Classificação de severidade de Oídio em folhas de melão

Vitor Oliveira Ropke
Avenida Francisco Mota, 572
Pres. Costa e Silva, Mossoró – RN, 59625-900
vitor.ropke@alunos.ufersa.edu.br

Resumo *O uso da computação agiliza muito a resolução de vários problemas. Na área da horticultura é possível diagnosticar doenças fúngicas como o Oídio, que se manifesta visivelmente podendo ter sua severidade de infecção classificada por um algoritmo que fornece um resultado a partir de uma foto capturada por qualquer dispositivo. Os pixeis dessa imagem são contados e uma proporção de cinza (cor do fungo), verde e/ou amarelo (cor da folha) determina o quão severamente a folha está afetada de um nível de 0 a 5 de menos a mais impactada, respectivamente. É um trabalho que pode ser aprimorado por técnicas de remoção de plano de fundo através de inteligência artificial com aprendizado de máquina. Desta forma, o diagnóstico poderá ser mais preciso e diminuirá o prejuízo causado por falsos positivos e/ou negativos.*

Palavras-Chave: Classificação de severidade de Oídio. Contagem de pixeis. Melão. Oídio. Processamento digital de imagens. Proporção de cores.

Abstract *The use of computing speeds up the resolution of many problems. In the field of horticulture it is possible to diagnose fungal diseases such as Powdery mildew, which manifests itself visibly, and can have its infection severity classified by an algorithm that provides a result from a photo taken by any device. The pixel in this image are counted and a ratio of gray (fungus color), green and/or yellow (leaf color) determines how severely the leaf is affected from a level of 0 to 5 from least to most impacted, respectively. It is a work that can be improved by background removal techniques through artificial intelligence with machine learning. In this way, the diagnosis may be more accurate and will reduce the losses caused by false positives and/or negatives.*

Keywords: Color ratio. Digital image processing. Melon. Pixel count. Powdery mildew. Powdery mildew severity classification.

1 Introdução

O Oídio é uma doença causada por várias espécies de fungos que ocorrem em regiões ou épocas secas e quentes, mas que também pode aparecer em clima úmido e frio. Sua principal característica são manchas brancas que prejudicam as folhas, cobrindo-as, interferindo na fotossíntese e, se não tratado, pode se espalhar e matar a planta, reduzindo a produção da lavoura. Em alguns casos, o Oídio pode gerar perdas de até 60% da cultura [1].

O controle dessa doença passa por estudos para que seja necessário agir de forma eficiente, rápida e barata. Para isso, é importante o uso de recursos computacionais que automatizam e armazenam informações que agilizam no monitoramento do Oídio.

1.1 O uso da computação

A infecção do Oídio apresenta uma escala de progressão que vai de 0 a 5. Com uma foto da folha, é possível saber em qual estágio de infecção, ela está. A Figura 1

mostra isso em uma folha de melão.

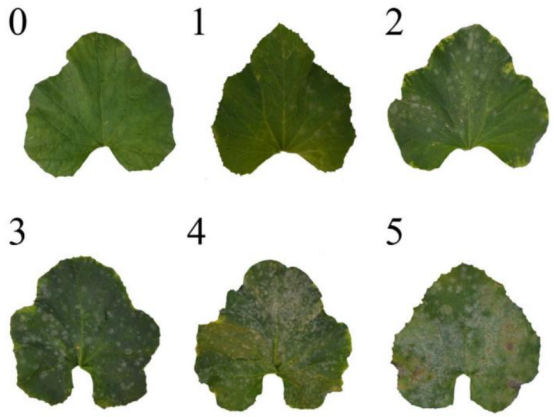


Figura 1: Diferentes estágios de infecção de uma folha por Oídio. Nível 0 – Sem infecção; 1 – Sem infecção visível; 2 – Baixa infecção; 3 – Média infecção; 4 – Alta infecção; 5 – Infecção total [2]

Com um aplicativo, é possível fotografar a folha e ele retorna o nível de infecção da mesma. O programa pode até mesmo armazenar essa classificação em um mapa, onde o usuário consegue ver quais regiões da plantação estão sendo afetadas e executar as ações necessárias.

2 Metodologia

A linguagem utilizada foi Python, devido à facilidade de implementação, entendimento de código e fácil integração com bibliotecas que trabalham com processamento digital de imagens, como OpenCV.

2.1 Contagem de pixeis por cores

A ideia principal é contar o número de pixeis cinzas, amarelos e verdes e obter a proporção de cinza sobre as outras cores da folha do melão. Com a porcentagem de cinza é possível determinar em qual classe de severidade de infecção, a folha se enquadra.

A figura 2 mostra o código usado para contar os pixeis da imagem. As principais cores da folha, verde e amarelo, e a principal cor da doença, cinza, foram definidas.

```
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    lower_gray = np.array([0, 0, 46])
    upper_gray = np.array([180, 43, 220])

    lower_yellow = np.array([26, 43, 46])
    upper_yellow = np.array([34, 255, 255])

    lower_green = np.array([35, 43, 46])
    upper_green = np.array([77, 255, 255])

    gray_mask = cv2.inRange(hsv_image, lower_gray, upper_gray)
    yellow_mask = cv2.inRange(hsv_image, lower_yellow, upper_yellow)
    green_mask = cv2.inRange(hsv_image, lower_green, upper_green)

    gray_count = np.count_nonzero(gray_mask)
    yellow_count = np.count_nonzero(yellow_mask)
    green_count = np.count_nonzero(green_mask)

    percentage = (gray_count / (gray_count + green_count +
yellow_count)) * 100.0
```

Figura 2: Código usado para contar o número de diferentes pixeis, separar suas respectivas máscaras (imagem com determinado intervalo de cor selecionado) e contar a porcentagem de cinza sobre outras cores da folha

A imagem foi convertida de RGB para HSV por ser mais fácil e mais preciso de trabalhar com intervalo de cores nesse formato [3]. Essa estratégia é necessária, pois raramente as configurações para captura de imagens por câmeras é algo fixo. O ambiente muda e definições de brilho e balanço de branco também se alteram, caso o dispositivo esteja configurado automaticamente. Até mesmo as características da folha não são homogêneas. Sendo as-

sim, dificilmente existirá uma imagem que tenha todos os pixeis de uma folha, definidos como 100% verde, ou, em valor RGB, (0, 255, 0), sendo necessário utilizar um intervalo de valores para cada cor entre "lower_color" e "upper_color" em várias linhas de comando da figura 2.

Após definir os valores máximo e mínimo para cada cor, encontra-se as partes da imagem que estejam dentro de cada intervalo. O código "*cv2.inRange*" retorna uma imagem com as cores nesse intervalo, definido o resto da imagem com pixeis pretos. Essa imagem passa a se chamar máscara.

Todas as imagens são interpretadas como vetores com valores de 0 a 255. Pixeis pretos possuem valor 0. Então, para contar os pixeis da máscara, basta contar os valores não nulos.

Com a contagem de pixel de cada máscara divide-se os pixeis cinza (cor da doença) com a soma dos pixeis da folha (verde e/ou amarelo) e obtém-se a porcentagem multiplicando por 100.

Também houve uma tentativa de obter todas os pixeis com o valor acima de 100, zerando todos os outros, para gerar uma máscara onde a folha seria identificada. O resultado não foi como esperado e a técnica foi descontinuada.

Figura 3: À esquerda, foto da folha e à direita, máscara da cor verde, encontrada na imagem original. Regiões brancas são consideradas verdes pelo algoritmo. Então, o valor 100 não é suficiente para cobrir todos os pixeis da folha.

2.2 Classificação da severidade da doença

Com a porcentagem encontrada, determina-se a classificação através de intervalos mostrados no código da figura 2.

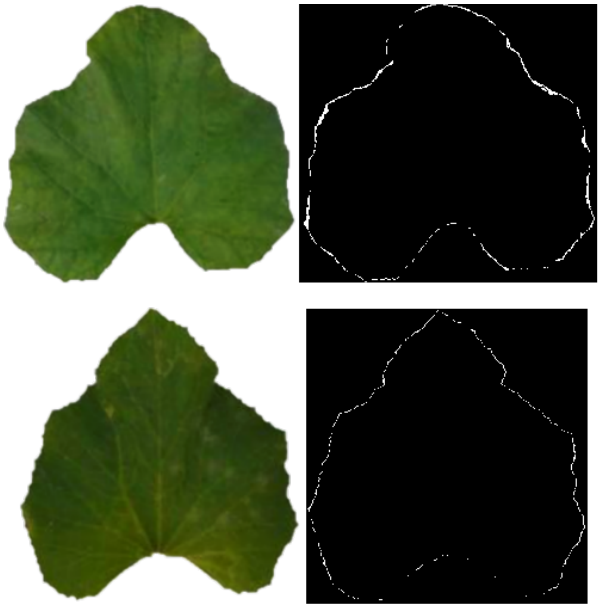
```
    illness_level = percentage

    # Valores são considerados com margem de erro em caso de falsa
detecção
    # Sem infecção
    if illness_level < 3.0:
      class = 0
    # Sem infecção visível
    elif illness_level < 10.0:
      class = 1
    # Baixa infecção
    elif illness_level < 20.0:
      class = 2
    # Média infecção
    elif illness_level < 30.0:
      class = 3
    # Alta infecção
    elif illness_level < 40.0:
      class = 4
    # Altíssima infecção
    else:
      class = 5
```

Figura 4: Código usado para definir a classe de infecção em que a folha se encontra

A variável "*illness_level*" armazena a porcentagem de cinza encontrada na folha. Caso esse valor fique abaixo de 2, a folha é classificada como sem infecção, na classe 0. O valor da porcentagem considerado, foi 3 para considerar uma margem de erro de possíveis pixeis que foram erroneamente definidos como cinza. Para as próximas

classes, ficou definido o seguinte: porcentagem maior ou igual a 3 e menor que 10, define-se como classe 1 (sem infecção visível); porcentagem maior ou igual a 10 e menor que 20, classe 2 (baixa infecção); porcentagem maior ou igual a 20 e menor que 30, classe 3 (média infecção); porcentagem maior ou igual a 30 e menor que 40, classe 4 (alta infecção) e porcentagem maior ou igual a 40, classe 5 (altíssima infecção ou infecção total).

Os valores foram definidos com base em resultados conhecidos de folhas, mostrado na figura 1. Essas folhas foram passadas pelo algoritmo e suas porcentagens de cinza foram calculadas, alguns com resultados inesperados, por diferença no balanço de branco e outros parâmetros que devem ser tratados, futuramente.



3 Resultados

A maioria das folhas foi classificada no nível 5, sendo que algumas foram falsos positivos devido a não remoção do plano de fundo, levando a ruído na detecção da cor cinza mostrado na figura 7.

Figura 5: Cima, esquerda é a folha classificada no nível 0. À direita é a máscara cinza identificada na folha. A parte cinza está destacada como branca. Embaixo, esquerda é a folha classificada no nível 1 e à direita é a sua máscara cinza. Observa-se que, na folha debaixo, as manchas não foram identificadas pelo algoritmo, classificando-o como 0.

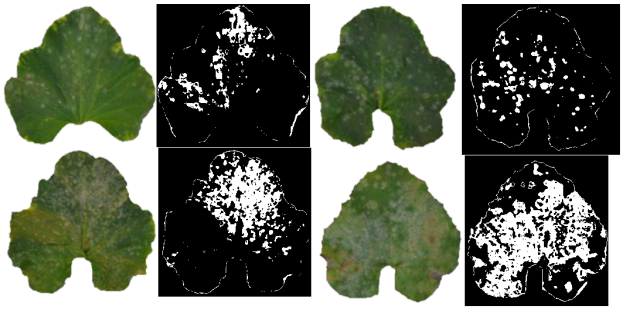


Figura 6: Nessa imagem, a imagem original está à esquerda e a máscara da parte cinza identificada na imagem está à direita da folha original. A parte branca destacada na máscara é a cor cinza que foi identificada. Em cima, à esquerda é a folha que foi classificada no nível 2. À cima, direita é a folha classificada no nível 3. Embaixo, esquerda, nível 4. Embaixo, direita, nível 5.

A partir dessa classificação, as porcentagens obtidas foram: folha 0 → 2.04%; folha 1 → 1.02%; folha 2 → 11.5%; folha 3 → 9.09%; folha 4 → 22.97% e folha 5 → 47.47%.

Nesse projeto, foi definido que a classe em que a folha foi enquadrada, ficará impressa na foto de saída do programa, bem como no nome do arquivo, onde o sistema operacional pode classificar automaticamente os arquivos em ordem alfanumérica e, desta forma, pode-se avaliar, em sequência, a taxa de acerto do programa para cada classe.

Figura 7: À esquerda, a imagem original, classificada como nível de infecção 5 (número na parte superior esquerda). À direita, a imagem processada, mostrando todos os pixeis considerados cinzas, marcados em branco. Observa-se que a maioria cinza da imagem está fora da folha

Com uma iluminação do ambiente mais uniforme e sem reflexos na imagem, a folha consegue ser classificada corretamente conforme mostrado na figura 8.

Figura 8: A folha foi classificada no nível 1 e as partes cinzas da folha foram corretamente identificadas, apesar de algumas partes do cenário também terem sido contadas.

Observando todos os resultados, entende-se que é necessário um tratamento nas imagens com a remoção do plano de fundo. A maior parte da falha no programa está na detecção de pixeis cinzas que não estão na folha.

Num momento desse trabalho foi usada a detecção de bordas para destacar a folha. Porém, vários ruídos na captura e as particularidades de cada cenário impossibilitam uma alta taxa de acerto nessa abordagem.

A melhor técnica seria o uso do aprendizado de máquina, onde é fornecido imagens de folhas e o algoritmo consegue destacar onde elas estão e remover completamente o plano de fundo.

Problemas com versões e compatibilidade de bibliotecas prejudicaram a busca de novas formas para detectar a folha.

4 Conclusão

A contagem de pixeis para classificação de severidade de Oídio em folhas de melão é uma técnica simples mas que pode ter resultados muito diferentes com a interferência do ambiente. A mesma folha pode ser classificada em diferentes níveis com a mudança do cenário.

Remover o plano de fundo é muito importante para assegurar uma alta precisão do programa, além de simplificar o código, sendo necessário apenas contar os pixeis cinzas e não mais os amarelos e verdes. Isso também elimina a contagem de pixeis dessas cores no plano de fundo o que prejudicaria a classificação.

Referências

[1] Gressa Chinelato, Tudo sobre oídio e como manejá-lo em sua área, https://blog.aegro.com.br/oidio/, 30 de Julho de 2019.

[2] B. Li, Y. Zhao, Q. Zhu, Z. Zhang, C. Fan, S. Amanullah, P. Gao, F. Luan, Mapping of powdery mildew resistance genes in melon (Cucumis melo L.) by bulked segregant analysis; Scientia Horticulturae, volume 220, 2017, Pages 160-167, ISSN 0304-4238, https://doi.org/10.1016/j.-scienta.2017.04.001.

[3] S.Am, How to define a threshold value to detect only green colour objects in an image with Python OpenCV?, https://stackoverflow.com/questions/47483951/how-to-define-a-threshold-value-to-detect-only-green-colour-objects-in-an-image, 25 de Novembro de 2017.

[4] Wikimedia Foundation, Inc., Powdery mildew, https://en.wikipedia.org/wiki/Powdery_mildew, 18 de Outubro de 2022.

[5] Universo Agrogalaxy, O que é oídio e como tratar?, https://universo.agrogalaxy.com.br/2022/05/30/o-que-e-oidio-como-tratar/, 30 de Maio de 2022.

[6] Aniket Thorat, How to count number of pixels for specific color from Image Using python, https://stackoverflow.com/questions/71469552/how-to-count-number-of-pixels-for-specific-color-from-image-using-python, 14 de Março de 2022.

[7] Aviel Ovadiya, Christoph Rackwitz, Detect if there is gray color in image by opencv Python, https://stackoverflow.com/questions/64038736/detect-if-there-is-gray-color-in-image-by-opencv-python, 9 de Agosto de 2022.

[8] Elaine, Rayryeng, Using OpenCv to detect an image and count some basic color of it, https://stackoverflow.com/questions/61996893/using-opencv-to-detect-an-image-and-count-some-basic-color-of-it, 25 de Maio de 2020.

[9] Bwrr, Christoph Rackwitz, Simple method to extract specific color range from an image in Python?, https://stackoverflow.com/questions/52315895/simple-method-to-extract-specific-color-range-from-an-image-in-python, 23 de Julho de 2022.