Projeto de Reconhecimento de Sinais em Libras

Este projeto realiza o reconhecimento de gestos em Libras, utilizando visão computacional para capturar, processar e classificar gestos de mãos com landmarks. A estrutura inclui coleta de imagens, criação de dataset, treinamento de um modelo de classificação e inferência em tempo real.

Estrutura de Arquivos

1. collect_imgs.py - Coleta de imagens de gestos para cada classe, criando um dataset de imagens.
2. create_dataset.py - Processamento de imagens para extrair landmarks das mãos e calcular distâncias entre elas, gerando um conjunto de dados serializado.
3. train_classifier.py - Treinamento de um classificador Random Forest para reconhecer os gestos com base nas distâncias calculadas entre landmarks.
4. inference_classifier.py - Inferência em tempo real para classificar gestos a partir de uma câmera, utilizando o modelo treinado.
5. interface.py - Interface gráfica com `tkinter` para realizar inferência em tempo real de forma mais intuitiva e visualmente atrativa.

Bibliotecas Utilizadas

1. OpenCV (`cv2`): Utilizado para captura e manipulação de imagens em tempo real. Captura quadros da câmera, adiciona anotações em imagens, converte entre espaços de cores e exibe imagens para o usuário.

2. Mediapipe: Biblioteca desenvolvida pelo Google para reconhecimento e rastreamento de mãos, fornecendo coordenadas precisas de landmarks das mãos. Permite a extração de pontos de referência que são usados para identificar gestos.

3. Scikit-Learn: Utilizado para o treinamento e avaliação do modelo. Inclui métodos para dividir o dataset, calcular métricas de acurácia e treinar modelos de aprendizado de máquina, como Random Forest.

4. Numpy: Utilizado para manipulação de arrays numéricos e cálculos matemáticos, como cálculo de distâncias euclidianas entre landmarks.

5. Pickle: Permite salvar e carregar dados e modelos de forma serializada. Usado para armazenar o dataset processado e o modelo treinado.

6. Matplotlib (apenas `create_dataset.py`): Usado para visualizar landmarks nas imagens, embora seja opcional. Pode ajudar a verificar visualmente as landmarks detectadas.

7. Tkinter: Biblioteca padrão do Python para criar interfaces gráficas. Utilizada para criar uma interface de usuário que exibe a câmera em tempo real e a predição dos gestos detectados.

Arquivo `interface.py`

A interface em `interface.py` permite uma experiência de usuário mais intuitiva e visualmente agradável para a inferência em tempo real. Ela captura o vídeo da câmera, exibe o gesto detectado em tempo real, e exibe a letra do gesto previsto.

Principais Elementos:

- calculate_3d_distances: Função que calcula as distâncias 3D entre landmarks das mãos para serem usadas pelo modelo.
- RealTimeInferenceApp: Classe que define a interface de usuário com `tkinter`.
- update_frame: Função que atualiza o vídeo da câmera, processa landmarks e faz predições para cada gesto detectado.

Aprimoramentos Visuais:

- Fundo personalizado em tons escuros para conforto visual.
- Instruções claras para o usuário ("Mova a mão para que o modelo detecte o gesto!") para facilitar o uso.
- Fontes e cores personalizadas para tornar a interface mais agradável e moderna.

Resumo do Processo Completo

1. Coleta de Dados - `collect_imgs.py`: Captura imagens para cada classe de gesto e salvar em pastas separadas.
2. Criação do Dataset - `create_dataset.py`: Processa as imagens, extrai landmarks e calcula distâncias entre elas.
3. Treinamento - `train_classifier.py`: Treina um modelo Random Forest com os dados de distâncias e rótulos.
4. Inferência - `interface.py`: Realiza inferência em tempo real a partir de uma câmera, exibindo a classe prevista como letras.

Essa interface adiciona uma camada visual que torna o projeto acessível e fácil de usar. O projeto pode ser expandido para incluir mais classes e ajustes para melhorar a precisão do modelo.