Relatório: Realce de imagens usando o Laplaciano e Filtragem Espacial

Davi de Lima Cruz
PROJETO 02-2024.2: Grupo 1

11 de dezembro de 2024

Resumo

Este relatório apresenta a aplicação de técnicas de realce de imagem utilizando o filtro Laplaciano e o realce de High-Boost. A biblioteca libtiff foi utilizada para leitura e escrita de imagens no formato TIFF. Os resultados foram analisados por meio de comparações visuais e histogramas, demonstrando a eficácia das técnicas aplicadas.

1 Discussão Técnica

1.1 Biblioteca libtiff e Histograma

Para a leitura e escrita de imagens, foi usada a biblioteca libtiff que pode ser instalada no linux com o comando `sudo apt-get install libtiff-dev`. Para compilar o programa, foi utilizado o seguinte comando:

```
gcc -o app main.c -ltiff
```

O programa aceita dois argumentos:

```
./app <command> <filename> [<k>]
```

onde:

- `<command>` pode ser `hist`, `laplacian` ou `high-boost`.
- `<filename>` é o nome do arquivo TIFF (sem a extensão .tif).
- `<k>` é um parâmetro usado nos comandos `laplacian` e `high-boost` para definir a constante k.

1.2 Filtro Laplaciano

O filtro Laplaciano é um operador de detecção de bordas que realça as regiões de alta frequência na imagem. Ele é útil para destacar as bordas e os detalhes finos na imagem. O programa aplica o filtro Laplaciano em uma uma mascara 3x3 e, em seguida, combina a imagem original com a imagem filtrada usando uma constante ajustável. Para essa aplicação, foi utilizado o seguinte kernel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

e a constante foi definida como -1.

1.3 Realce de High-Boost

O realce de high-boost é uma técnica de realce de imagem que realça as regiões de alta frequência na imagem. Ele é útil para realçar os detalhes finos e melhorar a nitidez da imagem. O programa aplica o realce de high-boost em uma imagem usando a seguinte fórmula:

$$f_{\text{realcada}}(x, y) = f_{\text{original}}(x, y) + k \times (f_{\text{original}}(x, y) - f_{\text{suavizada}}(x, y))$$

onde k é um parâmetro ajustável que controla a intensidade do realce. Para essa aplicação, a constante foi definida como 4,5. Para a imagem suavizada, foi utilizado um filtro gaussiano com kernel 3x3 e desvio padrão 5.

2 Discussão dos Resultados

Os histogramas das imagens 3-33 e 3-38 do livro não são muito didáticos para explicar as diferenças entre as imagens originais e as imagens com filtro Laplaciano e realce de High-Boost. Por isso, foi utiliza a imagem da praia para ilustrar as diferenças entre as imagens originais e as imagens com filtro Laplaciano e realce de High-Boost.

(a) Imagem Original (b) Filtro Laplaciano (c) Realce de High-Boost

Figura 1: Comparação entre a imagem original, a imagem com filtro Laplaciano e a imagem com realce de High-Boost.

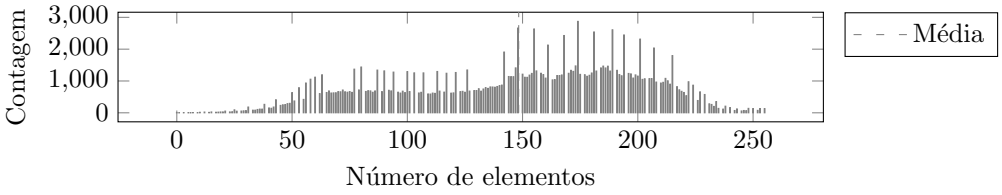


Figura 2: Histograma Imagem original.

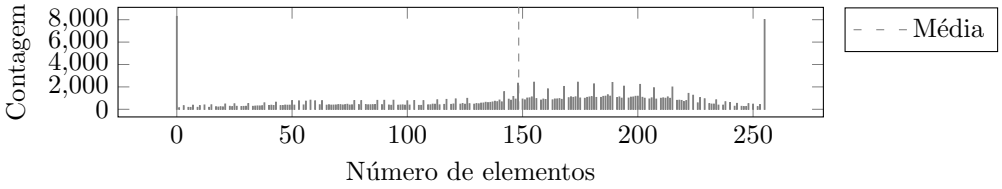


Figura 3: Histograma filtro Laplaciano.

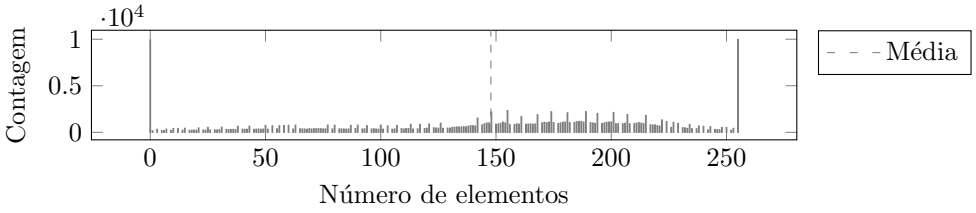


Figura 4: Histograma realce de High-Boost.

Os resultados apresentados nas Figuras 1, 2, 3 e 4 mostram a comparação entre a imagem original, a imagem com filtro Laplaciano e a imagem com realce de High-Boost, bem como seus respectivos histogramas.

Na Figura 1, podemos observar que o filtro Laplaciano realça as bordas da imagem, destacando os detalhes finos. Já o realce de High-Boost aumenta a nitidez da imagem, tornando os detalhes mais visíveis.

Os histogramas das Figuras 2, 3 e 4 mostram a distribuição dos níveis de cinza nas imagens. O histograma da imagem original (Figura 2) apresenta uma distribuição mais uniforme, enquanto o histograma da imagem com filtro Laplaciano (Figura 3) mostra um aumento na frequência dos níveis de cinza correspondentes às bordas. O histograma da imagem com realce de High-Boost (Figura 4) apresenta picos mais acentuados, indicando um aumento no contraste da imagem.

(a) Imagem Original 3-38 (b) Filtro Laplaciano 3-38 (c) High-Boost 3-38

Figura 5: Comparação entre a imagem original, a imagem com filtro Laplaciano e a imagem com realce de High-Boost 3-38.

As imagens 3-33 e 3-38 têm histogramas semelhantes, com uma distribuição nenhum pouco uniforme. Tornando a analise visual mais difícil. No entanto, segue os histogramas das imagens 3-33 e 3-38.

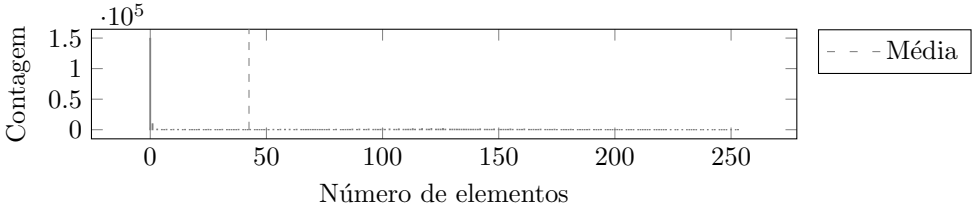


Figura 6: Histograma Imagem original 3-38.

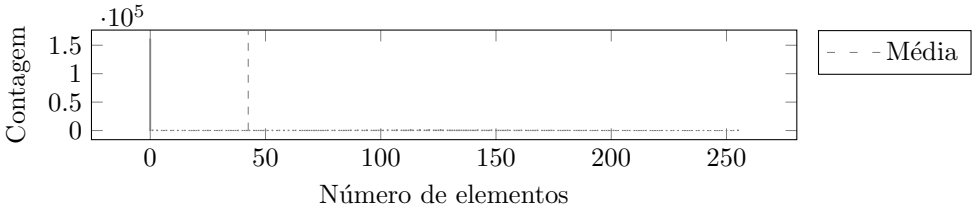


Figura 7: Histograma filtro Laplaciano 3-38.

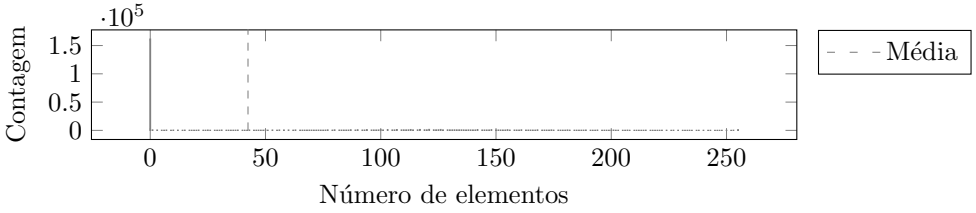
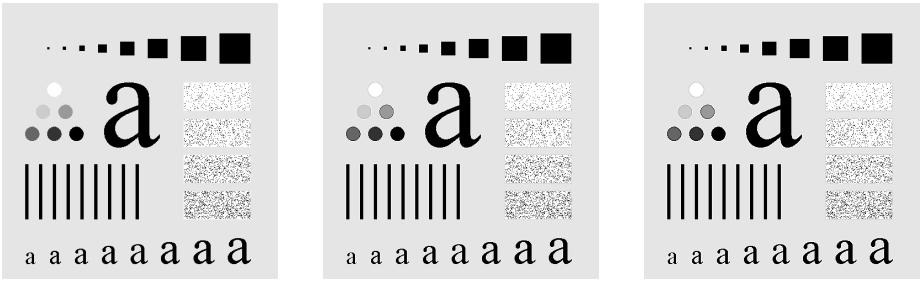


Figura 8: Histograma realce de High-Boost 3-38.



(a) Imagem Original 3-33 (b) Filtro Laplaciano 3-33 (c) High-Boost 3-33

Figura 9: Comparação entre a imagem original, a imagem com filtro Laplaciano e a imagem com realce de High-Boost 3-33.

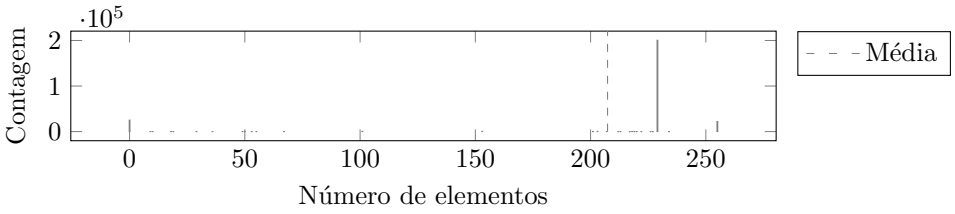


Figura 10: Histograma Imagem original 3-33.

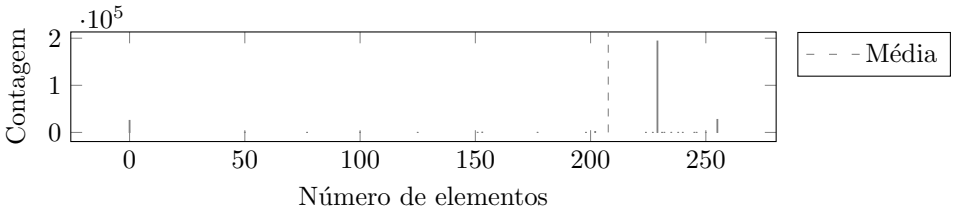


Figura 11: Histograma filtro Laplaciano 3-33.

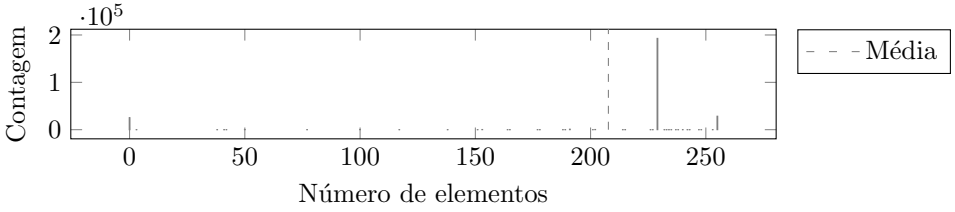


Figura 12: Histograma realce de High-Boost 3-33.

A outra foto utilizada foi a da mulher, que apresenta um histograma mais uniforme, facilitando a análise visual. Tendo a mesma explicação do histograma anterior, onde o filtro Laplaciano realça as bordas da imagem, destacando os detalhes finos. Já o realce de High-Boost aumenta a nitidez da imagem, tornando os detalhes mais visíveis. Basicamente, o histograma vai mais para os extremos, mostrando que a imagem foi realçada.

(a) Imagem Original Mulher

(b) Filtro Laplaciano Mulher

(c) High-Boost Mulher

Figura 13: Comparação entre a imagem original, a imagem com filtro Laplaciano e a imagem com realce de High-Boost Mulher.

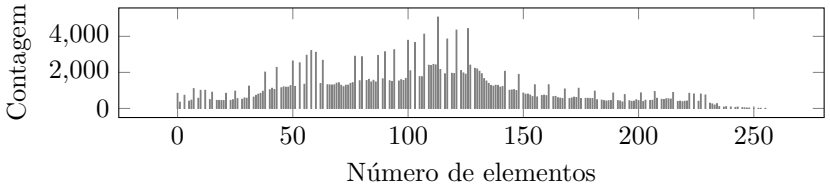


Figura 14: Histograma Imagem original Mulher.

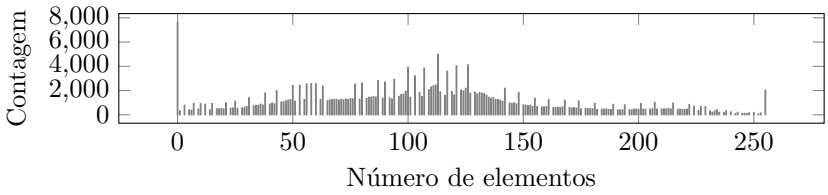


Figura 15: Histograma filtro Laplaciano Mulher.

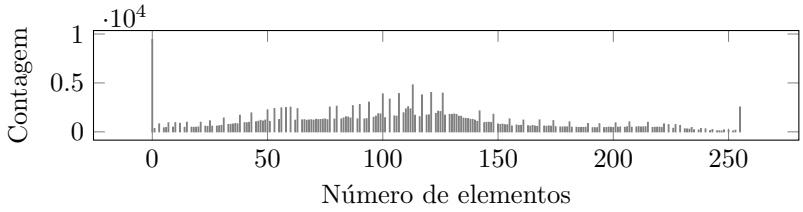


Figura 16: Histograma realce de High-Boost Mulher.

3 Código Fonte em C

Listing 1: Código fonte do programa em C.

```
#include <stdio.h>
#include <stdlib.h>
#include "tiffio.h"
#include <string.h>
#include <math.h>

int discovery_dimensions(const char* filename, uint32_t* w,
    uint32_t* h) {
  char* filenametiff = (char*) malloc(strlen(filename) + 5);
  strcpy(filenametiff, filename);
  strcat(filenametiff, ".tif");
  TIFF* tiff = TIFFOpen(filenametiff, "r");
  TIFFGetField(tiff, TIFFTAG_IMAGEWIDTH, w);
  TIFFGetField(tiff, TIFFTAG_IMAGELENGTH, h);
  TIFFClose(tiff);
  free(filenametiff);
  return 1;
}

int readTiff(const char* filename, uint8_t** pixels) {
  char* filenametiff = (char*) malloc(strlen(filename) + 5);
  strcpy(filenametiff, filename);
  uint32_t w, h;
  TIFF* tiff = TIFFOpen(strcat(filenametiff, ".tif"), "r");
  TIFFGetField(tiff, TIFFTAG_IMAGEWIDTH, &w);
  TIFFGetField(tiff, TIFFTAG_IMAGELENGTH, &h);
  uint32_t npixels = w * h;
```

```
27   *pixels = (uint8_t*) malloc(npixels * sizeof(uint8_t));
28   uint32_t* raster = (uint32_t*) _TIFFmalloc(npixels * sizeof(
         uint32_t));
29   TIFFReadRGBAImage(tiff, w, h, raster, 0);
30   for (uint32_t row = 0; row < h; row++) {
31     for (uint32_t col = 0; col < w; col++) {
32       uint32_t pixel = raster[row * w + col];
33       uint8_t r = TIFFGetR(pixel);
34       uint8_t g = TIFFGetG(pixel);
35       uint8_t b = TIFFGetB(pixel);
36       (*pixels)[row * w + col] = (uint8_t)(0.299 * r + 0.587 * g +
             0.114 * b);
37     }
38   }
39   _TIFFfree(raster);
40   TIFFClose(tiff);
41   free(filenametiff);
42   return 1;
43 }
44
45 int writeTiff(char* filename, uint32_t w, uint32_t h, uint8_t*
       pixels) {
46   char* filenametiff = (char*) malloc(strlen(filename) + 5);
47   strcpy(filenametiff, filename);
48
49   TIFF* tiff = TIFFOpen(strcat(filenametiff, ".tif"), "w");
50   TIFFSetField(tiff, TIFFTAG_IMAGEWIDTH, w);
51   TIFFSetField(tiff, TIFFTAG_IMAGELENGTH, h);
52   TIFFSetField(tiff, TIFFTAG_BITSPERSAMPLE, 8);
53   TIFFSetField(tiff, TIFFTAG_SAMPLESPERPIXEL, 1);
54   TIFFSetField(tiff, TIFFTAG_PHOTOMETRIC, PHOTOMETRIC_MINISBLACK);
55   TIFFSetField(tiff, TIFFTAG_ORIENTATION, ORIENTATION_BOTLEFT);
56   TIFFSetField(tiff, TIFFTAG_RESOLUTIONUNIT, RESUNIT_INCH);
57   for (uint32_t row = 0; row < h; row++) {
58     TIFFWriteScanline(tiff, &pixels[row * w], row, 0);
59   }
60   TIFFClose(tiff);
61   free(filenametiff);
62   return 1;
63 }
64
65 void histogram(const char* filename, uint8_t* pixels, uint32_t w,
       uint32_t h) {
66   uint32_t hist[256];
67   for (uint32_t i = 0; i < 256; i++) hist[i] = 0;
68   for (uint32_t row = 0; row < h; row++) {
69     for (uint32_t col = 0; col < w; col++) {
70       hist[pixels[row * w + col]]++;
71     }
72   }
73   char* filenamedat = (char*) malloc(strlen(filename) + 5);
74   strcpy(filenamedat, filename);
75   FILE *f = fopen(strcat(filenamedat, ".dat"), "w");
76   for (uint32_t i = 0; i < 256; i++) {
77     if (hist[i] > 0)
78     fprintf(f, "%d %d\n", i, hist[i]);
79   }
```

```
  fclose(f);
  free(filenamedat);

  char *filenamemean = (char*) malloc(strlen(filename) + 9);
  strcpy(filenamemean, filename);
  FILE *fmean = fopen(strcat(filenamemean, "_mean.dat"), "w");
  uint32_t sum = 0;
  uint32_t count = 0;
  for (uint32_t i = 0; i < 256; i++) {
    sum += i * hist[i];
    count += hist[i];
  }
  fprintf(fmean, "%f\n", (float)sum / count);
  fclose(fmean);
  free(filenamemean);

  char *filenamecount = (char*) malloc(strlen(filename) + 11);
  strcpy(filenamecount, filename);
  FILE *fcount = fopen(strcat(filenamecount, "_count.dat"), "w");
  fprintf(fcount, "%d\n", count);
  fclose(fcount);
  free(filenamecount);
}

void start_histogram(char* filename) {
  uint32_t w, h;
  uint8_t* pixels;
  discovery_dimensions(filename, &w, &h);
  pixels = (uint8_t*) malloc(w * h * sizeof(uint8_t));
  readTiff(filename, &pixels);
  histogram(filename, pixels, w, h);
  free(pixels);
}

void laplacian(uint8_t * pixels, float k, uint32_t w, uint32_t h) {
  int mask[3][3] = {
    {0, 1, 0},
    {1, -4, 1},
    {0, 1, 0}
  };
  uint8_t* pixels2 = (uint8_t*) malloc(w * h * sizeof(uint8_t));
  for (uint32_t row = 1; row < h - 1; row++) {
    for (uint32_t col = 1; col < w - 1; col++) {
      int sum = 0;
      for (int i = -1; i <= 1; i++) {
        for (int j = -1; j <= 1; j++) {
          sum += pixels[(row + i) * w + (col + j)] * mask[i + 1][j
              + 1];
        }
      }
      sum = pixels[row * w + col]+k*sum;
      if (sum < 0) sum = 0;
      if (sum > 255) sum = 255;
      pixels2[row * w + col] = (uint8_t) sum;
    }
  }
  for (uint32_t row = 1; row < h - 1; row++) {
```

```
136     for (uint32_t col = 1; col < w - 1; col++) {
137       pixels[row * w + col] = pixels2[row * w + col];
138     }
139   }
140   free(pixels2);
141 }
142
143 void start_laplacian(char* filename, float k) {
144   uint32_t w, h;
145   uint8_t* pixels;
146   discovery_dimensions(filename, &w, &h);
147   pixels = (uint8_t*) malloc(w * h * sizeof(uint8_t));
148   readTiff(filename, &pixels);
149   laplacian(pixels, k, w, h);
150   char* filenameout = (char*) malloc(strlen(filename) + 5);
151   strcpy(filenameout, filename);
152   strcat(filenameout, "_laplacian");
153   writeTiff(filenameout, w, h, pixels);
154   free(pixels);
155 }
156
157 void high_boost(uint8_t * pixels, float k, uint32_t w, uint32_t h)
      {
158   // gaussian mask
159   float sigma = 5.0;
160   float K = 1.0;
161   float mask[3][3] = {
162     {K*exp(-1/sigma), K*exp(-1/sigma/2), K*exp(-1/sigma)},
163     {K*exp(-1/sigma/2), K, K*exp(-1/sigma/2)},
164     {K*exp(-1/sigma), K*exp(-1/sigma/2), K*exp(-1/sigma)}
165   };
166   float sum_mask = 0;
167   for (int i = 0; i < 3; i++) {
168     for (int j = 0; j < 3; j++) {
169       sum_mask += mask[i][j];
170     }
171   }
172   uint8_t* pixels2 = (uint8_t*) malloc(w * h * sizeof(uint8_t));
173   for (uint32_t row = 1; row < h - 1; row++) {
174     for (uint32_t col = 1; col < w - 1; col++) {
175       double sum = 0;
176       for (int i = -1; i <= 1; i++) {
177         for (int j = -1; j <= 1; j++) {
178           sum +=  pixels[(row + i) * w + (col + j)] * mask[i + 1][j
                + 1] / sum_mask;
179         }
180       }
181       sum = pixels[row*w+col] - sum;
182       sum = pixels[row*w+col] + k*sum;
183       if (sum < 0) sum = 0;
184       if (sum > 255) sum = 255;
185       pixels2[row * w + col] = (uint8_t)sum;
186     }
187   }
188   for (uint32_t row = 1; row < h - 1; row++) {
189     for (uint32_t col = 1; col < w - 1; col++) {
190       pixels[row * w + col] = pixels2[row*w+col];
```

```
    }
  }
  free(pixels2);
}

void start_high_boost(char* filename, float k ) {
  uint32_t w, h;
  uint8_t* pixels;
  discovery_dimensions(filename, &w, &h);
  pixels = (uint8_t*) malloc(w * h * sizeof(uint8_t));
  readTiff(filename, &pixels);
  high_boost(pixels, k, w, h);
  char* filenameout = (char*) malloc(strlen(filename) + 5);
  strcpy(filenameout, filename);
  strcat(filenameout, "_high-boost");
  writeTiff(filenameout, w, h, pixels);
  free(pixels);
}

int main(int argc, char* argv[]) {

  char* command = argv[1];
  char* filename = argv[2];
  if (strcmp("hist",command)==0){
    start_histogram(filename);
  } else if (strcmp("laplacian",command)==0){
    char* karg = argv[3];
    start_laplacian(filename, atof(karg));
  } else if (strcmp("high-boost",command)==0){
    char* karg = argv[3];
    start_high_boost(filename, atof(karg));
  } else {
    printf("Command not found\n");
  }

  return 0;
}
```