

Nome: Leopoldo Kenji Sugata Naves
Nome: Francisco Dênis Bezerra Farias Simão

RA:11201722022
RA: 11053313

Lab_5_Subtração de Fundo e Detecção de Movimento

Experimentos

1- Desenvolva um programa para fazer a leitura de seu vídeo utilizando os vídeos com movimento lento e com movimento rápido, gravados anteriormente.

Para este experimento foi realizada a implementação dos três métodos de subtração apresentados nos exercícios anteriores. Ambos começam com a instalação do objeto “**capture**” que representa o vídeo de entrada e também o objeto “**out**”, que será a gravação da subtração aplicada no vídeo de entrada. Deve-se notar que o vídeo gerado terá a mesma resolução do vídeo de saída.

```
# import the opencv module
import cv2 as cv

# capturing video
capture = cv.VideoCapture("./videos/Leopoldo_rapido.mp4")
#video write
width = capture.get(cv.CAP_PROP_FRAME_WIDTH)    # float
# Get current height of frame
height = capture.get(cv.CAP_PROP_FRAME_HEIGHT) # float
# Define Video Frame Rate in fps
fps = 30
fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('./output/ex1b/Leopoldo_rapido.avi', fourcc, fps, (int(width),int(height)) )
```

Agora será detalhada a implementação de cada método de subtração, começando pelo algoritmo **MOG2**.

Para implementar o algoritmo MOG2 foi criado o objeto **backSub** com a classe do OpenCV “**createBackgroundSubtractorMOG2**”. Este objeto é responsável por gerar uma máscara em escala de cinza, variando de 0 a 255. A região que está mais escura representa o fundo subtraído.

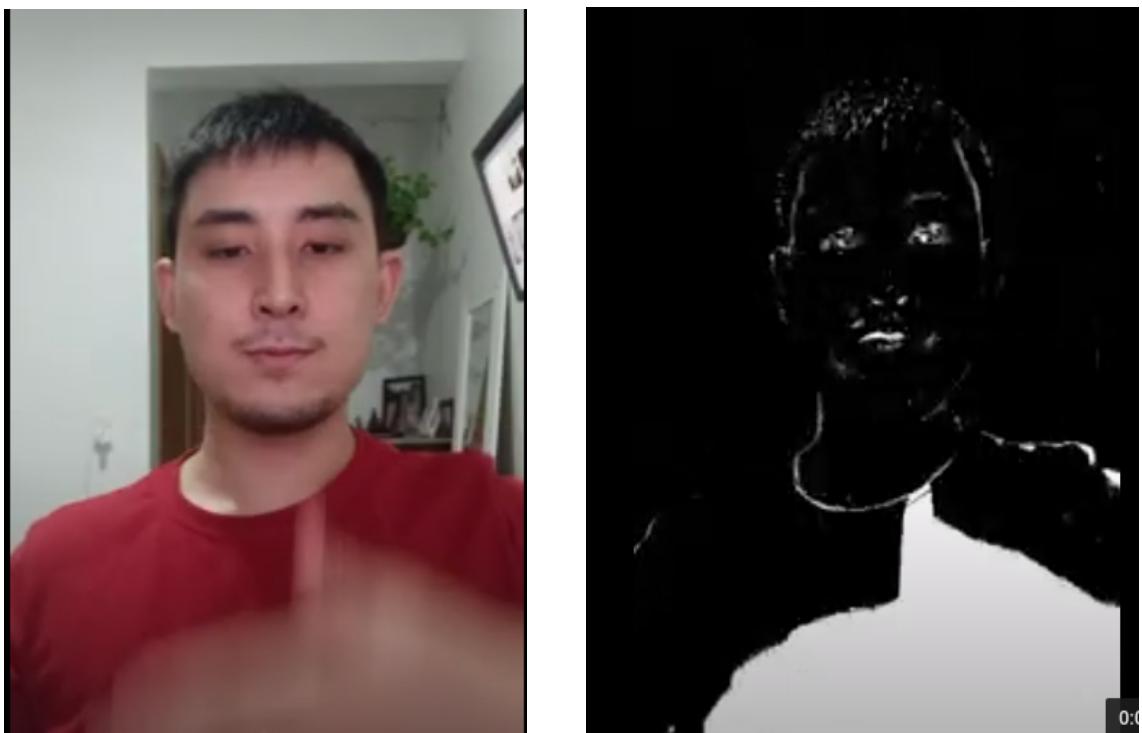
```
backSub = cv.createBackgroundSubtractorMOG2()

while True:
    ret, frame = capture.read()
    if frame is None:
        break

    fgMask = backSub.apply(frame)

    cv.rectangle(frame, (10, 2), (100, 20), (255,255,255), -1)
    cv.putText(frame, str(capture.get(cv.CAP_PROP_POS_FRAMES)), (15, 15),
    cv.FONT_HERSHEY_SIMPLEX, 0.5 , (0,0,0))

    cv.imshow('Frame', frame)
    cv.imshow('FG Mask', fgMask)
    fgMask_rgb = cv.cvtColor(fgMask, cv.COLOR_GRAY2RGB)
    out.write(fgMask_rgb)
```



Figuras 1 e 2. À esquerda, a imagem original e à direita, a imagem com fundo subtraído utilizando o algoritmo MOG2.

O segundo método utilizado foi o **Difference**. Esse é um método de detecção por diferenças de frames, que compara dois frames consecutivos e retorna uma imagem com as diferenças significativas. Desta forma o programa detecta apenas o primeiro plano, em inglês *foreground*, descartando tudo que está estático na imagem. Após a detecção, é feita a conversão para uma imagem em escala de cinza e aplicar um filtro *blur*, para suavizar as formas detectadas.

Após, é feito um processo de binarização da imagem, onde os pixels com valores menores que 20 são transformados em 0, e os demais, em 255.

Com a imagem binarizada, foi aplicada a função **findContours**. Esta função retorna um *array* com as coordenadas de detecção de todas as formas em branco na imagem. Estas formas são os movimentos captados anteriormente pela função **absDiff** (função que retorna a diferença entre dois frames). Com as coordenadas dos contornos, é aplicado um retângulo verde em volta dos contornos na imagem original.

```
# to find contours
contours, hierarchy = cv.findContours(thresh_bin, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

# to draw the bounding box when the motion is detected
for contour in contours:
    x, y, w, h = cv.boundingRect(contour)
    if cv.contourArea(contour) > 300:
        cv.rectangle(img_1, (x, y), (x+w, y+h), (0, 255, 0), 2)
img_1 = cv.flip(img_1, 0)
img_1 = cv.flip(img_1, 1)
out.write(img_1)
# display the output
cv.imshow("Detecting Motion...", img_1)
if cv.waitKey(100) == 13:
    exit()
```



Figuras 3 e 4. À esquerda, a imagem original, e à direita, a mesma imagem, com retângulos verdes indicando o reconhecimento de contornos.

O terceiro método e o mais complexo dentre eles é o método de detecção por meio do *K-nearest neighbors* (KNN). O programa inicia com a instância de um subtrator de fundo que aplica o algoritmo KNN, com a função **createBackgroundSubtractorKNN**. Este algoritmo compara por aproximação os elementos no *foreground* e *background*.

```
mog = cv2.createBackgroundSubtractorKNN()
```

Com o subtrator de fundo instanciado, é possível iniciar a análise do vídeo, frame a frame. Cada frame analisado é convertido para escala de cinza e então aplicado o subtrator, sendo o resultado guardado em uma matriz chamada **fgmask**.

```
while True:  
    ret, frame = cap.read()  
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
  
    fgmask = mog.apply(gray)
```

Similar ao exercício anterior que utilizou um filtro gaussiano para suavizar as bordas da máscara, a metodologia deste experimento utilizou operações morfológicas de erosão e dilatação, para melhorar as bordas e remover “buracos” na imagem.

getStructuringElement - método que cria um kernel, que pode ser de tamanho 3x3, 5x5 ou 7x7.

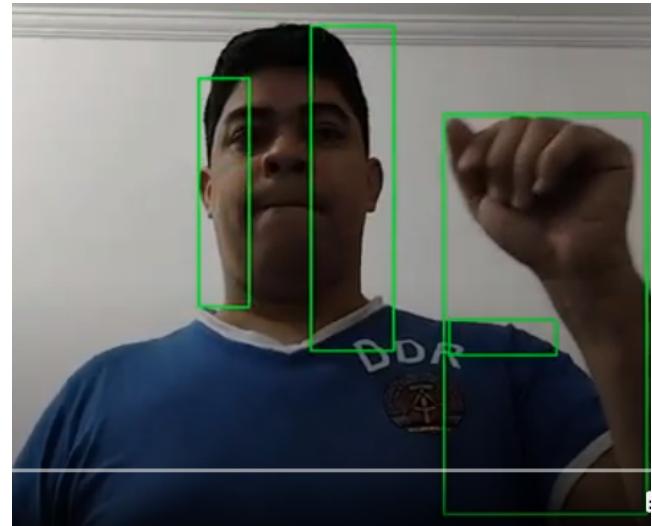
erode - executa erosão na imagem. Ou seja, a erosão é a redução dos limites da imagem em primeiro plano. É um recurso que pode ser útil para dividir objetos e remover extrusões. Um pixel de uma imagem original, seja 0 ou 1, será considerado 1 apenas se todos os pixels sob o kernel também forem 1, caso contrário, é considerado erodido (transformado em 0).

dilate - executa a dilatação na imagem, aumentando a área do objeto, através da convolução de uma imagem com um kernel (obtido pela função **getStructuringElement**). Nesse caso, o princípio é o contrário do erode: um pixel de uma imagem original, seja este 0 ou 1, é considerado 1 se ao menos um dos pixels sob o kernel for igual a 1.

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
fgmask = cv2.erode(fgmask, kernel, iterations=1)
fgmask = cv2.dilate(fgmask, kernel, iterations=1)

contours, hierarchy = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Em resumo, o programa detecta movimento. O vídeo analisado mostra um cenário onde pessoas se movimentam por uma rua. Quando a pessoa está se movendo, o algoritmo detecta esse movimento e “cerca” a pessoa com um retângulo verde. Já a pessoa parada não desperta nenhuma reação do algoritmo.



Figuras 3 e 4. À esquerda, a imagem original, e à direita, a mesma imagem, com retângulos verdes indicando o reconhecimento de contornos.

2- Outro programa modificando o item (1), agora fazendo a leitura de imagem da webcam.Neste caso, o programa deve adicionalmente mostrar uma janela ao vivo com a imagem e o resultado da imagem de rastreamento.



Figuras. À esquerda, a imagem original, no centro, a máscara,e à direita, a mesma imagem, com retângulos verdes indicando o reconhecimento de contornos.

```
#mog = cv2.createBackgroundSubtractorMOG2()
mog = cv2.createBackgroundSubtractorKNN()

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    frame_clear = frame

    fgmask = mog.apply(gray)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
    fgmask = cv2.erode(fgmask, kernel, iterations=1)
    fgmask = cv2.dilate(fgmask, kernel, iterations=1)

    contours, hierarchy = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        # Ignore small contours
        if cv2.contourArea(contour) < 1000:
            continue

        # Draw bounding box around contour
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    #out.write(frame)

    frame = cv2.flip(frame, 1)
    cv2.imshow('Motion Detection', frame)
    out.write(frame)
    out2.write(fgmask)
    out3.write(frame_clear)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```