

Uma linguagem de programação para processamento de imagens

Henrique Miyamoto e Thiago Benites

I. CONTEXTUALIZAÇÃO

Apresentamos uma linguagem de programação voltada para o processamento de imagens digitais. Ela foi implementada com um analisador léxico e utilizando de gramática livre de contexto. Como uma linguagem de programação de propósito específico, seu objetivo é ser intuitiva para o usuário [1]. As funcionalidades que nossa linguagem é capaz de executar, assim como as respectivas sintaxes são apresentadas na Tabela I.

TABLE I

FUNCIONALIDADES E SINTAXE DA LINGUAGEM DE PROGRAMAÇÃO

Funcionalidade	Sintaxe
Salvar uma imagem	destino.jpg = origem.jpg
Alterar brilho	destino.jpg = origem.jpg * float destino.jpg = origem.jpg / float
Detectar valor máximo	[origem.jpg]

Nos processos de salvar uma imagem e de alterar seu brilho, é feita uma cópia do arquivo original. Calculamos o valor de cada pixel como a intensidade na conversão do espaço RGB para o HSI $I = \frac{1}{3}(R + G + B)$ [2] na detecção do valor máximo da imagem.

II. DEMONSTRAÇÃO

Para demonstração das funcionalidades, tomemos por base a imagem em sua forma original apresentada na Figura 1(a). Quando aplicado brilho de fator multiplicativo 2, tem-se o resultado apresentado na Figura 1(b). Já ao dividir pelo mesmo fator, obtém-se o resultado da Figura 1(c).

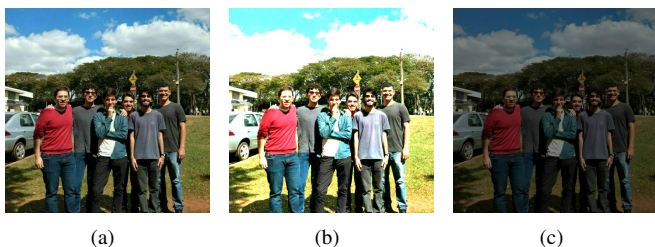


Fig. 1. Imagens: (a) original, (b) com brilho dobrado e (c) reduzido à metade.

Para as três figuras é possível aplicar a função que detecta o máximo valor de brilho, de acordo com a seguinte implementação e resultados:

```
./main
[figura1.jpg]
Intensidade maxima: 254.00
[figura2.jpg]
Intensidade maxima: 255.00
[figura3.jpg]
Intensidade maxima: 133.00
```

III. ANÁLISE

Comparamos a ideia de usar um comando específico para alterar o brilho com aplicações equivalentes de propósito geral, como OpenCV. Em sua versão para C++, a aplicação de brilho a uma imagem pode ser feita através do comando `convertTo()`, com a seguinte sintaxe [3]:

```
void Mat::convertTo(OutputArray m, int rtype,
double alpha=1, double beta=0) const
```

em que α e β são fatores de escala que realizam a operação linear $g(x) = \alpha f(x) + \beta$, sobre a imagem $f(x)$, resultando na imagem $g(x)$ [4].

O comando utilizado na nossa linguagem para tal função é mais simples, no sentido de que tem menos argumentos e lida com menos conceitos de programação que os outros dois, exigindo menos conhecimento do usuário. Por exemplo, no comando do OpenCV, é necessário entender que aplica-se uma transformação linear à imagem, com parâmetros que definem contraste e brilho, além de lidar com a imagem na forma matricial. Na nossa sintaxe, é necessário utilizar apenas os nomes dos arquivos e a intensidade de brilho que se deseja aplicar, ficando oculto para o usuário final o formato como a imagem é tratada e procedimentos realizados.

Se por um lado comandos como o do OpenCV possam gerar maior flexibilidade para usuários avançados, sintaxes mais diretas oferecem ganho em expressividade e facilidade no uso, principalmente quando comparadas à linguagens de uso geral [1].

REFERÊNCIAS

- [1] When and How to Develop Domain-Specific Languages. MERNIK, Marjan; HEERIN, Jan; SLOANE, M Anthony. Disponível em: <https://pdfs.semanticscholar.org/bd06/01088d5f217dc136a898f577763df92891cb.pdf>. Acesso em: 17 set. 2017.
- [2] Wikipedia. RGB color model. Disponível em: https://en.wikipedia.org/wiki/RGB_color_model. Acesso em: 10 set. 2017.
- [3] OpenCV. Basic Structures. Disponível em: http://docs.opencv.org/2.4/modules/core/doc/basic_structures.html. Acesso em: 11 set. 2017.
- [4] OpenCV. Changing the contrast and brightness of an image!. Disponível em: http://docs.opencv.org/2.4/doc/tutorials/core/basic_linear_transform/basic_linear_transform.html. Acesso em: 11 set. 2017.