



Programação de Software Básico - Trabalho I

03/2023

1 Introdução

O primeiro trabalho da disciplina consiste em implementar um programa que realiza zoom em imagens, com resolução de sub-pixel. Para isso, uma imagem será utilizada como entrada e deverá ser gerada como saída uma imagem com um zoom de três vezes, contendo pixels amplificados compostos apenas por sub-pixels puramente vermelhos, verdes, azuis e pretos.

A imagem de entrada deve ser uma imagem colorida com precisão de 24 bits por pixel (R, G, B), onde cada componente possui um valor de intensidade de 0 a 255. Para a manipulação de imagens, está sendo fornecida uma biblioteca que permite ler e escrever imagens no formato PPM (*Portable Pix Map*).

2 Funcionamento

Para cada pixel da imagem de entrada, após o processo de zoom, será gerada uma matriz de sub-pixels que podem apenas estar ligados em função de sua respectiva cor (vermelho, verde, azul), ou desligados (preto). A matriz utilizada é organizada com um arranjo de três colunas e três linhas, contendo as cores vermelho, verde e azul, nessa ordem, como ilustra o exemplo[A][A]As bordas escuras são apresentadas apenas a título ilustrativo.:

My Research Folder

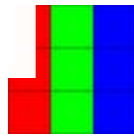


Figura
2.1
Exemplo
de
subpixels

Cada sub-pixel pode apenas estar ligado ou desligado, dessa forma será necessário reduzir a precisão das cores da imagem de entrada e utilizar diferentes padrões de sub-pixel para ter-se quatro níveis de brilho. Para cada pixel da imagem de entrada, devem ser gerados os seguintes padrões:

- Para níveis de cor de 0 a 74, todos os sub-pixels devem ser pretos;
- Para níveis de cor de 75 a 134, o sub-pixel do meio deve ser da cor respectiva (R, G ou B) e os outros dois devem ser pretos;
- Para níveis de cor de 135 a 179, o sub-pixel do meio deve ser preto e os outros dois devem ser da cor respectiva (R, G ou B);
- E para níveis de cor de 180 a 255, todos os sub-pixels devem ser de sua cor respectiva.

2.1 Exemplos

- Um pixel RGB(40, 130, 175) irá produzir o seguinte padrão:

My Research Folder

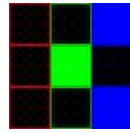


Figura
2.2
Exemplo
de
pixel
completo
(1)

- Um pixel RGB(160, 240, 100) irá produzir o seguinte padrão:

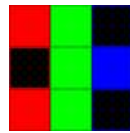


Figura
2.3
Exemplo
de
pixel
completo
(2)

No exemplo a seguir é apresentada uma imagem de entrada e a saída resultante após o processamento. A imagem resultante possui uma dimensão três vezes maior, e a resolução da imagem, apesar de bastante limitada com relação a cores, possui um efeito visual bastante agradável.

My Research Folder

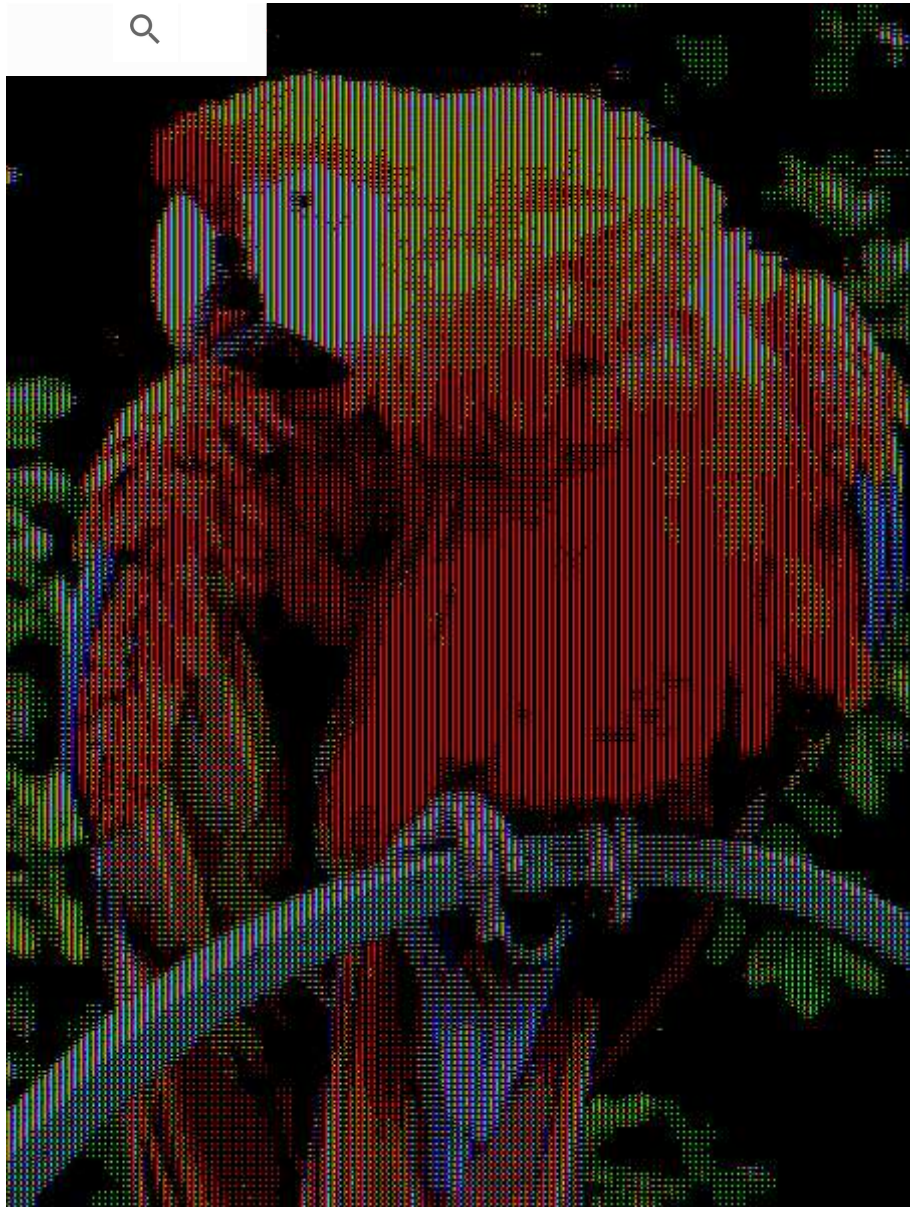


Figura 2.4 Exemplo de imagem original e resultado

2.2 Leitura e escrita de imagens do tipo PPM

Uma imagem pode ser representada por uma matriz de pixels onde cada cor é definida por 3 componentes: vermelho (R), verde (G) e azul (B). Cada uma dessas componentes usualmente é codificada em um byte, o que produz 3 bytes por pixel (24 bits) - ou seja, 16 milhões de possíveis cores. Em outras palavras, as intensidades (R, G, B) variam de 0 a 255, onde 0 é escuro e 255 é claro. Veja abaixo como diversas cores são representadas nesse formato - cada cor está expressa pelas componentes RGB em hexadecimal.

My Research Folder



999999	CCCCCC	CCCC99	9999CC	666699
003333	003399	000066	330066	660066
990000	993300	CC9900	006600	336666
CC0000	CC3300	FFCC00	009900	006666
FF0000	FF3300	FFFF00	00CC00	009999
CC3333	FF6600	FFFF33	00FF00	00CCCC
FF6666	FF6633	FFFF66	66FF66	66CCCC
FF9999	FF9966	FFFF99	99FF99	66FFCC
FFCCCC	FFCC99	FFFFCC	CCFFCC	99FFCC

Figura 2.5 Cores em RGB

O código fornecido define duas *structs*: uma para representar um pixel e outra para representar a imagem inteira. Após a leitura da imagem, os pixels estarão disponíveis a partir do ponteiro *image*.

```

/* lib_ppm.h */
struct pixel_s {
    unsigned char r, g, b;
};

struct image_s {
    int width;
    int height;
    struct pixel_s *pix;
};

int read_ppm(char *file, struct image_s *image);
int write_ppm(char *file, struct image_s *image);
int free_ppm(struct image_s *image);

/* exemplo */
struct image_s data;
struct image_s *image = &data;

r = read_ppm("lena.ppm", image);
...
/* modificando o valor do pixel [20, 50] (posição X, Y) */
image->pix[50 * image->width + 20].r = 255;
image->pix[50 * image->width + 20].g = 255;
image->pix[50 * image->width + 20].b = 255;
...
write_ppm("copy.ppm", image);

```


My Research Folder  

Este trabalho deverá ser realizado em duplas ou trios e apresentado no dia marcado no cronograma (apresentação em torno de 10 minutos). Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle, até a data e hora especificadas, um arquivo *.tar.gz* ou *.zip* do projeto contendo o código fonte desenvolvido.

Observações

- O código deve estar indentado corretamente (qualquer editor ou programador decente faz isso automaticamente).
- A cópia parcial ou completa do trabalho terá como consequência a atribuição de nota ZERO ao trabalho dos alunos envolvidos. A verificação de cópias é feita inclusive entre turmas.
- A cópia de código ou algoritmos existentes da Internet também não é permitida. Se alguma ideia encontrada na rede for utilizada na implementação, a referência deve constar no código em um comentário.

Document generated by [eLyXer 1.2.5 \(2013-03-10\)](#) on 2023-03-23T17:53:21.799229