Salt and pepper

```
#@title Salt and pepper
#salt and pepper

import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('bin-noise.png')
img_g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresh = 140
maxValue = 255
th, bin1 = cv2.threshold(img_g, thresh, maxValue, cv2.THRESH_BINARY)
img = bin1
cv2.imshow('Normal image', img)
cv2.waitKey(0)
x, y = img.shape
noiseimg = np.zeros((x,y), dtype=np.float32)
pep = 0.05
sal = 1-pep
for i in range(x):
    for j in range(y):
        rdn = np.random.random()
        if rdn < pep:
            noiseimg[i][j] = 0
        elif rdn > sal:
            noiseimg[i][j] = 1
        else:
            noiseimg[i][j] = img[i][j]
cv2.imshow('Noise image', noiseimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Unable to connect to the runtime.

```
        if rdn < pep:
            noiseimg[i][j] = 0
        elif rdn > sal:
            noiseimg[i][j] = 1
        else:
            noiseimg[i][j] = rdn
thresh = 0
maxValue = 1
th, bin1 = cv2.threshold(noiseimg, thresh, maxValue, cv2.THRESH_BINARY)
plt.hist(bin1.flat,bins=100,range =None,),plt.title("bin1 noise graph")
plt.show()
```

Gaussian noise

```
#@title Gaussian noise
#Gaussian noise
import cv2
import numpy as np
from scipy.stats.kde import gaussian_kde
import matplotlib.pyplot as plt
```

0s completed at 10:03 AM

```
img_g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
t = 140
maxValue = 255
th, binary = cv2.threshold(img_g, t, maxValue, cv2.THRESH_BINARY)
img= binary
cv2.imshow('Orignal image', img)
cv2.waitKey(5000)
x, y =img.shape
mean =0
var=0.05
sigma = np.sqrt(var)
n=np.random.normal(loc=mean, scale=sigma, size=(x,y))
kde = gaussian_kde(n.reshape(int(x*y)))
dist_space=np.linspace(np.min(n), np.max(n), 100)
plt.plot(dist_space, kde(dist_space))
plt.xlabel('Noise pixel value')
plt.ylabel('Frequency')
plt.show()
noiseimg=img+n
cv2.imshow('Noise image', noiseimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

rayleigh noise

```
#@title rayleigh noise
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('bin-noise.png')
img_g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Unable to connect to the runtime.

```
th, binary = cv2.threshold(img_g, t, maxV, cv2.THRESH_BINARY)
img = binary
cv2.imshow('Original image', img)
cv2.waitKey(0)
x, y = img.shape
gfg = np.random.rayleigh(scale=0.2, size=(x, y))
plt.figure()
plt.hist(gfg.flat,bins=100,range =None,),plt.title("rayleigh graph")
plt.show()
noiseimg=img+gfg
cv2.imshow('Rayleigh noise image', noiseimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

median filter

```
#@title median filter
import cv2
import numpy as np
from scipy.stats.kde import gaussian_kde
img = cv2.imread("binary.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresh = 140
maxValue = 255
th, binary = cv2.threshold(gray, thresh, maxValue, cv2.THRESH_BINARY)
img = binary
x, y = img.shape
saltpepimg = np.zeros((x,y), dtype=np.float32)
pepper = 0.05
salt = 1-pepper
for i in range(x):
  for j in range(y):
    rdn = np.random.random()
    if rdn < pepper:
      saltpepimg[i][j] = 0
    elif rdn > salt:
      saltpepimg[i][j] = 1
    else:
    saltpepimg[i][j] = img[i][j]
cv2.imshow('Salt and pepper', saltpepimg)
cv2.waitKey(0)
mean =0
var=0.05
sigma = np.sqrt(var)
n=np.random.normal(loc=mean, scale=sigma, size=(x,y))
kde = gaussian_kde(n.reshape(int(x*y)))
gaussimg=img+n
cv2.imshow('Gaussian', gaussimg)
cv2.waitKey(0)
gfg = np.random.rayleigh(scale=0.2, size=(x, y))
rayimg=img+gfg
cv2.imshow('Rayleigh', rayimg)
cv2.waitKey(0)
saltp_medain = cv2.medianBlur(saltpepimg.astype(np.float32),(3),0)
gaussian_medain = cv2.medianBlur(gaussimg.astype(np.float32),(3),0)
rayleigh_medain = cv2.medianBlur(rayimg.astype(np.float32),(3),0)
                                 er', saltp_medain)
cv2.imshow('Gaussian after filter', gaussian_medain)
cv2.waitKey(0)
cv2.imshow('Rayleigh after filter', rayleigh_medain)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Unable to connect to the runtime.

bilateral filter

```
#@title bilateral filter
import cv2
import numpy as np
from scipy.stats.kde import gaussian_kde
img = cv2.imread('binary.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
t = 140
maxV = 255
th, binary = cv2.threshold(gray, t, maxV, cv2.THRESH_BINARY)
img = binary
x, y = img.shape
saltpepimg = np.zeros((x,y), dtype=np.float32)
pepper = 0.05
salt = 1-pepper
```

```
salt = 1-pepper
for i in range(x):
  for j in range(y):
    rdn = np.random.random()
    if rdn < pepper:
      saltpepimg[i][j] = 0
    elif rdn > salt:
      saltpepimg[i][j] = 1
    else:
      saltpepimg[i][j] = img[i][j]
cv2.imshow('Salt and pepper', saltpepimg)
cv2.waitKey(0)
mean =0
var=0.05
sigma = np.sqrt(var)
n=np.random.normal(loc=mean, scale=sigma, size=(x,y))
kde = gaussian_kde(n.reshape(int(x*y)))
gaussimg=img+n
cv2.imshow('Gaussian', gaussimg)
cv2.waitKey(0)
gfg = np.random.rayleigh(scale=0.2, size=(x, y))
rayimg=img+gfg
cv2.imshow('Rayleigh', rayimg)
cv2.waitKey(0)
saltp_bilateral = cv2.bilateralFilter(saltpepimg, 9, 70, 70)
gaussimg = np.float32(gaussimg)
rayimg = np.float32(rayimg)
gaussian_bilateral = cv2.bilateralFilter(gaussimg, 9, 50, 50)
rayleigh_bilateral = cv2.bilateralFilter(rayimg, 15, 50, 50)
cv2.imshow('Saltandpepper after filter', saltp_bilateral)
cv2.waitKey(0)
cv2.imshow('Gaussian after filter', gaussian_bilateral)
cv2.waitKey(0)
cv2.imshow('Rayleigh after filter', rayleigh_bilateral)
cv2.waitKey(0)
```

Unable to connect to the runtime.

weighted filter

```
#@title weighted filter
import cv2
import numpy as np
from scipy.stats.kde import gaussian_kde
import matplotlib.pyplot as plt
from scipy.ndimage.filters import convolve
img = cv2.imread('Binary.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresh = 140
maxValue = 255
th, binary = cv2.threshold(gray, thresh, maxValue, cv2.THRESH_BINARY)
img = binary
x, y = img.shape

saltpepimg = np.zeros((x,y), dtype=np.float32)
pepper = 0.05
salt = 1-pepper
for i in range(x):
  for j in range(y):
```

```
    rdn = np.random.random()
    if rdn < pepper:
      saltpepimg[i][j] = 0
    elif rdn > salt:
      saltpepimg[i][j] = 1
    else:
      saltpepimg[i][j] = img[i][j]

mean =0
var=0.05
sigma = np.sqrt(var)
n=np.random.normal(loc=mean, scale=sigma, size=(x,y))

kde = gaussian_kde(n.reshape(int(x*y)))
gaussimg=img+n

gfg = np.random.rayleigh(scale=0.2, size=(x, y))
rayimg=img+gfg

wkernel3 = np.array([
    [1,2,1],
    [2,4,2],
    [1,2,1]])
wkernel5 = np.array([
    [1,2,4,2,1],
    [2,4,8,4,2],
    [4,8,16,8,4],
    [2,4,8,4,2],
    [1,2,4,2,1]])

k3=list(map(sum,wkernel3))
k31=sum(k3)
k5=list(map(sum,wkernel5))
k51=sum(k5)
k3 = wkernel3/k31
```

Unable to connect to the runtime.

```
saltpep3 = convolve(saltpepimg,k3)
gauss3 = convolve(gaussimg,k3)
ray3 = convolve(rayimg,k3)
saltpep5 = convolve(saltpepimg,k5)
gauss5 = convolve(gaussimg,k5)
ray5 = convolve(rayimg,k5)
```

mean filter

```
#@title mean filter
import cv2
import numpy as np
from scipy.stats.kde import gaussian_kde
import matplotlib.pyplot as plt
from scipy.ndimage.filters import convolve
img = cv2.imread('binary.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
t = 140
maxV = 255
th, binary = cv2.threshold(gray, t, maxV, cv2.THRESH_BINARY)
img = binary
```

```
x, y = img.shape

saltpepimg = np.zeros((x,y), dtype=np.float32)
pepper = 0.05
salt = 1-pepper
for i in range(x):
  for j in range(y):
    rdn = np.random.random()
    if rdn < pepper:
      saltpepimg[i][j] = 0
    elif rdn > salt:
      saltpepimg[i][j] = 1
    else:
      saltpepimg[i][j] = img[i][j]

mean=0
var=0.05
sigma = np.sqrt(var)
n=np.random.normal(loc=mean, scale=sigma, size=(x,y))

kde = gaussian_kde(n.reshape(int(x*y)))
gaussimg=img+n

gfg = np.random.rayleigh(scale=0.2, size=(x, y))
rayimg=img+gfg

wkernel3 = np.array([
[1,1,1],
[1,1,1],
[1,1,1]])
k3=list(map(sum,wkernel3))
k31=sum(k3)
k3 = wkernel3/k31
saltpep3 = convolve(saltpepimg,k3)
```

Unable to connect to the runtime.