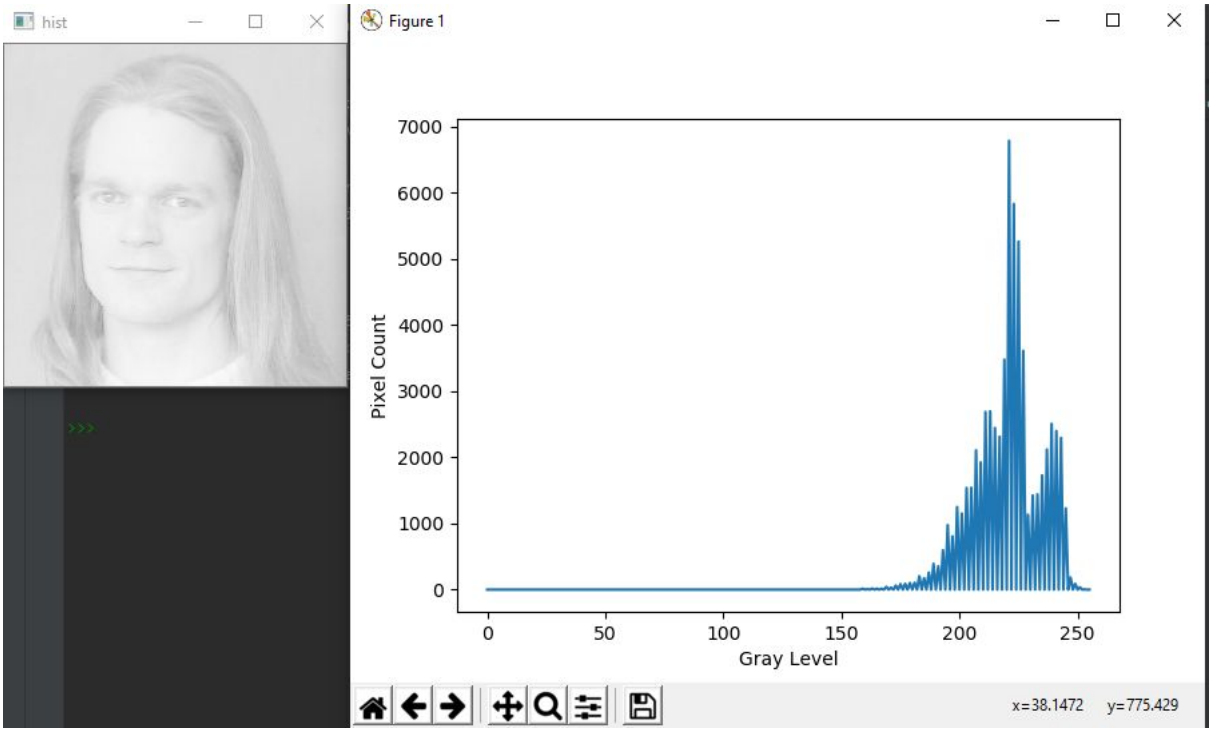
Procesamiento de Imágenes Digitales - Entrega 2 **Gálvez Camacho,Dionisio**

a)

En el histograma representado para la imagen “**low.jpg**” se ve claramente que tiene bajo contraste ya que para la mayoría de los píxeles (eje Y) , los valores en la escala de grises (eje X) están entre 200 y 250 por ello para el ojo humano es casi imperceptible distinguirlos ya que son grises muy parecidos , y por tanto habrá partes de la imagen que se vean con un poco de dificultad. Además este tipo de imágenes con bajo contraste al visualizar el histograma se puede apreciar que es una montaña de valores entre un pequeño rango de valores.

Imagen en escala de grises , e histograma de la misma.



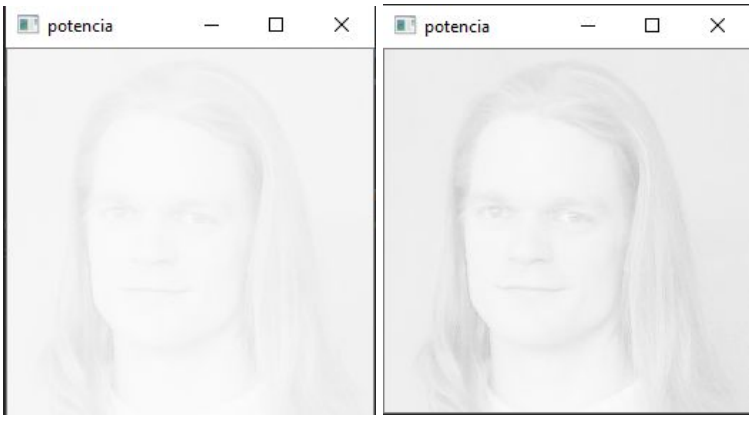
```python
img = cv.imread("low.jpg", 0)
filas, columnas = img.shape  # dimensiones de la imagen
img3 = np.zeros((filas, columnas),dtype=np.uint8)  # devuelve un array de la shap
img4 = np.zeros((filas, columnas),dtype=np.uint8)

#A)
def histograma():
    img2 = cv.calcHist([img], [0], None, [256], [0, 256])  # calcula histograma
    plt.xlabel('Gray Level')
    plt.ylabel('Pixel Count')
    plt.plot(img2)  # monto la gráfica
    plt.show()
    cv.imshow("hist",img)
```

b)

La **transformación logarítmica** se usa para visualizar bajos niveles de intensidad con mayor margen dinámico , osea , es ideal cuando tenemos una imagen oscura para poder verla mejor. En nuestro caso al tener una imagen con poco contraste lo que va a ocurrir es que se va a visualizar peor que la original , se verá practicamente en blanco. Es decir , se usa para expandir los valores de píxeles oscuros a píxeles más claros en una imagen mientras se comprimen los valores de alto nivel , y lo que antes era negro (0,0,0) ahora será (255,255,255).
Por ello pienso que es mejor coger la **transformación de potencia** , con valores fraccionados de "**n**" estrecha los valores oscuros de entrada y amplía los niveles más claros en la salida de la imagen , por tanto si queremos que nuestra imagen de bajo contraste se vea mejor habrá que ponerla más oscura , esto es dándole un valor de "**n**" tal que **x>1** y **x<4**. Con el color negro y blanco ocurre lo mismo que en la función logarítmica.

Transformación de potencia aplicada para n=½ y n=¼ respectivamente.



Transformación de potencia aplicada para n=2 y n=3 respectivamente

Implementación de dicho código:

```
#B)
def transfLog():
    r = np.max(img)  # L-1
    c = r / math.log(1 + r)
    print(c)
    for x in range(filas):
        for y in range(columnas):  # para cada pixel
            img3[x, y] = c * math.log(1 + img[x, y])
    cv.imshow("log",img3)


def transPotencia(n):
    r = np.max(img) #L-1
    c = r/pow(r,n)
    print(c)
    for x in range(filas):
        for y in range(columnas):
            img4[x,y] = c*(pow(img[x,y],n))
    cv.imshow("p",img4)
```

c)

En nuestro caso la imagen posee valores negativos , los cuales los trata de la siguiente manera: por ejemplo si tenemos un píxel en la imagen original con un valor de 96 y en la imagen ecualizada otro con 114 en la escala de grises , y hacemos la diferencia de original-ecualizada su resultado será -18. Pero claro , no puede haber un número negativo en la escala de grises por tanto lo que hace es que al salir negativo , se lo resta al número máximo de L-1 , 256-18 = 238.

Lo cuál va a invertir los colores de algunos píxeles en la escala de grises siempre que el pixel de la imagen original sea menor que la ecualizada. Siguiendo esta regla y el ejemplo

expuesto anteriormente , lo que antes teníamos un gris claro/medio , ahora obtendríamos un color claro al estar cerca de 256.
Si es al contrario , que el píxel de imagen original > imagen ecualizada lo que vamos a conseguir es que salga más oscura al estar más cerca de 0. Lo que antes eran colores blancos o grises claros en ambas fotos se convertirán en negros, y viceversa.


```
#C)
def ecualiza():
    cv.imshow("ORIGINAL",img)
    img5=cv.equalizeHist(img) #aplicamos la
    cv.imshow("EQUALIZADA",img5)
    img2=img/2 #divido/2 para coger la mitad
    img5=img5/2
    imgDif = img2-img5 #hacemos la diferenci
    imgDif = imgDif.astype(np.uint8) #transf

    #MOSTRAMOS MATRICES
    print("ORIGINAL\n")
    print(img2)
    print("\n\n")
    print("MODIFICADA\n")
    print(img5)
    print("\n\n")
    print("DIF\n")
    print(imgDif)

    cv.imshow("DIFERENCIA",imgDif)
```

NOTA: Al final del código en el archivo de Python están las funciones comentadas para poder descomentar una a una y ejecutarlas paso a paso , y así poder visualizar mejor los resultados.