pratica4

December 10, 2023

[ ]:

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import random
import time
```

[ ]:

```python
def lerVideo(caminho):
    cap = cv2.VideoCapture(caminho)
    frames = []

    if not cap.isOpened():
        raise("Erro abrindo o arquivo!")

    while True:
        ret, frame = cap.read()

        if not ret:
            print("Último frame ou erro")
            break

        frames.append(frame)

        if cv2.waitKey(1) == ord('q'):
            break


    print("Leitura completa")
    cap.release()
    return np.array(frames)
```

[ ]:

```python
#frames = lerVideo('EAFC24.mp4')
```

[ ]:

```python
from matplotlib import cm
from matplotlib.colors import ListedColormap, LinearSegmentedColormap
```

[ ]:

```python
values_list = [0, 1, 2]
colors_list = ['#000000', '#FFFFFF', '#00FF00']
```

```
cmap = ListedColormap(colors_list, name='custom_cmap', N=len(colors_list))
```

[ ]:
```
def reproduzirVideo(frames):
    for frame in frames:
        cv2.imshow("Sem fundo", frame)

        if cv2.waitKey(1) == ord('q'):
            break

        time.sleep(1/30)

    cv2.destroyAllWindows()
```

0.0.1 Filtrando apenas objetos brancos e (roxos)

[ ]:
```
#img_teste = frames[400]

##plt.imshow(img_teste)
#plt.show()
```

[ ]:
```
def onlyWhite(img, sensitivity=90, k_size = 3):
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lower_white = np.array([0, 0, 255 - sensitivity])
    upper_white = np.array([255, sensitivity, 255])

    mask = cv2.inRange(img_hsv, lower_white, upper_white)

    #Openning na imagem
    kernel = np.ones((k_size, k_size))
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

    return mask
```

[ ]:
```
def blueAndRed(img, k_size = 3):
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lower_blue = np.array([80, 140, 0])
    upper_blue = np.array([160, 255, 255])

    lower_red = np.array([130, 90, 100])
    upper_red = np.array([200, 255, 255])

    mask_blue = cv2.inRange(img_hsv, lower_blue, upper_blue)
    mask_red = cv2.inRange(img_hsv, lower_red, upper_red)
    mask = cv2.bitwise_or(mask_blue, mask_red)

    #Openning na imagem
    kernel = np.ones((k_size, k_size))
    #mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
    mask = cv2.dilate(mask, kernel)


    """
    plt.subplot(2, 2, 1)
    plt.imshow(mask_blue)
    plt.subplot(2, 2, 2)
    plt.imshow(mask_red)
    plt.subplot(2, 2 ,3)
    plt.imshow(mask)
    plt.subplot(2, 2 ,4)
    plt.imshow(img)
    plt.show()
    """

    return mask
```

[ ]:

```
#frames= lerVideo('EAFC24.mp4')
```

[ ]:

```
#apenas_brancos = np.array([onlyWhite(frame, 110, k_size=2) for frame in frames])
```

[ ]:

```
#reproduzirVideo(apenas_brancos)
```

Retirando o fundo do vídeo Utilizando a diferença de frame em relação ao anterior

$Foreground = |I_t - I_{t-1}| > T$

[ ]:

```
def difFrames(frames):
    n = frames.shape[0];
    print(n)
    dif = []
    for i in range(n-1, 0, -1):
        #print(i, i-1)
        dif.append(cv2.absdiff(frames[i], frames[i-1]))

    return np.array(dif[::-1])
```

[ ]:
```
#dif = difFrames(frames)
```

0.0.2 Usando a mediana para subtrair o fundo

$Foreground_t = |I_t - B | > T $, onde $B = median(I_1, I_2, ..., I_k)$

[ ]:
```
def difFramesMedian(frames, k, threshold=30):
    newFrames = []
    background_frames = frames[-k:]
    median_frame = np.median(background_frames, axis = 0).astype(np.uint8)


    for i, frame in enumerate(frames):
        abs_diff = cv2.absdiff(frame, median_frame)

        #abs_diff = cv2.cvtColor(abs_diff, cv2.COLOR_BGR2GRAY)

        _, foreground_mask = cv2.threshold(abs_diff, threshold, 255, cv2.
 ↪THRESH_BINARY)

        newFrames.append(foreground_mask)

    return np.array(newFrames)
```

[ ]:
```
#for k in [30, 60, 90, 120]:
#    dif = difFramesMedian(frames, k, 110)
#    reproduzirVideo(dif)
```

[ ]:
```
#dif = difFramesMedian(apenas_brancos, k = 250, threshold = 230)
```

[ ]:
```
#reproduzirVideo(dif)
```

[ ]:
```
def evaluateKmeans(img, k=2):
    image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # reshape the image to a 2D array of pixels and 3 color values (RGB)
    pixel_values = img.reshape((-1, 3))
    # convert to float
    pixel_values = np.float32(pixel_values)

    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

    # number of clusters (K)

    _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.
 ↪KMEANS_RANDOM_CENTERS)
```

```
    # convert back to 8 bit values
    centers = np.uint8(centers)

    # flatten the labels array
    labels = labels.flatten()


    # convert all pixels to the color of the centroids
    segmented_image = centers[labels.flatten()]

    # reshape back to the original image dimension
    segmented_image = segmented_image.reshape(img.shape)
    # show the image
    #plt.imshow(segmented_image)
    #plt.show()
    return segmented_image
```

[ ]:

```
def aplicarKmeanFrames(frames, k =2):
    return np.array([evaluateKmeans(img, k) for img in frames])
```

[ ]:

```
#img = frames[400]
```

[ ]:

```
def recortaCampo(frames):
    new_frames = np.array([img[200: , :, :] for img in frames])
    return new_frames
```

[ ]:

```
def aplicaOppening(frames, kernel):
    return np.array([cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel) for img in frames])
```

0.0.3 Pipeline de procesamento

[ ]:

```
frames_original = lerVideo('EAFC24.mp4')
```

```
Último frame ou erro
Leitura completa
```

Apenas branco

[ ]:

```
frames = recortaCampo(frames_original)



apenas_brancos = np.array([onlyWhite(frame, 110, k_size=2) for frame in frames])
apenas_brancos = aplicaOppening(apenas_brancos, np.ones((4, 4)))
```

Apenas roxo

[ ]:

```
frames = recortaCampo(frames_original)

apenas_roxos = np.array([blueAndRed(frame, k_size = 3) for frame in frames])
apenas_roxos = aplicaOppening(apenas_roxos, np.ones((4, 4)))
```

Aplicando um label

[ ]:

```
apenas_brancos = apenas_brancos/255 * 1
apenas_roxos = apenas_roxos/255 * 2
```

Unindo os dois elementos

[ ]:

```
apenas_roxos = cv2.bitwise_or(apenas_brancos, apenas_roxos)
apenas_roxos = difFrames(apenas_roxos)
apenas_roxos = aplicarKmeanFrames(apenas_roxos)
```

```
982
```

[ ]:

```
reproduzirVideo(apenas_roxos)
```

Salvando resultado em formato .npy

[ ]:

```
np.save(arr= apenas_roxos, file='pratica4_opendepois.npy')
```

[ ]:

```

```