Project1 Document

Weicheng Dai

1. The dependencies

The important dependencies are listed below:

1.1. Jupyter notebook version 6.4.4

1.2. Python version 3.9.7.

1.3. NumPy version 1.21.2

1.4. matplotlib version 3.4.3

1.5. PIL version 8.3.2

2. Instructions on using the script

There are indexes above each block. In figure 1, we can see the block that imports useful packages, the block that implements gaussian smoothing to given images.

1. The useful packages

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

2. ATTENTION! If this is not executed in colab, this cell doesn't need to be run ¶

In [ ]:
```
from google.colab.patches import cv2_imshow
from google.colab import drive
drive.mount('/content/drive')
%cd drive/MyDrive/computerVis
```

3. This is the function which apply gaussian smoothing to the given image

In [2]:
```
def gaussian_smoothing(img):
  """ apply gaussian filter to the given image

  Parameters
  ----------
  img: {ndarray} shape = [row * col]
       The img that is given, Which is denoted with an ndarray,
       for example 128*128 means 128 rows by 128 columns.

  Returns
```

Figure 1: Example of indexes

2.1 cells from #1 to #7

The cells from #1 to #7 are definitions and pre-works. Users should click each cell and execute them one by one, **EXCEPT** for cell #2, which doesn't need to be run unless it is in Colab. Basically, the functions of each cell are:

cell#1: import packages

cell#2: import packages that will be used in google Colab

cell#3: the function that applies gaussian smoothing

cell#4: the function that computes gradients of both horizontal and vertical direction

cell#5: the function that computes gradient magnitude and gradient angle

cell#6: the function that applies non-maxima suppression

cell#7: the function that implements p-tile algorithm

2.2 cell from #8 to #9

The cells that apply the functions defined above to a given image. If the user wants to apply the operations on another image, **PLEASE** change the 'FILE' variable in cell #8.1. Please look at Figure 2. Users should follow the sequence and execute each cell. Most of the cells would directly show the results. Cell #9 enables user to save the results after each operation. **PLEASE** change the file names you want to save in order not to overwrite the other results (Figure 3).

8.1 Read file and apply gaussian smoothing

```
In [40]: # Please be noted that in this experiment, all of the results are float numbers, so to show the image we have to convert into int
         # if it is another file, please change the 'FILE' variable
         # FILE = "House.bmp"
         FILE = "Test patterns.bmp"

         # open the image and convert it to grayscale, then convert to numpy array
         img = Image.open(FILE).convert('L')
         img = np.asarray(img)
```

Figure 2: modify cell#8.1 to choose another file

9. save the above images

In [47]:
```
# The reasons that some of the variables are modified are the same as in their corresponding cells.
plt.imsave("test_after_gaussian.png", after_gaussian.astype(np.int16), cmap = 'gray')
plt.imsave("test_horizontal_grad.png", (np.absolute(grad_hori)/3).astype(np.int16), cmap = 'gray')
plt.imsave("test_vertical_grad.png", (np.absolute(grad_vert)/3).astype(np.int16), cmap = 'gray')
plt.imsave("test_gradient.png", (gradient.astype(np.int16))/np.max(gradient.astype(np.int16)) *255, cmap = 'gray')
plt.imsave("test_after_suppression.png", magnitude.astype(np.int16), cmap = 'gray')
plt.imsave("test_edge_25.png", edge_25.astype(np.int16), cmap = 'gray')
plt.imsave("test_edge_50.png", edge_50.astype(np.int16), cmap = 'gray')
plt.imsave("test_edge_75.png", edge_75.astype(np.int16), cmap = 'gray')
```

Figure 3: modify cell#9 to choose other names