在opencv中，可以利用随机噪声、滤波器等方法为图像叠加仿真的雨滴的运动轨迹，使通常情况下拍摄的图像有了烟雨蒙蒙的效果。

1.生成随机噪声

首先，我们需要生成不同密度的随机噪声来模拟不同大小的余量，于是利用了下面的函数来生成。主要使用了均匀随机数和阈值来控制噪声的水平。由于生成的噪声是浮点数，所以在value上乘了尺度缩小因子。

```
import cv2
import numpy as np


def get_noise(img, value=10):


    noise = np.random.uniform(0, 256, img.shape[0:2])#在函数内部，首先创建了一个与输入图像大小相同的随机数矩阵noise，取值范围在0到256之间，表示噪声的强度
    #接下来，根据value参数的值，通过设置阈值来控制噪声水平。将噪声矩阵中小于(256 - v)的元素设为0，以保留最大的一部分作为噪声。这里的v是通过将value乘以0.01来计算得到的。
    v = value * 0.01
    noise[np.where(noise < (256 - v))] = 0
 #进行初次模糊处理，使用一个3x3的卷积核k来平滑噪声。使用cv2.filter2D函数对噪声进行卷积操作。
    k = np.array([[0, 0.1, 0],
                  [0.1, 8, 0.1],
                  [0, 0.1, 0]])

    noise = cv2.filter2D(noise, -1, k)

    # 可以输出噪声看看
    '''cv2.imshow('img',noise)
    cv2.waitKey()
    cv2.destroyWindow('img')'''
    return noise
```

2.生成雨滴模糊噪声

随后，需要对噪声拉长、旋转方向，模拟不同大小和方向的雨水。

```
def rain_blur(noise, length=10, angle=0,w=1):
    '''
    将噪声加上运动模糊,模仿雨滴

    >>>输入
    noise：输入噪声图，shape = img.shape[0:2]
    length: 对角矩阵大小，表示雨滴的长度
    angle：  倾斜的角度，逆时针为正
    w:       雨滴大小

    >>>输出带模糊的噪声

    '''


    #这里由于对角阵自带45度的倾斜，逆时针为正，所以加了-45度的误差，保证开始为正
    trans = cv2.getRotationMatrix2D((length/2, length/2), angle-45, 1-length/100.0)
    dig = np.diag(np.ones(length))   #生成对角矩阵，对角矩阵中的对角元素都为1，其大小由 length 参数指定。
    k = cv2.warpAffine(dig, trans, (length, length))  #使用 cv2.warpAffine 函数将对角矩阵 dig 进行仿射变换，得到模糊核 k。
    k = cv2.GaussianBlur(k,(w,w),0)     #高斯模糊这个旋转后的对角核，使得雨有宽度

    #k = k / length                             #是否归一化

    blurred = cv2.filter2D(noise, -1, k)     #用刚刚得到的旋转后的核，进行滤波

    #转换到0-255区间
    cv2.normalize(blurred, blurred, 0, 255, cv2.NORM_MINMAX)
    blurred = np.array(blurred, dtype=np.uint8)
    '''
    cv2.imshow('img',blurred)
    cv2.waitKey()
```

```
    cv2.destroyWindow('img')'''

    return blurred
```

3.图像叠加

最后对将生成的雨滴噪声和原始图像叠加即可得到模拟的下雨场景了。
第一种方式是利用逐个通道的像素与生成雨滴噪声进行操作，即在噪声中为黑色的位置，保留原图；有雨滴噪声的位置修改为带权重的雨滴噪声的纹理（赋值为噪声值）：

```
def alpha_rain(rain,img,beta = 0.8):

    #输入雨滴噪声和图像
    #beta = 0.8   #results weight
    #显示下雨效果

    #expand dimensin
    #将二维雨噪声扩张为三维单通道
    #并与图像合成在一起形成带有alpha通道的4通道图像
    rain = np.expand_dims(rain,2)
    rain_effect = np.concatenate((img,rain),axis=2)  #add alpha
channel

    rain_result = img.copy()     #拷贝一个掩膜
    rain = np.array(rain,dtype=np.float32)      #数据类型变为浮点数，后面
要叠加，防止数组越界要用32位
    rain_result[:,:,0]= rain_result[:,:,0] * (255-
rain[:,:,0])/255.0 + beta*rain[:,:,0]
    rain_result[:,:,1] = rain_result[:,:,1] * (255-rain[:,:,0])/255
+ beta*rain[:,:,0]
    rain_result[:,:,2] = rain_result[:,:,2] * (255-rain[:,:,0])/255
+ beta*rain[:,:,0]
    #对每个通道先保留雨滴噪声图对应的黑色（透明）部分，再叠加白色的雨滴噪声部分
（有比例因子）

    cv2.imshow('rain_effct_result',rain_result)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

第二种方式就是利用图像加权的方法，将雨滴图叠加到原图上。由于背景是黑色，所以调整后的图像亮度会下降，在某些场景下更适合雨天的模拟（乌云，光线暗）。

```
def add_rain(rain,img,alpha=0.9):
    #输入雨滴噪声和图像
    #alpha：原图比例因子
    #显示下雨效果

    #chage rain into  3-dimenis
    #将二维rain噪声扩张为与原图相同的三通道图像
    rain = np.expand_dims(rain,2)
    rain = np.repeat(rain,3,2)

    #加权合成新图
    result = cv2.addWeighted(img,alpha,rain,1-alpha,1)
    cv2.imshow('rain_effct',result)
    cv2.waitKey()
    cv2.destroyWindow('rain_effct')
```

读入图像后，设置value大小控制雨滴数量，length大小控制雨水水痕长度，angle大小来控制雨下落的角度，w来控制雨点粗细程度。

```
img = cv2.imread('demo.jpeg')
noise = get_noise(img,value=500)
rain = rain_blur(noise,length=50,angle=-30,w=3)
alpha_rain(rain,img,beta=0.6)  #方法一，透明度赋值
#add_rain(rain,img)     #方法二,加权后有玻璃外的效果
```