Processamento Digital de Imagens em Plataformas Embarcadas

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. *Image processing is a method that performs a series of mathematical operations on an image. The intent is to get an enhancement or extract some useful information about the image being processed. Although image processing algorithms generally run on more robust machines, it is also possible to employ these tasks on embedded systems. In this sense, this work aims to use the Raspberry Pi platform, more specifically the Raspberry Pi 3 model B version. The idea is to apply grayscale conversion algorithms on low cost platforms that have a multicore architecture to be able to perform parallelism techniques, making the most of the processing power of the platform used.*

Resumo. *O processamento de imagens é um método que executa uma série de operações matemáticas em uma imagem. intuito é obter um aprimoramento ou extrair algumas informações úteis a cerca da imagem a ser processada. Apesar dos algoritmos de processamento de imagens serem executados geralmente em máquinas mais robustas, é possível também empregar essas tarefas em sistemas embarcados. Nesse sentido, este trabalho tem como proposta utilizar a plataforma Raspberry Pi, mais espicificamente a versão Raspberry Pi 3 modelo B. A ideia é aplicar algoritmos de conversão em escala de cinza em plataformas de baixo custo e que possuem uma arquitetura multicore para que seja capaz de aplicar tecnicas de paralelismo, aproveitando ao máximo o poder de processamento da plataforma utilizada.*

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.

Apesar desses algoritmos serem executados geralmente em máquinas mais robustas, é possível também empregar essas tarefas em sistemas embarcados. Os sistemas embarcados são sistemas de computação de propósito específico que geralmente são desenvolvidos para realizar algumas funções dedicadas, muitas vezes com restrições computacionais de temp real. São chamados de embarcados por serem parte de um dispositivo completo, incluindo *hardware* e partes mecânicas [Lamb 2015].

Um sistema embarcado bastante conhecido atualmente entre os desenvolvedores é o Raspberry Pi [Raspberry 2019], uma plataforma de baixo custo com o intuito de ser um computador *desktop* "completo". Esse dispositivo possui um grande poder computacional, se comparado com os seus concorrentes. Em relação ao processamento digital de imagens, existem muitos projetos que utilizam o raspberry Pi para realizar essa tarefa como em [S et al. 2015].

Nesse sentido, tem como proposta utilizar a plataforma Raspberry Pi, mais espicificamente a versão Raspberry Pi 3 modelo B, um dispositivo *quad core* que possui 1 GB de RAM e é muito utilizado na comunidade. A ideia é aplicar algoritmos de conversão em escala de cinza em plataformas de baixo custo, como os sistemas embarcados, e que possuem uma arquitetura *multicore* para que seja capaz de aplicar tecnicas de paralelismo, aproveitando ao máximo o poder de processamento da plataforma utilizada neste trabalho.

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. Objetivos

O objetivo geral do trabalho é aplicar algoritmos e processamento de imagem em plataformas embarcadas de modo que as tarefas sejam executadas de forma eficiente e de forma paralela para que seja utilizado todo o poder de processamento da arquitetura escolhida. A principio, apenas algoritmos referentes a conversão em escala de cinza foram implementados, como prova de conceito, utilizando a biblioteca de computação visual Opencv [OpenCV 2019] em um Raspberry Pi 3 Modelo B [Raspberry 2019]. No entanto, o sistema projetado pode ser amplicado para ser utilizado em outros sistemas embarcadas. É importante também que o trabalho seja de baixo custo (por isso a utilização de sistemas embarcadas) e que os algoritmos implementados de forma sequencial e paralela seja livremente disponibilizado.

3.1. Objetivos Específicos

Os objetivos específicos são: 1) estudo da biblioteca utilzada e dos algoritmos de conversão em escala de cinza; 2) implementação sequencial; 3)implementação paralela.

Uma breve descrição sobre cada tópico é feita a seguir.

1. Estudo da biblioteca utilzada e dos algoritmos de conversão
    - Estudo da blioteca OpenCV com o intuito de como funciona a manipulação dos dados de cada *pixel*.
2. Implementação sequencial
    - Implementação sequencial da forma mais eficiente dos algoritmos de conversão em escala de cinza utilizados neste trabalho.
3. Implementação paralela
    - Implementação paralela das tarefas de conversão em escala de cinza de modo que a execução dessas tarefas sejam executadas de forma mais rápida e aproveitando ao máximo o poder de processamento da plataforma embarcada utilzada.

4. Metodologia

Esta seção tem como objetivo mostrar a metodogia para implementação do trabalho proposto. Primeiramente é mostrada a forma sequencia de como foi implementado os algoritmos. Logo em seguida é apresentada a proposta de paralelização das tarefas sequenciais utilizando a metodologia PCAM.

4.1. Forma Sequencial Implementada

A Figura 1 mostra o fluxograma da forma sequencial das tarefas implementadas neste trabalho. Cada algoritmo realiza a conversão em escala de cinza de imagens contidas em uma base com 5 mil imagens que são processadas um por vez por cada algoritmo seguindo uma ordem sequencial de cada algoritmo aplicado.

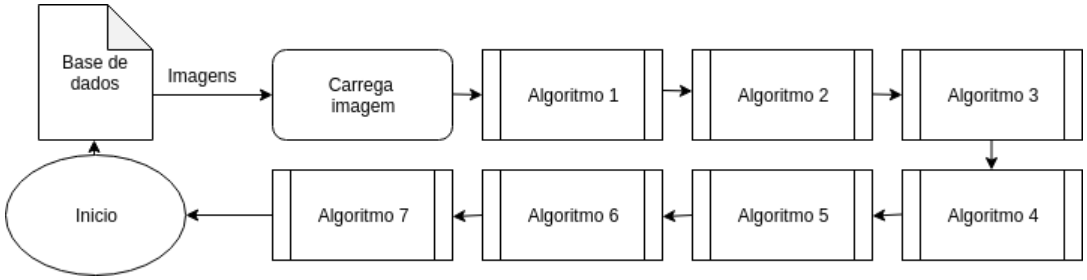

Figura 1. Forma sequencial do algoritmo.

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
}
```

A Figura 2 mostra o resultado da conversão em escala de cinza para cada algortimo utilizado neste trabalho. Os métodos para obtenção do valor cinza para cada algorimo são apresentados a seguir.

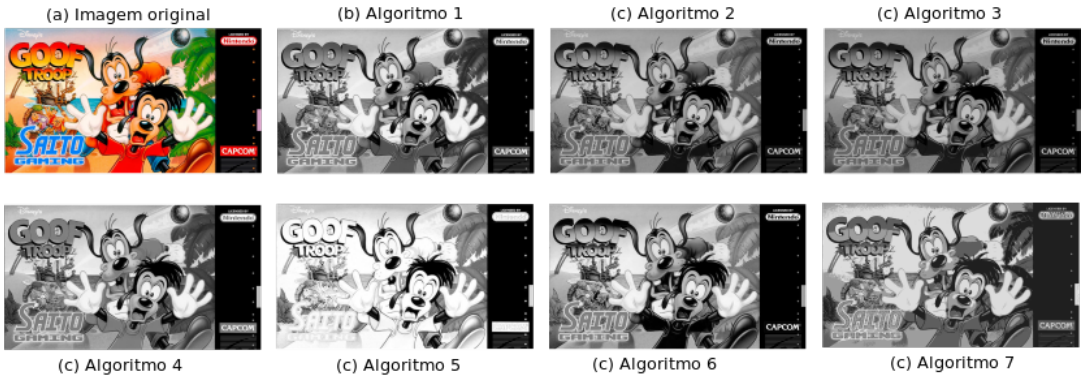

Figura 2. Resultado dos algoritmos.

4.1.1. 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).

4.1.2. 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.

$$cinza = (0.3 * vermelho + 0.59 * verde + 0.11 * azul) \quad (2)$$

4.1.3. 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

$$cinza = (0.2126 * vermelho + 0.7152 * verde + 0.0722 * azul) \quad (3)$$

4.1.4. 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.

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

4.1.5. 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.

$$cinza = Max(vermelho, verde, azul) \tag{5a}$$

$$cinza = Min(vermelho, verde, azul) \tag{5b}$$

4.1.6. 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

$$cinza = vermelho \tag{6a}$$

$$cinza = verde \tag{6b}$$

$$cinza = azul \tag{6c}$$

4.1.7. 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
}
```

4.2. Projeto da Forma Paralela

O projeto da forma paralela segue a metodologia PCAM (Particionamento, Comunicação, Agrupamento e Mapeamento) descritra a seguir.

- **Particionamento**: Quebra do problema em subproblemas menores.
- **Comunicação**:Definição da comunicação entre as várias tarefas que foram subdivididas.

- **Agrupamento**: Combinação das tarefas de forma a se estabelecer a comunicação, se necessário, visando reduzir custos
- **Mapeamento**: Escolha de quais tarefas vão ficar em cada processador.

4.2.1. Particionamento

Como o Raspberry Pi Modelo B utilizado para executar o programa desenvolvido para teste trabalho possui 4 núcleos de processamento, é importante subdividir o problema em subtarefas. Uma alternativa seria dividir os 7 algoritmos entre os processadores. O problema desta abordagem é que um dos processadores ia executar apenas um algoritmo, subutilizando-o em comparação aos demais núcleos. Outra abordadem que pode ser utilizado, e a que provavelmente vai ser utilizada, é o processamento de 4 imagens em paralel, onde cada núcleo seria encarregado de executar os 7 algoritmos para suas respectivas imagens.

4.2.2. Comunicação

O programa desenvolvida a presenta 7 algoritmos de conversão em escala de cinza de uma imagem e esses algoritmos são independentes entre si, o que torna a implementação do paralelismo nessas tarefas sem a necessidade de realizar comunicações entre os núcleos de processamento.

4.2.3. Agrupamento

Como não há a necessidade de comunicação entre as unidades de processamento, cada núcleo será responsável por processar uma imagem, visto que se o particion amento fosse projetado levando em consideração as tarefas, o agrupamento de tarefas ficaria desbalanceado, subutilizando uma das unidades de processamento (UP).

4.2.4. Mapeamento

O paralelismo vai ser aplicado na quantidade de imagens que devem ser processadas e não na quantidade de algoritmos de conversão em escala de cinza. Dessa forma, 4 imagens devem ser trabalhadas em paralelo, onde a primeira imagem deve ser processada pela UP1, a segunda imagen pela UP2, a terceira imagem pela UP3 e a quarta imagem pela UP4. Assim que as 4 imagens forem processadas, mais 4 novas imagens são dadas como entrada no sistema que refaz todo o processo.

5. Paralelismo

Esta seção tem como objetivo mostrar aplicação do paralelismo sobre os algoritmos analisados neste trabalho.

5.1. OpenMP

Para a aplicação do paralelismo utilizando a biblioteca OpenMP, foi utilizada a abordagem de processar uma imagem por vez, mas subdividindo as imagens em quatro partes iguais. Um exemplo dessa subdivisão pode ser vista na Figura 3.

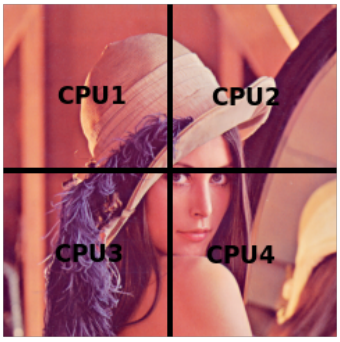

Figura 3. Subdivisão das imagens.

Cada processador fica responsável por realizar o algoritmo de conversão em escala de cinza em uma determinada região da imagem (quando é utilizado realmente 4 *threads*). Todo esse processo é realizado para cada imagem até que todas as imagens sejam processadas.

6. Avaliação de Desempenho

Para realizar as simulações, foi determinado cenários diferentes onde cada cenário executou o código com uma certa quantidade de processadores. Dessa forma foi possível medir o desempenho conforme a quantidade de processadores utilizadas. Para cada cenário, o código foi executado cinco vezes e foi coletado a média dessas cinco execuções para representar o tempo de execução.

Para a análise do desempenho foi utilizado as seguintes métricas:

- Tempo de execução
- *Speedup*
- Eficiência
- Desempenho

O tempo de execução é a soma do tempo de computação, tempo de comunicação e o tempo ocioso, como mostrado na Equação 7.

$$T_{exec} = T_{comput} + T_{comunic} + T_{ocioso} \tag{7}$$

O *speedup* é a razão entre o tempo de execução em apenas um processador e o tempo de execução em múltiplos processadores. A equação 8 mostra o cálculo do *speedup*.

$$S = \frac{tempo\ 1\ processador}{tempo\ N\ processadores} \tag{8}$$

A eficiência é determinada pela relação entre o speedup obtido e o número de processadores necessários para obtê-lo, como mostrado na Equação 9.

$$E = \frac{Speedup}{numero\ de\ processadores} \tag{9}$$

Já o desempenho Determina o "quanto" os processadores estão sendo utilizados e é calculado pela Equação 10.

$$D = E \times 100\% \tag{10}$$

6.1. Desempenho OpenMP

A Tabela 1 mostra o resultados da avaliação de desempenho para o paralelismo aplicado utilizando a biblioteca OpenMP.

Tabela 1. OpenMP: análise de desempenho

Nº CPU	Tempo exec.	*Speedup*	Eficiência	Desempenho
1	4h8m7s	–	–	–
2	2h12m38s	1.870	0.935	93.5%
3	1h39m22s	2.496	0.832	83.2%
4	1h17m17s	3.210	0.802	80.2%

É possível perceber que o tempo de execução diminuiu consideravelmente a medida que a quantidade de processadores aumentou. Ou seja, a utilização do paralelismo foi imprescindível para que o tempo de execução melhorasse. Contudo o desempenho dos processares foi inversamente proporcional, ao número de *threads* utilizados.

7. Avaliação de Desempenho

Este trabalho apresentou uma proposta de aplicação de algoritmos de processamento digital de imagens, mais especificamente os de conversão em escala ed cinza, utilizando a plataforma embarcada Raspberry Pi Modelo B. O algoritmo desenvolvido para reali-

zar a conversão em escala de cinza em imagens atualmente está na sua fase de execução sequecial.

A partir do projeto de aplicação paralela mostrados neste trabalho, um desenpenho significativamente superior à execução sequencial é esperado para os algoritmos de processamento de imagens. Dessa forma, o tempo de resolução dos processos deve ser reduzindo, sendo possível processar mais imagens em um tempo menor.

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.

Lamb, F. (2015). *Automação Industrial na Prática - Série Tekne*. Tekne. AMGH Editora.

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).

Raspberry (2019). Teach, learn, and make with raspberry pi – raspberry pi. `https://www.raspberrypi.org/`. (Acessado em 09/29/2019).

S, S., Hlmg, L., and Shivkumar, H. (2015). Implementation of image processing on raspberry pi. *IJARCCE*, 4:199–202.

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).