Visão Computacional - Lista 9

Aqui serão resolvidas as atividades da terceira lista de Visão Computacional pelo aluno Sillas Rocha da Costa, começaremos realizando alguns imports:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

1 - Implementação Câmera Lenta

In [ ]:
```
# 8 frames inseridos a cada par de frames consecutivos do vídeo original
fator = 8

cap = cv2.VideoCapture('./BusterKeaton.mp4')
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

In [ ]:
```
#  criar os vídeos de output
# interpolacao por repeticao
outrep_width = width
outrep_height = height
# interpolacao linear
outlin_width = width
outlin_height = height
# interpolacao por fluxo optico
outopt_width = width
outopt_height = height
# video com os 3 métodos combinados lado a lado
outcomb_width = 3*width
outcomb_height = height

# Criar objeto VideoWriter para salvar os vídeos
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
outrep_path = './output/out_rep.mp4'
outlin_path = './output/out_lin.mp4'
outopt_path = './output/out_opt.mp4'
outcomb_path = './output/out_comb.mp4'

outrep = cv2.VideoWriter(outrep_path, fourcc, fps, (outrep_width, outrep_height))
outlin = cv2.VideoWriter(outlin_path, fourcc, fps, (outlin_width, outlin_height))
outopt = cv2.VideoWriter(outopt_path, fourcc, fps, (outopt_width, outopt_height))
outcomb = cv2.VideoWriter(outcomb_path, fourcc, fps, (outcomb_width, outcomb_height))
```

In [ ]:
```
# Função auxiliar para combinar os frames dos vídeos em um único frame
def combinar_frames(frames):
    # Define as dimensões do novo frame
    height = frames[0][0].shape[0]
    width = frames[0][0].shape[1]
    channels = frames[0][0].shape[2]
    combined_frame = np.zeros((height, width * len(frames), channels), dtype=np.uint8)

    # Combina os frames dos vídeos
```

```
    for i, frame in enumerate(frames):
        combined_frame[:, i * width : (i + 1) * width, :] = frame[0]

    return combined_frame

# mapa das coordenadas x e y (video original) - para o método por fluxo óticocoord_x, coord_y = n
coord_x, coord_y = np.meshgrid(np.arange(width), np.arange(height))

cont_frames = 0; total_frames = cap.get(cv2.CAP_PROP_FRAME_COUNT); bloco = int(total_frames/10)

#rebobinar o vídeo original para o início
cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
ret, prev_frame = cap.read()
while cap.isOpened():
    cont_frames += 1

    #imprimir o progresso do processamento
    if cont_frames % bloco == 0:
        print('Processando: ', int(cont_frames/bloco)*10, '%')
    ret, frame = cap.read()

    if not ret:
        break
    # sequencia comeca com o frame anterior (prev_frame)
    frame_repeat = cv2.resize(prev_frame, (outrep_width, outrep_height))
    frame_linear = cv2.resize(prev_frame, (outlin_width, outlin_height))
    frame_optflow = cv2.resize(prev_frame, (outopt_width, outopt_height))

    frame_combinado = combinar_frames([[frame_repeat], [frame_linear], [frame_optflow]])
    frame_combinado = cv2.resize(frame_combinado, (outcomb_width, outcomb_height))
    # escreve cada frame no video de saida correspondente
    outrep.write(frame_repeat)
    outlin.write(frame_linear)
    outopt.write(frame_optflow)
    outcomb.write(frame_combinado)

    # Efetuar o fluxo ótico
    prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # calcular o fluxo ótico.  Use o método de Farneback já implementado no OpenCV
    flow = cv2.calcOpticalFlowFarneback(prev_gray, gray, None, 0.5, 3, 15, 3, 5, 1.2, 0)

    # inserir frames intermediários
    for i in range(1, fator):
        # Interpolação por repetição
        frame_repeat = prev_frame

        # Interpolação linear
        frame_linear = (prev_frame * (fator - i) + frame * i) / fator

        # Interpolação por fluxo ótico
        map_x = coord_x + flow[..., 0] * (i / fator)
        map_y = coord_y + flow[..., 1] * (i / fator)
        map_x = np.clip(map_x, 0, width - 1).astype(np.float32)
        map_y = np.clip(map_y, 0, height - 1).astype(np.float32)

        frame_optflow = np.zeros_like(prev_frame)
        for j in range(3):
            frame_optflow[..., j] = cv2.remap(prev_frame[..., j], map_x, map_y, interpolation=cv

        frame_optflow = (frame_optflow * (fator - i) + frame * i) / fator
```

```
        # Combina os frames dos vídeos (repetição, linear e fluxo ótico)
        frame_combinado = combinar_frames([[frame_repeat], [frame_linear], [frame_optflow]])
        #frame_combinado = cv2.resize(frame_combinado, (outcomb_width, outcomb_height))
        # escreve cada frame no video de saida correspondente
        outrep.write(frame_repeat.astype(np.uint8))
        outlin.write(frame_linear.astype(np.uint8))
        outopt.write(frame_optflow.astype(np.uint8))
        outcomb.write(frame_combinado.astype(np.uint8))

    prev_frame = frame
```

```
Processando:  10 %
Processando:  20 %
Processando:  30 %
Processando:  40 %
Processando:  50 %
Processando:  60 %
Processando:  70 %
Processando:  80 %
Processando:  90 %
Processando:  100 %
```

In [ ]:
```
cap.release()
```

In [ ]:
```
outrep.release()
```

In [ ]:
```
outlin.release()
```

In [ ]:
```
outopt.release()
```

In [ ]:
```
outcomb.release()
```

2 - Comentários

Repeat:

Apresenta um movimento mais "duro" e travado, talvez por apenas diminuir a transição dos frames, assim, fazendo com que algo que tivesse 30 fps meio que ficasse com 15 fps, ou seja, apenas diminui o tempo de transição dos frames.

Linear:

Apresenta uma transição de frames melhor, pois o novo frame é um meio termo entre dois frames, deste modo, continua com uma transição ao invés de apenas travar o frame, mesmo que ela não seja perfeita e talvez possa possuir borrados, é uma solução melhor que a repeat no sentido de gerar um movimento mais fluido.

Opt Flow:

Bem similar a linear mas até um pouco mais fluida a movimentação.

3 - Fluxo ótico e Pinhole

a) - O fluxo óptico é a diferença da posição dos pixels entre dois frames consecutivos, assim, a projeção de um ponto no plano é tal que:

$$x' = \frac{f \cdot x}{d}$$

Onde x' e x são o ponto projetado e o ponto original, respectivamente. Sabemos que a distância entre dois frames consecutivos é a velocidade da câmera vezes a tempo entre dois frames, assim:

$$\Delta x = v_{cam} * \frac{1}{fps} = \frac{v_{cam}}{60}$$

Por fim, chegamos que a nova projeção de x é:

$$x'_1 = \frac{f \cdot (x + \Delta x)}{d}$$

Assim:

$$of_x = x'_1 - x' = \frac{f \cdot (x + \Delta x)}{d} - \frac{f \cdot x}{d} = \frac{f \cdot \Delta x}{d} = \frac{f \cdot v_{cam}}{60 \cdot d}$$

Portanto:

$$d = \frac{f \cdot v_{cam}}{60 \cdot of_x}$$

b) - Quanto $d \to \infty$, temos por a que $of_x \to 0$, ou seja, o fluxo ótico tende a 0, o que significa que a diferença entre dois frames será praticamente nenhuma, o que corresponde sim a expectativa, já que, quando mais distante o objeto, mais movimento é necessário na câmera para que seja perceptível que o vídeo está em movimento.