ALESSANDRO WILLIAN DE SOUZA DIAS

SISTEMA DE AQUISIÇÃO E TRATAMENTO DE IMAGEM PARA AUXÍLIO NA CONTAGEM DE CÉLULAS SANGUÍNEAS

FLORIANÓPOLIS, 2014

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – IFSC
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE PÓS-GRADUAÇÃO EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

ALESSANDRO WILLIAN DE SOUZA DIAS

SISTEMA DE AQUISIÇÃO E TRATAMENTO DE IMAGEM PARA AUXÍLIO NA CONTAGEM DE CÉLULAS SANGUÍNEAS

Monografia submetida ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos de obtenção do Título de Especialista em Desenvolvimento de Produtos Eletrônicos.

Professor Orientador:
Fernando S. Pacheco, Dr. Eng

FLORIANÓPOLIS, 2014

CDD 621.38132
D541s

Dias, Alessandro Willian de Souza
Sistema de aquisição e tratamento de imagem para auxílio na contagem de células sanguíneas [MP] / Alessandro Willian de Souza Dias; orientação de Fernando S. Pacheco. – Florianópolis, 2014.

1 v.: il.

Monografia de Pós-Graduação (Desenvolvimento de Produtos Eletrônicos) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.

1. Engenharia de protótipos. 2. Contadores. 3. Processamento de imagem. 4. Células sanguíneas. 5. Microscópio. I. Pacheco, Fernando S. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis
Catalogado por: Ana Paula F. Rodrigues Pacheco CRB 14/1117

SISTEMA DE AQUISIÇÃO E TRATAMENTO DE IMAGEM PARA AUXÍLIO NA CONTAGEM DE CÉLULAS SANGUÍNEAS

ALESSANDRO WILLIAN DE SOUZA DIAS

Este trabalho foi julgado adequado para obtenção do Título de Especialista em Desenvolvimento de Produtos Eletrônicos e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 12 de junho de 2014.

Banca Examinadora:

_____
Fernando S. Pacheco, Dr. Eng.
Orientador

_____
Carlos Gontarski Speranza, M. Eng.

_____
Leandro Schwarz, M. Eng.

À minha mãe, Neide.
Ao meu pai, João.
Ao meu irmão, Wesley.

AGRADECIMENTOS

Começando, quero agradecer em primeiro lugar a minha família. Minha mãe Neide, por ter sempre me apoiado em todas as minhas escolhas da vida. Meu pai João, infelizmente não estando mais presente nesse plano, mas me criou e me deu uma boa educação, juntamente com a minha mãe. Meu irmão Wesley, por ser a pessoa que acompanhou e acompanhará toda a minha vida.

Quero agradecer aos meus amigos, que sempre estiveram quando eu precisei, sempre me dando conselhos e balizando a minha vida. Agradecimento especial para o Uilter, que se mudou comigo para Florianópolis, onde nós cursarmos essa pós-graduação, que dividiu casa com o nosso amigo João, onde podemos viabilizar esse curso. Quero agradecer meu amigo Alex por ter contribuído na correção e no suporte desse trabalho.

Agradeço aos meus mestres pelo conhecimento transmitido ao longo do curso, em especial ao professor, Dr. Fernando S. Pacheco pela orientação, pela sugestão do trabalho e confiança depositada na minha capacidade.

“Imaginação é mais importante que o conhecimento”.

Albert Einstein

“Os talentos atingem metas que ninguém mais pode atingir. Os gênios atingem metas que ninguém jamais consegue ver”.

Arthur Schopenhauer

RESUMO

Este trabalho apresenta um protótipo de contador de células sanguíneas a partir de uma imagem, o sistema opera em uma placa Raspberry Pi e utiliza a biblioteca OpenCV. O contador de células destaca as células do fundo da imagem, utilizando a técnica de limiarização. O sistema cria marcadores para as células utilizando a técnica de linhas divisoras de águas. As vantagens em destaque do protótipo são portabilidade, baixo consumo de energia, baixo custo e uma boa velocidade em contagem de células sanguíneas. Os resultados experimentais mostram que o protótipo pode detectar de forma rápida e corretamente o número de células em imagens sem e com sobreposição de células.

Palavras-Chave: Raspberry Pi, Células, OpenCV, ImageJ.

ABSTRACT

This work presents a prototype blood cell counter from an image. The system works over a Raspberry Pi board. The implementation uses OpenCV, a library for computer vision. The cell counter highlights the cells from the background using a threshold technique. The system marks the cells using the Watershed technique. The advantages of this prototype are its portability, low power consumption, low cost and high speed in blood cell counting. The experimental results show that the prototype can detect quickly and accurately the number of cells for images with and without cell superposition.

Keywords: Raspberry Pi, Cells, OpenCV, ImageJ.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARM …… Advanced RISC Machine

BLOB ..... Binary Large Object

IPP ……...……..... Integrated Performance Primitives

MLL ……...……. Machine Learning Library

RAM …….Random Access Memory

ROM …….…… Read Only Memory

SDRAM ……..... Synchronous dynamics random access

SUMÁRIO

1 INTRODUÇÃO

O sangue é a porção líquida do meio interno que circula rapidamente dentro de um sistema fechado de vasos denominado sistema circulatório. É constituído por um fluido no qual existem células em suspensão, moléculas e íons dissolvidos em água, apresentando propriedades das soluções coloidais.

O sangue é constituído de duas frações combinadas, sendo 55% para o plasma e 45% para as células. A porção acelular ou plasma é constituído de 91,5% de água que serve de solvente das substâncias orgânicas e minerais e ainda de veículo para as células, moléculas e íons. Os restantes 8% são formados por proteínas, sais e outros constituintes orgânicos em dissolução (CARVALHO, 2008).

A porção celular apresenta três tipos de células em suspensão no plasma:

- Glóbulos vermelhos, hemácias ou eritrócitos.
- Glóbulos brancos ou leucócitos.
- Plaquetas ou trombócitos.

Na área da bioquímica, um dos processos fundamentais em vários exames é a contagem de células sanguíneas, tanto para o diagnóstico médico quanto para pesquisa biológica. O método tradicional de contagem de células sanguíneas é manual, geralmente utilizando um hemocitômetro, como é mostrado na Figura 1, o que implica em um trabalho intensivo e demorado. Além da demora, o processo manual está sujeito a falhas, impactando nos resultados da contagem.

A contagem de células é expressa em concentrações — células por unidade de volume de sangue. A unidade de volume é expressa em milímetros cúbicos (mm3) em decorrência das dimensões lineares da câmara do hemocitômetro (contador de células): 1 mm3 = 1,00003 µL.

Qualquer procedimento de contagem celular deve passar por três etapas: diluição do sangue, coleta da suspensão diluída

em um volume específico, e por fim, contagem das células contidas neste volume. (CARVALHO, 2008).

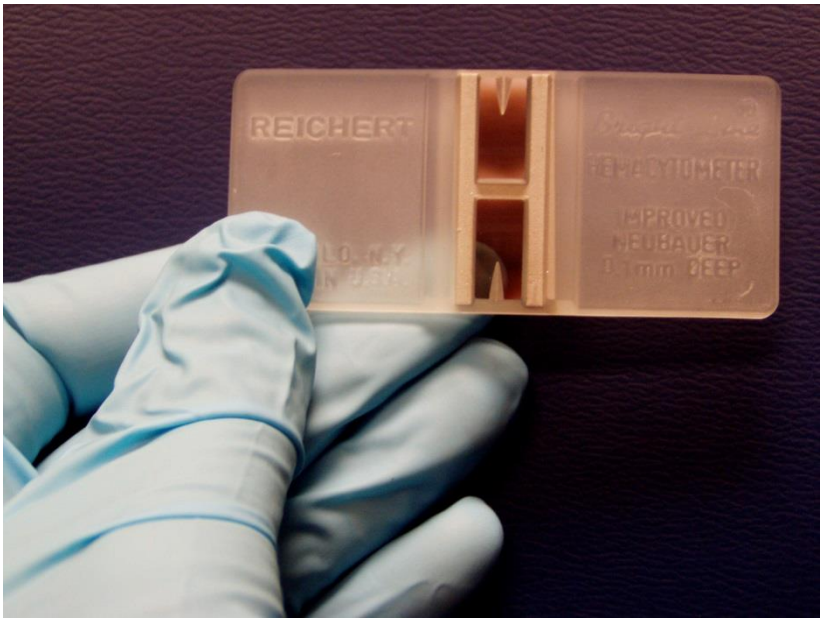


FIGURA 1 - Foto de um hemocitômetro

Fonte: HEMOCYTOMETER, 2014

O hemocitômetro é um dispositivo de cristal espesso com o tamanho de uma lâmina de vidro (30 mm x 70 mm x 4 mm), dividido em 3 partes, como mostra a Figura 2 (BASTIDAS, 2013).

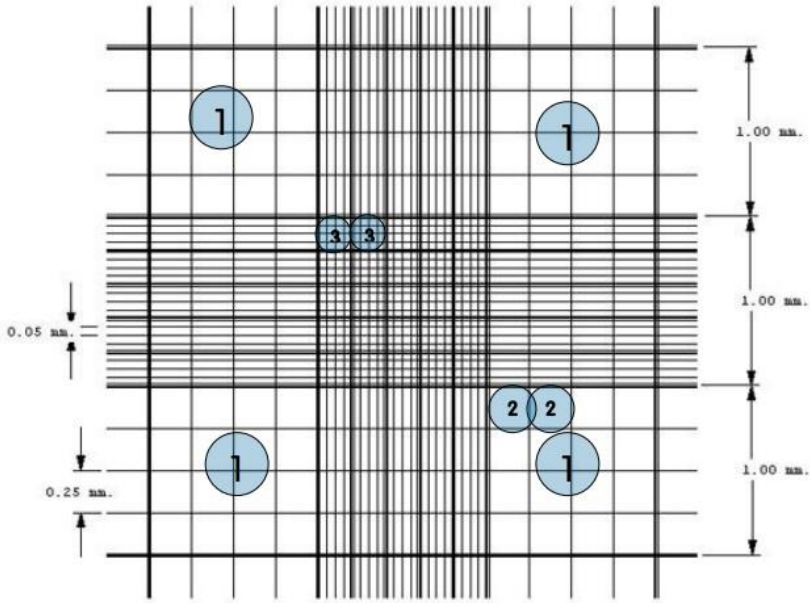


FIGURA 2 - Grade de contagem do hemocitômetro

Fonte: BASTIDAS, 2013

Na parte central da lâmina, existem várias linhas perpendiculares com marcações em quadrantes. Ao analisar em microscópico, com a objetiva de imersão, pode-se perceber que existem três tipos de quadrantes, que juntos formam um quadrado maior (CARVALHO, 2008). Esses quadrantes são utilizados para fazer as contagens da concentração de células em um determinado volume de fluido. A Figura 3 apresenta uma estimativa da contagem de concentração de células, utilizando o hemocitômetro (CARVALHO, 2008).

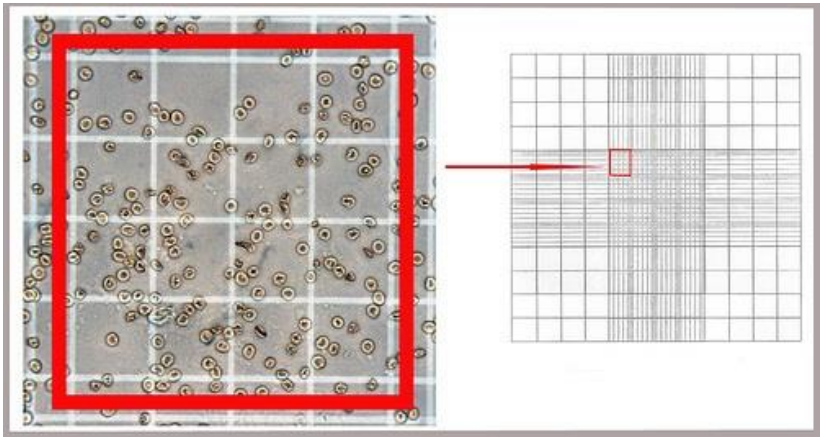


FIGURA 3 - Contagem de células utilizando um hemocitômetro

Fonte: LABMEDVET, 2014

É possível observar um exemplo de cálculo de concentração no ANEXO C. Como o cálculo de concentração é aproximado, a porcentagem de seu erro gira em torno de 20% a 30%, o que representa um índice alto, devido à importância da contagem de células sanguíneas (BASTIDAS, 2014).

Há equipamentos à venda no mercado nacional que automatizam a contagem, mas como são importados, têm alto custo.

Tendo em vista essa problemática, este trabalho propõe desenvolver um sistema para auxiliar na contagem de células sanguíneas. Uma visão dos blocos do sistema proposto é apresentada na Figura 4. O primeiro elemento do sistema é um microscópio com interface com um computador, de modo a capturar a imagem de interesse. Para esse equipamento, propõe-se que o processamento da imagem seja realizado por um computador de placa única (SBC, do inglês *single-board computer*). A escolha por um SBC deve-se ao interesse de ter-se um equipamento portátil, de fácil transporte. Ao computador, está ligada uma tela LCD com interface *touch*, de modo a facilitar a interação com o usuário.

Em relação ao *software*, planeja-se que realize a análise da imagem utilizando técnicas de processamento digital de imagens. Além disso, o *software* é responsável pela interação com o usuário. Planeja-se que o protótipo final seja composto por essas partes citadas, para melhorar a interação com o usuário, facilitada com a interface *touch*.

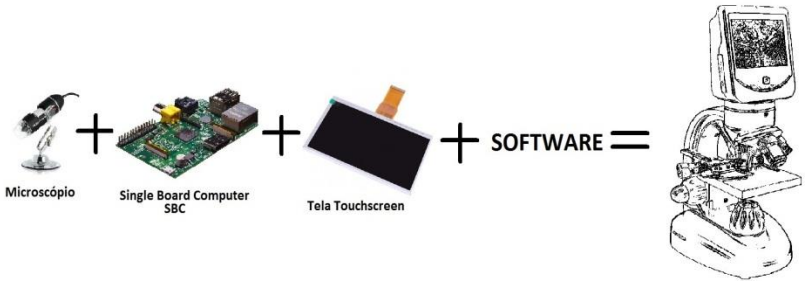


FIGURA 4 - Composição do equipamento proposto

Fonte: elaborada pelo autor

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um protótipo de um sistema que auxilie na contagem de células sanguíneas.

1.1.2 Objetivos Específicos

Como objetivos específicos deste trabalho têm-se:

- Estudar técnicas de processamento de imagens para contagem de células sanguíneas;
- Propor uma estrutura de hardware para o equipamento, considerando que o sistema deve ser portátil;
- Obter imagem de células e construir um software com seja capaz de fazer contagem;
- Desenvolver uma interface amigável para apresentar o resultado a ser verificado pelo usuário;
- Avaliar as técnicas desenvolvidas em relação à taxa de acerto.

1.2 Justificativa

Normalmente o processo de contagem de células sanguíneas é feito de forma manual ou utilizando equipamentos de alto custo. Além disso, algumas vezes os exames são realizados em lugares remotos, então uma solução portátil torna-se interessante. O protótipo proposto irá auxiliar o usuário a agilizar a contagem de células sanguíneas, diminuindo o tempo e o custo das análises bioquímicas.

1.3 Estrutura do trabalho

Estre trabalho está organizado como segue. No Capítulo 2, apresenta-se uma revisão bibliográfica de trabalhos relacionados à contagem de células, empregando tanto técnicas de processamento de imagens quanto outras abordagens. Também, relata-se o processo de prototipagem e avaliação de técnicas de contagem de células. A descrição do *hardware* empregado e das operações de processamento de imagem utilizadas é realizada no Capítulo 3, sobre a implementação. A avaliação do processo de contagem, com diferentes conjuntos de imagens, é descrita no Capítulo 4. O trabalho finaliza com a apresentação das conclusões e sugestões para trabalhos futuros nos Capítulos 5 e 6, respectivamente.

2 REVISÃO BIBLIOGRÁFICA

Foram pesquisados soluções e sistemas que se assemelham o que está sendo proposto no projeto. Na literatura, destacam-se a contagem por análise de imagem *blob* (GUO e YU, 2013) e através da técnica de linhas divisoras de águas *(Watershed)* (*JIN*, 2010). Nas próximas seções, são apresentados um resumo básico desses sistemas de contagem, e em seguida, os equipamentos com funções similares no mercado. Também são discutidas técnicas de processamento de imagens utilizadas no trabalho.

2.1 Método de automatizar a contagem de células com imagens microscópicas

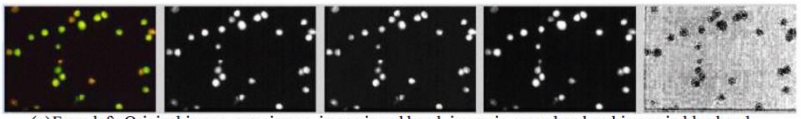
Guo e Yu (2013) apresenta um eficiente contador de células automático, baseado em imagem microscópica. O contador separa a célula e o fundo por histograma fazendo limiarização dupla, preenche a célula por uma técnica chamada *flood fill*, e assim detecta as células utilizando análise de *blob* (*Binary Large Object*). Em seguida, utiliza uma técnica chamada K-means, que segmenta as células aderentes. A vantagem desse método de contagem de células é a boa adaptabilidade para diferentes tipos de células e robustez à intensidade de luz.

Os resultados experimentais mostram que o contador pode detectar rapidamente o número de células, independente da morfologia celular (GUO e YU, 2013).

2.2 Segmentação automática de células com linhas divisoras de águas (Watershed)

Jin (2010) apresenta um método para contagem automática de células em um microscópio, usando algoritmos baseados em linhas divisoras de águas para a segmentação de células agrupadas. Uma característica do algoritmo reside no fato de que incorpora várias cores na análise da imagem. O método utiliza linhas divisórias para transformar o alinhamento de forma iterativa e é mais preciso em reter o formato da célula. Segundo os autores, apresenta uma sensibilidade de 97%, quando todas

as faixas de cor são utilizadas. Os métodos são interessantes para sistemas projetados para interpretar objetivamente imagens microscópicas, uma vez que eles proporcionam um meio para a segmentação das células (JIN, 2010). A Figura 5 mostra o processo de segmentação.



(a) From left: Original image, gray image, image in red band, image in green band and image in blue band

FIGURA 5 - Passos da imagem original até o *watershed*

Fonte: JIN, 2010

2.3 Equipamentos com funções semelhantes do projeto

Após fazer pesquisas sobre métodos e *softwares* para contagem de células, foram encontrados alguns equipamentos com função semelhante no mercado. Assim tem-se uma ideia do que os usuários esperam de um dispositivo para equipar um laboratório de bioquímica. Dois equipamentos são comentados aqui. O primeiro equipamento é o analisador de hematologia chamado BC-5800 (MINDRAY, 2014). Ele utiliza um método clássico chamado de impedância elétrica, no qual a célula passa através de uma abertura por vácuo. Alterações na resistência elétrica caracterizam uma contagem de célula.

Já o segundo é da empresa BioRad e se chama TC20 (BIORAD, 2014). Ele utiliza o método óptico para contagem automática de células, em amostras imersas em fluido, e segundo o fabricante garante a contagem de células da amostra em 30 segundos. Além disso, o fabricante informa que opera para uma vasta gama de tipos de células.

FIGURA 6 - Equipamentos com função semelhante ao protótipo. Da esquerda é um BC-5800, e o da direito um TC20

Fonte: elaborada pelo autor

2.4 Análise e avaliação de ferramentas

2.4.1 Software ImageJ

Para iniciar o desenvolvimento do projeto, foi preciso fazer uma análise e avaliar algumas técnicas de processamento de imagens. Para tanto, buscou-se um ambiente que permitisse a realização dessas tarefas de forma mais rápida e amigável. Escolheu-se assim o software ImageJ (IMAGEJ, 2014). É um software de domínio público e sua principal vantagem é a possibilidade de visualizar em tempo real o resultado de diferentes técnicas de processamento de imagens, através de uma interface gráfica.

ImageJ é um programa de processamento de imagem, desenvolvido na linguagem Java por Wayne Rasband, pesquisador do National Institute of Mental Health, nos Estados Unidos (FERREIRA e RASBAND, 2012).

ImageJ pode ser executado como um applet on-line ou um aplicativo para download, em qualquer computador com uma máquina virtual Java 5 ou mais nova. Distribuições para download estão disponíveis para diversos sistemas operacionais (IMAGEJ, 2014).

Ele possui várias ferramentas de ajuste de brilho e contraste, de segmentação e análise, medição de distâncias e ângulos, possibilidade de processar e analisar imagens de uma só vez, entre outros.

Outra característica importante desse software é que foi projetado com uma arquitetura que permite extensibilidade através de plugins e macros. Plugins desenvolvidos por usuários permitem resolver muitos problemas de análise e processamento de imagem.

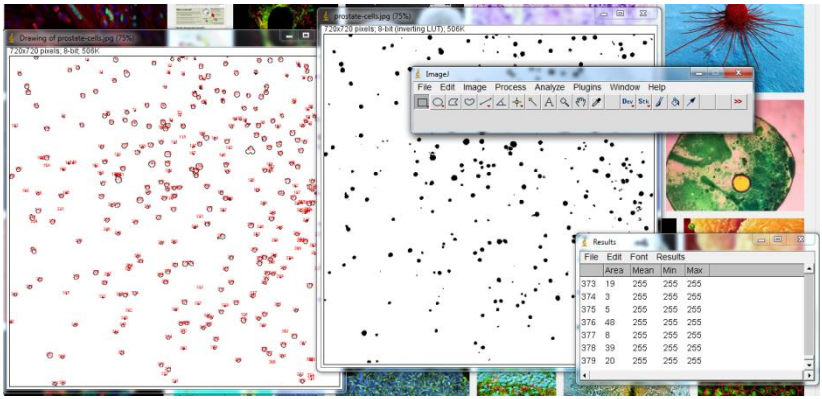


FIGURA 7 - Exemplo de teste de contagem realizado com o software ImageJ

Fonte: elaborada pelo autor

Durante a pesquisa, encontrou-se um tutorial de contagem de células usando o ImageJ (LABNO, 2014). Para familiarização com técnicas de processamento de imagem, decidiu-se reproduzi-lo, como mostra a Figura 7. Nesse tutorial, são empregados os seguintes comandos do ImageJ: (LABNO, 2014)

- Process –> Subtract Background
  - rolling=12

- Image –> Adjust –> Threshold
  - Histograma do Threshold(x, y)
  - Auto Threshold = "Default dark background"

- Process –> Binary –> Fill Holes

- Process –> Binary –> Convert to Mask
- Process –> Binary –> Watershed
- Analyze –> Analyze Particles
    - size=”120-Infinity”
    - circularity=”0.00-1.00”
    - show=”Outlines display exclude clear summarize”

Aplicados esses passos do tutorial, foi possível fazer a contagem de vários exemplos de imagens de células. Pela facilidade e velocidade no desenvolvimento, poderia ter sido escolhido o ImageJ, porque ele é portável para vários sistemas e já possui uma interface. O problema é que ele requer especificações superiores às do Raspberry Pi. Para manter a característica de portabilidade, era necessário um software que fosse mais aderente ao hardware escolhido, então foi escolhido o OpenCV, que funciona ano Raspberry Pi.

2.4.2 Visual Studio com o OPENCV

Antes de embarcar o algoritmo do projeto proposto no Linux, foi utilizado o Visual Studio 2010 para escrever e testar o algoritmo inicial. Um dos motivos para empregá-lo no Visual Studio 2010 nos testes iniciais, foi a facilidade de criação da interface para o usuário na linguagem C# (MAYO, 2010. Outro motivo, é a quantidade de material disponível sobre desenvolvimento para o OpenCV no Linux como o material do próprio OpenCV (OPENCV, 2014). A Figura 8 mostra o algoritmo sendo executado no Visual Studio 2010. No ANEXO A, mostra os passos para a instalação do OpenCV no Visual Studio 2010.

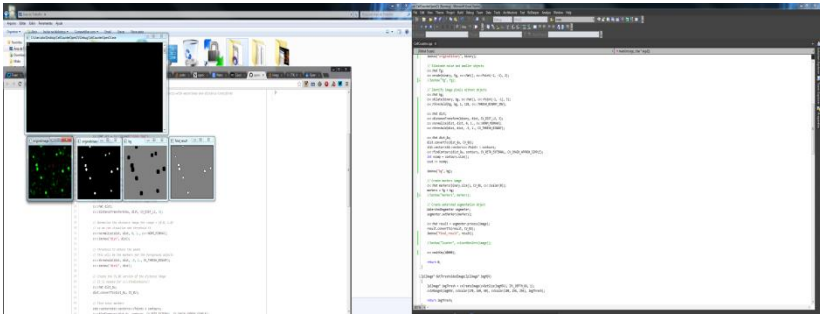


FIGURA 8 - Executando o algoritmo com o OpenCV pelo Visual Studio

Fonte: elaborada pelo autor

Com isso, a curva de aprendizado é muito mais rápida, além de facilitar testes de interoperabilidade entre sistemas. No caso deste projeto, foi possível testar os algoritmos tanto no sistema operacional Windows quanto em Linux, com arquiteturas diferentes, e pode-se comprovar a diferença nos cálculos de arredondamento da contagem de células sanguíneas.

2.5 Hardware embarcado utilizado – Raspberry Pi

Raspberry Pi é um projeto que tem como objetivo disponibilizar um computador simples e de baixíssimo custo para que jovens e crianças do mundo todo possam ter acesso às ferramentas básicas para o aprendizado de programação. (RICHARDSON, 2013)

Como a placa eletrônica desenvolvida com o objetivo educacional mostrou-se de muito baixo custo (US$ 35,00 com interface de rede), seu uso expandiu-se. Há placas Raspberry Pi em centrais multimídia, controle de máquinas e equipamentos diversos, por conta do custo e da facilidade de desenvolvimento com Linux. Desse modo, das diversas opções de hardware embarcado, com diferentes configurações de memória, interface de vídeo e arquitetura do processador, optou-se pelo Raspberry Pi, com arquitetura *ARM*. O Raspberry Pi contempla um

microprocessador modelo ARM1176JZF-S de 700 MHz embutido em um chip BCM2835 da Broadcom.

Além disso, tem interface Ethernet, entrada para cartão SD *(Secure Digital)* e 512 MB de memória de *SDRAM* (RASPBERRY PI, 2014)

2.6 Biblioteca OpenCV

OpenCV é uma biblioteca multiplataforma de visão computacional, muito popular em todo o mundo, com grandes comunidades de usuários na China, Japão, Rússia, Europa e Israel. É liberado sob a licença BSD e, portanto, é gratuito tanto para uso acadêmico e comercial. Possui interfaces para C++, C, Python e Java é suportado pelos sistemas operacionais Windows, Linux, Mac OS, iOS e Android. OpenCV foi projetado para ter eficiência computacional e com um forte foco em aplicações em tempo real. Escrito em C / C++ otimizado, a biblioteca também pode tirar proveito do processamento multi-core (BRADSKI e KOEHLER, 2008).

Essa biblioteca provê as funções de processamento de imagem necessárias para o projeto. A biblioteca OpenCV contém mais de 500 funções que abrangem aplicações em diversas áreas de processamento de imagens, incluindo inspeção visual em fabricação, diagnóstico médico, segurança, interfaces com usuário, calibração de câmeras, visão estereoscópica e robótica. O OpenCV é composto pela biblioteca MLL (*Machine Learning Library*), que é focada em reconhecimento de padrões estatísticos e *clustering*. No ANEXO B, mostra os passos para a instalação do OpenCV no Raspberry Pi.

2.6.1 Técnicas e algoritmos do OpenCV utilizados para contagem

Como já mencionado, o algoritmo do projeto consta de 4 passos. A imagem original deve ser transformada em imagem binária, depois reduz-se os ruídos, identificam-se os pixels na imagem e por fim, criam-se marcadores. A redução de ruídos e identificação dos pixels na imagem faz-se por meio da técnica de erosão, com a função de limiarização *(Threshold),* dilatação e a transformada da distância que irá segmentar e identificar as

células na imagem. Para criar marcadores e mostrar o resultado para o usuário, utilizam-se as linhas divisoras de águas *(Watershed)*. Nas seções seguintes, serão apresentadas essas funções e métodos do OpenCV.

2.6.2 Limiarização

A limiarização é um dos métodos mais simples de segmentação de imagens. Como exemplo de aplicação, tem-se a separação de regiões externas a objetos que se queria analisar em uma dada. Esta separação baseia-se na variação de intensidade entre os *pixels* (menor ponto que forma uma imagem digital) dos objetos e os *pixels* de fundo.

Para diferenciar os *pixels* que deseja-se analisar em relação ao resto da imagem, pode ser realizada uma comparação de cada valor de intensidade do pixel em relação a um valor limiar.

Uma vez que se diferenciam corretamente os pixels importantes, pode-se defini-los com um valor determinado para identificá-los (ou seja, pode-se atribuir um valor de 0 (preto) a 255 (branco) ou qualquer valor que se adapte às suas necessidades) (OPENCV, 2014).

2.6.2.1 Tipos de Limiarização

O OpenCV oferece a função de limiarização por 5 tipos de operações: limiarização binária, binária invertida, truncamento, limiarização para zero e limiarização para zero invertido.

Para ilustrar como os processos de limiarização operam, considera-se uma imagem original dos pixels com valores de intensidade *src*(x, y). A Figura 9 apresenta em vermelho a intensidade (eixo vertical) de uma linha essa imagem, tendo-se no eixo horizontal o número de cada pixel dessa linha. Em azul, tem-se um limiar qualquer, que servirá para a apresentação dos tipos de limiarização.

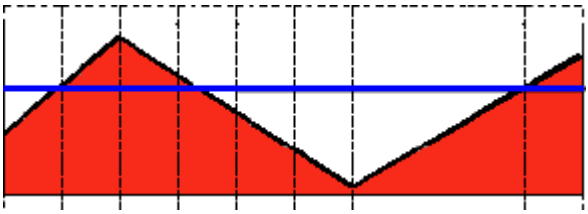


FIGURA 9 - Processo de limiarização

Fonte: OpenCV, 2014

2.6.2.2 Limiarização Binária

Esta operação de limiar pode ser expressa como:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxVal} & \text{Se } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{Senão} \end{cases}, \quad (1)$$

onde *dst*(x, y) representa a imagem intensidade resultante do processo de limiarização, *thresh* o limiar e *maxVal* é a nova intensidade aplicada aos pixels da imagem original que tem intensidade que supera o limiar *thresh*.

Assim, se a intensidade do pixel em *src*(x, y) é maior do que o limiar *(thresh)*, então a nova intensidade de pixel é definido como *MaxVal*. Caso contrário, a intensidade dos pixels é definida como 0. Aplicando a limiarização binária, descrita pela Equação 1, na Figura 8, tem-se como resultado a Figura 10.

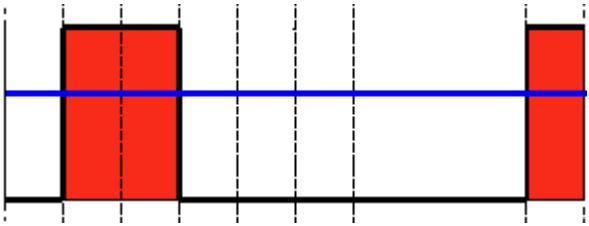


FIGURA 10 - Processo de limiarização binária

Fonte: OpenCV, 2014

2.6.2.3 Limiarização binária invertida

Esta operação de limiar pode ser expressa como:

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{Se } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{maxVal} & \text{Senão} \end{cases} \quad (2)$$

Se a intensidade do pixel *src*(x, y) é maior do que o limiar, então a nova intensidade de pixel é definido como um 0. Caso contrário, ela é definida como *MaxVal*. Aplicando a limiarização binária invertida, descrita pela Equação 2, na Figura 8, tem-se como resultado a Figura 10.

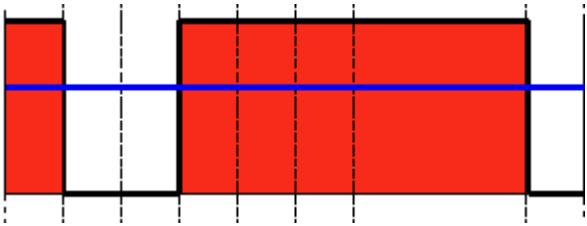


FIGURA 11 - Processo de limiarização binária invertida

Fonte: OpenCV, 2014

2.6.2.4 Truncamento

Esta operação de limiar pode ser expressa como:

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{threshold} & \text{Se } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{Senão} \end{cases} \quad (3)$$

Se o valor da intensidade máxima para os pixels é maior, e se *src*(x, y) é maior, então o seu valor é truncado. Aplicando o truncamento, descrito pela Equação 3, na Figura 8, tem como resultado a Figura 11.

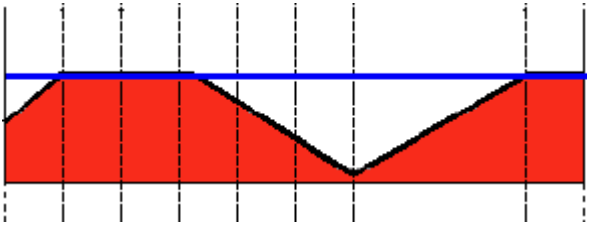


FIGURA 12 - Processo de truncamento

Fonte: OpenCV, 2014

2.6.2.5 Limiarização para zero

Esta operação pode ser expressa como:

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{src}(x,y) & \text{Se } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{Senão} \end{cases} \quad (4)$$

Se *src*(x, y) é inferior ao limiar, o novo valor de pixel será definido como 0. Aplicando a limiarização para zero, descrita pela Equação 4, na Figura 8 tem-se o resultado apresentado na Figura 12.

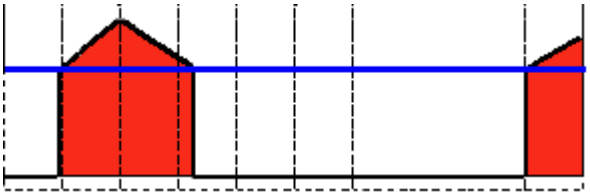


FIGURA 13 - Processo de limiarização para zero

Fonte: OpenCV, 2014

2.6.2.6 Limiarização para zero invertido

Esta operação pode ser expressa como:

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{Se } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{Senão} \end{cases} \quad (5)$$

Se *src*(x, y) é maior do que o limiar, o novo valor de pixel será definido como 0. Aplicando a limiarização para zero invertido, descrita pela Equação 5, na Figura 8, tem-se como resultado a Figura 13.

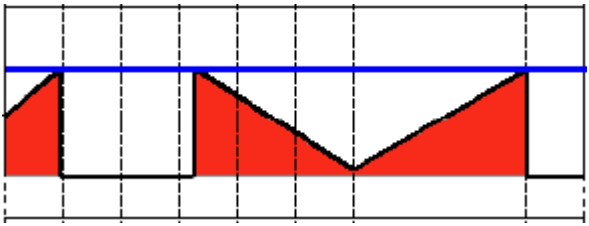


FIGURA 14 - Processo de limiarização para zero invertido

Fonte: OpenCV, 2014

2.6.2.7 Métodos do OpenCV para Limiarização

A partir de uma imagem, que pode ser lida com a função *imread*, deve convertê-la em tons de cinza de 8 bits, como pode ser visto na Figura 14. Para isso, pode ser utilizado o método cvtColor:

```
src = imread( argv[1], 1 );


/// Converte a image para cinza
cvtColor( src, src_gray, CV_RGB2GRAY );
```

Depois, pode-se criar uma janela para exibir o resultado.

```
namedWindow(window_name, CV_WINDOW_AUTOSIZE );
```

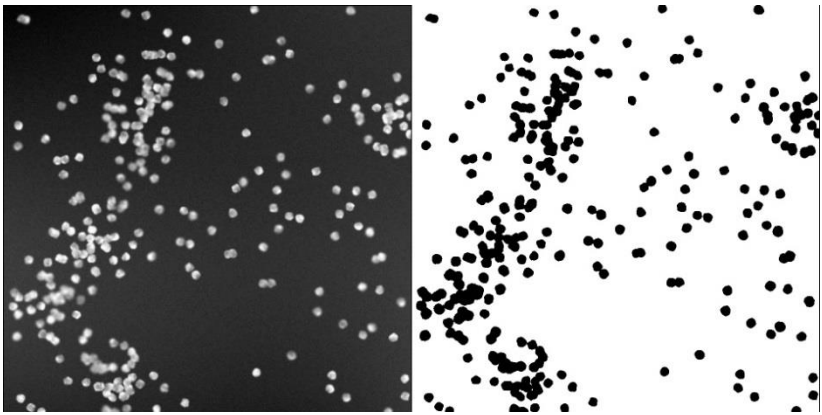


FIGURA 15 - Transformando uma imagem de células para imagem de 8 bits com cvtColor

Fonte: elaborada pelo autor

No OpenCV, pode-se também criar dois campos de texto para que o usuário digite a entrada:

- Tipo de threshold: Binary, a zero, etc.
- Valor de threshold

```
createTrackbar(trackbar_type,           window_name,
&threshold_type, max_type,

Threshold_Demo);


createTrackbar(trackbar_value,
window_name, &threshold_value, max_value,

Threshold_Demo);
```

Aguarda-se, então, até que o usuário digite o valor limite e o tipo de limiarização (ou até que o programa saia)

Sempre que o usuário alterar o valor de qualquer dos campos de texto, deve ser chamado o método ThresholdDemonstration:

```
void ThresholdDemonstration(int, void*)
{
  /* 0: Binario
     1: Binario Invertido
     2: Threshold truncado
     3: Threshold para zero
     4: Threshold para zero invertido
   */

  threshold(src_gray,dst,threshold_value,
max_BINARY_value,threshold_type);

  imshow(window_name, dst);
}
```

Como se observa, o método de limiarização é invocado e passam-se cinco parâmetros:

- *src_gray*: imagem de entrada;
- *dst*: Destino (output) imagem;
- *threshold_value*: O valor de limiar em relação ao qual a operação de limiar deverá ser feita;
- *max_BINARY_value*: O valor utilizado com as operações de limiarização binária (para definir os pixels escolhidos);
- *threshold_type*: Uma das cinco operações de limiarização. Eles estão listados na seção de comentários da função acima.

2.6.2.8 Exemplos de aplicação da função de Limiarização

Utilizando a Figura 16 como exemplo de imagem de entrada, para se alcançar os resultados das Figuras 17 e 18, tem-se aplicação das seguintes limiarizações:

FIGURA 16 - Imagem original

Fonte: OpenCV, 2014

1- Limiarização binária invertida. Os pixels de maior intensidade (de maior valor binário) ficarão escuros, como mostrado na Figura 17.

FIGURA 17 - Imagem resultante que passou pelo processo de limiarização binária invertida

Fonte: OpenCV, 2014

2- Limiarização para zero. Com isso, espera-se que os pixels mais escuros (abaixo do limiar definido) irão se tornar completamente pretos, enquanto os pixels com valor acima do limiar definido, irão manter seu valor original. Isso é verificado na Figura 18.

FIGURA 18 - Imagem resultante após processo de limiarização para zero

Fonte: OpenCV, 2014

2.6.3 Erosão

Uma das operações empregadas no algoritmo utilizado é a erosão. Nesta operação, o elemento estruturante é sobreposto à imagem em todas as posições possíveis por um *loop* (OPENCV, 2014). O pixel que fica na posição da origem do elemento estruturante é modificado de acordo com uma regra simples

Se o elemento estiver em parte sobre o objeto e em parte sobre o fundo, o pixel que está na posição da origem passa a ser fundo. Ou seja, um pixel só pode permanecer no objeto (com valor 1) se a origem do elemento estruturante estiver sobre ele e todos os elementos estiverem sobre o objeto.

O resultado é o que se pode ver na Figura 18. A imagem original está em cinza, e o resultado foi sobreposto a ela em branco. A operação de erosão eliminou uma borda de um pixel ao longo de todo o perímetro da imagem original.

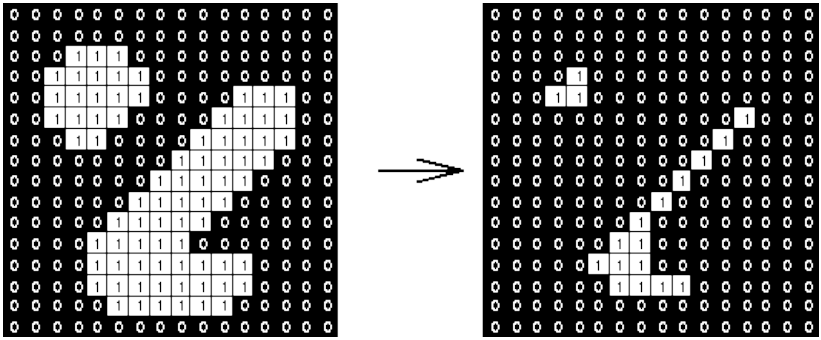


FIGURA 19 - Imagem que demonstra a erosão de uma imagem binária

Fonte: elaborada pelo autor

2.6.4 Dilatação

Na dilatação, a diferença em relação à erosão é que quando o elemento está em parte sobre o fundo e em parte sobre o objeto, o seu centro passa a ser objeto. Isto aumenta o tamanho do objeto, como o nome dilatação sugere.

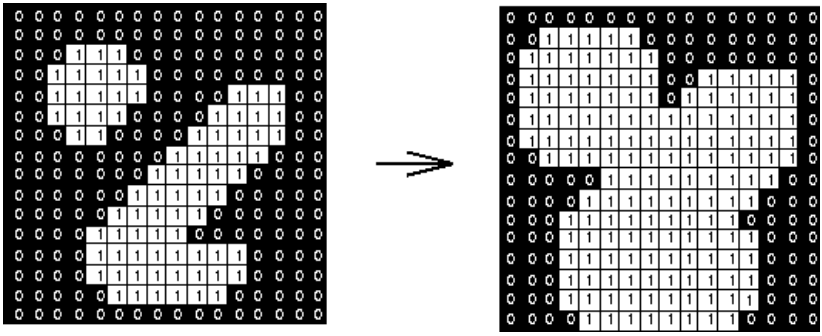


FIGURA 20 - Imagem que demonstra o processo de dilatação

Fonte: elaborada pelo autor

2.6.5 Transformada de distância

Para entender a transformada de distância, pode-se utilizar como exemplo uma imagem binária de um círculo mostrada na Figura 20.

FIGURA 21 - Imagem binária de um círculo de uma imagem

Fonte: elaborado pelo autor

A transformada de distância *T*, aplicada a um objeto gráfico *O*, calcula um campo escalar (ou vetorial) que representa distâncias mínimas entre o objeto e os pontos do espaço no qual ele está envolvido (BORGEFORS, 1986). A transformada T pode ser definida pela Equação 6:

$$T(O) = \min_{p_i \in O} dist(p, p_i), \quad (6)$$

Onde *p* representa pontos arbitrários do espaço, e *dist* representa uma função distância ou métrica utilizada. Assim, para cada ponto *p* do espaço, a transformada calcula a distância de *p* ao ponto p_i (p_i pertence a borda de *O*) que está mais próximo de *p*. Com esta definição, para os pontos *p* situados na borda do objeto, tem-se T(O)=0. A Figura 21 mostra a distância mínima entre um ponto *p* (interno ao objeto O) e a borda do objeto e a distância mínima entre um ponto *q* (externo) e o objeto (BORGEFORS, 1986).

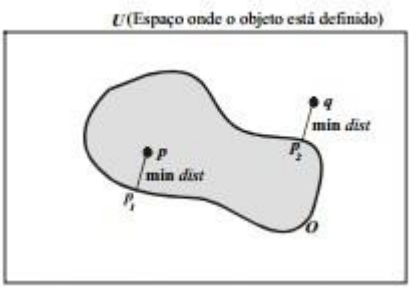


FIGURA 22 - Distância mínima

Fonte: BORGEFORS, 1986

O resultado da transformada *T* aplicada a um objeto *O* depende da métrica ou função distância *dist*, como será visto mais adiante.

2.6.5.1 Métricas

O resultado da transformada de distância aplicada a um objeto *O* depende de qual métrica ou função distância será utilizada. Dentre as várias métricas destacam-se: a *chessboard* e a *city block.*

2.6.5.1.1 Métrica *City Block*

Nesta métrica, também conhecida como métrica de Manhattan, a função *dist* é definida como mostra a Equação 7.

$$dist(p,q) = |p_1 - q_1| + |p_2 - q_2| + ... + |p_n - q_n|. \quad (7)$$

Assim a norma de um vetor é dada pela soma de suas componentes em cada eixo principal. A grande utilidade desta métrica surge quando aplicada a problemas em espaços discretos. Neste caso há uma interpretação interessante sobre a métrica *city block*: quando aplicada a objetos do espaço discreto, esta métrica assume que, para ir de um ponto *p* a um ponto *q*, só é possível andar nas direções dos eixos principais do sistema de

coordenadas onde o objeto está definido (não é permitido andar nas direções diagonais). Esta observação fica clara quando o objeto considerado é uma imagem 2D ou um dado volumétrico. No caso de imagens esta métrica define a topologia 4-conectado e no caso de volumes, define a topologia 6-conectado (BORGEFORS, 1986).

No caso de imagens, onde um pixel *p* possui 8 pixels adjacentes (4 que compartilham uma aresta e 4 que compartilham um vértice com *p*) esta métrica determina que, saindo do pixel *p* só é possível andar nas direções dos pixels que compartilham uma aresta (Figura 22). Assim, se um pixel *q* compartilha uma aresta com *p*, então dist(*p,q*)=1 e *q* é um pixel vizinho a *p*. Se *q* compartilha um vértice com *p*, então dist(*p,q*)=2 (*q* não é vizinho de *p*). Por esta razão a métrica *city block* também é referenciada como métrica “1-2”, no sentido de que, dado um ponto *p*, seus pixels adjacentes por aresta têm distância 1 (pixels vizinhos) e seus pixels adjacentes por vértices têm distância 2. Como cada pixel possui 4 pixels vizinhos (por aresta), esta métrica define a topologia 4-conectado. Para calcular a distância entre dois pixels quaisquer basta ir andando nas direções permitidas (horizontal e vertical) e contar a quantidade de pixels percorridos entre a origem e o destino. Na Figura 22a a distância entre *p* e *q* é 5. A Figura 22b mostra a transformada de distância usando a métrica *city block*, aplicada à imagem da Figura 20.

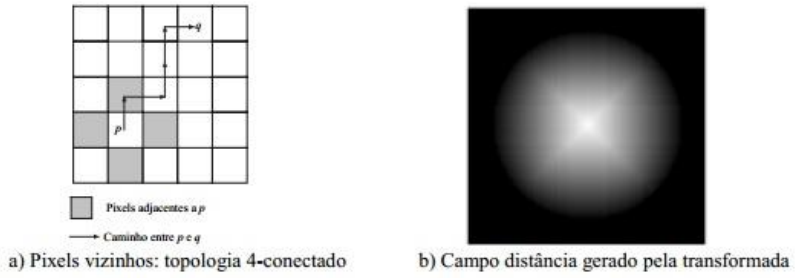


FIGURA 23 - Transformada de distância usando a métrica *city block*

Fonte: BORGEFORS, 1986

2.6.5.1.2 Métrica Chessboard

Na métrica *chessboard* a função *dist* é definida como na Equação 8.

$$dist(p,q) = \max(|p_1 - q_1|, |p_2 - q_2|, \ldots, |p_n - q_n|). \quad (8)$$

Assim a norma de um vetor é definida como sendo a sua maior componente. Assim como a métrica *city block*, a maior motivação para o uso da métrica *chessboard* está voltada às aplicações discretas. A interpretação desta métrica, quando aplicada a objetos do espaço discreto, é que, para ir de um ponto *p* a um ponto *q*, é permitido se deslocar em todas as direções. Conforme descrito abaixo, no caso de imagens esta métrica define a topologia 8-conectado e no caso de volumes, define a topologia 26-conectado (BORGEFORS, 1986).

A principal motivação para o nome dessa métrica vem das aplicações discretas 2D, onde, saindo de um pixel *p* é possível fazer os movimentos que um rei faz em um tabuleiro de xadrez. Portanto é permitido andar tanto nas direções dos eixos principais (horizontal e vertical) quanto nas direções diagonais da Figura 23. Devido a isto, um pixel *p* possui 8 pixels vizinhos: os 4 vizinhos por aresta (direções horizontal e vertical) e os 4 vizinhos por vértices (direções diagonal). Diz, então que esta métrica define a topologia 8- conectado. No caso 2D a métrica *chessboard* é também chamada de métrica “1-1”, no sentido de que, dado um ponto *p*, tanto os pixels que compartilham arestas quanto os que compartilham vértices com *p* possuem distância 1. A Figura 23 mostra a transformada de distância usando a métrica *chessboard*, aplicada à Figura 20.

a) Pixels vizinhos: topologia 8-conectado b) Campo distância gerado pela transformada.

FIGURA 24 - Transformada de distância usando a métrica *chessboard*

Fonte: BORGEFORS, 1986

2.6.6 Linhas Divisoras de Águas

A técnica de linhas divisoras de águas simula o preenchimento uniforme de um relevo por águas que o inundam a partir dos seus mínimos regionais (todos os pontos da função discreta cujos vizinhos são maiores que ele mesmo). O encontro de duas águas diferentes provenientes de dois destes mínimos define a LDA da função relevo ou imagem (OPENCV, 2014).

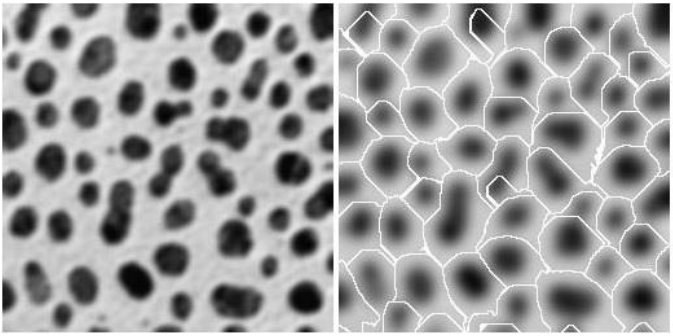


FIGURA 25 - Transformada de distância usando a métrica *chessboard*

FONTE: PRAJDIĆ, 2014

Essa técnica consiste em 2 fases:

- Inicialização: uma fila ordenada é criada contendo um número *l* de filas. Este número é o valor máximo presente na matriz da representação discreta da altura do relevo. Por exemplo, esta matriz pode conter valores no intervalo

[0,255], em que 0 corresponde ao nível mais baixo e 255, ao nível mais alto do relevo. Neste caso, *l* = 256 e cada fila simples da fila ordenada está associada a um nível da matriz. Em processamento de imagens este intervalo é conhecido como o intervalo dos níveis de cinza de uma imagem. Considere os mínimos regionais da função *f* como os pontos a partir dos quais o processo de preenchimento do relevo se inicia. Estes pontos são denominados marcadores. Cada um destes marcadores deve estar associado a uma etiqueta (rótulo) *etq* (um valor inteiro diferenciado para cada marcador: *etq* = 1, 2, ..., *n*) que possibilita a discriminação dos diferentes mínimos regionais (estas etiquetas podem ser vistas como águas coloridas que inundarão a superfície). Cada região será invadida pela etiqueta (água) do seu respectivo mínimo regional. O ponto de fronteira de um marcador é aquele que contém, na sua vizinhança, um outro ponto não pertencente a este marcador. Todos estes pontos de fronteira são inseridos na fila ordenada e o valor de cada ponto na matriz *f* determina o seu nível de prioridade nesta fila.

- Processamento: Crie uma nova matriz g, com a mesma dimensão de *f*, contendo as etiquetas do conjunto dos marcadores. A seguir, execute as seguintes etapas:

```
Enquanto fila ordenada não vazia

{

       Um cliente x é retirado da fila ordenada.
Para cada vizinho y de

x sem etiqueta em g faça:

         - atribua a mesma etiqueta de x a y na
matriz g.

         - insira o ponto y na fila ordenada; o
seu valor em f
```

```
      indica seu nível de prioridade na fila
ordenada.
}
```

3 IMPLEMENTAÇÃO DO PRÓTOTIPO

Neste capítulo são apresentadas, a estrutura geral proposta para o protótipo, detalhando o hardware empregado, as ferramentas de processamento de imagem e a implementação do algoritmo para contagem de células.

A Figura 25 apresenta o fluxo do processo da contagem de células sanguíneas. Com a imagem de entrada é processada no computador, através de um algoritmo de 4 passos (descrito nas próximas Seções). A interface gráfica mostra a imagem de entrada, sobrepondo uma marcação das células encontrada. Com um sistema *touch*, o usuário pode marcar facilmente as células que o sistema não tenha detectado.

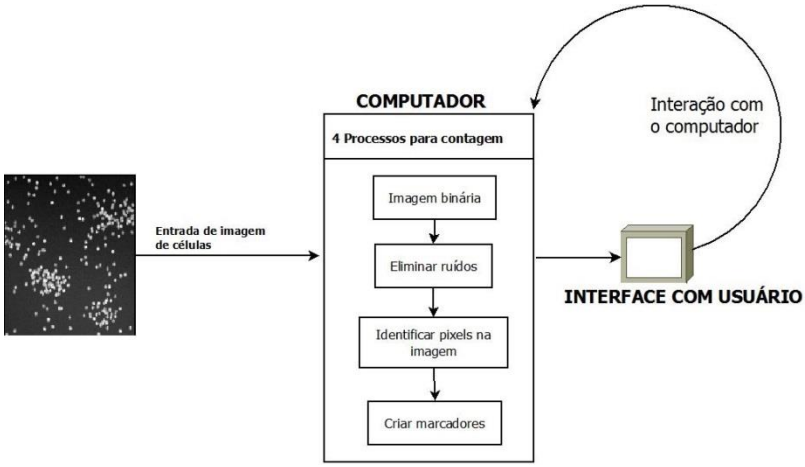


FIGURA 26 - Processos para a contagem de célula

Fonte: elaborada pelo autor

Para o escopo do projeto, é necessário avaliar alternativas tanto para o *hardware* quanto para o *software*, sendo assim, com todos os passos a serem seguidos até a finalização do produto, foram estudadas soluções para esses papéis. Dois critérios foram avaliados para a definição do *hardware* básico: a possibilidade de uma fácil interação do usuário com o

equipamento e o custo de aquisição do *hardware* para o desenvolvimento deste projeto.

Existem várias soluções no mercado, porém a que se mostrou viável, por conta do custo e logística de aquisição, foi a utilização do Raspberry Pi(Seção 2.5), com Linux embarcado, e a biblioteca OpenCV para prover funções de processamento de imagem. Porém antes, de iniciar a implementação nessa plataforma, verificou-se a viabilidade do desenvolvimento do protótipo.

3.1 Passo a Passo para contagem utilizando o OpenCV

Com as técnicas e algoritmos que a API (Application Programming Interface) do OpenCV dispõe, apresenta-se aqui o algoritmo para contar células por 4 passos. Para fins didáticos, mostra-se o exemplo com moedas, apresentando, passo a passo, o resultado de cada etapa.

FIGURA 27 - Imagem de moedas dispostas uma perto da outra

Fonte: MORDVINTSEV, 2014

O primeiro passo é converter a imagem original em uma imagem binária, como é visto na Figura 27.

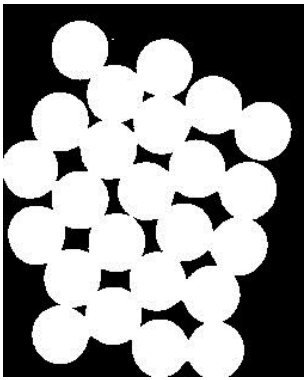


FIGURA 28 - Imagem das moedas depois de ter transformado em imagem de 8 bits

Fonte: MORDVINTSEV, 2014.

Feito a conversão da imagem, é aplicado a técnica de transformada da distância e se observa como fica o processo na Figura 28.

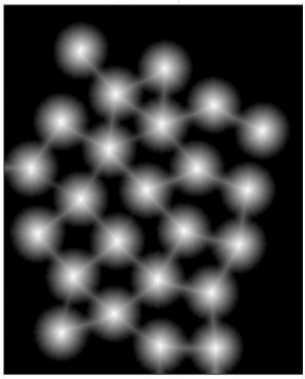


FIGURA 29 - As moedas depois de ser utilizada a técnica de transformada da distância

Fonte: MORDVINTSEV, 2014.

O próximo passo é utilizar a técnica de limiarização binária invertida, para segmentar as moedas, como mostra a Figura 29.

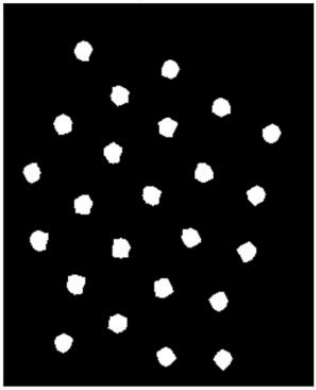


FIGURA 30 - As moedas depois de sido segmentada por limiarização binária invertida

Fonte: MORDVINTSEV, 2014.

Em seguida, inclu-se a erosão com a dilatação, como mostra a Figura 30, para obter os contornos, que serão usados pelas linhas divisoras de águas.

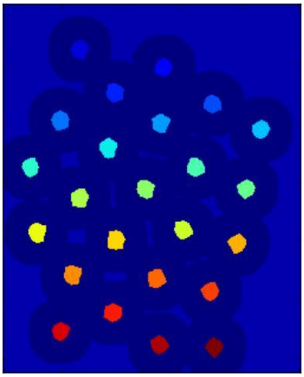


FIGURA 31 - Imagem das moedas com as técnicas de dilação e erosão

Fonte: MORDVINTSEV, 2014.

No próximo passo, aplicar-se a técnica de linhas divisoras de águas, assim colorindo com uma cor diferente cada moeda encontrada, como mostra a Figura 31.

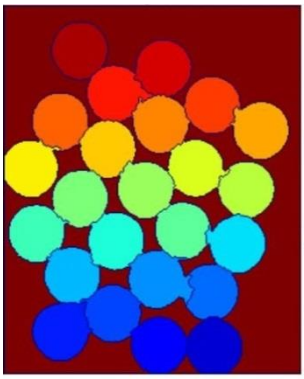


FIGURA 32 - Imagem das moedas depois de ter aplicado a técnica de linhas divisoras de águas

Fonte: MORDVINTSEV, 2014

E finalizando o processo, chega-se ao resultado da Figura 32, com os contornos que o algoritmo julgou serem moedas. Observa-se, neste exemplo, que o processo resultou em 100% de acerto.

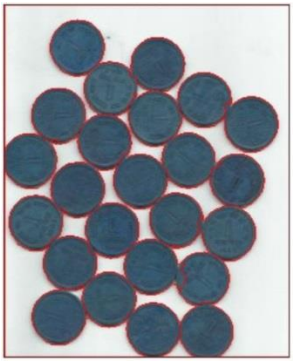


FIGURA 33 - Resultado final

Fonte: MORDVINTSEV, 2014

3.2 Interface com o usuário

Finalizando todo o processo, além do hardware e do software, é preciso a interação do usuário com o sistema. Apesar do processo ser boa parte automático, ainda é necessário que passe o resultado do algoritmo passe pelo crivo do usuário.

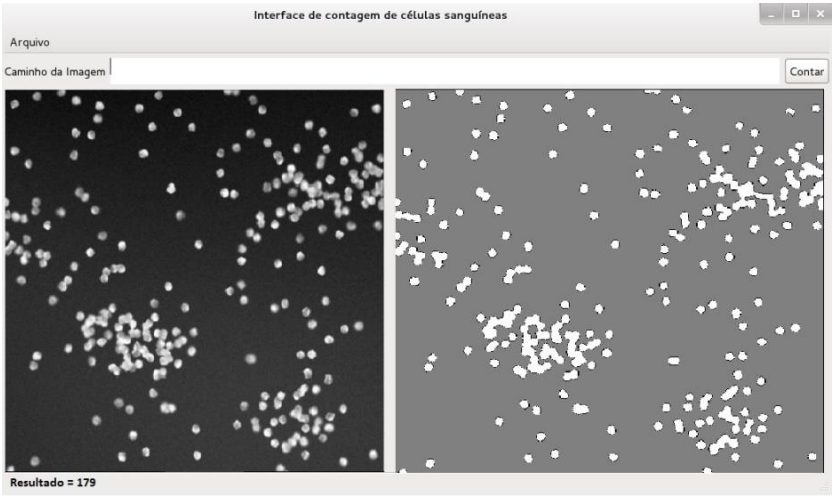


FIGURA 34 - Interface do usuário feito em C# no Linux

Fonte: elaborada pelo autor

No Linux embarcado do Raspberry Pi, pode ser utilizado o C# como solução para front-end. Além de utilizar boa parte do algoritmo da prototipagem no Visual Studio feito no início do projeto, o ambiente do Monodevelop do Linux, consegue cumprir a tarefa com bastante semelhança, apesar de que a produtividade cai consideravelmente em comparação com a ferramenta do Windows. O Monodevelop é muito poderoso e cumpre de forma adequada seu papel nesse projeto, como pode ser visto na Figura 33. No sistema, o usuário informa a imagem e utilizando os algoritmos do OpenCV, eles marcam as células que são possíveis de serem reconhecidas. No primeiro quadro é mostrado a imagem original e ao lado é o resultado da imagem

com o seus marcadores. Caso o usuário julgue que o sistema não contou uma determinada célula da imagem original, o usuário marca manualmente a célula restante que o sistema não reconheceu.

4 TESTES E RESULTADOS OBTIDOS

4.1 Conjunto de dados de teste

Como o protótipo inicialmente não possui um microscópio eletrônico foram utilizadas imagens de células de laboratórios. O instituto *Broad Image* (BROAD INSTITUTE, 2014), disponibiliza em seu site imagens de células de vários tipos. Para fazer o teste, foram utilizadas 5 imagens de células sintéticas, com diferentes graus de sobreposição: nenhuma, 15, 30, 45 e 60 %. Todas as imagens possuem 300 células como informado pelo instituto *Broad Image*.

4.2 Procedimento

Na Figura 34, é possível observar todos os processos que a imagem passa, até chegar ao resultado final. O resultado é praticamente instantâneo quando é executado o algoritmo nestas imagens, que tem tamanho 480x320 pixels.

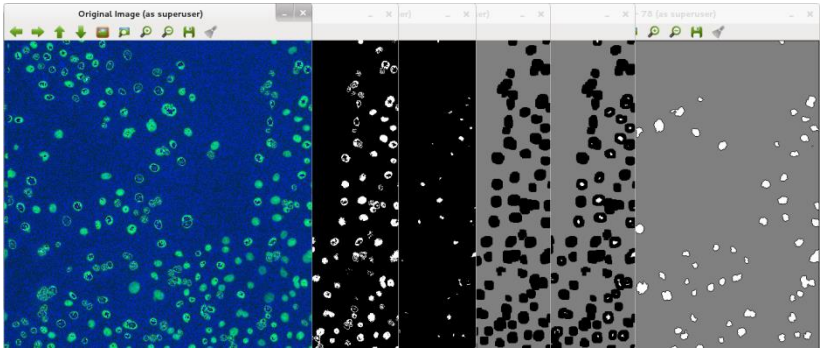


FIGURA 35 - Imagem que demonstra todo o processo de filtragem até a contagem

Fonte: elaborada pelo autor

4.3 Condições de teste

No primeiro caso, onde tem-se que transformar a imagem em imagem binária, o valor de limiarização foi 100, sendo o máximo valor de até 255. O tipo de limiarização foi binária.

Já na identificação dos pixels da imagem, é utilizada uma função para dilatar a imagem, utilizando o elemento da estrutura que determina a forma que um vizinho do pixel pode expandir. A técnica limiarização é chamada novamente com o valor 1, com o valor máximo de até 128, com tipo de limiarização binária invertida. Deve ser calculada a transformada de distância, que é a distância mais próxima de zero de pixel para pixel da imagem. Foram utilizadas apenas 3 iterações no cálculo.

4.4 Taxa de acerto de teste

Tendo as imagens sintéticas e os parâmetros para contagem, foi feito a contagem tanto no *ImageJ*, para caráter de comparação quanto no *OpenCV*. Do lado direito de cada coluna de cada software, tem a porcentagem em relação do total de células que foi informado pela *Broad Image*. Pode ser visto sem sobreposição na Tabela 1, com 15% de sobreposição na Tabela 2, com 30% de sobreposição na Tabela 3, com 45% de sobreposição na Tabela 4 e por fim, com 60% de sobreposição na Tabela 5. É possível ver a imagem de células sem sobreposição na Figura 35 e imagem de células com 60% de sobreposição na Figura 36.

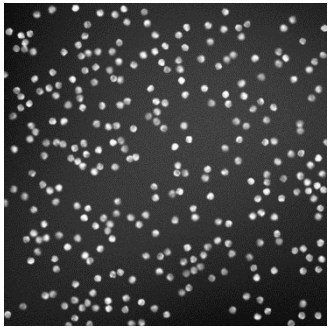


FIGURA 36 - Imagem de células sintéticas sem sobreposição

Fonte: INSTITUTE, 2014

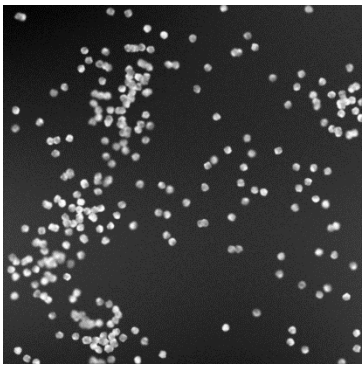


FIGURA 37 - Imagem de células sintéticas com 60% de sobreposição

Fonte: INSTITUTE, 2014

Imagem	Total	ImageJ	%	OpenCV	%
1	300	300	100	287	95,66
2	300	300	100	286	95,33
3	300	300	100	283	94,33
4	300	300	100	280	93,33
5	300	300	100	288	96

TABELA 1 - Nenhuma sobreposição de células sintéticas

Fonte: elaborada pelo autor

Imagem	Total	ImageJ	%	OpenCV	%
1	300	278	92,66	225	75
2	300	276	92	233	77,66
3	300	272	90,66	231	77
4	300	273	91	219	73
5	300	272	90,66	217	72,33

TABELA 2 - 15% de sobreposição das células sintéticas

Fonte: elaborada pelo autor

Imagem	Total	ImageJ	%	OpenCV	%
1	300	277	92,33	221	73,66
2	300	253	84,33	191	63,66
3	300	261	87	201	67
4	300	272	90,66	217	72,33
5	300	269	89,66	210	70

TABELA 3 - 30% de sobreposição das células sintéticas

Fonte: elaborada pelo autor

Imagem	Total	ImageJ	%	OpenCV	%
1	300	257	85,66	199	66,33
2	300	258	86	191	63,66
3	300	260	86,66	210	70
4	300	260	86,66	207	69
5	300	269	89,66	209	69,66

TABELA 4 - 45% de sobreposição das células sintéticas

Fonte: elaborada pelo autor

Imagem	Total	ImageJ	%	OpenCV	%
1	300	236	78,66	167	55,66
2	300	227	75,66	173	57,66
3	300	239	79,66	167	55,66
4	300	235	78,33	166	55,33
5	300	251	83,66	179	59,66

TABELA 5 - 60% de sobreposição das células sintéticas

Fonte: elaborada pelo autor

4.5 Resultados com imagem real de células sanguíneas

Neste capítulo são apresentados os resultados obtidos com imagens de células reais em cada etapa do algoritmo de contagem de células. Foram utilizados os mesmo parâmetros no teste com as células sintéticas, para fazer a comparação na porcentagem final.

Para fazer a validação, foram utilizadas 11 imagens de células de câncer de cólon, disponibilizados pela *Broad Image* (Figura 37) (BROAD INSTITUTE, 2014).

4.6 Taxa de acerto com imagem real de células sanguíneas

Como nessa amostragem disponibilizada pela *Broad Image*, não é fornecido a referência (total de células por imagem), foi feito o cálculo de porcentagem de acerto só em relação entre o *ImageJ* e *OpenCV,* como é observado na Tabela 6.

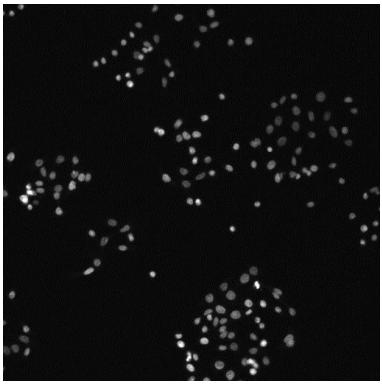


FIGURA 38 - Imagem de células reais de câncer de cólon

Fonte: BROAD INSTITUTE, 2014

Imagem	ImageJ	OpenCV	%
1	171	112	65,49
2	326	202	61,96
3	273	163	59,70
4	180	101	56,11
5	67	40	59,70
6	160	93	58,12
7	90	7	7,78
8	94	23	24,46
9	53	30	56,60
10	77	13	16,88
11	33	16	48,48

TABELA 6 - Resultado final da contagem de células sanguíneas com os 2 softwares (ImageJ e OpenCV)

Fonte: elaborada pelo autor

5 CONCLUSÃO

Foi desenvolvido um protótipo para auxiliar a contagem de células sanguíneas, passando por todas as etapas. Foram estudados as técnicas de contagem de células sanguíneas por meio de várias dissertações, livros e artigos, além de pesquisar produtos que tem a mesma finalidade, porém com características diferentes. O objetivo proposto de construir um hardware foi atingido em parte, já que, como comentado a seguir, no Capítulo 6, infelizmente o microscópio digital USB e o LCD touchscreen não puderam ser adquiridos. No entanto, o Raspberry Pi foi adquirido, feitos os testes no seu sistema operacional e avaliados os resultados obtidos.

O software foi construído com todos os passos para auxiliar a contagem de células sanguíneas, com a arquitetura do OpenCV, e logo depois desenvolvido outro software feito em C# para interface do usuário, que importa para seu ambiente o software que conta as células sanguíneas.

Por fim, foram comparados e avaliados os resultados com as amostras tanto com células sintéticas quanto células reais. O ImageJ consegue chegar mais próximo do ideal, principalmente chegando a 100% de acerto quando a imagem das células não possuem sobreposição. Ainda que o protótipo proposto tivesse que utilizar o OpenCV, ele cumpriu o papel, observando que a contagem de células feita manualmente é muito mais complicada e árdua.

6 MELHORIAS E TRABALHOS FUTUROS

Na proposta inicial, planejou-se a utilização de um microscópio digital USB, que pode ser visto no ANEXO D. Infelizmente, o preço desse dispositivo, no início do projeto, era relativamente alto. No decorrer do desenvolvimento, após consulta a vários sites de venda de produtos eletrônicos, verificou-se que o microscópio teve o preço reduzido para menos de R$80,00. Como essa redução só ocorreu em um momento avançado do projeto, a logística de entrega não permitiu a aquisição em tempo hábil para o desenvolvimento do projeto. O microscópio pesquisado é compatível com vários sistemas operacionais, inclusive o Linux que é utilizado no Raspberry Pi, então parece viável para a continuidade do projeto.

Além do microscópio digital USB, outro problema no decorrer do projeto, foi o funcionamento do LCD touchscreen, que pode ser visto no ANEXO E, no Raspberry Pi. No projeto, para fazer os testes e ser viável sua construção, foi utilizado um monitor e mouse de computador. Porém para melhorar a interação com o usuário, e seguir o desenho proposto para esse projeto, o ideal seria utilizar o LCD touchscreen. Também no inicio do projeto, os drivers para LCD touchscreen para o Raspberry Pi estavam em desenvolvimento pela comunidade de software livre. Com a inclusão desses dois elementos atingiria-se o formato proposto originalmente.

Tendo em vista que o OpenCV está em constante atualização pela comunidade de software livre, as técnicas utilizadas neste projeto podem ser alteradas e melhoradas.

REFERÊNCIAS BIBLIOGRÁFICAS

BASTIDAS, Oscar. **Cell Counting with Neubauer Chamber – Basic Hemocytometer Usage**. Celetromics, 2013.

BIORAD. **Counting Cells with Bio-Rad's TC20™ Automated cell counter**. Disponível em: <http://www.bio-rad.com/en-us/category/cell-counting?WT.mc_id=yt-GXD-ww-tc20-20120907-sDHOL7UEL1M>. Acesso em: 10 mar de 2014.

BORGEFORS, G. **Distance Transformations on Digital Images. Computer Vision and Image Processing**, 34, 344-371, 1986.

BRADSKI, Gary. KAEHLER, Adrian. **Learning OpenCV: computer vision with the OpenCV library**. O' Reilly, 2008.

CARVALHO, W. F. **Técnicas Médicas de Hematologia e Imuno-hematologia** - 8ª Ed. Coopmed, 2008.

FERREIRA, Tiago. RASBAND, Wayne. **ImageJ User Guide**. Fiji 1.46, 2012.

GUO, Xiaomin. YU, Feihong. **A Method of Automatic Cell Counting Based on Microscopic Image.** Optical Engineering Department, Zhejiang University, Hangzhou. 2013.

HEMOCYTOMETER. Disponível em < http://hemocytometer.wordpress.com/2013/04/11/hemocytometer-square-size/>. Acesso em: 10 mar. 2014.

HU, Fan-Jin. **The Newest 2MP 8 LED 800X USB Digital Microscope Endoscope Magnifier Camera**. Disponível em: <http://www.aliexpress.com/item/The-Newest-2MP-8-LED-800X-USB-Digital-Microscope-Endoscope-Magnifier-Camera/610144554.html>. Acesso em: 10 mar de 2014.

IMAGEJ. Disponível em: <http://imagej.nih.gov/ij/plugins/index.html>. Acesso em: 10 mar de 2014.

BROAD INSTITUTE. **Broad Bioimage Benchmark Collection. Annotated biological image sets for testing and validation**. Disponível em:<http://www.broadinstitute.org/bbbc/image_sets.html>. Acesso em: 10 mar de 2014.

JIN, J.S. PENG, Y. SUMMONS, P. YU, D. CUI, Y. LUO, S. WANG, F. SANTOS, L. XU, M. **Automatic cell segmentation in microscopic color images using ellipse fitting and watershed.** M. Park, Member, IEEE. Gold Coast, Australia, 2010.

LABMEDVET. Disponível em: <http://labmedvet.blogspot.com/2011/12/como-contar-leucocitos-plaquetas-e.html>. Acesso em: 10 mar de 2014.

LABNO, Christine. Two Ways to Count Cells with ImageJ. Disponível em: <http://digital.bsd.uchicago.edu/resources_files/cell%20counting%20automated%20and%20manual.pdf>. Acesso em: 10 mar de 2014.

MAYO, Joe. **Microsoft ® Visual Studio® 2010 - A Beginner's Guide**. McGraw Hill, 2010.

MINDRAY. **Analisador automático de hematologia BC-5800**. Disponível em: <http://www.mindray.com/pt/products/25.html>. Acesso em: 10 mar de 2014.

MITCHELL, Michael. **Raspberry Pi with OpenCV**. Disponível em: <http://mitchtech.net/raspberry-pi-opencv/>. Acesso em: 10 mar de 2014.

MORDVINTSEV, Alexander. K., Abid. **Watershed**. Disponível em: <http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html>. Acesso em: 10 mar de 2014.

OPENCV. Disponível em: <http://opencv.org/>. Acesso em: 10 mar de 2014.

PRAJDIĆ, Mladen. **Watershed Image Segmentation in C#**. Disponível em: <http://weblogs.sqlteam.com/mladenp/archive/2008/02/11/Watershed-Image-Segmentation-in-C.aspx>. Acesso em: 10 mar de 2014.

RASPBERRY PI. Disponível em: <http://www.raspberrypi.org/>. Acesso em: 10 mar de 2014.

RICHARDSON, Matt. WALLACE, Shawn. **Getting started with Raspberry Pi**. O' Reilly, 2013.

KOERICH, Alessandro L. **Instalando Visual Studio 2010 Express com o OpenCV**. Disponível em: <http://www.ppgia.pucpr.br/~alekoe/AM/2013/InstalacaoOPENCV-2.4.txt>. Acesso em: 10 mar de 2014.

YANG, Jennifer. **Tontec 7" Raspberry Pi LCD Touch Screen Display TFT Monitor AT070TN90 with Touchscreen Kit HDMI VGA Input Driver Board**. Disponível em: <http://www.aliexpress.com/item/Tontec-7-Raspberry-Pi-LCD-Touch-Screen-Display-TFT-Monitor-AT070TN90-with-Touchscreen-Kit-HDMI-VGA/1745270139.html>. Acesso em: 10 mar de 2014.

ANEXOS

ANEXO A – INSTALAÇÃO DO OPENCV NO VISUAL STUDIO 2010

PASSO 1: Instalar o Microsoft Visual Studio 2010 Express (KOERICH, 2014)

1) Fazer o download do Visual C++ 2010 Express em: http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express

2) Não é necessário instalar nenhum produto opcional (Silverlight ou SQL).

3) O diretório padrão é: C:\Program Files (x86)\Microsoft Visual Studio 10.0\

4) O arquivo será baixado durante a instalação e tem aproximadamente 70MB.

PASSO 2: Instalar a Biblioteca OPENCV 2.4.6

1) Fazer download do arquivo OpenCV-2.4.6.exe em http://downloads.sourceforge.net/project/opencvlibrary/opencv-win/2.4.6/opencv-2.4.6.exe

2) Instalar/descompactar diretamente em C:\opencv para ficar com os "caminhos mais curtos". Atenção, a biblioteca tem 2,3GB descompactada!

PASSO 3: Configurar o Visual Studio para utilizar a OPENCV

Deve ser criado um projeto teste com o OPENCV. Clicar sobre o projeto com o botão direito do mouse e selecionar Properties. Nas propriedades do projeto:

1) Configurar primeira para DEBUG. Assim, selecione no canto superior esquerdo CONFIGURATION: DEBUG

A partir daí:

Em CONFIGURATION PROPERTIES

-> C/C++
-> General

-> ADDITIONAL INCLUDE DIRECTORIES

C:\opencv\build\include

C:\opencv\build\include\opencv

->LINKER
-> General
-> ADDITIONAL LIBRARY DIRECTORIES

C:\opencv\build\x86\vc10\lib

*****C:\opencv\build\x86\vc10\bin

-> Input
-> ADDITIONAL DEPENDENCIES
opencv_calib3d246d.lib
opencv_contrib246d.lib
opencv_core246d.lib
opencv_features2d246d.lib
opencv_flann246d.lib
opencv_gpu246d.lib
opencv_haartraining_engined.lib
opencv_highgui246d.lib
opencv_imgproc246d.lib
opencv_legacy246d.lib
opencv_ml246d.lib
opencv_nonfree246d.lib
opencv_objdetect246d.lib
opencv_photo246d.lib
opencv_stitching246d.lib
opencv_ts246d.lib
opencv_video246d.lib
opencv_videostab246d.lib

(OBS: As lib acima são somente para a configuração DEBUG).

No final, apertar o botão *Apply* para gravar as configurações.

Voltar lá em cima e selecionar CONFIGURATION: RELEASE e refazer tudo, mudando somente o último passo,

onde as libs abaixo devem ser adicionadas ao invés das anteriores.

Libraries para RELEASE:

opencv_calib3d246.lib
opencv_contrib246.lib
opencv_core246.lib
opencv_features2d246.lib
opencv_flann246.lib
opencv_gpu246.lib
opencv_haartraining_engine.lib
opencv_highgui246.lib
opencv_imgproc246.lib
opencv_legacy246.lib
opencv_ml246.lib
opencv_nonfree246.lib
opencv_objdetect246.lib
opencv_photo246.lib
opencv_stitching246.lib
opencv_ts246.lib
opencv_video246.lib
opencv_videostab246.lib

PASSO 4: Configurar o PATH do Windows

O instalador do OpenCV deve fazer isso, mas pode haver algum problema de permissão.

Se houver problema para executar os arquivos compilados no VS2010, com uma mensagem de DLL não encontrada, será necessário inserir no PATH do Windows o caminho para a pasta onde estão as DLLs do OPENCV. (KOERICH, 2014)

COMPUTER -> PROPERTIES -> ADVANCED SYSTEM SETTINGS -> ENVIRONMENT VARIABLES

Em "System variables", editar a variável PATH e adicionar:

;C:\opencv\build\x86\vc10\bin

Pode ser necessário adicionar também no PATH este caminho aqui:

;C:\opencv\build\common\tbb\ia32\vc10

ANEXO B – INSTALAÇÃO E CONFIGURAÇÃO DO OPENCV NO LINUX

OpenCV é uma biblioteca de visão computacional. Seguem os passos que foram utilizados para instalar o OpenCV no Raspberry Pi com Debian Wheezy com data de release 10-09-2013. É bom se certificar de ter expandido o cartão SD para permitir a instalação do OpenCV. É um pacote grande, com muitas dependências. Depois de ter expandido o cartão SD, abra um terminal e instale os seguintes pacotes: (MITCHELL, 2014)

```
sudo apt-get -y install build-essential cmake
cmake-qt-gui pkg-config libpng12-0 libpng12-dev
libpng++-dev libpng3 libpnglite-dev zlib1g-dbg
zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4
libtiffxx0c2 libtiff-tools


sudo apt-get -y install libjpeg8 libjpeg8-dev
libjpeg8-dbg libjpeg-progs ffmpeg libavcodec-dev
libavcodec53 libavformat53 libavformat-dev
libgstreamer0.10-0-dbg libgstreamer0.10-0
libgstreamer0.10-dev libxine1-ffmpeg libxine-dev
libxine1-bin libunicap2 libunicap2-dev libdc1394-
22-dev libdc1394-22 libdc1394-utils swig libv4l-0
libv4l-dev python-numpy libpython2.6 python-dev
python2.6-dev libgtk2.0-dev pkg-config
```

Existem alguns problemas de dependência no final da instalação, principalmente no que diz respeito à biblioteca libjpeg, por isso é bom certificar de instalar nessa ordem. Alguns erros podem aparecer se caso for tentar instalar todas as dependências em uma única etapa.

Em seguida, baixe os arquivos de origem para OpenCV usando wget: (OPENCV, 2014)

```
wget
http://sourceforge.net/projects/opencvlibrary/files
/opencv-unix/2.4.6/opencv-2.4.6.tar.gz
```

Uma vez terminado o download, descompacte o arquivo, remova o arquivo caso não seja mais necessário (para economizar espaço), altere o diretório para a raiz do programa, crie um diretório para a compilação, e mude para ele:

```
tar -xvf opencv-2.4.6.tar.gz
rm opencv-2.4.6.tar.gz
cd opencv-2.4.6/
mkdir build
cd build
```

Em seguida, é necessário configurar a build usando cmake. Se caso não tiver certeza sobre quais as opções queira / precisa ou não estão estiver familiarizados com cmake, esta linha vai criar uma configuração padrão:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D
BUILD_PYTHON_SUPPORT=ON -D BUILD_EXAMPLES=ON ..
```

Alternativamente, pode configurar a compilação usando uma interface GUI. Isso pode ser útil para construir o pacote com suporte para recursos do OpenCV adicionais. Para usar o GUI cmake, execute:

```
cmake-gui ..
```

Na GUI cmake, clique em 'configure' para preencher as opções de compilação. Selecione ou remova quaisquer características desejadas e clique em 'configure' novamente, verifique a saída e que não exista quaisquer módulos que cmake não consegue encontrar. Se tudo estiver adequado, clique em "Gerar" para criar os makefiles e então feche cmake-gui. Agora está pronto para começar a compilação da biblioteca. Para compilar, execute o make, e em seguida, instale com make install:

```
make
sudo make install
```

Isso vai levar um longo tempo, por volta de quatro horas e meia para compilar.

Finalmente, é preciso fazer algumas configurações para OpenCV. Primeiro, abra o arquivo opencv.conf com o seguinte código:

```
sudo nano /etc/ld.so.conf.d/opencv.conf
```

Adicione a seguinte linha no final do arquivo (que pode ser um arquivo vazio) e depois salvá-lo:

```
/usr/local/lib
```

Em seguida, edite o arquivo bashrc de todo o sistema:

```
sudo nano /etc/bash.bashrc
```

Adicione as seguintes linhas no final do arquivo:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkg
config export PKG_CONFIG_PATH
```

Agora que tudo está instalado e configurado, para as demos. Os demos C está aqui:

```
cd ~/opencv/opencv-2.4.6/build/bin
```

ANEXO C – CÁLCULO DE CONTAGEM DE CÉLULAS SANGUÍNEAS UTILIZANDO UM HEMOCITÔMETRO

$$\frac{Vf}{Vi\ x\ Vc}\ x\ n^{o}\ de\ estruturas\ contadas = /mL$$

- Vf=Volume final → 01 mL
- Vi= Volume inicial → 10 mL
- Vc=Volume contado na câmara

Volume em 1□ = 0,0001 mL (Fator 1000 – Se contar em um quadrante);
Volume em 2□ = 0,0002 mL (Fator 500 – Se contar em dois quadrantes);
Volume em 3□ = 0,0003 mL (Fator 333 – Se contar em três quadrantes);
Volume em 4□ = 0,0004 mL (Fator 250 – Se contar em quatro quadrantes);

Contou-se nos quatro quadrantes 20 leucócitos, 26 hemácias e 14 células epiteliais. Vf= 01 mL e Vi= 10 mL.

$$\frac{Vf}{Vi\ x\ Vc}\ x\ n^{o}\ de\ estruturas\ contadas = /mL$$

$$\frac{01}{10\ x\ 0{,}0004}\ x\ n^{o}\ de\ estruturas\ contadas = /mL$$

$$250\ x\ n^{o}\ de\ estruturas\ contadas = /mL$$

$250\ x\ 20\ leucócitos = 5.000\ /mL$
$250\ x\ 26\ Hemácias = 6.500\ /mL$
$250\ x\ 14\ cél.epiteliais = 3.500\ /mL$

Exemplo para demonstração o cálculo de concentração de células sanguíneas (CARVALHO, 2008)

ANEXO D – MICROSCÓPIO DIGITAL USB

Fonte: HU, 2014

Especificações

Resolução: 2.0 Megapixels
Foco: 10mm ~ 500mm
Formato de imagem: jpeg / bmp
Formato de vídeo: AVI
Vídeo / Resolução de captura de imagem: 1600 x 1200, 1280 x 960, 800 x 600, 640 x 480, 352 x 288, 320 x 240, 160 x 120
Fonte de luz: 8 leds de luz branca
Frame rate (vídeo): 30 fps
Ampliação: 20x ~ 800x
Alimentação: 5V da porta USB
Interface: USB 2.0
Sistema Operacional: Windows 2000/XP/Vista/Win 7/Mac/Linux

ANEXO E - LCD 7’ + Touchscreen

Fonte: YANG, 2014

Especificações

Alimentação	5V DC
	3.3V via expansion header
	5V via expansion header
Display	7" TFT LCD
Resolução	800x480
Touchscreen	4-wire
Indicadores	One Power LED