Taichine–
A New Way to Learn Taiji

18-500 ECE Capstone Project, Team B4
Sirui (Ray) Huang, Hongzhe Cheng, Shiheng Wu, Jerry Feng

Picture from: http://taichisnob.blogspot.com/2013/03/an-introduction-to-24-posture-yang-tai.html

Use Case

The Problem:

- Many people self-learned Taiji, but doing Taiji incorrectly will not achieve the desired health benefits.

The Solution:

- Provide behavioral analysis through video processing using machine learning to compare against reference gesture.
  - Golden Rule – 24 Gesture from the basic Yang’s Taichi exercise
- Provide the flexibility for users to create instructions on their own gesture video

Use Case Requirement

- Need to provide verbal instruction for gesture adjustment
  - 90% accuracy for error joint detection, 10 degree tolerance for angle difference
  - 90% accuracy for correct posture detection
- Enable users to upload custom reference and provide automatic processing necessary for the app
- Ensure that the latency of software is reasonable for users to get real-time feedback when practicing
  - 10 frames per second, cool down time of 2 seconds for wrong postures
- Easy to use once app installed

Solution Approach

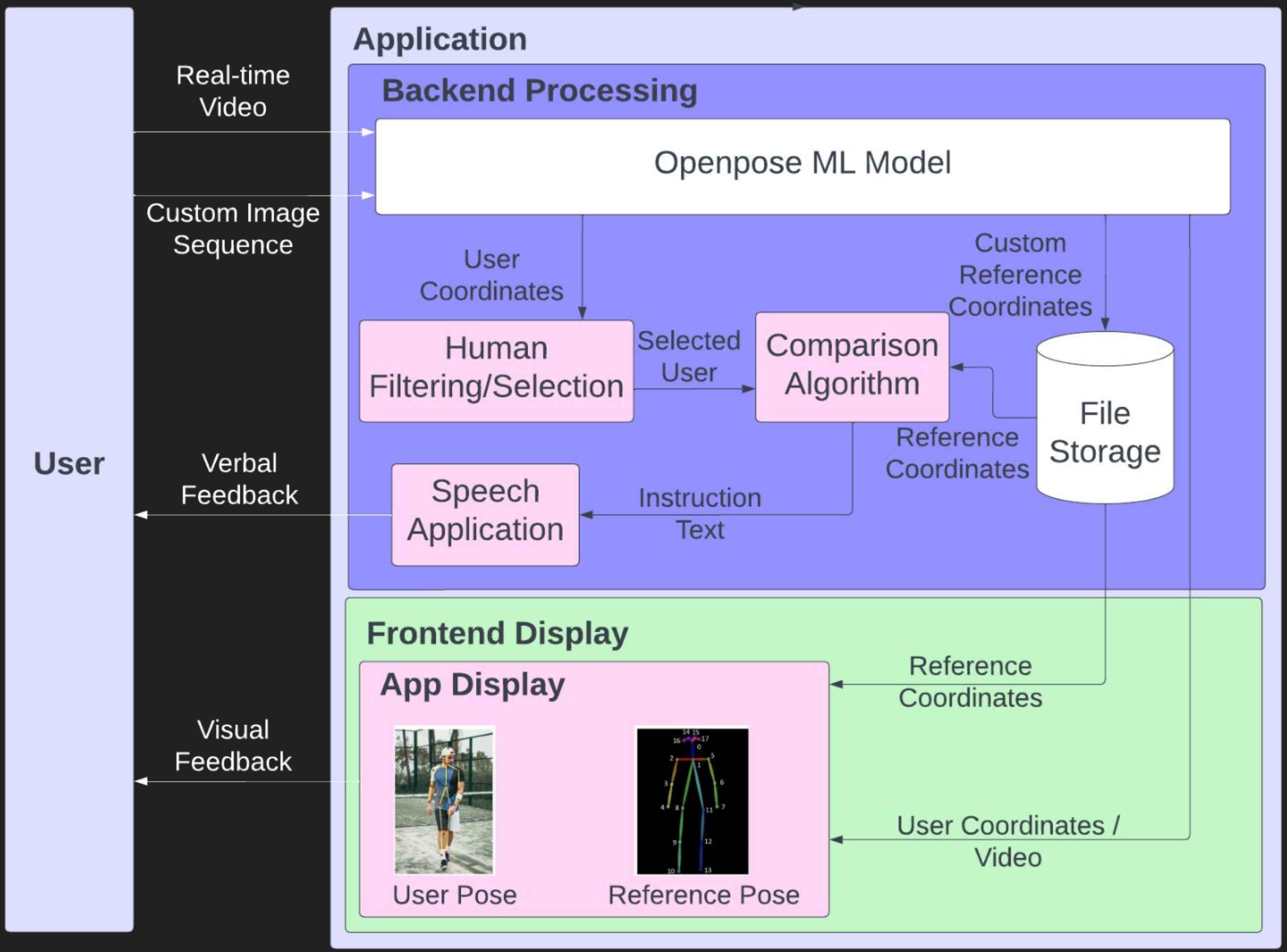
- **Real-time Instruction Pipeline**
  - Allows user to practice Taichi/Custom Pose in front of camera
  - Real-time user input -> Pose Processing -> Reference Comparison -> Verbal Instruction

- **Customization Pipeline**
  - Allows user to store postures as new reference in the app that could be used in the above instruction pipeline

System Diagram

Diagram Key

- □ Off the Shelf Library/Framework
- ▨ New Design
- 🟦 Backend Module
- 🟩 Frontend Module



User

Real-time Video

Custom Image Sequence

Verbal Feedback

Visual Feedback

Application

Backend Processing

Openpose ML Model

User Coordinates

Custom Reference Coordinates

Human Filtering/Selection

Selected User

Comparison Algorithm

File Storage

Reference Coordinates

Speech Application

Instruction Text

Frontend Display

App Display

Reference Coordinates

User Coordinates / Video

User Pose

Reference Pose

Implementation & Design Choices

- **OpenPose (Open source)**
  - Real-time human posture detection application
  - General Internal Logic
    - CNN based with Python/C++ support
    - Detects body parts from all “human” in the image
    - Compute affinity between parts
    - Delete low-affinity connections between body parts
  - Design Choice
    - Output coordinates that is easy to use for angle computation
    - Potential 3D estimation functionality to better support posture comparison and real-world modeling

Implementation & Design Choices

- **App Infrastructure**
    - Language of Implementation: Python
    - Visual Display Package: Python TKinter
    - Storage Model:
        - Folder 1: Reference Posture Image
        - Folder 2: Body Part Coordinates as JSON file
            - Purpose: Save Computation
            - Structure:
                - Name of the pose
                - List of pairs of body parts corresponding to their coordinates

Implementation & Design Choices

- **Comparison Algorithm (Python)**
    - Cosine Similarity Scoring
        - Based on posture instead of absolute position
    - Select correct user when more than one person detected in frame
        - Similarity Threshold: 70%+ (modifiable) to recognize user

- **Text to Speech Translation (Mozilla TTS on Python)**
    - Support multiple languages
    - Support high quality voice offline

Testing, Verification and Metrics

- **Gesture Comparison and Instruction Testing**
    - Ensure the requirements of accuracy is achieved
        - (80% - error detection, 80% - angle accuracy)
    - Angles – Elbow angle, Knee angle, Calf-Floor angle, Arm-body angle, Thigh-body angle, Head-body angle
- **Pipeline Testing**
    - Ensure the two pipeline are well connected from input to all backends
- **Latency Testing**
    - Ensure that the feedback is given with approximately 1 second latency after the input

Testing, Verification and Metrics

- **Input**
  - Real-time recognition – Taichi Beginner poses with intentional errors
    - Poses come from team members; use videos as tests if time allows
  - Customization – Images selected with obvious poses
- **Output & Verification**
  - Real-time recognition – Check if intentional errors match instructions
  - Customization – Check if the necessary data for real-time pipeline to use exist and can be used in the verification process for above
- **Risk Factor**
  - If Test FAIL – recheck the system pipeline and connections, debug comparison algorithm, ensure working solution for most cases
  - Log the pipeline to know which part of the application breaks

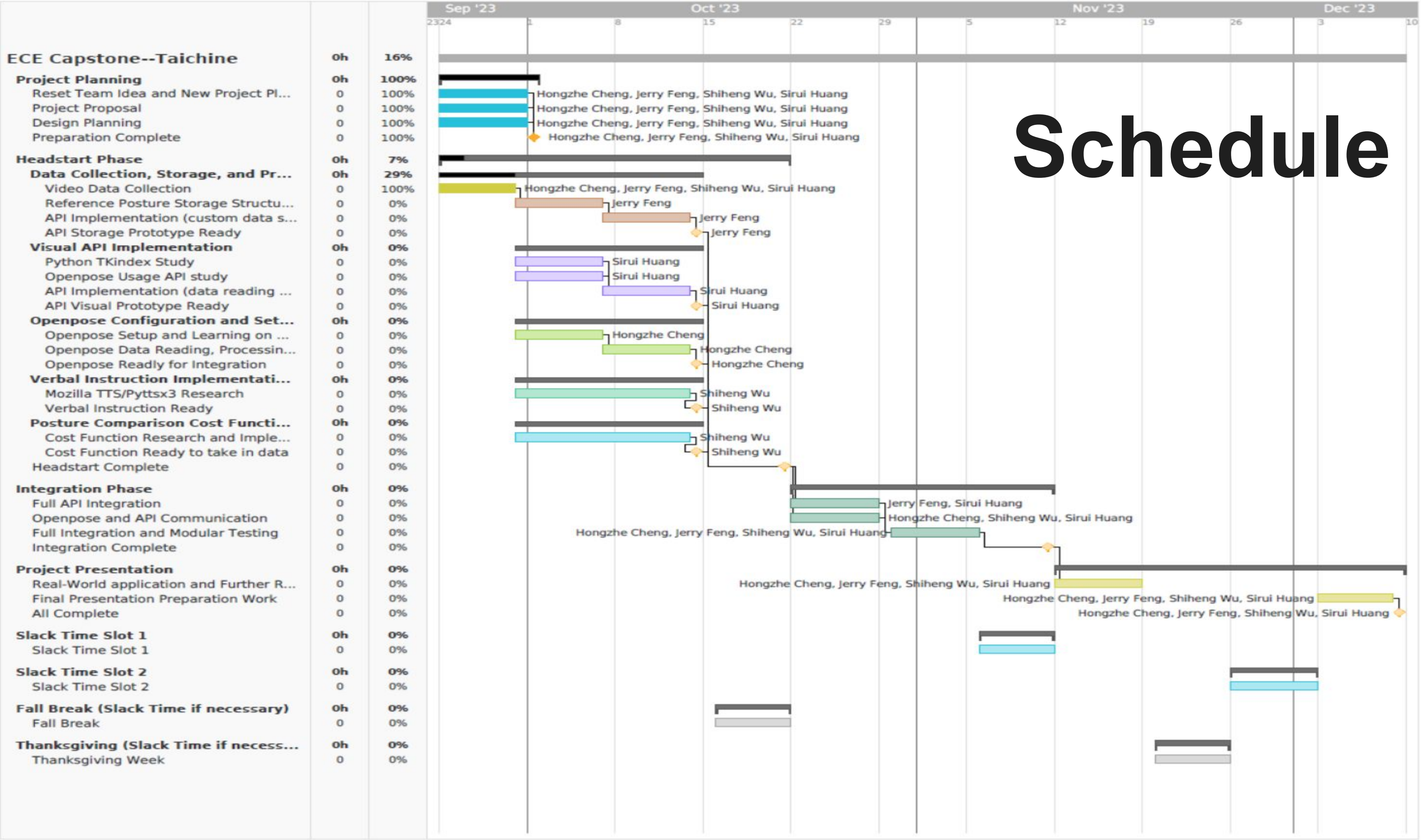
Tasks and Division of Labor

- **Hongzhe Cheng**
  - OpenPose Usage, Display setup
- **Sirui Huang**
  - Application Infrastructure on Posture Display, Data collection
- **Shiheng Wu**
  - Gesture comparison/Scoring algorithm, Verbal instruction
- **Jerry Feng**
  - Custom image’s pipeline and infrastructure, file storage system infrastructure

Stretch Goals:

- Dynamic Posture Instruction, 3D Openpose, Multi-Angle Integration, Cloud Storage

Schedule



ECE Capstone--Taichine	0h	16%
Project Planning	0h	100%
Reset Team Idea and New Project Pl...	0	100%
Project Proposal	0	100%
Design Planning	0	100%
Preparation Complete	0	100%
Headstart Phase	0h	7%
Data Collection, Storage, and Pr...	0h	29%
Video Data Collection	0	100%
Reference Posture Storage Structu...	0	0%
API Implementation (custom data s...	0	0%
API Storage Prototype Ready	0	0%
Visual API Implementation	0h	0%
Python TKindex Study	0	0%
Openpose Usage API study	0	0%
API Implementation (data reading ...	0	0%
API Visual Prototype Ready	0	0%
Openpose Configuration and Set...	0h	0%
Openpose Setup and Learning on ...	0	0%
Openpose Data Reading, Processin...	0	0%
Openpose Readly for Integration	0	0%
Verbal Instruction Implementati...	0h	0%
Mozilla TTS/Pyttsx3 Research	0	0%
Verbal Instruction Ready	0	0%
Posture Comparison Cost Functi...	0h	0%
Cost Function Research and Imple...	0	0%
Cost Function Ready to take in data	0	0%
Headstart Complete	0	0%
Integration Phase	0h	0%
Full API Integration	0	0%
Openpose and API Communication	0	0%
Full Integration and Modular Testing	0	0%
Integration Complete	0	0%
Project Presentation	0h	0%
Real-World application and Further R...	0	0%
Final Presentation Preparation Work	0	0%
All Complete	0	0%
Slack Time Slot 1	0h	0%
Slack Time Slot 1	0	0%
Slack Time Slot 2	0h	0%
Slack Time Slot 2	0	0%
Fall Break (Slack Time if necessary)	0h	0%
Fall Break	0	0%
Thanksgiving (Slack Time if necess...	0h	0%
Thanksgiving Week	0	0%

Sep '23
Oct '23
Nov '23
Dec '23
23
24
1
8
15
22
29
5
12
19
26
3
10
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Jerry Feng
Jerry Feng
Jerry Feng
Sirui Huang
Sirui Huang
Sirui Huang
Sirui Huang
Hongzhe Cheng
Hongzhe Cheng
Hongzhe Cheng
Shiheng Wu
Shiheng Wu
Shiheng Wu
Shiheng Wu
Jerry Feng, Sirui Huang
Hongzhe Cheng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang
Hongzhe Cheng, Jerry Feng, Shiheng Wu, Sirui Huang