Análise comparativa do impacto do vírus sincicial respiratório na célula humana através de técnicas de processamento de imagens

Bianca Aissa Santos
Universidade Estadual Paulista
São José do Rio Preto, Brasil
bianca.aissa@unesp.br

Bruno Vinicius Veronez de Jesus
Universidade Estadual Paulista
São José do Rio Preto, Brasil
bvv.jesus@unesp.br

Daphne Lie Haranaka Pereira
Universidade Estadual Paulista
São José do Rio Preto, Brasil
daphne.pereira@unesp.br

Gustavo Tomaello Juiz
Universidade Estadual Paulista
São José do Rio Preto, Brasil
gustavo.tomaello@unesp.br

***Abstract*—Relatório feito com o objetivo de explicar e analisar os procedimentos de processamento de imagens digitais aplicados ao problema proposta no desafio 01 na disciplina de Processamentos de Imagens Digitais, visando responder as perguntas introduzidas pelo professor.**

***Index Terms*—Processamento de imagem, Histograma, Estatísticas, Vírus sincicial respiratório humano**

I. Introdução

Existe uma linhagem celular (Vero) infectada ou não com o vírus sincicial respiratório humano (hRSV). Foi criado uma sequência de imagens, a partir de padrões previamente estabelecidos, que representam a célula humana exposta ao vírus em um diferentes intervalos de tempo. São sete imagens, que mostram o controle celular (célula sem exposição ao vírus) e outras 6 observações (em intervalos de 21h, 29h, 44h, 53h, 73h, 96h) que mostram o progresso do vírus na célula humana. A exposição de células a vírus pode resultar em diversas mudanças morfológicas e funcionais. Este estudo investiga três hipóteses principais:

- H1: A exposição ao vírus causa mudanças na cor das células ao longo do tempo.
- H2: A exposição ao vírus reduz o número de células ao longo do tempo.

Os objetivos deste estudo são:

- Analisar as mudanças na cor das células ao longo do tempo.
- Quantificar a redução do número de células ao longo do tempo.

II. Bibliotecas e linguagem utilizadas

Para realizar a manipulação das imagens foi utilizado a linguagem Python e algumas de suas bibliotecas e frameworks, como Tensor Flow, Python Imaging Library (PIL), OpenCV, NumPy, Matplotlib, e Skimage.

III. Metodologia

A. Análise Morfológica

Com objetivo de entender as mudanças morfológicas com a presença do vírus na célula humana, foram feitas análise de histograma e aplicados uma série de técnicas pré-processamento.

1) Análise de histograma: Com o objetivo de analisar os tons de cinza, foram utilizadas técnicas de pré-processamento como remoção do fundo e de borda. Após isso, foram feitos os histogramas das imagens fornecidas em todos os momentos, de forma a analisar se essa distribuição possui relação direta com o tempo de exposição das células ao vírus.

a) Filtros de fundo e remoção de borda: O código `filtra_fundo` é uma função de processamento de imagens que isola objetos em uma imagem, removendo o fundo. Primeiramente, a imagem de entrada é convertida para escala de cinza com `cv2.cvtColor`. Em seguida, aplica-se a limiarização de Otsu invertida para criar uma imagem binária onde os objetos são destacados do fundo (`cv2.threshold`). Os contornos dos objetos são então identificados com `cv2.findContours`, e uma máscara de objetos é criada com `cv2.drawContours`. Usando a transformação de distância euclidiana (`ndi.distance_transform_edt`), a função calcula a distância de cada pixel ao objeto mais próximo. Marcadores para a segmentação são gerados com `ndi.label` e, com a técnica de watershed, os objetos são segmentados (`watershed`). Finalmente, uma máscara de primeiro plano é aplicada à imagem original com `cv2.bitwise_and`, resultando na imagem com apenas os objetos, sem o fundo. Este processo é útil quando a distinção clara entre objetos e fundo é necessária.

```
1 def filtra_fundo (image):
2     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
3
```

```
    ret, binary_otsu = cv2.threshold(gray, 0, 255,
    cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    contours, _ = cv2.findContours(binary_otsu, cv2.
    RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    object_mask = np.zeros_like(binary_otsu)
    cv2.drawContours(object_mask, contours, -1, 255,
     thickness=cv2.FILLED)
    distance = ndi.distance_transform_edt(
    object_mask)
    markers = ndi.label(object_mask)[0]
    labels = watershed(-distance, markers, mask=
    object_mask)

    foreground = (labels > 0).astype(np.uint8)

    result = cv2.bitwise_and(image, image, mask=
    foreground)
    return result
```

Listing 1. Extração de características morfológicas utilizando pyhton

Após isso, geramos uma função `remove_fundo` que cria uma máscara binária para distinguir o fundo dos objetos e retorna uma imagem combinada com essa máscara. Primeiramente, a imagem de entrada é convertida para escala de cinza usando `cv2.cvtColor`. Em seguida, uma limiarização simples é aplicada com `cv2.threshold`, utilizando um valor fixo de 10 para separar os pixels claros (prováveis objetos) dos pixels escuros (provável fundo), criando uma imagem binária. A imagem binária resultante é convertida para o tipo `uint8` e usada como máscara. Finalmente, a imagem original é combinada com a máscara binária utilizando `cv2.merge`, resultando em uma imagem que inclui a máscara como um dos canais, permitindo a distinção entre fundo e objetos. Segue abaixo a função:

```
def remove_fundo(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, binary = cv2.threshold(gray, 10, 255, cv2.
    THRESH_BINARY)
    mask = binary.astype(np.uint8)
    result = cv2.merge((image, mask))
    return result
```

Listing 2. Aplicação de fundo transparente

Temos então, o seguinte resultado, a primeira após aplicação da função que filtra o fundo e o próximo da remoção de fundo:

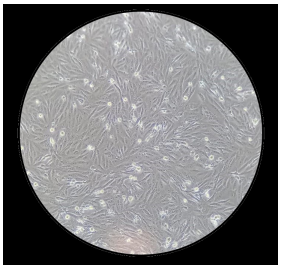


Fig. 1. Imagem com Fundo Preto

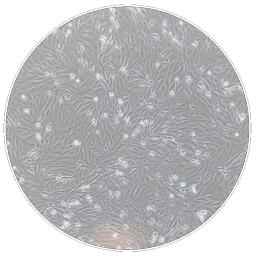


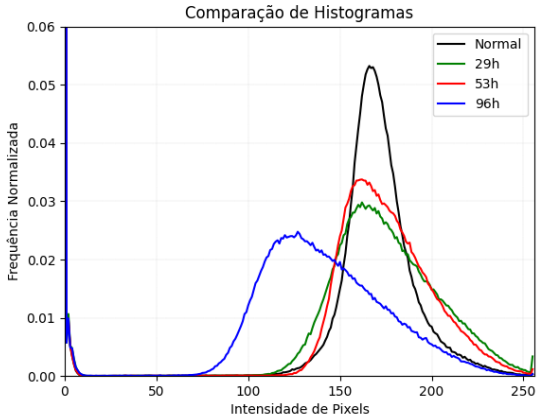
Fig. 2. Imagem Final

Por conseguinte, foram utilizadas as funções e gerados os histogramas referentes ao momento da célula sem exposição ao vírus, e após 29 horas, 53 horas e 96 horas.

```
img01 = cv2.imread("img/original/Normal.jpg")
img01_fundo = filtra_fundo(img01)
img01_final = remove_fundo(img01_fundo)
histo1 = cv2.calcHist([img01_final], [0], None,
    [256], [0, 256])
histograma1 = cv2.normalize(histo1, histo1)
```

Listing 3. Aplicação de fundo transparente

Por fim, foram gerados os histogramas após o pré-processamento, representando a frequência dos níveis de cinza na célula normal, e nos intervalos de 29, 53 e 96 horas:



Para entender melhor essa diferença, foi feita a subtração das frequências da célula normal e após 96h, que é possivelmente próximo a morte celular. Logo, temos o seguinte gráfico representando a a diferença nos níveis de cinza da célula após 96h de exposição e da célula normal. Nota-se um aumento da frequência de pixel de intensidades mais baixas e redução de intensidades mais altas, como mostrado a seguir:

Em segunda análise, podemos observar o formato da distribuição, a relação de média e mediana em momentos distintos (inicialmente - 0 horas e ao final do experimento

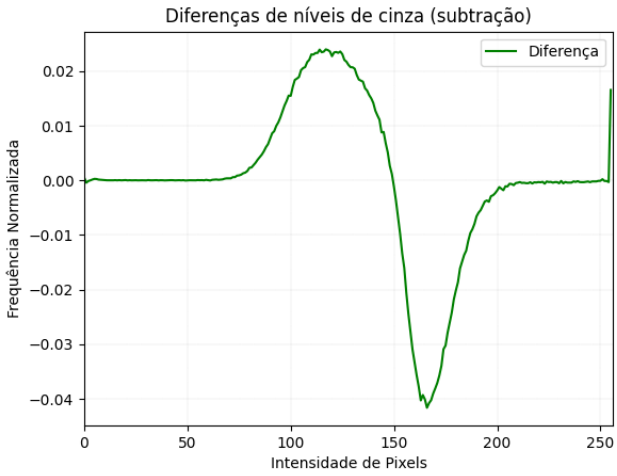
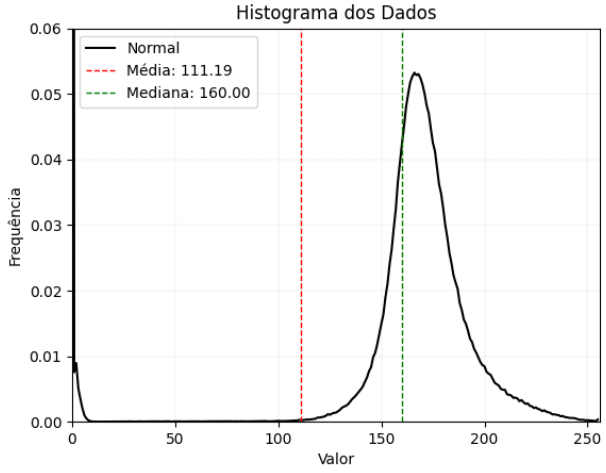
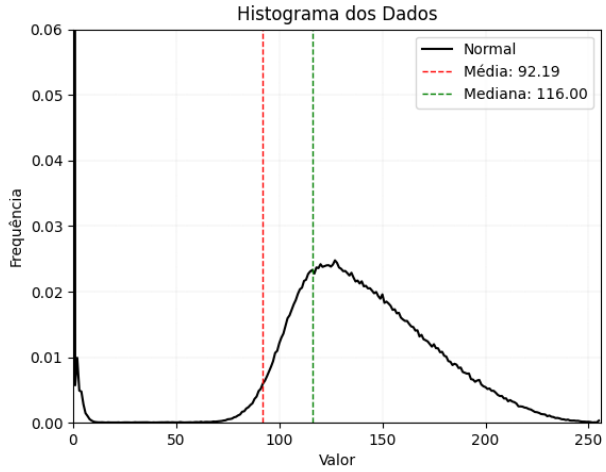


Fig. 3. Diferença de níveis de cinza da célula normal e da célula após 96 horas de exposição

- 96 horas de exposição). No primeiro momento, nota-se uma curtose maior, aproximadamente quase 0.06, que no segundo momento, apresenta pico da curva menor de 0.03. Mostrando uma diferenciação na distribuição dos dados.





2) Diferenças morfológicas: Para entender numericamente as diferenças de intensidades de cores das imagens, foram calculadas medidas estatísticas através do código abaixo:

```
def calcular_estatística(data, intensidades):
    if np.any(data < 0):
    raise ValueError("O array 'data' possui
    elementos negativos, o que não é permitido para
    saber a moda com np.bincount.")

    weighted_mean = np.average(intensidades, weights
    =data)
    weighted_median_value = weighted_median(
    intensidades, data)
    weighted_mode_value = weighted_mode(intensidades
    , data)

    info = {
        'mean': weighted_mean,
        'median': weighted_median_value,
        'std_dev': np.sqrt(np.average((intensidades
    - weighted_mean)**2, weights=data)),
        'variance': np.average((intensidades -
    weighted_mean)**2, weights=data),
        'data_range': np.ptp(intensidades),
        'quartiles': np.percentile(intensidades,
    [25, 50, 75]),
    }
    return info
```

Listing 4. Gráfico de estatística descritiva sobre as observações

A partir do código, geramos informações de estatística descritiva sobre o histogramas das 7 imagens.

	Média	Mediana	Desvio Padrão
0	111.187862	160	83.478576
1	114.607949	161	86.288101
2	113.773168	155	86.754994
3	115.219705	159	87.360845
4	113.827644	157	86.172788
5	110.257321	152	83.586051
6	92.191029	116	72.755144

Os dados do histograma das imagens da placa de Petri, capturadas em intervalos horários após a infecção viral, mostram uma progressão clara da infecção. Inicialmente, a média das intensidades dos pixels permanece relativamente constante, com pequenas variações na mediana e no desvio padrão, sugerindo mudanças graduais na distribuição da intensidade dos pixels. Vemos que a mediana, responsável por dividir o conjunto de dados em dois, tem uma tendência de reduzir seu valor de intensidade de pixel.

No entanto, a partir da sexta imagem, observa-se uma queda acentuada na média e na mediana das intensidades, juntamente com uma redução no desvio padrão. Esta mudança abrupta indica uma significativa alteração na distribuição de intensidade dos pixels, possivelmente causada por um evento crítico na progressão da infecção, como morte celular ou saturação viral.

Esta análise dos histogramas revela que, conforme a infecção avança, há uma modificação notável na distribuição de intensidade dos pixels, refletindo as mudanças morfológicas e estatísticas na cultura celular infectada.

B. Segmentação de núcleos

O código processa uma imagem para identificar e contar objetos distintos. Primeiro, carrega a imagem e a converte para escala de cinza. Em seguida, aplica um filtro Gaussiano para suavização e usa o threshold multi-Otsu para segmentar a imagem em várias classes, mantendo as mais relevantes. Após remover pequenos objetos ruidosos, rotula os componentes conectados na imagem binária e conta os clusters únicos (objetos). Finalmente, exibe a imagem original e a imagem rotulada lado a lado usando matplotlib.

```
import numpy as np
from skimage import filters, measure, io, color,
    morphology
import matplotlib.pyplot as plt

image = io.imread('Normal.jpg')

gray_img = color.rgb2gray(image)

smoothed_img = filters.gaussian(gray_img, sigma=1)

val = filters.threshold_multiotsu(smoothed_img,
    classes=4)
otsu = smoothed_img > val[2]
```

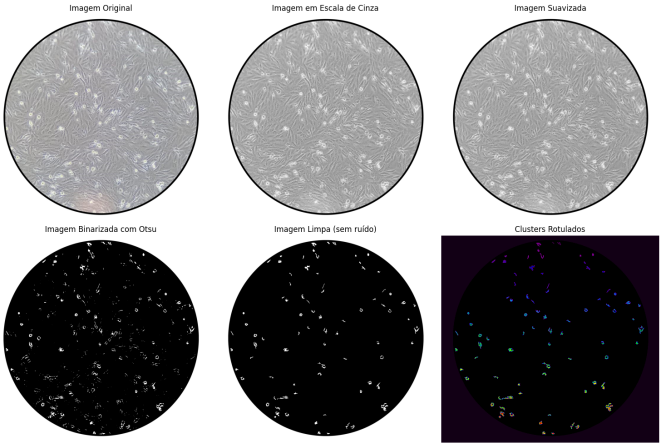


Fig. 4. Contagem de núcleos

```
13
14 cleaned_otsu = morphology.remove_small_objects(otsu,
       min_size=100)
15
16 clusters = measure.label(cleaned_otsu, connectivity
      =2)
17
18 unique_clusters = np.unique(clusters)
19
20 print(len(unique_clusters) - 1)
```

Listing 5. Aplicação de fundo transparente

A função de contagem foi então aplicada a todas imagens, . Obtivemos a maior precisão de contagem apenas em momentos iniciais, como o primeiro, representado abaixo, onde foi contabilizado cerca de 90 núcleos.

IV. Conclusão

Portanto, quanto as hipóteses concluímos que:

- Ocorre mudança morfológica nos tons de cinza com a exposição de da célula ao vírus. Através dos histogramas e da análise de estatística, vemos que a média reduz notavelmente de 111 para aproximadamente 92. Ou seja, há uma tendência de escurimento dos tons de cinza com o tempo de exposição, indicando possível morte celular. Ademais, foi observado redução da frequência de intensidades muito altas, pois há uma redução da curtose.
- Não é possível chegar a uma confirmação da segunda hipótese, pois ocorre a dispersão celular ao longo do tempo e formação de conglomerados que deixam a superfície irregular, dificultando a contagem de núcleos ao longo do tempo.