Documentação do Projeto: Remoção de Fundo de Vídeos com OpenCV e MediaPipe

Seu Nome

September 28, 2024

1 Introdução

Este projeto tem como objetivo realizar a remoção do fundo de vídeos utilizando técnicas de segmentação semântica com a biblioteca **MediaPipe**, em conjunto com o **OpenCV**para processamento de vídeo. O resultado final preserva o objeto em primeiro plano (geralmente uma pessoa) e substitui o fundo por uma cor sólida (branco, por padrão) ou outro tratamento desejado. Além disso, o vídeo processado é salvo em formato MP4.

2 Tecnologias Utilizadas

- \textbf{OpenCV}: Biblioteca para processamento de imagens e vídeos.
- \textbf{MediaPipe}: Biblioteca do Google que fornece modelos de aprendizado de máquina prontos para segmentação de imagens e vídeos.
- \textbf{NumPy}: Biblioteca para manipulação eficiente de arrays e matrizes numéricas.

3 Instalação e Configuração do Ambiente

3.1 Requisitos de Sistema

- Python 3.x
- Sistema operacional: Windows, Linux, ou macOS
- Ferramentas de linha de comando (ex: Git Bash ou terminal padrão)

3.2 Bibliotecas Necessárias

Para rodar este projeto, você precisa instalar algumas bibliotecas Python. Você pode fazer isso utilizando o **pip**dentro do seu ambiente virtual. Execute os seguintes comandos no terminal:

```
pip install opencv-python
pip install numpy
pip install mediapipe
```

3.3 Estrutura do Projeto

```
MMPG-Background/
|
+-- video/
|   \-- video.mp4               # Vídeo original de entrada
+-- venv/                       # Ambiente virtual (opcional)
+-- main.py                     # Script principal do projeto
+-- .gitignore                  # Arquivos ignorados pelo Git
+-- LICENSE                     # Licença do projeto
\-- README.md                   # Instruções sobre o projeto
```

4 Descrição do Código

4.1 1. Importação das Bibliotecas

O código começa com a importação das bibliotecas necessárias: `cv2`(OpenCV), `mediapipe`para segmentação, e `numpy`para operações de arrays.

```
import cv2
import mediapipe as mp
import numpy as np
```

4.2 2. Inicialização do MediaPipe

Inicializamos o MediaPipe com o modelo de segmentação, que detecta a pessoa no vídeo.

```
mp_selfie_segmentation = mp.solutions.selfie_segmentation
segmentation =
↪  mp_selfie_segmentation.SelfieSegmentation(model_selection=1)
```

4.3 3. Definição dos Caminhos de Arquivos

Definimos os caminhos do vídeo de entrada e de saída. O vídeo de entrada deve estar localizado no caminho especificado:

```
video_path = r'C:\\Caminho\\video.mp4'
output_path = r'C:\\Caminho\\video_processado.mp4'
```

4.4 4. Leitura e Escrita de Vídeos

O OpenCV é usado para ler o vídeo (`cv2.VideoCapture`) e configurar a escrita do vídeo processado (`cv2.VideoWriter`). O vídeo de saída mantém a resolução e o FPS (frames por segundo) do vídeo original.

```
cap = cv2.VideoCapture(video_path)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
```

4.5 5. Loop de Processamento de Frames

Cada frame do vídeo é lido e processado em um loop. Para cada frame:

- Ele é convertido para o formato RGB, necessário pelo MediaPipe.
- A segmentação de fundo é realizada.
- O fundo é substituído por uma cor sólida (branco por padrão).

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret: break
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = segmentation.process(frame_rgb)
    mask = result.segmentation_mask > 0.5
    mask_3d = np.dstack((mask, mask, mask))
    person_segmented = np.where(mask_3d, frame, BACKGROUND_COLOR)
    person_segmented = person_segmented.astype(np.uint8)
    out.write(person_segmented)
```

```
    cv2.imshow('Fundo Removido com Cores Preservadas',
    ↪  person_segmented)
    if cv2.waitKey(30) & 0xFF == ord('q'): break
```

4.6 6. Finalização e Liberação de Recursos

Ao final do processamento, o vídeo de saída é liberado e todas as janelas são fechadas:

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

5 Opções de Customização

- \textbf{Cor do Fundo}: A cor de fundo padrão é branco (255, 255, 255). Você pode alterar isso ajustando a variável BACKGROUND_COLOR.
- \textbf{Fundo Transparente}: Se desejar, pode implementar um fundo transparente usando máscaras alfa.
- \textbf{Alteração de Resolução e FPS}: É possível alterar as propriedades do vídeo de saída definindo os parâmetros de resolução e taxa de quadros no cv2.VideoWriter.

6 Melhorias Futuras

- \textbf{Aplicação de IA Avançada}: Você pode explorar o uso de redes neurais mais robustas para obter segmentações mais precisas, especialmente em vídeos com cenários complexos.
- \textbf{Integração com GUI}: Adicionar uma interface gráfica (usando Tkinter ou PyQt) pode facilitar o uso do programa por usuários não técnicos.
- \textbf{Substituição de Fundo por Imagem/Outro Vídeo}: Pode-se substituir o fundo removido por uma imagem ou outro vídeo, simulando um efeito de 'chroma key'.

7 Erros Comuns e Soluções

- \textbf{Erro ao carregar o vídeo}: Certifique-se de que o caminho para o vídeo de entrada esteja correto e o arquivo exista no local especificado.
- \textbf{Problemas de exibição de vídeo (cv2.imshow)}: O OpenCV pode gerar erros ao exibir vídeos dependendo do formato. A conversão para uint8 é crucial para evitar problemas de exibição.

8 Licença

Este projeto está licenciado sob a Licença MIT - consulte o arquivo LICENSE para mais detalhes.