**Web-Based Pose Detection and Classification System with Voice Activation**

**Introduction**

This project introduces a locally hosted, interactive system that combines real-time pose detection with speech recognition. Using MediaPipe for pose detection and Python's SpeechRecognition library, the system identifies specific human poses and responds to the keyword "LOVE," triggering predefined actions. This setup demonstrates the seamless integration of computer vision and audio processing, functioning entirely offline.

**Table of Contents**

* [Introduction](#introduction)
* [Project Overview](#project-overview)
* [Motivation](#motivation)
* [Methodology](#methodology)
* [Experiment](#experiment)
* [Results and Discussion](#results-and-discussion)
* [Conclusion](#conclusion)
* [Data Sources](#data-sources)
* [Installation and Usage](#installation-and-usage)
* [Contributing](#contributing)
* [License](#license)
* [Contact](#contact)

**Project Overview**

The system leverages MediaPipe for real-time pose detection and Python's SpeechRecognition library to enhance interactivity through verbal commands. The application highlights the potential of using pre-trained models for creating interactive systems that respond to both physical and verbal cues.

**Motivation**

This project explores the integration of AI and ML technologies to create an interactive system that enhances user experience through real-time pose detection and speech recognition. The aim is to demonstrate the practical application of these technologies in creating systems that respond to both visual and auditory inputs.

**Methodology**

The methodology integrates two primary components: the MediaPipe framework for pose detection and the Python SpeechRecognition library for auditory command processing. This integration is designed to run locally, ensuring robust performance without reliance on internet connectivity.

**System Workflow**

1. **Initialization:** Initializes both the pose detection and speech recognition components.
2. **Real-time Processing:** Processes video frames and audio input in parallel.
3. **Triggering the Special Effect:** Triggers a visual effect when both conditions are met.
4. **Running on Localhost:** The application runs locally, ensuring accessibility and ease of use.

**Experiment**

The project utilizes pre-trained models and Python libraries to create a comprehensive system for real-time pose detection and speech recognition. The focus is on effective integration and utilization of these models to build an interactive system that operates entirely offline.

**Tools and Libraries**

* **MediaPipe:** For comprehensive pose detection capabilities.
* **SpeechRecognition:** For converting speech into text.
* **OpenCV (cv2):** For capturing and processing video feed.
* **NumPy:** For mathematical operations and pose analysis.
* **PyAudio:** For audio input support.

**Results and Discussion**

The system successfully detects human poses and recognizes the keyword "LOVE" to trigger responses. Running on localhost, it demonstrates the feasibility of combining computer vision and audio processing in real-time.

**Performance Evaluation**

* **Pose Detection:** High accuracy with MediaPipe.
* **Speech Recognition:** Responsive and accurate under various conditions.

**Challenges and Limitations**

* **Latency:** Minor delays observed between pose detection and speech recognition.
* **Environmental Conditions:** Variations in lighting and background noise affected accuracy.

**Conclusion**

The project demonstrates the integration of computer vision and speech recognition technologies to create an interactive user experience. Future improvements aim at performance optimization and user interface development.

**Data Sources**

* **MediaPipe Pose Model:** For pose detection.
* **SpeechRecognition Library:** For speech recognition.

**Installation and Usage**

1. **Clone the repository:**

bash

Copy code

git clone https://github.com/yourusername/Pose-Detection-Voice-Activation.git

cd Pose-Detection-Voice-Activation

1. **Set up the environment:**
   * Create and activate a virtual environment:

bash

Copy code

python -m venv venv

source venv/bin/activate # On Windows use `venv\Scripts\activate`

1. **Install the required dependencies:**

bash

Copy code

pip install -r requirements.txt

1. **Run the application:**

bash

Copy code

python app.py

**Contributing**

We welcome contributions to improve this project. To contribute, please follow these steps:

1. Fork the repository.
2. Create a new branch (git checkout -b feature-branch).
3. Make your changes and commit them (git commit -m 'Add new feature').
4. Push to the branch (git push origin feature-branch).
5. Create a new Pull Request.

**License**

This project is licensed under the MIT License. See the LICENSE file for details.

**Contact**

For any questions or feedback, please contact:

* **Emmanuel Nwonye**
* Email: e.nwonye0320231@arts.ac.uk
* LinkedIn: [linkedin.com/in/emmanuel-nwonye-40a024183](https://www.linkedin.com/in/emmanuel-nwonye-40a024183/)