que el rango de las intensidades de salida sea el mismo que las intensidades de entrada. Estas dos propiedades son que
   1) $T(r)$ sea una función monótona en el intervalo $0 \leq r \leq L-1$.
   2) $0 \leq T(r) \leq L-1$ para $0 \leq r \leq L-1$

   Una discusión más profunda sobre la definición de una función monótona y estrictamente monótona se puede encontrar en las páginas 191-194 de [1].

   Sabiendo que se cumplen las anteriores dos condiciones para la transformación lineal $T(r)$, se procede a definir el proceso de *ecualización de histograma* como aplicarle la transformación $s_k = T(r_k)$ de la ecuación 4 al histograma de una imagen digital f.

   $$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \tag{4}$$

con $k = 0 : L - 1$

De manera general, se obtiene una salida procesada que mapea cada píxel en la imagen de entrada con una intensidad r_k a otro píxel con una intensidad s_k en la imagen de salida. Esto quiere decir que el efecto que tiene la ecualización de un histograma, descrita completamente por la ecuación 4, es el de esparcir el histograma de una imagen de entrada para que los niveles de intensidad de la imagen ecualizada (salida) cubran un rango más amplio de la escala de intensidad definida por los límites 0 y $L - 1$. En términos técnicos, esto se llama mejorar el contraste de la imagen.

b) Para confirmar la información presentada en el literal anterior, se procedió a ecualizar la imagen mostrada en la figura 1 y mostrar sus histogramas correspondientes.

Figura 1: Imagen original (entrada)

Para hacer esto, se utilizó la función `histeq` de MATLAB, que tomó la imagen de la figura 1 y le realizó el proceso de ecualización. Una comparación entre la imagen original y la imagen ecualizada se puede observar en la figura 2.

Figura 2: Comparación entre imagen original e imagen ecualizada

Como se puede notar, el resultado fue el esperado: se mejoró el contraste de la imagen, permitiendo ver más detalladamente los objetos mostrados dentro de la misma.

En la figura 3 se puede observar el histograma de la imagen original (figura 1), junto con su función de distribución acumulada. Es importante mencionar que todos los histogramas mostrados en este punto corresponden a la definición presentada en el literal *a)* de histograma, es decir, son histogramas normalizados, en donde el eje y del histograma representa $p(r_k)$, o la función de densidad de probabilidad de los valores de intensidad de la imagen, que en este caso fue descrita usando la escala RGB, lo que significa que en este caso, $L = 256$.

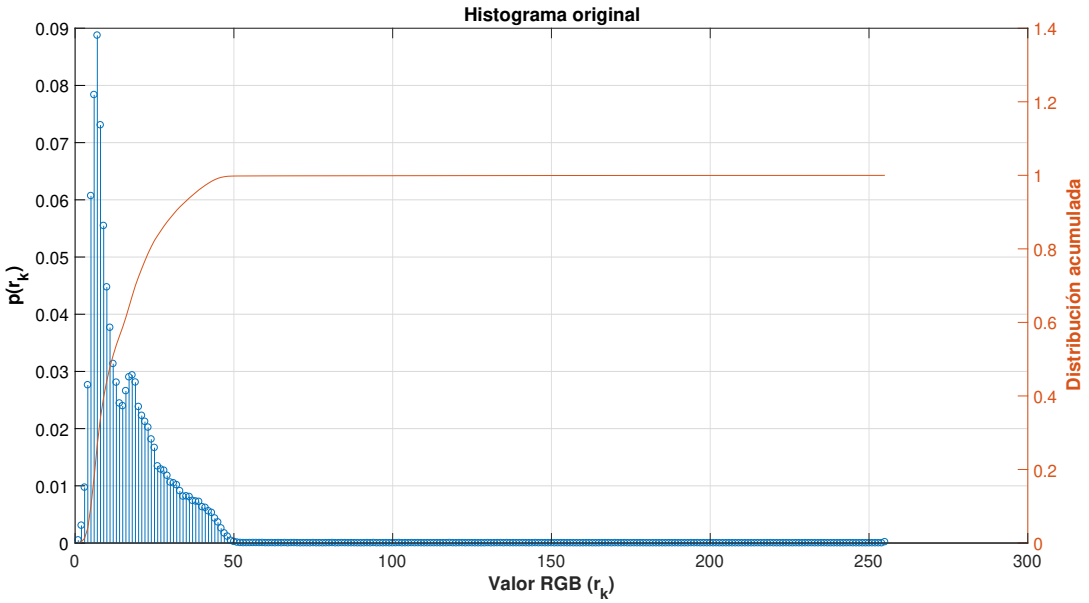

Figura 3: Histograma de la imagen original

Como se puede notar, se confirma todo lo discutido en el literal *a)*. La mayoría de los valores de r_k en este caso están concentrados cerca del valor 0, generando que se vean colores más oscuros en la imagen original, mientras que la presencia de valores de intensidad alta (colores claros) es nula. También es claro que la función de distribución acumulada converge al valor esperado de 1. En este caso, esta función de distribución acumulada es creciente al principio, pues es al principio donde se encuentran concentrados todos los valores de intensidad, mientras que a medida que los valores de intensidad (r_k) crecen, la función de distribución acumulada queda estancada en el valor de 1, pues los valores de intensidad se vuelven 0 muy pronto (generando los colores oscuros de la imagen original).

En la figura 4 se puede observar el histograma correspondiente a la imagen ecualizada, junto con su respectiva función de distribución acumulada.

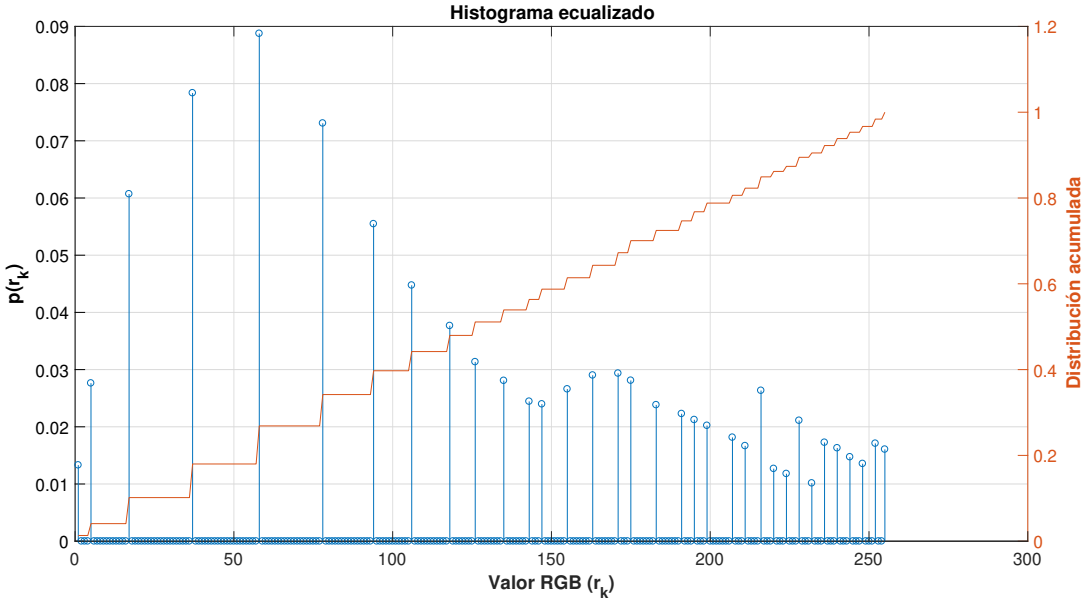

Figura 4: Histograma de la imagen ecualizada

Nuevamente, se puede notar que se logró el efecto mencionado en el literal *a)*: se esparció el histograma de la imagen de entrada (original) generando que los niveles de intensidad de la imagen ecualizada (salida) cubran un rango más amplio de la escala de intensidad definida por los límites 0 y 255. Esto genera que haya más presencia de valores altos de r_k, enclareciendo la imagen y generando la mejora del contraste de la misma. En cuanto a la función de distribución acumulada, se puede observar que en este caso esta va subiendo de forma escalonada, pues ya empieza a haber más presencia de valores altos de intensidad (r_k) y finalmente converge al valor esperado de 1.

Se puede concluir que el proceso de ecualización llevado a cabo por la función `histeq` de MATLAB implementa el procedimiento de ecualización presentado por [1] al pie de la letra.

2. Punto 2

Para realizar este punto primero, usando la función *colorbar*, se visualizaron los coeficientes de la transformada del coseno que se suministraron, obteniendo la figura 5.

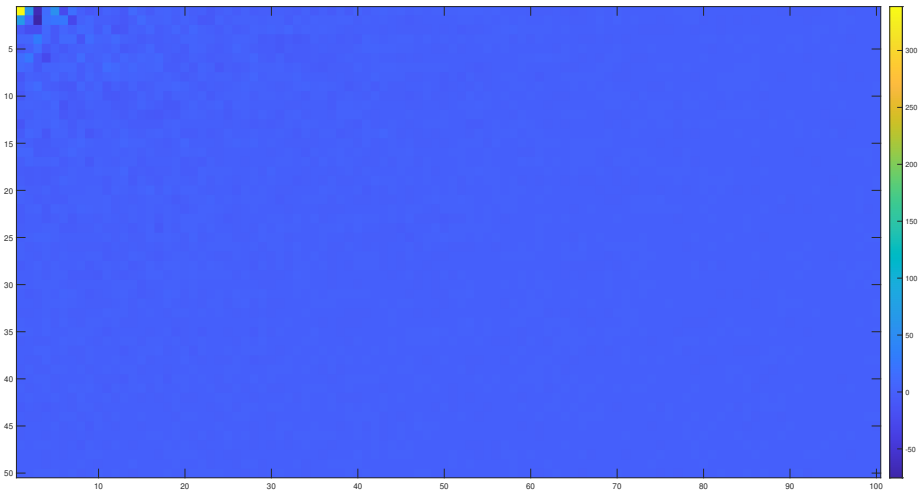

Figura 5: Coeficientes de la Transformada

Ahora, dado que por enunciado se sabe que la dimensión original de la imagen es 630x945, se crearon dos recorridos de tipo *For* que se encargaran de recorrer estas dimensiones, con el fin de obtener el valor en la escala de grises de la nueva imagen. El siguiente paso fue crear otros dos ciclos *For* con la finalidad de recorrer la matriz de coeficientes C suministrada, lo que implica que estos dos nuevos ciclos irán desde 1 hasta 50 y desde 1 hasta 100, respectivamente.
A continuación, se calculan, para cada iteración, el valor de α_u y α_v, de acuerdo a las ecuaciones 5 y 6, respectivamente, donde m_{imagen} y n_{imagen} corresponden a las dimensiones de la imagen original.

$$\alpha_u = \begin{cases} \frac{1}{\sqrt{m_{imagen}}} & \text{si } u = 1 \\ \sqrt{\frac{2}{m_{imagen}}} & \text{De lo contrario} \end{cases} \tag{5}$$

$$\alpha_v = \begin{cases} \frac{1}{\sqrt{n_{imagen}}} & \text{si } v = 1 \\ \sqrt{\frac{2}{n_{imagen}}} & \text{De lo contrario} \end{cases} \tag{6}$$

Finalmente, se empleó la ecuación 7 para obtener el valor en escala de grises de cada pixel de la imagen reconstruida. donde C corresponde a la matriz de coeficientes de la transformada del coseno,

M_{coef} y N_{coef} corresponden a las dimensiones de la matriz C.

$$f(x,y) = \sum_{u=1}^{M_{coef}} \sum_{v=1}^{N_{coef}} \alpha_u \alpha_v C(u,v) \cos\left(\frac{\pi(2(x-1)+1)(u-1)}{2m_{imagen}}\right) \cos\left(\frac{\pi(2(y-1)+1)(v-1)}{2n_{imagen}}\right) \quad (7)$$

Usando la ecuación 7 para todos los 630x945 píxeles de la imagen original se obtuvo la imagen mostrada en la figura 6.

Figura 6: Imagen Obtenida con Coeficientes Suministrados

3. Punto 3

   a) En el script titulado `convolucionImagen.m` se encuentra una rutina que implementa el algoritmo de convolución de dos dimensiones para procesamiento de imágenes. Para un kernel $w(s,t)$ de tamaño nxn, lo que hace este script es implementar la ecuación 8

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \quad (8)$$

   en donde $g(x,y)$ corresponde a la imagen convolucionada, $f(x,y)$ corresponde a la imagen de entrada, los índices s y t recorren el kernel, y los límites de dicho kernel están dados por el valor de a y b.

   b) Para probar que la rutina implementada en el literal pasado estuviera funcionando de manera adecuada, se probó su funcionamiento aplicándole diferentes filtros a la imagen `ladron.jpg`. Uno de estos filtros se aplicó con el objetivo de borrosear la imagen (efecto de *blurring*), y el otro se aplicó con el objetivo de resaltar los bordes de la imagen (*edge detection*).

   En cuanto al filtro de *blurring*, el kernel elegido fue

$$K_{blur} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 0 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

   Para el filtro de *edge detection*, el kernel elegido fue

$$K_{edge} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

En la figura 7 se puede observar la imagen `ladron.jpg` original.

Figura 7: Imagen original

Se procedió a aplicar la rutina `convolucionImagen.m` para esta imagen con el filtro de *blurring*. El resultado obtenido se puede observar en la figura 8.

Figura 8: Imagen convolucionada con el filtro de *blurring*

Como se puede observar de esta figura y comparándola con la imagen original de la figura 7, se puede observar que el efecto de borroseado se logra efectivamente y se puede notar por la forma en que se nota menos definido el contorno de los ojos del ladrón, así como la calle, y el contorno que diferencian los ladrillos del andén.

Posteriormente, se aplicó la rutina a la imagen pero ahora con el filtro de *edge detection*. El resultado se presenta en la figura 9.

Figura 9: Imagen convolucionada con el filtro de *edge detection*

Como se puede notar de esta última figura, la calle y el andén que se podían ver en la imagen original han desaparecido y solo se encuentra enfatizado el contorno del ladrón. También se puede notar que los bordes de la silueta del cuerpo, como de la cachucha se encuentran suficientemente pronunciados y esto es lo que permite extraer de la imagen original solo al sujeto (el ladrón).

c) Finalmente, se analizó el efecto que tuvo cada filtro sobre la iamgen original comparando respectivamente la transformada de Fourier de dos dimensiones de cada caso. Para calcular esta transformada, se usaron las funciones `fft2`, `fftshift` e `imagesc` de MATLAB. Adicionalmente, se recurrió a una última conversión de la transformada bidimensional para adecuarse al tamaño de las imágenes de 8 bits con las que se está trabajando. Al aplicar directamente las funciones `fft2` y `fftshift`, se obtiene un resultado que no es el esperado. Esto se debe al hecho de que las imágenes resultantes contienen muchos datos, y la imagen suministrada está limitada a 8 bits. Para adecuar la transformada obtenida con estas funciones a los 8 bits con los que se está trabajando, se recurre a una conversión logarítmica de la transformada que arroja los resultados esperados de la misma. La implementación de esta conversión se puede encontrar en el script `Punto3.m`.

En la figura 10 se puede observar la transformada de Fourier de la imagen original (figura 7).

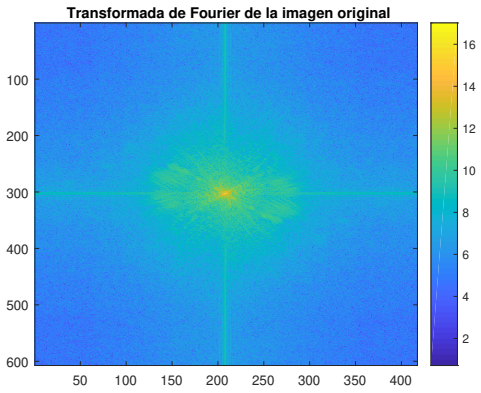

Figura 10: Transformada de Fourier de la imagen original

Por otro lado, en la figura 11 se puede observar la transformada de Fourier de la imagen resultante de convolucionar la original con el filtro de borroseado (*blurring*), que corresponde

a la figura 8.

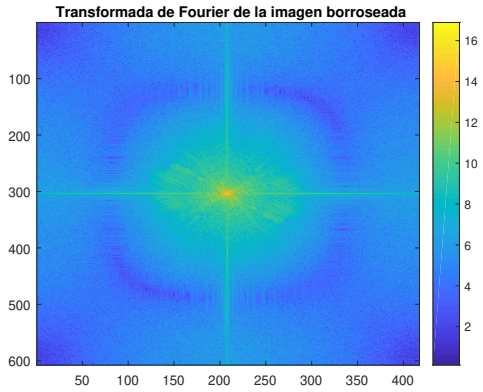

Figura 11: Transformada de Fourier de la imagen borroseada

Finalmente, en la figura 12 se puede observar la transformada de Fourier de la imagen resultante de convolucionar la imagen original con el filtro de *edge detection*, que corresponde a la figura 9.

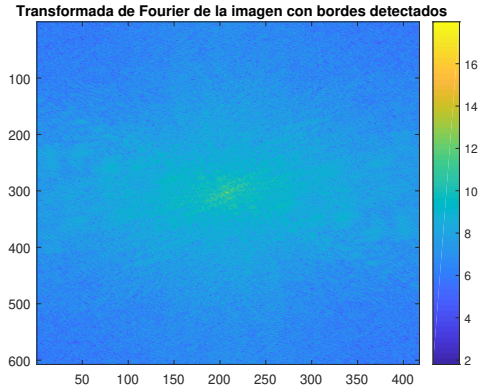

Figura 12: Transformada de Fourier de la imagen con bordes detectados

Como se puede notar de todas estas figuras, que fueron graficadas usando la función `imagesc`, la barra de color del costado derecho indica la magnitud de la transformada de Fourier, en donde los colores amarillo-verde indican las frecuencias más prominentes, mientras que los colores azules claros y oscuros indican una ausencia o presencia poco significativa de dichas frecuencias.

Hablando de la figura 11 y comparándola con la figura 10, se puede notar que algunas de las frecuencias altas de la imagen permanecen después del filtrado. Sin embargo, se puede notar que algunas de estas frecuencias altas se ven disminuidas y reemplazadas por frecuencias bajas. Esto se ve reflejado en el hecho de que hay más magnitudes bajas (azules oscuras) en la figura 11 que en la figura 10 y que algunas magnitudes altas (verdes) de esta última se vieron reemplazadas por las magnitudes bajas (azules) en la transformada de la imagen borroseada.

Finalmente hablando de la transformada de Fourier de la imagen con bordes detectados (figura 12) y comparándola con la transformada de Fourier de la imagen original (figura 10), es claro que las frecuencias más altas han sido canceladas, pues ya no se encuentra una magnitud alta (amarilla) y ha sido reemplazada por una magnitud media (verde). Dado a que las frecuencias altas corresponden a cambios repentinos de contraste, se puede decir que en la figura 12 hay frecuencias bajas porque hay muy pocos cambios de contraste repentinos, pues la mayoría de la imagen tiene un fondo blanco. Esto está reflejado en las magnitudes medias (verdes) de frecuencias en la figura 12.

4. Punto 4

   a) Se puede observar que se detectan los bordes de la imagen, ya que estos se encuentran marcados en un color blanco mientras que el resto de la imagen está en color negro.

   b) Para resolver el problema de optimización que permite encontrar los coeficientes del kernel se formuló lo siguiente:

$$\min \ ||V - H\{U\}||_F^2 \qquad H = \begin{bmatrix} x_1 & x_2 & x_3 \\ 0 & 0 & 0 \\ x_4 & x_5 & x_6 \end{bmatrix}$$

   H corresponde al kernel y U a la imagen que se quiere filtrar; por ende, $H\{U\}$ es el filtrado de la imagen U con el kernel H. Para resolver este problema de optimización se utilizó la función *fmincon* para hallar el valor de los coeficientes x y *conv2* para realizar la operación del filtrado.

   c) El kernel correspondiente a la solución del problema de optimización mostrado anteriormente es:

$$H = \begin{bmatrix} 0{,}0027 & -1{,}4469 & -0{,}0358 \\ 0 & 0 & 0 \\ 0{,}0425 & 1{,}4476 & 0{,}0493 \end{bmatrix}$$

   Por la ubicación de los coeficientes diferentes a cero, se puede decir que se están resaltando los bordes que van en todas las direcciones menos hacia la izquierda y la derecha. Esto se podría evidenciar en la gorra en la imagen derecha de la figura 13, puesto que esta tiene un borde casi horizontal y este no es tan notorio.

   d) A continuación se muestra el resultado de la imagen filtrada con el kernel H encontrado, junto con la imagen original y la entrega para realizar la comparación.

Figura 13: Original (izquierda), imagen filtrada (centro), imagen filtrada con el kernel hallado (derecha)

   Se puede observar que los bordes de la imagen entregada son más marcados y de un color blanco más notorio. Sin embargo, en ambas imágenes se muestra el mismo efecto, solo que en la imagen filtrada con el kernel encontrado este efecto no es tan fuerte como en la otra imagen.

Referencias

[1] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 4th ed. Pearson, 2018.