WENZHOU-KEAN UNIVERSITY

Wenzhou-Kean University

Software Requirements Specifications Document

CPS 4951 W01: SENIOR CAPSTONE

Student Name&ID:	Keming Xing-1162866 Yuchao Xi-1162956 Hantao Xie-1162962 Fucheng Xie-1162814
Project Title:	Pen Grip Pose Detection based on MideaPipe
Instructor:	Pinata Winoto
Date:	February 24, 2024

Spring Semester-2024

List of contents

1. Introduction

1.1 Purpose

The main goal of the project is to develop a system for detecting and analyzing handwriting gestures using MediaPipe, an open source framework for building machine learning solutions. The scope of the project includes:

- Implementing hand detection and tracking using MediaPipe's Hand Tracking solution to locate and track the position and movement of the hand.
- Developing a model for recognizing different handwriting postures, such as pen grip variations and hand orientations.
- Integrating real-time feedback to provide users with guidance on improving their handwriting posture.
- Creating a user-friendly interface for visualizing and analyzing handwriting posture data.

1.2 Scope

The project aims to assist individuals, especially child and professionals who frequently write or draw, in improving their handwriting techniques by providing actionable insights into their posture and grip.

1.3 Definitions, Acronyms, and Abbreviations

MediaPipe is an open-source framework developed by Google for building applications involving perception and machine learning. It

provides pre-built models and modules, enabling developers to easily integrate real-time computer vision functionalities such as hand tracking, face detection, and pose estimation. Renowned for its efficiency in real-time processing, MediaPipe is well-suited for applications requiring quick and accurate analysis of visual data. The framework features a modular and flexible architecture, allowing developers to combine different components to create custom solutions tailored to their specific needs. In the provided proposal overview, MediaPipe is employed for hand detection and tracking, showcasing its capabilities in developing a handwriting posture detection system.

1.4 References

[1] Blackburn-Brockman, E. (2001). Prewriting, planning, and professional communication. The English Journal, 91(2), 51. https://doi.org/10.2307/822345 .

[2] Nagaraju, Mr. M. (2022). Digital handwriting recognition using hand tracking by using media pipe and OPENCV libraries. International Journal for Research in Applied Science and Engineering Technology, 10(7), 659–666. https://doi.org/10.22214/ijraset.2022.44647

[3] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., & Grundmann, M. (2020, June 18). MediaPipe hands:

On-device real-time hand tracking. arXiv.org. https://arxiv.org/abs/2006.10214

[4] Tao, W., Leu, M. C., & Yin, Z. (2018). American sign language alphabet recognition using convolutional neural networks with Multiview Augmentation and inference fusion. Engineering Applications of Artificial Intelligence, 76, 202–213. https://doi.org/10.1016/j.engappai.2018.09.006.

1.5 Overview

The core objective of this project is to develop a sophisticated system utilizing MediaPipe, an open-source framework, for the detection and analysis of handwriting gestures. Key components of the project include hand detection and tracking, posture recognition modeling, real-time feedback integration, and a user-friendly interface for data visualization. Targeted towards individuals, especially children and professionals, who seek to improve their handwriting techniques, the system aims to provide actionable insights into posture and grip. Inspired by challenges faced by learners in Hong Kong, the project addresses the crucial role of handwriting posture in writing legibility, speed, and comfort. Supported by relevant case studies leveraging MediaPipe, the project will culminate in deliverables such as a fully functional system, comprehensive documentation, a research report, and open-source release of source code

and model weights. These outcomes aim to empower stakeholders for further research and development in handwriting posture detection.

2. The Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

a. Name: MediaPipe Hand Tracking API

Fuction: API for recognizing hand gestures.

Use: Analyzing hand images

b. Name: Camera Interface API

Fuction: API for interacting with the device camera.

Use: Used to capture hand images for processing by Mediapipe.

c. Name: Voice module interface

Fuction: API for interacting with the device camera.

Use: Used to capture hand images for processing by Mediapipe.

d. Name: Data preprocessing model

Fuction: Preprocess images

Use: Preprocess the image and convert the format into an array form that can be processed by the mediapipe model.

e. Name: Posture coefficient acquisition model

Fuction: Get coordinates and calculate results

Use: Obtain the three-dimensional coordinates of the key points of the hand and calculate the hand posture results based on the coordinates

f. Name: Posture judgment model

Fuction: Get pen holding posture judgment results

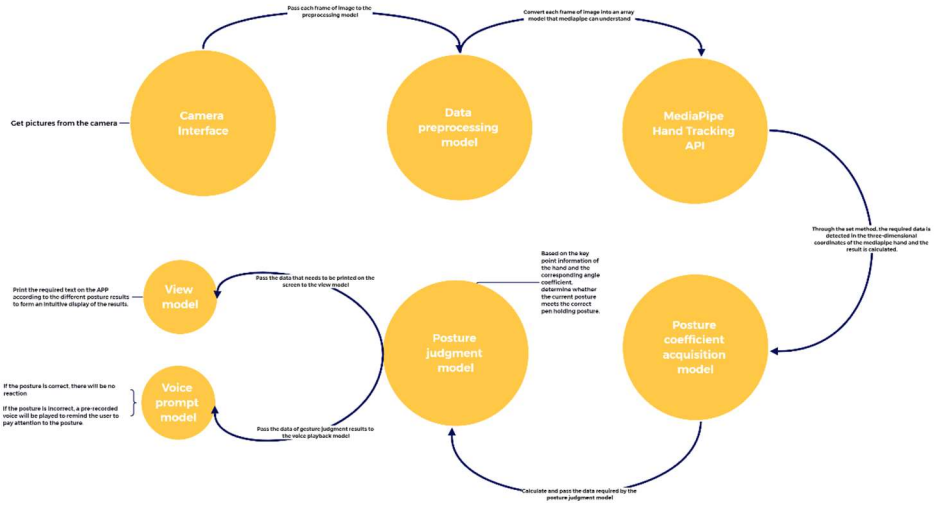
Use: Based on the key point information of the hand and the corresponding angle coefficient, determine whether the current posture meets the correct pen holding posture.

g. Name: View model

Fuction: Display view

Use: Print the required text on the APP according to the different posture results to form an intuitive display of the results.

The relationship between them:



2.1.2 Interfaces

Reference 2.1.1

2.1.3 Hardware Interfaces

This project provides the author with two ways to develop programs: Windows system or Android system:

CPU usage rate <= 50%.

Memory usage rate <= 50%.

Video analysis frame rate >= 30 FPS.

In 95% of cases, the response time during regular periods should not exceed 1.5 seconds, and during peak periods, it should not exceed 4 seconds.

The time required for the location system to display the first interface from the click should not exceed 300 milliseconds.

In a well-connected network, the time required to establish a GPRS network connection should not exceed 5 seconds.

In a well-connected network, the electronic map refresh time should not exceed 10 seconds.

In the recommended configuration environment: login response time within 2 seconds, refresh column response time within 2 seconds, refresh entry pagination list response time within 2 seconds, open information entry response time within 1 second, refresh department and personnel list response time within 2 seconds.

During non-peak hours, specific searches based on number and name conditions should yield results within 3 seconds.

The system should support Android and Windows operating systems.

The system should support Oracle and DB2 database systems.

Only a maximum of 5% of system implementations need to be specific to a particular operating system.

The average time to replace a relational database system should not exceed 2 hours, and data loss must be guaranteed not to occur.

2.1.4 Software Interfaces

The following is a description of the non-functional requirements for the software:

1. Strict Access Control: Users, after authentication, can only access data within their authorized scope and perform operations permitted by their assigned permissions.

2. Differentiated User Identities and Permissions: Users with distinct identities and permissions require a trustworthy authorization management service, ensuring data protection against unauthorized access, tampering, and maintaining data confidentiality and integrity, contingent upon the real and trustworthy identity of users.

3. Run Log Management and Security Auditing: Provide functions for managing runtime logs and security audits, allowing tracking of the historical usage of the system.

4. Resilience Against General Malicious Attacks: The system should withstand general malicious attacks from the internet, including viruses (including trojans), password guessing attacks, and hacking attempts. Detection for at least 99% of attacks should occur within 10 seconds.

5. Input Prompt and Data Validation: Provide input prompts and data checks to prevent data anomalies.

6. Strong System Robustness: The system should demonstrate robustness by correctly handling various abnormal situations during its operation, such as human-operated errors, input of illegal data, and hardware device failures.

7. Low Probability of Business Disruption: The probability of business disruption due to software system failure should be less than 5‰.

8. Encrypted Data Transmission: Network-transmitted data should undergo encryption to ensure data confidentiality, preventing eavesdropping, theft, or tampering during data collection, transmission, and processing. Business data should be encrypted during storage to ensure its inviolability.

9. System Availability in Low Configuration: Ensure system availability even under poor hardware capabilities.

10. Fast System Restart: In the event of a failure, the system should restart within a maximum of 20 seconds in 95% of cases.

11. Data Backup and Recovery: Provide data backup and recovery functions, enabling timely restoration and recovery of data in case of system errors or data loss due to other reasons, facilitated by hardware and third-party software.

12. Unit Testing and Regression Testing: The delivered system must pass unit testing with 100% coverage. Development activities must utilize regression testing, allowing a complete retest within 12 hours.

2.1.5 Communications Interfaces

This project does not use communication protocols, but uses internal database test sets and self-built models as the basis for judgment. The output and completion are all in the program.

2.1.6 Memory Constraints

CPU usage rate <= 50%.

Memory usage rate <= 50%.

2.1.7 Operations

First, this is the most core process of the application, and the subsequent requirements can be iteratively developed.

i. Startup Procedure:

- Upon launching the application, the user should be presented with the main user interface.
- The application should initialize the webcam or camera for capturing live video input.

- Necessary resources and models for hand detection and pen grip recognition should be loaded into memory.

ii. Normal Operation:

- The application should continuously capture video frames from the webcam and perform real-time hand detection using MediaPipe.
- Detected hand positions should be analyzed to recognize various pen grip postures.
- Real-time feedback regarding the detected pen grip should be displayed to the user through the user interface.
- Users should be able to interact with the application to adjust settings or pause/resume detection as needed.

iii. Shutdown Procedure:

- Upon closing the application, the webcam or camera should be released, and all resources should be properly cleaned up.
- Any temporary files or logs generated during the session should be appropriately handled or saved, if necessary.

iv. Maintenance Tasks:

- Regular updates and maintenance should be performed to ensure compatibility with new versions of MediaPipe or Python libraries.

- Users should be provided with instructions for troubleshooting common issues and contacting support if needed.
- Any changes to hardware or system configurations that may affect application performance should be documented and communicated to users.

v. Support Procedures:

- Users should have access to online documentation and tutorials for using the application effectively.
- A support channel, such as an email address or forum, should be available for users to seek assistance or report bugs.
- Feedback from users should be collected and considered for future updates and improvements to the application.

2.1.8 Site Adaptation Requirements

Python cross-platform compilation refers to the conversion of Python code into executable files that can run on different platforms and operating systems. This allows developers to develop, test, and build applications on their own local computers, and then deploy those applications to a variety of different platforms and operating systems without having to repeatedly write and test code. Users do not need to download python on their local computers, and can use python packaging programs such as Pyinstaller to create executable applications

2.2 Product Functions

i. Real-time Hand Detection:

- The application should be able to detect and track the position of the hand in real-time using the MediaPipe hand tracking solution.
- Hand detection should be robust and accurate, even under varying lighting conditions and hand orientations.

ii. Pen Grip Recognition:

- The Pen Grip Recognition algorithm is designed or improved by using the results of mediapipe's handdetection
- Various grip styles, including tripod grip, dynamic tripod grip, and lateral tripod grip, should be distinguished and analyzed (optional).

iii. User Interface (UI):

- The application should feature an intuitive and user-friendly interface for displaying live video feed and detected pen grip postures.
- The UI should provide clear visual feedback to the user regarding their current pen grip posture.
- Interactive elements such as buttons or sliders should allow users to adjust settings or interact with the application easily.

iv. Feedback Mechanism:

- Real-time feedback should be provided to users to guide them in correcting their pen grip posture.
- Feedback could include visual cues, such as highlighting incorrect grip postures or displaying textual instructions for improvement.

v. Customization Options:

- Users should have the ability to customize feedback settings, such as the sensitivity of grip detection or the type of feedback provided.
- Settings for adjusting camera resolution, frame rate, or other parameters should be accessible to users for optimal performance.

vi. Data Logging and Analysis:

- The application should include functionality for logging pen grip data, including timestamps and detected grip postures.
- Logged data should be available for further analysis, allowing users to track their progress and identify areas for improvement over time.

vii. Export Functionality:

- Users should be able to export logged data or generated reports for offline analysis or sharing with others.

- Exported data should be compatible with common file formats for easy integration with other applications or platforms.

viii. Accessibility Features:

- The application should adhere to accessibility standards, ensuring that it can be used by individuals with disabilities.
- Features such as keyboard shortcuts or screen reader compatibility should be implemented to enhance accessibility.

2.3 User Characteristics

The desktop program is intended for young schoolchildren and others who need to write for long periods of time. Most of the users are the children of middle and low production families, with small learning space, insufficient school supplies, and no supervision and guidance

2.4 Constraints

The app is a desktop application that requires users to have medium performance computer, resolution 720p or more cameram, and bright environment.

2.5 Assumptions and Dependencies

i. Assumption: MediaPipe library provides accurate hand detection and tracking functionalities.

Dependency: The accuracy and reliability of hand detection and tracking heavily rely on the performance of MediaPipe library.

ii. Assumption: The desktop application will have access to a webcam or camera for real-time hand detection.

Dependency: The availability and proper functioning of a webcam or camera are necessary for capturing live video input.

iii. Assumption: Users have basic computer literacy and can install and run Python-based applications on their desktops.

Dependency: Users must have access to compatible hardware and software environments to install and run the application.

2.6 Apportioning of Requirements

i. Core Requirements (Phase 1):

A. Real-time Hand Detection: Implement real-time hand detection using MediaPipe to locate the position of the hand in the video stream.

B. Pen Grip Recognition: Develop a model for recognizing various pen grip postures using the detected hand positions.

C. User Interface (UI): Create a simple and intuitive user interface to display live video feed and detected pen grip postures.

D. Feedback Mechanism: Provide real-time feedback to users regarding their pen grip posture, such as visual cues or textual instructions.

ii. Additional Features (Phase 2):

A. Data Logging: Implement functionality to log pen grip data, including timestamps and detected grip postures, for further analysis.

B. Performance Optimization: Optimize the application for performance and resource usage to ensure smooth operation on various hardware configurations.

C. Customization Options: Allow users to customize feedback settings or adjust detection sensitivity according to their preferences.

D. Export Functionality: Include the ability to export logged data or generated reports for offline analysis or sharing.

iii. Future Enhancements (Phase 3):

A. Multi-camera Support: Add support for multiple cameras to enable simultaneous hand detection and analysis from different angles.

B. Handwriting Analysis: Expand the application to analyze handwriting quality and provide suggestions for improvement based on grip posture.

C. Integration with External Devices: Integrate with external devices, such as stylus pens or interactive whiteboards, for enhanced input and interaction capabilities.

D. Localization and Internationalization: Support multiple languages and cultural preferences to make the application accessible to a wider audience.

3. Specific Requirements

3.1 External Interfaces

The application will have the following external interfaces:

i. User Interface

The application will have a graphical user interface (GUI) that will display the following:

- Live video feed from the webcam or camera
- Detected hand positions and recognized pen grip postures
- Real-time feedback on pen grip posture
- Settings and customization options

ii. Camera Interface

The application will interface with the system's webcam or camera to capture live video input for hand detection and analysis.

iii. File Interface

The application will support importing and exporting data files in common formats (e.g., CSV, JSON) for logging and sharing pen grip data and analysis reports.

3.2 Functions

The application will provide the following core functions:

i. Real-time Hand Detection and Tracking

Detect and track the position and movement of the user's hand in real-time using the MediaPipe Hand Tracking solution.

ii. Pen Grip Recognition

Analyze the detected hand positions and recognize various pen grip postures, such as tripod grip, dynamic tripod grip, and lateral tripod grip.

iii. Real-time Feedback

Provide real-time visual feedback and instructions to the user regarding their current pen grip posture and suggestions for improvement.

iv. Data Logging

Log pen grip data, including timestamps and detected grip postures, for further analysis and progress tracking.

v. Customization

Allow users to customize feedback settings, detection sensitivity, and other application preferences.

vi. Export Functionality

Enable users to export logged data and analysis reports in common file formats for offline analysis or sharing.

3.3 Performance Requirements

i. Hand Detection and Tracking Accuracy

The hand detection and tracking algorithm should achieve an accuracy of at least 90% under optimal lighting conditions and hand orientations.

ii. Real-time Performance

The application should be capable of processing and analyzing video frames at a minimum rate of 30 frames per second (FPS) on a system with recommended hardware specifications.

iii. Responsiveness

The application should provide real-time feedback and instructions to the user within a maximum latency of 200 milliseconds.

iv. Resource Usage

The application should have a maximum CPU and memory usage of 50% on a system with recommended hardware specifications.

3.4 Logical Database Requirements

The application will not use a traditional database system. Instead, it will rely on the following data structures and storage mechanisms:

i. Hand Pose Data

The application will store and process hand pose data, including the positions of key hand landmarks and joint angles, for pen grip recognition and analysis.

ii. Logged Data

The application will store logged pen grip data, including timestamps and detected grip postures, in local files or in-memory data structures for subsequent analysis and reporting.

iii. Configuration Settings

The application will store user preferences and customization settings in local configuration files or in-memory data structures.

3.5 Design Constraints

3.5.1 Standards Compliance

The application will comply with the following standards and guidelines:

- Accessibility guidelines (e.g., Web Content Accessibility Guidelines (WCAG) 2.1) to ensure usability for individuals with disabilities.
- User interface design guidelines for the target operating system (e.g., Material Design for Android, Human Interface Guidelines for macOS).
- Data privacy and security best practices for handling and storing user data.

3.6 Software System Attributes

3.6.1 Reliability

The application should have a failure rate of less than 5% during normal operation and should gracefully handle and recover from errors and exceptions.

3.6.2 Availability

The application should be available for use at least 99% of the time during normal operation, with scheduled downtime for updates and maintenance not exceeding 1% of the total operating time.

3.6.3 Security

The application should follow best practices for data privacy and security, including secure handling and storage of user data, and protection against common cyber threats and vulnerabilities.

3.6.4 Maintainability

The application should be designed with modular architecture and well-documented code to facilitate future updates, bug fixes, and feature enhancements.

3.6.5 Portability

The application should be cross-platform compatible and able to run on multiple operating systems (e.g., Windows, macOS, Linux) with minimal modifications.

3.7 Organizing the Specific Requirements

3.7.1 System Mode

Normal Operation Mode: The application continuously captures video frames, performs real-time hand detection, analyzes detected hand positions, recognizes pen grip postures, and provides real-time feedback to the user through the user interface.

Maintenance Mode: This includes regular updates, maintenance tasks, and support procedures to ensure compatibility with new versions of external libraries and optimal performance.

3.7.2 User Class

The primary users of the system are young schoolchildren and individuals who require assistance in maintaining proper pen grip postures during writing activities.

3.7.3 Objects

i. Hand Images

A. Description:

Photographs captured by the device camera, showcasing the user's hand in various writing scenarios.

B. Attributes:

- Image Resolution: Minimum of 720p for optimal clarity.
- Lighting Conditions: The system operates best in well-lit environments for accurate hand detection.

C. Interactions:

- Submitted to the MediaPipe Hand Tracking API for instantaneous hand detection.
- Employed as input for the data preprocessing model, transforming the image format into a suitable array for subsequent Mediapipe model processing.

ii. Hand Keypoints and Posture Results

A. Description:

Numerical representation of three-dimensional coordinates for pivotal hand points and the resultant posture calculations.

B. Attributes:

- Coordinate System: X, Y, and Z coordinates detailing key hand locations.
- Angle Coefficients: Essential data for posture assessment.

C. Interactions:

- Derived through the MediaPipe Hand Tracking API, providing accurate hand position information.
- Utilized by the posture coefficient acquisition model to compute posture outcomes.
- Input for the posture judgment model, determining the appropriateness of the current pen-holding posture.

iii. User Interface Elements

A. Description:

Visual components integrated into the application interface, enabling user interaction and furnishing feedback.

B. Attributes:

- Live Video Feed: Real-time exhibition of the ongoing video stream from the camera.
- Detected Pen Grip Feedback: Visual cues signaling the recognized pen grip posture.
- Customization Options: Interactive elements empowering users to modify settings.

C. Interactions:

- Supplies instantaneous feedback on pen grip postures to guide users.
- Allows users to interact with the application, making adjustments or pausing/resuming detection based on individual preferences.

3.7.4 Feature

- Real-time hand detection using the MediaPipe Hand Tracking API.
- Pen grip recognition algorithm based on hand detection results.
- User-friendly interface for displaying live video feed and pen grip feedback.

- Customization options for adjusting detection sensitivity and feedback settings.
- Data logging and export functionality for further analysis.

3.7.5 Stimulus

- User interaction with the application interface.
- Hand movements captured by the device camera.
- Changes in hand posture and grip during writing activities.

3.7.6 Response

- Displaying real-time feedback on pen grip postures.
- Logging pen grip data for analysis.
- Providing customization options for users to adjust settings.

3.7.7 Functional Hierarchy

i. Real-time Hand Detection:

- Capture live video input from the webcam.
- Utilize MediaPipe Hand Tracking API for hand detection.
- Obtain hand keypoints and positions.

ii. Pen Grip Recognition:

- Analyze hand keypoints to recognize various pen grip postures.
- Determine correct pen holding posture based on angle coefficients and key point information

iii. User Interface (UI):

- Display live video feed with overlaid hand detection results.

- Provide real-time feedback on detected pen grip postures.
- Allow users to adjust settings and customize feedback options.

iv. Data Logging and Export:

- Log pen grip data, including timestamps and detected grip postures.
- Enable users to export logged data for offline analysis or sharing.

v. Customization Options:

- Allow users to adjust detection sensitivity and feedback settings according to their preferences.
- Provide options for adjusting camera resolution and frame rate for optimal performance.

3.8 Additional Comments

- Hardware and Software Requirements: Clearly communicate the system's hardware and software prerequisites, emphasizing the need for a moderately performing computer, a webcam with at least 720p resolution, and a well-lit environment for optimal performance.
- Simplicity and Accessibility: Prioritize a simple and accessible design, especially catering to the intended user base of young schoolchildren. Incorporate visual aids and straightforward language for a seamless user experience.

- Efficient Processing: Strive for efficient algorithms to guarantee real-time hand gesture recognition, minimizing delays, and ensuring a smooth user experience.

4. Document Approvals

   The project was provided by 4 people in the project team and reviewed by professors Pinata Winoto and Tiffany Tang

5. Supporting Information

   See Reference for supporting information.