CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

NATHAN MESSIAS FIUSA MENÊSES
LEONARDO ANTONIO DE DEUS

CLASSIFICAÇÃO DE IMAGENS EMPREGANDO CLASSIFICADOR ESTATÍSTICO PARA CONTAGEM DE COLÔNIAS DE BACTÉRIAS

ANÁPOLIS
2020

NATHAN MESSIAS FIUSA MENÊSES
LEONARDO ANTONIO DE DEUS

CLASSIFICAÇÃO DE IMAGENS EMPREGANDO CLASSIFICADOR ESTATÍSTICO PARA CONTAGEM DE COLÔNIAS DE BACTÉRIAS

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador: Prof. Dr. Clarimar José Coelho

Anápolis
2020

NATHAN MESSIAS FIUSA MENÊSES
LEONARDO ANTONIO DE DEUS

CLASSIFICAÇÃO DE IMAGENS EMPREGANDO CLASSIFICADOR ESTATÍSTICO PARA CONTAGEM DE COLÔNIAS DE BACTÉRIAS

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em [dia] de [mês] de 2020, composta por:

_____
Prof. Clarimar José Coelho
Orientador

_____
Prof. [nome do professor]

_____
Prof. [nome do professor]

Agradecimentos

Agradecemos em primeiro lugar a Deus, pelo dom da vida, saúde, inteligência e por nos fortalecer espiritualmente nos momentos mais difíceis dessa caminhada.

Ao Prof. Dr. Clarimar José Coelho, por todos os ensinamentos prestados, orientações, pela paciência que teve, e principalmente pela amizade obtida durante o período de convívio.

Aos nossos familiares, pelo apoio emocional, companheirismo e por acreditar que estaríamos finalizando esse curso.

Resumo

A contagem manual de colônia de bactérias é um trabalho difícil, já que o mesmo envolve processos ainda manuais sujeitos a erros, cansativos e podendo ter um custo de tempo relevante. O objetivo deste trabalho é fazer a contagem automática de colônias de bactérias a partir de imagens digitais utilizando visão computacional, a fim de substituir a tradicional contagem manual. Os resultados finais indicam que a contagem automática, dependendo da imagem utilizada, é capaz de contar 71% das colônias de bactérias válidas presentes em uma placa de Petri.

Palavras-Chave: Classificador Estatístico; Visão Computacional; Bactérias.

Abstract

The manual counting of bacteria colonies is a tough task, since it involves methods that are still manual, prone to errors, tiring and may have a relevant time cost. The goal of this study is to make the automatic counting of bacteria colonies from digital images using computer vision, in order to replace the traditional manual counting. The end results indicate that the automatic counting, depending on the image used, is capable of counting 71% of the valid bacteria colonies present in a Petri dish.
Keywords: Statistical Classifier; Computer Vision; Bacteria.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

CLAHE	*Contrast Limited Adaptive Histogram Equalization*
DM	*Minimum Distance*
HT	*Hough Transform*
IDE	*Integrated Development Environment*
ML	*Machine Learning*
MLE	*Maximum-Likelihood Estimation*
RGB	*Red, Green e Blue*
OpenCV	*Open Source Computer Vision*

Sumário

1. INTRODUÇÃO

A contagem de colônias de bactérias é um procedimento importante no processo de análise e identificação desse tipo de agrupamento de microrganismos e serve como fonte de informação para diagnósticos de doenças e pesquisas. Tradicionalmente, o processo de contagem é realizado manualmente ou usando contadores automático. Os contadores automáticos têm alto custo e a contagem manual está sujeita a erros (CLARKE, et al., 2010). O processo de contagem de colônias de bactérias pode demandar tempo, paciência e materiais que auxiliem os pesquisadores, como lâminas, microscópios especiais, etc. A contagem manual é um trabalho difícil e sujeito a erros. Enquanto os contadores automáticos têm um custo alto de aparelhos, o que muitas vezes inviabiliza essa opção.

Nesse contexto é possível imaginar uma solução, através do processamento de imagens, para que o esforço seja diminuído e ainda aumentando a assertividade no processo de contagem de colônias. A classificação de imagens é o processo de extração de informações em imagens digitais que tem por objetivo reconhecer padrões e formas homogêneas a fim de encontrar áreas de interesse nessa representação gráfica. Segundo Gonzalez (2010), dificilmente existem áreas de empreendimento que não sejam, de alguma forma, impactadas pelo processamento digital de imagens.

Tendo conhecimento do poder computacional que existe na atualidade e sabendo da necessidade de se melhorar o processo de contagem de colônias de bactérias, surge o questionamento: como criar um método, ou métodos, com ajuda da visão computacional, altamente eficazes para contagem de colônias de bactérias?

Dessa forma, objetivo desse estudo é desenvolver um módulo de pré-processamento e contagem de colônias de bactérias a partir de imagens digitais empregando classificador estatístico. Para alcançar esse objetivo será necessário pré-processar imagens desse agrupamento de microrganismos para melhorar a qualidade da imagem, após esse processo é feita a segmentação da imagem pré-processada para facilitar e melhorar o processo de contagem, e por fim contar a quantidade de colônias a partir da imagem segmentada.

Uma das principais motivações é reduzir consideravelmente o tempo necessário para que todo o processo químico e físico seja realizado, dessa maneira será possível

acelerar diagnósticos e tratamentos de doença. Como consequência teremos: maior chance de cura e recuperação acelerada de pacientes; ajuda no gerenciamento da qualidade dos alimentos, ajudando a evitar doenças; redução do uso e o custo com materiais aplicados nas amostras, com base nos métodos utilizados; melhora na fidelidade dos resultados obtidos, uma vez que hoje é possível que alguns métodos atuais não tragam o número exato de colônias em seus resultados.

Por isso, foi desenvolvido um módulo de software para a contagem automática de colônias de bactérias que contribui para a redução do tempo necessário para identificar bactérias e acelera o processo de diagnósticos e tratamentos de doenças para maior chance de cura e recuperação acelerada de pacientes. Este módulo de software é parte de um sistema de gerenciamento da rotina do Laboratório de Bioquímica de Microrganismos (LBMic) do Laboratório de Ciência da Computação (LCC) da Universidade Federal de Goiás (UFG).

2. REFERENCIAL TEÓRICO

Nesta fundamentação teórica serão abordadas as definições de conceitos utilizados e apresentados nesse trabalho.

2.1 PROCESSAMENTO DE IMAGENS

MARQUES FILHO e VIEIRA NETO (1999), colocam isso de maneira bem direta, dizendo que o processamento de imagens vem sendo objeto de estudos em duas categorias bem distintas: (1) o aprimoramento de informações pictóricas para interpretação humana; e (2) a análise automática por computador de informações extraídas de uma cena.

Nesse trabalho, será utilizado o entendimento que Gonzalez e Woods (2010) sugerem: “processamento digital de imagens envolve processos cujas entradas e saídas são imagens e, além disso, envolve processos de extração de atributos de imagens até — e inclusive — o reconhecimento de objetos individuais”. Para elucidar o entendimento do termo, e contextualizando com a proposta do trabalho, considere a classificação de uma colônia de bactérias. Todo o processo de aquisição da imagem onde se apresenta a colônia, o pré-processamento dessa imagem, a segmentação da imagem, a detecção de padrões e a classificação em si estão no escopo do que é chamado de processamento digital de imagens nesse trabalho.

2.1.1 Técnicas de pré-processamento

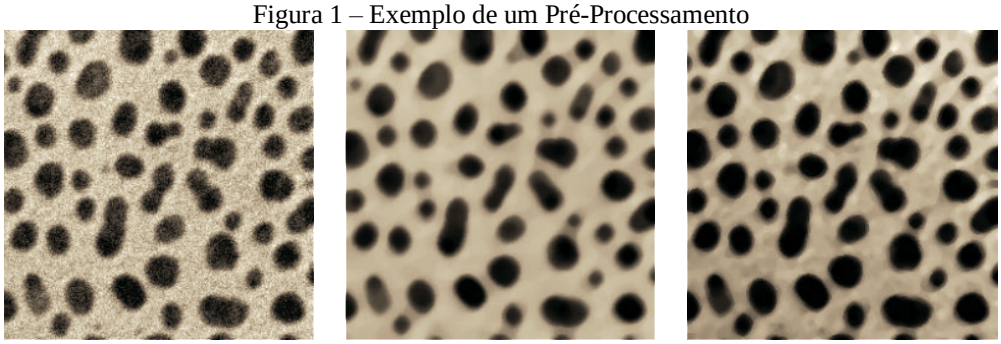
A imagem digital que resulta do processo de aquisição, pode apresentar problemas como imperfeições ou degradações que muitas vezes ocorrem por condições de iluminação, características do dispositivo utilizado na aquisição, ou até mesmo por especificidades da imagem, ser extremamente pequena é um exemplo, como é o caso do nosso objeto de estudo. Assim, as técnicas de pré-processamento, que têm a função de aprimorar a qualidade da imagem, auxiliando na melhora dos resultados obtidos.

Segundo GONZALES e WOODS (2010), a ideia da utilização de pré-processamento remonta a um antigo documento feito por White e Rohrer na década de 80 que combinou limiarização, gradiente e o laplaciano na solução de um problema de difícil segmentação.

A Figura 1 mostra um exemplo de pré-processamento: Em (A) pode ser vista a imagem original com ruído gaussiano, em (B) a imagem após aplicação de um filtro

mediana, e em (C) a imagem após a aplicação de um filtro passa-alta para realce dos contornos.

Figura 1 – Exemplo de um Pré-Processamento



Fonte: Esquef (2002, p. 23)

Dentre as técnicas de pré-processamento existentes, são destacadas duas que serão utilizadas nessa pesquisa: Equalização de Histograma e Suavização.

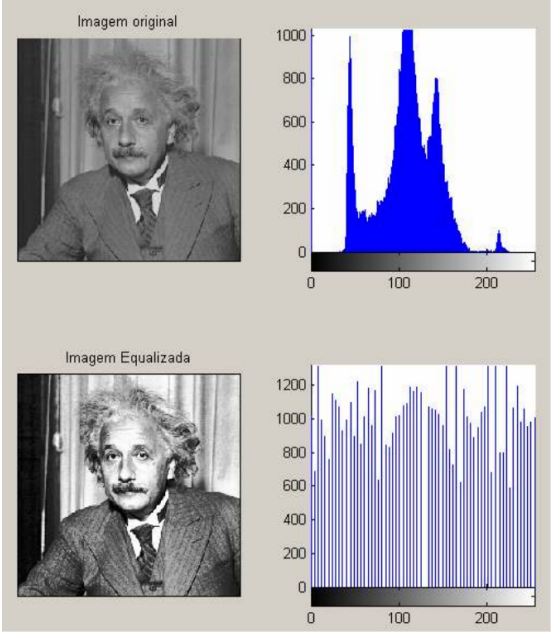
2.1.1.1 Equalização de Histograma

> O histograma de uma imagem digital com níveis de intensidade no intervalo [0, L – 1] é uma função discreta $h(rk) = nk$, onde rk é o k-ésimo valor de intensidade e nk é o número de pixels da imagem com intensidade rk. Costuma-se normalizar um histograma dividindo cada um desses componentes pelo número total de pixels da imagem, expresso pelo produto MN, onde, geralmente, M e N são as dimensões de linha e coluna da imagem. Dessa forma, um histograma normalizado é dado por $p(rk) = rk/MN$ para k = 0, 1, 2, … , L – 1. De modo geral, $p(rk)$ é uma estimativa da probabilidade de ocorrência do nível de intensidade rk em uma imagem. A soma de todos os componentes de um histograma normalizado é igual 1.(GONZALEZ, Rafael C; WOODS, Richard C. Processamento Digital de Imagens, 2010, p. 78)

Em uma linguagem mais simplificada pode-se dizer que o Histograma é um diagrama que representa a frequência de ocorrência e intensidade dos pixels em uma imagem. A equalização de histograma é uma técnica que se propõe a fazer com que o histograma final da imagem seja o mais constante possível, dessa forma distribuindo essa concentração de intensidade dos pixels. Na Figura 2, é possível observar uma imagem com equalização de histograma, percebe-se ao lado direito de cada imagem seu histograma. O histograma tem um limite esquerdo que representa o preto e um limite direito que corresponde ao branco. Na imagem original observa-se facilmente uma concentração de preto e no resultado final, após a técnica aplicada, é visto um melhor espalhamento do

histograma ao longo da gama de valores (0 a 255), o que favorece uma melhor análise da imagem.

Figura 2 – Exemplo de Aplicação de Equalização de Histograma



Fonte: Pinho e Couto (2004, p. 5)

2.1.1.2 Suavização

A suavização é uma técnica de pré-processamento que utiliza algoritmos que são aplicados à imagem afim de reduzir ruídos e preparar as imagens para outros processos, como a segmentação que será apresentada a seguir. Essa técnica é obtida pela aplicação de um filtro (matriz) que realiza a alterações no valor do pixel com base no valor dos pixels ao seu redor. Um ponto importante a se observar é que um uso indiscriminado ou sem o devido cuidado pode remover características essenciais de uma imagem. “Os filtros de suavização são utilizados para borramento e redução de ruído. O borramento é aplicado em tarefas de pré-processamento, como remoção de pequenos detalhes da imagem antes da extração de objetos (grandes) e conexão de pequenas descontinuidades em linhas ou curvas.” (GONZALEZ; WOODS, 2010, p. 100)

2.1.2 Segmentação

A segmentação é um processo que particiona uma imagem em várias regiões, com o objetivo de facilitar a identificação de detalhes. Essa técnica é muito importante dentro do processamento digital de imagens, uma vez que analisar imagens não segmentadas pode ser uma tarefa difícil e demorada. QUEIROZ e GOMES falam sobre a segmentação, que nada mais é que a subdivisão da imagem em partes ou objetos que constituem o todo. Algoritmos modernos conseguem identificar diferenças entre um ou mais objetos, ou entre objetos e o *background*.

Ainda segundo Gonzalez e Woods (2010), a precisão da segmentação determina o sucesso ou o fracasso final dos procedimentos de análise computadorizada. Portanto, deve ser feita de maneira criteriosa para refletir êxito ao final de qualquer pesquisa.

2.1.2.1 Detecção de Ponto, Linha e Borda

Três tópicos importantes dentro de técnicas de segmentação são abordados aqui. Eles tratam da detecção de mudanças locais abruptas de intensidade. Os três tipos são os pontos isolados, as linhas e as bordas. Para essas técnicas são utilizadas máscaras sobre os pixels da imagem para detecção de uma descontinuidade. Cada pixel e seus pixels vizinhos têm um valor do nível de cinza multiplicados por uma constante e a soma destes valores representa a máscara de resposta daquele ponto.

A técnica mais simples é a detecção de pontos, onde é percebida uma mudança drástica do valor de cinza em relação aos seus vizinhos. Segundo Gonzalez e Woods (2010), intuitivamente, a ideia é que a intensidade de um ponto isolado será muito diferente do seu entorno e, portanto, será facilmente detectável. A Figura 3, mostra uma máscara laplaciana de detecção de pontos.

Figura 3 – Filtro Laplaciano – Detecção de Ponto

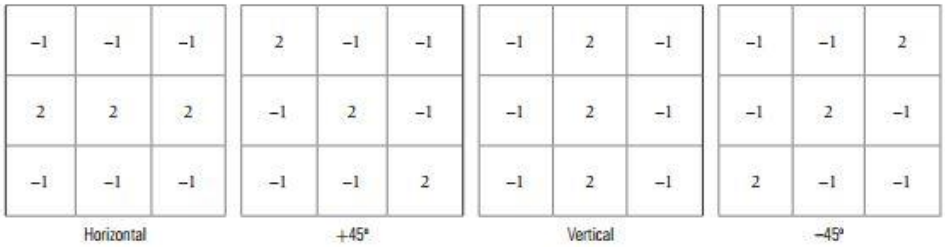
1	1	1
1	−8	1
1	1	1

Fonte: Gonzalez e Woods (2010, p. 459).

Já a detecção de linha avança na complexidade uma vez que é necessário achar pixels (pontos) semelhantes e verificar se fazem parte de uma linha comum. “Uma linha

pode ser vista como um segmento de borda em que a intensidade do fundo de cada lado da linha ou é muito superior ou muito inferior à intensidade dos pixels da linha" (GONZALEZ; WOODS, 2010, p. 456). Para detecção de linhas, como exemplo, pode-se ver o filtro Laplaciano na Figura 4, uma máscara composta de valores positivos e negativos que são calculados em todas as posições de uma imagem selecionada.

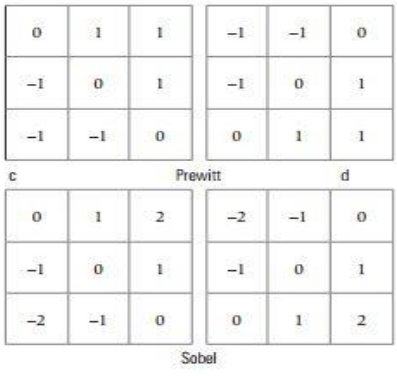
Figura 4 – Filtro Laplaciano – Detecção de Linhas



Fonte: Gonzalez e Woods (2010, p. 461).

De acordo com Marques Filho e Vieira Neto (1999), define-se borda como a fronteira entre duas regiões cujos níveis de cinza predominantes são razoavelmente diferentes. Computacionalmente, para detectar uma borda, os pixels de borda mudam de valor abruptamente em relação aos seus vizinhos, com isso é fácil detectar os níveis de intensidade de cada pixel e identificar possíveis regiões de borda. Para detecção de bordas, como exemplo, existem as máscaras de Prewitt e de Sobel, utilizadas para detecção de bordas diagonais. A Figura 5 mostra os valores dessas matrizes.

Figura 5 – Máscaras de Prewitt e Sobel



Fonte: Gonzalez e Woods (2010, p. 467).

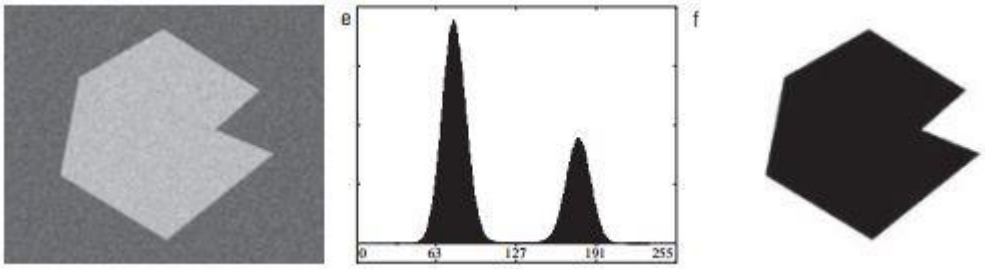
2.1.2.2 Limiarização

Limiarização é uma abordagem para segmentação que está baseada na análise das semelhanças de níveis de cinza. Dessa maneira são extraídos objetos de interesse mediante

a definição de um limiar *T* que separa os grupos de níveis de cinza da imagem (QUEIROZ; GOMES, 2001).

É possível citar como exemplo de limiarização, o método de Otsu. Esse método utiliza uma imagem em escala de cinza, e determina o melhor valor para o threshold fazendo com que consiga separar o fundo e a frente da imagem, atribuindo a cor branca ou preta para cada pixel, conforme o threshold. A Figura 6 mostra um exemplo do resultado desse método.

Figura 6 – Resultado Método de Otsu



Fonte: Gonzalez e Woods (2010, p.493).

2.1.2.3 Transformada de Hough

A Transformada de Hough (TH) é um método padrão para detecção de formas (linhas, círculos, elipses, etc.) facilmente parametrizadas em imagens computacionais. Em geral, a transformada é aplicada após a imagem sofrer um pré-processamento, fazendo com que se destaque as bordas presentes na imagem.

A TH foi desenvolvida por Paul Hough em 1962 e patenteada pela IBM. Para o cálculo de um círculo, ela se baseia em achar o centro de um possível círculo (pixels borda sobressalentes), e depois encontrar o raio desse possível círculo. Apenas será detectado aqueles que obedecerem aos limites colocados pela função. “Muitas vezes temos de trabalhar com ambientes não estruturados em que tudo o que temos é uma imagem da borda e não sabemos nada sobre onde possam estar os objetos de interesse” (GONZALEZ; WOODS, 2010, p. 488).

2.2 CLASSIFICAÇÃO DE IMAGENS

Classificação de imagens é um procedimento em que, a partir de dados observados da imagem, faz-se a extração de informações para posteriormente reconhecer

padrões e objetos homogêneos presentes nela. A classificação de imagens pode ser divida em duas partes principais:

- Classificação Supervisionada: é baseada na ideia de que o programador irá auxiliar, o programa, na descrição(rótulo) e características (features) dos dados para a criação de um datasets (conjunto de dados) tratado, o qual será utilizado para o treinamento do algoritmo desenvolvido. A porcentagem de acertos se baseia na qualidade de um datasets para o reconhecimento de padrões. Serve para identificar um ou vários objetos específicos.

- Classificação Não-Supervisionada: diferente do supervisionado, o programador não utiliza um conjunto de dados tratados, fazendo com que force o algoritmo a identificar os padrões existentes de forma abstrata, agrupando possíveis pixels relacionados em classes. Ele consegue identificar um objeto na imagem, mas não consegue descrevê-lo por falta de um Dataset rotulado.

2.3 RECONHECIMENTO DE PADRÕES

> O objetivo do reconhecimento de padrões é identificar objetos na cena a partir de um conjunto de medições. Cada objeto é um padrão e os valores medidos são as características desse padrão. Um conjunto de objetos similares, com uma ou mais características semelhantes, é considerado como pertencente à mesma classe de padrões. (QUEIROZ, José Eustáquio Rangel de; GOMES, Herman Martins. Introdução ao Processamento Digital de Imagens, 2001, p. 26)

O reconhecimento de padrões tem diversas aplicações, desde o reconhecimento facial que é utilizado para desbloquear smartphones modernos até mesmo a biometria. Uma das técnicas para reconhecimento de padrões é o uso de algoritmos machine learning (ML). O ML é uma área de pesquisa da Inteligência Artificial que visa ao desenvolvimento de programas de computador com a capacidade de aprender a executar uma dada tarefa com sua própria experiência (FACELI et al., 2011).

2.4 CLASSIFICADORES ESTATÍSTICOS

A classificação estatística é o problema de identificar a qual de um conjunto de categorias (subpopulações) uma nova observação pertence, com base em um conjunto de dados de treinamento contendo observações (ou instâncias) cuja participação na categoria é conhecida. É assumido em relação à seleção de variáveis de transformação e estruturação geral do problema.

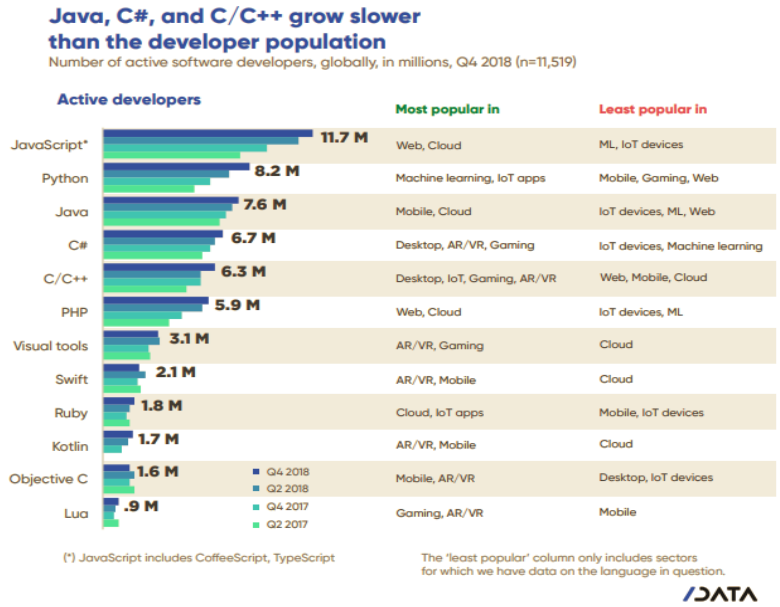
Pode-se utilizar como exemplo de classificação estatística, atribuir um determinado e-mail a caixa de spam ou não-spam, ou até mesmo atribuir um determinado diagnóstico ao paciente baseado nas informações apresentadas pelo seu exame.

2.5 PYTHON

"*Python* é uma linguagem de altíssimo nível (em inglês, Very High Level Language) orientada a objetos, de tipagem dinâmica e forte, interpretada e interativa." (BORGES, 2009, p. 10). Foi criada em 1990 por Guido Van Rossum. É uma linguagem de sintaxe simples, mais muito poderosa e funcional. Sua curva de aprendizagem é rápida facilitando o seu uso desde aplicações simples, até programas mais pesados e complexos.

Python, por ter grandes vantagens sobre outras linguagens de programação, vem sendo cada vez mais utilizado. Como pode ser visto na Figura 7, hoje é a segunda linguagem de programação mais utilizada do mundo, com cerca de 8,2 milhões de usuários ativos, de acordo com o State of The Developer Nation – 16th edition (2019). A maior e mais perceptível vantagem de uma linguagem interpretada, como o Python, é a rapidez para iniciar a execução do código já escrito. Uma vez que a máquina esteja com o Python instalada em seu sistema, ela poderá executar o script perfeitamente.

Figura 7 – Ranking de Linguagens de Programação



Fonte: *State of The Developer Nation – 16th edition* (2019, p. 28)

2.6 OPENCV

Segundo BRADSKI e KAEHLER, a visão computacional é uma importante ferramenta usada para que se possa transformar dados de uma imagem ou de um vídeo em uma decisão ou uma nova representação. Pode-se ter uma imagem inicial que não nos fala muita coisa e que após o uso da visão computacional vai nos mostrar a distância entre objetos, quantas pessoas existem numa cena, ou que tipo de colônia de bactéria está presente na imagem.

OpenCV é uma biblioteca de visão computacional de código aberto. Foi escrita na linguagem C e C++ e executa nos 3 principais sistemas operacionais: Windows, Linux e macOS. Pode ser uma biblioteca integrada com as interfaces Python, Ruby, Matlab, entre outras linguagens de programação. “O OpenCV foi projetado para eficiência computacional e com forte foco em aplicativos em tempo real” (BRADSKI; KAEHLER, 2008, p. 1, tradução nossa).

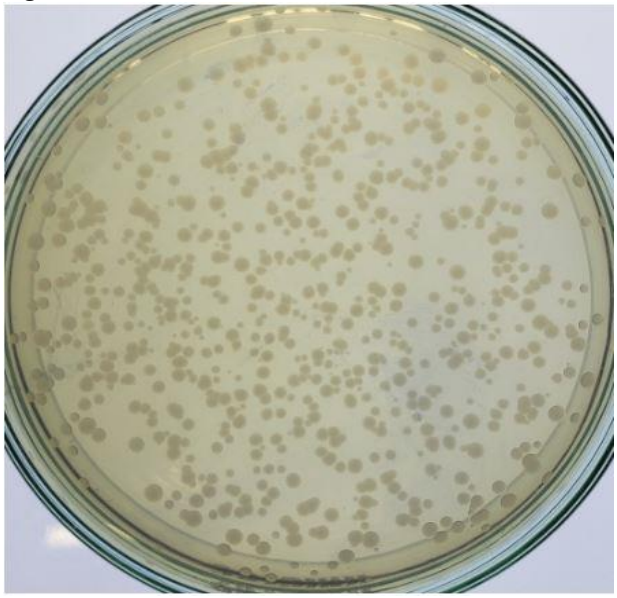
O OpenCV visa fornecer as ferramentas básicas necessárias para resolver problemas de visão computacional. Possui várias funções que auxiliam no pré-processamento, segmentação, processamento dos dados obtidos, entre várias outras funções. Muito utilizada por ser de fácil implementação e aprendizado, além de ser muito completa também.

2.7 COLÔNIAS DE BACTÉRIA

As bactérias são organismos microscópicos unicelulares. Elas estão entre as formas de vida mais primitivas da terra. Há milhares de tipos diferentes de bactérias, e elas vivem em todos os ambientes concebíveis em todo o mundo. Elas vivem no solo, na água do mar e nas profundezas da crosta da Terra. (BUSH, Larry M. Considerações gerais sobre bactérias, 2018, p.1)

A Figura 8 mostra um exemplo de cultivo de bactérias em uma placa de Petri – recipiente tratado com líquido de Agar onde possui misturas de nutrientes, sais e aminoácidos propícios para o desenvolvimento da amostra a ser analisada, e em seguida, é colocado em algum lugar para fazer com que as colônias possam se reproduzir.

Figura 8 – Colônia de Bactérias sobre uma Placa de Petri.



Fonte: LBMic/LCC.

Uma colônia é definida como uma massa visível de microrganismos todos originários de uma célula mãe solteira; portanto, uma colônia constitui um clone de bactérias geneticamente similares e agrupadas em uma determinada região.

3. METODOLOGIA

Classificação é a parte mais abstrata do processo de visão por computador. Ela representa o alto nível e permite obter a compreensão e a descrição final da imagem analisada. A classificação parte da premissa que a similaridade entre objetos implica que eles possuam características similares, formando classes/agrupamentos. O resultado da classificação pode ser percentual (indicando o % de chance da ocorrência de alguma classe) ou também pode ser uma imagem com algumas características enfatizadas para auxiliar o especialista em sua tomada de decisão (ZHANG, 2019). De outra maneira, podemos considerar que a fase de classificação consiste em reconhecer um objeto, uma forma ou, de modo geral, uma entidade particular da imagem. Dado um conjunto de classes e um padrão apresentado como entrada para o sistema, o problema consiste em decidir a que classe o padrão pertence. Deve haver a alternativa de rejeição do padrão (NIXON, M.; AGUADO, 2002).

O processo de classificação aplicado nesse trabalho consiste de duas etapas. Primeiro, será aplicado métodos de pré-processamento e segmentação, para tratar a imagem de entrada, onde será aplicada funções da própria função OpenCV. A segunda etapa consiste em determinar qual das funções de processamento tem maior probabilidade de resultado para o classificador– etapa de teste. A que possuir maior valor será considerada como o modelo mais provável para representar este novo padrão desconhecido (SHIH, 2010). Uma das características principais dos classificadores estatísticos é que necessitam de informações detalhadas a respeito das estatísticas sobre as imagens a serem classificadas. Este fato mostra-se como uma limitação na utilização dos mesmos, já que processos de manipulação dos dados, anteriores à classificação propriamente dita, tornam-se indispensáveis, ou, principalmente, quando nenhuma informação a priori a respeito dos dados está disponível.

Os classificadores estatísticos frequentemente assumem que valores de atributos possuem distribuição normal de probabilidade e usam os dados fornecidos para determinar a média e a covariância da distribuição (DUDA; HART; STORK, 2000). O classificador por máxima verossimilhança gaussiana (Maximum-Likelihood Estimation, MLE) baseia-se na estimação de modelos estatísticos para cada uma das classes envolvidas no processo de classificação. Estes modelos representam uma função cujo argumento é um vetor de feições e cuja saída é um número relacionado à probabilidade de este vetor

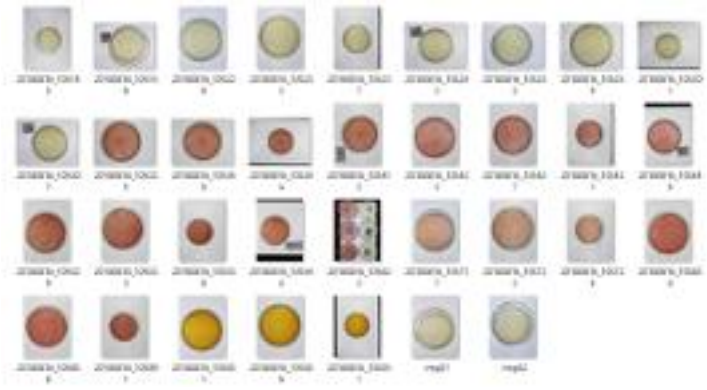
pertencer ao modelo em questão. Este modelo é representado por uma função gaussiana multivariada em *n* dimensões, onde *n* é o número de feições.

A estimação consiste na determinação do vetor média e da matriz de covariância, que são os parâmetros que definem uma gaussiana multivariada (GONZALEZ; WOODS, 2010). A eficácia da MLE depende da obtenção de uma acurada estimação da média e da covariância para cada classe. Para que isto ocorra, deve existir um número suficiente de amostras para cada uma destas classes. Quando isto não ocorre obtêm-se estimações incorretas dos elementos da covariância resultando num fraco desempenho de classificação. Quando o número de amostras de treinamento por classe é limitado pode ser mais eficaz recorrer a um classificador que não faz uso da covariância, dependendo somente da média das classes, observando que para um dado número de amostras estes podem ser estimados mais eficazmente que as covariâncias (RICHARDS, 1993). O classificador por distância mínima (Minimum Distance, DM) possui estas características, consistindo em um caso particular da classificação por MLE onde a matriz de covariâncias para as diferentes classes é igual e, além disso, múltipla da matriz identidade (RICHARDS, 1993).

4. RESULTADOS ALCANÇADOS

Para os estudos feitos nessa monografia, foram utilizadas imagens de colônias de bactérias cultivadas no LBMic do IPTESP cujas imagens foram obtidas com um smartphone no Laboratório de Computação Científica da Pontifícia Universidade Católica de Goiás. A Figura 9 mostra uma coleção dessas imagens, sendo que alguma delas são a visualização da mesma colônia de bactérias, apenas com variações de ângulos. Foram usadas 34 imagens de colônias de bactérias.

Figura 9 – Base de Dados para Estudo



Fonte: LBMic/LCC.

Para escrita e compilação dos códigos foi escolhida uma IDE específica. A IDE, Integrated Development Environment, em português Ambiente de Desenvolvimento Integrado, que nada mais é que um software que conta com ferramentas de apoio ao desenvolvimento com o objetivo de facilitar e agilizar o processo. No estudo em questão foi utilizada a ferramenta PyCharm. O PyCharm fornece complementação de código inteligente, inspeções de código, realce dinâmico de erros e correções rápidas, juntamente com refatorações de código automatizado e recursos de navegação avançados. (PyCharm, O IDE Python para desenvolvedores, 2020)

A Figura 10 mostra o algoritmo onde como primeiro passo se faz a leitura da imagem na linha. Após o carregamento da imagem é feita uma suavização na imagem lida. Como a imagem tem dimensões maiores que o limite da janela onde ela seria exibida, também se fez necessário executar um redimensionamento dessa imagem, pensado como o terceiro passo. Além disso, foi necessário fazer um recorte na imagem para poder dar destaque à região onde continha as colônias. A intenção desse recorte seria exatamente para aumentar a relevância da região onde vai ser feito o processamento, de forma que o

restante da imagem não influenciasse na futura contagem, e com isso melhorar os possíveis resultados.

Figura 10 – Algoritmo do Pré-Processamento

Title

```
1 Pré- Processamento
2
3 1° Ler a Imagem
4
5 2° Aplicar um filtro de suavização
6
7 3° Redmensionar a imagem. (Onde nesse caso está
  sendo em 21% da imagem original)
8
9 4° Fazer o recorte da imagem, destacando apenas o
  local de processamento
10
11
```

Fonte: Os autores.

O processo final de pré-processamento é feito com a ajuda de um filtro de Equalização de Histograma do OpenCV. Na Figura 11 pode-se ver as linhas 27 e 28 onde a função createCLAHE() é utilizada com esse fim. Através dessa função é possível equilibrar a intensidade dos pixels na matriz da imagem e o resultado final é importante pois é possível destacar, de forma constante, todos os detalhes da imagem. Também na Figura 11 é mostrado o resultado da imagem recortada na linha 24 e o resultado final do pré-processamento na linha 31.

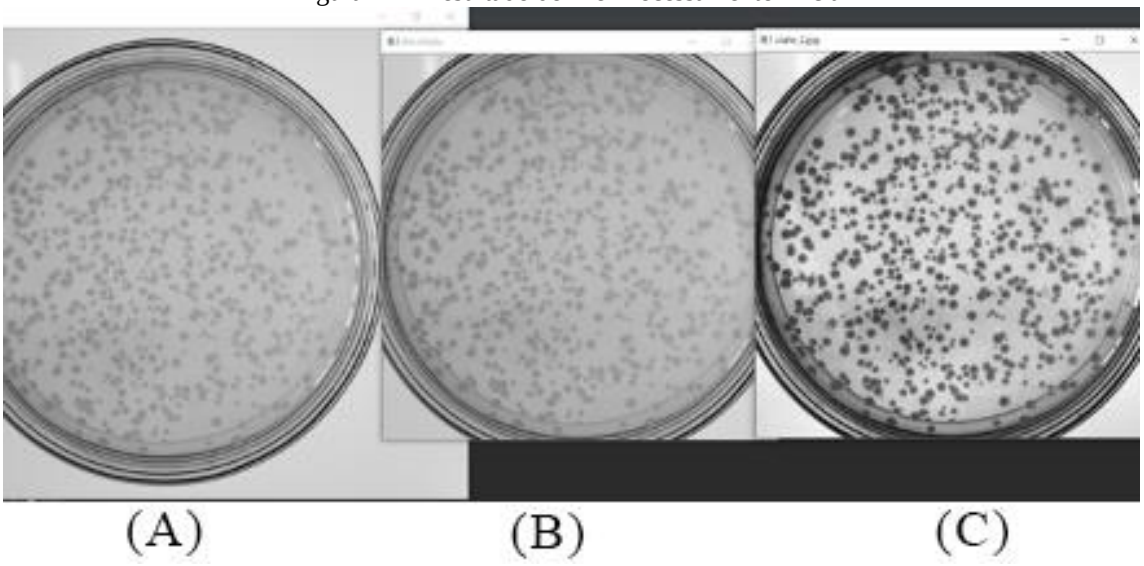
Figura 11 – Código com Comentários

```
23  # Mostrando a imagem recortada
24  cv.imshow("Recortada", cropped)
25
26  # Equalização de Histograma com a função createCLAHE()
27  clahe = cv.createCLAHE(clipLimit=4.0, tileGridSize=(4,4))
28  cl1 = clahe.apply(cropped)
29
30  # Mostrando a imagem equalizada
31  cv.imshow('clahe_2.jpg',cl1)
```

Fonte: Os autores.

Através do processamento, nessas três simples etapas (redimensionamento, recorte e equalização), pode-se trazer uma qualidade muito maior à imagem, fazendo com que fique muito mais fácil executar a contagem. Na Figura 12, pode-se ver o resultado desse pré-processamento inicial, onde a Figura 12(a) é a parte redimensionada, a Figura 12(b) está com um recorte dando foco apenas à área onde contém as colônias e a Figura 12(c) mostra a imagem equalizada e que se pode facilmente perceber as colônias.

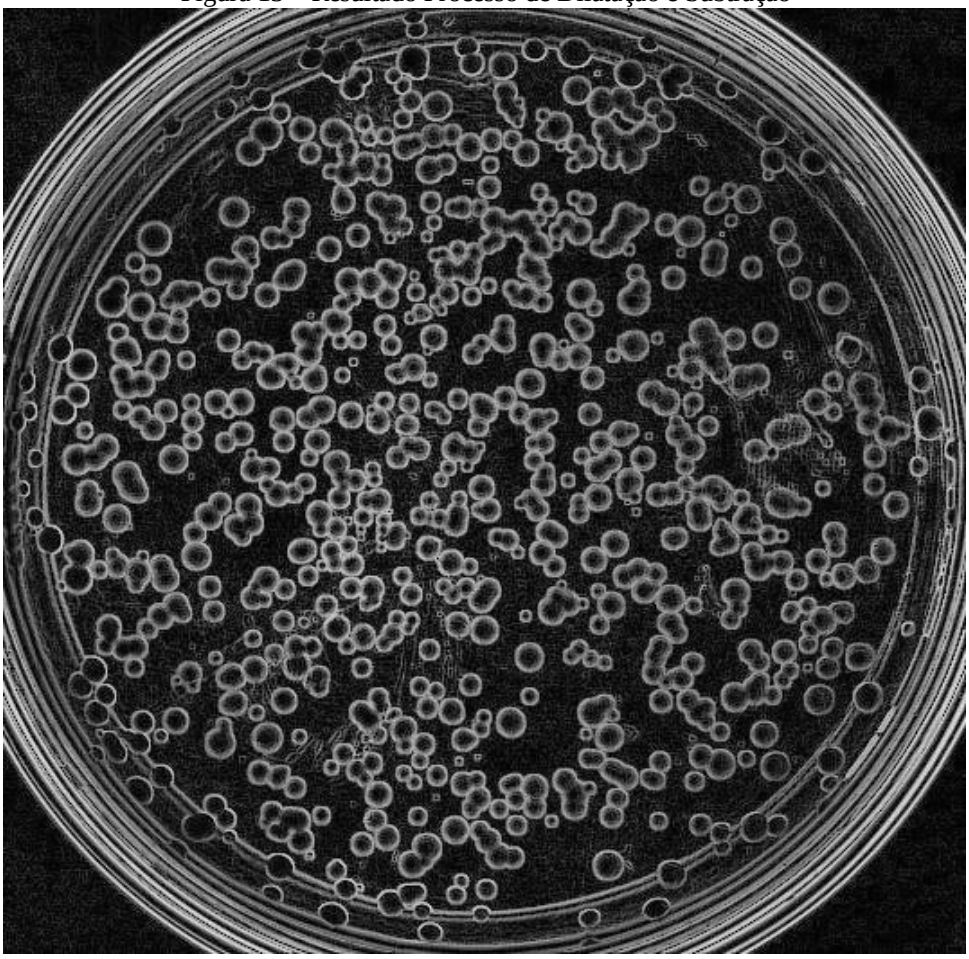
Figura 12 – Resultado do Pré-Processamento Inicial



Fonte: Os autores.

Após essa fase de pré-processamento foi iniciada a fase de segmentação da imagem, para extrair os principais dados (colônias) que serão processados. Utiliza-se, como ponto de partida para essa tarefa um processo de dilatação e subtração da imagem. Esse processo consiste em aplicar o processo de dilatação à uma imagem, o qual expande as características presentes na mesma, e depois subtrair o resultado dessa expansão com a imagem original. Esse processo faz com que todo o conteúdo da amostra original seja retirado da imagem expandida, fazendo com que seja ressaltado as bordas da colônia. Em seguida, é aplicado a Equalização de Histograma na imagem resultante, para melhorar o realce da imagem. A Figura 13, mostra o resultado obtido em uma das amostras utilizadas

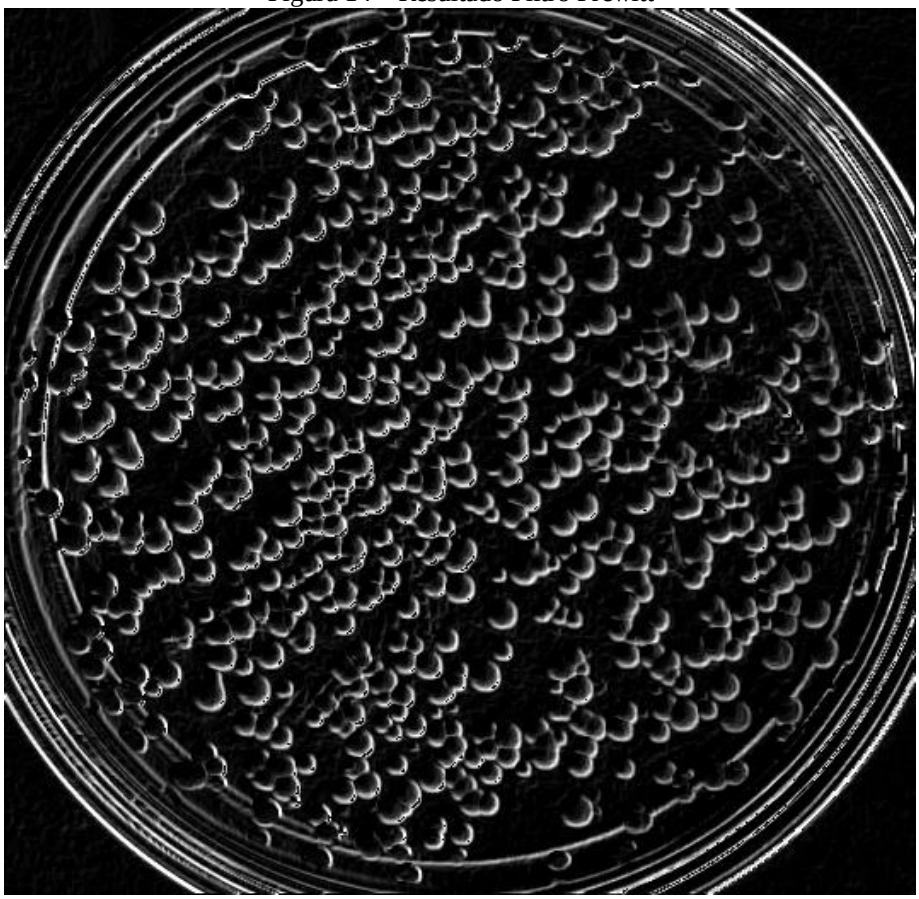
Figura 13 – Resultado Processo de Dilatação e Subtração



Fonte: Os autores.

Outra técnica utilizada para segmentação foi o uso de filtro passa-alta, para isso foi utilizado o filtro de Prewitt. Esse filtro consiste em 2 matrizes de 3x3, capazes de evidenciar bordas de acordo com a matriz utilizada. Essas matrizes são capazes de detectar bordas na horizontal e na vertical. Foram utilizadas as duas matrizes (Horizontal e Vertical) separadamente, e no final foi somado o resultado desse processamento para obter o resultado final utilizável. A Figura 14 mostra como ficou o resultado da aplicação desse filtro em uma das amostras.

Figura 14 – Resultado Filtro Prewitt



Fonte: Os autores.

Com as bordas definidas através do processo anterior, partimos para a detecção e contagem das colônias. Foi utilizada a Transformada de Hough, como classificador estatístico, para detecção de círculos presentes nas imagens. Foi utilizada quase que toda a configuração padrão da documentação desse processo. Entretanto, foram alterados os parâmetros – parâmetro 1 e parâmetro 2 – e o mínimo e o máximo raio, onde pode ser observado na Tabela 1 os valores configurados para a imagem de exemplo dessa monografia.

Tabela 1 – Valores da Função HT Configurados

Campos	**Valores**
param1	250
param2	20
minRadius	1
maxRadius	11

Os valores foram alterados para obter um melhor resultado para o algoritmo. A Figura 15 mostra como ficou a configuração de toda essa parte, mostrando como foi arquiteto e pensado para uma sequência lógica. Já que a função necessita de uma imagem em escala de cinza para fazer o todo o seu processo, ainda é copiada a imagem para uma escala colorida, nesse caso sendo formato BGR, para realizar os desenhos da detecção

Figura 15 – Configuração Hough Transform

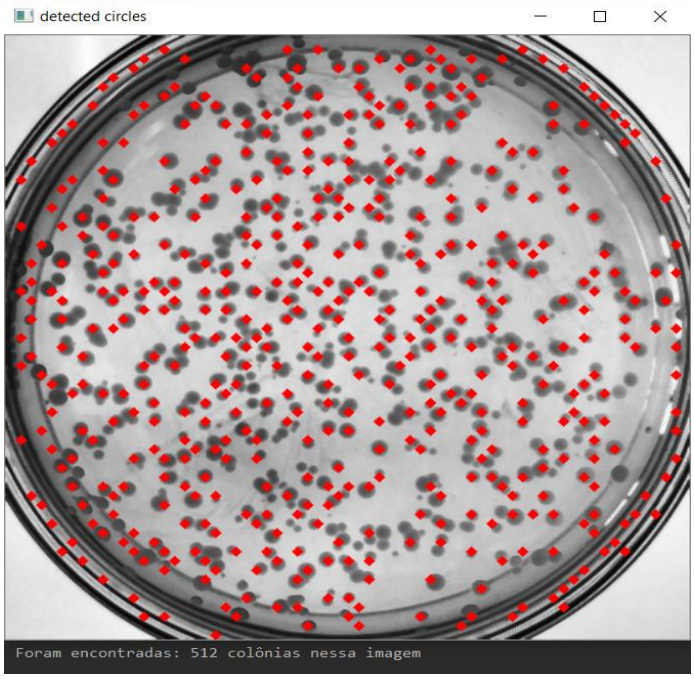
Title

```
# Detecção

import cv2

img = cv2.imread('imgred.jpeg',0)

cimg = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 8, 1,
                           param1=250, param2=20, minRadius=1, maxRadius=11)

circles = np.uint16(np.around(circles))
for i in circles[0, :]:

    cv2.circle(cimg, (i[0], i[1]), 2, (0, 0, 255), 3)
    contador += 1

cv2.imshow('detected circles', cimg)
print(contador)
```

Fonte: Os autores.

A seguir são mostrados os resultados de cada fase do processamento de imagens onde são feitas a detecção e contagem, começando pela Figura 16. Nessa imagem é possível ver o resultado da contagem realizada na imagem apenas pré-processada. O resultado foi positivo com boa detecção de verdadeiros positivos, entretanto, houve detecção de falsos positivos na borda da placa. Em colônias que ficaram sobrepostas, não houveram detecções, já que existe uma dificuldade computacional maior para identicar círculos nesse formato.

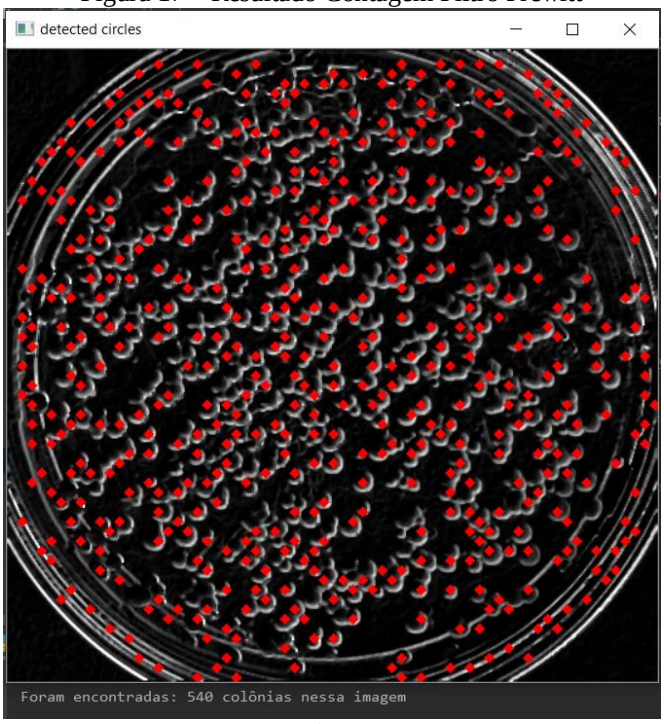
Figura 16 – Resultado Contagem Imagem Pré-Processada



Fonte: Os autores.

Já na Figura 17, é mostrado o resultado da contagem utilizando a imagem segmentada com o filtro de Prewitt. Com ajuda do filtro houve uma melhora onde foi obtido uma boa detecção de verdadeiros positivos, contudo, obteve uma detecção bastante significativa em falsos positivos presente na borda da placa.

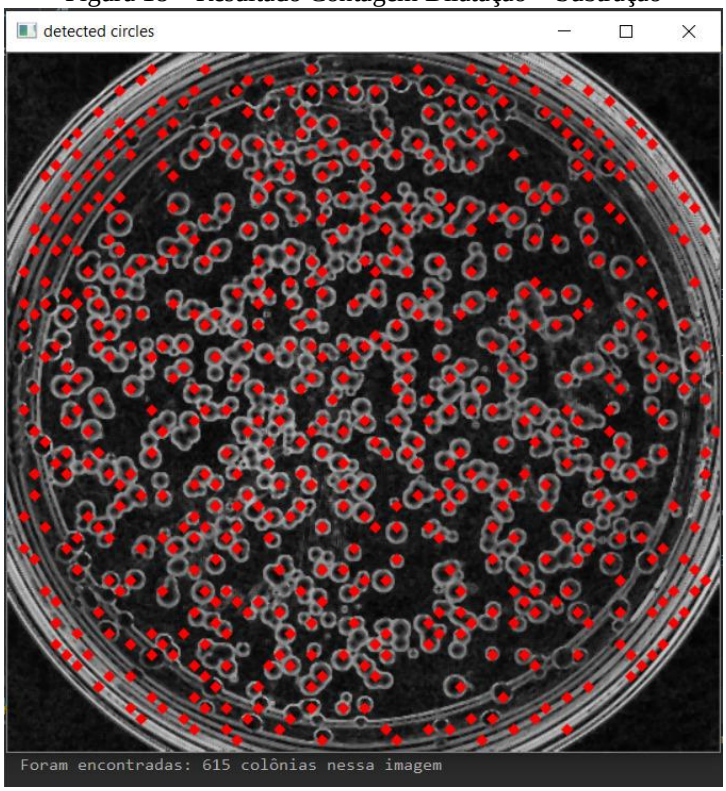
Figura 17 – Resultado Contagem Filtro Prewitt



Fonte: Os autores.

O próximo passo foi utilizar o resultado do processo de dilatação e subtração. A Figura 18 apresenta esse resultado. Assim como no resultado da Figura 17, é perceptível uma boa detecção de verdadeiros positivos, até melhor que no exemplo anterior, apesar disso, ainda existe uma detecção bastante significativa em falsos positivos presente na borda da placa. Também houve boas detecções em colônias que ficaram sobrepostas.

Figura 18 – Resultado Contagem Dilatação + Subtração



Fonte: Os autores.

Como foi visto, o primeiro resultado obteve 512 detecções de colônias, seguido por 540 e por fim 615 do outro método de processamento utilizado. Essa amostra que foi utilizada, como exemplo didático, possui 678 colônias no total presente na placa de Petri, as quais foram contadas manualmente. Considerando o melhor caso de resultado, que foi o último caso apresentado, onde foi levado em conta apenas os verdadeiros positivos, obteve – se aproximadamente 71% de acerto do resultado exato.

5. CONSIDERAÇÕES FINAIS

Tendo como resultado das contagens aplicadas nas amostras, foi apresentado que existe variância entre cada método adotado (aplicação de filtros, morfologia matemática, etc), detectando mais ou menos circunferências, também tendo como influência a aquisição, e a composição de cada amostra.

Se for levado em consideração o tempo utilizado por um ser humano para realizar a contagem, nesse exemplo do documento, o algoritmo obteve um melhor resultado, pois o mesmo precisou de um tempo inferior a 7s para apresentar o resultado. Esse tempo é bem inferior ao que poderia ser feito por um ser humano numa contagem manual, já que, dependendo do exemplo, um pesquisador pode gastar até mais de 20min contando uma amostra populosa. Todavia, o seu resultado obteve detecções verdadeiramente negativas, o que faz com que ele tenha um ponto de desvantagem. Além disso, foi necessário empenhar tempo adicional – uma variação de aproxidamente 5min adicionais, sendo de acordo com a imagem utilizada – para o recorte, redimensionamento e equalização da imagem, além dos processos de segmentação e para configuração dos valores do contador, a fim de obter um resultado final mais satisfatório. É importante também dizer que houve sobreposições de colônias, fazendo com que não houvesse detecção em alguns casos.

Apesar dos pontos de desvantagem da contagem feita com o algoritmo, ainda é possível dizer que existe mais vantagens que desvantagens nesse processo.

Para trabalhos futuros, será interessante utilizar redes neurais para potencializar os resultados. Por exemplo, poderia ser utilizado redes neurais convolucionais para a realização de toda a operação de detecção e contagem empregado em imagens digitais.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, L. E. **Python para Desenvolvedores**. 1° ed. Rio de Janeiro: Edição do Autor, 2009.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. 1° ed. United States of America: O'Reilly, 2008.

BRAS, J. **Nossa capa**: Alexander Fleming e a descoberta da penicilina. Med. Lab., Rio de Janeiro, v. 45, n. 5, p. I, out. 2009. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1676-24442009000500001&lng=pt&nrm=iso&tlng=pt. Acesso em: 21 de abr 2020.

BUSH, L. M. **Considerações gerais sobre bactérias**. Manual MSD, Kenilworth, EUA, abr. 2018. Disponível em: https://www.msdmanuals.com/pt/casa/infec%C3%A7%C3%B5es/infec%C3%A7%C3%B5es-bacterianas-considera%C3%A7%C3%B5es-gerais/considera%C3%A7%C3%B5es-gerais-sobre-bact%C3%A9rias. Acesso em 28 abr. 2020.

CARVALHO, I. T. **Microbiologia básica** / Irineide Teixeira de Carvalho. – Recife: EDUFRPE, 2010.

DUDA, R. O.; HART, P. H.; STORK, D. G. **Pattern Classification**, Wiley-Interscience, 2000.

ESQUEF, I. A. **Técnicas de entropia em processamento de imagens**. 2002. 127 f. Tese (Mestrado em Instrumentação Científica) – Centro Brasileiro de Pesquisas Físicas. Rio de Janeiro.

FACELI, K. et. al. **Inteligência artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011.

FILHO, O. M.; NETO, H. V. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3° ed. São Paulo: Pearson, 2010.

JETBRAINS. **PyCharm**: O IDE Python para desenvolvedores profissionais. Disponível em: https://www.jetbrains.com/pt-br/pycharm/. Acesso em 23 abr. 2020.

LAGO, A.; PÁDUA, J. A.. **O que é ecologia**. 1ª Ed. ebook. São Paulo: Editora Hedra Ltda., 2017.

NIXON, M.; AGUADO, A. S. **Feature extraction and image processing for computer vision**, Academic Press, 2002.

PINHO, S. R.; COUTO, R. M. F. **Equalização e modificação de histogramas**. 2004. Disponível em: https://web.fe.up.pt/~mandrade/tvd/2006/trabalhos1-2004/TD-trab1-histogramas.pdf. Acesso em: 23 abr. 2020.

QUEIROZ, J. E. R.; GOMES, H. M. **Introdução ao Processamento Digital de Imagens**. RITA - UFCG, Volume VIII, Número 1, 2001. Disponível em: http://www.dsc.ufcg.edu.br/~hmg/disciplinas/graduacao/vc-2016.2/Rita-Tutorial-PDI.pdf. Acesso em 21 abr. 2020.

RICHARDS, J. A. **Remote Sensing Digital Image Analysis**. Berlin: Springer-Verlag, 1993.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 3° ed. Rio de Janeiro: Elsiever, 2013.

SHIH, F. Y, **Image processing and pattern recognition**: Fundamentals and Techniques, IEEE Press, John Willey & Sons, 2010.

SLASHDATA. **State of the developer Nation 16th edition**. Disponível em: https://slashdata-website-cms.s3.amazonaws.com/sample_reports/ZAamt00SbUZKwB9j.pdf. Acesso em 15 abr. 2020.

TRABULSI, L. R.; ALTERTHUM, F. **Microbiologia**. 5° ed. São Paulo: Atheneu, 2008.

UNIRIO. **Métodos de Contagem Bacteriana**. Dísponível em: http://www4.unirio.br/dmp/nutricao-integral/aulas-teoricas/16-%20Metodos%20de%20Contagem%20Bacteriana%2001-2019.pdf. Acesso em: 26 de Outubro de 2019.

ZHANG, D. **Fundamentals of Image Data Mining**: Analysis, Features, Classification and Retrieval, Springer, 2019.

ANEXO A – Arquivo Redmensionamento.py

```
import cv2 as cv
import numpy as np

img = cv.imread('img03.jpg', 0)

img = cv.medianBlur(img, 5)

scale_percent = 21  # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
#resize image
resized = cv.resize(img, dim, interpolation=cv.INTER_AREA)

# crop image       Altura x  Largura
cropped = resized[60:580, 170:705]

# Aplicando Histograma
clahe = cv.createCLAHE(clipLimit=4.0, tileGridSize=(4, 4))
cl1 = clahe.apply(cropped)

equ = cv.equalizeHist(cropped)

# Visualizando Dados
cv.imshow('clahe_2.jpg', cl1)
# print('Resized Dimensions : ', resized.shape)
cv.imshow('Redimensionado', resized)
cv.imshow("Recortada", cropped)

# Salvando a imagem
# cv.imwrite("imgred03.jpg", cl1)

cv.waitKey(0)
cv.destroyAllWindows()
```

ANEXO B – Arquivo Filtro_Prewitt.py

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('imgred03DIL.jpg',0)
vetV = np.array([
    [-1, 0, 1],
    [-1, 0, 1],
    [-1, 0, 1]
])
vetH = np.array([
    [-1, -1, -1],
    [0, 0, 0],
    [1, 1, 1],

])

vetV2 = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])
vetH2 = np.array([
    [1, 1, 1],
    [0, 0, 0],
    [-1, -1, -1],

])

res = cv2.filter2D(img, -1, vetH)
res2 = cv2.filter2D(img, -1, vetV)
res3 = res + res2

res4 = cv2.filter2D(img, -1, vetH2)
res5 = cv2.filter2D(img, -1, vetV2)
res6 = res3 + res4 + res5
cv2.imshow("Original", img)
cv2.imshow("Horizontal", res)
cv2.imshow('Vertical', res2)
cv2.imshow('Resultado final', res3)
cv2.imshow('Outro Resultado', res6)


# Salvando arquivos
cv2.imwrite('imgred03DIL1.jpg', res3)
cv2.imwrite('imgred03DIL2.jpg', res6)
cv2.waitKey()
cv2.destroyAllWindows()
```

ANEXO C – Arquivo Morfologia.py

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# Image operation using thresholding
img = cv2.imread('imgred05.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel = np.ones((3, 3), np.uint8)

# Dilatando
opening = cv2.erode(img, kernel)

# Subtraindo
res = img - opening

# Salvando Resultados
cv2.imwrite('imgred05ERO.jpg', res)

cv2.waitKey()
```

ANEXO D – Arquivo TransformadaHou.py

```
import cv2
import numpy as np

contador = 0
img = cv2.imread('imgred03.jpg', 0)

img = cv2.medianBlur(img, 3)

cimg = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

# Detectando
circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 8, 1,
                           param1=250, param2=20,
minRadius=1, maxRadius=11)

circles = np.uint16(np.around(circles))
# Desenhando Resultados
for i in circles[0, :]:
    # draw the outer circle
    # cv2.circle(cimg, (i[0], i[1]), i[2], (0, 255, 0), 2)
    # draw the center of the circle
    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)
    contador += 1

# Mostrando Resultados
cv2.imshow('detected circles', cimg)
# cv2.imshow('Detector Edge', img2)
print('Foram encontradas: %s colônias nessa imagem' %
contador)
cv2.waitKey(0)
cv2.destroyAllWindows()
```