TUGAS

IMAGE ENHANCEMENT DENGAN METODE *IMAGE NEGATIVES* DAN *POWER LAW GREY LEVEL*

Oleh:

ALMAS FAUZIA WIBAWA

17/409427/PA/17734

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVESITAS GADJAH MADA

YOGYAKARTA

2019

BAB I

PENDAHULUAN

1. Metode *Image Negatives*

Image Negatives merupakan metode *image enhancement* yang paling sederhana. Setiap piksel pada *image* yang dikenai *image negatives*, akan diubah nilai intensitasnya dengan rumus berikut:

$$s = T(r) = (L - 1) - r$$

L-1 yang dimaksud adalah nilai intensitas maksimal yang mungkin ada di piksel. Apabila gambar 8-bit, maka intensitas maksimalnya adalah $2^8 - 1 = 255$. Sedangkan, s disitu melambangkan nilai intensitas yang baru, dan r melambangkan nilai intensitas sebelumnya. Fungsi T merupakan fungsi *enhancement*. Sehingga, apabila sebuah gambar 8-bit akan dikenai *enhancement* dengan metode *image negatives*, ia dapat dikenai rumus berikut:

$$s = T(r) = 255 - r$$

2. Metode *Power Law Gray Level*

Power Law Gray Level merupakan metode *enhancement* yang sedikit lebih kompleks. Intensitas setiap piksel pada citra yang akan dikenai metode ini, diubah dengan rumus berikut:

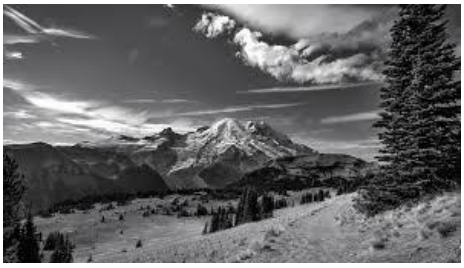
$$s = T(r) = cr^{\gamma}$$

Huruf s dan r, sama seperti sebelumnya, melambangkan intensitas citra setelah dan sebelum *enhancement*. Sedangkah, c disitu melambangkan sebuah konstanta dan γ adalah gamma, melambangkan sebuah nilai yang menjadi perpangkatan pada transformasi ini.

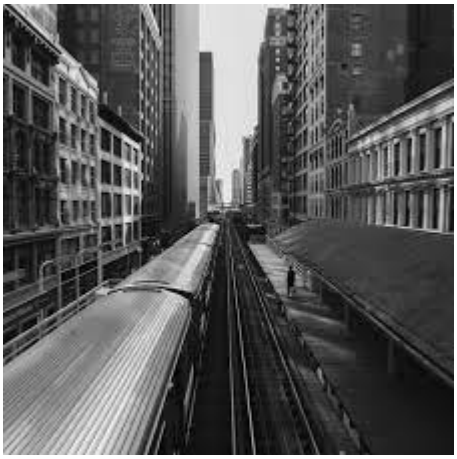
BAB II
PEMBAHASAN

Pada eksperimen kali ini, akan dicoba dilakukan *image enhancement* dengan metode *image negatives* dan *power law gray level* dengan nilai gamma di bawah 1 dan di atas 1. Kedua metode tersebut akan dikenai pada tiga citra di bawah ini:

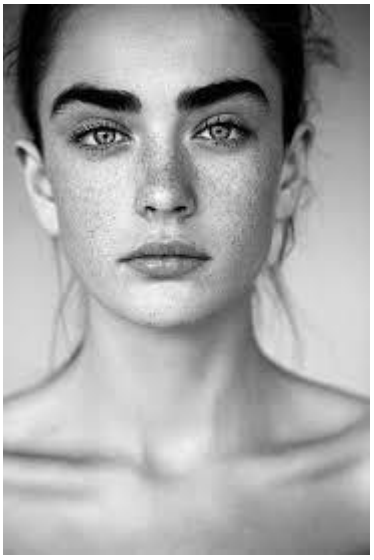
(1)



(3)



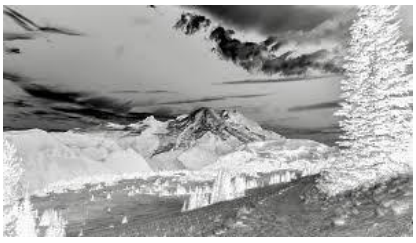
(2)

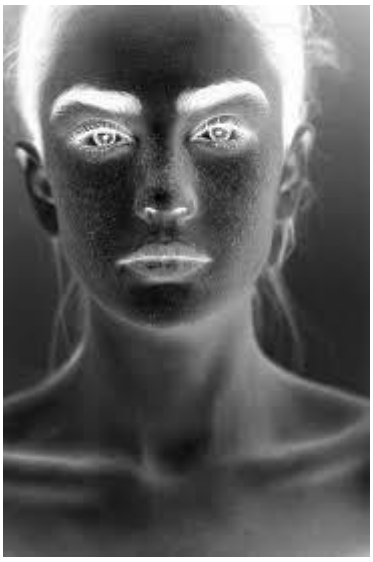


1. *Image Negatives*

Setelah dikenai *image negatives*, ketiga citra tersebut menjadi:

(1)



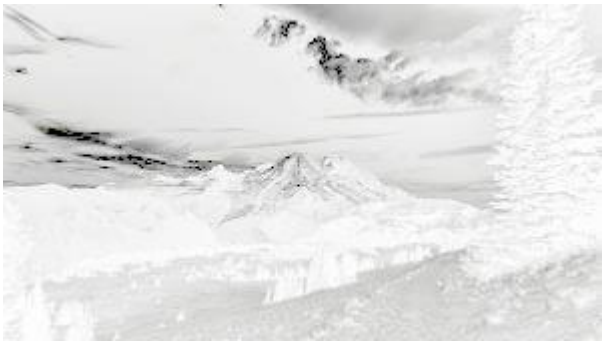
(2) 

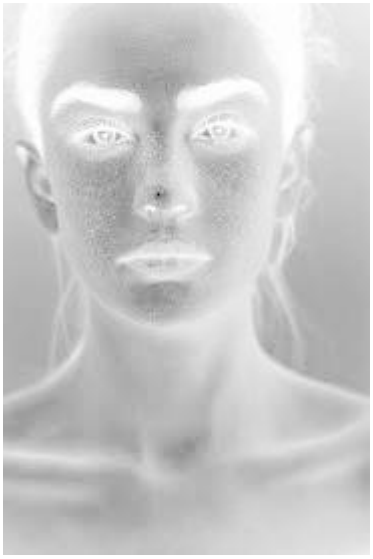
(3) 

Apa yang coba direpresentasikan dari rumus metode ini adalah bahwa *image negatives* akan membalik nilai intensitas setiap piksel pada citra yang akan dikenainya. Contoh, piksel yang semula berwarna sangat putih akan menjadi sangat hitam dan sebaliknya. Hal ini jelas terlihat pada ketiga citra di atas. Piksel yang awalnya berwarna gelap, berubah menjadi terang, dan sebaliknya. Intensitas kegelapannya pun tetap. Hanya saja, pada citra hasil, intensitas kegelapan tersebut berbalik menjadi intensitas keterangan.

2. *Power Law Gray Level*

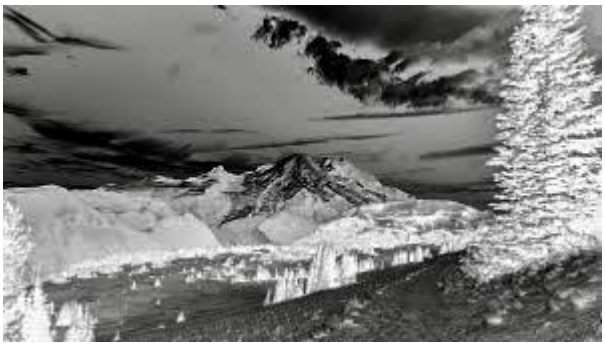
Dalam eksperimen ini, digunakan nilai konstanta sebesar 70. Pada percobaan pertama, gamma yang digunakan berada di bawah 1. Lebih tepatnya, diberi nilai 0.2. Hasil dari citra yang dikenai metode *Power Law Gray Level* dengan nilai konstanta 70 dan gamma 0.2 adalah sebagai berikut:

(1) 

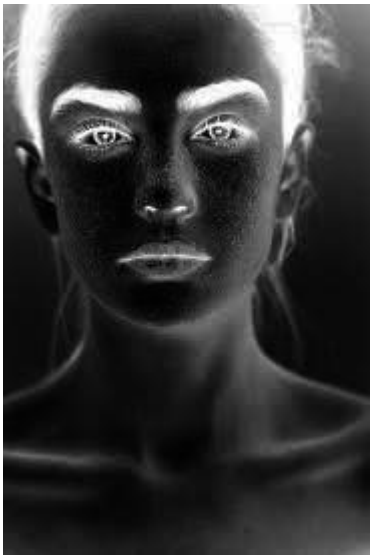
(2) 

(3) 

Selanjutnya, dilakukan eksperimen dengan mengganti nilai gamma menjadi di atas 1. Lebih tepatnya, diberikan nilai 1.7 pada gamma. Setelah dikenai metode *power law gray level* dengan konstanta 70 dan gamma 1.7, ketiga citra berubah menjadi:

(1) 

(3) 

(2) 

Dari kedua hasil di atas, dapat kita perhatikan dampak dari dinaikkannya nilai gamma. Ketika gamma dinaikkan nilainya, kontras pada hasil citra akan berkurang. Walaupun begitu, detail dari citra tersebut semakin baik.

BAB III

LAMPIRAN

1. Kode Program *Image Negatives*

```
35      public static void imageNegative(BufferedImage img) {
36          File file;
37
38          //get image's width and height
39          int width = img.getWidth();
40          int height = img.getHeight();
41
42          //change the pixel to its negatives
43          int pixel, address, red, green, blue;
44          for (int y = 0; y < height; y++) {
45              for (int x = 0; x < width; x++) {
46                  pixel = img.getRGB(x, y);
47                  address = (pixel >> 24)&0xff;
48                  red = (pixel >> 16)&0xff;
49                  green = (pixel >> 8)&0xff;
50                  blue = pixel&0xff;
51
52                  red = 255 - red;
53                  green = 255 - green;
54                  blue = 255 - blue;
55
56                  pixel = (address << 24) | (red << 16) | (green << 8) | blue;
57                  img.setRGB(x, y, pixel);
58              }
59          }
60
61          //make the output file
62          try {
63              file = new File("D:\\Kuliah\\Semester 5\\Tugas\\"
64                      + "Pengolahan Citra Digital (1)\\ImageTransformation\\"
65                      + "outputImageNegative.jpg");
66              ImageIO.write(img, "jpg", file);
67          } catch(IOException e) {
68              System.out.println(e);
69          }
70      }
```

2. Kode Program *Power Law Gray Level*

```
72      public static void powerLaw(BufferedImage img, double gamma, int c) {
73          File file;
74
75          //get image's width and height
76          int width = img.getWidth();
77          int height = img.getHeight();
78
79          int pixel, alpha, red, green, blue;
80          //get minimum and maximum value
81          double redMax = -1, greenMax = -1, blueMax = -1;
82          int redMin = 999, greenMin = 999, blueMin = 999;
83          for (int y = 0; y < height; y++) {
84              for (int x = 0; x < width; x++) {
85                  pixel = img.getRGB(x, y);
                    alpha = (pixel >> 24)&0xff;
87                  red = (pixel >> 16)&0xff;
88                  green = (pixel >> 8)&0xff;
89                  blue = pixel&0xff;
90
91                  red = (int) (c * pow(red, gamma));
92                  green = (int) (c * pow(green, gamma));
93                  blue = (int) (c * pow(blue, gamma));
94
95                  if (red < redMin) {
96                      redMin = red;
97                  }
98                  else if (red > redMax) {
99                      redMax = red;
100                 }
101                 if (green < greenMin) {
102                     greenMin = green;
103                 }
104                 else if (green > greenMax) {
105                     greenMax = green;
106                 }
107                 if (blue < blueMin) {
108                     blueMin = blue;
109                 }
110                 else if (blue > blueMax) {
111                     blueMax = blue;
112                 }
113             }
114         }
115
```

```
116             //change the pixel with the power law formula
117             for (int y = 0; y < height; y++) {
118                 for (int x = 0; x < width; x++) {
119                     pixel = img.getRGB(x, y);
120                     alpha = (pixel >> 24)&0xff;
121                     red = (pixel >> 16)&0xff;
122                     green = (pixel >> 8)&0xff;
123                     blue = pixel&0xff;
124 
125                     red = (int) (c * pow(red, gamma));
126                     green = (int) (c * pow(green, gamma));
127                     blue = (int) (c * pow(blue, gamma));
128 
129                     red = (int) ((red-redMin)/(redMax-redMin) * 255);
130                     green = (int) ((green-greenMin)/(greenMax-greenMin) * 255);
131                     blue = (int) ((blue-blueMin)/(blueMax-blueMin) * 255);
132 
133                     pixel = (alpha << 24) | (red << 16) | (green << 8) | blue;
134                     img.setRGB(x, y, pixel);
135                 }
136             }
137 
138 
139             //make the output file
140             try {
141                 file = new File("D:\\Kuliah\\Semester 5\\Tugas\\"
142                         + "Pengolahan Citra Digital (1)\\ImageTransformation\\"
143                         + "outputPowerLaw.jpg");
144                 ImageIO.write(img, "jpg", file);
145             } catch(IOException e) {
146                 System.out.println(e);
147             }
148         }
```