Taichine: Tai Chi Form Practice Tool for Beginners

Hongzhe Cheng, Sirui Huang, Jerry Feng, Shiheng Wu

Department of Electrical and Computer Engineering, Carnegie Mellon University

Abstract—**A system for comparing poses performed by users against professional Tai Chi poses and generating verbal feedback to help with Tai Chi learning. The system also allows users to upload image/image sequences as custom training, thus broadening its usage beyond the scope of Tai Chi.**

***Index Terms*—OpenPose, Posture Detection & Evaluation, Tai Chi, Machine Learning, Software Systems**

I. INTRODUCTION

Tai Chi, as a form of Chinese martial art, highlights controlled movements and sustained postures. Tai Chi has been a popular exercise among senior citizens, and it is also gaining popularity in younger generations. According to research, Tai Chi can yield a multitude of health benefits, including but not restricted to stress reduction, improved balance, and alleviation of knee osteoarthritis pain [1]. Despite the benefits, Tai Chi can pose challenges to beginners and might lead to injuries if not practiced properly [13].

Our team aims to create an application that incorporates state-of-the-art machine learning technology to aid practitioners in evaluating their Tai Chi postures. Our application analyzes user image input from the practitioner with the OpenPose deep learning system and compares the user's poses with those of a Tai Chi professional.

Our system will rate the user's pose using a scoring algorithm and give audio and visual feedback on correcting one's pose if the limb angle error exceeds the set tolerance. Our system integrates the 24-form Yang Style Tai Chi Poses, while also supporting the option for users to upload their custom poses/pose sequences that they would like to practice onto our system, with custom pose sequences having a maximum of 5 poses. In addition, our system supports pose sequencing and will only progress to the next pose once the user has met a minimum score on the current pose.

In general, utilizing machine learning technology, our system provides an interactive approach for Tai Chi learning and offers an accessible and flexible training experience to the system users.

II. USE-CASE REQUIREMENTS

Our system provides three main functionalities: pose comparison, corrective verbal instructions, and interactive user interface.

For pose comparison, the system is expected to achieve 90% accuracy in identifying coordinates of the user's body (knees, shoulders, elbows, etc.) and user's pose correctly. According to Pedersen et al. [4], OpenPose's lowest accuracy performance in a lit environment was 89.74%. Due to the significant decrease in OpenPose's recognition accuracy in a dark environment, the users are expected to use the application where the camera is able to capture clear images of the user. Additionally, joint angles have been used for pose detection in several studies examining computer-vision based approaches for physical rehabilitation and assessment according to a survey by Debnath et al. [2]. In particular, De Gama et al. [3] show that an angle-based posture detection algorithm could identify correct movements approximately 100% of the time on a limited sample size under controlled conditions based on motor rehabilitation therapies with stroke victims. These preliminary studies ensure the accuracy of the approach used for the Taichine's development.

The application interface has to be easy for all age groups to use. In testing, an average score of at least 8 out of 10 is expected in terms of users' evaluation on how intuitive the application is. Meanwhile, the application has to take into account different health and movement conditions with respect to the users. The system needs to provide enough approaches for its users to follow the instructions. Taichine uses visual and audio feedback for guiding its users.

The audio has to be minimal but effective so that the users can follow easily in a short time frame, thus requiring clarity and appropriate speed for the speech. To ensure users receive timely feedback from our system, there should be a maximum 4-second latency from when the system finishes receiving user image input to when the audio feedback is played.

Considering the minimum distance away from the camera the user has to be in order for the camera to capture valid footage, the information displayed by the system should be visible from meters away. Meanwhile, Taichine should be able to generate optimal feedback in edge cases, such as having multiple people in captured images or operating in dark environments, which will be discussed later.

III. ARCHITECTURE & PRINCIPLE OF OPERATION

As in the system diagram (Fig. 1), our system is software-based. The application takes two different types of input. Depending on which type of input it gets, the system either carries out real-time pose recognition and instruction, or lets users upload custom poses and expand the reference set.

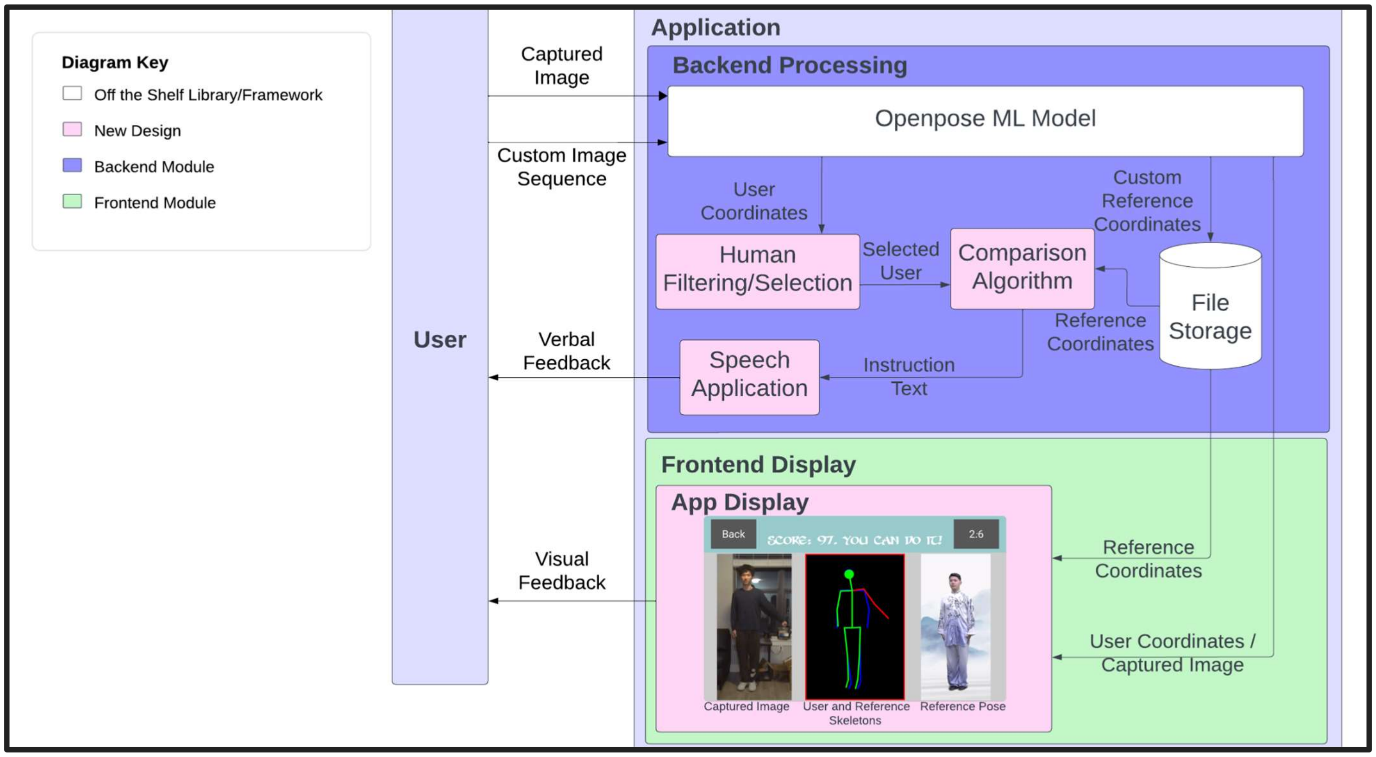


Fig 1. Block Diagram of System Diagram

As shown at the top of the diagram, the first type of input is a real-time captured image, which represents the captured image input of the user practicing the reference Tai Chi pose. OpenPose is a pose recognition software that takes image input and returns a set of coordinates representing the core body part associated with each "human" detected in the input.

As per Taichine, the pipeline feeds OpenPose-generated coordinate sets to the user filtering function. The application will load the reference pose and corresponding coordinates for user-filtering. While it is recommended that only one single trainee should be present in the camera frame for their training, it is highly possible for users to use Taichine in environments where other people are present. In cases where passersby are present, Taichine assumes that the trainee's posture would be the most similar to the reference posture compared with noisy coordinates from passersby. Consequently, the filtering system works closely with the comparison algorithm that evaluates all captured human forms, and selects the one closest to the reference as the actual user. For the selected user and their pose data, the comparison algorithm generates body part correctness data and angle differences if there are any. It then formats them with verbal instruction templates and passes them to the speech application. The verbal instruction consists of detailed information on where and how the user's poses are off from the reference. Such instructions will be in the form of feedback like "Your left arm needs to be 20 degrees up". Alongside the verbal feedback, as at the very bottom of the diagram, the frontend will also offer visual feedback comparing the user and the reference pose.

The second arrow pointing from the user to the system represents the second type of input, the custom image sequences. The customization pipeline allows users to pass in images of poses that they are interested in practicing, and use them as references beyond the built-in 24-form Yang Style Tai Chi poses.

Users should input images containing one and only one "human" to guarantee the effectiveness of the system. The images will be passed to OpenPose and receive the coordinate set for the pose in the image. The application will store the image and coordinates separately into the file system for future usage. This pipeline enables users to update the available training options. After they upload images as custom pose sequences, the application will show them on the selection page next time the users open it up. Also, given that the application processes the general definition of "Poses", the custom pipeline also supports other poses in 2D space (without too many body parts perpendicular to the screen and camera). Examples include Yoga, Dancing, etc.

IV. DESIGN REQUIREMENTS

To meet the User Requirements listed in section II, Taichine has to first satisfy corresponding design requirements for its various subsystems. Below are the design requirements considered during the planning, development, and testing process.

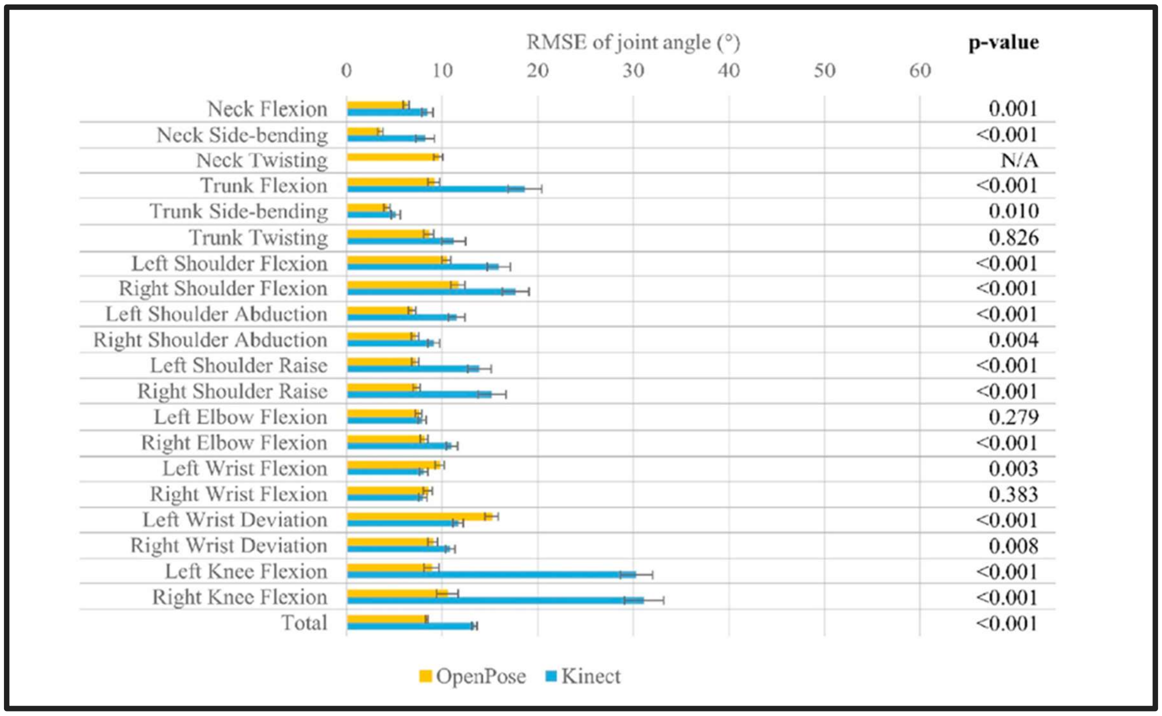


Fig. 2: OpenPose Measurement of Joint Angle Error Compared to Older Posture Detection Algorithm Kinect [6]

A. *Footage Input Calibration*

Regarding the input footage, the integrated cameras of the users' laptops would suffice for the purpose of getting clear input images. To help users set up the camera, a calibration stage should be present in the training process. Taichine will give the user time to position themselves and calibrate the camera to capture their full body. Still, the user is expected to utilize Taichine in a well-lit environment for more accurate pose recognition. There will be no low-lighting support. The user should also only train on a flat ground for safety concerns.

B. *Angle Detection*

After calibrating and making sure the user stays inside the camera frame, we aimed to control the difference between user body angle and reference angle should be within 10 degrees of the real-world setting. For example, when the user is 20 degrees lower in their arm position, the comparison algorithm should report that the arm is lower by an angle of 10 to 30-degrees. Evaluating OpenPose models' performance on laptops with resolutions from 720p to 1080p reveals 90% accuracy in angle detection after manually adjusting the camera angle and distance of the user from the camera.

C. *Verbal Feedback*

Verbal feedback should be emitted within 1 second after the evaluation period. This is to let the user know in real time what and how to improve when practicing. To achieve this, some requirements have to be met in receiving and processing feedback. Mean human listening rate is around 309 words per minute as in a 2018 study [5] and Taichine instructions are mostly around 10 words (e.g. "Move your left arm up twenty degrees"). Based on these, TTS should take at most 2 seconds to instruct the user on the most different body parts. This design will help users correct deviations in each limb step-by-step. The TTS loudness can be adjusted with the user's laptop volume control, while the TTS loudness at maximum laptop volume should ensure the verbal instructions be audible and clear to a user 5 meters from the laptop. Still, it is preferable to practice in environments with low background noise to reduce distraction in general.

D. *User Interface*

The UI has to be easy to use. In particular, it should have a short learning span even for people with minimal knowledge in computers. Navigating from functionality to functionality should feel like swiping on a smartphone, and most of the information on screen ought to be visible from meters away.

E. *Customization Pipeline*

The customization pipeline has to support the most common image file types, such as png and jpeg. The pipeline should let the user batch upload images and edit the sequence, including changing the sequence name, switching pose order, and deleting poses from the sequence. Still, whether it is a screenshot or photograph, a posture with all body parts visible helps the application label all joints accurately. The user should upload images with one and only one human present doing the desired pose, to prevent misidentifying the reference due to overlapping limbs or multiple poses within the picture.

F. *Supported OS*

Taichine will support Windows. Most potential users of Taichine, especially the senior citizens, tend to be more familiar with Windows rather than MacOS and Linux.

V. Design Trade Studies

A. *Decision of Local Application*

While internet access improves compatibility and enables access to cloud computations, local applications have the advantage of faster performance and responsiveness as they leverage the device's hardware and resources more efficiently compared to web apps requiring cloud servers. Therefore, Taichine works offline to ensure uninterrupted functionality in environments with limited Internet support, like parks and large outdoor spaces that have larger space and stronger lighting. Local also provides more privacy and security. With camera captured data stored locally, exposure to online threats and insecure connections is reduced.

B. *Decision for Laptop Platform*

Smartphones have become so prevalent today that, when it comes to application development, a smartphone application comes to mind as an option naturally. In fact, a smartphone application can be more convenient for Tai Chi practitioners, as smartphones are much less cumbersome than laptops. However, phone screens are not the best option to provide detailed feedback. Taichine has to produce and convey real-time images of the user, reference images processed by OpenPose, and voice guidelines at the same time. A laptop screen is often between 13 to 17-inches, as shown in Fig. 3, making them more than 4 times larger than 7-inch smartphone screens. Laptops allow Taichine to present much more information on the training page than what smartphones can do, leading to more detailed feedback. Meanwhile, with the web app option ruled out for reasons discussed in Section A, a local smartphone application becomes the only viable option. Unfortunately, the models and engines Taichine uses only support Windows and Linux, thus making a local Windows application the most reasonable option for Taichine's platform.

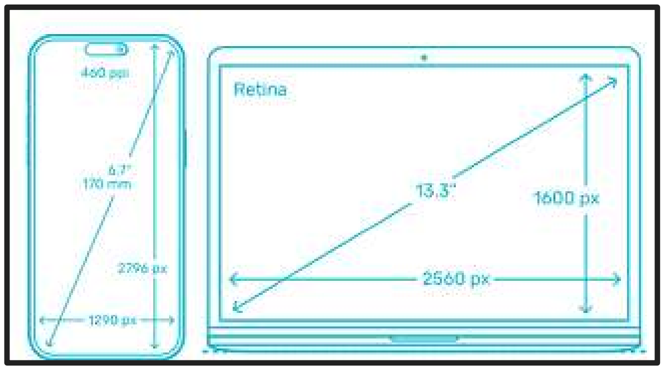


Fig. 3: Comparison of Smartphone Screen to Laptop Screen [7]

C. *Posture Model Selection*

The recent boom of AI significantly increased the number of options for posture detection models. Among them, MoveNet and Openpose caught our attention. Google's MoveNet specializes in capturing detailed facial expressions and tracking atypical/fast-moving postures. Conversely, OpenPose works locally on both GPU (CUDA support) and CPU without occupying too much resource. It focuses more on the details of body part joints, in contrast to other models developed on Tensorflow platforms, excelling at capturing quicker motions. The said models also place more weight in capturing the upper body instead of limb movements. One particular functionality of Openpose is its multi-people detection, which later proves particularly useful in materializing our human filtering/selection module.

Models like MoveNet and Posenet provide relatively higher FPS and quicker responses. According to a 2022 study, MoveNet has a processing speed of (0.49±0.05s/frame), followed by Openpose (0.51±0.08s/frame)[8]. Although MoveNet is faster than Openpose, the few-millisecond response difference is negligible. Considering additional functionality it provides, OpenPose becomes the selected model for Taichine for its posture detection specialty and manageable resource utilization.

Tai Chi is an exercise aiming towards balance and control. Therefore, a higher FPS camera and quick responses from the model is unnecessary for tracking Tai Chi practitioners' movements. Instead, posture accuracy is the major concern. OpenPose performs best on angle errors in median numbers in detecting critical joint positions, such as knees and hip, according to a 2022 study conducted by Edward P. Washabaugh et al (Fig. 4). OpenPose also produces the lowest errors, averaging at 5.1±2.5 degrees of error over one iteration, compared with 9.1±3.0 degrees found in MoveNet [9]. In addition, OpenPose can handle complex and intricate poses. This suits particularly well for analyzing Tai Chi, which requires detailed pose information, including detailed hand key points assigned to various hand joints. MoveNet and DeepLabCut are more geared towards fitness tracking, which only stands out in detecting simple and repetitive movements like squats and jumps with no demand of giving accurate instructions.

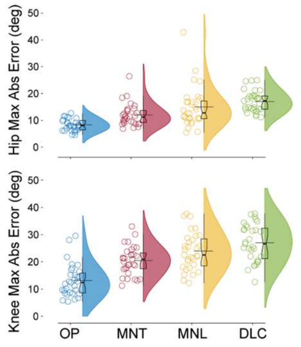


Fig. 4: Rain cloud plots depicting maximum absolute errors for hip and knee kinematics. The circles on the plot indicate individual data points. Abbreviations: Deg (degrees), OP (OpenPose), MNL (MoveNet Lightning), MNT (MoveNet Thunder), DLC (DeepLabCut). [10]

Model Name	Model Performance		
	Posture Detection Accuracy	***Avg. FPS***	***Multiple Person***
OpenPose	Great	15	Supported
MoveNet	Difficulty for Overlaps	160	Not Supported
PoseNet	Difficulty for Overlaps & Accuracy issues	90	Not Supported

Table I. Posture Detection Model Comparison

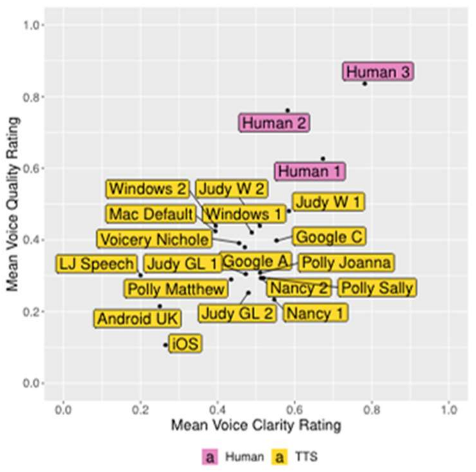


Fig. 5: Comparison of VoiceEngine Clarity and Quality (Windows 2 represents pyttsx3,Judy represents Mozilla TTS) engines) [11]

D. *Voice Engine Choice*

For the voice engine, Taichine uses the Pyttsx3 module. Pyttsx3 works without an internet connection, which fits the application's local system design. Another choice considered during the implementation process is Mozilla TTS, which is also open source and provides high voice quality. Unfortunately, the installation of the package is too bulky and redundant for our system, with the total package size reaching around 2 GB. In contrast, Pyttsx3 provides similar functionality with only 40 KB size and much faster response time (2-3 seconds compared to 10+ seconds wait time for Mozilla TTS to synthesize the speech on full scale algorithm without significant improvement).

While the voice quality might not be as high as more advanced packages, it is imperative for the instructions to be delivered with a pronounced clarity, ensuring ease of comprehension for users to adhere to. Based on the data given in Fig. 5, Pyttsx3 module sacrifices its voice quality and clarity and, in exchange, becomes particularly fast and light-weighted. Pyttsx3 uses Microsoft's latest SAPI (Speech Application Programming Interface) 5 drivers that are integrated in Windows machines with XP or newer operating systems installed. This makes it highly compatible with current Windows laptops. Similar applications utilizing the same driver include Microsoft Narrator and Adobe Reader, but since the application development is in Python, Pyttsx3 offers the most convenience and meets all the design requirements.

E. *Comparison Algorithm*

Taichine adapts vector representation for comparing user body postures with reference postures presets. This is mainly because OpenPose returns its data representing each body posture as a high-dimension vector, with each dimension corresponding to a key point. For example, when Openpose detects 18 key points, each posture can be represented as a set of 18-2D vectors.

The major advantage of representing posture data in vector form surfaces when cosine similarity comes into the picture. Cosine similarity ranges between 1 and –1 and lies mostly between 1 and 0 in the case of Taichine because users will realize from negative values that their posture is mirrored. Any results less than 1 indicate dissimilarity between the user posture and the reference, which quantifies similarity concisely. Moreover, the calculated similarity can be converted to a score with ease. The frontend can then display a score to show how the user is doing and decide if a posture is accurate.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| \cdot |\mathbf{B}|}$$

$A \cdot B$ represents the dot product of the two vectors (postures).

$|A|$ and $|B|$ are the magnitudes (Euclidean lengths) of the vectors A and B, respectively.

Additionally, using numpy.arctan2 can help determine the signed angles for generating instructions related to pose direction. The function takes in two vectors A and B as defined above and outputs a signed angle difference between two vectors with a range of $[-\pi, \pi]$ relative to the positive x-axis. After converting it back to angles, the absolute value of the difference represents the dissimilarity and the sign of the difference indicates its direction. Each limb vector is also compared with positive x-axis to yield a signed angle which is passed to the frontend for skeleton drawing.

One algorithm Taichine could have used for comparison is to measure the absolute positions of the user posture and seeing how different the body joints are from those of the reference. In this case, the user can get each of their body parts as close to those of the reference as possible, which seems more intuitive. Yet, given that Taichine is not a personalized application, the evaluation result will be influenced by the varied lengths of body parts different users have. For example, a person with shorter limbs will not be able to reach reference points that are easily reachable by a taller demonstrator. Even if they have done the absolute correct posture, their score will be low since their body parts are not at the same positions as the reference body parts. Additionally, the vector representation approach does not need ground reference. So if the user attempts to do Tai Chi on an inclined ground, the biased absolute positions will mess with the absolute-position comparison scheme, but the joint angles measured with the vector will be the same. Based on hands-on testing as well as the regular criterion for evaluating a Tai Chi pose, the joint angles for a Tai Chi pose should be universal even for people of different body sizes. As people approach the same posture within a 10 degree tolerance, their posture will end up being the same based on joint angles. For a more detailed discussion on implementing more complex statistical and math models in case joint angles are proven wrong, please refer to Risk Mitigation section VIII.D.

F. Frontend Library

TKinter, PyQt, WxPython, and Kivy are the four potential frontend libraries for Taichine. Considering system complexity and time restriction, Kivy ends up as the best option. The widgets Kivy provides are helpful for implementing all different parts of the frontend infrastructure, and the http/css-like structure helps separate UI design from functionality development.

To briefly cover the disadvantages of the unchosen packages, WxPython is only compatible with Python 2.7 and is also not actively maintained. Issues with restricting Taichine to Python 2.7 include the inability to work with an improved "os" module and the incompatibility with f-strings in python 3 [14]. These issues would have made file system handling much more difficult to implement. Additionally, the inactive maintenance makes WxPython an especially risky choice.

The complexity of PyQt is the central reason it is not chosen. PyQt requires a compile stage for its code, and many PyQt users have run into a variety of compile stage issues when using the package. PyQt also requires learning and understanding other languages to be able to use some of its features (for example, needing to understand javascript to use its declarative programming feature and some C++ for the general library).

Finally, due to its limited amount of widgets available and overall less appealing aesthetics compared with Kivy, TKinter is not used.

VI. System Implementation

A. Frontend Infrastructure

The frontend app infrastructure is the interface our user uses for interacting with the backend evaluation system. The hierarchy of the app infrastructure is as follows:

1. Main Menu

The main menu is the first page the user will see when they enter the app. It will consist of the app title "Taichine" and a list of five option items. The user can select the first four options to transition to different pages. They can transition to the Mode Selection Page, the Customization Upload Page, the Options page, or the Tutorial page from the main menu by clicking the corresponding actions. Or, they can exit the application by selecting the "Exit" option at the bottom.

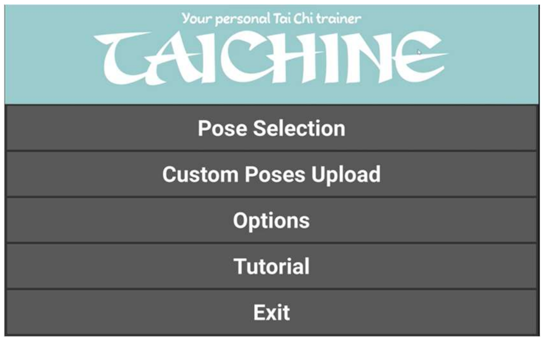


Fig. 6A. Screenshot of the Main Manual

2. Mode Selection Page

The user can reach this page from the main menu when they click the "Pose Selection" option. The Mode Selection Page is where the user can select between the integrated 24-form Tai Chi poses or the Custom poses the user uploaded previously.

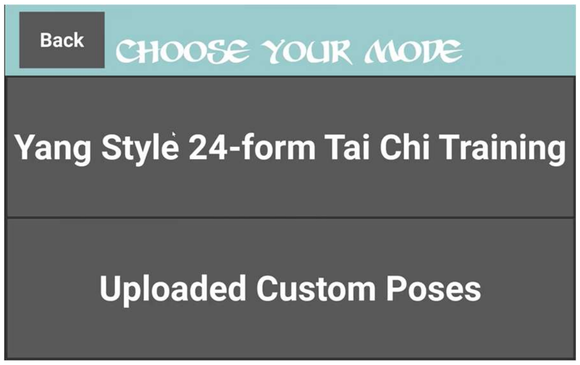


Fig. 6B. Screenshot for Mode Selection Page

3. Pose Selection Page

The user can reach this page from the Mode Selection Page or if they just completed a training from the Training Page. This page is dedicated to helping the user select the pose they want to practice. The default 24-form Tai Chi poses and the user custom poses will be displayed here depending on the user's selection on the Mode Selection Page. The poses are displayed in list view, and a preview of the pose will be displayed as a small image near the name of the pose.

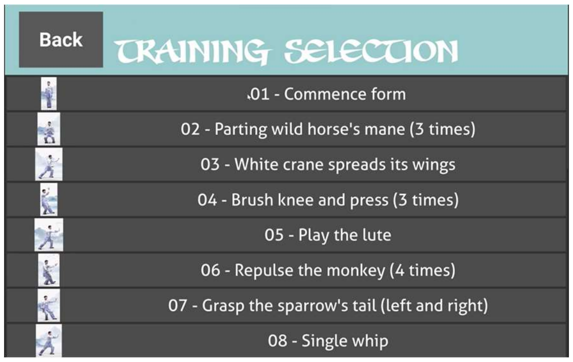

Back
TRAINING SELECTION
01 - Commence form
02 - Parting wild horse's mane (3 times)
03 - White crane spreads its wings
04 - Brush knee and press (3 times)
05 - Play the lute
06 - Repulse the monkey (4 times)
07 - Grasp the sparrow's tail (left and right)
08 - Single whip

Fig. 6C. Screenshot of pose selection screen for integrated poses (Yang-Style 24-form Tai Chi Training)

4. *Customization Image Upload Page*

The Customization Image Upload Page is where the user can upload images of postures into the system and practice with the system functionalities. The user will access the page to upload custom images from the main menu. The user will first arrive at the "upload" screen as can be seen in Fig. 6D, where they will click the white button at the bottom of the screen. A window of the native windows file manager will pop up, where the user can select the png file/files they would like to upload as a custom pose/pose sequence. The input images in one upload attempt will be packed as a posture sequence, and the user can switch the order of the poses on this page using the left and right arrows at the bottom of each image, with the left arrow swapping the position of the image with the image to the left of it. The user can also click the red "X" button above each pose image to remove it from the pose sequence. Once the user hits the green "confirm" button at the bottom of the screen, the poses are imported into the Pose Selection Page, where the sequence order will be fixed according to the order they are inputted when the user clicked the green "confirm" button.

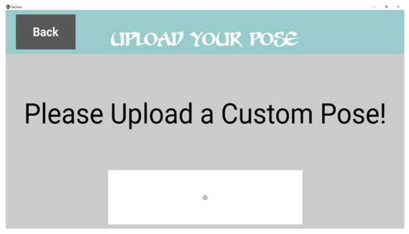
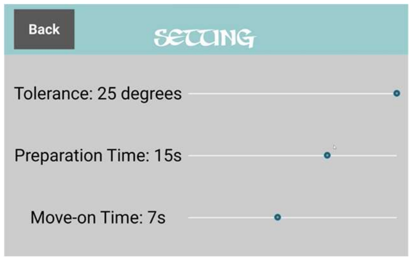


Fig. 6D. Screenshot of Options Page

5. *Options Page*

The user can reach this page from the main menu when they click the "Options" option. The Options Page is where the user adjusts the parameters of the User Feedback Subsystem (described in the next section) to personalize their testing experience. The user can control the evaluation strictness in terms of "Tolerance" angle (maximum angle of the angle between user limb and reference limb without the system recognizing it as error) of the system's pose evaluation with slider control. The user can also change the preparation time and the automatic move-on time in between incorrect poses. "Preparation Time" signifies the number of seconds for the user to prepare for the first pose, and "Move-on Time" signifies the number of seconds for the user to adjust their body positions between different reference poses.



Tolerance: 25 degrees

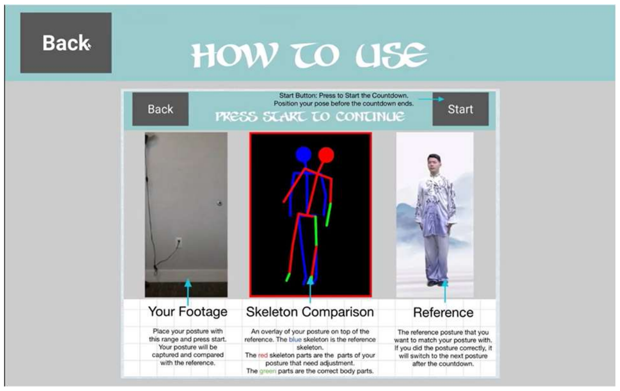
Preparation Time: 15s

Move-on Time: 7s

Fig. 6E. Screenshot of Options Page

6. *Tutorial Page*

The user can reach this page from the main menu when they click the "Tutorial" option. The Tutorial page is where the user can get a quick overview of how to use the main functionality of the program, the training screen.



Start Button: Press to Start the Countdown. Position your pose before the countdown ends.

Place your posture with this range and press start. Your posture will be captured and compared with the reference.

An overlay of your posture on top of the reference. The blue skeleton is the reference skeleton. The red skeleton parts are the parts of your posture that need adjustment. The green parts are the correct body parts.

The reference posture that you want to match your posture with. If you did the posture correctly, it will switch to the next posture after the countdown.

Fig. 6F. Screenshot of tutorial page

7. *Training Page*

The training page is the main functionality page where the user is able to start practicing the Tai Chi poses they selected in the Pose Selection Page. This page will mainly consist of three large widgets. The widget to the left is a camera widget, showing the live footage from the laptop's camera. The center widget is the skeleton drawing canvas, where the user skeleton and the reference skeleton will be drawn. The rightmost widget is the reference image widget, which will display the reference the user is supposed to follow for this training.

After the user accesses this page, the user can press the "Start" button at the top-left corner of the screen. A countdown will start, and the number of seconds counted will be equal to the value "Preparation Time" the user configured in the Options page. When the countdown ends, the application will capture the user's posture at the moment and pass it to the backend. After the backend has finished processing the input screenshot, it will take the coordinates and the body part correctness checks from the backend. With this data, the application will draw two skeletons on the skeleton canvas. The skeleton in blue is the reference skeleton, showing what the reference pose is like, and the skeleton in green and red is the user skeleton, showing the user's pose and the correctness of each of the user's body parts. If the body part is marked red, it indicates that the user needs to adjust it to an angle within the set tolerance. Otherwise, the body part is correct and needs no correction.

When the user's pose does not pass the system check (any of the user body parts is marked red), the skeleton drawing canvas will draw itself a red border. The countdown will restart, this time counting the number of seconds equivalent to the "Move-on Time" value the user configured in the Options page. Also, verbal instructions on how to improve the pose will be played from the laptop's speakers. The reference pose will be the same pose as before, and the user has to pass all the body parts tests before moving on to the next pose in the training sequence.

When the user has performed the pose accurately (all user body parts are marked green), the skeleton drawing canvas will draw a green border. The system will proceed to the next pose if the current training is not yet over, i.e. there are references to images left in the set of training. Verbal instructions will simply say "Great, you did it!" in this case. If the current posture is the last one, the user will move on to the Result screen.

The backend will also pass back a score, evaluating the user's overall performance in carrying out the previously tested pose. This score will be displayed above the three widgets.

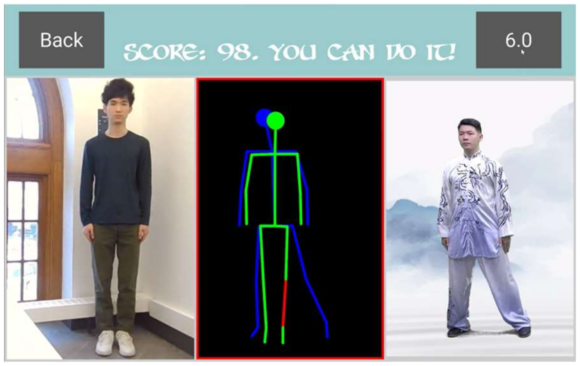
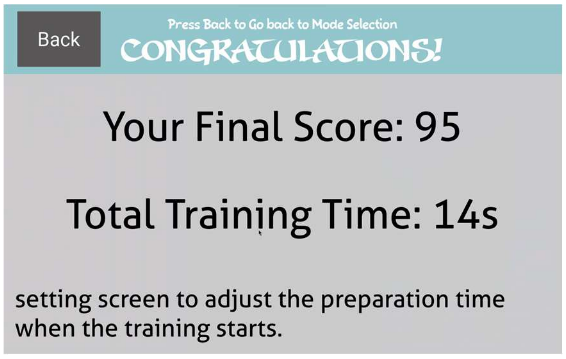


Fig. 6G. Screenshot of ongoing Training Page

8. Result Page

The user can reach this page by passing their current training. The Result page is where the user gets an overview of their previous training. An overall score of the user's performance and the total time the user used to pass the training will be displayed at the center of the screen. Meanwhile, at the bottom of the screen, a randomized tip for using the Taichine system will be displayed. Clicking the back button at the top-left corner will bring the user back to the Pose Selection page.



Your Final Score: 95

Total Training Time: 14s

setting screen to adjust the preparation time when the training starts.

Fig. 6H. Screenshot of End of Training Result Page

For the implementation details, the page transitions will be implemented with Kivy's Screen Manager functionality, with each page in our application being a separate Kivy screen object.

According to the user inputs, the frontend sends and retrieves relevant data from the backend, such as the "Tolerance" variable for the User Feedback Subsystem and posture images. Depending on which page the user is on, the frontend should activate relevant backend functionalities when and only when necessary.

On the training page, the frontend will retrieve reference coordinates from the backend every time a new posture training starts. Then it should capture the user's footage and send it to the User Feedback Subsystem, taken in by OpenPose. After the backend has finished processing the footage image, the frontend will retrieve the score of the user's pose and display it on screen. The frontend will also decide whether it should move onto the next pose based on the body part checklist passed from the backend. If the backend reports correctness for all the user body parts, the frontend will proceed to the next pose automatically. Otherwise, the frontend will listen for backend verbal instructions and play it when requested. A high level diagram of this process can be seen in Fig. 15 in the Appendix.

B. User Feedback Subsystem

The user feedback subsystem will contain two main parts inside the Python development environment, the comparison algorithm system and voiceover feedback system.

The backend system would first load the processed output of the user image from OpenPose and run the user selection algorithm. Upon deciding on one single user and getting the corresponding coordinate set consisting of 18-key points representing the current user's body posture, the Python algorithm will first convert the key points into different vectors representing arms, legs, head and upper body. From received data, all four limbs, upper body, and head position should be clearly reflected in vector form. Then the set of vectors is compared with the stored reference position (prerecorded or user uploaded) to generate differences between the user's posture and reference posture and generate a 'score' visible to users. This process is repeated for each picture captured over an interval the same length as the user defined move-on time (5s by default). While utilizing that dataset to generate text, lower body parts are prioritized and mentioned first to the user from the voice engine output. The lower body provides a stable foundation for movement and balance. Developing a strong connection with the ground, known as "rooting," is essential for maintaining stability during Tai Chi movements.[12] An example of one person having multiple issues on their right leg and head for one posture, the only instruction passed to the voice engine will be the one on their right leg: 'Move your left leg inward by 20 degrees'.

The instruction is structured in the following format:

"Turn your {limb name}

{inward/outward/upward/downward}

by {angle_degrees} degrees."

Example instructions could sound like: "Turn your right upper arm upward by 20 degrees", "Turn your left calf inward by 15 degrees." based on the coordinates and angles returned.

Another case will be if the user has all their limbs angle difference within the to the posture, the system will generate text strings like 'Good job! You've mastered this posture!' and informed the pipeline that the user has passed this posture. The pipeline will then trigger a change in the user interface to allow them to proceed into the next posture.

The second part will be the Python package of voiceover system pyttsx3. After receiving the text string, it generates the corresponding speech waveform matching every instruction generated. This .mp3 file can then be played utilizing the pygames module to create the spoken instructions through the speaker to inform the user on how to improve their Tai Chi posture or congratulation comments when the user passed the posture similarity check. The engine will be initialized when first called upon, and will become dormant until a new set of text strings is passed in.

The third part will be the backend processing system for the coordinates received from the OpenPose algorithm for frontend skeleton drawing. Similar but different from the first system mentioned above to measure joint angles between user and reference pose, this system aims to provide relative angles utilizing the function of arctan2 from the numpy package. Through measuring the relative signed angle between the positive x-axis and the given limb vector, the set of data could be passed to the frontend and combined with joint key points data for skeleton drawing purposes.

For a visual demonstration of the backend workflow over analyzing user poses and producing feedback, please refer to the system diagram Fig. 1. This section is meant to dig deep into the system implementations and talk about the program logic for each functionality box in the right side (backend) of the diagram.

C. Customization Subsystem

The customization subsystem will be a subsystem that allows users to upload custom Tai Chi poses, not natively in our application, that they want to practice using our application. The user will interact with the interface from our application accessible from the main menu of our application. The user will first arrive at the "upload" screen as can be seen in Fig. 7A, where they will click the white button at the bottom of the screen, which will create a popup window of the native windows file manager, where users can select the png file/files they would like to upload as a custom pose/pose sequence. The popup window of the native windows file manager is done by calling a function from the "plyer" library. We did not use Kivy's native file chooser object was because it had a different UI from the native windows file manager, and we thought this would decrease the user experience as they would have to get used to a new UI they were unfamiliar with, as compared to the native Windows file manager UI that users would be familiar with. A screenshot of the upload screen with the native windows file manager popup can be seen in Fig. 7B. Users will not be able to upload a directory of images for ease of parsing the files. After the user finishes selecting the file/files they would like uploaded, they are then taken to the confirmation page, as can be seen in Fig. 7C.

For reasons of visual cleanliness, we limit the number of images the user can upload in a pose sequence to 5 images, if the user tries to upload more than 5 images, their upload fails and they are met with a popup telling them they tried to upload too many images. By default, the pose/pose sequence is named after the first pose in the sequence (pose/pose sequence name is underneath the text box), however the users can type in the name they would like to give the pose/pose sequence into the text box at the top of the screen and once they hit enter the change will be registered by the system, and the name of the pose underneath the text bar will change accordingly. As can be seen in Fig. 7C, the confirmation screen has the names of the png files above the png file/files the user uploaded. This is kept track of using global variables, as this has been found to produce cleaner and simpler code compared to writing code to navigate Kivy's parent-child screen and object structure. The user can modify this order, by clicking the left arrow button underneath one of the poses to swap the order of that pose with the pose to the left of that pose with the position of the poses in the screen changing to reflect this. A similar process happens for the right arrow, except the right arrow is used to move a pose "forward" in the pose sequence. The red "X" is used to remove an image from the pose sequence. This is done by having each image and button collection be a "pose_sequence" object that is then made a child of the Kivy grid layout object. For reasons of speed, it was decided that the reordering would be done by swapping the elements in the child array of the grid layout object, instead of clearing the widgets, and adding the "pose_sequence" objects as children back to the grid layout object in the correct order.

Once the user is satisfied with the name of the pose and the order of the images the user hits the green "confirm" button at the bottom of the screen. This will then start the following process:

1. A new directory in the "user_poses" directory will be created with the following name: "{number of the custom pose sequence} - {pose sequence name input by user in text box}"

2. The image files of the pose/s the user wants to store will be uploaded into the new directory mentioned previously

3. A python function will feed the images the user uploaded into OpenPose for processing to pick out the poses' key points coordinates

4. OpenPose will finish processing and upload JSON files of all of the poses' key points into the new directory created in step 1

The uploaded pose sequence is now integrated with the rest of our app. The user can now see their custom pose sequence in the "Pose Selection" screen, under the "Uploaded Custom Poses" page, ready for them to train.

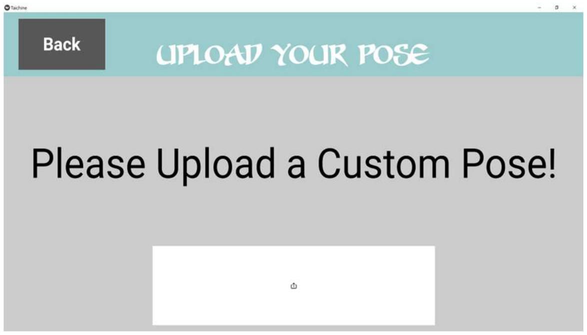


Fig. 7A. Screenshot of UI for upload pose/pose sequence

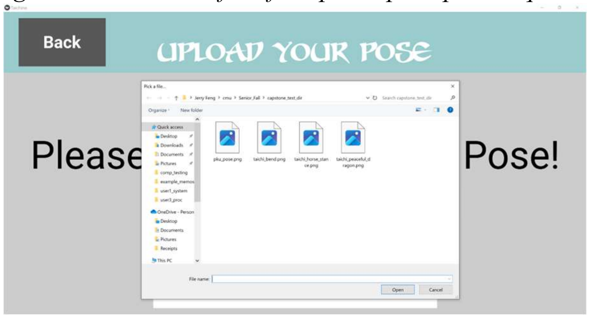


Fig. 7B. Screenshot of UI for upload pose/pose sequence with native windows file browser

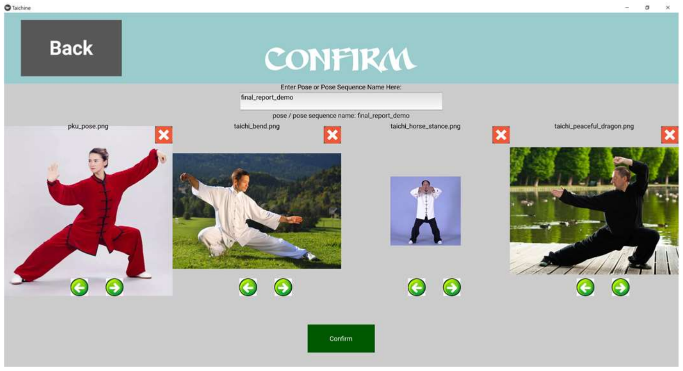

Taichine
Back
CONFIRM
Enter Pose or Pose Sequence Name Here:
final_report_demo
pose / pose sequence name: final_report_demo
pku_pose.png
taichi_bend.png
taichi_horse_stance.png
taichi_peaceful_dragon.png
Confirm

Fig. 7C. Screenshot of UI for confirmation of pose/pose sequence to upload to application

VII. Test, Verification and Validation

In this section, we will introduce the test and verification procedure that we conducted to verify the performance of our system and algorithm as well as the usability of the training against real users.

A. *Modular Tests for Angle Calculation*

The Degree Difference Modular Tests will target the comparison algorithm of the system. The goal of these tests is to ensure that the deviation of system detected angles from the actual user body angles is within 5 degrees. For instance, if the angle between that arm and body of the user is 20 degrees, the system should report angle to be between 15 to 25 degrees. Each test will consist of a reference skeleton (body part coordinates derived from a reference pose picture) and a test skeleton. The test skeleton will be generated from a photo of one of the team members or users doing the reference pose via OpenPose. Then for each detected angle that we care about, we manually determine the best actual angle from human decision and compare it to the reported angle from the system in the backend. Fig. 8 shows a single example of how we determine the actual angle.

In total, we tested on 20 different images, and for each image our system generated 16 key angles used for pose comparison, which in total created 320 unit test cases for angle calculation. From the statistics we gather, we are 95% confident that the angle error is within 5 degrees, which matches the requirement of 10 degrees as desired. Since the tolerance is at least 5 degrees in the system choices, we also can say that we are 95% confident that if the user meets the reference, we will definitely report a success.

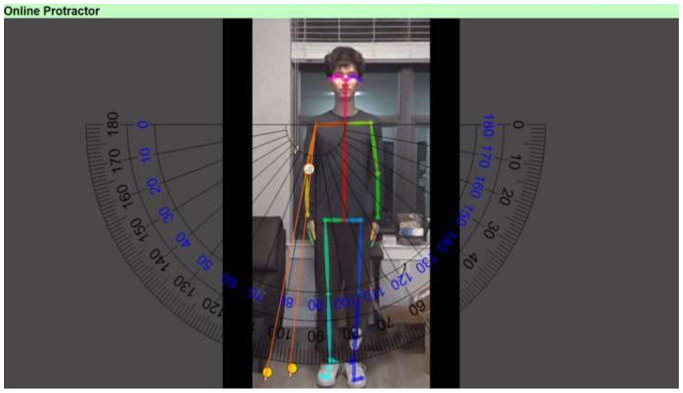


Fig. 8. Angle measurement example

B. *Modular Tests for Error Body Part*

The Error Body part tests are designed to assess the system's capability in detecting and evaluating errors related to Tai Chi postures, particularly instances where a user's limbs deviate beyond a specified threshold of degrees. This test focuses on the system's ability to analyze body positions during designated Tai Chi sequences and how the system provides correct responses. When a deviance is detected in users posture, we expect the system to respond in the following manner: report the error body part correctly, identify the deviance in degrees and if the degree is beyond a certain error threshold, pass on the specified limb and difference in angle for further processing.

As shown in Fig. 9, we verified the test results on our frontend UI by seeing on the skeleton picture drawn that the user has the limb identified as red, hearing the angle reported by the voice engine matches raw output on the backend, and having the angle difference measured as mentioned in subsection A. Combined with the test result from part A, we are able to say that the system is 95% confident to capture errors with more than 5 degrees off from the reference poses. Most of the time, the user would be even further from the reference pose if they have not fully learned the pose, in which case the system would be even more confident to catch the error.

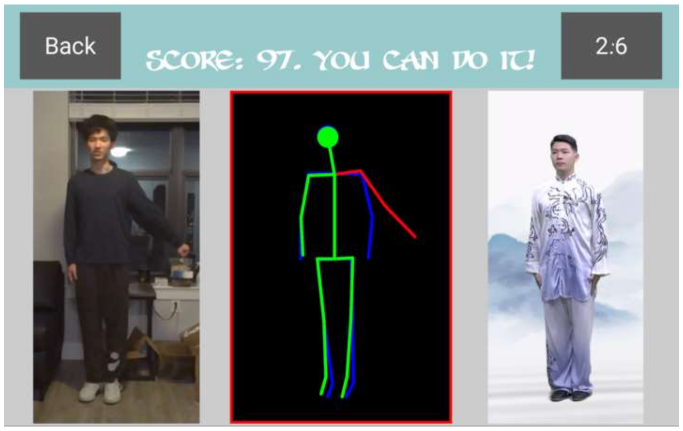


Fig. 9. Intentional error detection test

C. Modular Tests for Error Priority

The Error Priority tests aim to grant assurance in functionality of the system when multiple errors are detected in one capture of the user's Tai Chi posture. The system is prioritized to recognize lower body part errors of the sequence in the following order: starting with the feet, moving to the calf, then progressing to the thigh, followed by the waist, torso, upper arm, and finally, the lower arm and the head. After identifying the prioritized sequence for the error, the system should also behave in the same scope as mentioned in the subsection A and B in terms of identifying the degree differences and the incorrect limb for further processing of the error body part.

We are 100% confident that if multiple errors are detected, the system behaves as expected to report the error on the lowest body part first. Fig. 10 shows an example of how multiple errors could happen, and how the visual display demonstrates it.

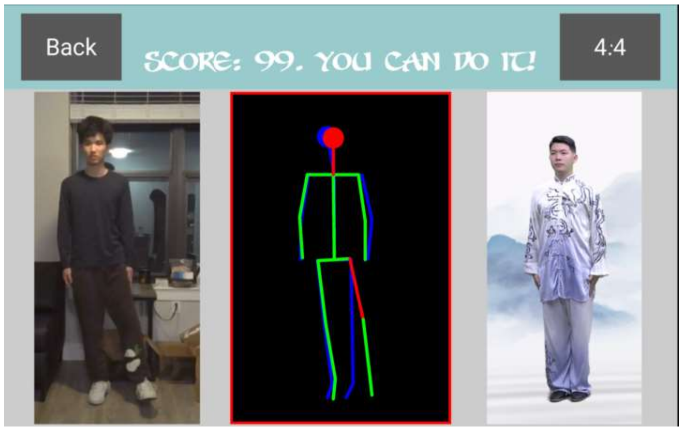


Fig. 10. Multiple pose errors example

D. Modular Tests for Environmental Lighting

The Environmental Lighting tests aim to understand the system performance in low lighting situations in terms of detecting persons and limbs. This performance is also limited to the ability of the camera the user aim to use,

Fig. 11 shows that during low light settings, our system, specifically OpenPose, is unable to detect the person in frame and give instructions on deviances in limbs. In this case, the system will treat as if there is no one in the frame and proceed to evaluate the next captured picture. This is due to the limitations of OpenPose and the webcam quality of the user's computer. We assume that the user would actively change the lighting environment since too much brightness or darkness would basically disable their own visual system.

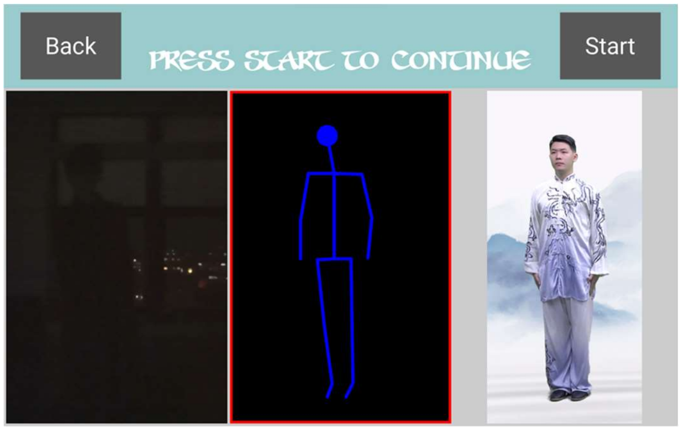


Fig 11. System running in pitch-dark environment

E. Modular Tests for User Selection

The User Selection tests aim to have the system choose the correct user in frame. We assume that the user would be the one in frame who tries the hardest to produce the reference pose, and thus should be the closest to the reference pose. The system thus achieves this functionality by always selecting the person with the most similar posture to the reference pose. The backend evaluates every person with complete posture and chooses the one with the highest similarity to be passed for further evaluation.

Fig. 12 shows how we conduct the test. We intentionally let two humans get fully detected by the system with the left one closer to the standing pose, while the right one is wrong on the arms. We verify the result is correct by checking which user pose gets displayed as the skeleton in the front end. In Fig. 12, we can see that the standing pose is chosen by the user as expected. The accuracy of this functionality is 100% when only 2 people are tested for different poses. However, when the system is used in public with 5-10 potential bodies captured by the camera and passed to OpenPose, the accuracy is significantly compromised as OpenPose is less reliable in that setting. Thus, we still highly recommend the users to use the system with fewer people in the camera, but without the need to worry about a single person passing by like family members.

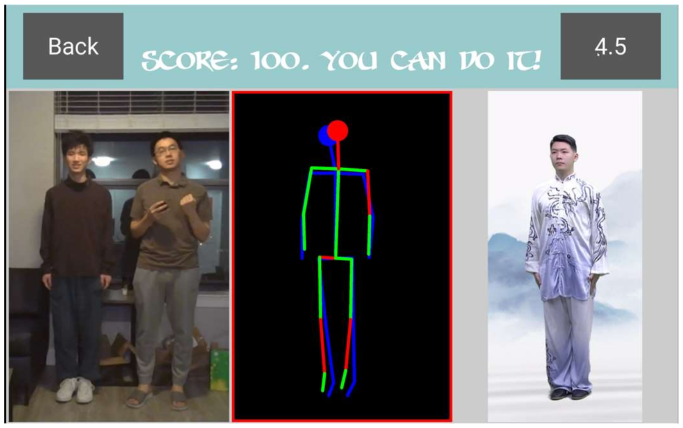


Fig. 12. User selection test outcome

F. *Modular Tests for Verbal Instructions*

The Verbal Instruction Modular Tests will ensure that the correct verbal instructions text is output from the speaker. We conducted the test by manually feeding instruction messages to the text to speech application and listening to the output. The accuracy is 100% as expected since we are using a fully built text-to-speech module in Python. One concern is the limitation of the laptop speaker volume when loud background noise persists and makes it hard to hear from a distance of 5-6 feet, for example, during a public demo session where multiple groups and various people are having discussions. We prefer the user to perform Tai Chi in a relatively quieter environment as mentioned in Section IV, but thanks to our vocal and visual approach, users will still be able to see the limbs color correctly and follow visual instructions if that could not be achieved.

G. *Modular Tests for Customization Pipeline*

The customization pipeline modular test aims to check that all user uploaded images can be correctly uploaded, ordered and processed by OpenPose to generate necessary files for actual training. The accuracy is 100% as expected. Note that we do not check in the background whether the user is uploading a valid image or not (i.e. an image without even a human in it). Thus, the user needs to choose their input wisely.

After the whole system is integrated, we also conducted the following overall system tests to gain performance parameters and tests the functionality workflow are working as expected. We will also include the result for user feedback and the ability of pose instruction here.

H. *Overall Tests on Training Loop Functionality*

This test aims to show the capability of training a sequence of poses and automatically move on to the next pose when the user passes the current one. It also tests the move to a train summary page after the end of the sequence. The test requires backend information of the user training result and correctness, and the front end adaptability to different sequence length and capability of showing the correct image or page after each iteration. The accuracy for this functionality is 100% as expected, and the exact way of using the training loop can be found in the project video.

I. *Overall Tests on Verbal Instruction Latency*

Verbal Instruction Latency tests aim to measure the duration in the training loop between the second the user image is captured and the second verbal instruction actually starts to speak. The duration basically represents waiting time for each evaluation cycle until some intuitive instructions are generated. We inserted timing code in the system to get the metrics.

The final waiting time is 3.6 seconds which is lower than the 4 seconds requirement of the system as expected. To break down the waiting time for understanding the bottleneck, we found that the majority (3.1 seconds) is occupied by OpenPose runtime. The rest represents the time used for angle computation, pose comparison and instruction speech generation.

J. *Overall Frontend Verification*

The overall frontend verification aims to test that all the front end UI buttons, page changes and widgets are displayed and working properly including the pop up file explorer window for custom pose upload. The accuracy is 100% as expected.

Here in table II, before we move on to the user investigation for feedback on the usability of the system, we would like to summarize the systematic tests introduced so far and aggregate the results here.

Metric	**Desired**	**Actual**
User Wait Time for verbal feedback	4s	3.6s
Angle Accuracy	90%	95%
Angle Confidence Interval	10 degrees	5 degrees

Frontend Accuracy	100%	100%
Verbal Feedback Accuracy	100%	100%
Error Priority Accuracy	100%	100%
User Selection Accuracy (*Under normal private environment*)	100%	100%

Table II. Final System Metrics

K. *User Investigation*

We did two rounds of user testing to gather feedback about our application. For the first round of user testing, 5 users were asked to perform the "commence form" and "repulse the monkey" pose sequences using our application. Users were asked to perform the pose sequences with a 15 degree and 20 degree tolerance, and then asked about their preferred tolerance. Preparation time and move-on time were set at 15 seconds and 5 seconds by default, however users were instructed that they were free to change these settings as they desired. Users then filled out a google form asking for usability of UI, speed of the app, usefulness of voice generated instructions, the usefulness of the score, optimal move-on time, optimal tolerance, and other miscellaneous feedback and comments about how we might be able to improve our application. Unfortunately, one user neglected to answer the questions on the usefulness of voice generated instructions and score, leading to us only having 4 responses for those questions. For the questions about usability of UI, speed of the app, and usefulness of voice generated instructions, and usefulness of the score; users were asked to score the application out of 5, with a score closer to 1 indicating a more negative view, and a score closer to 5 indicating a more positive view.

As can be seen in Fig. 13A, we found that users found our app's UI fairly usable, with an average user score of around 3.8. As can be seen in Fig. 13B, found that users found the speed of our application somewhat lacking with an average user score of around a 3, however this is in part due to the user testing being done on a laptop that was not compatible with CUDA, so OpenPose had to be configured to run on CPU which is slower than being run on a GPU with CUDA. As can be seen in Fig. 13C, we found that users thought the voice generated instructions were moderately useful, with an average user score of 3.25, with most users finding the instructions about the feet being unhelpful and not reflective of their actual feet position. The instructions for the feet position were adjusted in the algorithm to be more lax before the final demo. As can be seen in Fig. 13D, we found that users only found the score to be moderately useful as well, with an average user score of 3.25. As shown in Fig. 13E, we found that 80% of users preferred 20 degree tolerance over 15 degrees, however (as shown in Fig. 13F) there was little consensus on move-on time, with 60% of users preferring a move-on time between 5 and 10 seconds and the remaining 40% of users preferring a move-on time of 15 seconds.

In the second round of user testing, 3 users were asked to practice the "repulse the monkey" pose sequence with a video that we used for our reference poses for 5 minutes, and then at the end of the 5 minutes a picture of the user doing the first pose in the "repulse the monkey" pose sequence would be taken. Users were then asked to practice the "repulse the monkey" pose sequence with our application for 5 minutes, with our application modified to store the latest user pose (users were notified of this and pictures of the users' pose was deleted after data analysis was complete). At the end of the 5 minutes of training with our application, users were asked to try the first pose one more time and this is the final pose we would take. The photo of the user doing the pose without our application was input into OpenPose, and the json file of the user's key points coordinates generated by OpenPose was then fed into our comparison algorithm along with the json file of key points coordinates of the reference pose which would give us a similarity score for all the joints we use in our application and an average similarity score. The same process was done with the json file of the user's key points coordinates generated when they practiced with our application for 5 minutes.

It was found that the average similarity score (similarity scores for all limbs and all users summed and divided by number of users multiplied by number of limbs) was slightly higher when the user's trained without our system (cosine similarity score is higher by 0.0026), however when we broke the similarity scores down by specific joints it was found that for most joints users showed a higher similarity score when using our application, there was just one notable outlier for the angle between the right lower leg (below the knee) and right foot that skewed the data, and when this coordinate was removed, users showed a higher average similarity score using our app (cosine similarity score using our app is 0.071). We believe that this outlier is simply due to our small sample size of 3 users and were we to test with more users for longer, this effect would be mitigated and the average similarity score for users practicing with our app would be higher than the similarity score for users practicing without our app.

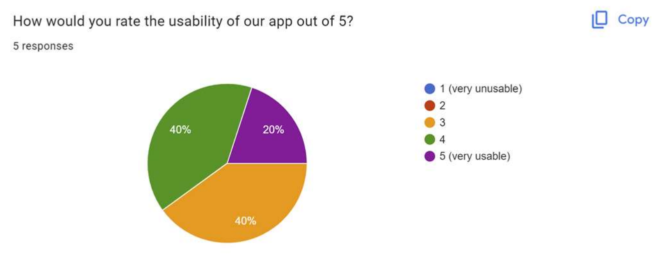
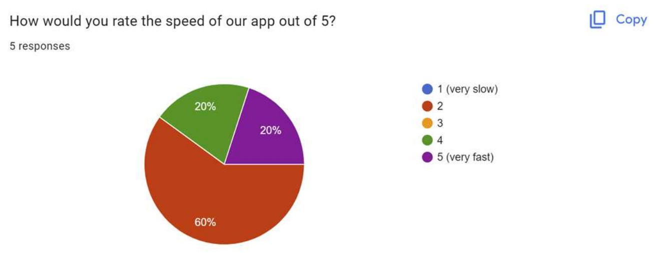
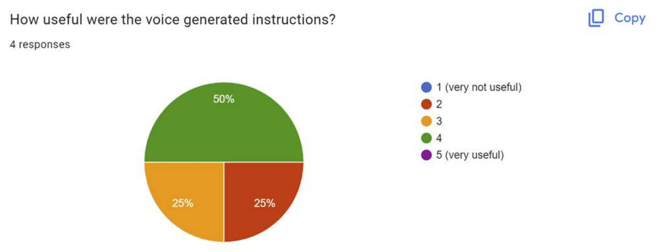


Fig. 13A: User scores for usability of app


How would you rate the speed of our app out of 5?

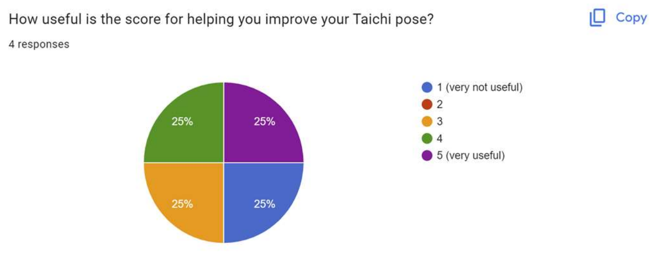
5 responses

Fig. 13B: User scores for speed of app


How useful were the voice generated instructions?

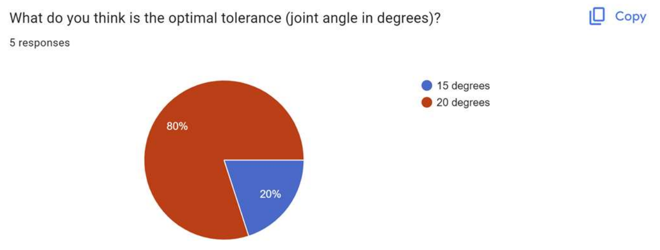
4 responses

Fig. 13C: User scores for usefulness of voice generated instructions


How useful is the score for helping you improve your Taichi pose?

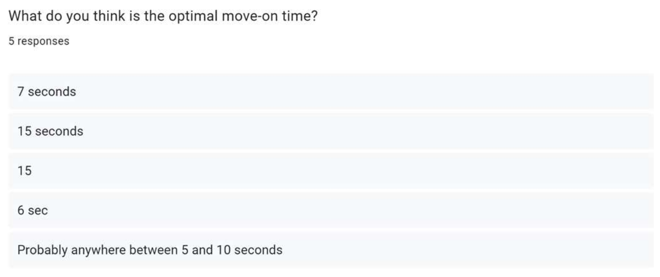
4 responses

Fig. 13D: User scores for usefulness of score in training page


What do you think is the optimal tolerance (joint angle in degrees)?

5 responses

Fig. 13E: User selection of optimal tolerance for joint angles in degrees


What do you think is the optimal move-on time?

5 responses

7 seconds

15 seconds

15

6 sec

Probably anywhere between 5 and 10 seconds

Fig. 13F: User preferences for the optimal move-on time

VIII. Project Management

A. Schedule

As a team, we cooperate throughout the way to work on the parts assigned as in design documents. After we have gained sufficient progress on each of our subsystems, we cooperate in multiple different ways for different purposes. Hongzhe and Shiheng worked primarily on updating the backend infrastructure, enabling and supporting new features from OpenPose and using them in comparison. Jerry and Sirui worked together for the whole frontend architecture and learned from each other's experiences on the Kivy library. We also work together to determine the final training loop and display logic in order to provide the best experience and enough information for the user to practice.

In terms of extra events in the schedule, we used much more time on integration than expected over multiple different system bugs and new features we tried to add in. The result came back positive since we were finally able to enable the pose sequencing feature which was not in the MVP goal of the project. We also split up the work for testing and verification on the real system testing and user investigation. Overall, we as a team were able to cooperate and communicate efficiently that leads to the accomplishment of this project, as can be seen in our Gantt chart in Fig. 14 in appendix.

B. Team Member Responsibilities

Name	Responsibility	Input	Output
Hongzhe Cheng	OpenPose Usage, Integration, Metric Testing	User pose image passed by application	JSON file containing coordinates
Sirui Huang	Real-time instruction pipeline and application infrastructure	User real-time video captured by camera	Cut video into frames. Interact with OpenPose to get real-time coordinates, pass to user selection and comparison
Shiheng Wu	User selection and pose comparison algorithm, Speech Application, Backend Verification	Coordinates output from OpenPose	User Selection, Error Detection, mp3 files of instruction played to the user
Jerry Feng	Custom poses pipeline and file storage infrastructure, User testing	User uploaded image/sequence of images	JSON file and image storage

Table III. Team B4 Work Division

C. Bill of Materials and Budget

After careful consideration, we decided not to order any materials externally but decided just to use a laptop as our platform for our project.

For normal laptop cameras, they usually already have over 30fps and 720p+ resolution, which already exceeds the minimum requirements that OpenPose requires (15 fps). In addition, they are usually embedded inside the system that are default setup with video drivers. As Fig. X shown in the Section V, both pictures are taken using the integrated cameras on different laptops which achieve the posture detection goal perfectly with excess fps for the system to process.

We initially also considered using a Logitech webcam as a usb-addon to the laptop for better capture qualities. After testing its effect based on the one that Shiheng owns, we figured out that the differences between image and video capture by laptop cameras are not significant and laptop cameras already fulfilled our requirements. We also accounted for the fact that different laptops have different USB ports supports (e.g. Macbook and mainstream Ultrabooks lack USB-A ports) and it's hard to find a one-size fit-all camera which we could not extensively test on all laptops considering compatibility issues with video drivers and hardware requirements.

For other parts of our program, we concluded that a laptop already integrated all the functionality we require, and purchasing of services like cloud servers or computing devices are not necessary. Most of the work is developing software systems, processing video inputs, and giving feedback locally without the need of hardware and cloud service requirements.

With all the factors mentioned above, we decided to stay with the idea of developing our project on laptops alone and rejected the idea of purchasing an extra camera for pose recognition purposes.

D. Risk Mitigation Plans

One Risk that we face is the fact that OpenPose is a black box system. OpenPose has its own internal accuracy and might generate invalid data by miscategorizing body parts or returning bad coordinates. Since we are not defining our own pose recognition model, we might try to add some data filtering to sort out the coordinates that seem inaccurate and unusable, based on coordinates and distances.

Another risk is that we assume that the angle representation of poses would be acceptable for most of the postures. The most statistically significant way is to gather a huge dataset of people with different heights and weights doing the same pose, and create a model that maps body sizes to the best angle for users to practice. However, this requires a significant amount of data collection, calculation and even the necessities of professional Tai Chi players of different body sizes. Thus, we are not taking this approach for now.

IX. Ethical Issues

During the development of Taichine, we have majorly considered two ethical issues related to our users.

The first issue is how to maintain the privacy of users while still getting the appropriate data for the system to work as expected. The project we are building requires camera access and human image storage to enable the pose processing feature of the pipeline. We do not want to leak the user information to the outside or other malicious users. In order to protect privacy, we first made the decision not to create a web application online with backend server and cloud pose storage. Even though this would make the development experience more preferable as there are far more existing frameworks and services for online and cloud systems, it creates the risk that if insufficient data protection or encryption are used, user information would be leaked. At the same time, we enabled flash local storage of the user image. Instead of taking a new snapshot and storing it as a different local file, the system would delete the previous image once a new one is generated from the next training iteration. This ensures that there will not be a local image still existing after the user finishes using the application.Users also store their custom uploaded poses inside the same directory as the application, in case they want to uninstall the whole application, their custom information will be wiped as well.

The second issue is how to ensure the usability of the application to users with limited mobility or certain body disabilities. Specifically since Tai Chi is an exercise among seniors in China, we would expect that the user age also varies in a wide range. We do not want the application to be too strict a virtual instructor that forces every user to match exactly the reference pose 100%. We hope that each user could get a tailored learning experience given their own body situation and expectation of learning. Thus, we enabled a tunable tolerance parameter so that the user could tune to get a more lenient training experience.

X. Related Work

OpenPose – underlying pose recognition software

Virtual Yoga Coach – instruct Yoga based on OpenPose comparison, but without customization

XI. Summary

To summarize, Taichine is a piece of software and application that enables and enhances the accessibility of pose training in general with a default emphasis on Tai Chi poses. The system supports one pipeline for real-time pose recognition and instruction feedback for pose practice. It also contains the other pipeline allowing custom reference poses to be uploaded and processed.

Our major challenge would be enabling the OpenPose software to work, ensuring the connectivity between our infrastructure and the OpenPose algorithm module. Meanwhile, we need to ensure the pipeline and data flow works well, instructions are generated with correct and informative content, and create an easy-to-use interface and setup environment for the user.

For future work, we plan to accomplish the various stretch goals for the project. One of our stretch goals was to use 3D OpenPose, and have the user's pose skeleton and reference pose skeleton that is drawn in the training screen be 3 dimensional. We think this would provide even better feedback to the user as the feedback would then be able to account for depth in the image, and provide users an idea of how to place their limbs in the pose in terms of distance from the laptop they are using. We also could show a real-time video of the user's pose skeleton, as interpreted by OpenPose to be displayed on the screen and do our pose recognition and checking with a video of the user's pose and a fixed image of the reference pose, as opposed to just using an image of the user and an image of the reference pose.

We learned a lot about project management and ethics over the course of our project, which forced us to think in new ways about our project. One thing we relied very heavily upon to complete our project was the importance of slack time, as we used up basically all of our slack time to finish the implementation of our project. In terms of ethical issues, we were forced to consider the ways our project could be misused and harm our users, whether it be physically or violate their privacy. As a result, we made changes to our project: such as adding in an adjustable tolerance and deleting photos of the user after they were used for processing.

Overall, we learned a lot about the ethical responsibilities we had to ensure our product was an ethical contribution to society and to the wellbeing of our users, and would like to add functionalities to enhance the user experience of our project in the future.

References

[1] Fenneld. (2023, September 6). "Slow and steady: The health benefits of Tai Chi." Cleveland Clinic. https://health.clevelandclinic.org/the-health-benefits-of-tai-chi/

[2] Debnath, B., O'Brien, M., Yamaguchi, M. et al. A review of computer vision-based approaches for physical rehabilitation and assessment. Multimedia Systems 28, 209–239 (2022). https://doi.org/10.1007/s00530-021-00815-4

[3] A. D. Gama, T. Chaves, L. Figueiredo and V. Teichrieb, "Guidance and Movement Correction Based on Therapeutics Movements for Motor Rehabilitation Support Systems," 2012 14th Symposium on Virtual and Augmented Reality, Rio de Janeiro, Brazil, 2012, pp. 191-200, doi: 10.1109/SVR.2012.15.

[4] Pedersen, Jannik & Jensen, Niklas & Lahrissi, Jonas & Hansen, Mikkel & Staalbo, Patrick & Wulff-Abramsson, Andreas & Sander, Mattias. (2019). Improving the Accuracy of Intelligent Pose Estimation Systems Through Low Level Image Processing Operations.

[5] Danielle Bragg, Cynthia Bennett, Katharina Reinecke, and Richard Ladner. 2018. A Large Inclusive Study of Human Listening Rates. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). Association for Computing Machinery, New York, NY, USA, Paper 444, 1–12. https://doi.org/10.1145/3173574.3174018

[6] Woojoo Kim, Jaeho Sung, Daniel Saakes, Chunxi Huang, Shuping Xiong, Ergonomic postural assessment using a new open-source human pose estimation technology (OpenPose), International Journal of Industrial Ergonomics,Volume 84,2021,103164,ISSN 0169-8141,https://doi.org/10.1016/j.ergon.2021.103164.

[7] Apple MacBook Pro 13" (4th Gen) Dimensions & Drawings. RSS. https://www.dimensions.com/element/apple-macbook-pro-13-inch-4th-generation

[8] Edward P. Washabaugh, Thanikai Adhithiyan Shanmugam, Rajiv Ranganathan, Chandramouli Krishnan, Comparing the accuracy of open-source pose estimation methods for measuring gait kinematics, Gait & Posture, Volume 97, 2022, Pages 188-195, ISSN 0966-6362, https://doi.org/10.1016/j.gaitpost.2022.08.008.

[9] Washabaugh, E.P., Shanmugam, T.A., Ranganathan, R., Krishnan, C. (2022). "Comparing the accuracy of open-source pose estimation methods for measuring gait kinematics." Gait & Posture, 97, 188-195. https://doi.org/10.1016/j.gaitpost.2022.08.008.

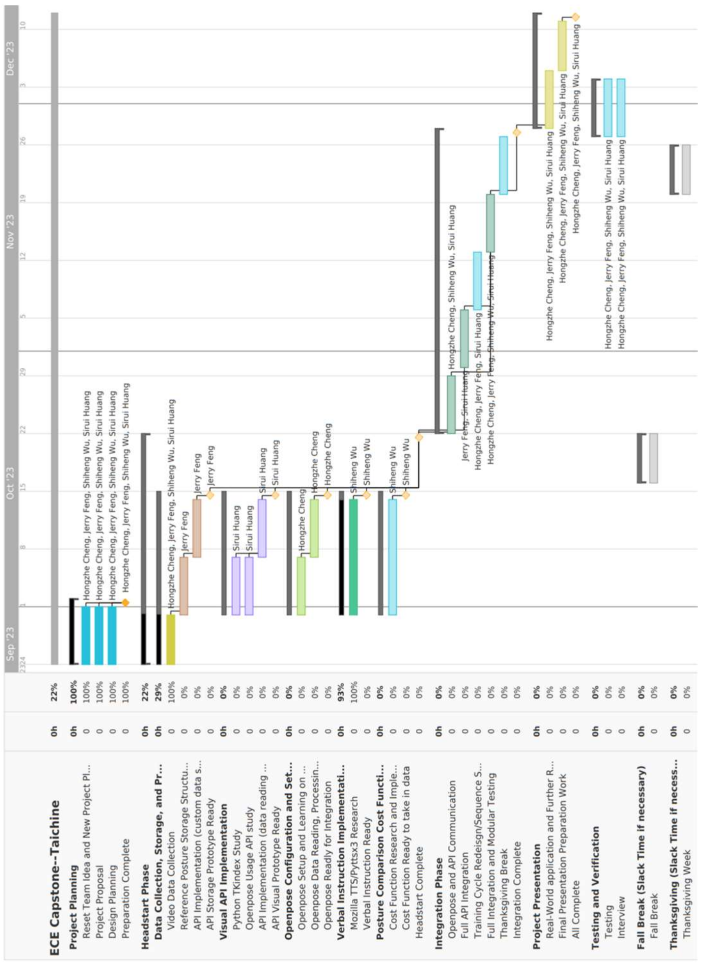
[10] Washabaugh et al, (2022). "Comparing the accuracy of open-source pose estimation methods for measuring gait kinematics." Gait & Posture, 97, 188-195. https://doi.org/10.1016/j.gaitpost.2022.08.008.

[11] Julia Cambre, Jessica Colnago, Jim Maddock, Janice Tsai, and Jofish Kaye. 2020. Choice of Voices: A Large-Scale Evaluation of Text-to-Speech Voice Quality for Long-Form Content. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376789.

[12] Strawberry K. Gatts, Marjorie Hines Woollacott, How Tai Chi improves balance: Biomechanics of recovery to a walking slip in impaired seniors, Gait & Posture, Volume 25, Issue 2, 2007,Pages 205-214, ISSN 0966-6362, https://doi.org/10.1016/j.gaitpost.2006.03.011.

[13] Wayne PM, Berkowitz DL, Litrownik DE, Buring JE, Yeh GY. What do we really know about the safety of tai chi?: A systematic review of adverse event reports in randomized trials. Arch Phys Med Rehabil. 2014 Dec;95(12):2470-83. doi: 10.1016/j.apmr.2014.05.005. Epub 2014 May 27. PMID: 24878398; PMCID: PMC4499469.

[14] What's new in Python 3.12. (n.d.). Python Documentation. https://docs.python.org/3/whatsnew/3.12.html#summary-release-highlights



Task	Hours	Progress
ECE Capstone--Taichine	0h	22%
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	22%
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	93%
Mozilla TTS/Pyttsx3 Research	0	100%
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%
Openpose and API Communication	0	0%
Full API Integration	0	0%
Training Cycle Redeisgn/Sequence S...	0	0%
Full Integration and Modular Testing	0	0%
Thanksgiving Break	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%
Testing and Verification	0h	0%
Testing	0	0%
Interview	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%

Fig 14. Gantt Chart Schedule

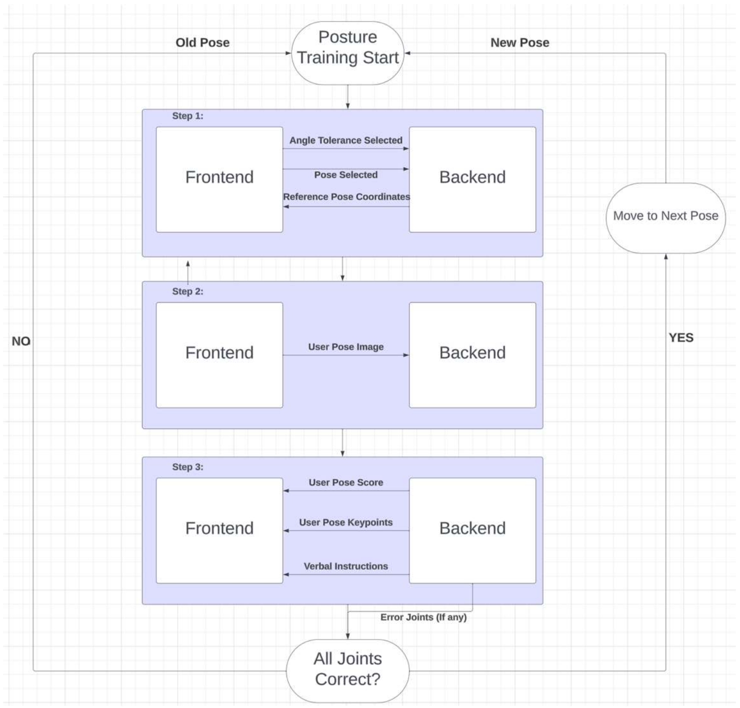


Fig 15. High Level Front-end Back-end Interaction Diagram