Instructions for Authors of SBC Conferences Papers and Abstracts

Erick Modesto Campos

[1]Instituto de Ciência Exatas e Naturais (ICEN) – Universidade Federal do Pará (UFPA)
Laboratório de Visualização, Interação e Sistemas Inteligentes (LabVis - UFPA)
Cep 66075110 – Guamá– Belém – Pará – Brazil

erick.c.modesto@gmail.com / erickcampos@ufpa.br

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Este meta-artigo descreve o estilo a ser usado na confecção de artigos e resumos de artigos para publicação nos anais das conferências organizadas pela SBC. É solicitada a escrita de resumo e abstract apenas para os artigos escritos em português. Artigos em inglês deverão apresentar apenas abstract. Nos dois casos, o autor deve tomar cuidado para que o resumo (e o abstract) não ultrapassem 10 linhas cada, sendo que ambos devem estar na primeira página do artigo.*

1. Introdução

O processamento de imagens é um método que executa uma série de operações matemáticas em uma imagem. O intuito é obter um aprimoramento ou extrair algumas informações úteis a cerca da imagem a ser processada. Esse método nada mais é do que um processamento digital de sinais onde a entrada é uma imagem e a saída pode ser uma imagem ou características (*features*) associadas com a determinada imagem processada [Tartu 2019].

A área de processamento de imagens existem diversos algoritmos que são utilizados para diversos propósitos. O reconhocimento de objetos é um bom exemplo de aplicação das técnicas de processamento. No entanto, assim como outros algoritmos, é necessário realizar várias aplicações chamadas de pré-processamento. E geralmente os sistemas que implementam o reconhecimento de objetos em uma imagem, utilizam conversão da imagem original para a escala de cinza como primeira etapa de pré-processamento.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos que utilizaram algum tipo de sistema embarcado para realizar tarefas de processamento digital de imagens. Apesar deste trabalho utilizar o Raspberry Pi, os trabalhos apresentados a seguir utilizam também outras palataformas embarcadas. O objetivo desta seção é mostrar os trabalhos que utilizaram processamento de imagens através de uma abordagem sequencial e paralela.

2.1. Processamento Sequencial

Técnicas de processamento digital de imagens foram aplicadas np trabalho desenvolvido em [Karthik G. and Praburam N. 2016] para detectar doenças em bananeiras através do reconhecimento de padrões em imagens de folhas de bananeiras. O algoritmo alaborado foi executado em uma BeagleBone Black [BeagleBone 2019], uma plataforma embarcado que possui um ARM Cortex-A8 com dois núcleos de processamento. Apesar de ser *dual core* todas as tarefas do algoritmo foram projetadas para serem executadas de forma sequential, visto que não era realizado um processamento em massa, já que bastava uma foto capturada por uma *webcam* no momento desejado para que a detecção da imagem fosse concluída.

O trabalho proposto em [Batista et al. 2017] utilizou algoritmos sequenciais de processamento de imagens para realzar o reconhecimento de gestos da cabeça do usuário e transformá-los em comandos de controle para um televisor. Para isso, foi utilizados as plataformas embarcadas C.H.I.P. [C.H.I.P 2019] e Arduino [Arduino 2019]. O C.H.I.P. é uma placa de baixo custo, assim como o Raspberry Pi, mas que possui um processador *single core* o que inviabilizou o uso de técnicas de processamento paralelo para o reconhecimento de gesros da cabeça do usuário. Já o Arduíno, é uma plataformna que possui um microcontrolador de 8 bits que foi utilizado apenas para o envio de sinais infravermelho de controle para o televisor a ser controlado.

Já em [Li et al. 2009] foi utilizado um FPGA (do inglês *Field-Programmable Gate Array*), são dispositivos semicondutores baseados em uma matriz de blocos lógicos configuráveis (BLC) conectados via interconexões programáveis [Xilinx 2019], para realzar a tarefa de aquisição e também de processamento de imagens. O objetivo do trabalho era realizar tarefas de processamento de imagens utilizando uma plataforma simples e relativamente mais barata. Os algoritmos foram utilizados de forma sequencial, pois as tarefas constistiam basicamente em calculos matemáticos simples baseados nos espaços de cores.

2.2. Processamento Paralelo

O trabalho proposto em [Sivaranjani and Sumathi 2015] utilizou algoritmos de processamento de imagens para extração de características de impressões digitais dos mãos

e dos pés. Para isso, foi utilizado um Raspberry Pi com uma distribuição do Debian modificada para plataformas embarcadas. O algoritmo necessário para o reconhecimento de imagens para extração de recursos biométricos é realizado usando o OpenCV-2.4.9 [OpenCV 2019] — uma biblioteca de código aberto multiplataforma voltado para visão computacional — usando o CMake, g ++, Makefile. Para realzar essa detecção, quatro diferentes algoritmos foram modificados para serem executados de forma paralela com o intuito de aproveitar ao máximo o poder de processamento da plataforma utilizada.

Em [Markovic et al. 2018] foi utilizado um *cluster* contendo dez Raspberry Pi Modelo B foram utilizados com o intuito de aumentar o poder computacional para realizar tarefas de processamento de imagens utilizando um processamento paralelo. Foi aplicado um algoritmo de detecção de bordas de objetos com a intenção de segmentar a imagem em bordas. O experimento conduzido no trabalho utilizou três tarefas de segmentação. A primeira tarefa foi segmentar três imagens, em seguida seis imagens e por fim nove imagens. Cada experimento foi executado utilizando técnicas de processamento paralelo em apenas um arduíndo e também no *cluster*.

3. Aplicações de Pipelining

Esta seção tem como objetivo mostrar a metodogia para implementação do trabalho proposto.

3.1. Forma Sequencial Implementada

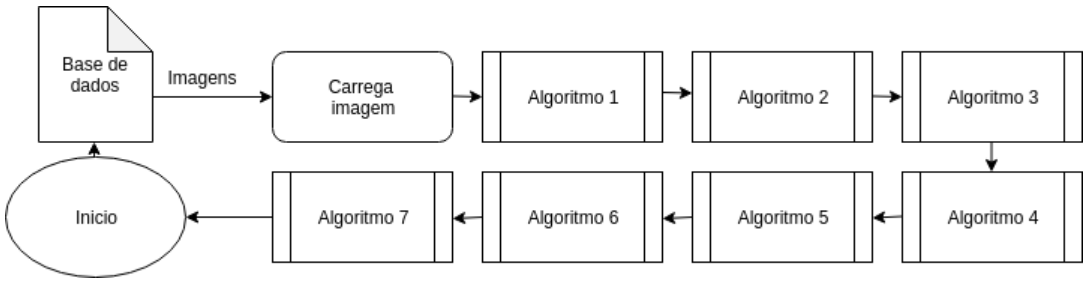

Figura 1. Forma sequencial do algoritmo.

A maioria das imagens digitais é composta por três canais de cores separados: um canal vermelho, um canal verde e um canal azul. A sobreposição desses canais em camadas cria uma imagem colorida. Modelos de cores diferentes têm canais diferentes (às vezes os canais são cores, às vezes são outros valores como luminosidade ou saturação), mas este trabalho se concentra apenas no RGB.

Todos os algoritmos de conversão do canal RGB para escala de cinza utilizam o mesmo processo básico que consistem em três etapas:

- Obter os valores dos canais vermelho, verde e azul de um *pixel*.

- Relacionar esses valores para transformar em apenas um valor.
- Substituir os valores originais de vermelho, verde e azul pelo valor calculado

```
for each pixel na imagem{

        vermelho = pixel.Red()
        verde = pixel.Green()
        azul = pizel.Blue()

        Obter_Valor_Cinza = cinza

        pixel.Red = cinza
        pixel.Green = cinza
        pixel.Blue = cinza

}
```

Os métodos para obtenção do valor cinza para cada algorimo são apresentados a seguir.

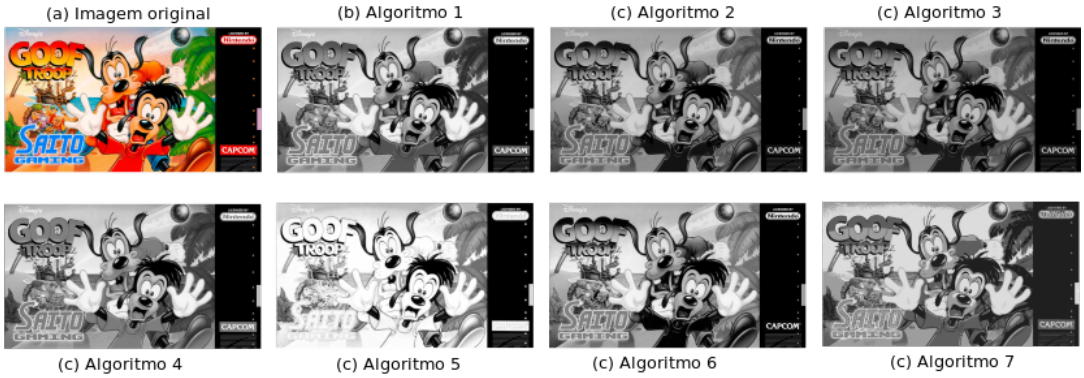

Figura 2. Resultado dos algoritmos.

3.1.1. Algoritmo 1: Média

O algoritmo baseado na média é a rotina de conversão mais comum em escala de cinza e o calculo do valor de cinza é obtido a partir da Equação 1.

$$cinza = (vermelho + verde + azul)/3 \tag{1}$$

Essa fórmula gera um equivalente em escala de cinza razoavelmente boa e sua simplicidade facilita a implementação e otimização. No entanto, esse método não deixa de ter falhas — embora rápida e simples, ela faz um mau trabalho em representar tons de cinza em relação à maneira como os seres humanos percebem a luminosidade (brilho).

3.1.2. Algoritmo 2: Luminância

Esse segundo algoritmo mostra que a densidade do cone no olho humano não é uniforme entre as cores. Os seres humanos percebem o verde mais fortemente que o vermelho e o vermelho mais fortemente que o azul. Isso faz sentido do ponto de vista da biologia evolucionária — grande parte do mundo natural aparece em tons de verde, de modo que os humanos desenvolveram maior sensibilidade à luz verde.

Como os humanos não percebem todas as cores da mesma forma, o "método da média" de conversão em escala de cinza é impreciso. Em vez de tratar as luzes vermelha, verde e azul igualmente, uma boa conversão em escala de cinza pondera cada cor com base na maneira como o olho humano a percebe. Uma fórmula comum em processadores de imagem (Photoshop, GIMP) é apresentada na Equação 2.

$$cinza = (0.3 * vermelho + 0.59 * verde + 0.11 * azul)/3 \tag{2}$$

3.1.3. Algoritmo 3: Luma

Assim como no algoritmo 2, este método tenta uniformizar a forma como o ser humano perceber as cores. Para isso, também é utilizada uma média ponderada entre os canais como é mostrado na Equação 3

$$cinza = (0.2126 * vermelho + 0.7152 * verde + 0.0722 * azul)/3 \tag{3}$$

3.1.4. Algoritmo 4: Dessaturação

A maioria dos programadores usa o modelo de cores RGB. Embora seja uma boa maneira de uma máquina descrever cores, o espaço de cores RGB pode ser difícil para a percepção humana. Nesse sentido, o espaço de color HSL (do inglês *hue, saturation, lightness*) é algumas vezes utilizado para a vizualização das cores. A matiz (*Hue*) pode ser considerada o nome da cor — vermelho, verde, laranja, amarelo etc. A saturação (*saturation*) descreve como uma cor é vívida; uma cor muito vívida tem saturação total, enquanto o cinza não tem saturação.

A dessaturação de uma imagem funciona convertendo o espaço RGB no espaço HSL e forçando a saturação para zero. Um pixel pode ser dessaturado encontrando o ponto médio entre o máximo de (R, G, B) e o mínimo de (R, G, B), como mostra a Equação 4.

$$cinza = (Max(vermelho, verde, azul) + Min(vermelho, verde, azul))/2 \quad (4)$$

3.1.5. Algoritmo 5: Decomposição

A decomposição de uma imagem oPde ser considerada uma forma mais simples de dessaturação. Para decompor uma imagem, cada *pixel* é forçado parao valor mais alto (máximo) ou mais baixo (mínimo) de seus valores em vermelho, verde e azul. As Equações 5a e 5b mostram como calcular a decomposição máxima e mínima, respectivamente.

$$cinza = Max(vermelho, verde, azul) \quad (5a)$$
$$cinza = Min(vermelho, verde, azul) \quad (5b)$$

3.1.6. Algoritmo 6: Canal de cor única

Este algoritmo representa o método computacional mais simples para conversão em escala de cinza. Ao contrário de todos os métodos apresentados, este método não requer cálculos. Apenas é necessário escolher o valor de um determinado canal e aplicar para ao valor de cinza, como mostrado na Equação 6a, 6b e 6c

$$cinza = vermelho \quad (6a)$$
$$cinza = verde \quad (6b)$$
$$cinza = azul \quad (6c)$$

3.1.7. Algoritmo 7: Tons de cinza

Este método permite ao usuário especificar quantos tons de cinza a imagem resultante utilizará. O algoritmo definir o número de tons de cinza da imagem convertida. Esse método é um pouco mais trabalhoso se comparado com os demais algoritmos apresentados. Os procedimentos do algoritmo é mostrado a seguir.

```
for each pixel na imagem{
        fator = 255 / (TonsDeCinza - 1)
        metodo = (vermelho + verde + azul)/3
        cinza = (int)((medio / fator) + 0.5) * fator
```

```
        //TonsDeCinza: valor entre 2 e 256
}
```

4. Conclusão

ok

Referências

Arduino (2019). Arduino - home. https://www.arduino.cc/. (Acessado em 09/28/2019).

Batista, C. T., Campos, E. M., and Neto, N. C. S. (2017). A proposal of a universal remote control system based on head movements. In *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems*, IHC 2017, New York, NY, USA. ACM.

BeagleBone (2019). Beagleboard.org - black. https://beagleboard.org/black. (Acessadoe em 09/28/2019).

C.H.I.P (2019). Pocket c.h.i.p. formerly from next thing co. https://shop.pocketchip.co/. (Acessado em 09/28/2019).

Karthik G. and Praburam N. (2016). Detection and prevention of banana leaf diseases from banana plant using embeeded linux board. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5.

Li, C., Zhang, Y., and Zheng, Z. (2009). Design of image acquisition and processing based on fpga. In *2009 International Forum on Information Technology and Applications*, volume 3, pages 113–115.

Markovic, D., Vujicic, D., Dragana, M., and Ranđić, S. (2018). Image processing on raspberry pi cluster. 2:83 – 90.

OpenCV (2019). Opencv -the latest release is now available. https://opencv.org/. (Acessado em 09/28/2019).

Sivaranjani, S. and Sumathi, S. (2015). Implementation of fingerprint and newborn footprint feature extraction on raspberry pi. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–6.

Tartu, U. (2019). Digital image processing. https://sisu.ut.ee/imageprocessing/book/1. (Acessado em 09/28/2019).

Xilinx (2019). What is an fpga? field programmable gate array. https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. (Acessado em 09/28/2019).