PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO


HIGOR ALVES FERREIRA
LUCAS MACEDO DA SILVA
VITOR DE ALMEIDA SILVA

# CLASSIFICAÇÃO DE CELULAS CONTAMINADAS POR MALÁRIA

Goiânia,
2020/01

HIGOR ALVES FERREIRA

LUCAS MACEDO DA SILVA

VITOR DE ALMEIDA SILVA

# CLASSIFICAÇÃO DE CÉLULAS CONTAMINADAS POR MALÁRIA

Trabalho apresentado como requisito parcial para obtenção de nota na disciplina Processamento Digital de Imagens no Curso de Engenharia da computação, na Pontifícia Universidade Católica de Goiás.

Arlindo Rodrigues Galvão Filho

GOIÂNIA,

2020/01

**Sumário**

## 1. Introdução

O presente trabalho, tem o propósito de utilizar os conceitos de processamento digital de imagem (PDI), vistos ao longo da disciplina na classificação de células infectadas por malária. Deste modo, é proposto a criação de um script em Matlab, que faz o pré-processamento das imagens, para aprimorar os resultados do classificador PCA.

O documento está dividido em seções apresentando as discussões sobre cada tópico abordado ao longo do desenvolvimento do trabalho. A seção 2 descreve o *dataset* empregado no trabalho. A seção 3 resume brevemente a forma que é realizada a classificação manual de malária. A seção 4 apresenta sucintamente a análise de componentes principais empregada para a classificação. A seção 5 apresenta as técnicas de processamento digital de imagens empregadas e as duas propostas de pré-processamento. A seção 6 descreve os resultados. A seção 7 apresenta a conclusão. Por fim, a seção 8 apresenta as referências bibliográficas. Os códigos utilizados estão presentes nos apêndices deste trabalho.

## 2. Descrição do Dataset

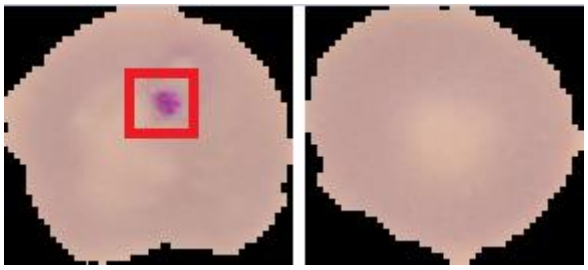
O *dataset* (conjunto de dados) utilizado para a classificação com o *Principal Component Analysis* (PCA) é um *dataset* de células infectadas/ não infectadas pela malária, retirado de Jaeger (2020). Este *dataset* conta com duas classes (JAEGER, 2020):

- ***Uninfected* (Não infectada):** Contém as imagens das células não infectadas pela **malária.** O total de imagens dessa classe é de 13780;
- ***Parasitized* (Infectada):** Contém as imagens das células infectadas pela malária. O total de imagens dessa classe é de 13780.

Existem, portanto, 27560 imagens em RGB no banco de dados. Devido ao tamanho do banco de dados o processamento dos dados acaba sendo demasiadamente lento portanto, ele foi subamostrado aleatoriamente. Em tal subamostrado, foram selecionadas **880 imagens** de cada classe, dessa forma o novo banco de dados é aproximadamente 6,386% do banco de dados original. Possuindo no total 1760 imagens.

No banco de dados células infectadas contém o *Plasmodium*, o parasita que causa a malária, já as células não infectadas, não contém o parasita (RAHMAN et al., 2019). Junto a classe de células infectadas, existem células não infectadas, porém, que foram retiradas de um indivíduo infectado (JAEGER, 2020). A Figura 1 ilustra um exemplo de célula infectada a esquerda e de uma célula não infectada a direita.

Figura 1 – Célula infectada, com parasita destacado em vermelho à esquerda. Célula não infectada à direita



Fonte: Autoral

Conforme destacado em vermelho na Figura 1, o parasita está visível na imagem. Entretanto, nem todas as células presentes na classe de infectadas estão infectadas pelo parasita, como pode ser visto na Figura 2, onde se tem uma célula não infectada, porém, presente na classe das infectadas. Isso porque, ela foi retirada de um paciente infectado;

Figura 2 – Célula da classe infectada, porém, não infectada.

Fonte: Autoral

Como essas células são uma minoria no banco de dados original, foram selecionadas manualmente 80 imagens com essa característica e adicionadas na classe de infectadas.

## 3. Classificação manual da malária

Para diagnosticar a malária convencionalmente, amostras de sangue do paciente a ser testadas são colocadas em uma lâmina e são observadas no microscópio para que seja realizada a contagem de hemácias infectadas. Porém, esse processo é lento e cansativo (KUNWAR; SHRESTHA; SHIKHRAKAR, 2018).

Figura 3 – Diagnostico manual da Malária

Fonte: (ALMEIDA, 2020)

Dessa forma, é importante desenvolver sistemas capazes de determinar se uma pessoa possui ou não malária. Com isso, pretende se desenvolver um classificador, utilizando a PCA e técnicas PDI para construir um classificador de células infectadas por malária utilizando o banco de dados supracitado.

## 4. Análise de Componentes Principais

Análise de Componentes Principais (em inglês *Principal Component Analysis* – PCA). A PCA, consiste em uma formulação matemática que permite reduzir a dimensionalidade dos dados e permite a identificação de padrões nos dados de maneira que suas semelhanças e diferenças sejam detectadas (SANTO, 2012). Sendo assim, é possível utilizar a PCA como algoritmo de classificação para o presente trabalho.

## 5. Técnicas de Processamento de Imagens

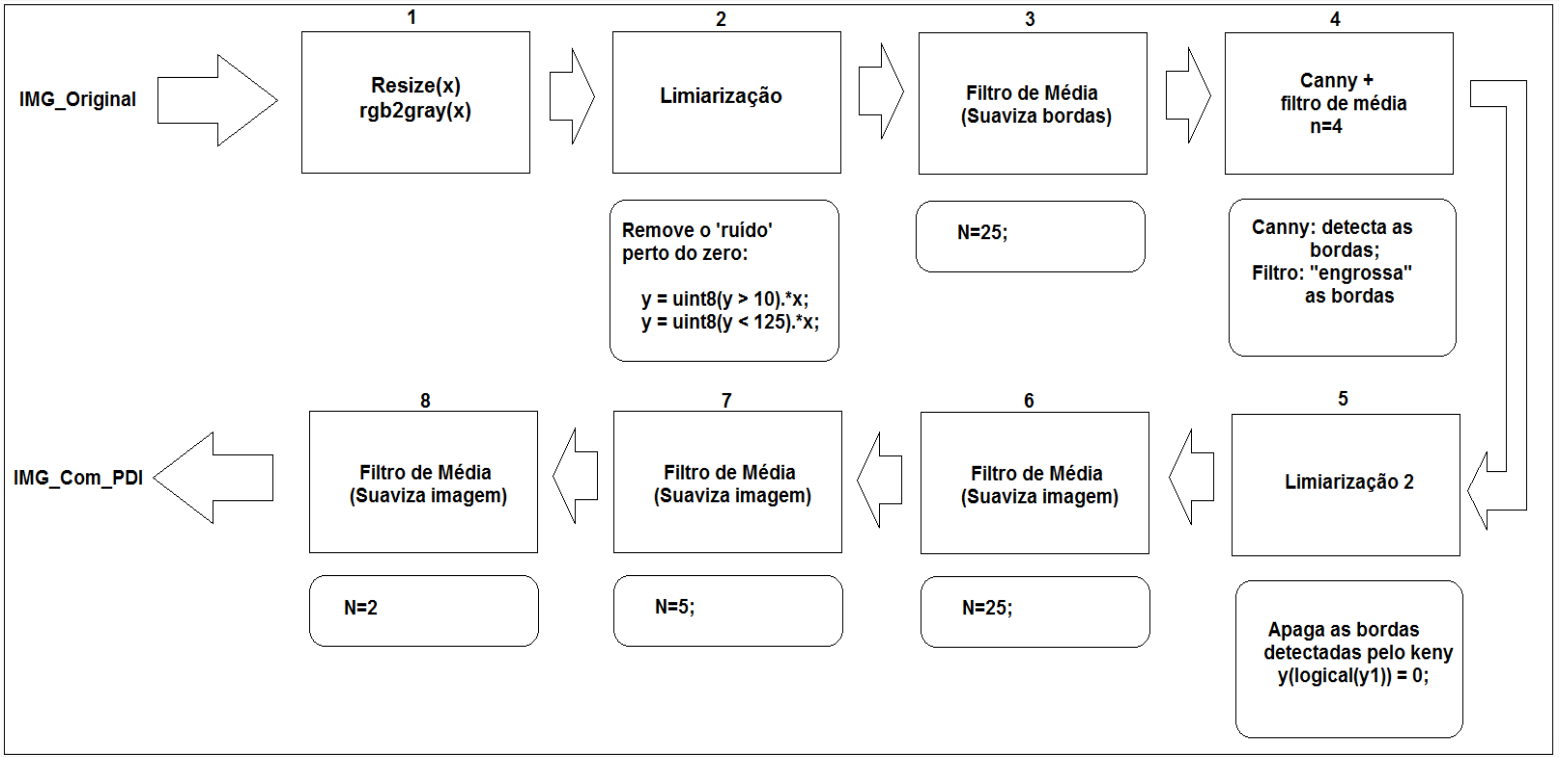
De acordo com a proposta, foi desenvolvido uma função para aplicar as técnicas de PDI na imagem antes da classificação. Deste modo, de acordo com as análises das características e propriedades das imagens, foram levantadas duas

possíveis soluções ambas baseadas em segmentação da imagem e separação do parasita.

### 5.1. Proposta 1 – Limiarização e filtragem com filtros de média

A primeira proposta consistiu na separação do parasita para utilizá-lo como característica para classificação. A Figura 4 apresenta o fluxograma do processamento das imagens

Figura 4: Fluxograma do PDI




Fonte: Autoral

Para uma total compreensão acerca das análises realizadas para gerar o fluxograma de 8 estados, faz se necessário expor algumas definições. Tais, definições são realizadas nos seguintes pontos:

1) **Redimensionamento e escala de cinza:** Como citado anteriormente, foram observadas imagens de tamanhos irregulares no dataset. Deste modo, o tamanho delas foi padronizado para **150x150**. Da mesma forma, todas foram transformadas em **escalas de cinza** para serem passadas para o PCA por meio da função *reshape();*
2) **Limiarização:** Foi realizada para remover alguns ruídos desnecessários da imagem. Além de retirar parte do preenchimento interior da célula, evidenciando o parasita e as bordas;
3) **Filtro de média:** Foi utilizado algumas vezes para **suavizar as imagens** das células. Assim como os outros filtros no domínio do tempo, o filtro de média aplicado em uma imagem, consiste na realização de uma convolução ou correlação da matriz dessa imagem com uma máscara. No caso do filtro de média, é gerado um efeito de suavização na imagem (GONZALEZ, WOODS, 2010).

Tanto a correlação quanto a convolução, consistem no somatório de produtos que é realizado entre a máscara e a imagem. Desse modo, o efeito gerado por esse processo é o de uma média ponderada, onde são realizados cálculos com a máscara deslocando sobre a matriz da imagem, fazendo média dos pixels vizinhos a cada interação (GONZALEZ, WOODS, 2010).

A diferença entre a convolução e a correlação, são os sinais dos coeficientes de deslocamento, que na convolução, são negativos e provocam uma rotação na imagem de 180 graus. Com isso, a equação 1 e equação 2 mostra um as operações de correlação e de convolução respectivamente (GONZALEZ, WOODS, 2010).

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \ (equação\ 1)$$

Onde,

$w(x,y)$ é a máscara, também conhecida como filtro, que percorre a imagem;

$f(x,y)$ é a imagem;

$a = \frac{m-1}{2}$, onde m é um número inteiro e ímpar;

$b = \frac{n-1}{2}$ , onde m é um número inteiro e ímpar;

$$w(x,y)\circledast f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t) \ \ (equação\ 2)$$

Onde,

$w(x,y)$ é a máscara, também conhecida como filtro, que percorre a imagem;

$f(x,y)$ é a imagem;

$a = \frac{m-1}{2}$, onde m corresponde a dimensão do filtro, e é um número inteiro e ímpar;

$b = \frac{n-1}{2}$ , onde n corresponde a dimensão do filtro, é um número inteiro e ímpar;

As máscaras, por sua vez, são matrizes nxn de coeficientes gerados com base no efeito que ele deve ter sobre a imagem. A Figura 4 mostra o cálculo realizado no Matlab para gerar uma máscara de tamanho n.

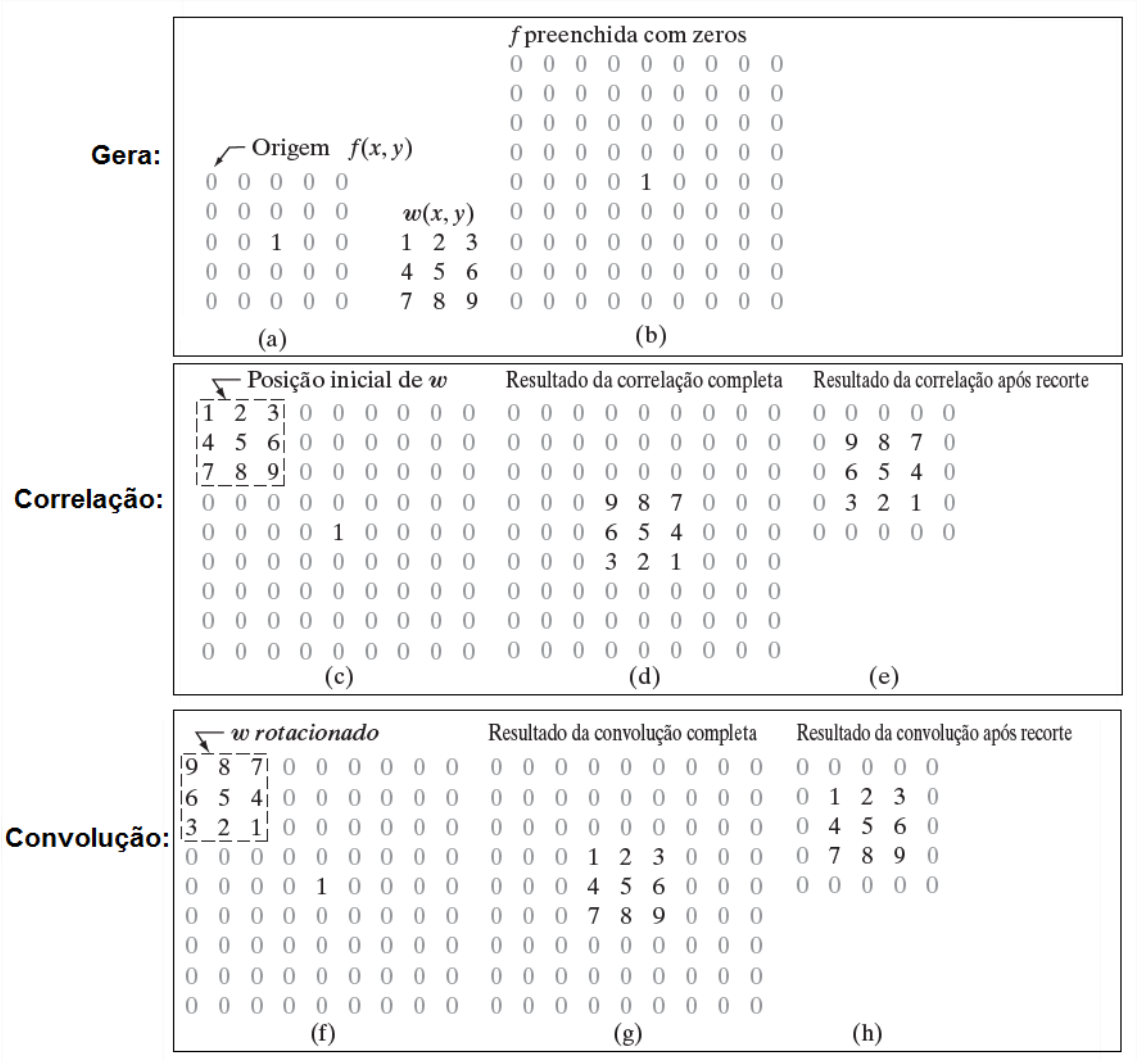
Figura 4: máscara de tamanho n

```
mask = 1/(n*n) * ones(n,n);
```

Fonte: Autoral

Desse modo, a Figura x1 mostra como funciona o processo de correlação e convolução entre uma máscara e uma matriz de impulso unitário.

Figura 5: correlação vs convolução




Fonte: Autoral, adaptado de (GONZALEZ, WOODS, 2010)

4) **Filtro edge(y,'canny'):** Função que retorna uma imagem binária contendo 1's onde a função encontra bordas, e 0's onde não encontra. Essa função utiliza o método de detecção de cantos de Sobel. Foi utilizada nessa etapa, para destacar as bordas das células, isso foi feito, por que as bordas estavam prejudicando a classificação do PCA, o qual buscava classificar as células pela borda (MATHWORKS, 2020).

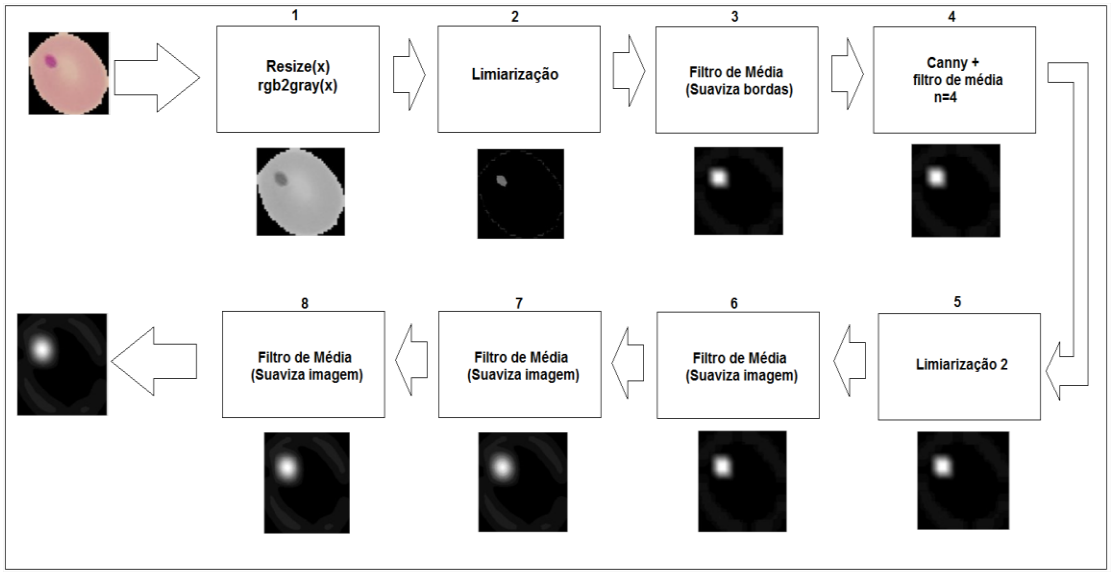
Da mesma forma, o parâmetro '**canny**', define um método específico de detecção de bordas, baseado nas derivadas de um filtro Gaussiano, isso destaca com mais fidelidade as bordas da imagem (MATHWORKS, 2020);

5) **Limiarização 2**: Limiarização realizada com o propósito de apagar as bordas geradas pelo canny. Foram atribuídos valores lógicos 0 nas posições da matriz y pela função 'y(logical(y1))=0';
6) **Filtro de média**: para além de suavizar a imagem, ele gera um efeito de 'engrossar' as bordas em volta do parasita;

7) **Filtro de média**: da mesma forma da anterior, essa chamada de função suaviza a imagem e destacar mais o parasita;
8) **Filtro de média:** Finaliza o PDI, deixando o parasita o mais destacado o possível para com essas funções.

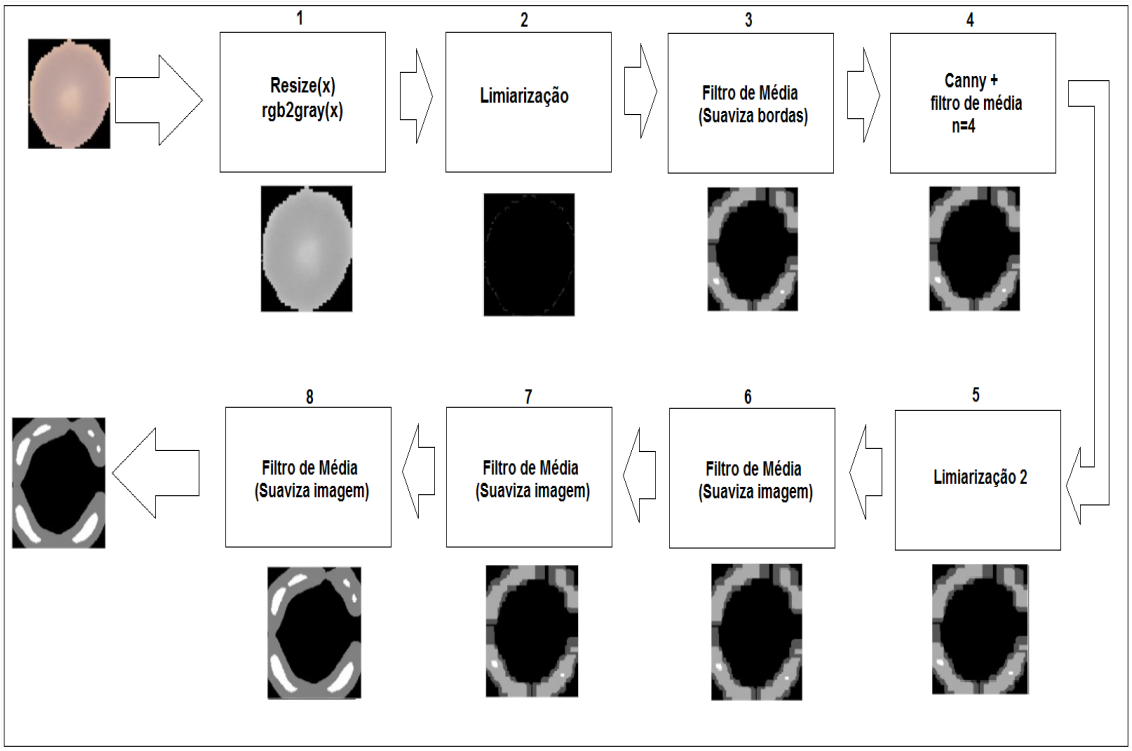
Tendo em vista, todos os passos supracitados, a Figura 8 e 9 mostram o processamento de duas imagens de exemplo, tendo como base o fluxograma exposto, a primeira é uma célula parasitada e a segunda é uma célula normal.

Figura 6: fluxograma com imagem parasitada




Fonte: Autoral

Figura 7: fluxograma com imagem não parasitada

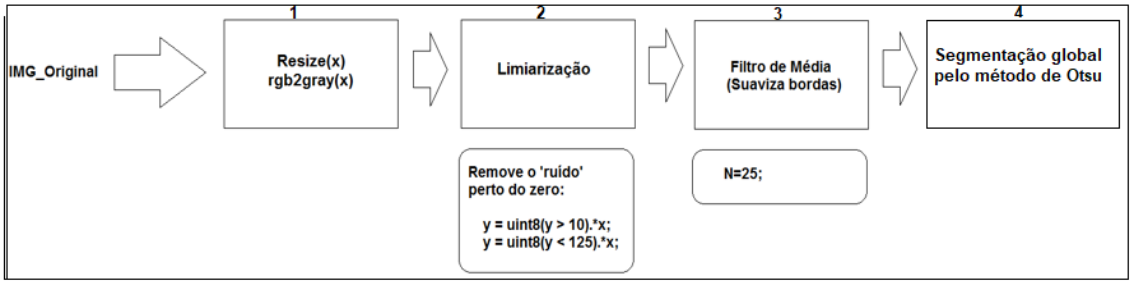



Fonte: Autoral

## 5.2. Proposta 2 – Segmentação com Otsu

A segunda proposta consistiu na separação do parasita para utilizá-lo como característica para classificação, para tanto foi empregada a limiarização global de Otsu. A Figura 8 apresenta o fluxograma do processamento das imagens.

Figura 8: fluxograma com imagem não parasitada




Fonte: Autoral

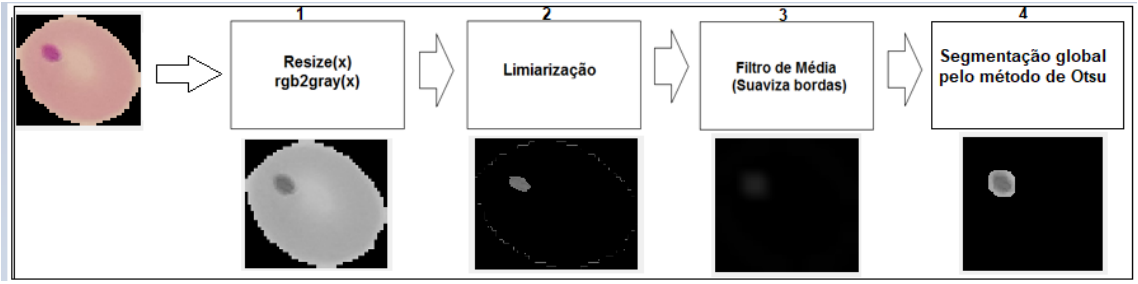
Como pode-se perceber os passos de 1 á 3 são os mesmos que da proposta 1, assim sendo, o passo 4 foi adicionado para segmentar o parasita. Para realizar essa segmentação foi utilizado o método de Otsu global

O método de Otsu é um método de limiarização ótima global. Ele busca maximizar a variância entre as classes, de forma que as classes com limiares bem estabelecidos devem ser distintas em relação aos valores de intensidade de seus pixels e, inversamente, que um limiar que oferece a melhor separação entre as classes em termos de valores de intensidade seria o melhor limiar (GONZALEZ; WOODS 2010).

O método de Otsu então, a partir do histograma da imagem tenta encontrar um limiar que separe a imagem em duas classes. O valor do limiar k é determinado de forma que a variância entre as classes seja máxima.

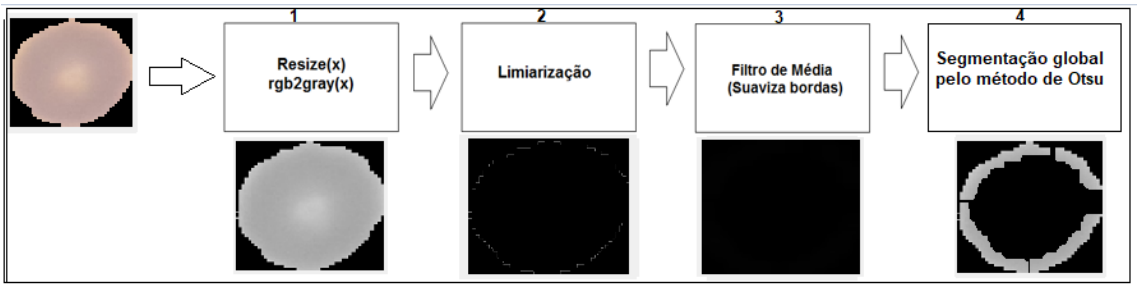
Tendo em vista, todos os passos supracitados, a Figura 9 e 10 mostram o processamento de duas imagens de exemplo, tendo como base o fluxograma exposto, a primeira é uma célula parasitada e a segunda é uma célula normal.

Figura 9: fluxograma com imagem parasitada




Fonte: Autoral

Figura 10: fluxograma com imagem não parasitada




Fonte: Autoral

A partir de uma inspeção visual das imagens resultantes do processo para a classe infectada (célula com parasita) apenas o parasita ficou visível, enquanto para a classe não infectada (célula que não apresenta o parasita) apenas as bordas ficaram visíveis. Desta forma, esse método possibilitou a aquisição de outras características da célula.

# 6. Resultados

## 6.1. Resultados somente com PCA

Para a aplicação do PCA as imagens foram redimensionadas para **escala 100 x 100** e convertidas para a escala de cinza. A Figura a seguir ilustra esse pré-processamento, em “a” é a imagem original, em “b” a imagem redimensionada para 100x100 e em “c” a imagem redimensionada convertida para a **escala de cinza**.

Figura 11 – Pré-processamento para a aplicação da PCA


Fonte: Autoral

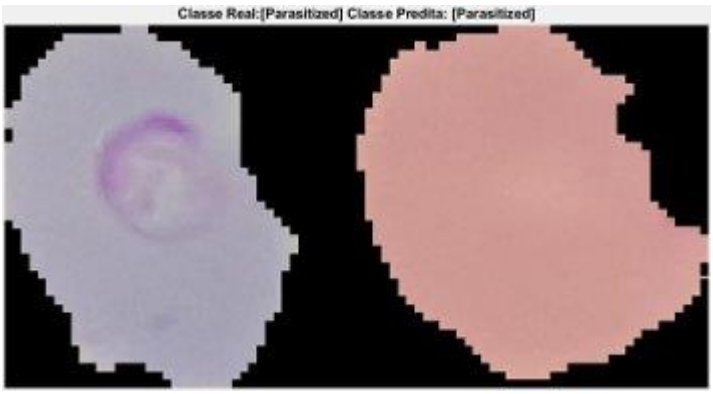
Esse pré-processamento foi necessário, pois, as imagens **não estavam nas mesmas dimensões** e para ser possível utilizar as funções do Matlab. Para comparação dos resultados os dados foram separados em treino e teste, nas seguintes **proporções 50%/50%, 60%/40% e 70%/30%** respectivamente. As seções seguintes trazem os resultados da aplicação do classificador nas proporções supracitadas.

### *6.1.1. Proporção 50%/50%*

Para a proporção 50% de dados para teste e 50% de dados para treino, um total de 880 amostras dos dados foram utilizadas para o teste. Foram obtidas **525 classificações corretas e 355 classificações incorretas**, a **acurácia geral de acerto foi de 59,66 %** e de **erro foi de 40,34%.** A **acurácia de acerto por classe foi de 56,59 % para a classe infectada e 62,73 % para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Porém ambas as médias foram bastante próximas.

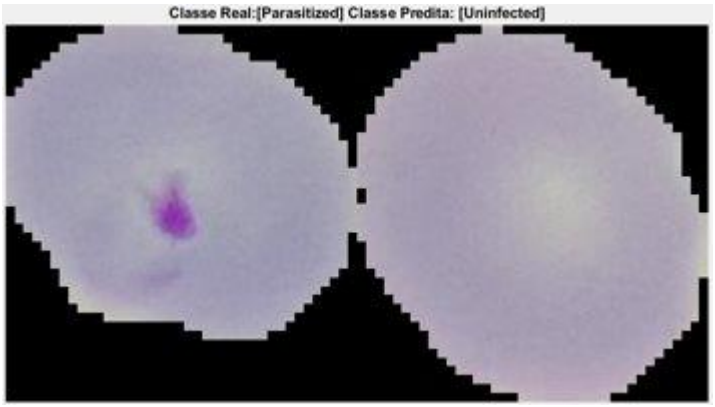
As **Figuras 12, 13 e 14** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes.

Figura 12 – Classificação correta



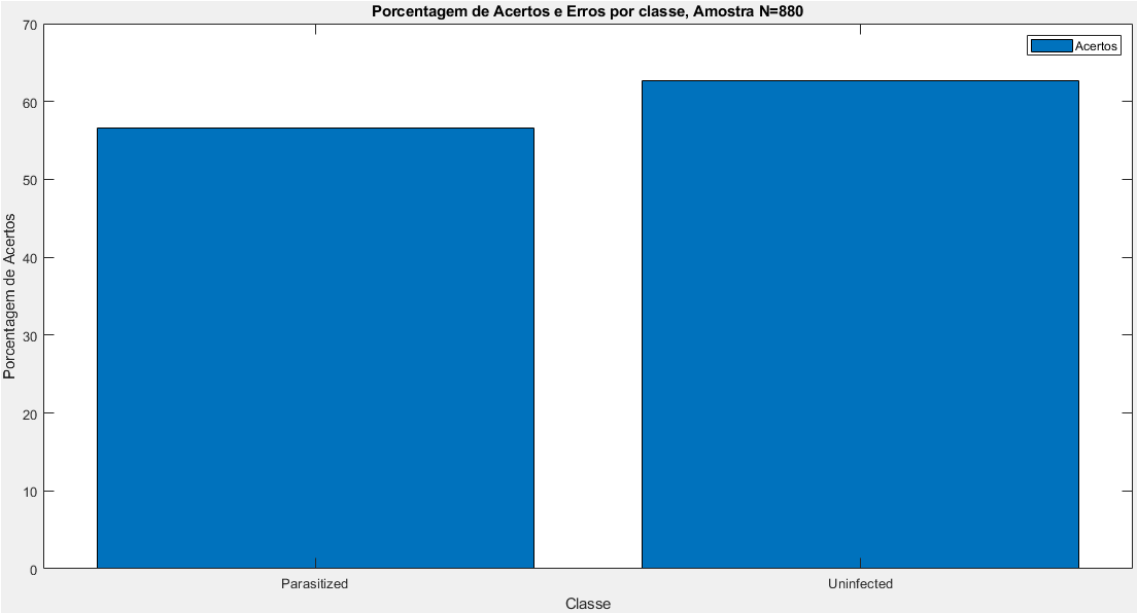

Fonte: Autoral

Figura 13 – Classificação incorreta




Fonte: Autoral

Figura 14 – Gráfico de barras da acurácia de acertos e erros por classe



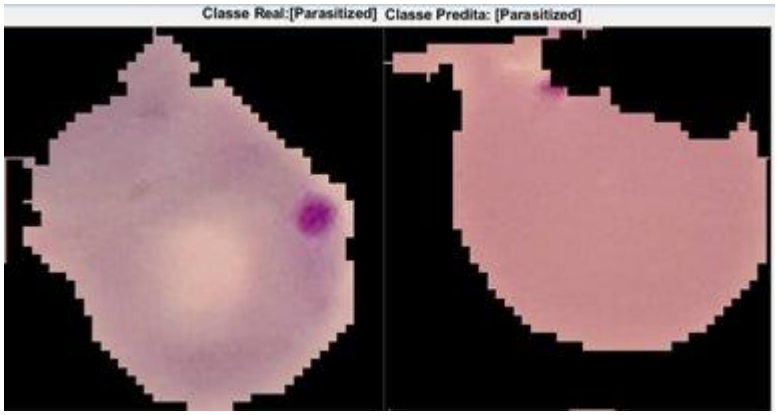

Fonte: Autoral

### *6.1.2. Proporção de 60%/40%*

Para a proporção 40% de dados para teste e 60% de dados para treino um total de 704 amostras dos dados foram utilizadas para o teste. Foram obtidas **443 classificações corretas e 261 classificações incorretas, a acurácia de acerto foi de 62,93% e de erro foi de 37,07%**. A **acurácia de acerto por classe foi de 63,35% para a classe infectada e 62,50 % para a classe não infectada.** Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Porém ambas as médias foram bastante próximas. Mas, em comparação com o resultado anterior percebe-se uma leve melhoria.

As **Figuras 15, 16 e 17** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes.

Figura 15 – Classificação correta

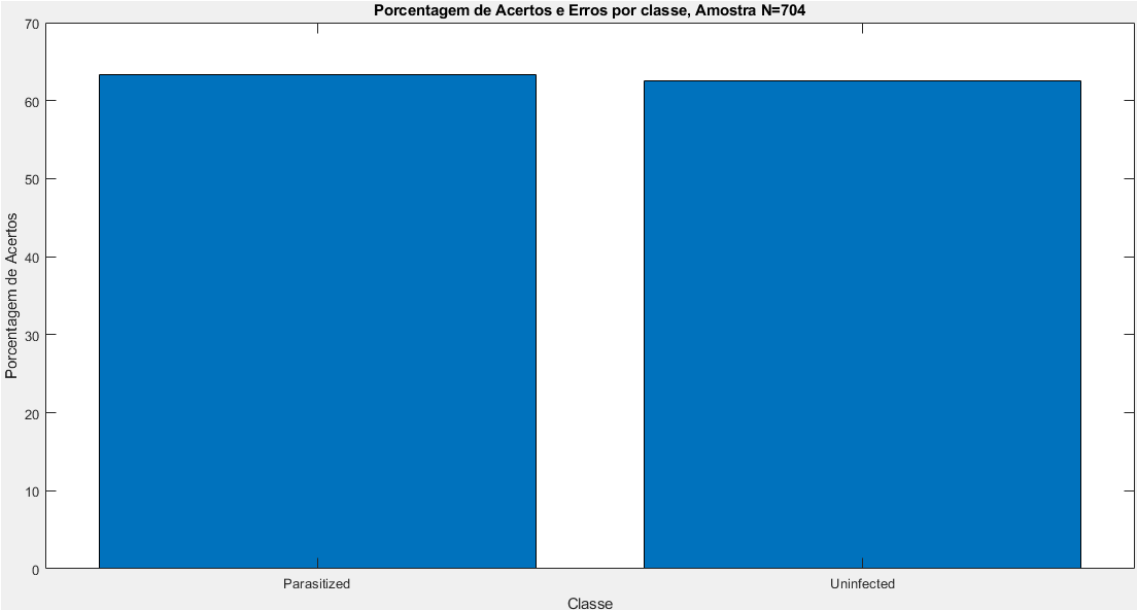



Fonte: Autoral

Figura 16 – Classificação incorreta


Fonte: Autoral

Figura 17 – Gráfico de barras da acurácia de acertos e erros por classe




Fonte: Autoral

### *6.1.3. Proporção 70%/30%*

Para a proporção 30% de dados para teste e 70% de dados para treino um total de 528 amostras dos dados foram utilizadas para o teste. Foram obtidas **316 classificações corretas e 212 classificações incorretas, a acurácia de acerto foi de 59,85% e de erro foi de 40,15%.** A **acurácia de acerto por classe foi de 64,02 % para a classe infectada e 55,68 % para a classe não infectada.** Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Porém ambas as médias foram bastante próximas. Em comparação com a configuração 50%/50% o resultado obtido foi muito próximo.

As **Figuras 18, 19 e 20** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes.

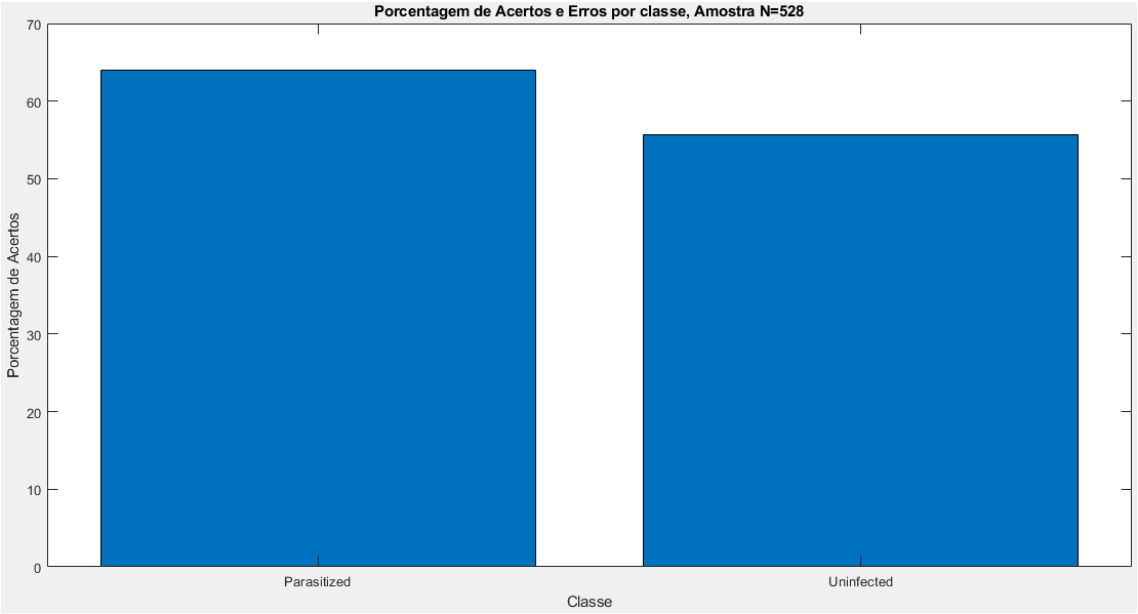
Figura 18 – Classificação correta


Fonte: Autoral

Figura 19 – Classificação incorreta


Fonte: Autoral

Figura 20 – Gráfico de barras da acurácia de acertos e erros por classe




Fonte: Autoral

### *6.1.4. Resumo dos resultados com as diferentes proporções de treino e teste*

A Tabela 1 apresenta o resumo dos erros e acertos do classificador para o banco de dados utilizando apenas o classificador PCA.

Tabela 1– Resumo de acertos e erros para o classificador usando apenas PCA

| **Proporção** | **50/50 %** | | **60/40%** | | **70/30%** | |
|---|---|---|---|---|---|---|
| **Acurácia** | Acurácia | Acertos | Acurácia | Acertos | Acurácia | Acertos |
| **Infectada** | 56,59 % | 249 | 63,35 % | 223 | 64,02 % | 169 |
| **Não infectada** | 62,73 % | 276 | 62,50 % | 220 | 55,68 % | 147 |
| **Geral** | 59,66 % | 525 | 62,93 % | 443 | 59,85 % | 316 |

Fonte: Autoral

A acurácia de acertos geral em relação as configurações 50/50 e 70/30 não variou significativamente. Um fato que pode corroborar para esse resultado, é a separação de dados em treino e teste que é feita de forma sequencial. A exploração da separação aleatória pode ser realizada, mas fugiria do escopo deste trabalho.

Outra questão sobre as imagens apresentadas na seção 5.1, é que, o classificador tende a classificar as imagens com base no seu formato. As figuras desta seção ilustram este fato. Espera-se que, a aplicação de técnicas de PDI possa melhorar esse resultado, já que, podem destacar outras características da imagem, como a presença ou não do parasita.

Em conclusão desta seção, o uso somente do PCA se mostrou eficiente para a classificação dos dados, porém apresenta uma taxa de acerto relativamente baixa.

## 6.2. Resultados somente com o PDI – Proposta 1

A presente seção expõe os dados de classificação com a PCA após o tratamento com o PDI explicado anteriormente no trabalho. Os dados são mostrados de acordo com as especificações do trabalho em três configurações diferentes 50/50, 60/40, 70/30.

### *6.2.1. Proporção 50%/50%*

Foram obtidas **751 classificações corretas e 129 classificações incorretas**, a **acurácia de acerto foi de 85,34 %** e de **erro foi de 14,66%.** A

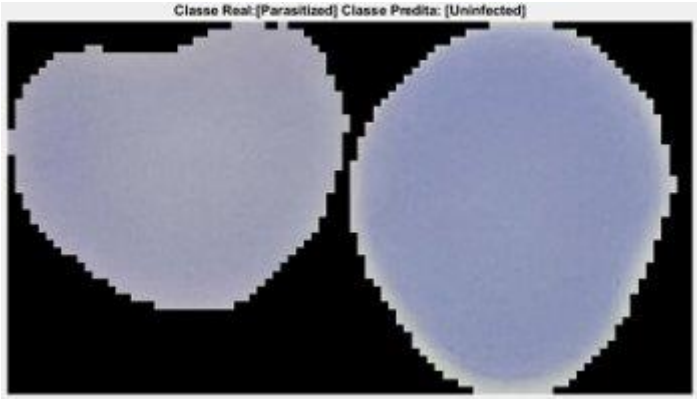
**acurácia de acerto por classe foi de 82,27% para a classe infectada e 88,41 % para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Percebe se que, a quantidade de acertos foi bem maior que a de erros.

As **Figuras 21, 22** e **23** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes respectivamente.

Figura 21 – Classificação correta

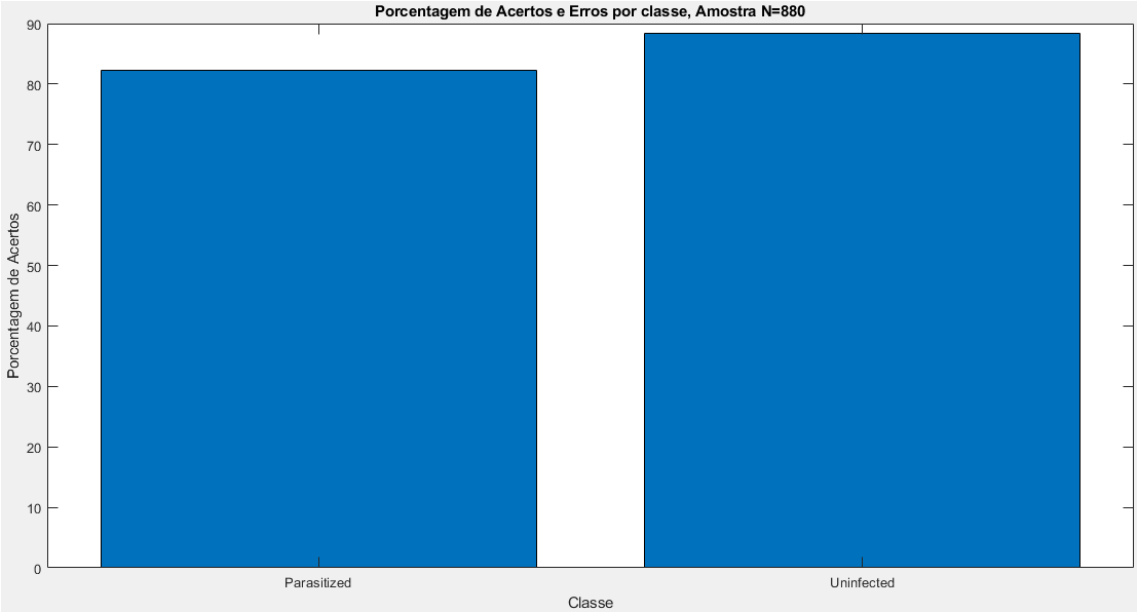

Fonte: Autoral

Figura 22 – Classificação incorreta




Fonte: Autoral

Figura 23 – Gráfico de barras da acurácia de acertos por classe 85.34



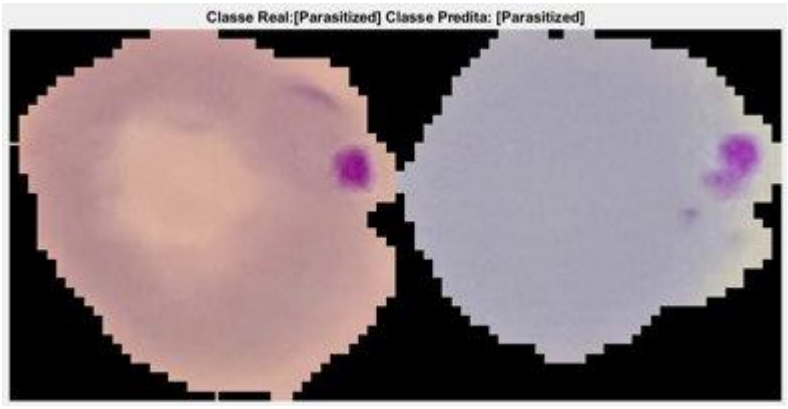

Fonte: Autoral

### *6.2.2. Proporção de 60%/40%*

Foram obtidas **599 classificações corretas e 105 classificações incorretas**, a acurácia de **acerto foi de 85,09%** e de **erro foi de 14,91%**. A **acurácia de acerto por classe foi de 83,24% para a classe infectada e 86,93 % para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Em comparação com o resultado anterior percebe-se uma pequena queda no número de acertos.

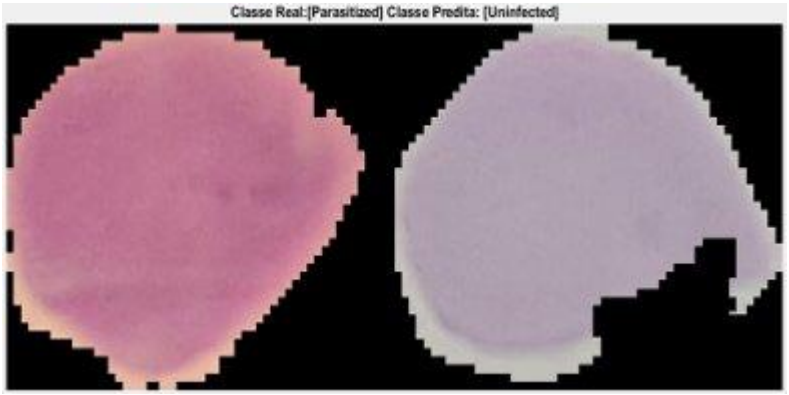
As **Figuras 24, 25 e 26** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes.

Figura 24 – Classificação correta



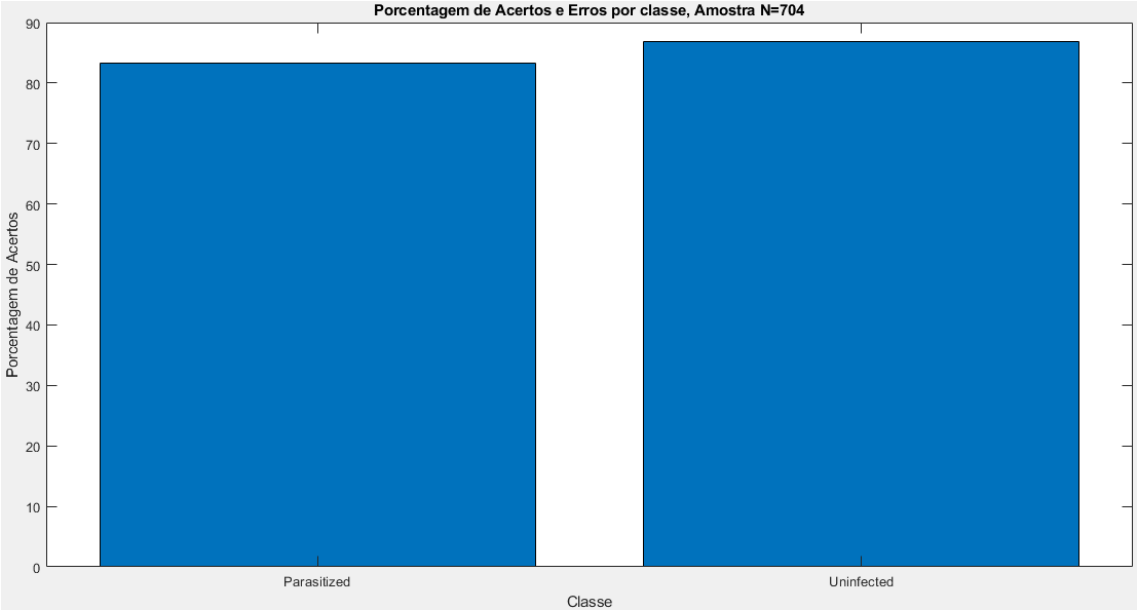

Fonte: Autoral

Figura 25 – Classificação incorreta




Fonte: Autoral

Figura 26 – Gráfico de barras da acurácia de acertos por classe 85.09



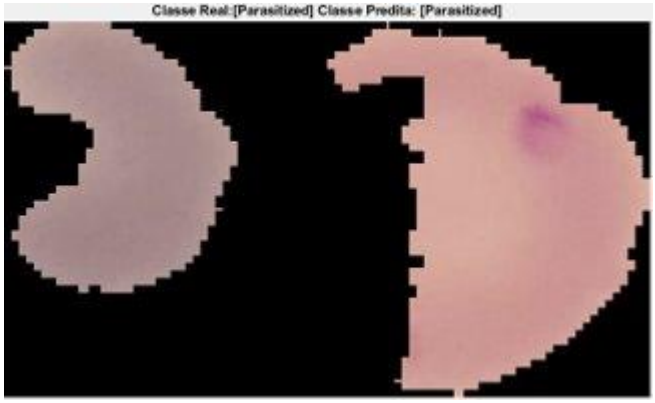

Fonte: Autoral

### *6.2.3. Proporção 70%/30%*

Foram obtidas **467 classificações corretas** e **61 classificações incorretas**, a acurácia de **acerto foi de 88.45%** e de **erro foi de 11,55 %**. A **acurácia de acerto por classe foi de 91,67% para a classe infectada e 85,23 % para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Em comparação com a configuração 50%/50% o resultado obtido apresentou um ganho significativo em termos de acertos de classificação.

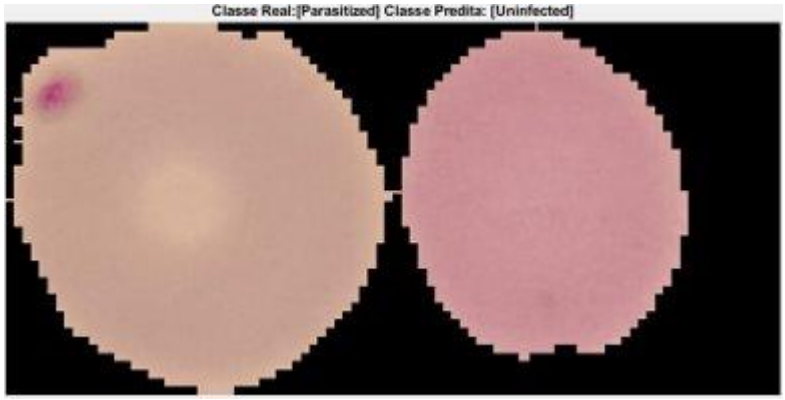
As **Figuras 27, 28 e 29** apresentam uma classificação correta, uma classificação incorreta e os gráficos dos resultados obtidos.

Figura 27 – Classificação correta



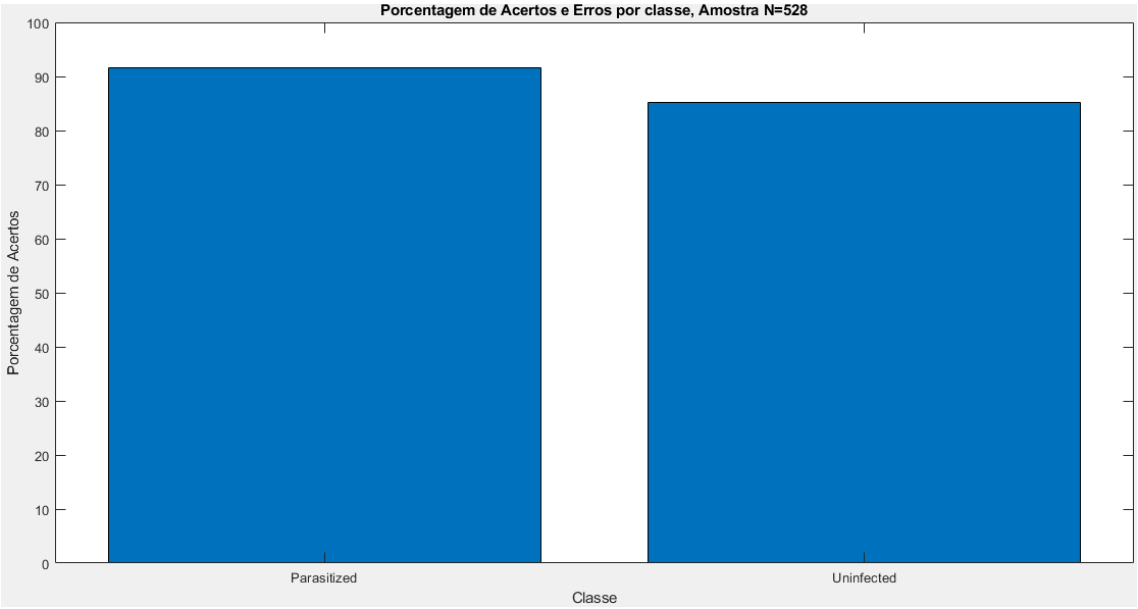

Fonte: Autoral

Figura 28 – Classificação incorreta




Fonte: Autoral

Figura 29 – Gráfico de barras da acurácia de acertos por classe 88.45%




Fonte: Autoral

### *6.2.4. Resumo dos resultados com as diferentes proporções de treino e teste*

A Tabela 2 apresenta o resumo da acurácia do classificador para o banco de dados utilizando apenas o classificador o método de pré-processamento da proposta 1

Tabela 2– Resumo de acertos e erros para o classificador usando apenas PCA

| **Proporção** | **50/50 %** | | **60/40%** | | **70/30%** | |
|---|---|---|---|---|---|---|
| **Acurácia** | Acurácia | Acertos | Acurácia | Acertos | Acurácia | Acertos |
| **Infectada** | 82,27 % | 362 | 83,24 % | 293 | 91,67 % | 242 |
| **Não infectada** | 88,41 % | 389 | 86,93 % | 306 | 85,23 % | 225 |
| **Geral** | 85,34 % | 751 | 85,09 % | 599 | 88,45 % | 467 |

Fonte: Autoral

A acurácia para as diferentes proporções ficou superior a 85 % que em comparação com a classificação apenas com o PCA melhorou significativamente. A melhor proporção foi proporção de 70 % dos dados para treino e 30 % dos dados para teste alcançando 88,45 %.

A aplicação do pré-processamento nas imagens aumentou a performance do classificador PCA melhorando os resultados obtidos. Além disso percebe-se a proposta não causou *overfitting* no modelo. Devido ao banco de dados das células infectadas conterem células que não apresentam o parasita, o modelo tendeu a classificá-las incorretamente como era esperado.

Portanto, percebe-se que o tratamento das imagens antes do classificador melhorou significativamente o resultado obtido. Outras formas de pré-processamento foram avaliadas, porém a que apresentou os melhores resultados foi a forma proposta nessa seção.

## 6.3. Resultados com o PDI – Proposta 2

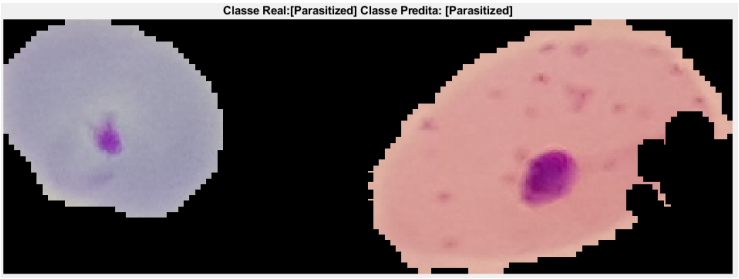
A presente seção expõe os dados de classificação com a PCA após o tratamento com o PDI explicado anteriormente no trabalho. Os dados são mostrados de acordo com as especificações do trabalho em três configurações diferentes 50/50, 60/40, 70/30.

### *6.3.1. Proporção 50%/50%*

Foram obtidas **737 classificações corretas e 143 classificações incorretas**, a **acurácia de acerto foi de 83,75 %** e de **erro foi de 16,25 %.** A **acurácia de acerto por classe foi de 85,00 % para a classe infectada e 82,50% para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Percebe se que, a quantidade de acertos foi bem maior que a de erros.

As **Figuras 30, 31 e 32** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes respectivamente.

Figura 30 – Classificação correta

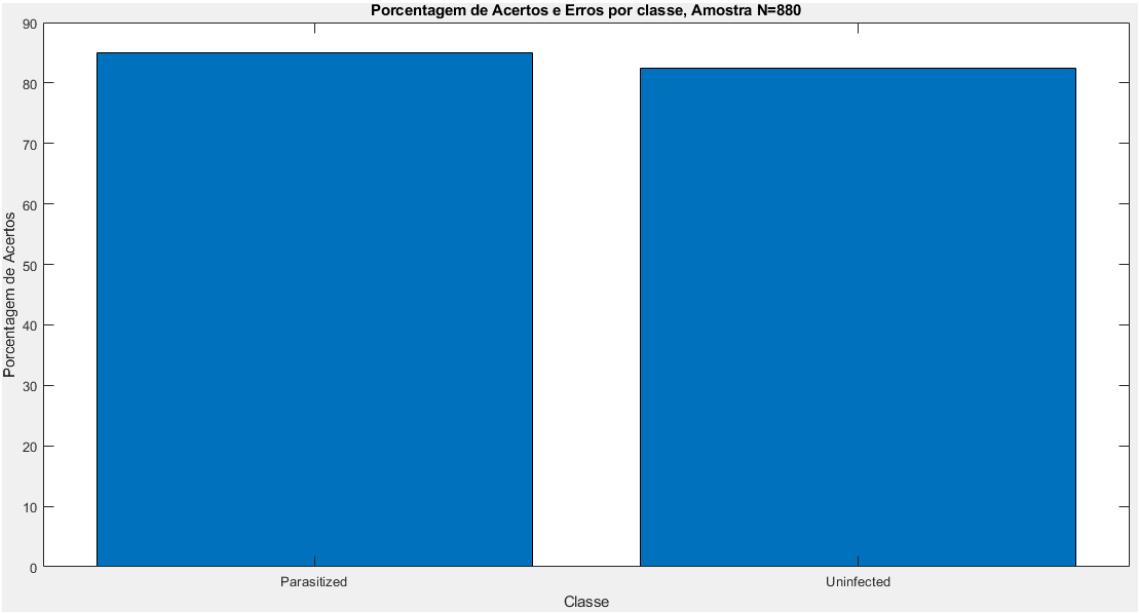



Fonte: Autoral

Figura 31 – Classificação incorreta


Fonte: Autoral

Figura 32 – Gráfico de barras da acurácia de acertos por classe 85.34



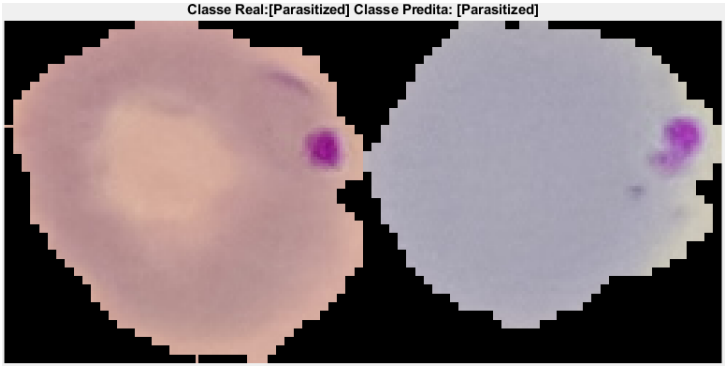

Fonte: Autoral

### *6.3.2. Proporção de 60%/40%*

Foram obtidas **585 classificações corretas e 119 classificações incorretas**, a acurácia de **acerto foi de 85,09%** e de **erro foi de 14,91%**. A **acurácia de acerto por classe foi de 84,38 % para a classe infectada e 81,82% para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente. Em comparação com o resultado anterior percebe-se uma pequena queda no número de acertos.

As **Figuras 33, 34 e 35** apresentam uma classificação correta, uma classificação incorreta e o gráfico de barras para a acurácia das classes.

Figura 33 – Classificação correta

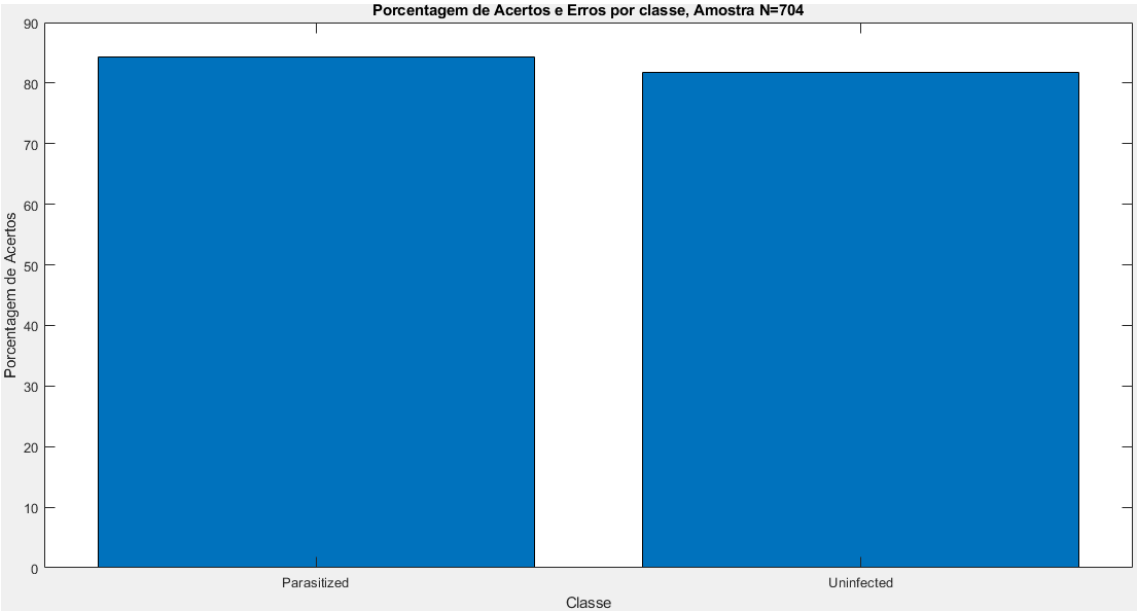



Fonte: Autoral

Figura 34 – Classificação incorreta


Fonte: Autoral

Figura 35 – Gráfico de barras da acurácia de acertos por classe 85.09



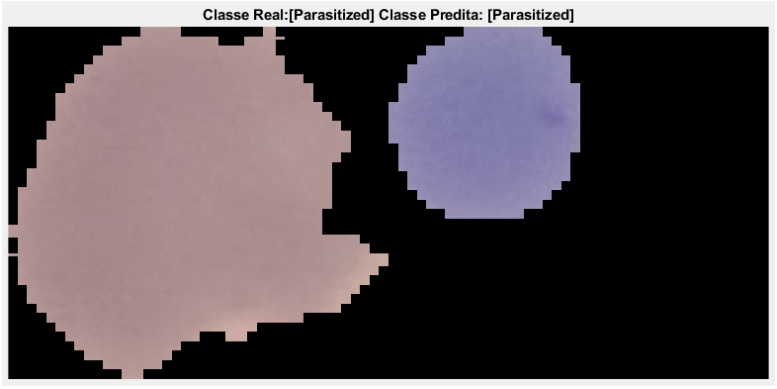

Fonte: Autoral

### *6.3.3. Proporção 70%/30%*

Foram obtidas **443 classificações corretas** e **81 classificações incorretas**, a acurácia de **acerto foi de 83,90%** e de **erro foi de 16,10 %**. %. A **acurácia de acerto por classe foi de 89,39 % para a classe infectada e 78,41 % para a classe não infectada**. Nessa proporção o classificador conseguiu classificar corretamente mais imagens do que classificar incorretamente.

As **Figuras 36, 37 e 38** apresentam uma classificação correta, uma classificação incorreta e os gráficos dos resultados obtidos.

Figura 36 – Classificação correta

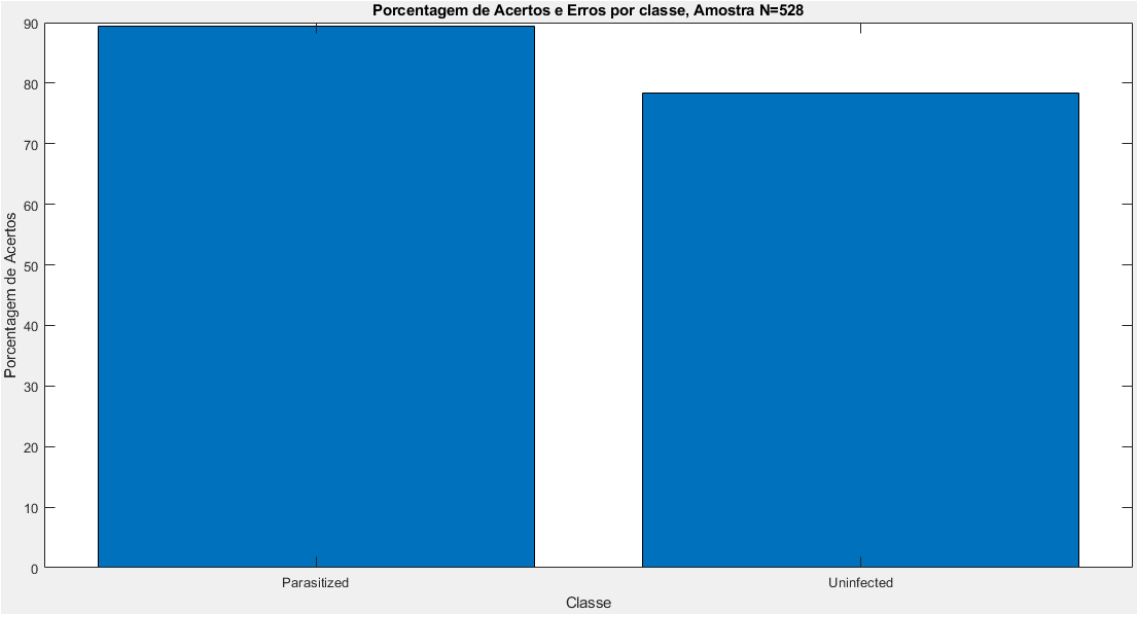



Fonte: Autoral

Figura 37 – Classificação incorreta


Fonte: Autoral

Figura 38 – Gráfico de barras da acurácia de acertos por classe 88.45%




Fonte: Autoral

### *6.3.4. Resumo dos resultados com as diferentes proporções de treino e teste*

A Tabela 2 apresenta o resumo da acurácia do classificador para o banco de dados utilizando apenas o classificador o método de pré-processamento da proposta 1

Tabela 2– Resumo de acertos e erros para o classificador usando apenas PCA

| **Proporção** | **50/50 %** | | **60/40%** | | **70/30%** | |
|---|---|---|---|---|---|---|
| **Acurácia** | Acurácia | Acertos | Acurácia | Acertos | Acurácia | Acertos |
| **Infectada** | 85,00 % | 374 | 84,38 % | 297 | 89,39 % | 236 |
| **Não infectada** | 82,50 % | 363 | 81,82 % | 288 | 78,41 % | 207 |
| **Geral** | 83,75 % | 737 | 83,10 % | 585 | 83,90 % | 443 |

Fonte: Autoral

A acurácia para as diferentes proporções ficou superior a 83 % que em comparação com a classificação apenas com o PCA melhorou significativamente. A melhor proporção foi proporção de 70 % dos dados para treino e 30 % dos dados para teste alcançando 83,90 %.

A aplicação do pré-processamento nas imagens aumentou a performance do classificador PCA melhorando os resultados obtidos. Além disso percebe-se a proposta não causou *overfitting* no modelo. Devido ao banco de dados das células infectadas conterem células que não apresentam o parasita, o modelo tendeu a classificá-las incorretamente como era esperado.

Portanto, percebe-se que o tratamento das imagens antes do classificador melhorou significativamente o resultado obtido. Em comparação com a proposta 1, a proposta 2 obteve um resultado levemente pior. Sendo assim, a proposta 2 foi escolhida como a melhor proposta para a classificação dos dados.

## 7. Conclusão

A classificação manual de malária é uma tarefa lenta e exige concentração por parte do especialista que está realizando o processo. Além disso é muito suscetível a erros, já que a contagem manual exige atenção. A classificação automática é um procedimento que pode ser empregado para diminuir a carga de trabalho dos especialistas e consequentemente gerar resultados mais fidedignos.

O presente trabalho buscou classificar células contaminadas por malária utilizando o classificador PCA em conjunto com técnicas de processamento digital de imagens. Os dados foram avaliados com PCA puro e com o PCA com duas propostas de pré-processamento. Além disso os dados foram separados em três proporções de treino e teste sendo elas, proporção 50/50, 60/40 e 70/30.

O PCA puro apresentou desempenho regular classificando corretamente aproximadamente 60 % das amostras de teste nas três proporções de dados. Já o PCA com a proposta 1 apresentou o melhor desempenho classificando mais de 85 % das amostras de teste corretamente nas três proporções de dados e na proporção 70/30 a classificação foi a mais bem obtida neste trabalho, ficando em 88,45 %. Por fim a proposta 2 classificou corretamente aproximadamente 83 % das amostras nas três proporções de dados. Desta forma, a melhor proposta é a proposta 1.

Por fim, a limitação do trabalho se encontra no uso de um banco de dados subamostrado. Mas, espera-se que ao aplicar a classificação com o banco de dados completo os resultados sejam semelhantes já que o banco foi subamostrado de forma a refletir o banco de dados completo.

## 8. Referências


ALMEIDA, Jaqueline. **Aula de Parasitologia Médica sobre a Malária**. Disponível em: https://www.slideshare.net/JaquelineAlmeida26/aula-de-parasitologia-mdica-sobre-a-malria. Acesso em: 09 jun. 2020.

GONZALEZ, Rafael C.; WOOD, Richard E.. **Processamento Digital de Imagens**. 3. ed. São Paulo: Pearson Pratice Hall, 2010.

JAEGER. **Malaria Datasets**. Disponível em: https://lhncbc.nlm.nih.gov/publication/pub9932. Acesso em: 08 jun. 2020.

Kunwar, Suman et al. “Malaria Detection Using Image Processing and Machine Learning.” *ArXiv* abs/1801.10031 (2018): n. pag.

MATHWORKS. **Edge**. Disponível em: https://www.mathworks.com/help/images/ref/edge.html. Acesso em: 12 jun. 2020.

Rahman, H. Zunair, M. S. Rahman, J. Q. Yuki, S. Biswas, M. A. Alam, et al., Improving malaria parasite detection from red blood cell using deep convolutional neural networks, 2019.

SANTO, Rafael do Espírito. Principal Component Analysis applied to digital image compression. **Einstein (são Paulo)**, [s.l.], v. 10, n. 2, p. 135-139, jun. 2012. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s1679-45082012000200004.

## Apêndice A – Função Classificar.m

```
%Classificador: Distância Euclidiana
%Entrada:
%   PC = Conjunto de treinamento
%   x = amostra a ser comparada
%Saída
%d = posição do item com a menor distância euclidiana entre a
amostra e as
%colunas do conjunto de treinamento
function d = Classificar(PC,x)
    minDist = norm(PC(:,1) - x);
    d = 1;
    for j = 2:size(PC)
        EucDist = norm(PC(:,j) - x);
        if minDist > EucDist
            d = j;
            minDist = EucDist;
        end
    end
end
```

## Apêndice B – Função GerarPCs.m

```
%Gera a matriz de componentes principais
%Entrada:
%   data = matriz de dados. Cada amostra de medições é representada
em uma
%coluna
%saída:
%   P = dados no novo espaço vetorial (autofaces no caso de imagens)
%   PC = matriz componentes principais
%   mn = média de cada coluna da matriz de dados
function [P PC mn] = GerarPCs(data)
    [M,N] = size(data);
    mn = mean(data,2); %calcular a média
    data = double(data) - repmat(mn,1,N); %imagens - media
    covariancia = data' * data; %matriz de covariância
    [PC,V] = eig(covariancia); %autovetores e autovalores

    V = diag(V);

    clear covariancia, clear M, clear N;

    %ordenar autovetores e autovalores

    [junk,rindices] = sort(-1*V);
    V = V(rindices);
    V = diag(V);
    PC = PC(:,rindices);

    clear junk , clear rindices, clear V;

    %cálculo das autofaces

    P = data * PC;
    i = size(PC,2);

    clear PC;

    %projeção das imagens no espaço de faces
    PC = [];
    for j = 1:i
        temp = P' * data(:,j);
        PC = [PC temp];
    end

    clear i, clear j, clear temp;
end
```

## Apêndice C – Função lerImgs.m

```
function [label_treino, label_teste, dados_treino, dados_teste ....
    img_treino, img_teste, qtd_classes] = lerImgs(porcentagem, path,
extensao, tipo)
    % Parametros
    %   porcentagem -> Porcentagem de dados que serão da classe de
treino
    %   path -> Caminho da pasta que contém as subpastas que são as
classes
    %   extensao -> Extensão das imagens
    %   Tipo -> Indica o tipo de operação que será empregada nas
imagens
    % Retornos
    %   label_treino -> Vetor auxiliar com as labels (classes) da
parte da
    %   parte do banco que será usada como treino
    %   label_teste -> Vetor auxiliar com as labels (classes) da
parte da
    %   parte do banco que será usada como teste
    %   dados_treino -> Dados que serão usados como treino da PCA
    %   dados_teste -> Dados que serão usados como teste da PCA
    %   qtd_classes -> Quantidade de classes do banco de dados

    dados_treino = [];
    dados_teste = [];
    label_treino = [];
    label_teste = [];
    img_treino = [];
    img_teste = [];

    % Acessa as pastas
    S = dir(fullfile(path));

    % Percore cada pasta que contém as classes
    qtd_classes  = numel(S) - 2;
    for pasta = 3:numel(S)
        % Concatena as strings para formar o nome de cada pasta
        pasta_classe = S(pasta).name;
        caminho = strcat(S(pasta).folder,'\', pasta_classe,'\');
        classe = dir(fullfile(caminho,strcat('*',extensao)));

        % Quantidade treino e teste
        qtd_dados = (size(classe, 1));

        % Separa a porcentagem de treino
        qtd_treino = round((qtd_dados * porcentagem) / 100);
        i = 1;
        for i = 1: qtd_treino


            w = fullfile(caminho,classe(i).name);
            x = imread(w);

            if (tipo == "SIMPLES")
                y = SIMPLES(x);
            end
```

```
            if (tipo == "PDI")
                y = PDI(x);
            end
                        dados_treino = [dados_treino, y];
            label_treino = [label_treino, string(pasta_classe)];
            img_treino = [img_treino, string(classe(i).name)];
        end
        % Separa a porcentagem de teste
        for j = i+1: qtd_dados
            w = fullfile(caminho,classe(j).name);
            x = imread(w);

            if (tipo == "PDI")
                y = PDI(x);
            end

            dados_teste = [dados_teste, y];
            label_teste = [label_teste, string(pasta_classe)];
            img_teste = [img_teste, string(classe(j).name)];
        end
    end
end

function [y] =  SIMPLES(x)
    x = rgb2gray(x);
    x = imresize(x, [100 100]);
    y = (reshape(x,[size(x,1)*size(x,2),1]));
end
```

## Apêndice D - main.m

```
clear all;
close all;
clc;
%% --
% Parametros
% porcentagem -> Porcentagem de dados que serão da classe de treino
% path -> Caminho da pasta que contém as subpastas que são as
classes
% extensao -> Extensão das imagens
% tipo, tipo de algoritmos para extração de características
% tipo = "SIMPLES" -> Emprega apenas o PCA;
% tipo = "PDI" -> Aplica a técnica de pré-processamento no conjunto
de dados
% para a proposta de técnica de PDI 2

porcentagem = 70;
path = 'D:\Estudos\Eng. de Computação\Processamento Digital de
Imagens\PCA e Classificação\Imagens_Separadas';
extensao = '.png';
tipo = 'SIMPLES';

[label_treino, label_teste, dados_treino, dados_teste ....
    img_treino, img_teste, qtd_classes, P, PC, mn, img_e, img_a]
....
= funcao(path ,porcentagem, extensao , tipo);

%%
% %% --
% Realiza testes 'online'
 i = 1;
while(i)
    disp('Informe se a imagem está no conjunto de treino ou de
teste');
    temp = input('[1] Para treino\n[2] Para teste\nQualquer outra
tecla para sair\nOpção: ');

    if temp == 1
        n = size(img_treino);
    else
        if temp == 2
            n = size(img_teste);
        else
            disp('Saindo...');
            break;
        end
    end

    % Garante a opção escolhida
    if temp == 1 || temp == 2
        % Pega o valor no indice do vetor para capturar a imagem
        val = input(strcat('Informe um valor de [', num2str(n),']:
'));

        % Determina a classe e pega a imagem
        if temp == 1
```

```
            classe = label_treino(val);
            image = img_treino(val);
        else
            classe = label_teste(val);
            image = img_teste(val);
        end

        % Cria o caminho da imagem
        caminho = strcat(path,'\', classe, '\', image);
        caminho = caminho{1};

        x = imread(caminho);

        x = PDI(x);

        d = Classificar(PC, ProjetarAmostra(x,mn,P));


        % Cria o caminho da imagem correspondente no conjunto de
treino
        caminho_treino = strcat(path,'\', label_treino(d), '\',
img_treino(d));
        caminho_treino = caminho_treino{1};

        figure;

        imshowpair(imread(caminho),
imread(caminho_treino),'montage');
        title(strcat('Classe Real: ','[',classe,']', ....
                    ' Classe Predita: [',label_treino(d), ']'));

        if isequal(label_treino(d), classe)
           disp('Acertou Miseravel');
           disp(' ');
        else
           disp('Errou');
           disp(' ');
        end
        clear x;
        clear d;
    end

end

function [label_treino, label_teste, dados_treino, dados_teste ....
   img_treino, img_teste, qtd_classes, P, PC, mn, img_e, img_a] ....
   = funcao(path, porcentagem, extensao, tipo)
    % Parametros
    %   porcentagem -> Porcentagem de dados que serão da classe de
treino
    %   path -> Caminho da pasta que contém as subpastas que são as
classes
    %   extensao -> Extensão das imagens
    % Retornos
    %   label_treino -> Vetor auxiliar com as labels (classes) da
parte da
    %   parte do banco que será usada como treino
```

```
    %   label_teste -> Vetor auxiliar com as labels (classes) da
parte da
    %   parte do banco que será usada como teste
    %   dados_treino -> Dados que serão usados como treino da PCA
    %   dados_teste -> Dados que serão usados como teste da PCA
    %   qtd_classes -> Quantidade de classes do banco de dados

    % Para mostrar uma imagem certa e uma imagem errada
    aux = 0;
    aux1 = 0;

    img_a = [];
    img_e = [];

    [label_treino, label_teste, dados_treino, dados_teste ....
        img_treino, img_teste, qtd_classes] = lerImgs(porcentagem,
path, extensao, tipo);

    %% --
    % Qunatidade de amostras de teste
    tam = size(dados_teste, 2)
    qtd_dados = tam;

    % Gera a PCA
    [P PC mn] = GerarPCs(dados_treino);

    % Vetores que contém a quantidade de acertos e erros por cada
classe
    acertos = zeros(qtd_classes, 1);
    erros = zeros(qtd_classes, 1);

    % Converte as categorias para números para ser possível contar a
    % quantidade de erros e acertos

    %%--
    c = grp2idx(categorical(label_treino));

    for i = 1: tam

        % Gera a amostra
        amostra = ProjetarAmostra(dados_teste(:, i),mn,P);
        d = Classificar(PC,amostra);

        indice_amostra = c(d);

        if isequal(label_teste(i), label_treino(d))

            acertos(indice_amostra) = acertos(indice_amostra) + 1;
            if(aux == 0)
               figure;
               c1 =
strcat(path,'\',label_teste(i),'\',img_teste(i));
               c2 = strcat(path,'\',label_treino(d),
'\',img_treino(d));

               real = imread(c1{1});
               pred = imread(c2{1});
               imshowpair(real, pred, 'montage');
```

```
                title( strcat('Classe Real: ','[',label_teste(i),']',
....
                        ' Classe Predita: [',label_treino(d), ']'));

                aux = aux + 1;
            end
        else
            erros(indice_amostra) = erros(indice_amostra) + 1;
            c1 = strcat(path,'\',label_teste(i),'\',img_teste(i));
            c2 = strcat(path,'\',label_treino(d),
'\',img_treino(d));
            if(aux1 == 0)
                f = figure;

                real = imread(c1{1});
                pred = imread(c2{1});
                imshowpair(real, pred, 'montage');
                title( strcat('Classe Real: ','[',label_teste(i),']',
....
                        ' Classe Predita: [',label_treino(d), ']'));
                aux1 = aux1 + 1;
            end
            c1 = strcat('\',label_teste(i),'\',img_teste(i));
            c2 = strcat('\',label_treino(d), '\',img_treino(d));
            img_e = [img_e, c1];
            img_a = [img_a, c2];
        end
        clear d;
    end

    qtd_acerto = sum(acertos);
    qtd_erro = sum(erros);

    media_acerto = (100 * qtd_acerto) / qtd_dados;
    media_erro = (100 * qtd_erro) / qtd_dados;

    % Converte as labels para categorias
    labels = categorical(unique(label_treino));

    % ---- Plots -----
    % Grafico de barras de porcentagem de acertos e erros
    figure;

    bar(labels, ([acertos]./(tam/2))*100);
    title(strcat('Porcentagem de Acertos e Erros por classe, Amostra
N=',num2str(tam)));
    xlabel('Classe');
    ylabel('Porcentagem de Acertos');

    % Grafico de piza de porcentagem da amostra de acertos e erros
    figure;
    labels = {'Acertos', 'Erros'};
    pie([qtd_acerto, qtd_erro]);
    title(strcat('Proporção acertos e erros, Amostra
N=',num2str(tam)));
    legend(labels);

    % Quantidade de acerto e erro geral e porcentagem de erro e
acerto geral
```

```
    fprintf('\nQuantidade de Acertos: %i\nPorcentagem de Acertos:
%.2f\n', qtd_acerto, media_acerto);
    fprintf('\nQuantidade de Erros: %i\nPorcentagem de Erros:
%.2f\n', qtd_erro, media_erro);

    % Quantidade de acerto e erro por classe
    % Infectada
    fprintf('\n --- Classe: Infectada ---');
    fprintf('\nQuantidade de Acertos: %i\nPorcentagem de Acertos:
%.2f\n', acertos(1), acertos(1)/(tam/2)*100);

    fprintf('\n --- Classe: Não Infectada ---');
    fprintf('\nQuantidade de Acertos: %i\nPorcentagem de Acertos:
%.2f\n\n', acertos(2), acertos(2)/(tam/2)*100);
end
```

## Apêndice E - PDI.m (Proposta 1)

```
function y = PDI(x)

    % ------------------------------------------------------------
-------
    % Passo 1 - Redimensionamento e conversão para escala de cinza
    x = imresize(x, [150 150]);
    x = rgb2gray(x);
    y = x;
    % ------------------------------------------------------------
-------

    % ------------------------------------------------------------
-------
    % Passo 2 - Limiarização
    y = uint8(y > 10).*x;
    y = uint8(y < 125).*x;
    % ------------------------------------------------------------
-------



    % ------------------------------------------------------------
-------
    % Passo 3 - Filtragem
    n = 25;
    mask = 1/(n*n) * ones(n,n);
    y = imfilter(y, mask);
    y = round(y/1.3);

    % y = otsu(y) .* x; % 70/30 = 84% de acerto
    % ------------------------------------------------------------
-------

    % ------------------------------------------------------------
-------
    % Passo 4 - Filtro Edge + Canny
    y1 = uint8(edge(y,'canny'));
    n = 4;
    mask = 1/(n*n) * ones(n,n);
    y1 = imfilter(y1, mask);
    y(logical(y1)) = 0;
    % ------------------------------------------------------------
-------

    % ------------------------------------------------------------
-------
    % Passo 5 - Filtragem
    n = 25;
    mask = 1/(n*n) * ones(n,n);
    y = imfilter(y, mask);
    y = round(y/1.3);
    % ------------------------------------------------------------
-------

    % ------------------------------------------------------------
-------
```

```
      % Passo 6 - Filtragem
      n = 5;
      mask = 1/(n*n) * ones(n,n);
      y = imfilter(y, mask);
      % ------------------------------------------------------------
-------

      % ------------------------------------------------------------
-------
      % Passo 7 - Filtragem
      n = 2;
      mask = 1/(n*n) * ones(n,n);
      y = imfilter(y, mask);
      % ------------------------------------------------------------
-------

      % ------------------------------------------------------------
-------
      % Passo 8 - Imagem resultante
        y =  SIMPLES(y);
%       % ------------------------------------------------------------
---------
end

function [y] =  SIMPLES(x)
    y = (reshape(x,[size(x,1)*size(x,2),1]));
end
```

## Apêndice F - PDI.m (Proposta 2)

```
function y = PDI(x)

     % ------------------------------------------------------------
-------
     % Passo 1 - Redimensionamento e conversão para escala de cinza
     x = imresize(x, [150 150]);
     x = rgb2gray(x);
     y = x;
     % ------------------------------------------------------------
-------

     % ------------------------------------------------------------
-------
     % Passo 2 - Limiarização
     y = uint8(y > 10).*x;
     y = uint8(y < 125).*x;
     % ------------------------------------------------------------
-------



     % ------------------------------------------------------------
-------
     % Passo 3 - Filtragem
     n = 25;
     mask = 1/(n*n) * ones(n,n);
     y = imfilter(y, mask);
     y = round(y/1.3);

     y = otsu(y) .* x; % 70/30 = 84% de acerto
     % ------------------------------------------------------------
-------
     y = simples(y);

end

function [y] =  SIMPLES(x)
    y = (reshape(x,[size(x,1)*size(x,2),1]));
end
```

## Apêndice G – ProjetaAmostra.m

```
%Projeta uma nova amostra no espaço vetorial P
%Entrada:
%   x = amostra a ser projetada
%   mn = média de cada coluna da matriz de dados
%   P = dados no novo espaço vetorial (autofaces no caso de imagens)
%Saída:
%   x = amostra no novo espaço vetorial
function x = ProjetarAmostra(x,mn,P)
    %x = reshape(x,[size(x,1)*size(x,2),1]);
    x = reshape(x,[size(x,1)*size(x,2),1]);
    x = double(x) - mn;
    x = P' * x;
end
```