```
# -*- coding: utf-8 -*-
"""cv_hw3_filters_saraGhavampour_9812762781.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/16EApS67OkKQpXiBmjVgm4eGGgQJYHW0I
"""

!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate

!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import cv2
from math import *
from sklearn.metrics import mean_squared_error
# %matplotlib inline

elaine_img=cv2.imread('Elaine.bmp')
elaine_img = cv2.cvtColor(elaine_img,cv2.COLOR_BGR2GRAY)
plt.imshow(elaine_img,cmap='gray')
plt.axis('off')
plt.show()
#(512, 512) uint8

lena_img=cv2.imread('Lena.bmp')
lena_img = cv2.cvtColor(lena_img,cv2.COLOR_BGR2GRAY)
plt.imshow(lena_img,cmap='gray')
plt.axis('off')
plt.show()

def show_img(*args, figsize=10, is_gray=True, title=None, fontsize=12):
    if isinstance(figsize, int):
        figsize = (figsize, figsize)
    images = args[0] if type(args[0]) is list else list(args)
    cmap=None
    if not is_gray:
        images = list(map(lambda x: cv2.cvtColor(x, cv2.COLOR_BGR2RGB), images))
    else:
        cmap = 'gray'
    plt.figure(figsize=figsize)
    for i in range(1, len(images)+1):
        plt.subplot(1, len(images), i)
        if title is not None:
            plt.title(title[i-1], fontsize=fontsize)

        plt.imshow(images[i-1], cmap=cmap)
        plt.axis('off')

"""3.1.3"""

def apply_box_filter_same_padding(img,window_size):
  out=img.copy()
  pad = floor((window_size-1)/2)
  img=np.pad(img.copy(),((pad,pad),(pad,pad)),mode='constant',constant_values=(0,0))
  filter = np.ones((window_size,window_size))
  for i in range(img.shape[0]-window_size+1):
    for j in range(img.shape[1]-window_size+1):
      window=img[i:i+window_size,j:j+window_size]
      out[i,j]=np.sum(window*filter)/(window_size**2)

  return out.astype('uint8')

# for i in range (0,20):
#   elaine_box_filter_same_1 = apply_box_filter_same_padding(elaine_img,3)

# show_img(elaine_box_filter_same_1)

for i in range (0,50):
  elaine_box_filter_same_2 = apply_box_filter_same_padding(elaine_img,3)

show_img(elaine_box_filter_same_2)

# for i in range (0,150):
#   elaine_box_filter_same_3 = apply_box_filter_same_padding(elaine_img,3)

# show_img(elaine_box_filter_same_3)

# for i in range (0,350):
#   elaine_box_filter_same_3 = apply_box_filter_same_padding(elaine_img,3)

# show_img(elaine_box_filter_same_3)

"""3.1.4"""

from skimage.util import random_noise
def noise_distributer(img,noise_mode,noise_density,var):
  if noise_mode == 's&p':
    out=random_noise(img,noise_mode,amount=noise_density)

  if noise_mode == 'gaussian': # mean?
    out=random_noise(img,noise_mode,var=var)
  return (out*255).astype('uint8')

elaine_noise = noise_distributer(elaine_img,'gaussian',0,0.05)
show_img(elaine_noise,figsize=8)

elaine_denoise1=apply_box_filter_same_padding(elaine_noise,3)
show_img(elaine_denoise1,figsize=6)

elaine_denoise2=apply_box_filter_same_padding(elaine_noise,5)
show_img(elaine_denoise2,figsize=6)

elaine_denoise3=apply_box_filter_same_padding(elaine_noise,7)
show_img(elaine_denoise3,figsize=6)

elaine_denoise4=apply_box_filter_same_padding(elaine_noise,11)
show_img(elaine_denoise4,figsize=6)

"""3.1.5"""

print(mean_squared_error(elaine_img,elaine_denoise1))
print(mean_squared_error(elaine_img,elaine_denoise2))
print(mean_squared_error(elaine_img,elaine_denoise3))
print(mean_squared_error(elaine_img,elaine_denoise4))
```

```
"""3.1.6"""

elaine_120_blured=cv2.imread('120.jpeg')
elaine_120_blured = cv2.cvtColor(elaine_120_blured,cv2.COLOR_BGR2GRAY)
plt.imshow(elaine_120_blured,cmap='gray')
plt.axis('off')
plt.show()

laplacian_HV_filter = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
elaine_120_blured_laplacian_1 = cv2.filter2D(src=elaine_120_blured, ddepth=-1, kernel=laplacian_HV_filter)
show_img(elaine_120_blured_laplacian_1)

copy = elaine_120_blured.copy()
for i in range(0,2):
  copy = cv2.filter2D(src=copy, ddepth=-1, kernel=laplacian_HV_filter)
show_img(copy)

copy = elaine_120_blured.copy()
for i in range(0,3):
  copy = cv2.filter2D(src=copy, ddepth=-1, kernel=laplacian_HV_filter)
show_img(copy)

copy = elaine_120_blured.copy()
for i in range(0,4):
  copy = cv2.filter2D(src=copy, ddepth=-1, kernel=laplacian_HV_filter)
show_img(copy)

copy = elaine_120_blured.copy()
for i in range(0,5):
  copy = cv2.filter2D(src=copy, ddepth=-1, kernel=laplacian_HV_filter)
show_img(copy)

"""3.2

3.2.1
"""

noisy_sp_img_results=[]

noisy_sp_img_results.append(noise_distributer(elaine_img,'s&p',0.05,0))
noisy_sp_img_results.append(noise_distributer(elaine_img,'s&p',0.1,0))
noisy_sp_img_results.append(noise_distributer(elaine_img,'s&p',0.2,0))
noisy_sp_img_results.append(noise_distributer(elaine_img,'s&p',0.4,0))
titles=['salt&pepper noise with noise density 0.05','salt&pepper noise with noise density 0.1','salt&pepper noise with noise density 0.2','salt&pepper noise with noise density 0.4']

show_img(noisy_sp_img_results, figsize=30,title=titles)

def apply_median_filter(img,window_size):
  img_copy=img.copy()
  shift = floor(window_size/2)
  x,y=img_copy.shape
  out = np.zeros((x-2*shift,y-2*shift))
  for i in range(shift,x-shift):
    for j in range(shift,y-shift):
      window = img_copy[i-shift:i+shift,j-shift:j+shift]
      window = window.ravel()
      median=np.median(window)
      out[i-shift,j-shift]=median

  return out

elaine_sp_01_median_5=apply_median_filter(noisy_sp_img_results[3],3)
show_img(elaine_sp_01_median_5)

elaine_sp_01_median_5

filter_wondow_sizes=[3,5,7,9,11]
median_filter_results=[]
median_filter_results_titles=[]
noise_densities=[0.05,0.1,0.2,0.4]

for (index,noisy_img) in enumerate(noisy_sp_img_results):
  median_results_window=[] # each row for each imahe with density
  median_results_window_title=[]
  for size in  filter_wondow_sizes:
    median_img=apply_median_filter(noisy_img,size)
    median_results_window.append(median_img)# each col is size
    median_results_window_title.append(f'median filter size: {size} on s&p noise: {noise_densities[index]}')
    shift=floor(size/2)
    x,y=elaine_img.shape
    bounded_noisey_image = elaine_img[shift:x-shift,shift:y-shift]
    mse = mean_squared_error(bounded_noisey_image,median_img)
    print(f'mse elaine_img & median filter {size} size on sp noise density {noise_densities[index]} is :{mse}')
  print('-----------------------------------------------------------')
  median_filter_results.append(median_results_window)
  median_filter_results_titles.append(median_results_window_title)

for i in range(0,np.array(median_filter_results).shape[0]):
  show_img(median_filter_results[i], figsize=30,title=median_filter_results_titles[i])

"""3.2.2"""

noisy_gaussian_img_results=[]

noisy_gaussian_img_results.append(noise_distributer(elaine_img,'gaussian',0,0.01))
noisy_gaussian_img_results.append(noise_distributer(elaine_img,'gaussian',0,0.05))
noisy_gaussian_img_results.append(noise_distributer(elaine_img,'gaussian',0,0.1))

titles=['gaussian noise with noise density 0.01','gaussian noise with noise density 0.05','gaussian noise with noise density 0.1']

show_img(noisy_gaussian_img_results, figsize=20,title=titles)

filter_wondow_sizes=[3,5,7,9,11]
median_filter_results=[]
median_filter_results_titles=[]
noise_densities=[0.01,0.05,0.1]

for (index,noisy_img) in enumerate(noisy_gaussian_img_results):
  median_results_window=[] # each row for each imahe with density
  median_results_window_title=[]
  for size in  filter_wondow_sizes:
    median_img=apply_median_filter(noisy_img,size)
    median_results_window.append(median_img)# each col is size
    median_results_window_title.append(f'median filter size: {size} on gaussian noise: {noise_densities[index]}')
    shift=floor(size/2)
    x,y=elaine_img.shape
    bounded_noisey_image = elaine_img[shift:x-shift,shift:y-shift]
    mse = mean_squared_error(bounded_noisey_image,median_img)
    print(f'mse elaine_img & median filter {size} size on gaussian noise density {noise_densities[index]} is :{mse}')
  print('-----------------------------------------------------------')
```

```
  median_filter_results.append(median_results_window)
  median_filter_results_titles.append(median_results_window_title)

for i in range(0,np.array(median_filter_results).shape[0]):
  show_img(median_filter_results[i], figsize=30,title=median_filter_results_titles[i])

def apply_box_filter(img,window_size):
  img_copy=img.copy()
  shift = floor(window_size/2)
  x,y=img_copy.shape
  out = np.zeros((x-2*shift,y-2*shift))
  for i in range(shift,x-shift):
    for j in range(shift,y-shift):
      window = img_copy[i-shift:i+shift+1,j-shift:j+shift+1]
      #print(window)
      box_avg = (np.sum(window)) / (window_size**2)
      out[i-shift,j-shift]=box_avg

  return out.astype('uint8')

filter_wondow_sizes=[3,5,7,9,11]
boxfilter_filter_results=[]
boxfilter_filter_results_titles=[]
noise_densities=[0.01,0.05,0.1]

for (index,noisy_img) in enumerate(noisy_gaussian_img_results):
  boxfilter_results_window=[] # each row for each imahe with density
  boxfilter_results_window_title=[]
  for size in  filter_wondow_sizes:
    boxfilter_img=apply_box_filter(noisy_img,size)
    boxfilter_results_window.append(boxfilter_img)# each col is size
    boxfilter_results_window_title.append(f'boxfilter filter size: {size} on gaussian noise: {noise_densities[index]}')
    shift=floor(size/2)
    x,y=elaine_img.shape
    bounded_noisey_image = elaine_img[shift:x-shift,shift:y-shift]
    mse = mean_squared_error(bounded_noisey_image,boxfilter_img)
    print(f'mse elaine_img & boxfilter filter {size} size on gaussian noise density {noise_densities[index]} is :{mse}')
  print('--------------------------------------------------------')
  boxfilter_filter_results.append(boxfilter_results_window)
  boxfilter_filter_results_titles.append(boxfilter_results_window_title)

for i in range(0,np.array(boxfilter_filter_results).shape[0]):
  show_img(boxfilter_filter_results[i], figsize=30,title=boxfilter_filter_results_titles[i])

def apply_guessian_filter(img,window_size):
  img_copy=img.copy()
  shift = floor(window_size/2)
  x,y=img_copy.shape
  out = np.zeros((x-2*shift,y-2*shift))
  for i in range(shift,x-shift):
    for j in range(shift,y-shift):
      window = img_copy[i-shift:i+shift+1,j-shift:j+shift+1]
      #print(window)
      box_avg = (np.sum(window)) / (window_size**2)
      out[i-shift,j-shift]=box_avg

      x, y = np.mgrid[-kernel_size//2 + 1:kernel_size//2 + 1, -kernel_size//2 + 1:kernel_size//2 + 1]
      g = np.exp(-((x**2 + y**2)/(2.0*sigma**2)))
      return g/g.sum()

  return out.astype('uint8')

"""3.4"""

edge_detector_341a=1/2*np.array([1,0,-1])
edge_detector_341b=1/6*np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
edge_detector_341c=1/8*np.array([[1,0,-1],[2,0,-2],[1,0,-1]])

def convolve2D(img,filter):
  f=filter.shape[0]
  pad = floor((f-1)/2)
  out=img.copy()
  img=np.pad(img.copy(),((pad,pad),(pad,pad)),mode='constant',constant_values=(0,0))
  for i in range(img.shape[0]-f+1):
    for j in range(img.shape[1]-f+1):
      window=img[i:i+f,j:j+f]
      out[i,j]=np.sum(window*filter)

  return out

elaine_341a = convolve2D(elaine_img,edge_detector_341a)
show_img(elaine_341a,figsize=10)

elaine_341a = cv2.filter2D(src=elaine_img, ddepth=-1, kernel=edge_detector_341a)
show_img(elaine_341a,figsize=10)

elaine_341b = cv2.filter2D(src=elaine_img, ddepth=-1, kernel=edge_detector_341b)
show_img(elaine_341b,figsize=10)

elaine_341c = cv2.filter2D(src=elaine_img, ddepth=-1, kernel=edge_detector_341c)
show_img(elaine_341c,figsize=10)

edge_detector_342a=np.array([[1,0],[0,-1]])
edge_detector_342b=np.array([[0,1],[-1,0]])
elaine_342a = cv2.filter2D(src=elaine_img, ddepth=-1, kernel=edge_detector_342a)
elaine_342b = cv2.filter2D(src=elaine_img, ddepth=-1, kernel=edge_detector_342b)
show_img(elaine_342a,elaine_342b,title=['3.4.2 filter a','3.4.2 filter b'],figsize=20)

"""3.5     unsharp masking"""

lena_gus3=cv2.GaussianBlur(lena_img,(3,3),2)
lena_gus5=cv2.GaussianBlur(lena_img,(5,5),2)
lena_gus7=cv2.GaussianBlur(lena_img,(7,7),2)
lena_gus9=cv2.GaussianBlur(lena_img,(9,9),2)
lena_gus11=cv2.GaussianBlur(lena_img,(11,11),2)

show_img(lena_gus11,lena_img)

def unsharp_masking_formula(alpha,img,smooth_img):
  out = (1-alpha)*img + alpha*smooth_img
  return out

lena_gus3_alpha0=unsharp_masking_formula(0,lena_img,lena_gus3)
lena_gus3_alpha1=unsharp_masking_formula(1,lena_img,lena_gus3)
show_img(lena_img,lena_gus3_alpha0,lena_gus3_alpha1,title=['lena_img','lena_gus3_alpha0','lena_gus3_alpha1'],figsize=16)

lena_gus5_alpha0=unsharp_masking_formula(0,lena_img,lena_gus5)
lena_gus5_alpha1=unsharp_masking_formula(1,lena_img,lena_gus5)
show_img(lena_img,lena_gus5_alpha0,lena_gus5_alpha1,title=['lena_img','lena_gus5_alpha0','lena_gus5_alpha1'],figsize=16)
```

```
lena_gus7_alpha0=unsharp_masking_formula(0,lena_img,lena_gus7)
lena_gus7_alpha1=unsharp_masking_formula(1,lena_img,lena_gus7)
show_img(lena_img,lena_gus7_alpha0,lena_gus7_alpha1,title=['lena_img','lena_gus7_alpha0','lena_gus7_alpha1'],figsize=16)

lena_gus9_alpha0=unsharp_masking_formula(0,lena_img,lena_gus9)
lena_gus9_alpha1=unsharp_masking_formula(1,lena_img,lena_gus9)
show_img(lena_img,lena_gus9_alpha0,lena_gus9_alpha1,title=['lena_img','lena_gus9_alpha0','lena_gus9_alpha1'],figsize=16)

lena_gus11_alpha0=unsharp_masking_formula(0,lena_img,lena_gus11)
lena_gus11_alpha1=unsharp_masking_formula(1,lena_img,lena_gus11)
show_img(lena_img,lena_gus11_alpha0,lena_gus11_alpha1,title=['lena_img','lena_gus11_alpha0','lena_gus11_alpha1'],figsize=16)

def unsharp_masking_improved(alpha,img,smooth_img):
  out = img + alpha*(img-smooth_img)
  return out

lena_gus3_alpha0=unsharp_masking_improved(2,lena_img,lena_gus3)
lena_gus3_alpha1=unsharp_masking_improved(3,lena_img,lena_gus3)
show_img(lena_img,lena_gus3_alpha0,lena_gus3_alpha1,title=['lena_img','lena_gus3_alpha2','lena_gus3_alpha3'],figsize=16)

lena_gus5_alpha0=unsharp_masking_formula(2,lena_img,lena_gus5)
lena_gus5_alpha1=unsharp_masking_formula(3,lena_img,lena_gus5)
show_img(lena_img,lena_gus5_alpha0,lena_gus5_alpha1,title=['lena_img','lena_gus5_alpha2','lena_gus5_alpha3'],figsize=16)

lena_gus7_alpha0=unsharp_masking_formula(2,lena_img,lena_gus7)
lena_gus7_alpha1=unsharp_masking_formula(3,lena_img,lena_gus7)
show_img(lena_img,lena_gus7_alpha0,lena_gus7_alpha1,title=['lena_img','lena_gus7_alpha2','lena_gus7_alpha3'],figsize=16)

lena_gus9_alpha0=unsharp_masking_formula(2,lena_img,lena_gus9)
lena_gus9_alpha1=unsharp_masking_formula(3,lena_img,lena_gus9)
show_img(lena_img,lena_gus9_alpha0,lena_gus9_alpha1,title=['lena_img','lena_gus9_alpha2','lena_gus9_alpha3'],figsize=16)

lena_gus7_alpha0=unsharp_masking_formula(2,lena_img,lena_gus7)
lena_gus7_alpha1=unsharp_masking_formula(3,lena_img,lena_gus7)
show_img(lena_img,lena_gus7_alpha0,lena_gus7_alpha1,title=['lena_img','lena_gus7_alpha2','lena_gus7_alpha3'],figsize=16)
```