Lecture Notes – PGM

Bitmap vs Vector Graphics read for general knowledge only

Image Compression Techniques again, read for general knowledge only

There are different bitmap file formats, I have selected PGM file format.

PGM File Format

The **portable gray map** or **PGM** is a very simple raster image file format, perhaps the simplest possible. It is useful for low-level image manipulation and conversion. The important point is PGM is available in both binary and ASCII formats. ASCII format is human readable that is the reason for selection of this file format.

Here, is a sample PGM file, on the right hand side, you can see this beautiful flower. However, if you open this file in notepad, you can see some meta-data on the top.

For example, the first line has "P2" that is file type. The next line has a message "# created by IranView". The third line has image width and height. Fourth line has a number "255" that is the maximum color value, for any pixel. From fifth line, we have pixel color values that is shades of gray. Here, we have only few values, there will be 144000 values for this image.

```
P2
# Created by IrfanView
480 300
255
220 220 220 220 219 220 220 220 222 222 222 223 224 225 226 226 226 227 227 227 229 229
229 229 230 230 231 232 233 234 234 234 235 236 236 236 237 237 237 237 238 238 238 238
238 238 238 239 240 240 239 239 239 239 239 239 239 239 239 239 238 238 238 238 239 239
239 240 240 240 241 241 241 241 241 242 241 241 243 243 241 241 241 241 241 241 241 241
242 243 242 242 242 242 242 242 242 242 242 242 241 241 242 242 243 243 243 243 243 243
243 243 243 243 244 244 244 244
```

Reading PGM File in Python

```
def main():
    im1 = open('img1.pgm','r')
    signature = im1.readline()
    message = im1.readline()
    width, height = map(int, im1.readline().split())
    clrs = im1.readline()
    image = list(map(int, im1.read().split()))
    print ('Width: ', width, '\tHeight:', height)
    print ('Number of Pixels:', len(image))
    im1.close()
```

Output:

```
Width:  480       Height: 300

Number of Pixels: 144000
```

The output confirms that we have successfully read the image file. If we multiply 480 x 300, we will get 144000, number of pixels.

Image Operation 1

We can perform many simple operations on image. For example, I have divided all color values by 4 and I got the following image in result, which is darker than previous image. The simple operation is:

```
        for i in range(width * height):
            image[i] = image[i] // 4
```

After writing the modified data in file. We have following image, definitely much darker and color values are around 50-60; whereas previously, we have values around 220-230.

```
P2
# Created by IrfanView
480 300
255
55 55 55 55 54 55 55 55 55 55 55 55 56 56 56 56 56 56
56 56 57 57 57 57 57 57 57 58 58 58 58 58 58 59 59 59
59 59 59 59 59 59 59 59 59 59 59 59 60 60 59 59 59 59
59 59 59 59 59 59 59 59 59 59 59 59 59 60 60 60 60 60
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60 60 60 60 60 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
```

Writing PGM File in Python

Here, we are assuming that file is read using the previous code:

```
    im2 = open('img2.pgm','w')
    im2.write(signature)
    im2.write(message)
    im2.write(f'{width} {height}\n')
    im2.write(clrs)
    im2.write(str(image))
    im2.close()
```

Here, we are writing the same information already read in reading PGM into another file. Thus, this code will create a copy of the same image, if there is no change in the image.

Here, we have a complete, code to read and write image, where values are modified and in result, we have a darker image.

```
def main():
    im1 = open('img1.pgm','r')
    im2 = open('img2.pgm','w')
    signature = im1.readline()
    im2.write(signature)
    message = im1.readline()
    im2.write(message)
    width, height = map(int,
im1.readline().split())
    im2.write(f'{width} {height}\n')
    clrs = im1.readline()
    im2.write(clrs)
    image = list(map(int, im1.read().split()))
    for i in range(width * height):
        image[i] = image[i] // 4
    im2.write(str(image))
    im1.close()
    im2.close()

main()
```

Image Operation 2

We have to put a black strip horizontally in the image. For this we have to identify appropriate rows and set color value to 0, here is the simple code and the resultant image.

```
    for i in range(width * 50, width * 55):
      image[i] = 0
```

We have started loop from 51st row and we have ended on 54th row. n between we have width * 5 pixels having colors zero. In result, we can see black strip in the image.