UFPB-CI-DSC
Disciplina: Visão Computacional
Prof. Augusto de Holanda B. M. Tavares
25 de Junho de 2024

3ª Atividade

Manipulações básicas de vídeos e filtragem espacial

O código abaixo mostra como utilizar a biblioteca OpenCV para abrir um vídeo e exibí-lo em uma janela:

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

%matplotlib inline

cap = cv.VideoCapture('corgi_race.mp4')

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
size = (frame_width, frame_height)
result = cv.VideoWriter('filename.avi',
                        cv.VideoWriter_fourcc(*'MJPG'),
                        10, size)

while cap.isOpened():
   ret, frame = cap.read()
   # if frame is read correctly ret is True
   if not ret:
      print("Can't receive frame (stream end?). Exiting ...")
      break

   frame = cv.resize(frame, (640, 480))
   result.write(frame)
   cv.imshow('frame', frame)

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

cap.release()
cv.destroyAllWindows()
```

O código acima carrega algumas bibliotecas e define que o objeto de captura *cap* vem de um arquivo .mp4. Em seguida, é estabelecido um *loop* que determina que enquanto a fonte

de vídeo estiver aberta (reproduzindo) ou o usuário não der uma entrada para sair (neste caso a tecla 'q') algum processo será executado. Este processo verifica que a fonte de fato está fornecendo um quadro de imagem e, em seguida, redimensiona este quadro e o exibe. Ao final do processo a fonte de captura de vídeo é liberada e quaisquer janelas abertas são fechadas.

Utilizando o código acima como base, assim como o que foi explorado em sala de aula, escreva um código que realize as tarefas abaixo. Em todos os casos, mantenha o vídeo em dimensão 640×480 (ou menos), para evitar um custo computacional excessivo.

1. Converta um vídeo colorido em escala de cinza e exiba o resultado.
2. Implemente uma função que calcule a convolução de uma imagem em escala de cinza com uma máscara qualquer de dimensões ímpares. Utilize o preenchimento por zeros.
3. Utilize a função da etapa anterior para borrar o vídeo convertido em escala de cinza.
4. Utilize a função cv.filter2D com um *kernel* apropriado para borrar o mesmo vídeo, e compare o desempenho entre a sua implementação e a implementação do openCV.
5. Utilize a função de convolução implementada para calcular o Laplaciano do vídeo e exiba o resultado. Compare o seu resultado com a função cv.Laplacian().
6. Utilize a função de convolução implementada para calcular o gradiente de Sobel do vídeo. Exiba o resultado e compare com o cálculo do gradiente utilizando os métodos do OpenCV.

Funções de referência

Algumas funções de referência necessárias para implementar as tarefas acima são descritas abaixo:

```
np.pad(img_array,((pad_width,pad_width),),mode='constant',
       constant_values=((255,255),))
```

- Retorna um vetor preenchido com as dimensões *pad_width*. No caso acima, o preenchimento é realizado com as constantes especificadas.

```
np.multiply(a,b)
```

- Retorna o resultado do produto entre os arranjos matriciais a e b, de mesma dimensão.

```
filter2D(src, dst, ddepth, kernel)
```

- Retorna o resultado da convolução da imagem fonte *src* com a máscara *kernel* para o destino *dst*. O intervalo de quantização da saída é definido por *ddepth*. Se $ddepth = -1$ a resolução de intensidade da saída é a mesma da entrada.

```
dst = cv.Laplacian(src_gray, ddepth=-1, ksize=3)
```

- Calcula o Laplaciano de uma imagem em escala de cinza.

```
grad_x = cv.Sobel(gray, ddepth=-1, 1, 0, ksize=3)
 grad_y = cv.Sobel(gray, ddepth=-1, 0, 1, ksize=3)
```

- Calcula os gradientes de Sobel nas direções x e y de uma imagem em escala de cinza. O gradiente completo é dado pela média dos gradientes em x e em y.

```
cv.convertScaleAbs()
```

- Modifica a escala e devolve uma versão da entrada em valores absolutos de 8 *bits*.