

Wali Queiroz Santos

# Protótipo para Contagem Automática de Células Sanguíneas usando Processamento de Imagens

Vitória da Conquista/BA

2019

Wali Queiroz Santos

# Protótipo para Contagem Automática de Células Sanguíneas usando Processamento de Imagens

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação na Universidade Estadual do Sudoeste da Bahia – UESB.

Universidade Estadual do Sudoeste da Bahia – UESB

Curso de Bacharelado em Ciência da Computação

Departamento de Ciências Exatas e Tecnológicas

Orientador: Prof<sup>a</sup>. Dr<sup>a</sup>. Alexsandra Oliveira Andrade

Vitória da Conquista/BA

2019

Wali Queiroz Santos

Protótipo para Contagem Automática de Células Sanguíneas usando Processamento de Imagens/ Wali Queiroz Santos. – Vitória da Conquista/BA, 2019-  
64 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof<sup>a</sup>. Dr<sup>a</sup>. Alexandra Oliveira Andrade

Trabalho de Conclusão de Curso (Graduação)  
Universidade Estadual do Sudoeste da Bahia – UESB  
Curso de Bacharelado em Ciência da Computação  
Departamento de Ciências Exatas e Tecnológicas, 2019.

1. Processamento de imagens. 2. Raspberry Pi. 3. Contagem de células sanguíneas.  
I. Dr<sup>a</sup>. Alexandra Oliveira Andrade. II. Universidade Estadual do Sudoeste da Bahia.  
III. Curso de Ciência da Computação. IV. Protótipo para Contagem Automática de Células Sanguíneas usando Processamento de Imagens.

Wali Queiroz Santos

## **Protótipo para Contagem Automática de Células Sanguíneas usando Processamento de Imagens**

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação na Universidade Estadual do Sudoeste da Bahia – UESB.

Trabalho aprovado. Vitória da Conquista/BA, 23 de agosto de 2019:

---

**Prof<sup>a</sup>. Dr<sup>a</sup>. Alexandra Oliveira  
Andrade**  
Orientadora

---

**Prof. Dr. Odair Lacerda Lemos**  
UESB

---

**Prof. Me. Marcos Gomes Prado**  
FAINOR

Vitória da Conquista/BA  
2019

*Este trabalho é dedicado a Suely Pereira Queiroz, minha mãe, que não mediu esforços para que eu chegasse até essa etapa da minha vida.*

# Agradecimentos

Agradeço primeiramente a Deus, causa primordial de todas as coisas. Devo a Ele tudo o que sou.

Agradeço a minha mãe, Suely Pereira Queiroz, pelo carinho, dedicação, atenção e por cuidar de mim durante toda a trajetória, mesmo que de longe; essa mulher guerreira é minha maior inspiração. Agradeço ao meu pai, Apio Soares Santos, grande colaborador e incentivador.

Agradeço a minha avó, Adely Pereira Queiroz, sem ela eu não teria nem ingressado no ensino superior. Mesmo em meio às dificuldades, sempre fez de tudo para que eu concluísse a graduação, contribuindo de todas as formas possíveis.

Agradeço a minha tia, Lindinalva Santos, por sempre acreditar em mim; é minha segunda mãe. Mulher íntegra, educada, super inteligente e que sabe dar conselhos como ninguém. Agradeço ao meu irmão Inaldo Soares por ter contribuído financeiramente com o meu sonho.

Agradeço a minha madrinha, Cleusa Lima, e a sua filha, Edcleia Lima, por terem viabilizado minha mudança para Vitória da Conquista. Agradeço a Irece Barbosa por ter me acolhido em sua casa quando tudo parecia estar indo por água abaixo. Esta pessoa é um anjo que Deus colocou em minha vida; serei eternamente grato por tudo que ela fez por mim.

Aos amigos que fiz durante o curso, em especial a Brunna (com dois n's) pelo companheirismo, incentivos, conselhos e por me ajudar a comprar roupas, a João pelas parcerias nos trabalhos, a Léo por ir me buscar toda vez que eu voltava de Itaquara com bagagens, a Lucas pelas melhores piadas e situações inusitadas, a Wesley por ser meu confidente e a Bernardo por ensinar a dar patadas com classe. Agradeço também aos colegas de trabalho do NTI, especialmente a Rodrigo, Hatus, Jean, André e Badaró; companheiros de jornada com quem compartilho minha vida profissional.

Agradeço a todos os professores do curso de Ciência da Computação, sobretudo a Prof<sup>a</sup>. Dr<sup>a</sup>. Aleksandra Andrade, minha orientadora, que vem trabalhando comigo desde o terceiro semestre e aos professores Júlio Cesar, Cristina Reis, Stenio Longo e Marlos Marques pela didática em sala de aula. Não poderia deixar de agradecer a Celina Pereira, pessoa maravilhosa, carismática e que sempre está disposta a ajudar os alunos do curso.

Por fim, agradeço aos meus familiares que me incentivaram com palavras e gestos ao longo do processo e a todos que colaboraram direta ou indiretamente para que eu chegasse até aqui. Obrigado!

*“O conhecimento é uma arma. Arme-se bem antes de ir para a batalha.”*

*(Meistre Aemon)*

# Resumo

Este trabalho descreve as etapas para a construção de um protótipo capaz de capturar e processar automaticamente imagens de células sanguíneas e o desenvolvimento do algoritmo que utiliza técnicas de processamento de imagens para contar as hemácias. As imagens são obtidas de lâminas de extensão sanguínea e o equipamento é baseado num Raspberry Pi 3 modelo B+. Trata-se de um desenvolvimento experimental. No algoritmo de contagem das células vermelhas do sangue foram aplicadas técnicas de segmentação baseadas em similaridade, como limiarizações e o algoritmo *Watershed*. A quantificação é feita através de um método que utiliza o tamanho médio das células em *pixels*. A biblioteca OpenCV foi bastante utilizada no código devido à sua grande coleção de funções para manipulação de imagens. O protótipo desenvolvido realiza o processo automatizado de captura e processamento de 20 imagens das células em cerca de 3 minutos. Quando as imagens são capturadas sem avarias e dentro do padrão exigido, o percentual de acerto do algoritmo de contagem pode ultrapassar 95%.

**Palavras-chave:** Processamento de imagens. Contagem de células sanguíneas. Raspberry Pi. Protótipo. Contagem de Hemácias.

# Abstract

This paper describes the steps for building a prototype capable of automatically capturing and processing blood cell images and the development of the algorithm that uses image processing techniques to count red blood cells. The images are obtained from blood extension slides and the equipment is based on a Raspberry Pi 3 model B+. It is an experimental development. In the red blood cells counting algorithm, similarity-based segmentation techniques were applied, such as thresholds and the Watershed algorithm. Quantitation is done using a method that uses the average cell size in pixels. The OpenCV library has been widely used in code due to its large collection of image manipulation functions. The developed prototype performs the automated process of capturing and processing 20 cell images in about 3 minutes. When images are captured without malfunctions and within the required standard, the accuracy of the counting algorithm may exceed 95%.

**Keywords:** Image processing. Blood cells counting. Raspberry Pi. Prototype. Red blood cells counting.

# Lista de ilustrações

Figura 1 – Vizinhança-de-4 de $p$ . . . . .	20
Figura 2 – (a) Vizinhos Diagonais de $p$ . (b) Vizinhança-de-8 de $p$ . . . . .	21
Figura 3 – (a) Imagem $A$ . (b) Elemento Estruturante $B$ . (c) Dilatação de $A$ por $B$ . . . . .	26
Figura 4 – (a) Imagem $A$ . (b) Elemento Estruturante $B$ . (c) Erosão de $A$ por $B$ . . . . .	26
Figura 5 – (a) Imagem $A$ . (b) Elemento Estruturante $B$ . (c) Abertura de $A$ por $B$ . . . . .	27
Figura 6 – (a) Imagem $A$ . (b) Elemento Estruturante $B$ . (c) Fechamento de $A$ por $B$ . . . . .	28
Figura 7 – Raspberry Pi 3 modelo B+ . . . . .	29
Figura 8 – Motor de Passo . . . . .	30
Figura 9 – Protótipo - Visão frontal. . . . .	32
Figura 10 – Protótipo. . . . .	33
Figura 11 – Modelo 3D do Invólucro do Protótipo. . . . .	34
Figura 12 – Modelo 3D da Gaveta do Protótipo. . . . .	35
Figura 13 – Modelo Tridimensional do Invólucro Dividido. . . . .	35
Figura 14 – Tela <i>Touchscreen</i> . . . . .	36
Figura 15 – Raspberry Pi com Câmera e Tela conectadas. . . . .	37
Figura 16 – Micro Phone Lens . . . . .	38
Figura 17 – Modelo Tridimensional do Suporte para a Lente. . . . .	38
Figura 18 – Módulo de Movimentação da Câmera com a Lente - Vista Traseira. . . . .	39
Figura 19 – Módulo de Movimentação da Câmera com a Lente - Vista Frontal. . . . .	40
Figura 20 – Módulo de Movimentação da Lâmina. . . . .	40
Figura 21 – Chave Metálica para Movimentação da Lâmina. . . . .	41
Figura 22 – Diagrama do Circuito do Controle dos Motores. . . . .	42
Figura 23 – Lâmpada LED Branca utilizada como Fonte de Iluminação. . . . .	43
Figura 24 – Trecho do Algoritmo responsável por Ativar os Módulos do Protótipo. . . . .	44
Figura 25 – Imagem Capturada Automaticamente pelo Protótipo. . . . .	44
Figura 26 – (a) Imagem de Entrada. (b) Imagem Binarizada. (c) Região de Interesse Extraída. (d) Região de Interesse Recortada. . . . .	45
Figura 27 – (a) Imagem de Entrada após Passar pela Redução de Ruído. (b) Imagem Convertida para Tons de Cinza. (c) Imagem após Passar por Duas Aberturas Morfológicas. (d) Imagem após Passar pela Binarização Adaptativa. . . . .	47
Figura 28 – Elementos Estruturantes. . . . .	49

Figura 29 – (a) Imagem após Passar por Inversão de <i>Pixels</i> e um Fechamento Morfológico. (b) Resultado das Dilatações para Detecção do Fundo. (c) Imagem após Passar por 3 Erosões. (d) Imagem após Passar pela Transformada de Distância e uma Limiarização Simples; Área em Primeiro Plano. . . . .	49
Figura 30 – Região Desconhecida. . . . .	50
Figura 31 – Matriz de Marcadores Colorizada. . . . .	51
Figura 32 – Resultado Final do Processo de Separação da Figura 26 - (d). . . . .	52
Figura 33 – (a) Resultado da Filtragem. (b) Resultado da Inversão. . . . .	52
Figura 34 – Fragmento da Figura 33 (b) que destaca uma área onde o processo de separação falhou. . . . .	53
Figura 35 – Imagem de Lâmina com Esfregação Denso e sem Diluição. . . . .	57
Figura 36 – (a) Imagem com Mancha causada pelo Deslocamento da Lente. (b) Imagem com Iluminação Inadequada. (c) Imagem com Distorções. (d) Imagem Desfocada. . . . .	58

# Lista de abreviaturas e siglas

OpenCV: *Open Source Computer Vision Library*

GPIO: *General Purpose Input/Output*

LCD: *Liquid Crystal Display*

RAM: *Random Access Memory*

GB: *Gigabyte*

GHz: *GigaHertz*

LED: *Light Emitting Diode*

TFT: *Thin Film Transistor*

HDMI: *High-Definition Multimedia Interface*

USB: *Universal Serial Bus*

CD-ROM: *Compact Disc Read-Only Memory*

SAUF: *Scan Array Union Find*

BBDT: *Block-Based Decision Table*

# Sumário

<b>I</b>	<b>PREPARAÇÃO DA PESQUISA</b>	<b>14</b>
<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
1.1	Justificativa	16
1.2	Objetivos	16
1.2.1	Objetivo geral	16
1.2.2	Objetivos Específicos	16
1.3	Método de Pesquisa	16
1.4	Estrutura do Trabalho	17
<b>II</b>	<b>REFERENCIAIS TEÓRICOS</b>	<b>18</b>
<b>2</b>	<b>REFERENCIAIS TEÓRICOS</b>	<b>19</b>
2.1	Processamento de Imagens	19
2.2	Fundamentos de Imagens Digitais	20
2.2.1	Os vizinhos de um <i>pixel</i>	20
2.2.2	Conectividade	21
2.2.3	Rotulação de Componentes Conectados	22
2.3	Segmentação de Imagens	22
2.3.1	Limiarização	23
2.3.1.1	Limiarização global simples	23
2.3.1.2	Limiarização Adaptativa	24
2.3.2	O algoritmo <i>Watershed</i>	24
2.4	Morfologia Matemática	25
2.4.1	Dilatação	25
2.4.2	Erosão	26
2.4.3	Abertura	27
2.4.4	Fechamento	27
2.5	A Biblioteca <i>OpenCV</i>	28
2.6	Impressão 3D	28
2.7	Raspberry Pi	29
2.8	Motores de Passo	29
2.8.1	Ponte H	30

<b>III</b>	<b>O PROTÓTIPO PARA CONTAGEM AUTOMATIZADA DE CÉLULAS SANGUÍNEAS USANDO PROCESSAMENTO DE IMAGENS</b>	<b>31</b>
<b>3</b>	<b>O PROTÓTIPO PARA CONTAGEM AUTOMATIZADA DE CÉLULAS SANGUÍNEAS USANDO PROCESSAMENTO DE IMAGENS</b>	<b>32</b>
<b>3.1</b>	<b>Construção do Equipamento</b>	<b>33</b>
3.1.1	Modelagem do Invólucro e da Gaveta	33
3.1.2	Instalação da Tela	35
3.1.3	Instalação da Câmera e da Lente	36
3.1.4	Instalação dos Módulos de Movimentação com Motores de Passo	38
3.1.5	Instalação da Lâmpada de Retroiluminação	42
3.1.6	Implementação do algoritmo para captura das imagens e controle dos módulos	43
<b>3.2</b>	<b>Implementação do Algoritmo para Contagem das Células Sanguíneas</b>	<b>44</b>
3.2.1	Recorte da Área de Interesse	45
3.2.2	O Processo de Separação	46
3.2.3	O Processo de Contagem	51
<b>IV</b>	<b>RESULTADOS</b>	<b>55</b>
<b>4</b>	<b>RESULTADOS</b>	<b>56</b>
4.1	Análise do <i>Hardware</i> do Protótipo	56
4.2	Análise do Algoritmo de Processamento das Imagens	58
<b>5</b>	<b>CONCLUSÃO</b>	<b>60</b>
5.1	Trabalhos Futuros	61
	<b>REFERÊNCIAS</b>	<b>62</b>

# Parte I

## Preparação da Pesquisa

# 1 Introdução

O hemograma é um tipo de exame de sangue utilizado para medir a saúde geral de uma pessoa. Nele são analisados os três principais grupos de células do sangue: As hemácias, os leucócitos e as plaquetas. É o exame complementar mais requerido nas consultas ([FAILACE, 2015](#)).

O eritrograma é a parte do hemograma que corresponde a análise das células vermelhas do sangue, ou seja, dos eritrócitos, também conhecidos como hemácias ou glóbulos vermelhos. Nele é realizada a contagem dos eritrócitos, que é feita por contadores automáticos, a dosagem de hemoglobina, que é a substância responsável pelo transporte de oxigênio dos pulmões até os tecidos, o hematócrito, que é o volume de eritrócitos por coluna de sangue centrifugado, dentre outras dosagens e contagens ([FAILACE, 2015](#)).

Através das análises desses índices e associados aos exames clínicos feitos pelos pacientes, os médicos conseguem fechar o diagnóstico de algumas doenças relacionadas com as células de linhagem vermelha, tais como: diversas anemias (anemia ferropriva, anemia sideroblástica e anemia falciforme), hemoglobinopatias (talassemias), entre outras. Atualmente este exame é feito por contadores eletrônicos, que aspiram o sangue e fazem automaticamente todas as determinações em múltiplos canais, ou seja, a contagem é feita por impedância elétrica.

Este projeto descreve as etapas para a construção de um protótipo capaz de capturar e processar automaticamente fotos de células sanguíneas e o desenvolvimento do algoritmo que utiliza técnicas de processamento de imagens para contar as células vermelhas do sangue contidas nessas fotos.

A ideia original está na utilização de imagens de lâminas de extensão sanguínea, reproduzindo os valores encontrados num eritrograma, só que com imagens, o que nos levaria a desenvolver padrões novos para análise. Seria um aparelho de baixo custo que poderia facilitar diagnósticos de patologias em postos médicos e com um tempo bem curto.

O dispositivo em questão utiliza como base um Raspberry Pi, que é um microcomputador. Isso torna o equipamento portátil, logo, este pode facilitar a realização de exames em locais de difícil acesso como, por exemplo, zonas rurais.

Andrade ([ANDRADE et al., 2012](#)) criou um método de contagem de esporos de fungos micorrízicos usando operadores morfológicos *fuzzy* e desenvolveu um protótipo contador automático de esporos de fungos micorrízicos. Neste projeto, foi utilizado todo o conhecimento adquirido para este protótipo no aparelho de contagem de células sanguíneas, é claro, com suas devidas adaptações e especificações.

## 1.1 Justificativa

Um dispositivo para contagem automatizada de células sanguíneas vermelhas a partir de imagens de lâminas com esfregaços de sangue humano é importante para diagnósticos de doenças como diversas anemias, bem como a identificação das causas associadas. Neste caso, vamos adentrar numa área da computação muito utilizável, o processamento de imagens, que de 1964 até hoje vem crescendo vigorosamente e sendo aplicado nas mais diversas áreas.

## 1.2 Objetivos

### 1.2.1 Objetivo geral

Este trabalho tem por objetivo desenvolver um equipamento portátil e de baixo custo capaz de realizar a contagem automática de glóbulos vermelhos a partir de imagens digitais capturadas de lâminas com esfregaços de sangue humano.

### 1.2.2 Objetivos Específicos

- Criar um invólucro para o dispositivo e um modelo para encaixe de uma câmera com lente de aumento para captura de imagens microscópicas;
- Desenvolver um mecanismo automatizado para movimentar a câmera a fim de capturar imagens de diferentes partes da lâmina com amostra de sangue humano;
- Desenvolver um algoritmo para o processamento de imagens de células sanguíneas;
- Instalar o sistema e acoplar os módulos de movimentação e captura de imagens à plataforma Raspberry Pi;

## 1.3 Método de Pesquisa

As pesquisas desenvolvidas neste trabalho foram divididas nas vertentes formal e experimental e, com isso, a metodologia teve de ser adequada à vertente empregada.

Utilizou-se o método hipotético-dedutivo para a vertente formal. Esta metodologia procura uma solução através do estabelecimento de conjecturas, hipóteses e teorias, e realização de testes de falseamento para eliminação de erros. Ela se desenvolve através de raciocínios e trabalha do geral para o particular, levando o pesquisador do conhecido para o desconhecido. É um processo cíclico que evolui à medida em que as hipóteses são reformuladas e as técnicas para realizar observações e experimentos melhoram. (MARCONE; LAKATOS, 2003).

Para a vertente experimental, foi utilizada a metodologia incremental-iterativa. Nessa abordagem, várias partes do problema são desenvolvidas em paralelo, integradas quando completas e reelaboradas quando necessário, até que o objetivo seja atingido. É uma estratégia de planejamento escalonado onde o tempo de revisão e melhorias de partes do problema é pré-definido ([ANDRADE, 2014](#) apud [PRESSMAN; SOMMERVILLE, 2002, 2003](#)).

## 1.4 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma: O capítulo 2 apresenta conceitos e definições para a compreensão do escopo do trabalho e as ferramentas utilizadas; O capítulo 3 apresenta o protótipo, descreve a construção deste e o processo de desenvolvimento do algoritmo de contagem das células sanguíneas; No capítulo 4 são apresentados os resultados obtidos e as principais dificuldades encontradas; O capítulo 5 conclui o trabalho, apresentando as contribuições e trabalhos futuros.

## Parte II

### Referenciais Teóricos

## 2 Referenciais Teóricos

### 2.1 Processamento de Imagens

Nesta seção é descrita a fundamentação teórica sobre processamento digital de imagens, necessário para desenvolvimento do trabalho proposto. O conteúdo é baseado em Gonzalez ([GONZALEZ; WOODS, 2000](#)) e Andrade ([ANDRADE, 2014](#)).

O processamento de imagens digitais é uma área da computação que vem crescendo muito. Ela abrange as escalas de *hardware*, *software* e fundamentos teóricos.

Os conceitos são aplicados em duas categorias bem distintas: "(1) o aprimoramento de informações visuais para análise humana; e (2) a análise feita através de computador de informações obtidas de uma cena"([ANDRADE, 2014](#)).

As tarefas desempenhadas por um sistema de operações de processamento de imagens geralmente se dividem em: (1) aquisição, (2) armazenamento, (3) processamento, (4) comunicação e (5) exibição de imagens ([GONZALEZ; WOODS, 2000](#)).

A fase de **aquisição** é a responsável por capturar a imagem digital. Nela são necessários dois elementos: Um sensor de imageamento e um dispositivo capaz de digitalizar o sinal produzido pelo sensor. Um exemplo de equipamento para aquisição de imagens são as câmeras digitais.

Segundo Gonzalez ([GONZALEZ; WOODS, 2000](#)), um dos grandes desafios para projetos de sistemas de processamento de imagens é o **armazenamento**, devido à grande quantidade de memória requerida. As principais categorias desta etapa são: (1) armazenamento para uso durante o processamento, (2) armazenamento para acesso rápido e (3) arquivamento para acesso com menos frequência.

O **processamento** de imagens envolve tarefas algorítmicas, onde a maioria das funções podem ser implementadas em *software*. *Hardware* especializado é requerido somente para aumentar a velocidade da execução ou romper algumas barreiras fundamentais da computação.

A **comunicação** se refere à transferência de dados entre sistemas de processamento de imagens e a transmissão remota de um ponto a outro. A comunicação local é simples de ser executada pois existem *hardware* e *software* disponíveis para isso na maioria dos computadores. Já a transmissão a longas distâncias representa um grande desafio quando se quer enviar dados de imagens em vez de resultados abstratos. Isso se deve à grande quantidade de *bytes* a serem transferidos por canais de comunicação que nem sempre oferecem a velocidade necessária.

Atualmente a **exibição** de imagens pode ser feita por diversos dispositivos. O mais comum em sistemas modernos de processamento de imagens é o uso de monitores de TV, que podem ser monocromáticos ou coloridos. Dispositivos de impressão e tubos de raios catódicos de acesso aleatório são outras formas de exibição de imagens.

Uma das principais aplicações do processamento digital de imagens é a quantificação de objetos em uma imagem. Um "objeto" é qualquer elemento de interesse que pode ser identificado em uma figura. A variedade de problemas nesta categoria é muito ampla, cada um com suas peculiaridades e instigações inerentes (BARROS; ANDRADE; TRINDADE, 2019).

## 2.2 Fundamentos de Imagens Digitais

De acordo com Gonzalez (GONZALEZ; WOODS, 2000), uma imagem digital "é uma imagem  $f(x, y)$  discretizada tanto em coordenadas espaciais quanto em brilho". Ela pode ser considerada como sendo uma matriz onde cada coordenada identifica um ponto na imagem. Os elementos dessa matriz digital são chamados *pixels*.

Os valores armazenados em cada *pixel* são definidos de acordo com o tipo de imagem. Em uma imagem binária eles são restringidos a 0 ou 1. Em figuras em tons de cinza esses valores representam os níveis de cinza que variam de acordo com a quantidade de *bits* por *pixel*; numa imagem de 8 *bits*, por exemplo, estão disponíveis 256 níveis de cinza (GONZALEZ; WOODS, 2000). Já em figuras coloridas, os valores dos *pixels* variam conforme o espaço de cores utilizado e são representados por tuplas ternárias.

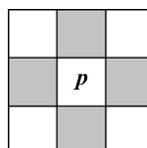
### 2.2.1 Os vizinhos de um *pixel*

Um *pixel*  $p$  na posição  $(x, y)$  possui quatro vizinhos horizontais e verticais definidos como:

$$(x + 1, y)(x - 1, y)(x, y + 1)(x, y - 1).$$

Este conjunto é denominado **vizinhança-de-4** de  $p$  (Figura 1) e é representado por  $N_4(p)$ . Se  $(x, y)$  estiver na borda da imagem digital, alguns dos vizinhos ficarão fora da figura (GONZALEZ; WOODS, 2000).

Figura 1 – Vizinhança-de-4 de  $p$ .



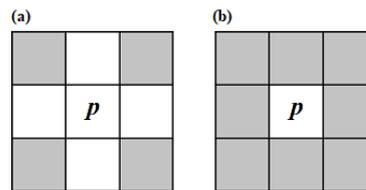
Fonte: O autor

Existem também quatro vizinhos diagonais de  $p$  (Figura 2 - (a)), dados por:

$$(x + 1, y + 1)(x + 1, y - 1)(x - 1, y + 1)(x - 1, y - 1)$$

Esses pontos são denotados por  $N_D(p)$  e, junto com a vizinhança-de-4, compõem a **vizinhança-de-8** de  $p$  (Figura 2 - (b)) representada por  $N_8(p)$  (GONZALEZ; WOODS, 2000).

Figura 2 – (a) Vizinhos Diagonais de  $p$ . (b) Vizinhança-de-8 de  $p$ .



Fonte: O autor

## 2.2.2 Conectividade

A conectividade de *pixels* é um conceito que define como os pontos em uma imagem digital se relacionam com seus vizinhos. A definição de bordas de objetos e componentes de regiões são feitas utilizando esta concepção.

Dois *pixels* estão conectados se eles são adjacentes e se suas cores satisfazem algum critério de similaridade. Gonzalez (GONZALEZ; WOODS, 2000) define os tipos de conectividade da seguinte forma:

Seja  $V$  o conjunto dos valores de níveis de cinza usados para definir conectividade; por exemplo, em uma imagem binária,  $V = \{1\}$  para a conectividade de *pixels* com valor 1. Em uma imagem em níveis de cinza, para conectividade de *pixels* com uma escala de valores de intensidade, digamos entre 32 e 64, segue que  $V = \{32, 33, \dots, 63, 64\}$ . Consideraremos três tipos de conectividade:

(a) conectividade-de-4. Dois *pixels*  $p$  e  $q$ , assumindo valores em  $V$ , são conectados-de-4 se  $q$  está no conjunto  $N_4(p)$ .

(b) conectividade-de-8. Dois *pixels*  $p$  e  $q$ , assumindo valores em  $V$ , são conectados-de-8 se  $q$  está no conjunto  $N_8(p)$ .

(c) conectividade-de- $m$  (conectividade mista). Dois *pixels*  $p$  e  $q$ , assumindo valores em  $V$ , são conectados-de- $m$  se:

(i)  $q$  está em  $N_4(p)$ , ou

(ii)  $q$  está em  $N_D(p)$  e o conjunto  $N_4(p) \cap N_4(q)$  for vazio. (Este é o conjunto dos *pixels* que são vizinhos-de-4 de  $p$  e  $q$ , e cujos valores estão em  $V$ ) (GONZALEZ; WOODS, 2000).

### 2.2.3 Rotulação de Componentes Conectados

A rotulagem de componentes conectados é uma abordagem para atribuir um conjunto finito de rótulos aos *pixels* de uma imagem, de modo que os rótulos correspondam a objetos na figura que estão sendo procurados.

O processo se dá com a imagem sendo varrida *pixel a pixel*, da esquerda para a direita e de cima para baixo. Na rotulagem de componentes conectados-de-4 são considerados o *pixel*  $p$ , que é qualquer pixel analisado no processo de varredura, e os *pixels*  $r$  e  $t$  que são os vizinhos superior e esquerdo de  $p$ , respectivamente. O padrão da varredura assegura que quando a análise chega a  $p$ , os pontos  $r$  e  $t$  já foram visitados. Com essas definições em foco, Gonzalez (GONZALEZ; WOODS, 2000) apresenta o algoritmo a seguir:

Se o valor de  $p$  é 0, mova para a próxima posição. Se o valor de  $p$  é 1, examine  $r$  e  $t$ . Se ambos forem 0, atribua a  $p$  um novo rótulo [...]. Se apenas um dos vizinhos for 1, atribua a  $p$  o seu rótulo. Se ambos forem 1 e possuem o mesmo rótulo, atribua a  $p$  aquele rótulo. Se ambos forem 1, mas possuem rótulos diferentes, atribua um dos rótulos a  $p$  e anote que os dois rótulos são equivalentes (isto é, os pontos  $r$  e  $t$  estão conectados por  $p$ ) (GONZALEZ; WOODS, 2000).

Quando a varredura terminar, todos os pontos com valor 1 terão sido marcados. Os pares de rótulos equivalentes deverão ser ordenados em classes de equivalência. Essas classes recebem rótulos, e posteriormente a imagem deve ser percorrida novamente para trocar cada rótulo pelo de sua respectiva classe (GONZALEZ; WOODS, 2000).

A rotulagem de componentes conectados-de-8 é feita através do mesmo processo, o que muda é que, além de  $r$  e  $t$ , são examinados os dois vizinhos diagonais superiores de  $p$ , denotados por  $q$  e  $s$ . Neste caso, o padrão da varredura também garante que todos esses vizinhos já terão sido processados quando a análise tiver chegado em  $p$ . O algoritmo descrito por Gonzalez (GONZALEZ; WOODS, 2000) é o seguinte:

Se  $p$  for 0, mova para a próxima posição. Se  $p$  for 1 e todos os seus quatro vizinhos forem 0, atribua a  $p$  um novo rótulo. Se apenas um dos vizinhos for 1, atribua a  $p$  o seu rótulo. Se dois ou mais vizinhos forem 1, atribua um dos rótulos a  $p$  e anote as equivalências apropriadas. Após terminar de percorrer a imagem, ordene os pares de rótulos equivalentes em classes de equivalência, atribua um rótulo único a cada classe e percorra a imagem uma segunda vez, trocando-se cada rótulo pelo atribuído a sua classe de equivalência (GONZALEZ; WOODS, 2000).

## 2.3 Segmentação de Imagens

A segmentação é a primeira etapa na análise de imagens. Essa tarefa tem o objetivo de repartir a imagem em suas partes ou objetos constituintes. A subdivisão é feita até que as regiões de interesse da figura estejam isoladas (GONZALEZ; WOODS, 2000).

A segmentação autônoma é uma das tarefas mais complicadas em processamento de imagens e pode determinar o sucesso ou o fracasso de uma análise. Os algoritmos que realizam esse trabalho em imagens monocromáticas se baseiam na descontinuidade e na similaridade; estas são propriedades básicas de valores de níveis de cinza (GONZALEZ; WOODS, 2000).

Algoritmos que se respaldam na descontinuidade particionam a imagem com base em mudanças bruscas nos níveis de cinza. Essa abordagem é útil na detecção de pontos isolados e detecção de linhas e bordas na figura. Já os que se fundamentam na similaridade utilizam limiarização e técnicas orientadas a regiões como crescimento, divisão e fusão de regiões (GONZALEZ; WOODS, 2000).

### 2.3.1 Limiarização

A limiarização é uma abordagem para segmentação de imagens. Essa operação segmenta uma imagem digital em tons de cinza com base nas características dos seus *pixels* como, por exemplo, o nível de intensidade de cinza. O objetivo é criar uma representação binária da figura. Isso é feito através da escolha de um limiar  $T$  que separa os *pixels* em dois grupos. Assim, em cada ponto  $(x, y)$ , se  $f(x, y) > T$ , o ponto é convertido para branco (recebe o valor 1), caso contrário é transformado em preto (recebe o valor 0) (GONZALEZ; WOODS, 2000). Por gerar uma imagem binária como saída, o processo também é chamado de **binarização** (ANDRADE, 2014).

A imagem  $g(x, y)$  binarizada pode ser definida como:

$$g(x, y) = \begin{cases} 1 & \text{se } f(x, y) > T \\ 0 & \text{se } f(x, y) \leq T \end{cases}$$

em que os *pixels* marcados com 1 (brancos) compõem objetos e os etiquetados com 0 (pretos) fazem parte do fundo (ANDRADE, 2014).

#### 2.3.1.1 Limiarização global simples

Esta é a técnica mais básica de limiarização e consiste na classificação de todos os *pixels* da imagem com base no mesmo valor  $T$  de limiar. Nessa abordagem, para cada *pixel*, se seu nível de cinza for menor que o limite, ele será definido como 0 (fundo), caso contrário, será definido como 1 (objeto).

Esse método não apresenta bons resultados quando a figura de entrada possui diferentes níveis de iluminação, sendo mais utilizado em imagens que são capturadas em ambientes controlados (GONZALEZ; WOODS, 2000).

### 2.3.1.2 Limiarização Adaptativa

A limiarização adaptativa consiste no cálculo de um valor  $T$  diferente para cada *pixel* na imagem. Assim, diferentes regiões da figura são classificadas com limites distintos e isso culmina em resultados melhores para imagens com iluminação variável.

Existem vários métodos que utilizam essa abordagem, cada um com suas particularidades. Bradeley e Roth (BRADLEY; ROTH, 2007) desenvolveram uma técnica de limiarização adaptativa em tempo real que usa tabelas de áreas somadas (imagens integrais) para fazer o cálculo do limiar; Sauvola e Pietikäinen (SAUVOLA; PIETIKÄINEN, 2000) apresentaram um método para binarização adaptativa de imagens de documentos. Outros exemplos podem ser encontradas em (WHITE; ROHRER, 1983; BERNSEN, 1986; PARKER, 1991; WELLNER, 1993; SHEN; IP, 1997; SAVAKIS, 1998; YANG; YAN, 2000).

### 2.3.2 O algoritmo *Watershed*

A transformada da bacia hidrográfica (*Watershed transform*) é uma ferramenta poderosa para a segmentação de imagens (BEUCHER et al., 1992). O método foi originalmente proposto por Digabel (DIGABEL; LANTUÉJOUL, 1978) e Lantuéjoul (LANTUÉJOUL, 1978) e posteriormente aprimorado por Beucher (BEUCHER; LANTUÉJOUL, 1979).

Segundo Roerdink e Meijster (ROERDINK; MEIJSTER, 2003), o algoritmo *Watershed* pode ser classificado como uma abordagem de segmentação orientada a regiões. Eles afirmam que a ideia intuitiva subjacente a este método vem da geografia: é a de uma paisagem ou relevo topográfico que é inundado pela água, sendo as bacias hidrográficas as linhas divisórias dos domínios de captação da chuva que cai sobre a região. Uma abordagem alternativa é imaginar uma paisagem com buracos perfurados em mínimos locais sendo imersa em um lago. As bacias se encherão de água a partir desses mínimos locais e, nos pontos onde a água que vem de bacias diferentes se encontrariam, as barragens são construídas. Quando o nível da água atinge o pico mais alto na paisagem, o processo é interrompido. O resultado é uma paisagem dividida em regiões ou bacias separadas por barragens, denominadas linhas de bacias hidrográficas. A ideia se aplica ao processamento de imagens pois qualquer imagem em escala de cinza pode ser vista como uma superfície topográfica onde valores mais altos de *pixels* representam picos e colinas, enquanto valores baixos equivalem a vales (mínimos locais).

De acordo com Beucher (BEUCHER et al., 1992), esse algoritmo muitas vezes leva à segmentação em excesso da imagem, seja por ruído ou qualquer outra irregularidade. Para contornar este problema, Meyer e Beucher (MEYER; BEUCHER, 1990) propuseram uma melhoria do *Watershed*, chamada segmentação controlada por marcador. Essa abordagem é aplicada da seguinte forma: primeiro, são definidas as propriedades que serão usadas para marcar os objetos na imagem. O mesmo é feito para o fundo, ou seja, para partes da

imagem na qual há a certeza de que não há *pixel* pertencente a algum objeto (BEUCHER et al., 1992). Em resumo, o algoritmo simula a inundação somente nos objetos marcados, evitando a segmentação em excesso.

## 2.4 Morfologia Matemática

A morfologia matemática é uma técnica não linear para análise espacial de estruturas. Ela foi proposta inicialmente por George Matheron (MATHERON, 1967) e Jean Serra (SERRA, 1969) na década de 1960. Por ser baseada na teoria dos conjuntos, oferece uma abordagem poderosa e unificada para o processamento de imagens. Pode ser aplicada na realização de segmentação, realce, detecção de bordas, afinamento, filtragem, dentre outras.

Segundo Andrade (ANDRADE, 2014), a morfologia matemática extrai informações referentes à geometria e à topologia de uma imagem interpretando-a como um conjunto desconhecido. A extração é realizada pela transformação através de outro conjunto cuja forma e tamanho são conhecidos, denominado **elemento estruturante**. A seguir são apresentados alguns conceitos importantes de morfologia matemática.

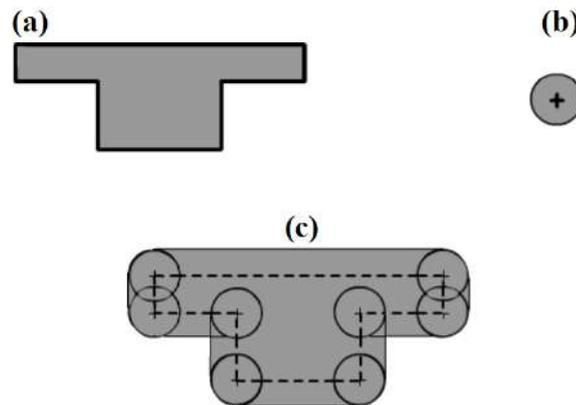
### 2.4.1 Dilatação

Gonzalez (GONZALEZ; WOODS, 2000) define formalmente a dilatação de um conjunto desconhecido  $A$  por um elemento estruturante  $B$  (ambos pertencentes ao espaço bidimensional  $Z^2$ ) como sendo "o conjunto de todos os  $x$  deslocamentos para os quais a interseção de  $(\hat{B})_x$  e  $A$  inclui pelo menos um elemento diferente de zero", podendo ser escrita da seguinte forma:

$$A \oplus B = \{x | (\hat{B})_x \cap A \neq \emptyset\}$$

onde  $(\hat{B})_x$  é a reflexão de  $B$  em relação a  $x$ .

Em outras palavras, o elemento estruturante passa por todos os *pixels* da figura. Se houver alguma interseção deste com um objeto na imagem, o ponto na figura que corresponde ao *pixel* central do elemento estruturante é preenchido, caso contrário será ignorado. A operação de dilatação em uma imagem faz com que todos os objetos na figura se expandam de acordo com o elemento estruturante utilizado. Ela permite conectar itens separados e preencher furos. A Figura 3 mostra uma interpretação geométrica de dilatação sobre uma imagem bidimensional.

Figura 3 – (a) Imagem  $A$ . (b) Elemento Estruturante  $B$ . (c) Dilatação de  $A$  por  $B$ .

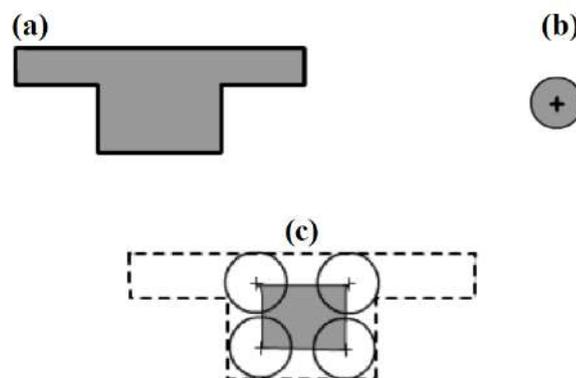
Fonte: (MACHADO, 2008)

### 2.4.2 Erosão

A erosão de um conjunto  $A$  por um elemento estruturante  $B$  em  $Z^2$  é "o conjunto de todos os pontos  $x$  tais que  $B$ , quando transladado por  $x$  fique contido em  $A$ " (GONZALEZ; WOODS, 2000). Formalmente, é denotada por:

$$A \ominus B = \{x | (B)_x \subseteq A\}.$$

No processo de erosão, como na dilatação, o elemento estruturante percorre todos os pontos da imagem. Se este estiver inteiramente contido em algum objeto, o ponto na figura que corresponde ao *pixel* central do elemento estruturante é preservado, caso contrário ele é "descartado". A erosão aplicada em uma imagem é análoga à erosão do solo, ela "corrói" pedaços dos objetos de acordo com o elemento estruturante utilizado. Essa operação separa objetos que estiverem próximos e aumenta furos interiores aos conjuntos. Uma interpretação geométrica é mostrada na Figura 4.

Figura 4 – (a) Imagem  $A$ . (b) Elemento Estruturante  $B$ . (c) Erosão de  $A$  por  $B$ .

Fonte: (MACHADO, 2008)

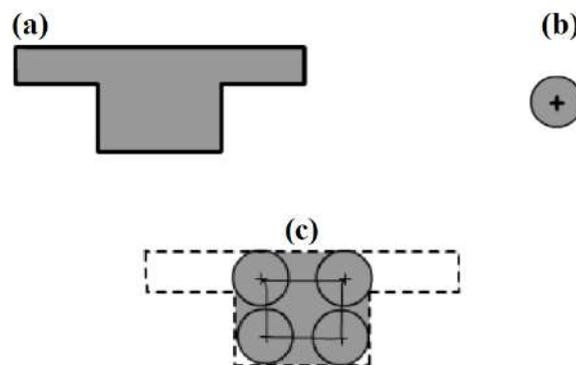
### 2.4.3 Abertura

Uma abertura de  $A$  por  $B$  nada mais é do que uma erosão de  $A$  pelo elemento estruturante  $B$ , seguida por uma dilatação do resultado por  $B$ . A definição matemática é a seguinte:

$$A \circ B = (A \ominus B) \oplus B.$$

A operação de abertura é útil para remoção de ruído, suavização de contornos, quebra de istmos estreitos e na eliminação de protusões finas (GONZALEZ; WOODS, 2000). O processo é mostrado na Figura 5.

Figura 5 – (a) Imagem  $A$ . (b) Elemento Estruturante  $B$ . (c) Abertura de  $A$  por  $B$ .



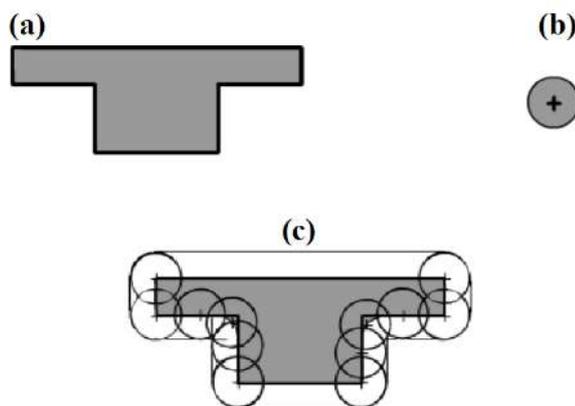
Fonte: (MACHADO, 2008)

### 2.4.4 Fechamento

O fechamento morfológico é a operação inversa da abertura. Consiste na dilatação de  $A$  pelo elemento estruturante  $B$ , seguida por uma erosão do resultado por  $B$ . É denotado por:

$$A \bullet B = (A \oplus B) \ominus B.$$

Assim como a abertura, o fechamento também é útil na remoção de ruídos e suavização de contornos, mas, ao contrário da primeira, funde quebras em golfos finos, elimina pequenos buracos e preenche fendas em um contorno (GONZALEZ; WOODS, 2000). A Figura 6 mostra um exemplo de fechamento.

Figura 6 – (a) Imagem *A*. (b) Elemento Estruturante *B*. (c) Fechamento de *A* por *B*.

Fonte: (MACHADO, 2008)

## 2.5 A Biblioteca OpenCV

OpenCV é uma biblioteca de visão computacional e processamento de imagens totalmente livre para uso acadêmico e comercial projetada para fornecer código portátil e otimizado. Ela possui versões para C, C++, Python, Java e outras linguagens.

Ao oferecer uma infraestrutura simples de usar, o OpenCV permite que sejam criadas aplicações razoavelmente sofisticadas de maneira rápida. A biblioteca contém mais de 500 funções que abrangem muitas áreas da visão computacional, incluindo inspeção de produtos, reconhecimento de objetos, imagens médicas, segurança, interface de usuário, calibração de câmera, visão estéreo, robótica, dentre outras (BRADSKI; KAEHLER, 2008). Há também funções para filtragem, transformações morfológicas, realce, segmentação, rotulagem de componentes conectados e outras operações de processamento de imagens.

## 2.6 Impressão 3D

A impressão 3D é uma técnica de fabricação que funciona basicamente através da adição de camadas sobrepostas de materiais (como metal ou plástico) para construir uma peça tridimensional. O objeto a ser impresso precisa ser criado em um software de edição em três dimensões, onde é possível desenvolver várias formas.

Um dos principais usos da impressão 3D é na engenharia para a criação de protótipos. Recentes avanços nos materiais de impressão permitiram a criação de itens comparáveis aos fabricados tradicionalmente. Em contraste com as impressoras convencionais, baseadas em tinta que produzem um objeto bidimensional (tinta sobre papel), a impressão 3D tem o potencial de permitir a personalização em massa de mercadorias em grande escala (SCHUBERT; LANGEVELD; DONOSO, 2014).

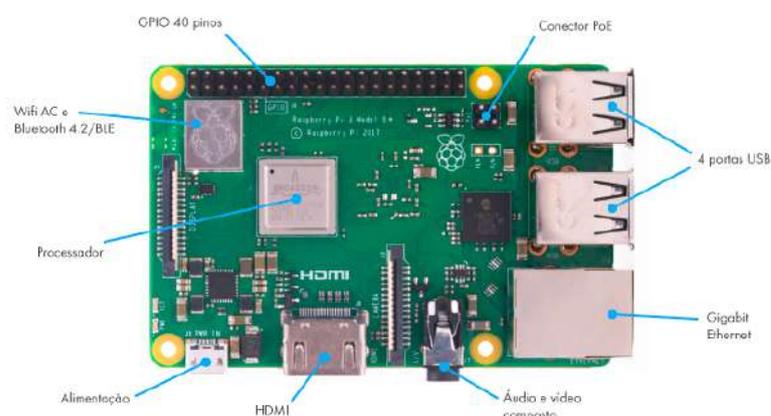
## 2.7 Raspberry Pi

O Raspberry Pi é um computador de baixo custo desenvolvido no Reino Unido pela Fundação Raspberry<sup>1</sup>. A função básica do dispositivo é oferecer uma alternativa barata, prática e acessível para que pessoas possam explorar as capacidades da computação.

É um computador como outro qualquer, apesar do tamanho diminuto e das feições pouco convencionais; possui portas USB, HDMI, Ethernet, áudio e vídeo composto e, nas versões mais recentes, conectividade sem fios através de Bluetooth e Wi-Fi. Além disso, vem equipado com pinos GPIO (Pinos de Entrada e Saída de Uso Geral). Estes pinos são responsáveis por fazer a comunicação de entrada e saída de sinais digitais. Tarefas como controle de motores, acionamento de LEDs, leituras de sensores e botões, dentre outras, são feitas através deles.

Neste projeto foi utilizado um Raspberry Pi 3 modelo B+. Essa versão, apresentada na Figura 7, conta com 1GB de memória RAM, conectividade *wireless*, processador de 1.4GHz e 40 pinos GPIO.

Figura 7 – Raspberry Pi 3 modelo B+



Fonte: <<https://blog.fazedores.com/nova-raspberry-pi-3-b-plus/>>

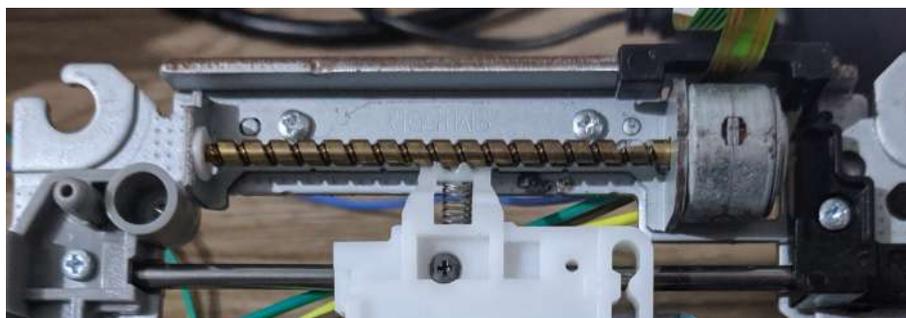
## 2.8 Motores de Passo

Brites e Santos (BRITES; SANTOS, 2008) descrevem motores de passo como sendo "dispositivos eletro-mecânicos que convertem pulsos elétricos em movimentos mecânicos que geram variações angulares discretas". Os "passos" são os incrementos angulares provocados pelos pulsos elétricos que fazem o rotor ou eixo rotacionar. A velocidade que o rotor gira depende da frequência em que os impulsos elétricos são recebidos, a sequência em que estes impulsos são aplicados definem a direção para qual o eixo gira e a quantidade de pulsos

<sup>1</sup> Ver <<https://www.raspberrypi.org/>>. Acesso em 01 de Ago de 2019.

define o tamanho do ângulo a ser rotacionado. Este tipo de motor é ideal para aplicações que exigem posicionamento preciso, pois controlam o ângulo de rotação, a velocidade, a posição e o sincronismo.

Figura 8 – Motor de Passo



Fonte: O autor

### 2.8.1 Ponte H

Uma ponte H é um tipo de circuito que pode determinar o sentido da corrente que flui através de uma carga. Quando utilizada com motores de passo, permite que o eixo destes rode tanto para um sentido quanto para o outro. Estes circuitos geralmente estão disponíveis em módulos prontos, mas também podem ser construídos por componentes. O nome "ponte H" se deve ao desenho típico deste circuito ([BRITES; SANTOS, 2008](#)).

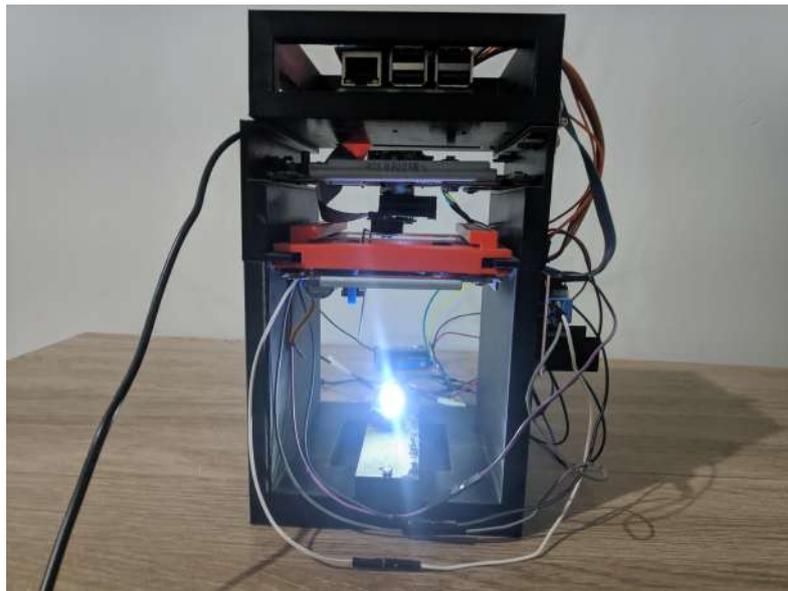
## Parte III

O Protótipo para Contagem Automatizada de  
Células Sanguíneas usando Processamento de  
Imagens

### 3 O Protótipo para Contagem Automatizada de Células Sanguíneas usando Processamento de Imagens

O protótipo para contagem automática das células sanguíneas desenvolvido neste trabalho consiste em um pequeno equipamento dotado de uma câmera com lente de aumento acoplada e uma gaveta para inserção de uma lâmina de microscopia com amostra de sangue humano. O dispositivo, mostrado nas Figuras 9 e 10, utiliza lâminas com esfregaços finos em vez de volume de sangue para realizar a análise.

Figura 9 – Protótipo - Visão frontal.

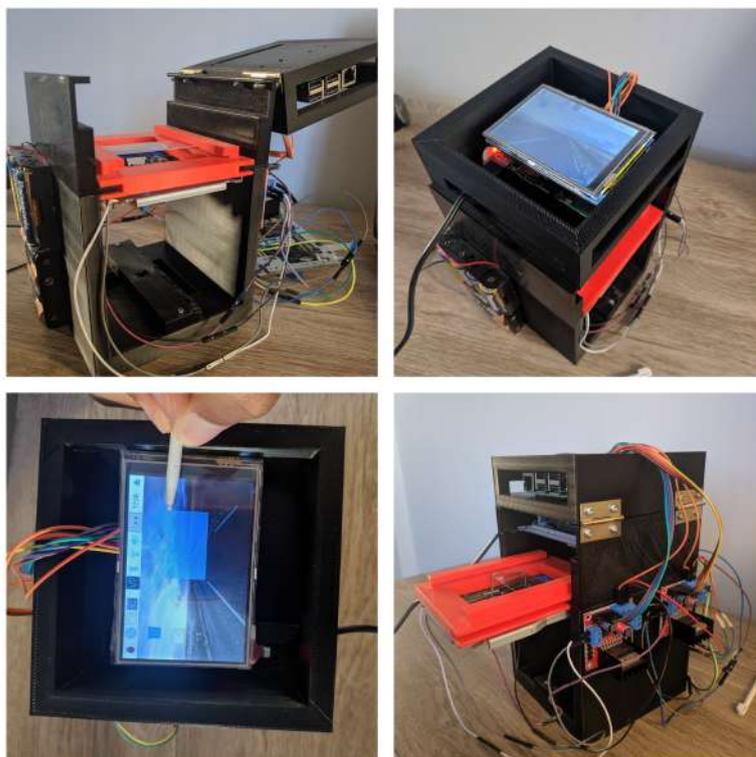


Fonte: O autor

Funciona da seguinte maneira: Após ser inserida a lâmina de forma apropriada na gaveta do dispositivo, este inicia um processo de captura de 20 fotos do esfregaço sanguíneo na lâmina. Posteriormente essas imagens são processadas por um algoritmo que faz uma varredura em cada foto seguindo um conjunto das regras e procedimentos lógicos perfeitamente definidos que permitem a contagem das células em cada imagem. Os resultados obtidos são somados e o cômputo é exibido na tela.

O equipamento utiliza como central de processamento um Raspberry Pi 3 modelo B+ com o sistema operacional Raspbian. O microcomputador é o responsável por controlar todos os módulos do protótipo.

Figura 10 – Protótipo.



Fonte: O autor

## 3.1 Construção do Equipamento

Nesta seção está descrito o processo de construção da parte física do dispositivo. São detalhadas todas as etapas, desde a modelagem da parte externa até a acoplagem ao Raspberry Pi dos módulos responsáveis pela aquisição automatizada das imagens das células sanguíneas.

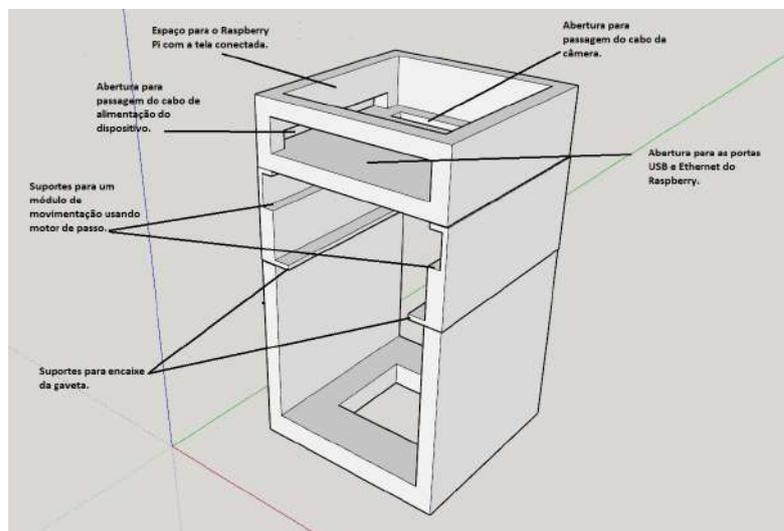
### 3.1.1 Modelagem do Invólucro e da Gaveta

Toda a estrutura do protótipo foi desenvolvida utilizando técnicas de modelagem 3D. O plano inicial era construir o equipamento de madeira ou acrílico, mas no fim decidiu-se utilizar a modelagem 3D devido a sua versatilidade, já que os modelos virtuais podem ser facilmente visualizados em perspectiva e alterados de acordo com a necessidade. Além disso, as peças podem ser construídas com bastante precisão e trazidas ao mundo real através de uma impressora 3D.

O invólucro do protótipo, mostrado na Figura 11, foi projetado para acomodar o Raspberry Pi e os módulos conectados a ele. A parte superior abriga o microcomputador e sua tela; neste local há aberturas para passagem dos cabos do módulo da câmera e da alimentação do dispositivo e para as portas USB e Ethernet do Raspberry. Logo abaixo

desta área do envoltório há um suporte para a conexão de um módulo de movimentação por motor de passo que serve para deslocar uma câmera com lente de aumento.

Figura 11 – Modelo 3D do Invólucro do Protótipo.



Fonte: O autor

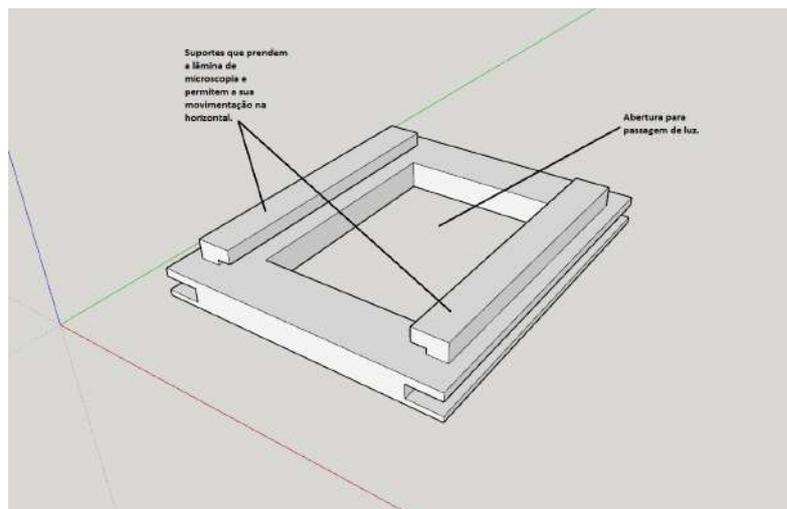
O invólucro também dá suporte para encaixe de uma gaveta. Esta foi projetada especificamente para acomodar uma lâmina de microscopia com amostra de sangue humano sendo análoga a uma platina de um microscópio tradicional. O design da gaveta permite que a lâmina seja movimentada na horizontal no eixo Y. A movimentação é necessária para permitir a captura de imagens de diferentes partes da amostra. Este processo será detalhado nas próximas seções. Esta gaveta também possui uma abertura central para permitir a passagem de luz para a lâmina. Seu modelo tridimensional pode ser visto na Figura 12.

A gaveta fica suspensa na estrutura do invólucro porque abaixo dela foi deixado um espaço para o posicionamento de uma fonte luminosa. O esquema desenvolvido para capturar imagens ampliadas neste projeto funciona através da captação de luz em uma área minúscula da lente, bem iluminada, a uma distância que permita que a câmera ligada ao Raspberry, posicionada acima da lâmina, consiga focalizar as células da amostra.

Para imprimir o invólucro foi necessário dividir o modelo 3D em partes menores, como mostrado na Figura 13.

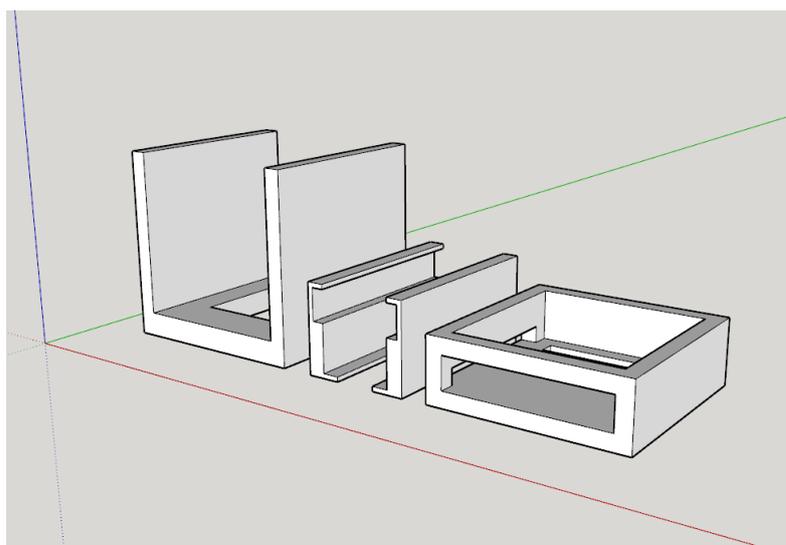
A gaveta e as outras peças foram impressas em plástico. Todos os componentes foram colados para montar novamente a estrutura original, com exceção da parte superior, que no invólucro real permaneceu solta, como uma tampa, para permitir a manutenção dos itens que viriam a ser instalados na parte interna do dispositivo.

Figura 12 – Modelo 3D da Gaveta do Protótipo.



Fonte: O autor

Figura 13 – Modelo Tridimensional do Invólucro Dividido.



Fonte: O autor

### 3.1.2 Instalação da Tela

A instalação de um display *touchscreen* fez-se necessária para garantir a mobilidade do projeto com o Raspberry Pi, já que reúne na mesma placa as funções de teclado, mouse e display e o objetivo desde o princípio era construir um equipamento portátil.

A tela escolhida foi um modelo genérico de 3.5 polegadas LCD TFT, com resolução de 480x320 *pixels*, resistiva. O display, mostrado na Figura 14, se encaixa nas portas GPIO do Raspberry Pi, ocupando os pinos do 1 ao 26, deixando os outros livres para a utilização do usuário.

Figura 14 – Tela *Touchscreen*.



Fonte: O autor

Por padrão o Raspbian exibe a sua interface gráfica através da porta HDMI do microcomputador, logo, para fazer com que a tela touchscreen exibisse alguma informação e funcionasse como o esperado foi necessário realizar a instalação de um *driver* para redirecionar o fluxo de exibição da porta HDMI para a GPIO.

Foi utilizado o *driver* LCD-show, disponível para download gratuitamente em um repositório do Github<sup>1</sup>. Este driver também realiza a calibração da tela sensível ao toque.

Após a calibragem inicial, pôde-se notar que a tela funciona como o desejado. Apesar de ser de tecnologia resistiva, não é preciso muita força no “clique” para acessar os menus do Raspbian, por exemplo. Alguns ícones e softwares aparecem cortados ou sobrepostos na tela devido a limitações do próprio sistema operacional do Raspberry, que não consegue lidar com telas muito pequenas. A tela funciona tanto com o dedo como com a caneta *touchscreen* ou algum objeto parecido.

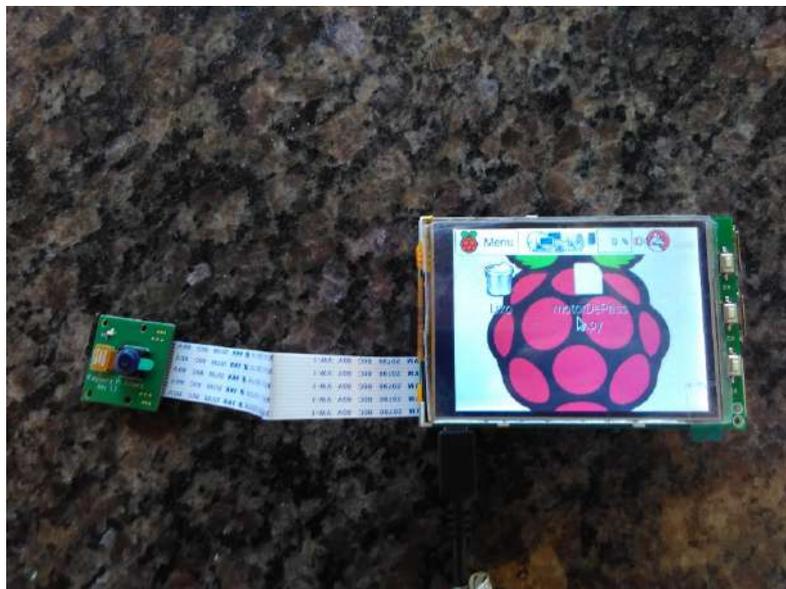
### 3.1.3 Instalação da Câmera e da Lente

Para aquisição automatizada das imagens das células sanguíneas, foi instalada uma câmera no Raspberry Pi, como pode ser visto na Figura 15. Ela é de um modelo desenvolvido especificamente para este microcomputador e pode de capturar imagens de até 5 *megapixels*. A vantagem dessa câmera é que esta utiliza o próprio conector presente no Raspberry Pi, não ocupando nenhuma porta USB na placa. Nenhuma configuração ou programação específica foi necessária para este módulo. Os próprios hardware e software

<sup>1</sup> Ver <<https://github.com/goodtft/LCD-show>>. Acesso em 10 de Jul de 2019.

do Raspberry Pi dão suporte nativo para a câmera, bastando ativar a funcionalidade nas configurações do Raspbian OS.

Figura 15 – Raspberry Pi com Câmera e Tela conectadas.



Fonte: O autor

Para a obtenção de imagens microscópicas utilizou-se em conjunto com a câmera a Micro Phone Lens<sup>2</sup> (Figura 16), uma lente desenvolvida para smartphones e tablets que consegue aumento de 150x antes do zoom digital. Ela foi escolhida por ser pequena, de fácil manipulação, e se adequar aos requisitos de mobilidade do protótipo, visto a necessidade da ampliação das imagens para que as células sanguíneas pudessem ser visualizadas, e a inviabilidade de instalação de lentes objetivas e oculares presentes em microscópios tradicionais.

Essa lente adere estaticamente à superfícies lisas de vidro e plástico, como as proteções das câmeras de celulares, por exemplo. Contudo, a câmera utilizada neste projeto possui um revestimento em plástico fosco que impediu a adesão da lente diretamente. Por este motivo, foi idealizado um suporte que contém uma película de plástico liso e transparente para ser utilizado como base de ligação para a lente. Ele fica posicionado imediatamente em frente à entrada de luz da câmera, garantindo a aderência e posicionamento adequados da lente para a captura de imagens das células. O suporte foi desenvolvido em 3D e impresso em plástico. O modelo tridimensional é mostrado na Figura 17.

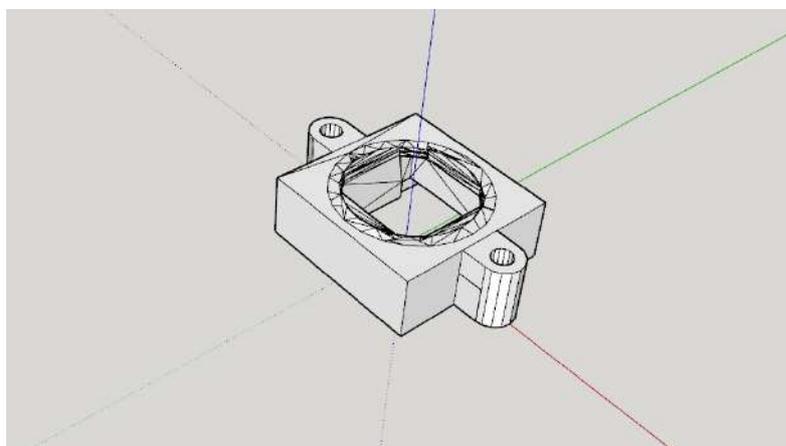
<sup>2</sup> Ver <<https://www.microphonelens.com/>>. Acesso em 17 de Jul. de 2019.

Figura 16 – Micro Phone Lens



Fonte: <<https://www.microphonelens.com/products/150x-micro-phone-lens-starter-kit>>

Figura 17 – Modelo Tridimensional do Suporte para a Lente.



Fonte: O autor

A câmera com a lente foi parafusada a um dos módulos de movimentação do protótipo. A movimentação da câmera é necessária para obter imagens de diferentes partes da amostra.

#### 3.1.4 Instalação dos Módulos de Movimentação com Motores de Passo

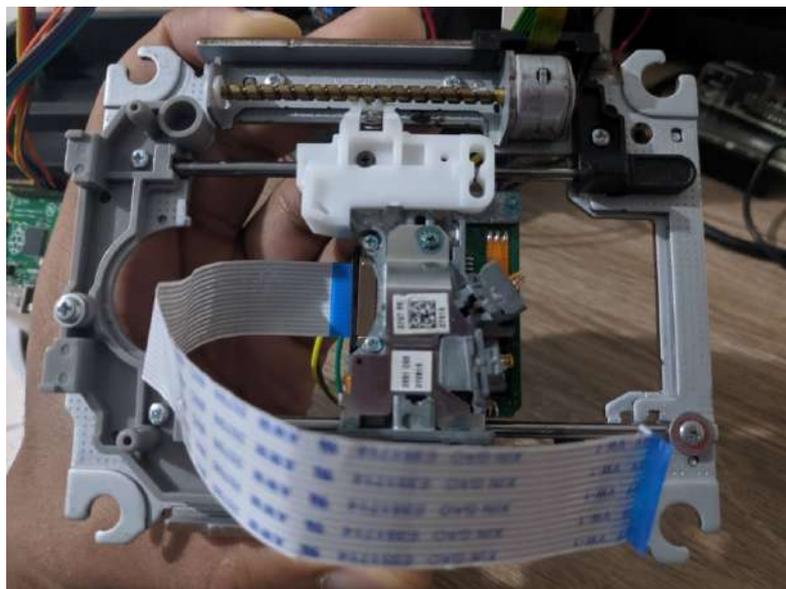
Os motores de passo são utilizados para se obter um posicionamento preciso por controle digital. "O ponto forte de um motor de passo não é a sua força (torque), tampouco sua capacidade de desenvolver altas velocidades - [...] - mas sim a possibilidade de controlar seus movimentos de forma precisa" (BRITES; SANTOS, 2008).

Neste projeto foram utilizados dois motores de passo: um para movimentar a câmera com a lente acoplada no eixo X e o outro para mover a lâmina com amostra de sangue no eixo Y. Dessa forma, foi possível sincronizar via código a movimentação dos motores nos dois eixos e capturar imagens de segmentos distintos da amostra na lâmina para garantir maior acurácia nos resultados do processamento.

Os motores foram aproveitados de unidades de CD-ROM. Os trilhos, os eixos espirais, estruturas de aço e alguns outros componentes internos dos drives também foram reciclados para construir os módulos de movimentação do protótipo. O mecanismo funciona da seguinte maneira: Cada motor possui um eixo em espiral preso ao seu rotor, posicionado paralelamente aos trilhos de uma estrutura reciclada de aço e há, conectada à espiral, uma armação de plástico com pequenos componentes metálicos que desliza sobre os trilhos quando a espiral é girada pelo rotor. O que estiver preso a essa armação é movimentado junto com ela. O sentido do movimento varia com o posicionamento e o sentido de rotação do rotor. Este mecanismo é semelhante ao utilizado para abrir e fechar gavetas de CD-ROM, aparelhos de DVD etc.

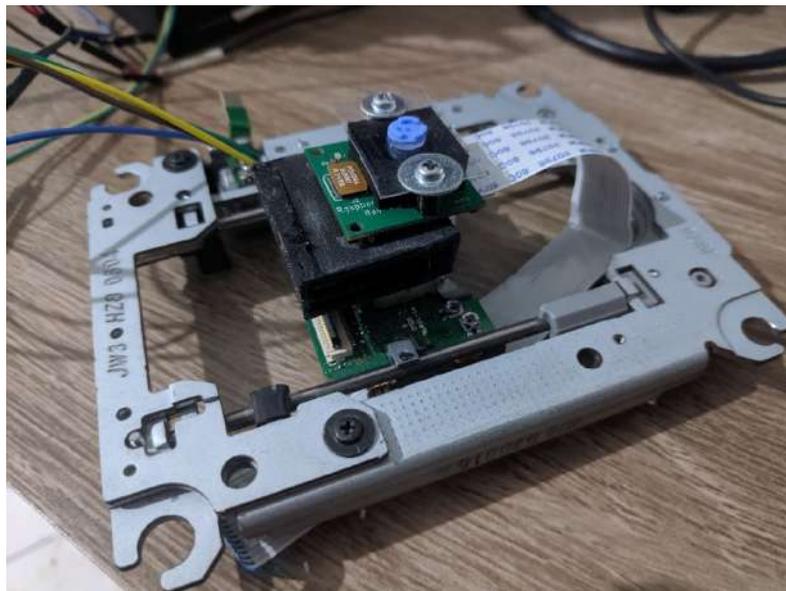
O módulo de movimentação equipado com o motor responsável pelo deslocamento da câmera com a lente no eixo X foi posicionado dentro do invólucro do protótipo no suporte desenvolvido para este fim, mostrado na Figura 11. A câmera foi parafusada à armação de plástico do mecanismo de movimentação. Confira as Figuras 18 e 19.

Figura 18 – Módulo de Movimentação da Câmera com a Lente - Vista Traseira.



Fonte: O autor

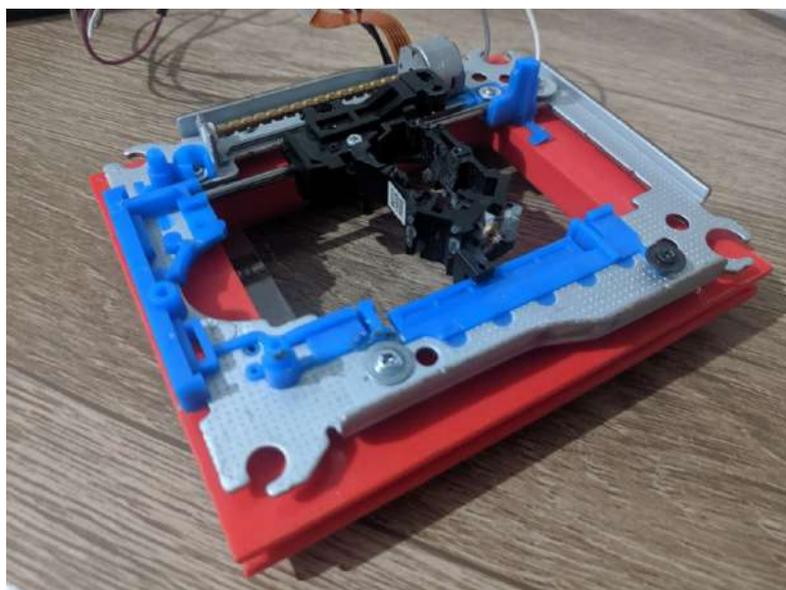
Figura 19 – Módulo de Movimentação da Câmera com a Lente - Vista Frontal.



Fonte: O autor

Um segundo módulo com o motor (Figura 20) responsável pelo deslocamento da lâmina no eixo Y foi parafusado à gaveta do invólucro, na parte oposta aos suportes que seguram a lâmina.

Figura 20 – Módulo de Movimentação da Lâmina.



Fonte: O autor

O mecanismo de movimentação no eixo das ordenadas possui alguns detalhes especiais: pelo fato da lâmina ser removível, não é possível fixá-la permanentemente à armação de plástico que se movimenta pelos trilhos, assim, para movê-la foi desenvolvida

uma chave metálica (Figura 21) em forma de gancho que se prende na armação de plástico e arrasta a lâmina forçando sua movimentação sobre a gaveta. Na parte central desta armação de plástico também há um furo que garante a passagem de luz para a lente da câmera.

Figura 21 – Chave Metálica para Movimentação da Lâmina.



Fonte: O autor

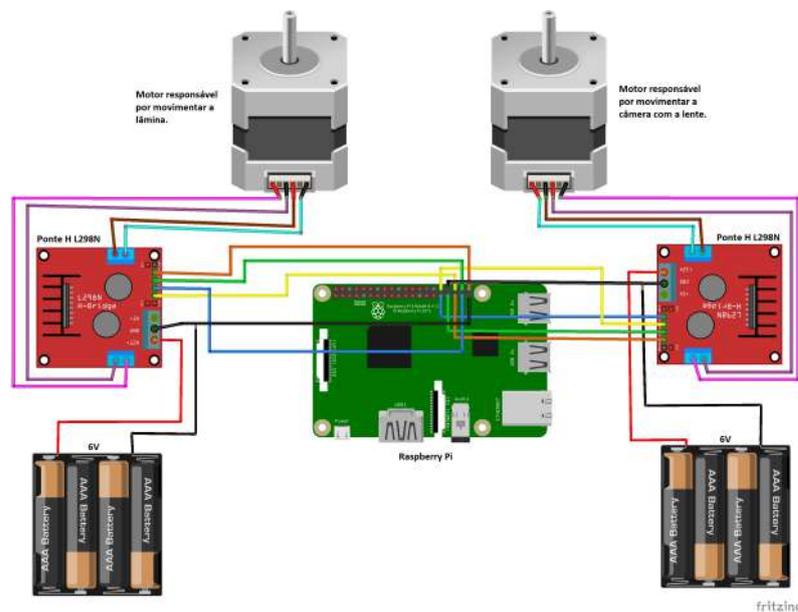
O circuito de controle dos motores é baseado no Raspberry Pi, e duas pontes H L298N foram usadas como *drivers*. Por terem sido desenvolvidos para outro tipo de equipamento, os motores não possuíam fios para conectividade adequada às pontes H, sendo necessária a soldagem de cabos *jumpers* às suas entradas de controle e alimentação. Cada motor possui quatro fios de controle que são conectados em seus respectivos *drivers*. A Figura 22 mostra o diagrama de conexão do controle dos motores de passo ao Raspberry Pi usando L298N. A modelagem do diagrama foi feita usando a ferramenta de código aberto Fritzing<sup>3</sup>.

As entradas para módulo *driver* responsável por controlar o motor que movimenta a câmera, aqui nomeadas IN1, IN2, IN3 e IN4, foram conectadas aos pinos físicos 31, 32, 33 e 35, isto é, GPIO6, GPIO12, GPIO13 e GPIO19 do Raspberry Pi. As entradas para o *driver* que controla o motor incumbido da movimentação da lâmina ficaram nos pinos 36, 37, 38 e 40, ou seja, GPIO16, GPIO26, GPIO20 e GPIO21 respectivamente.

Um conjunto das bobinas de cada motor está conectado a OUT1 e OUT2 do seu respectivo *driver* e o outro conjunto está conectado a OUT3 e OUT4. Uma fonte de alimentação externa ao Raspberry foi necessária para os motores. Para este fim, foram utilizadas oito pilhas alcalinas AA de 1,5V, quatro para cada motor. Elas foram agrupadas

<sup>3</sup> Ver <<https://fritzing.org/home/>>. Acesso em 10 de Mai. de 2019.

Figura 22 – Diagrama do Circuito do Controle dos Motores.



Fonte: O autor

em dois suportes separados e cada um é capaz de fornecer até 6V para o *driver* que estiver alimentando. Os terminais de aterramento do módulo do acionador do motor e do microcomputador são comuns. Uma fonte separada de 5V é usada para alimentar o Raspberry Pi.

A programação dos motores é feita por uma função escrita na linguagem Python. O código utilizado neste projeto é baseado no disponível no site Eletronics Hub<sup>4</sup> e faz uso da biblioteca **RPi.GPIO**<sup>5</sup> para ter controle completo da GPIO. Para aumentar a acurácia do motor de passo, ou seja, para obter um posicionamento angular do eixo um pouco mais preciso, o algoritmo utiliza uma técnica chamada “*Half Stepping*”. Este método consiste em deslocar o rotor em meio passo de cada vez, fazendo com que seja duplicado o número de passos necessários para completar uma volta.

A função de controle dos motores recebe como parâmetros os pinos da GPIO em que o motor a ser ativado está conectado e quantidade de passos a ser dada em números inteiros. Um valor positivo para a quantia de passos faz com que o rotor gire no sentido horário e um valor negativo faz com que ele se desloque no sentido anti-horário.

### 3.1.5 Instalação da Lâmpada de Retroiluminação

Como nos microscópios tradicionais, no protótipo também há a necessidade de iluminação da lâmina no sentido contrário ao da lente. O invólucro do dispositivo possui um espaço para a instalação de uma fonte de luz a uma distância adequada para que a

<sup>4</sup> Ver <<https://www.electronicshub.org/>>. Acesso em 05 de Dez. de 2018.

<sup>5</sup> Ver <<https://pypi.org/project/RPi.GPIO/>>. Acesso em 17 de Jul. de 2019.

câmera em conjunto com a lente consiga dar foco nas células e capturar imagens com qualidade suficiente para o processamento.

A Figura 23 mostra a lâmpada LED branca utilizada como fonte de iluminação. Ela é alimentada independentemente por uma bateria de 9V.

Figura 23 – Lâmpada LED Branca utilizada como Fonte de Iluminação.



Fonte: O autor

### 3.1.6 Implementação do algoritmo para captura das imagens e controle dos módulos

O código foi escrito em Python e realiza a manipulação da câmera do Raspberry Pi através da biblioteca **picamera**<sup>6</sup>.

O algoritmo executa um laço *for* para realizar a captura de 20 imagens em sequencia. Há um intervalo de 5 segundos entre uma captura e outra. Este tempo é necessário para permitir que o sensor de imagens defina seus níveis de iluminação.

A cada foto capturada, o algoritmo faz uma chamada da função de controle dos motores para deslocar a câmera ou a lente, dependendo da posição em que cada uma se encontra. Isso faz com que sejam obtidas imagens de diferentes locais da amostra na lâmina. As imagens são salvas em uma pasta temporária e processadas uma a uma pela função de contagem que será descrita na seção 3.2. A Figura 24 mostra um trecho do código.

---

<sup>6</sup> Ver <<https://pypi.org/project/picamera/>>. Acesso em 17 de Jul. de 2019.

Figura 24 – Trecho do Algoritmo responsável por Ativar os Módulos do Protótipo.

```
#Define a quantidade de imagens a serem capturadas
quantidadeImagens = 20

#Configura a câmera
camera = PiCamera()
camera.resolution = (2592,1944)

#Inicializa a câmera
camera.start_preview()

#Faz a captura das 20 imagens
for i in range(quantidadeImagens):
    sleep(5)
    camera.capture('Imagens/imagem%s.jpg' % i)
    if (i == 0):
        controleMotor(-20, 33, 32, 35, 31, i1, positivo1, negativo1, y1)
    ...
```

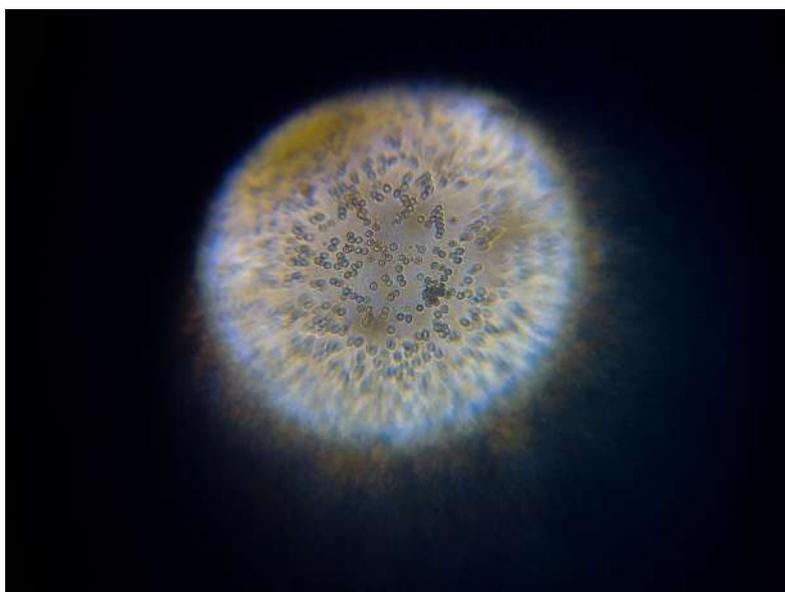
Fonte: O autor

## 3.2 Implementação do Algoritmo para Contagem das Células Sanguíneas

O algoritmo para processamento das imagens também foi desenvolvido na linguagem de programação Python. A versão 3.4.4 da biblioteca **OpenCV** para esta linguagem foi bastante utilizada no código, visto que ela possui uma coleção extensa de funções de manipulação de imagens que facilitam e agilizam o desenvolvimento.

As imagens capturadas pelo protótipo têm resolução de 2592x1944 *pixels* e possuem o padrão mostrado na Figura 25: Um fundo preto com um círculo de bordas desfocadas contendo as células.

Figura 25 – Imagem Capturada Automaticamente pelo Protótipo.



Fonte: O autor

Elas são capturadas dessa forma devido ao pequeno orifício da lente de aumento que recebe os raios de luz da fonte de iluminação mencionada na seção 3.1.2. Somente este orifício permite a passagem de luz para a câmera, logo, esta captura a região iluminada e o restante da imagem permanece escuro.

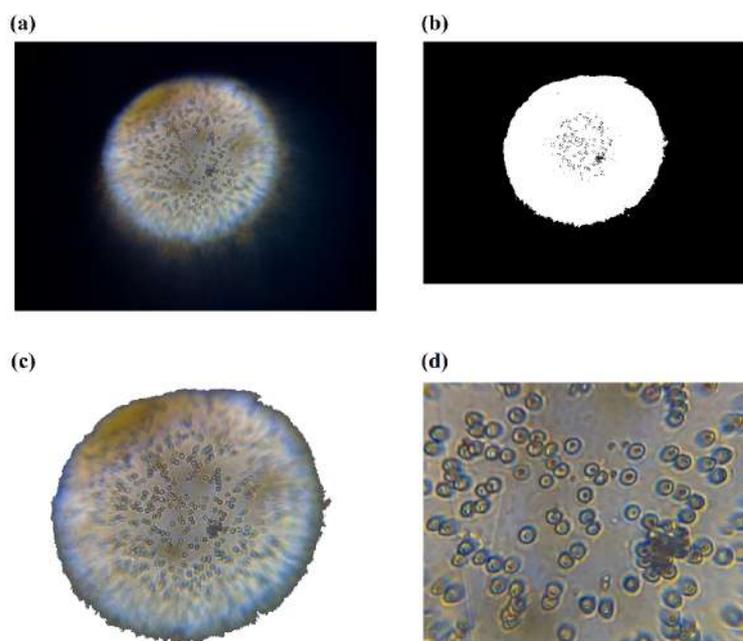
A Figura 25 é um exemplo de foto ideal para o processamento, pois exhibe as células bem focalizadas e razoavelmente distantes uma das outras, o que permite que o algoritmo realize a contagem com maior precisão.

### 3.2.1 Recorte da Área de Interesse

A primeira etapa do processamento das imagens com esse padrão é o recorte da área que apresenta as células. Para isso foi desenvolvida uma função que extrai a região de interesse através de uma máscara.

Para construção da máscara, a imagem de entrada (Figura 26 - (a)) é redimensionada para 90% do seu tamanho original, convertida para tons de cinza e passa por um processo de limiarização com o valor 80 de limiar, em seguida, na imagem resultante (Figura 26 - (b)), são detectadas todas as regiões formadas por *pixels* brancos e dentre essas regiões é escolhida a de maior área para ser a máscara. A área é medida pela quantidade de *pixels*.

Figura 26 – (a) Imagem de Entrada. (b) Imagem Binarizada. (c) Região de Interesse Extraída. (d) Região de Interesse Recortada.



Fonte: O autor

Feito isso, a região demarcada pela máscara é extraída da imagem original (Figura 26 - (c)) e passa por um processo de recorte das bordas onde são retirados 380 *pixels*

de cada lado da imagem retangular. Essa etapa é necessária para eliminar as bordas desfocadas que a máscara não consegue remover. A imagem resultante (Figura 26 - (d)) contém apenas as células em foco e é utilizada nas próximas etapas do algoritmo de contagem.

O processo de quantificação de células vermelhas através de imagens digitais se divide em duas partes: a primeira tem por objetivo separar as hemácias que aparecem grudadas na imagem por meio de operadores morfológicos, binarizações e do algoritmo *Watershed*; a segunda consiste em contar as células separadas na etapa anterior.

### 3.2.2 O Processo de Separação

O processo de separação visa afastar os objetos dentro da foto e destacá-los do fundo. No caso deste trabalho os objetos são as hemácias. Nessa etapa, o primeiro passo é fazer a imagem passar por alguns tratamentos para eliminar ruídos e realçar as células, com o objetivo de garantir maior precisão quando a separação for de fato iniciada.

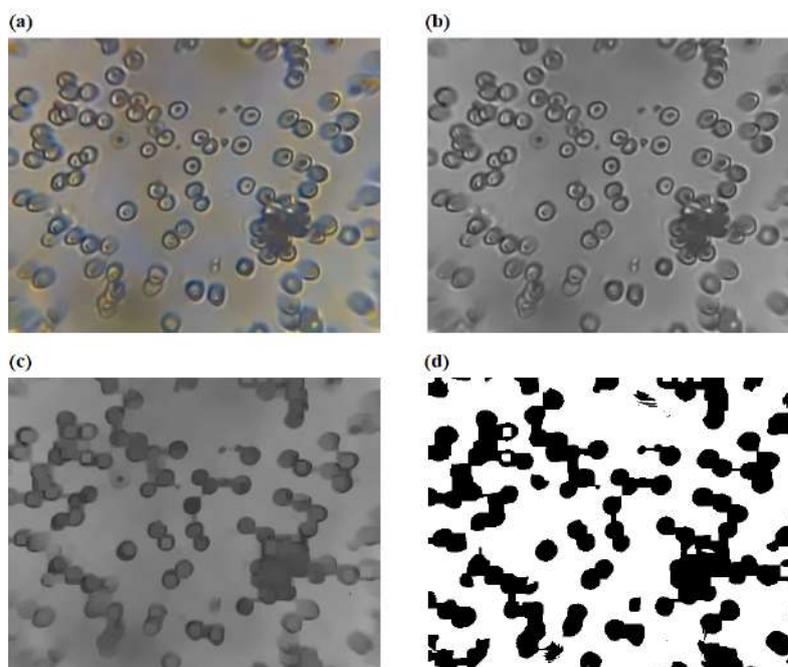
Para remoção de ruído foi utilizado o algoritmo *Non-Local Means Denoising*. O método é baseado em um princípio simples: substituir a cor de um *pixel* por uma média das cores de *pixels* semelhantes. E os *pixels* similares não necessariamente precisam estar na vizinhança do pixel que está sendo processado; uma vasta área da imagem é vasculhada em busca de todos os *pixels* semelhantes (BUADES; COLL; MOREL, 2011). A função *fastNlMeansDenoisingColored()* da biblioteca OpenCV implementa este algoritmo com várias otimizações computacionais. Depois de ter o ruído removido (Figura 27 - (a)), a imagem é convertida para tons de cinza (Figura 27 - (b)).

Os eritrócitos possuem uma região central mais clara que as bordas, chamada halo, que pode ser confundida com o fundo da foto durante o processamento, gerando um objeto com um furo no meio que dificulta a segmentação pelo *Watershed*. Para evitar esse problema, a figura é submetida a uma sequência de duas aberturas morfológicas, que resultam em um realce das células, destacando estas do fundo da imagem fechando seus halos. O resultado das operações pode ser conferido na Figura 27 - (c). Foi utilizada como elemento estruturante a matriz  $M1$ , mostrada a seguir, que equivale a uma imagem binária totalmente branca de resolução de 5x5 *pixels*. O próximo passo é submeter a figura a uma limiarização.

$$M1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Fatores como condições de iluminação ou concentração de fluídos na lâmina podem fazer com que imagens capturadas pelo protótipo apresentem manchas sutis, que, ao olho humano não representam nenhum problema, mas podem ocasionar perda de informação durante o processo de limiarização simples, pois alteram suavemente a tonalidade de alguns *pixels*. Para reduzir a ocorrência deste problema, foi utilizada uma binarização adaptativa em que, de acordo com a documentação oficial<sup>7</sup> do OpenCV, o valor de limiar é uma soma gaussiana ponderada dos valores de vizinhança menos uma constante  $C$ . Adotou-se  $C = 2$ . O resultado (Figura 27 - (d)) é uma imagem binária com as células marcadas em preto e o fundo branco. Com a imagem binarizada pronta, inicia-se a preparação para segmentação das hemácias contidas nela pelo algoritmo *Watershed*.

Figura 27 – (a) Imagem de Entrada após Passar pela Redução de Ruído. (b) Imagem Convertida para Tons de Cinza. (c) Imagem após Passar por Duas Aberturas Morfológicas. (d) Imagem após Passar pela Binarização Adaptativa.



Fonte: O autor

A biblioteca OpenCV implementa o *Watershed* baseado em marcadores. Antes de executá-lo é necessário rotular os objetos em primeiro plano na imagem, isto é, *pixels* que, sem dúvida alguma, compõem células, rotular também o fundo, ou seja, partes da figura nas quais há a certeza de que não existe nenhum *pixel* que compõe um objeto, e por fim rotular a região desconhecida, que é a área onde não se sabe quais *pixels* pertencem ao fundo ou objetos.

A partir da imagem binária é possível deduzir que: a região próxima ao centro das

<sup>7</sup> Ver <[https://docs.opencv.org/master/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html)>. Acesso em 25 de Jul. de 2019.

hemácias compõe o primeiro plano da imagem, as regiões distantes das células compõem o plano de fundo e a parte desconhecida é a região limítrofe das hemácias.

Para facilitar a marcação de cada região é executada uma inversão nos *pixels* da figura binária, fazendo com que todos os valores 1 se tornem 0 e vice-versa. A imagem resultante é submetida a um fechamento morfológico para eliminar possíveis furos nos objetos brancos em primeiro plano. O elemento estruturante utilizado é a matriz  $M2$  de dimensão 5x5 mostrada abaixo. O resultado dessas operações pode ser visto na Figura 29 - (a).

$$M2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A demarcação da área que corresponde ao fundo é feita através de uma sequência de três dilatações na imagem binária com a matriz  $M3$  de elemento estruturante. Essa operação aumenta o contorno dos objetos (Figura 29 - (b)); dessa forma é possível garantir que, no resultado final, tudo que está em segundo plano seja realmente parte do fundo, já que a região de borda é removida.

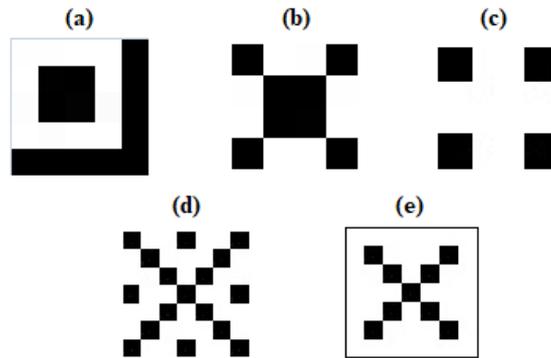
$$M3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Para extrair a região em primeiro plano foram feitas três erosões em sequência utilizando um elemento estruturante personalizado. A escolha deste elemento mostrou-se de vital importância para o processo de separação, já que este influencia diretamente a etapa de contagem. Durante o desenvolvimento do algoritmo foram feitos testes com diversos elementos, alguns estão listados na Figura 28. Após análises visuais dos resultados, decidiu-se utilizar o elemento mostrado na Figura 28 - (a) (5x5 *pixels*) como constante do processo. Ele é capaz de remover as bordas das células suavemente e isolar com maior precisão as regiões em primeiro plano, como mostra a Figura 29 - (c).

As erosões sozinhas não são suficientes para cindir algumas áreas em que as células se tocam, por isso o resultado do processo é submetido a uma transformada de distância. Esta operação em uma imagem binária especifica a distância de cada *pixel* até o ponto limite mais próximo (*pixel* diferente de zero) (FELZENSZWALB; HUTTENLOCHER, 2012).

Utilizou-se a função `distanceTransform()` do OpenCV, que calcula a distância

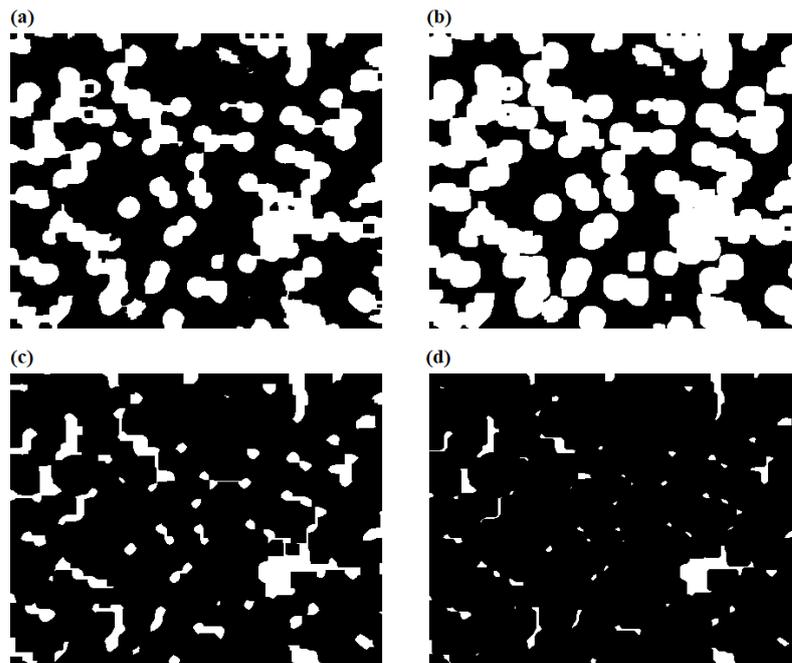
Figura 28 – Elementos Estruturantes.



Fonte: O autor

aproximada ou precisa de cada *pixel* da imagem binária até o *pixel* zero mais próximo. O resultado é uma figura em escala de cinza (com as mesmas dimensões da imagem de entrada) onde a intensidade do nível de cinza de cada *pixel* representa a sua respectiva distância até o correspondente ponto limite. Conseqüentemente, as regiões mais distantes das bordas dos objetos ficam mais claras e com a escolha do limiar adequado é possível segmentar estas partes com uma binarização.

Figura 29 – (a) Imagem após Passar por Inversão de *Pixels* e um Fechamento Morfológico. (b) Resultado das Dilatações para Detecção do Fundo. (c) Imagem após Passar por 3 Erosões. (d) Imagem após Passar pela Transformada de Distância e uma Limiarização Simples; Área em Primeiro Plano.

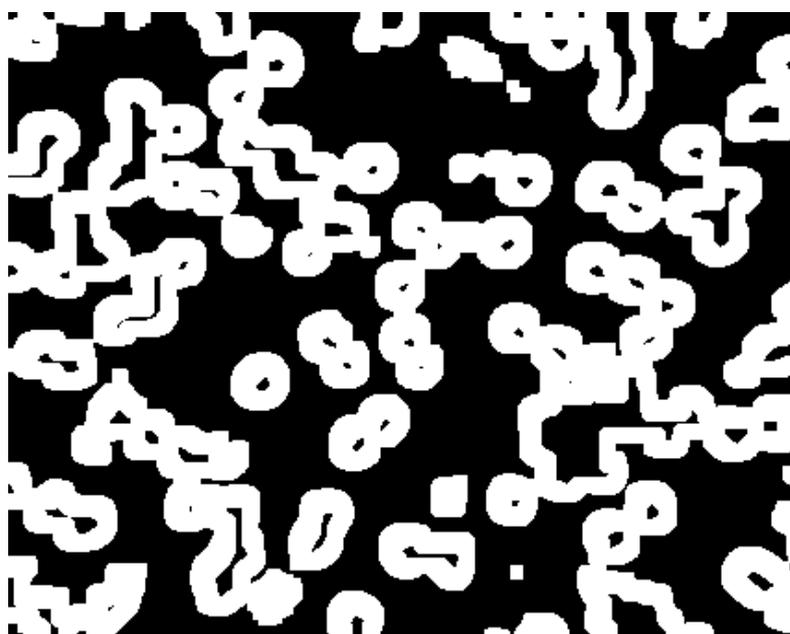


Fonte: O autor

Após alguns testes, verificou-se que usar 20% da distância máxima obtida pela transformada como limiar apresentava bons resultados e adotou-se este valor como padrão desta limiarização. A Figura 29 - (d) mostra um resultado.

Com o fundo e o primeiro plano definidos, tudo que resta na imagem compõe a região desconhecida que deverá ser inundada pelo *Watershed*. Estas áreas estão normalmente ao redor dos limites das células onde o primeiro plano e o fundo se encontram ou duas células diferentes se encontram. Chama-se região de fronteira. Ela pode ser obtida subtraindo área do primeiro plano da área do fundo, como mostra a Figura 30.

Figura 30 – Região Desconhecida.



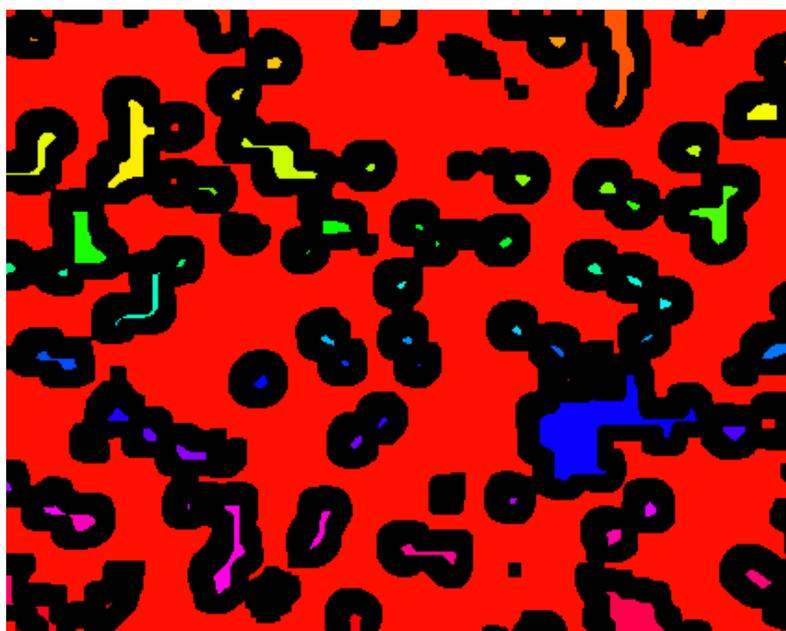
Fonte: O autor

Agora que cada região está devidamente definida, incia-se a fase de rotulagem dessas áreas. Isso é feito criando uma matriz com as mesmas dimensões da imagem binária, rotulando as regiões dentro dela. Para que a implementação do *Watershed* no OpenCV funcione de maneira correta, os *pixels* do fundo e de cada objeto na região em primeiro plano devem ser marcados com números inteiros positivos diferentes para cada área e os pontos da região desconhecida devem ser marcados como zero.

A rotulagem foi feita utilizando a função *connectedComponents()* do OpenCV. Ela implementa os algoritmos *Scan Array Union Find* (SAUF) (GRANA; BORGHESANI; CUCCHIARA, 2010) e *Block-Based Decision Table* (BBDT) (WU; OTOO; SUZUKI, 2005) para rotulagem de componentes conectadas. Por padrão, a função usa o algoritmo BBDT para conectividade-de-8 e o SAUF para conectividade-de-4 (sendo possível utilizar o SAUF também para conectividade-de-8, desde que isso seja especificado explicitamente). Para a rotulação utilizou-se os conceitos de conectividade-de-8 com o BBDT.

A função recebe como entrada a imagem com o primeiro plano e o fundo segmentados e retorna uma matriz de mesma dimensão da figura de entrada com os rótulos, onde os *pixels* do fundo são rotulados como zero e os pontos dos outros objetos encontrados são marcados com números inteiros a partir de 1. Mas, é sabido que, se o fundo estiver marcado com 0, a bacia hidrográfica considerará essa área como desconhecida. Então, é necessário marcá-lo com um número inteiro diferente; para isso soma-se 1 a todas as coordenadas da matriz com os rótulos e atribui-se zero às posições que realmente representam a região desconhecida usando a figura que corresponde a esta área, obtida anteriormente, como uma máscara para selecionar os pontos corretos. Para uma análise visual o resultado pode ser colorizado, como mostra a Figura 31. Nela, o fundo aparece marcado em vermelho, a região desconhecida aparece em preto e as áreas do primeiro plano aparecem com diferentes cores.

Figura 31 – Matriz de Marcadores Colorizada.



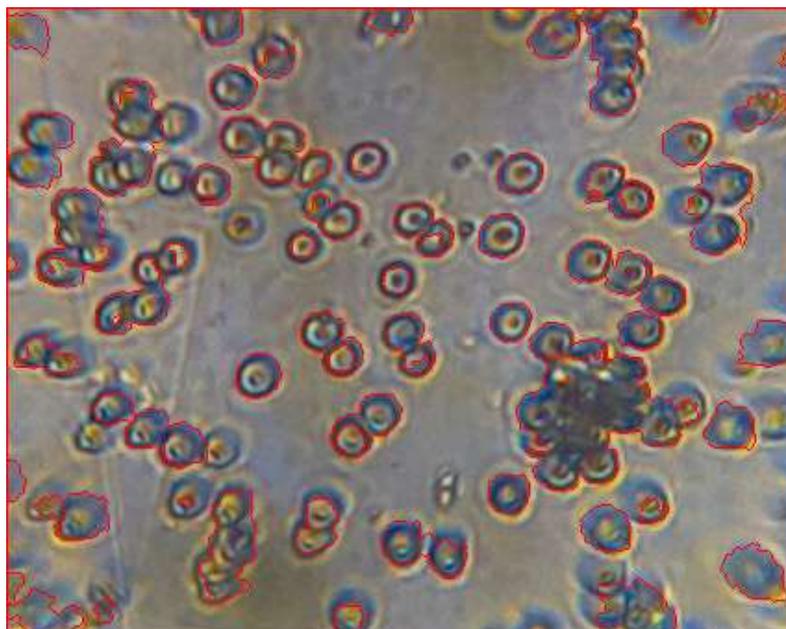
Fonte: O autor

Com a matriz de marcadores pronta, pode-se executar a função *watershed()* do OpenCV. Além desta matriz, a função recebe como entrada a imagem original, e retorna uma nova matriz de rótulos com os objetos devidamente segmentados e a região que separa uns dos outros rotulada com -1. Com essas informações foi possível marcar em vermelho na imagem original os *pixels* que correspondem a -1, completando o processo de separação das células. O resultado, tendo a Figura 26 - (d) de entrada, pode ser visto na Figura 32.

### 3.2.3 O Processo de Contagem

Essa etapa realiza a quantificação dos objetos na imagem. A entrada para a função de contagem é o resultado da etapa anterior: uma figura colorida com as barreiras de

Figura 32 – Resultado Final do Processo de Separação da Figura 26 - (d).

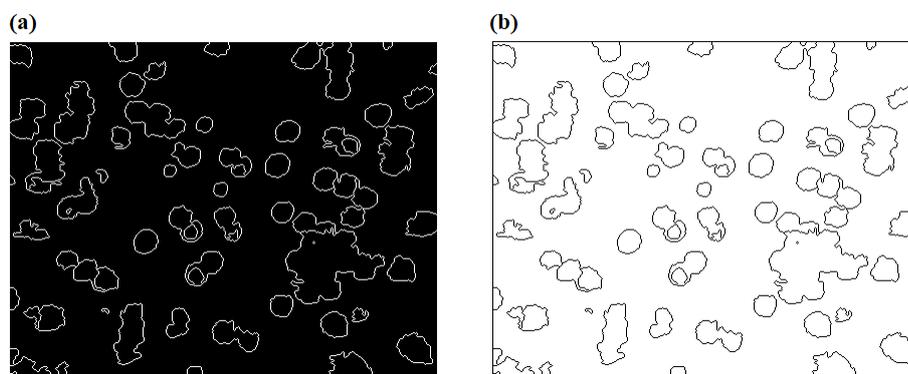


Fonte: O autor

segmentação marcadas em vermelho, como pode ser visto na Figura 32.

Para identificar cada objeto na figura é preciso rotulá-los, e essa operação exige que a imagem seja binarizada. A conversão da figura para binário mantendo as linhas que marcam a separação dos objetos é feita aplicando um filtro que transforma todos os *pixels* vermelhos em brancos e todos os outros em preto (Figura 33 - (a)). Por fim, a imagem tem seus *pixels* invertidos (Figura 33 - (b)).

Figura 33 – (a) Resultado da Filtragem. (b) Resultado da Inversão.



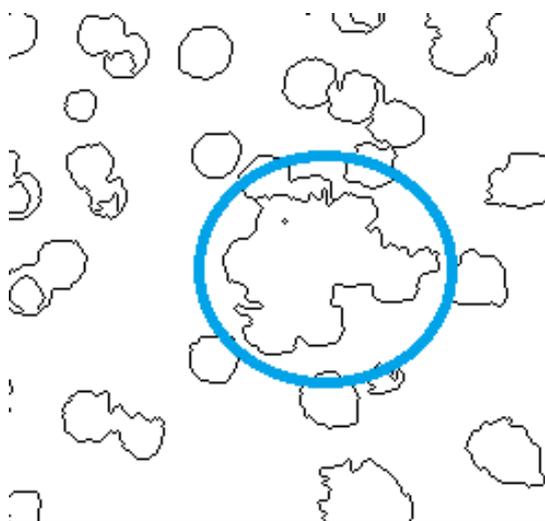
Fonte: O autor

Executou-se a rotulagem através da função `connectedComponentsWithStats()` da biblioteca OpenCV com o modo conectividade-de-4. Ela implementa os mesmos algoritmos de rotulagem de componentes conectados da função `connectedComponents()` descrita

anteriormente; a diferença é que a primeira, além de rotular os objetos, retorna informações sobre eles, inclusive uma lista com suas respectivas áreas medidas em *pixels*. Essa informação é importante para estimar o tamanho médio das hemácias. Após realizar a rotulagem e obter a relação que contém as áreas dos objetos, a quantização das células finalmente pode ser executada.

É realizada uma varredura na lista com as áreas para descartar os itens de tamanho maior que 3000 *pixels*. Isso é feito para evitar que o fundo ou possíveis manchas na imagem sejam contabilizados. O *array* resultante contém os valores que correspondem às áreas de cada objeto válido na figura, logo, seu tamanho representaria a quantidade de células. Mas esse método de contagem se mostrou inadequado, pois, em alguns casos, a etapa de segmentação não consegue separar todas as células, e apenas computar a extensão do *array* não leva em conta o tamanho dos objetos, assim, um objeto composto por 5 *pixels* tem o mesmo peso na contagem que outro de 300 *pixels*, por exemplo. O problema dessa abordagem pode ser observado no objeto circulado em azul na Figura 34. Ele é contado como uma única célula, mas na verdade corresponde a um aglomerado de aproximadamente 12 células que não puderam ser segmentadas na etapa anterior.

Figura 34 – Fragmento da Figura 33 (b) que destaca uma área onde o processo de separação falhou.



Fonte: O autor

Para contornar este problema e descobrir se um objeto equivale a uma única célula, várias ou um pequeno pedaço, utilizou-se um método de contagem por média proposto por Barros, Andrade e Trindade (BARROS; ANDRADE; TRINDADE, 2019). Este algoritmo calcula o tamanho médio dos objetos usando a quantidade de *pixels* que cada um tem. O cálculo médio terá o valor próximo ao do tamanho de uma célula na figura sempre que várias células únicas forem separadas. Os autores utilizaram essa técnica originalmente na contagem de esporos de fungos micorrízicos, que são morfologicamente semelhantes à

hemácias. O método é descrito a seguir:

A partir da média  $M$  e de cada conjunto  $Q$ , a quantidade de esporos  $P$  é calculada da seguinte forma:

1. Calcula-se a média  $M$  do tamanho dos objetos (esporos)

$M = \text{Quantidade de pixels pretos} / \text{quantidade de objetos}$ ;

2. Calcula-se os esporos  $P1$  inteiros descartando o resto da divisão.

$P1 = (Q) / M$  (divisão inteira sendo  $R$  o resto)

3. Calcula-se o resto  $R$  da divisão.

$R = (Q \text{ Módulo } M)$  (resto da divisão )

4. Calcula-se, se o resto corresponde a um esporo menor que a média.

$P2 = (R + (M * 7 / 100) / M)$  (soma aqueles que são superiores a 93% da média )

**Por último** soma-se os dois valores.

$P = P1 + P2$ ;

Com este recurso a contagem obteve melhores resultados, [...] (BARROS; ANDRADE; TRINDADE, 2019).

Utilizando o *array* com as áreas dos objetos válidos na imagem como entrada para este algoritmo foi possível melhorar a acurácia da contagem automática de células.

Parte IV

Resultados

## 4 Resultados

### 4.1 Análise do *Hardware* do Protótipo

Ao longo do desenvolvimento do protótipo, foram feitos diversos testes para avaliar seu desempenho. A primeira versão do equipamento utilizava como base um Raspberry Pi modelo B+. Este se mostrou inadequado para tarefas de manipulação de imagens, pois seu poder de processamento era baixo e conseqüentemente demorava demais para executar as funções de contagem e segmentação. O microcomputador foi substituído pelo Raspberry Pi 3 modelo B+ e este conseguiu realizar os cálculos com as matrizes sem maiores problemas.

Testes com os motores foram feitos através da função de controle onde eram enviados o número de passos em valores inteiros (tanto negativos como positivos) para verificar o deslocamento. Por conta da técnica de meio-passo que o algoritmo implementa, 400 passos são necessários para o eixo de cada motor completar uma volta em torno de si mesmo. Portanto, a cada 100 passos, os motores deslocam seu respectivo rotor em 90 graus.

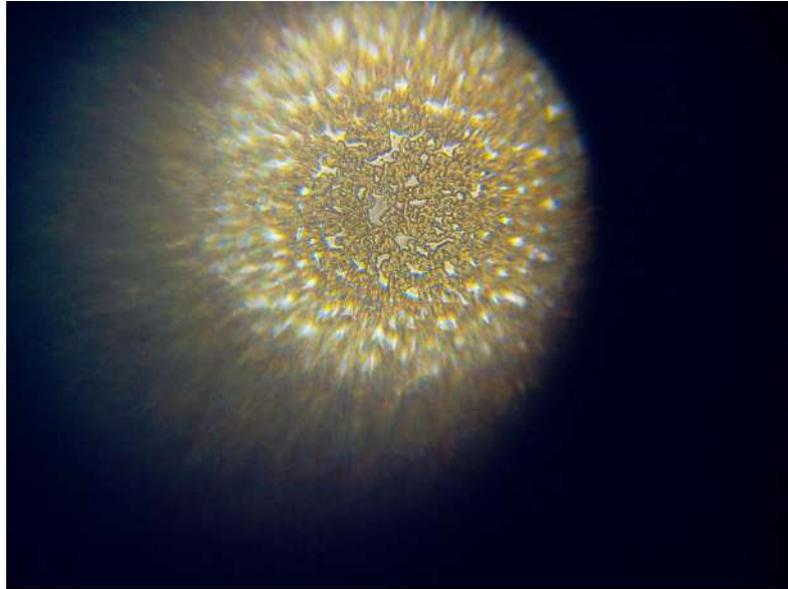
Para capturar as imagens das células, o módulo de movimentação da câmera foi configurado para deslocá-la de 20 em 20 passos no eixo X (a cada 18° de rotação do rotor), totalizando 400 passos, sendo o rotor girado 180° no sentido horário e 180° no anti-horário, e entre um deslocamento e outro uma imagem é capturada. O módulo que movimenta a lâmina a desloca 30 passos no eixo das ordenadas a cada 200 passos dados pelo módulo da câmera, totalizando 60 passos, sendo o eixo do motor rotacionado 27° no sentido horário e 27° no anti-horário. Esta configuração gera um movimento retangular da câmera sobre a amostra na lâmina. No final do processo, tanto a lâmina quanto a câmera retornam a sua posição inicial.

Verificou-se que, no geral, o protótipo desenvolvido cumpre sua tarefa de capturar e processar as imagens. O ciclo de captura de 20 fotos de uma amostra sanguínea e processamento destas dura cerca de 3 minutos. Contudo, o dispositivo ainda apresenta algumas limitações na fase de aquisição das imagens.

Encontrar uma lente que se adequasse às necessidades de portabilidade do projeto e fornecesse um aumento suficiente para visualização de células sanguíneas humanas não foi uma tarefa fácil. A Micro Phone Lens fornece um aumento de 150 vezes, mas percebeu-se que este nível de ampliação ainda não é adequado para capturar imagens de células em lâminas com esfregaço de sangue muito denso. Nessas fotos as células aparecem mescladas (Figura 35), isso impede a segmentação correta pois não é possível distinguir as regiões que compõem cada célula individualmente; o algoritmo não consegue processá-las. Este problema foi parcialmente contornado com a utilização de lâminas contendo sangue diluído

na proporção de 1:2.

Figura 35 – Imagem de Lâmina com Esfregaço Denso e sem Diluição.



Fonte: O autor

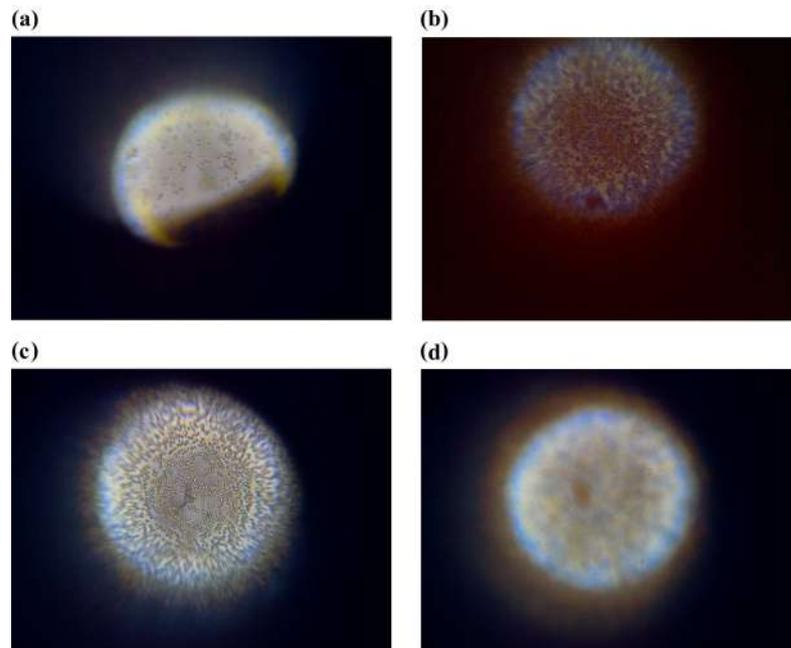
Outro inconveniente causado por esta lente é que ela não pode ser fixada de maneira permanente ao protótipo devido às suas características físicas. Ela adere estaticamente ao suporte desenvolvido para este fim, porém, durante a movimentação dos motores, pode se soltar, ou sair do seu posicionamento em frente à câmera, o que gera imagens desfocadas, com manchas (Figura 36 - (a)) ou totalmente escuras.

A iluminação é outro fator crucial para a captura de imagens com qualidade necessária para o processamento. A fonte luminosa do protótipo ilumina a lâmina e a lente através de aberturas na gaveta e no módulo de movimentação, estes devem estar alinhados de maneira que permitam a passagem de luz, caso contrário as imagens obtidas ficarão fora do padrão exigido pelo algoritmo, gerando inconsistências. Veja a Figura 36 - (b). A iluminação do ambiente também pode afetar negativamente a aquisição de imagens, provocando distorções e desfoques (Figura 36 - (c)).

A lâmina deve ser posicionada de maneira correta na gaveta para evitar que os motores emperrem; o lado que contém a amostra deve ficar virado para a fonte de luz, caso contrário, as fotos também serão capturadas com desfoque (Figura 36 - (d)).

Estas foram as principais dificuldades encontradas. É importante ressaltar que, em um ambiente controlado para testes, todos esses problemas são resolvidos, o protótipo adquire as imagens no padrão correto e realiza o processamento, como dito anteriormente.

Figura 36 – (a) Imagem com Mancha causada pelo Deslocamento da Lente. (b) Imagem com Iluminação Inadequada. (c) Imagem com Distorções. (d) Imagem Desfocada.



Fonte: O autor

## 4.2 Análise do Algoritmo de Processamento das Imagens

Os testes para validação do algoritmo foram feitos contando as hemácias manualmente nas imagens e comparando os resultados com as respostas obtidas da função de contagem.

Numa amostra de 20 imagens dentro do padrão correto, sem distorções, manchas, desfoques e iluminação inadequada, verificou-se que o percentual médio de acerto é de 81.58% com desvio padrão de 9.36%.

A variação dos percentuais de acerto é alta, como mostra o desvio padrão. Em algumas imagens do conjunto o acerto é menor que 75% enquanto em outras o algoritmo ultrapassa 95%. Isso se deve principalmente à diferença na concentração de hemácias em cada imagem. Observou-se que quanto mais dispersas pela foto, maior o acerto.

O algoritmo executado várias vezes com o mesmo conjunto de imagens da mesma lâmina sempre apresenta o mesmo resultado. Executando-se repetidas vezes o processo de captura e processamento das imagens em uma lâmina verificou-se que os resultados divergem se as imagens adquiridas em uma rodada não forem das mesmas regiões da lâmina capturadas em rodadas anteriores. Isso ocorre porque se o conjunto de imagens é adquirido de uma área diferente, a quantidade de células em cada figura será diferente, logo as contagens também serão. Contudo, como se trata da mesma lâmina, os resultados sempre são semelhantes quando não ocorrem problemas na aquisição.

A técnica para separação das células aplicada nesta pesquisa mostrou-se eficiente para a maioria dos casos, contudo, em regiões onde hemácias aparecem muito juntas ou sobrepostas o processo falha. A variação do tamanho das células na imagem também influencia negativamente a separação, visto que as erosões e a transformada de distância seguida de limiarização aplicadas, tendem a eliminar objetos muito pequenos, por isso, em alguns casos, células menores não são segmentadas corretamente.

O método de contagem por média se mostrou apropriado e mais eficiente do que simplesmente contar os objetos de tamanhos diferentes na figura, porém, é diretamente influenciado pela fase de segmentação, já que para que este chegue a uma aproximação correta do tamanho de uma hemácia, várias delas precisam ser separadas.

Quanto às imagens fora do padrão necessário, o algoritmo não consegue processá-las corretamente. Ao tentar contar hemácias em fotos como as mostradas na Figura 36, por exemplo, o código falha de diversas maneiras. Manchas e baixa iluminação ocasionam erros na fase de recorte da área de interesse e isso implica em inconsistências nas etapas seguintes do processamento. Em imagens com distorções e desfoques as células não são detectadas de forma satisfatória. Portanto, nesses casos, os resultados das contagens não são confiáveis.

## 5 Conclusão

A ideia de reproduzir os valores de um eritrograma a partir de imagens digitais e através de um dispositivo de baixo custo é bastante promissora, poderia facilitar a realização de exames tanto em áreas remotas quanto urbanas e o resultado seria obtido na hora.

A maior parte do *hardware* desenvolvido nesta pesquisa funciona como o planejado. A solução para processamento, o Raspberry Pi, se mostrou extremamente versátil e as possibilidades de expansão do projeto com ele são inúmeras.

Um dos obstáculos encontrados está relacionado à aquisição das imagens com o aumento necessário. Obter imagens microscópicas, em geral, exige equipamentos robustos. Conseguimos a ampliação de 150 vezes, suficiente para visualização das células, mas que não é adequada para todo tipo de situação. Um aumento superior a este poderia ter simplificado o desenvolvimento do algoritmo, pois as células seriam detectadas com maior facilidade e haveriam menos aglomerações e sobreposições.

Características físicas da lente utilizada para obtenção das imagens levaram à principal dificuldade encontrada no desenvolvimento do protótipo: fazê-lo capturar imagens sempre com as mesmas características. Toda tarefa de processamento de imagens exige um padrão de entrada.

Como pôde ser observado, a contagem de células sanguíneas a partir de imagens envolve várias técnicas. Os objetos podem ser detectados através de operações como limiarizações, transformações morfológicas e do algoritmo *Watershed*. Este último se mostrou uma ferramenta bastante robusta para a segmentação de hemácias. O código desenvolvido nessa pesquisa apresenta resultados satisfatórios, verificou-se que o percentual de acerto pode ultrapassar 95% em algumas figuras. Porém, está limitado pela fase de aquisição das imagens, já que se estas não estiverem no padrão correto, os resultados não serão confiáveis.

É importante ressaltar que a contagem de eritrócitos é apenas uma parte do eritrograma. Questões como a dosagem de hemoglobina e a determinação do hematócrito não foram abordadas nesta pesquisa. Além disso, é preciso estabelecer métricas para a realização de diagnósticos a partir da contagem obtida das 20 imagens.

Diante de todas essas observações, pode-se concluir que há necessidade de aprimoramento do protótipo físico e do seu algoritmo de processamento de imagens para que estes possam ser utilizados em ambientes reais.

## 5.1 Trabalhos Futuros

Como trabalho futuro, recomenda-se a pesquisa por uma lente que forneça um aumento maior e corrija os problemas causados pela utilizada atualmente.

Visando a melhoria dos resultados obtidos, o algoritmo de contagem das células pode ser aprimorado através de técnicas de inteligência artificial para detecção de objetos. Também deverão ser discutidas abordagens para definir a dosagem de hemoglobina e o hematócrito a partir de imagens. Além disso, a criação de uma interface gráfica para o sistema do protótipo é extremamente necessária para que este possa ser utilizado por usuários comuns.

Por fim, sugere-se a realização de estudos em parceria com pesquisadores da área da saúde com o objetivo de estabelecer metodologias para realizar diagnósticos a partir dos resultados obtidos pelo algoritmo.

## Referências

- ANDRADE, A. O. Sistema de contagem com morfologia matemática fuzzy. Universidade Federal do Rio Grande do Norte, 2014. Citado 4 vezes nas páginas 17, 19, 23 e 25.
- ANDRADE, A. O. et al. Uso da morfologia matemática fuzzy na contagem esporos de fungos micorrizicos. 2012. Citado na página 15.
- BARROS, A.; ANDRADE, A.; TRINDADE, R. Utilização da morfologia fuzzy para quantificação de esporos de fungos micorrizicos. *Revista de Ciência da Computação*, v. 1, n. 1, p. 1–7, 2019. ISSN 2596-2701. Disponível em: <<http://periodicos2.uesb.br/index.php/recic/article/view/4920>>. Citado 3 vezes nas páginas 20, 53 e 54.
- BERNSEN, J. Dynamic thresholding of gray-level images. In: *Proc. Eighth Int'l conf. Pattern Recognition, Paris, 1986*. [S.l.: s.n.], 1986. Citado na página 24.
- BEUCHER, S.; LANTUÉJOUL, C. Use of watersheds in contour detection. In: . [S.l.: s.n.], 1979. v. 132. Citado na página 24.
- BEUCHER, S. et al. The watershed transformation applied to image segmentation. *SCANNING MICROSCOPY-SUPPLEMENT-*, Scanning Microscopy International, p. 299–299, 1992. Citado 2 vezes nas páginas 24 e 25.
- BRADLEY, D.; ROTH, G. Adaptive thresholding using the integral image. *Journal of graphics tools*, Taylor & Francis, v. 12, n. 2, p. 13–21, 2007. Citado na página 24.
- BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer vision with the OpenCV library*. [S.l.]: "O'Reilly Media, Inc.", 2008. Citado na página 28.
- BRITES, F. G.; SANTOS, V. P. d. A. Motor de passo. *Pet Tele Universidade Federal Fluminense. Niterói/RJ*, 2008. Citado 3 vezes nas páginas 29, 30 e 38.
- BUADES, A.; COLL, B.; MOREL, J.-M. Non-local means denoising. *Image Processing On Line*, v. 1, p. 208–212, 2011. Citado na página 46.
- DIGABEL, H.; LANTUÉJOUL, C. Iterative algorithms. In: STUTTGART, WEST GERMANY: RIEDERER VERLAG. *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*. [S.l.], 1978. v. 19, n. 7, p. 8. Citado na página 24.
- FAILACE, R. *Hemograma: manual de interpretação*. [S.l.]: Artmed Editora, 2015. Citado na página 15.
- FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Distance transforms of sampled functions. *Theory of computing*, Theory of Computing Exchange, v. 8, n. 1, p. 415–428, 2012. Citado na página 48.
- GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Edgard Blucher, 2000. Citado 8 vezes nas páginas 19, 20, 21, 22, 23, 25, 26 e 27.

- GRANA, C.; BORGHESANI, D.; CUCCHIARA, R. Optimized block-based connected components labeling with decision trees. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, v. 19, p. 1596–609, 03 2010. Citado na página 50.
- LANTUÉJOUL, C. La squelettisation et son application aux mesures topologiques des mosaïques polycristallines. In: . [S.l.: s.n.], 1978. Citado na página 24.
- MACHADO, A. F. *Erosões e dilatações morfológicas binárias seqüenciais rápidas*. Tese (Doutorado) — Universidade de São Paulo, 2008. Citado 3 vezes nas páginas 26, 27 e 28.
- MARCONE, M. d. A.; LAKATOS, E. M. Fundamentos da metodologia científica. 5<sup>o</sup> edição. São Paulo: Editora Atlas SA, 2003. Citado na página 16.
- MATHERON, G. *Eléments pour une théorie des milieux poreux*. [S.l.], 1967. Citado na página 25.
- MEYER, F.; BEUCHER, S. Morphological segmentation. *Journal of visual communication and image representation*, Elsevier, v. 1, n. 1, p. 21–46, 1990. Citado na página 24.
- PARKER, J. Gray level thresholding in badly illuminated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 13, n. 8, p. 813–819, 1991. Citado na página 24.
- PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: McGraw Hill Brasil, 2002. Citado na página 17.
- ROERDINK, J.; MEIJSTER, A. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam Inf*, v. 41, 10 2003. Citado na página 24.
- SAUVOLA, J.; PIETIKÄINEN, M. Adaptive document image binarization. *Pattern recognition*, Elsevier, v. 33, n. 2, p. 225–236, 2000. Citado na página 24.
- SAVAKIS, A. E. Adaptive document image thresholding using foreground and background clustering. In: IEEE. *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*. [S.l.], 1998. p. 785–789. Citado na página 24.
- SCHUBERT, C.; LANGEVELD, M. C. V.; DONOSO, L. A. Innovations in 3d printing: a 3d overview from optics to organs. *British Journal of Ophthalmology*, BMJ Publishing Group Ltd, v. 98, n. 2, p. 159–161, 2014. Citado na página 28.
- SERRA, J. P. *Introduction à la morphologie mathématique*. [S.l.]: Centre de morphologie mathématique de Fontainebleau, 1969. Citado na página 25.
- SHEN, D.; IP, H. H. A hopfield neural network for adaptive image segmentation: An active surface paradigm. *Pattern Recognition Letters*, Elsevier, v. 18, n. 1, p. 37–48, 1997. Citado na página 24.
- SOMMERVILLE, I. Engenharia de software. ed. *Adison Wesley, 6<sup>a</sup>. Edição*, 2003. Citado na página 17.
- WELLNER, P. D. Adaptive thresholding for the digitaldesk. *Xerox, EPC1993-110*, Citeseer, p. 1–19, 1993. Citado na página 24.

---

WHITE, J. M.; ROHRER, G. D. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Journal of research and development*, IBM, v. 27, n. 4, p. 400–411, 1983. Citado na página 24.

WU, K.; OTOO, E.; SUZUKI, K. Two strategies to speed up connected component labeling algorithms. 01 2005. Citado na página 50.

YANG, Y.; YAN, H. An adaptive logical method for binarization of degraded document images. *Pattern recognition*, Elsevier, v. 33, n. 5, p. 787–807, 2000. Citado na página 24.