```
#
#  Fecha: 14 de septiembre del 2020
#
#  Autores: Santiago Márquez Álvarez
#           Sergio Mora Pradilla
#
#  Descripción: El código realiza la lectura de una imágen la cual se
convierte a escala de
#               grises y se le adicionan dos tipos de ruido por separado,
el ruido gaussiano
#               y el ruido sal y pimienta.
#               A cada una de las imágenes con ruido se le implementa 4
tipos de filtros, a
#               los cuales se les calcula su tiempo de ejecución, con el
fin de poder eliminar
#               el ruido de cada imágen. Cada uno de estos procesos da
como resultado una
#               imágen, las cuales se guardan en el ordenador como
archivos .JPG.
#               Por último, se lleva a cabo el calculo de la raiz
cuadrada del error cuadrático
#               medio (ECM) para cada uno de los filtros ante la imagen
con ruido que se quizo
#               filtrar en cada caso.
#

# Importación de librerias
import cv2
import numpy as np
from noise import noise
import os
from time import time

if __name__ == '__main__':

    # se lee la imagen a procesar del ordenadory se convierte a escala de
grises
    path = r'C:\Users\Usuario\Documents\Materias Javeriana\procesamiento
de imagenes y vision\semana 2\lena.jpeg'
    lena = cv2.imread(path)
    lena_gray = cv2.cvtColor(lena, cv2.COLOR_BGR2GRAY)

    # se adiciona el ruido S&P
    lena_sp_noisy = noise("s&p", lena_gray.astype(np.float) / 255)
    lena_sp_noisy = (255 * lena_sp_noisy).astype(np.uint8)

    # se adiciona el ruido Gaussiano
    lena_gauss_noisy = noise("gauss", lena_gray.astype(np.float) / 255)
    lena_gauss_noisy = (255 * lena_gauss_noisy).astype(np.uint8)

    # se establece la ruta de guardado de las imagenes del ordenador y se
guardan las imagenes con ruido
    path_img1 = r'C:\Users\Usuario\Documents\Materias
Javeriana\procesamiento de imagenes y vision\semana 6'
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_noisy.jpg'),
lena_gauss_noisy)
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_noisy.jpg'),
```

```
lena_sp_noisy)

    print('***************Tiempos de filtros************')

    # se establece el tamaño de la ventana
    N = 7

    # 1) se implementa el filtro pasa-bajos gaussiano para cada imágen
con ruido calculando su tiempo de ejecución

    tiempo_gauss_sp_antes = time()
    lena_sp_gauss_lp = cv2.GaussianBlur(lena_sp_noisy, (N, N), 1.5, 1.5)
    tiempo_gauss_sp_despues = time()
    tiempo_gauss_sp = tiempo_gauss_sp_despues - tiempo_gauss_sp_antes
    print("tiempo filtro gauss con ruido  s&p ", tiempo_gauss_sp)

    tiempo_gauss_gauss_antes = time()
    lena_gauss_gauss_lp = cv2.GaussianBlur(lena_gauss_noisy, (N, N), 1.5,
1.5)
    tiempo_gauss_gauss_despues = time()
    tiempo_gauss_gauss= float(tiempo_gauss_gauss_despues -
tiempo_gauss_gauss_antes)
    print("tiempo filtro gauss con ruido gaussiano ", tiempo_gauss_gauss)

    # se guardan las imágenes filtradas y la imágen de la estimación del
ruido según lo filtrado en cada una
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_gauss_lp.jpg'),
lena_sp_gauss_lp)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_gauss_lp.jpg'),
lena_gauss_gauss_lp)
    lena_noise_sp_gauss_lp = abs(lena_sp_noisy-lena_sp_gauss_lp)
    lena_noise_gauss_gauss_lp = abs(lena_gauss_noisy -
lena_gauss_gauss_lp)
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_gauss_lp_noise.jpg'),
lena_noise_sp_gauss_lp)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_gauss_lp_noise.jpg'),
lena_noise_gauss_gauss_lp)

    # 2) se implementa el filtro mediana para cada imágen con ruido
calculando su tiempo de ejecución

    tiempo_sp_median_antes = time()
    lena_sp_median = cv2.medianBlur(lena_sp_noisy,N)
    tiempo_sp_median_despues = time()
    tiempo_sp_median = tiempo_sp_median_despues - tiempo_sp_median_antes
    print("tiempo filtro mediana con ruido  s&p ", tiempo_sp_median)

    tiempo_gauss_median_antes = time()
    lena_gauss_median = cv2.medianBlur(lena_gauss_noisy,N)
    tiempo_gauss_median_despues = time()
    tiempo_gauss_median = tiempo_gauss_median_despues -
tiempo_gauss_median_antes
    print("tiempo filtro mediana  con ruido gaussiano ",
tiempo_gauss_median)

    # se guardan las imágenes filtradas y la imágen de la estimación del
ruido según lo filtrado en cada una
```

```
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_median.jpg'),
lena_sp_median)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_median.jpg'),
lena_gauss_median)
    lena_noise_sp_median = abs(lena_sp_noisy-lena_sp_median)
    lena_noise_gauss_median = abs(lena_gauss_noisy - lena_gauss_median)
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_median_noise.jpg'),
lena_noise_sp_median)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_median_noise.jpg'),
lena_noise_gauss_median)

    # 3) se implementa el filtro bilateral para cada imágen con ruido
calculando su tiempo de ejecución

    tiempo_sp_bilateral_antes = time()
    lena_sp_bilateral = cv2.bilateralFilter(lena_sp_noisy, 15, 25, 25)
    tiempo_sp_bilateral_despues = time()
    tiempo_sp_bilateral = tiempo_sp_bilateral_despues -
tiempo_sp_bilateral_antes
    print("tiempo filtro bilateral con ruido s&p", tiempo_sp_bilateral)

    tiempo_gauss_bilateral_antes = time()
    lena_gauss_bilateral = cv2.bilateralFilter(lena_gauss_noisy, 15, 25,
25)
    tiempo_gauss_bilateral_despues = time()
    tiempo_gauss_bilateral = tiempo_gauss_bilateral_despues -
tiempo_gauss_bilateral_antes
    print("tiempo filtro bilateral  con ruido gaussiano ",
tiempo_gauss_bilateral)

    # se guardan las imágenes filtradas y la imágen de la estimación del
ruido según lo filtrado en cada una
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_bilateral.jpg'),
lena_sp_bilateral)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_bilateral.jpg'),
lena_gauss_bilateral)
    lena_noise_sp_bilateral = abs(lena_sp_noisy-lena_sp_bilateral)
    lena_noise_gauss_bilateral = abs(lena_gauss_noisy -
lena_gauss_bilateral)
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_bilateral_noise.jpg'),
lena_noise_sp_bilateral)
    cv2.imwrite(os.path.join(path_img1,
'lena_gauss_bilateral_noise.jpg'), lena_noise_gauss_bilateral)

    # 4) se implementa el filtro nlm para cada imágen con ruido
calculando su tiempo de ejecución

    tiempo_sp_nlm_antes = time()
    lena_sp_nlm = cv2.fastNlMeansDenoising(lena_sp_noisy, 5, 15, 25)
    tiempo_sp_nlm_despues = time()
    tiempo_sp_nlm = tiempo_sp_nlm_despues - tiempo_sp_nlm_antes
    print("tiempo filtro nlm  con ruido s&p ", tiempo_sp_nlm)

    tiempo_gauss_nlm_antes = time()
    lena_gauss_nlm = cv2.fastNlMeansDenoising(lena_gauss_noisy, 5, 15,
25)
    tiempo_gauss_nlm_despues = time()
```

```
    tiempo_gauss_nlm = tiempo_gauss_nlm_despues - tiempo_gauss_nlm_antes
    print("tiempo filtro nlm  con ruido gaussiano ", tiempo_gauss_nlm)

    # se guardan las imágenes filtradas y la imágen de la estimación del
ruido según lo filtrado en cada una
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_nlm.jpg'), lena_sp_nlm)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_nlm.jpg'),
lena_gauss_nlm)
    lena_noise_sp_nlm = abs(lena_sp_noisy-lena_sp_nlm)
    lena_noise_gauss_nlm = abs(lena_gauss_noisy - lena_gauss_nlm)
    cv2.imwrite(os.path.join(path_img1, 'lena_sp_nlm_noise.jpg'),
lena_noise_sp_nlm)
    cv2.imwrite(os.path.join(path_img1, 'lena_gauss_nlm_noise.jpg'),
lena_noise_gauss_nlm)

    # se declaran e inicializan las variables para el calculo de los ECM
    ECM_gauss_gauss_lp = 0.0
    ECM_gauss_median = 0.0
    ECM_gauss_bilateral = 0.0
    ECM_gauss_nlm = 0.0
    ECM_sp_gauss_lp = 0.0
    ECM_sp_median = 0.0
    ECM_sp_bilateral = 0.0
    ECM_sp_nlm = 0.0

    # se guardan las dimensiones de la imágen a procesar
    M,N = lena_gauss_noisy.shape

    # se lleva a cabo la fórmula de ECM por medio de dos ciclos for que
representan las 2 sumatorias a realizar
    for i in range(M):
        for j in range(N):
            ECM_gauss_gauss_lp += np.square(abs(float(lena_gauss_noisy[i
- 1][j - 1]) - float(lena_gauss_gauss_lp[i - 1][j - 1])))
            ECM_gauss_median += np.square(abs(float(lena_gauss_noisy[i -
1][j - 1]) - float(lena_gauss_median[i - 1][j - 1])))
            ECM_gauss_bilateral += np.square(abs(float(lena_gauss_noisy[i
- 1][j - 1]) - float(lena_gauss_bilateral[i - 1][j - 1])))
            ECM_gauss_nlm += np.square(abs(float(lena_gauss_noisy[i -
1][j - 1]) - float(lena_gauss_nlm[i - 1][j - 1])))
            ECM_sp_gauss_lp += np.square(abs(float(lena_sp_noisy[i - 1][j
- 1]) - float(lena_sp_gauss_lp[i - 1][j - 1])))
            ECM_sp_median += np.square(abs(float(lena_sp_noisy[i - 1][j -
1]) - float(lena_sp_median[i - 1][j - 1])))
            ECM_sp_bilateral += np.square(abs(float(lena_sp_noisy[i -
1][j - 1]) - float(lena_sp_bilateral[i - 1][j - 1])))
            ECM_sp_nlm += np.square(abs(float(lena_sp_noisy[i - 1][j -
1]) - float(lena_sp_nlm[i - 1][j - 1])))

# se realiza la división de la fórmula ECM para cada uno de los valores
calculados
ECM_gauss_gauss_lp = np.sqrt(ECM_gauss_gauss_lp/(M*N))
ECM_gauss_median = np.sqrt(ECM_gauss_median/(M*N))
ECM_gauss_bilateral = np.sqrt(ECM_gauss_bilateral/(M*N))
ECM_gauss_nlm = np.sqrt(ECM_gauss_nlm/(M*N))
ECM_sp_gauss_lp = np.sqrt(ECM_sp_gauss_lp/(M*N))
ECM_sp_median = np.sqrt(ECM_sp_median/(M*N))
```

```
ECM_sp_bilateral = np.sqrt(ECM_sp_bilateral/(M*N))
ECM_sp_nlm = np.sqrt(ECM_sp_nlm/(M*N))

# se imprimen los resutaldos de los ECM calculados para cada uno de los
filtros
print('**************Calculo de sqrt(ECM)************')
print('ECM filtro gauss_lp con ruido gaussiano:  ',ECM_gauss_gauss_lp)
print('ECM filtro median con ruido gaussiano:  ',ECM_gauss_median)
print('ECM filtro bilateral con ruido gaussiano:  ',ECM_gauss_bilateral)
print('ECM filtro nlm con ruido gaussiano:  ',ECM_gauss_nlm)
print('ECM filtro gauss_lp con ruido S&P:  ',ECM_sp_gauss_lp)
print('ECM filtro median con ruido S&P:  ',ECM_sp_median)
print('ECM filtro bilateral con ruido S&P:  ',ECM_sp_bilateral)
print('ECM filtro nlm con ruido S&P:  ',ECM_sp_nlm)
```