In [1]:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os
%matplotlib inline
```

In [2]:

```
def load(path):
    img=cv.imread(path)
    #opencv reads the image in BGR, thus we have to turn it to RGB
    img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
    return img
```

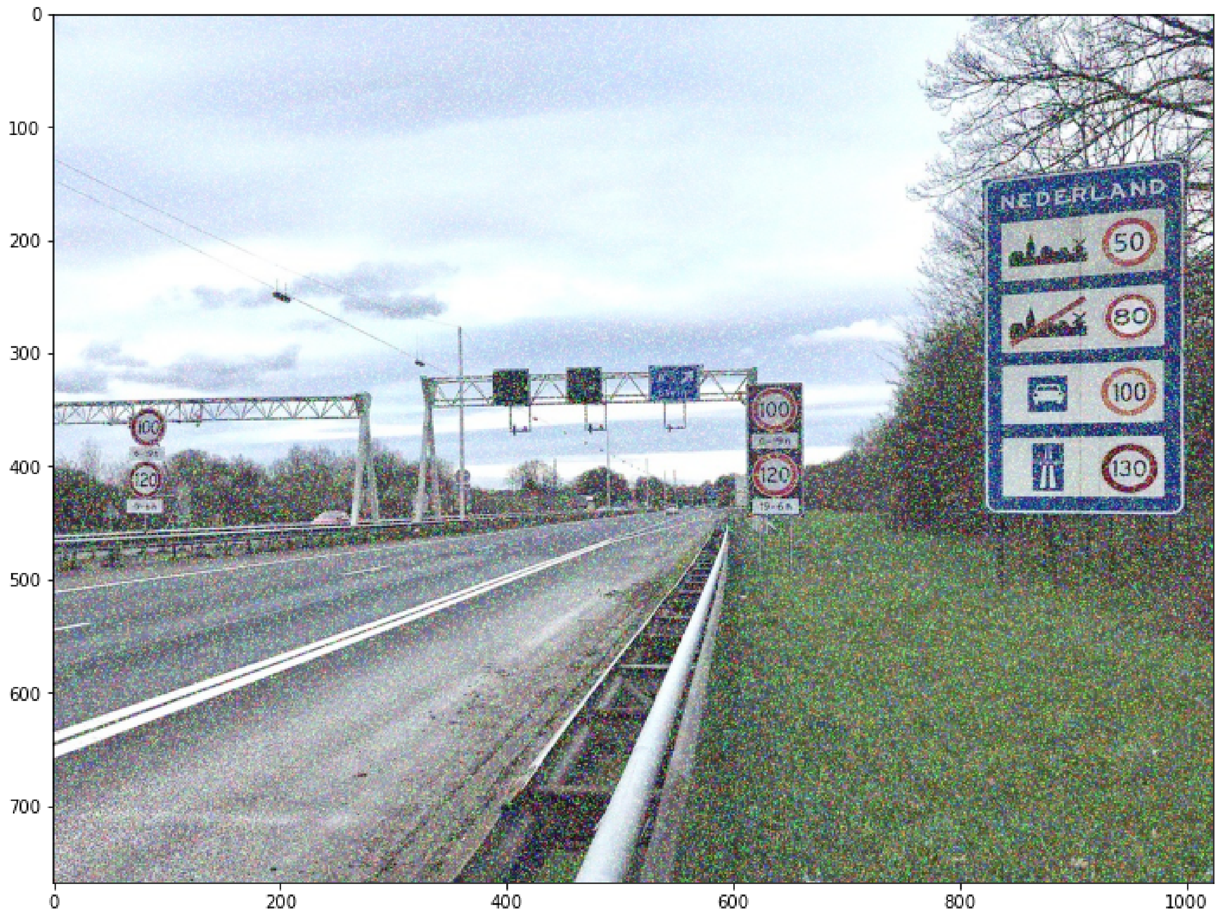
In [3]:

```
def display(img1,cmap="gray"):
    fig=plt.figure(figsize=(12,18))
    ax=fig.add_subplot()
    ax.imshow(img1,cmap="gray")
```

In [4]:

```
path="/Users/mehradhq/Downloads/drive-download-20220511T112708Z-001/22.jpeg"
img=load(path)
print ("this is our initial image")
display(img)
```

```
this is our initial image
```

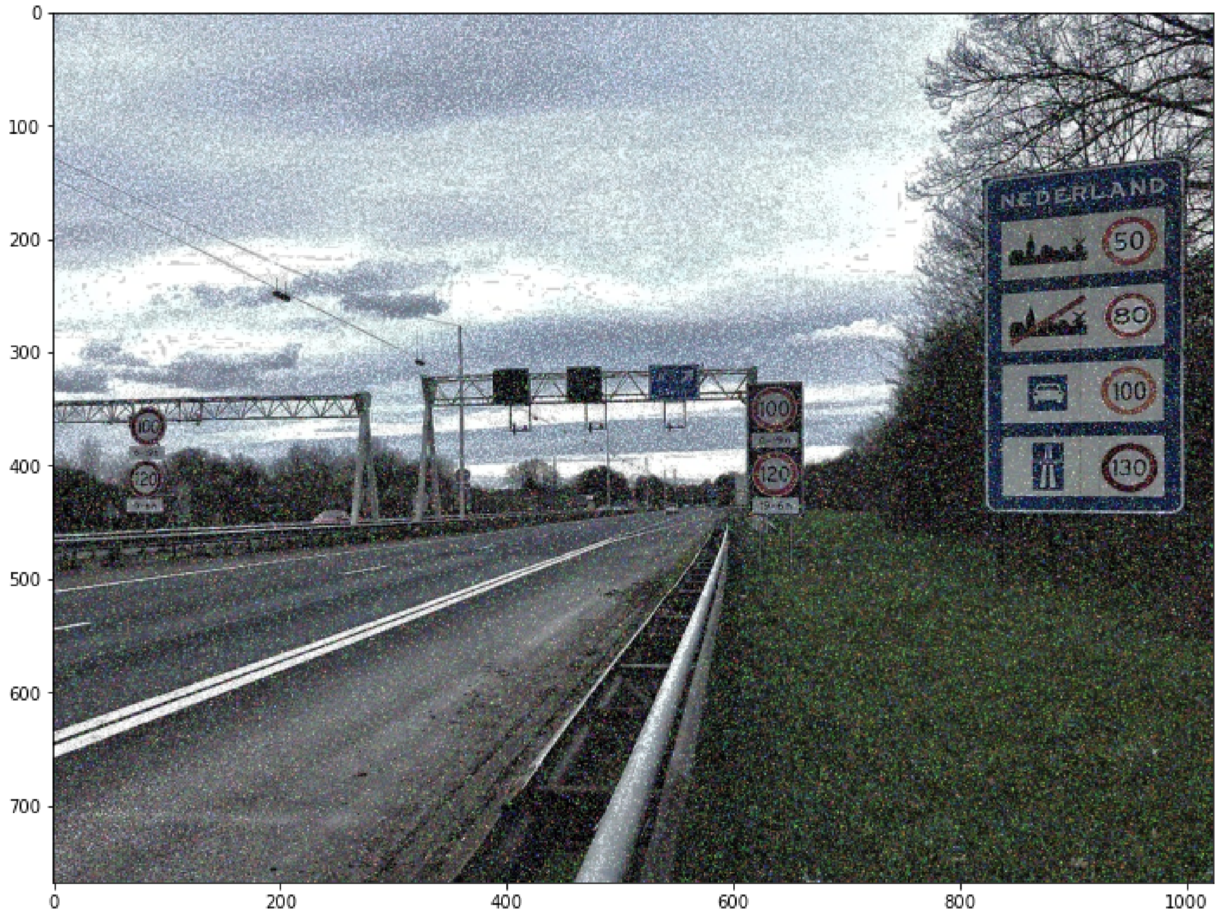


In [5]:

```
#for histogram equalization of an RGB image in openCV, we have to convert it to HSV.
HSV=cv.cvtColor(img,cv.COLOR_RGB2HSV)
HSV[:,:,2]=cv.equalizeHist(HSV[:,:,2])
eq_img=cv.cvtColor(HSV,cv.COLOR_HSV2RGB)
```

```
#this is the histogram equalization of our initial image
print ("this is the histogram equalization of our initial image")
display(eq_img)
```

```
this is the histogram equalization of our initial image
```



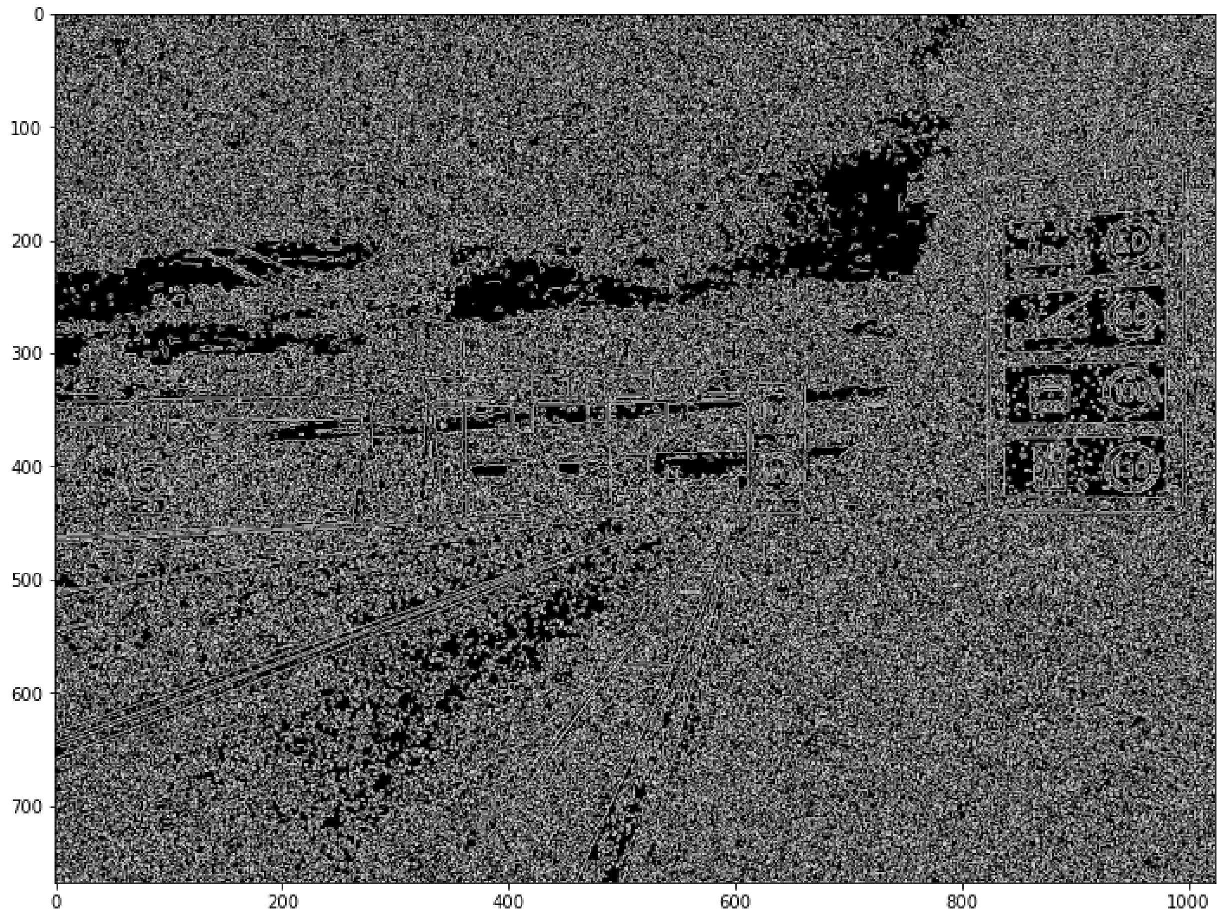
In [6]:

```
#for canny edge operator in opencv we have to define an upper and a lower threshold bound. Normally the default 
#finding out the median pixel value in the image. Then multiplying it by 0.7 and 1.3, these are your lower and upper
#be lower than 0 and the higher threshold is maximum 255.
med_val=np.median(img)
lower=int(max(0,0.7*med_val))
upper=int(min(255,1.3*med_val))

canny_eq_img=cv.Canny(eq_img,lower,upper)
canny_img=cv.Canny(img,lower,upper)
```

In [7]:

```
display(canny_eq_img)
```



In [8]:

```
display(canny_img)
```