TaiChine: Body Posture Guidance Tool for Beginners in Tai Chi

Hongzhe Cheng, Sirui Huang, Jerry Feng, Shiheng Wu

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of comparing human poses against professional Tai Chi poses and generating informative verbal feedback for users to practice and learn Tai Chi easier on their own. The system is also capable of allowing users to upload custom reference poses as images for customized targets to practice which could extend beyond the scope of only Tai Chi.**

***Index Terms*—Machine Learning, OpenPose, Posture Detection, Tai Chi**

I. Introduction

TAI Chi is a Chinese martial art that focuses on slow movements and holding certain poses. Tai Chi has become popular as a form of exercise, especially among the elderly, in our modern day. Tai Chi has numerous benefits to health such as helping to reduce stress, improve balance, and relieve pain from knee osteoarthritis [1]. However, Tai Chi is difficult to learn as a beginner, and if performed incorrectly, practitioners will not be able to receive all the health benefits Tai Chi has to offer and may even result in injuries. To overcome this barrier among beginner practitioners, our team aims to develop a computer application that utilizes state of the art machine learning to help beginner practitioners correct their Tai Chi postures. Our application will take in video input from the practitioner and run the Openpose deep learning software on the video input. Our system will then compare the user's poses with a professional Tai Chi practitioner's pose on the "Golden Rule" Tai Chi poses. A scoring algorithm will then calculate a rating for how accurate the user's pose is based on limb angles. If the limb angles are outside of our tolerance the system will give audio and visual feedback to the user on how to adjust their pose to be more accurate. Our system will have pre-programmed poses it can rate users on, however our system will also support users uploading images of custom poses not in our system that they would like to practice. Our system will also be able to support a sequence of poses as well, with the system only letting the user move on to the next pose after the user scores above a certain threshold on the current pose.

II. Use-Case Requirements

Our system has several features critical to benefiting users: pose comparison, corrective verbal instructions, and an easy-to-use interface. For pose comparison we want a 90% accuracy of being able to properly identify key points of the user's body (knees, shoulders, elbows, etc.) and user's pose correctly. The accuracy requirement of 90% was selected as Pedersen et. al. [2], tested Openpose's ability to correctly identify key points on 300 static image frames in different lighting environments and the lowest accuracy performance in a lit environment was 89.74%, and we do not expect users to use our application in a dark poorly lit environment. Additionally, we have found that using joint angles for pose detection has been used in several studies examining computer-vision based approaches for physical rehabilitation and assessment according to a survey by Debnath et. al. [3]. One of these studies by De Gama et. al. [4] showed a joint angle-based posture detection prototype was able to identify correct movements 100% of the time on a limited sample size under controlled conditions. Although there have not been extensive studies on how accurately joint angle-based pose detection extends to different body proportions, preliminary research we have done leaves us confident in the accuracy of our approach. To ensure users receive timely feedback from our system, there should be a maximum 1 second latency between when the system finishes receiving video input and the audio feedback is provided to the user. We also want to ensure that our application interface is easy to use, so in testing we would want to see users rating our app as an 8/10 on an out of 10 scale on how easy the interface is to use.

III. Architecture and/or Principle of Operation

As shown in the architecture image, our system is entirely based on software. The application in general takes two different types of input from the user and accomplishes the job of either real-time pose recognition and instruction or allowing users to upload and enlarge the reference dataset.

On the top, we can see the first type of input, real-time video, which represents the captured video input of a user practicing the target Tai Chi pose. The raw video data will first be pre-processed by the application to cut into frames and then passed to OpenPose for data processing. OpenPose is a pose recognition software that takes visual input data and returns a set of coordinates representing each "human" detected in the input and the core body coordinates associated with them. Thus, OpenPose generates potentially multiple coordinate sets, and then the pipeline feeds all of them to the user filtering function. The application would load the reference pose and corresponding coordinates now and use that as a helper for user filtering. We highly recommend that only one single user should be in the camera range at a given time. However, in case there are noises or others passing by, we made the assumption that the user would be the most similar to the reference pose compared with other noisy coordinates. Thus, the filtering system works closely with the comparison algorithm that generates evaluation to all captured human bodies and picks the closest to be recognized as the real user. For that given user and pose data, the comparison algorithm will generate meaningful parameters and angle differences if there are any, insert them in verbal instruction templates and finally pass them to the speech application. The verbal instruction provides users with detailed information and where and how they are wrong as “Your left arm needs to be 20 degrees up”. Alongside the verbal feedback, on the very bottom of the diagram, we illustrate that we will also put visual feedback comparing the user and the reference pose. This concludes the real-time pipeline for all the interaction with user input and information that we provide back to the user.

On the other hand, the second arrow pointing from the user to the system shows the customization pipeline. The pipeline allows users to pass in images of poses that they are interested in practicing, and use them as references beyond the default Tai Chi poses that we have built in. The images are required to have only one “human” inside to guarantee the effectiveness of the system. The image will also be passed to OpenPose and receive the only reference coordinate set for the only pose in the image. Then, the application will store the image and coordinates separately into the file system for future usage. This pipeline serves as a backend update option for users. After they upload images, the application will show the custom options next time they enter the real-time instruction pipeline and are choosing poses to practice. At the same time, given that the application processes the general definition of “Poses”, the custom pipeline could also support instruction to other poses that is significant in 2D space (without too many body parts perpendicular to the screen and camera). Examples that the system could easily be extended to include Yoga, Dancing etc.

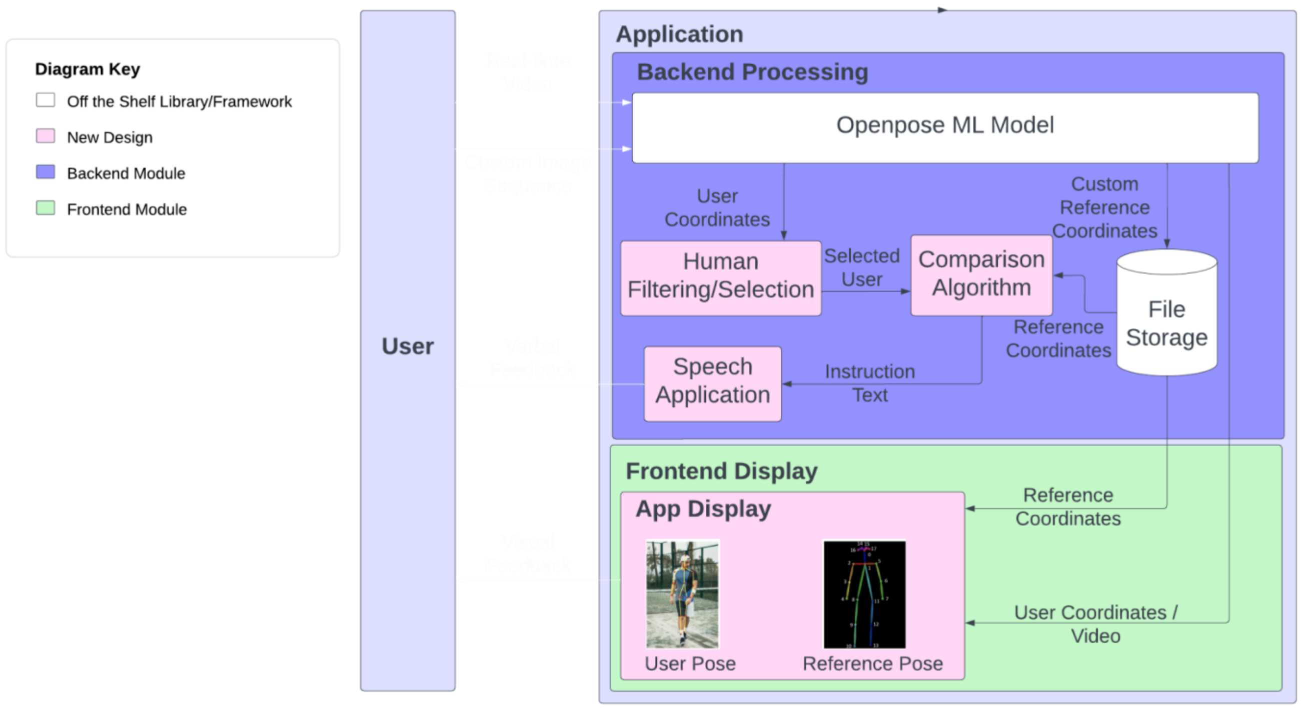


Figure 1: Block diagram for system architecture

IV. Design Requirements

Regarding the input footage, we suggest that a single integrated laptop camera will do the job of detecting body angles and recognizing the user. In setting up the camera, we will introduce a calibration stage for the user to place and calibrate the camera for our algorithm to capture the full body of the user for us to evaluate their Tai Chi postures. We evaluated the performance of OpenPose models on laptops with as low resolution as 720p to higher ones with 1080p, which they turned out achieving the 90% accuracy in detection after we manually adjusted the camera angle and distance of the user from the camera to capture all key points. After calibrating and making sure the user stays inside the camera frame, we aim to control the difference between user body angle and reference angle should be within 10 degrees of the real-world setting. For example, when a user has an actual 20 degree lower in their arm position, our algorithm should detect their arm is lower within the 10–30-degree range.

For recognizing the key points of the user's body, we aim to have at least capture 10 FPS from OpenPose to give real time feedback for users and showcase better visual information for them to improve their Tai Chi postures. We will expect the user to utilize our product in a well-lit place and preferably a flat ground for recognition and safety concerns as we do not aim to provide night vision/low lighting support. We will continue to pass these frames into our comparison algorithm for an evaluation period for at least 5 seconds where at least 50 different frames are passed into our program and around 1000 data points are being evaluated with our reference position. Our team aims to provide voice feedback within 1 second after the evaluation period to point out where the user could improve for a seamless experience of practicing Tai Chi.

In order to achieve the goal of giving voice feedback to human users and offer a smooth learning experience, we considered some requirements in receiving and processing feedback. Using a TTS speaking speed is similar to mean human listening rates around 309 words per minute discovered in a 2018 study [5] and a common instruction is around 10 words (e.g. Move your left arm up twenty degrees), we estimate 2-3 seconds and 2-3 sentences per posture and a total of at most 5 seconds time for TTS to give instructions on the most different limb postures. This design also considers that the user has no prior knowledge of Tai Chi and provides a step-by-step learning curve for them to correct problems in each limb.

To achieve our aim of ease to use on the interface, our UI should also be designed for people familiar with laptops who have basic computer knowledge to use without any additional training required. We aim to use the Kivy framework in Python as our basis for application development on laptops with the widgets and plugins provided from the open-source basis which will allow us to integrate various features and provide an appropriate user interface.

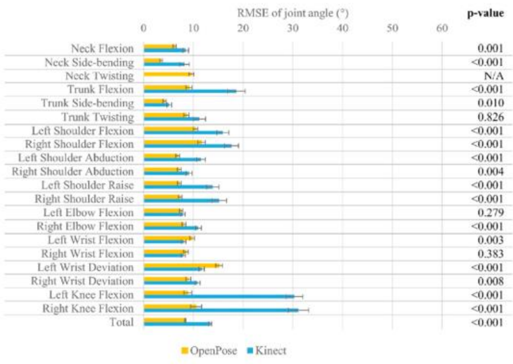


Figure 2: OpenPose Measurement of Joint Angle Error Compared to Older Posture Detection Algorithm Kinect[6]

For our customization pipeline, we prefer poses that are parallel to the screen whether it is a screenshot or picture taken which allows us to label all joints clearly on a 2D scale for best reference in future practices for the user. We also require that the user upload an image with only one human inside doing the desired pose to eliminate the idea of recognizing the wrong posture or misidentification caused by overlapping of the limbs or multiple limbs inside the given picture.

V. Design Trade Studies

A. Decision of Local Application

Our application installed locally will provide faster performance and responsiveness as they leverage the device's hardware and resources more efficiently compared to web apps requiring cloud servers. We understand that having internet access would allow us to improve compatibility, gain access to faster cloud computations and packages that need API support. However, we aim to make the application work offline, ensuring uninterrupted functionality in locations closer to nature, like parks and large outdoor open spaces that are not always equipped with fast and stable internet connection. Furthermore, local applications also provide better data privacy and security, as camera captured data is stored locally, reducing exposure to online threats and insecure connections.

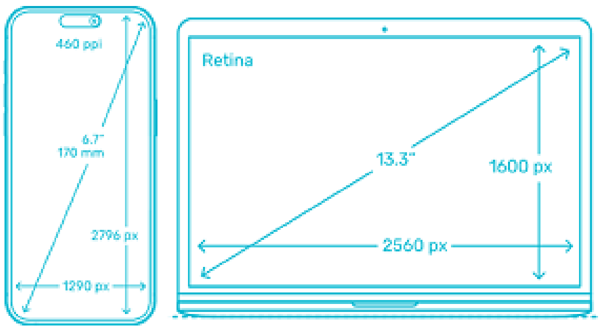


Figure 3: Comparison of 6.7 inch screen to 13.3 inch screen[7]

B. Decision for Laptop Platform

As most people are equipped with smartphones, it naturally comes to mind that developing a mobile application will be more convenient for Tai Chi practitioners. However, when it comes to providing detailed feedback, phone screens will not usually be the best option. Since we aim to output a real-time image of the user, reference image processed by OpenPose, and provide voice guidelines at the same time, a larger screen will be a better option. A laptop screen usually lies between 13 to 17-inches, which is more than 4 times larger than 7-inch smartphone screens. As mentioned in Section A that we would not consider a web application, developing a local application on IOS or Android platform would become the only viable option for phone applications. Unfortunately, the models and engines we aim to use mostly only provide Windows support and limited Linux system support, which left our team deciding the appropriateness of developing a laptop application.

C. OpenPose Posture Model

OpenPose works locally on both GPU (CUDA support) and CPU without occupying too many resources compared to Movenet from Google, which the latter focuses on capturing detailed facial expressions and tracking atypical/fast-moving postures. OpenPose focuses more on the details of joints compared to other models developed based on Tensorflow platforms that excels at capturing quicker motions and gives more weight in capturing upper body instead of more detailed limb movements which we are focused on.

Models like MoveNet and Posenet provide higher FPS and quicker responses to capture quick movements which take significantly more resources, but we sacrificed the efficiency and idea that taking in too many frames will cause pressure for processing on the backend and generate too much useless data points as the user is shifting into position or making adjustments before achieving the pose, causing an underestimation of accuracy and misjudgement of the posture. We utilized OpenPose for the accurate detection of postures and resource utilization concerns.

Due to the nature of Tai Chi, the practitioner will not perform atypically fast movements that require higher FPS from the camera and quick responses from the model. In our case of Tai Chi, we aim to prioritize the posture accuracy from the user where OpenPose performs best in detecting complex postures compared to examples below where MoveNet misclassified the hands' position. OpenPose can handle complex and

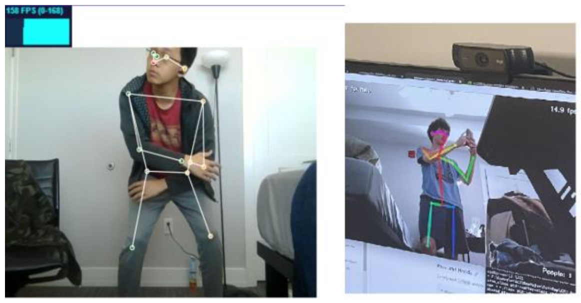


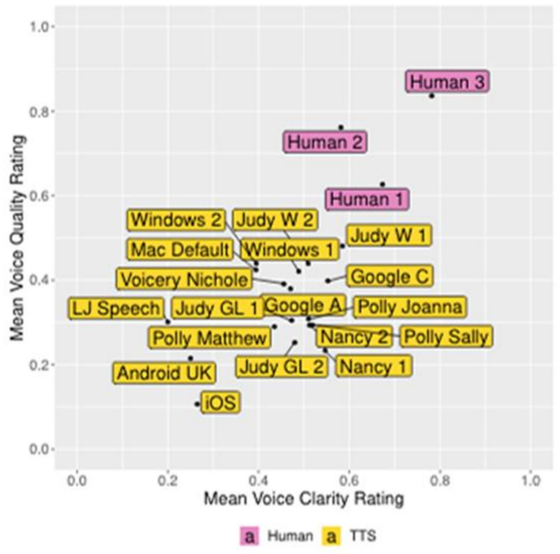
Figure 4: Comparison Posture Example of MoveNet and OpenPose [8][9]

intricate poses, making it valuable for Tai Chi analysis where detailed pose information is required. MoveNet is more geared towards fitness tracking which works great in detecting simple movements like squats and jumps.

TABLE I. POSTURE DETECTION MODEL COMPARISON

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

a.



a.

Figure 5: Comparison of VoiceEngine Clarity and Quality Ratings (Judy represents Mozilla TTS engines) [10]

D. Voice Engine Choice

Mozilla TTS can work without an internet connection with local packages and voice engines. This fits our system design of local applications compared to Google and Amazon products utilizing cloud services. Mozilla TTS is also open-source and we could modify the code locally to cut down on redundancy and improve latency on voice responses, given we will be only picking up specific instructions instead of utilizing a whole package of voice lines for more complex interactions.

While we do not value voice quality that much compared to more high-end applications, we need the instructions to be loud and clear for users to follow, where most Mozilla engines provide clarity ratings compared to average humans as mentioned above. In terms of voice quality, Mozilla engines like Judy W 1/2 outperform Google C engine and provide a near-human experience for users based on a 2020 research conducted through surveys provided to human listeners and evaluation of clarity and quality [11]. While online products like Google and Amazon provide larger possibilities of improvement with AI integrated voiceover, our team considers the smaller scale Mozilla could be more specific on finishing the task locally with lowest resource usage.

E. Comparison Algorithm

We adapted the idea for vector representation in comparing user body postures vs. reference postures presets. OpenPose will return results that represent each body posture as a vector in a high-dimensional space, where each dimension corresponds to a key point. For example, we have 18 key points detected by OpenPose, each posture can be represented as a set of 18-2D vectors.

A · 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.

$$cos(\theta) = (A \cdot B) / (||A|| * ||B||)$$

The Cosine similarity will be between 1 and –1, and mostly lying between 1 and 0 as users will realize from negative values where they are mirroring the posture is the wrong direction. Any results less than 1 will indicate dissimilarity between the user posture and reference position which we could utilize directly to calculate a score on our end to evaluate how the user is doing and decide if a posture is accurately reflected.

Another popular choice we considered is measuring the absolute positions of the user posture and seeing how different the body joints are from the reference posture joint coordinates. This approach makes sure that users could get as close to the reference posture as possible but given that we are not building a person specific application, evaluation of absolute positions will cause people of varied heights and weights to not reach optimized results. For example, a person with shorter limbs will not be able to reach reference points achieved by a taller demonstrator even if they have done the absolute correct posture. In addition, the vector representation approach does not need ground reference if the user attempts to do Tai Chi on an inclined ground which will bias the absolute positions, but the joint angles measured through the vector will be unaffected and allows us to calculate the similarity as usual. We are confident that joint angles should be universal among people of different body sizes based on our testing within the group and this level of abstraction for people should represent similar Tai Chi postures well. As people are approaching the same posture with detailed requirements and a 10-degree tolerance, we believe they should be reaching the same posture result based on joint angles. We have a more detailed discussion on implementing more complex statistical and math models in case joint angles are proven wrong inside our Risk Mitigation section VIII.D.

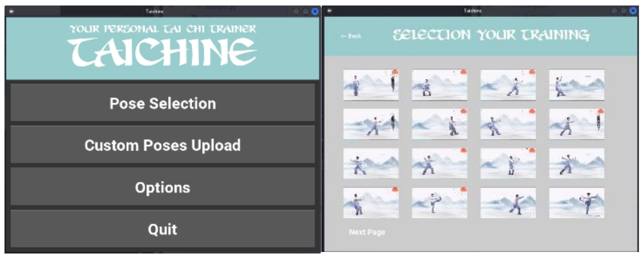


Figure 6. Reference UI for Pose Selection Page in block view

VI. System Implementation

A. Frontend Infrastructure

The app infrastructure will both be the interface our user interacts with the backend evaluation system. The hierarchy of the app infrastructure will be as follows:

1. Main Menu

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

2. Pose (Sequence) Selection Page

The user can reach this page from the main menu 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. Both the default 24-form Tai Chi poses and the user inputted poses will be displayed here. The pose can be displayed in list view or block view. In the list view, the user will be able to see how many poses this sequence contains. If in the block view, the user can see a preview of the pose. Clicking on the pose will bring the user to the training page.

3. Customization Image Upload Page

The Customization Image Upload Page is where the user can upload their selected pose into the system and practice with the system functionalities. There will be instructions helping the user to input their poses, and the user can drag images onto the screen or select from their directories the images they want to input. The inputted images in one upload attempt will be packed as a posture sequence, and the user will be able to switch the order of the poses on this page. However, the order will be fixed once the poses are imported into the Pose Selection Page. The user will be able to delete their custom sequence from the Pose Selection Page.

4. Options Page

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 strictness of the system's pose evaluation with slider control. The volume control will also be on this page.

5. 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 consist of a large screen to the right of the screen showing the user's live video footage captured from their laptop camera. A smaller screen will be on the top-left corner of the screen, displaying the reference posture. To the bottom-left corner, the score of the user's posture will be displayed. When the user's pose has not yet passed the system check, the score text (less than 90) will be displayed in red. When the user has performed the pose accurately, the score (greater than 90) will be displayed in green, and the screen displaying the user's life footage will have a green backlight. The system will proceed to the next pose if the current training is not yet over. If the current posture is the last one, a message box will appear telling the user that they have successfully completed the training, and they will be sent back to the Pose Selection Page after they confirm on the textbox.

The implementation of this hierarchy will be in Kivy, which we consider to be the optimal tool for our system for its abundance in available widgets and support for mobile apps. We have considered using TKinter, PyQt, and WxPython, but considering the complexity of our system and the time restriction on our project, we believe Kivy will be the best option. The widgets Kivy provides will be helpful for implementing all different parts of the frontend infrastructure, and the mobile app support will be handy if we decide to move onto our stretch goal in implementing a mobile app for our system.

As for the implementation details, the page transitions will be implemented with Kivy's Screen Manager functionality, with each screen being a separate page. According to the user inputs, the frontend sends and retrieves relevant data from the backend, such as the strictness 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. If the backend has sent a score higher than 90 for more than 2 seconds, the frontend will proceed to the next pose automatically. Otherwise, the frontend will listen for backend verbal instructions and play it when requested.

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.

1. Calibration Stage

In this stage, the user will be asked to adjust their camera and body positions (front-facing) to capture all 18 data points of their body. The comparison algorithm will continue to receive datasets that are missing certain data points and pass those missing values to the voice engine and produce feedback like 'Please include your left arm' and 'Please include your right foot' until the user keeps their whole body inside the frame for 2 seconds. At this time, the comparison algorithm will pass text string 'Calibration complete, please stay in the camera frame' to the voice engine to signal the user that the calibration is complete and transfer the system into the practice stage.

2. Practice Stage

Upon receiving a package consisting of 18-key points representing the current user's body posture, the Python algorithm written by Shiheng 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 our stored reference position to generate differences between the user's posture and reference posture and generate a 'score' which is only visible to developers. We repeat this process for all frames captured during the evaluation time (5s by default) and calculate an average score to determine the closest average posture we will use to evaluate the user's performance. We grab that image and pass that dataset to generate text, examples like: 'Raise your left arm by 20 degrees', 'Lower your thigh by 10 degrees'. These generated strings will be then passed to the voiceover feedback system. Another case will be if the user performs over an average of 90% similarity to the posture, the system will generate text strings like 'Good job! You've mastered this posture!' and inform 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 Mozilla TTS. After receiving the text string, it generates the corresponding speech waveform matching every instruction generated. This .wav file can then be played back on the default music player of the laptop 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. This system becomes dormant until a new set of text strings is passed in.

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 be able to upload images stored locally on their phone to our application. The application will then store the image in a custom images directory. Next, the application will make a Python API request containing the path to the image to OpenPose. OpenPose will then calculate the coordinates of the key points on the body of the practitioner in the image the user uploaded. These key points coordinates will then be stored in a Python dictionary and input into the backend of the application where an algorithm will calculate the joint angles between the users' limbs. These joint angles will also be stored in a Python dictionary as well. The joint angles and key points coordinates will then be written to a JSON file that will be stored in a separate JSON directory for lookup when the user selects the custom pose to practice in the application. Users can also upload a custom sequence of poses they would like to practice, where they will upload all the images and then order the images with a drag and drop mechanism in the order they are supposed to come in the sequence. These images will all go through the same pipeline as described before, except there will be an extra two fields in the JSON file for the name of the sequence they are a part of and their number order the pose is in the sequence to let the application know what order the poses should come in.

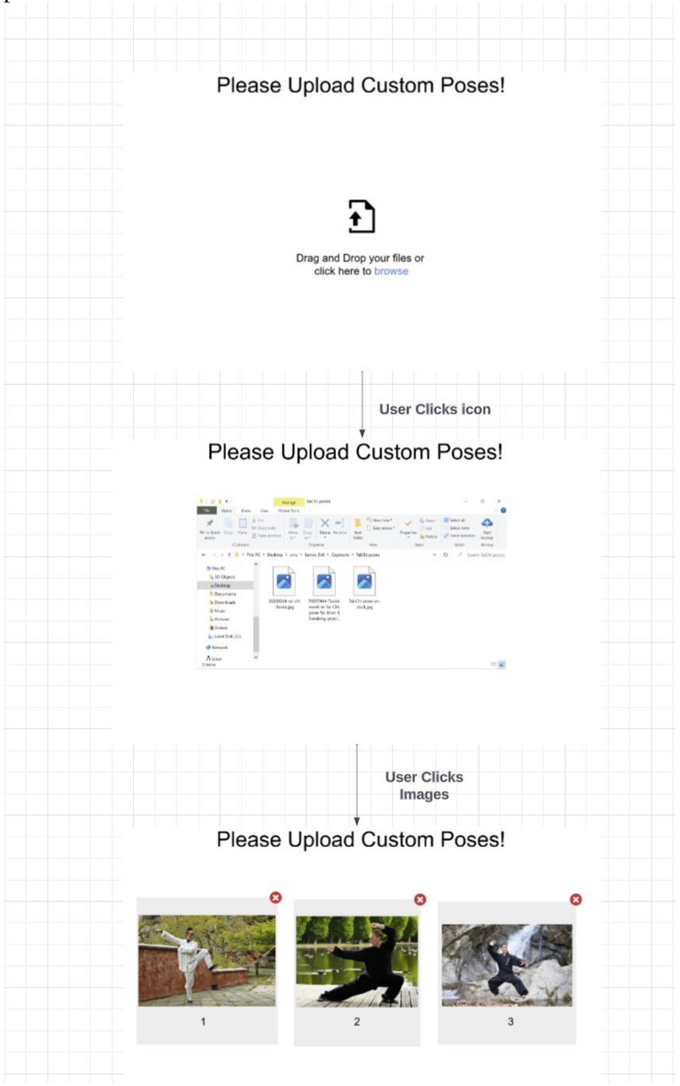

Please Upload Custom Poses!
Drag and Drop your files or click here to browse
User Clicks icon
Please Upload Custom Poses!
User Clicks Images
Please Upload Custom Poses!
1
2
3

Figure 7. Flowchart of user interface interaction to upload image sequence.

VII. Test, Verification and Validation

We will conduct modular and overall tests to evaluate the design implementation. After we have a working prototype, we will be refining the system by conducting the following Modular Tests for examining the system's performance in fulfilling the design requirements:

A. *Modular Tests for Degree Difference*

The Degree Difference Modular Tests will target the comparison algorithm of the system. The goal of these tests is to ensure that the deviation from system detected angles to actual user body angles is 10 degrees maximum. For instance, if the user is 20 degrees lower on their upper arm with respect to their chest, the system should report the arm to be lower by some degree between 10 to 30. The system should consider the user's arm position correct if it is within 10 degrees of the reference angle. 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 doing the reference pose while deliberately positioning one of the body parts incorrectly. All body parts will have at least three tests, one with a higher angle than the reference, one with a lower angle, and one with the correct angle.

B. *Modular Tests for FPS*

The FPS Modular Tests focus on the OpenPose Module. This test involves inputting videos of different resolutions and poses to OpenPose. The goal of these tests is to make sure the FPS of the video output always meets the required 10fps. Openpose will take in test videos and team members will manually inspect Openpose's fps. The test videos will be team members doing Tai Chi poses shot in different resolutions. Since we do not expect users to input videos or images with a resolution lower than 360p (lowest resolution for most video platforms/screenshots) or higher than 2160p (highest resolution for most video platforms/phone screenshots), we will be testing the system with videos of these two resolutions. We plan to use 10 random poses for this test.

C. *Modular Tests for Verbal Instructions*

The Verbal Instruction Modular Tests will ensure that the verbal instructions provided by the system have reasonable latency and clarity. The verbal instruction system will be provided with pairs of photos, one reference pose and one incorrect pose, and is expected to output instructions within 5 seconds consistently. The lengths of the verbal instructions will be manually evaluated.

D. *Modular Tests for UI*

For the UI, we plan to stick with functionality tests during the prototyping process. The tests will simply be trying to access each of the poses through the implemented UI. Also, the intuitiveness of the UI will be evaluated manually.

E. Modular Tests for Customization Pipeline

The customization pipeline is expected to work as long as the input images fulfill the expected requirements, i.e., the perspective of the image being parallel to the screen. We will choose 10 random poses for this test. Each test consists of an input image of one of the team members doing a chosen pose, and the system is expected to extract the pose correctly from the image with an accuracy close to 100%.

After each subsystem passes their modular tests, we will integrate the full system. Then we will run the full integrated system against the following overall tests to make sure it meets all the user case requirements.

B. Overall Tests on Accuracy of Posture Detection

These tests examine the system's ability in detecting users' inaccurate postures. The tests consist of images of team members doing correct and incorrect postures, and customized postures will also be tested. For correct postures, the system should report both with UI and verbal instructions, clearly enough to convey to the user the correctness of their pose. For incorrect postures, the system should detect the incorrect body parts and give verbal feedback on the body part with the most prominent mistakes. The system has to pass 90% of the tests consistently to reach the user case requirement on accuracy.

C. Overall Tests on Verbal Instruction Latency

Verbal Instruction Latency tests aim to verify that the verbal instruction system still has reasonable latency when running together with other segments of the system. These tests will be conducted together with the Accuracy tests. As the system provides verbal instructions during the Accuracy tests, we track the corresponding latency for each of the poses. The latency should be under 5 seconds consistently throughout all Accuracy tests and should be easy to follow in terms of word speed and clarity.

D. Overall Tests on Accessibility

On accessibility, we want to ensure our system is easy to use for the user group we are targeting. The accessibility tests will involve showing the system to people outside of the team and getting feedback on how accessible the system is to use. The criteria for evaluating the system include: the intuitiveness of the UI, the visual and audio latency, the posture detection accuracy, the difficulty of using the customization pipeline, and how easy it is to use under different environments.

VIII. Project Management

A. Schedule

All team members have worked together to complete research and design of the project and will work on collecting images of professional reference poses for the system. Jerry will work on interfacing with the OpenPose API using Python and implementing the file storage system described in the design. Sirui will also work on interfacing with the OpenPose API and building the user interface for the app. Hongzhe will set up and install OpenPose and work on integrating it with the rest of the application. Shiheng will work on developing the scoring algorithm for judging how accurately a user was able to mimic a posture and also develop the verbal feedback system. All team members will work on integrating each subsystem together and doing module and full system testing. All members will help to make the final presentation slides, and we also have accounted for unexpected roadblocks in our project by building in slack time into our schedule and accounted for Thanksgiving and fall break. Please refer to figure 7 at the end for the schedule Gantt Chart.

B. Team Member Responsibilities

Hongzhe Cheng: OpenPose Usage

- Input: User pose image passed by application
- Output: JSON file containing coordinates

Sirui Huang: Real-time instruction pipeline and application infrastructure

- Input: User real-time video captured by camera
- Output: Cut video into frames and interact with OpenPose to get real-time coordinates, pass to user selection and comparison

Shiheng Wu: User selection and pose comparison algorithm, Speech Application

- Input: Coordinates output from OpenPose
- Output: User Selection, Error Detection, generate instruction message and play them in .wav files

Jerry Feng: Custom poses pipeline and file storage infrastructure

- Input: User uploaded image/sequence of images
- Output: JSON file and image storage

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 Figure 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 factor that different laptops have differed 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. Related Work

OpenPose – underlying pose recognition software

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

X. 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.

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] Kendall, A., Grimes, M., & Cipolla, R. (2016, February 18). *PoseNet: A convolutional network for real-time 6-DOF camera relocalization*. arXiv.org. https://arxiv.org/abs/1505.07427

[9] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2019, May 30). *OpenPose: Realtime multi-person 2D pose estimation using part affinity fields*. arXiv.org. https://arxiv.org/abs/1812.08008

[10] 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

[11] Julia, Choice of Voice, Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376789

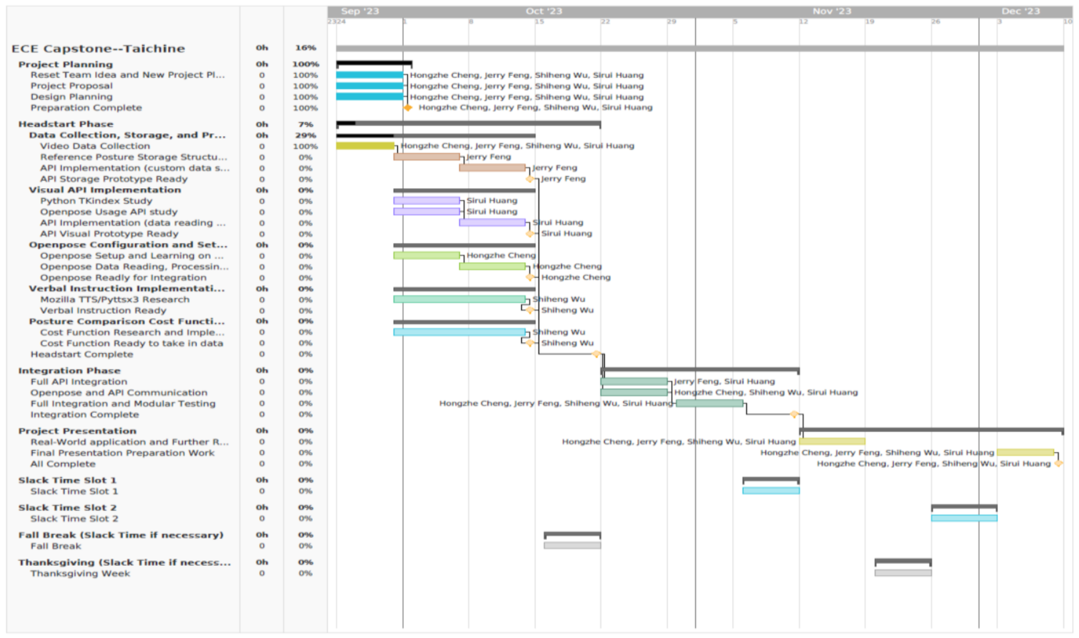


Figure 8. Gantt Chart Schedule