TAICHINE–

A New Way to Learn Tai Chi

Team B4: Hongzhe Cheng, Sirui Huang, Shiheng Wu, Jerry Feng
18-500 Capstone Design, Fall 2023
Electrical and Computer Engineering Department
Carnegie Mellon University

Product Pitch

Taichine is an interactive computer application that utilizes machine learning and joint angle comparison to visual and computer generated verbal feedback to help Taichi beginners practice and improve in Taichi.

We aimed to have a very intuitive and easy to use UI in our app. We aimed to have a 90% accuracy in joint detection, a 10 degree margin of error for joint angle detection, and a 4 second processing time for feedback generation.

Our product features a self curated list of sequences of Taichi poses and allows users to upload their own Taichi poses that they would like to train on.

Initial tests show users are satisfied with usability, but slightly disappointed with the speed of the application.

System Architecture

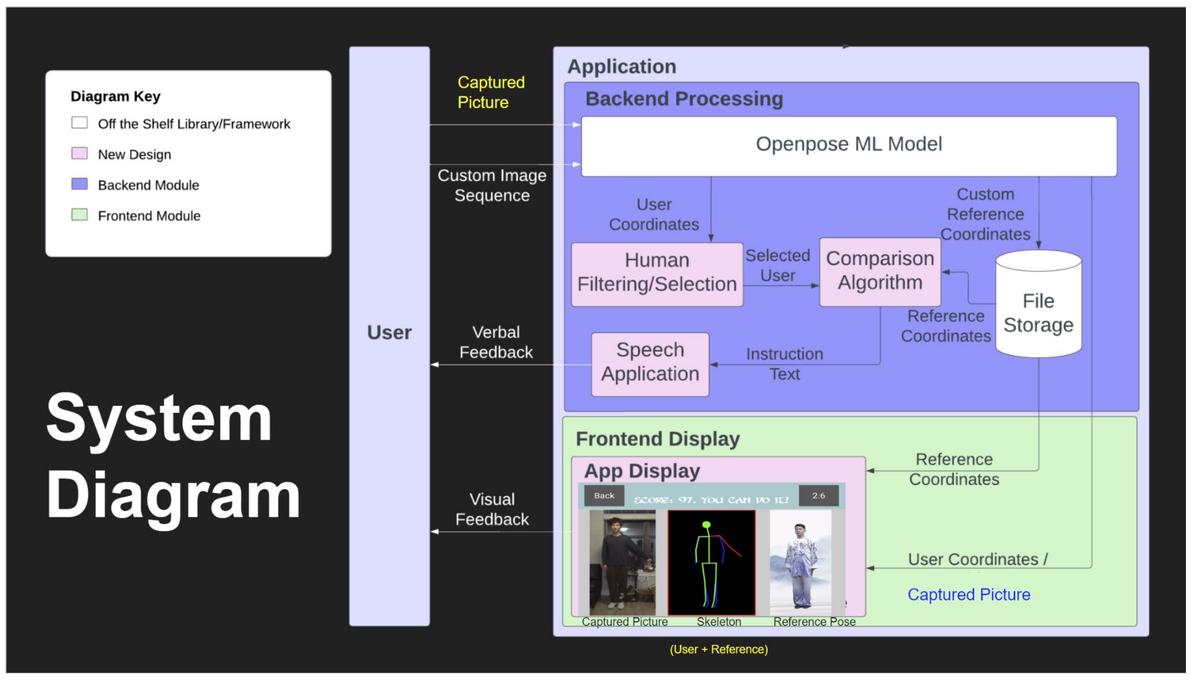
In our system, if the user chooses a native or custom Taichi pose sequence to train on and will be taken to a training screen. The user will click the start button and then do their best to imitate the reference, the user will have a countdown from an adjustable “move-on” time (default 5 seconds) before a picture of their pose is taken.

The system will inform the user to adjust to include their full body in frame, otherwise the system will use the Openpose machine learning model to determine the keypoints of the users body, such as elbows, head, knees, etc.

The backend pose comparison algorithm will then take in the keypoints of the users body and calculate the angles of joints, and compare with the reference angles of the reference pose. After which it will then find any joint angles not within an adjustable tolerance range and generate voice instructions for how to fix these errors, prioritizing those in the lower body.

The system also draws the user’s pose on top of the reference skeleton in blue on the training page, with any limb not within tolerance level being drawn in red.

If the user chooses to upload their own pose sequence consist of multiple images, they will be able to add, remove and reorder the sequence of pose pictures in the app.



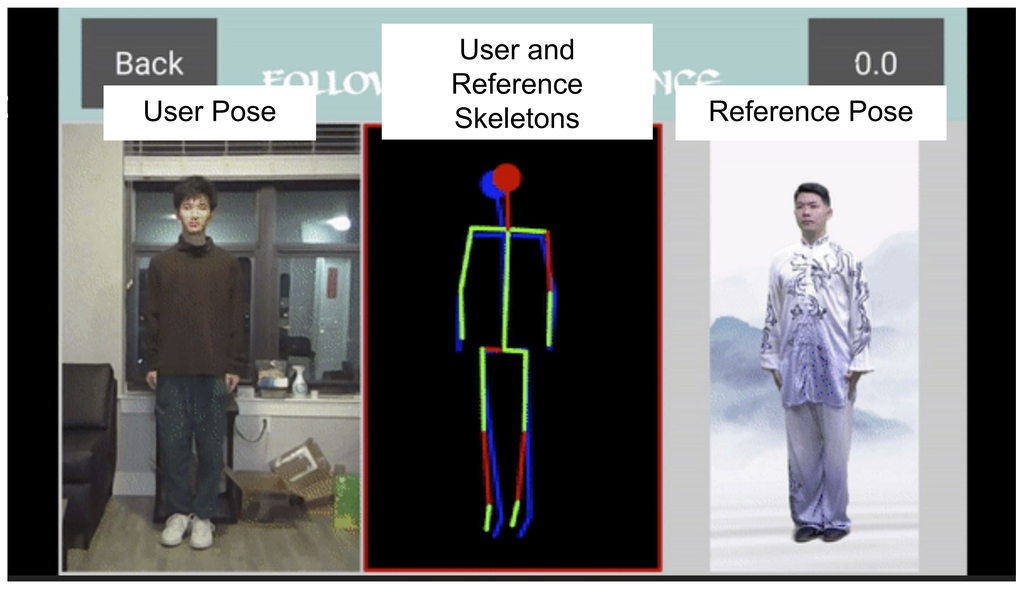
System Description

Our system consists of 2 main components: a software frontend and a backend machine learning backend with signal processing of images.

The frontend consists of settings screens to allow users to adjust tolerance range for joint angles and how long the system waits for users to get in the pose. The frontend has a training screen consisting of a live feed of the user, an image of the reference pose, and a skeleton of keypoints of the reference pose and user pose with correctness of limb positions being indicated by red and green.

The frontend also allows users to upload custom images to train on and dynamically reorder the pose sequence, delete poses, and name the sequence in the app.

User pose images will be processed for key points using Openpose and the backend compares the user’s joint angles with the joint angles of the reference pose. Any joint angles not within tolerance will be used to generate corrective verbal instructions on how to fix these errors, with errors in lower body prioritized.



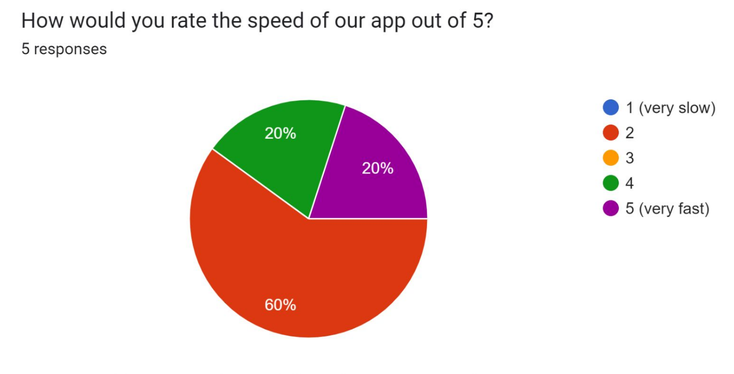
System Evaluation

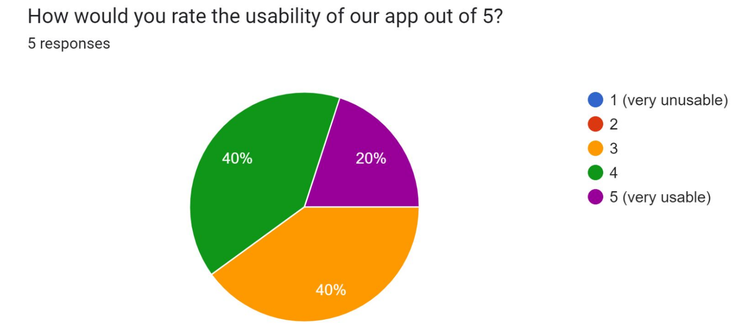
The backend was tested by training on poses and calculating the angles using online protractors on rendered pictures from Openpose and comparing with the angles calculated by the backend comparison algorithm.

Metric Namez	Desired	Actual
Total Backend Runtime	/	7s
Wait Time until Instruction	4s	3.6s
OpenPose Runtime	/	3.1s
Backend Runtime	/	0.5s
Accuracy for error body part	90%	95%
Average angle difference	10 degrees	5 degrees

Timing the application was tested by timing the system using timing modules in python when under use by team members.

User testing was done on 5 users recruited by team members who did the “Commence Form” and “Repulse the Monkey” pose sequences at selected tolerances of 15 and 20 degrees.





Conclusions & Additional Information

https://course.ece.cmu.edu/~ece500/projects/f23-teamb4/

We were able to achieve many of our goals for our system, meeting our runtime and accuracy requirements. We did have users express that the system is slow, so in the future, we would look at improving performance. We learned a lot about the difficulty of predicting how frontend elements in kivy will interact, and how difficult system integration is.

Electrical & Computer ENGINEERING
Carnegie Mellon