applied sciences

MDPI

Article

Development of AI Algorithm for Weight Training Using Inertial Measurement Units

Yu-Chi Wu [1,*], Shi-Xin Lin [1], Jing-Yuan Lin [1], Chin-Chuan Han [2], Chao-Shu Chang [3] and Jun-Xian Jiang [1]

[1] Department of Electrical Engineering, National United University, Maio-Li 360, Taiwan; shixin627@gmail.com (S.-X.L.); yuan@nuu.edu.tw (J.-Y.L.); guncen@gmail.com (J.-X.J.)
[2] Department of Computer Science and Information Engineering, National United University, Maio-Li 360, Taiwan; cchan@nuu.edu.tw
[3] Department of Information Management, National United University, Maio-Li 360, Taiwan; cschang@nuu.edu.tw
* Correspondence: ycwul@nuu.edu.tw; Tel.: +886-939-967722

Featured Application: Fitness, Weight Training, Medical Rehabilitation, Sport Training, Health Management.

Abstract: Thanks to the rapid development of Wearable Fitness Trackers (WFTs) and Smartphone Pedometer Apps (SPAs), people are keeping an eye on their health through fitness and heart rate tracking; therefore, home weight training exercises have received a lot of attention lately. A multi-procedure intelligent algorithm for weight training using two inertial measurement units (IMUs) is proposed in this paper. The first procedure is for motion tracking that estimates the arm orientation and calculates the positions of the wrist and elbow. The second procedure is for posture recognition based on deep learning, which identifies the type of exercise posture. The final procedure is for exercise prescription variables, which first infers the user's exercise state based on the results of the previous two procedures, triggers the corresponding event, and calculates the key indicators of the weight training exercise (exercise prescription variables), including exercise items, repetitions, sets, training capacity, workout capacity, training period, explosive power, etc.). This study integrates the hardware and software as a complete system. The developed smartphone App is able to receive heart rate data, to analyze the user's exercise state, and to calculate the exercise prescription variables automatically in real-time. The dashboard in the user interface of the smartphone App can display exercise information through Unity's Animation System (avatar) and graphics, and records are stored by the SQLite database. The designed system was proven by two types of experimental verification tests. The first type is to control a stepper motor to rotate the designed IMU and to compare the rotation angle obtained from the IMU with the rotation angle of the controlled stepper motor. The average mean absolute error of estimation for 31 repeated experiments is 1.485 degrees. The second type is to use Mediapipe Pose to calculate the position of the wrist and the angles of upper arm and forearm between the *Z*-axis, and these calculated data are compared with the designed system. The root-mean-square (RMS) error of positions of the wrist is 2.43 cm, and the RMS errors of two angles are 5.654 and 4.385 degrees for upper arm and forearm, respectively. For posture recognition, 12 participants were divided into training group and test group. Eighty percent and 20% of 24,963 samples of 10 participants were used for the training and validation of the LSTM model, respectively. Three-thousand-three-hundred-and-fifty-nine samples of two participants were used to evaluate the performance of the trained LSTM model. The accuracy reached 99%, and F1 score was 0.99. When compared with the other LSTM-based variants, the accuracy of one-layer LSTM presented in this paper is still promising. The exercise prescription variables provided by the presented system are helpful for weight trainers/trainees to closely keep an eye on their fitness progress and for improving their health.

Citation: Wu, Y.-C.; Lin, S.-X.; Lin, J.-Y.; Han, C.-C.; Chang, C.-S.; Jiang, J.-X. Development of AI Algorithm for Weight Training Using Inertial Measurement Units. *Appl. Sci.* **2022**, *12*, 1422. https://doi.org/10.3390/app12031422

Academic Editors: Teen-Hang Meen and Chun-Yen Chang

Received: 27 November 2021
Accepted: 25 January 2022
Published: 28 January 2022

Keywords: inertial measurement unit; 9-DoF sensor; weight training; motion tracking; posture recognition; machine learning; internet of things

1. Introduction

In recent years, home weight training exercises have received a lot of attention. Wearable products have also become an important tool to help people improve the quality of exercise. In terms of exercise data quantification, with the rapid development of Wearable Fitness Trackers (WFTs) and Smartphone Pedometer Apps (SPAs), people are keeping an eye on their health through heart rate, fitness, and sleep tracking [1].

In 2004, Kong [2] developed an inertial navigation system algorithm based on a low-cost inertial measurement unit to implement a nonlinear data fusion algorithm using a Distribution Approximation Filter (DAF). Gyroscopes are often used for system positioning in many applications, but gyroscopes generate error drift, a low-frequency drift phenomenon over time [3], which accumulates very large orientation errors over time, so that an attitude positioning system cannot use only one gyroscope to achieve accurate orientation. The causes of integration drift can be divided into two categories: one is the linear increase of drift due to the integration of the original offset in the signal, and the other is the integration of noise in the signal [4].

In the long term, both accelerometer and magnetometer have no drift. Inputting the accelerometer and magnetometer data into Kalman Filter (KF) can compensate the drift of gyroscope, so it can be used to determine the orientation, but both of them have more obvious short-term noise [5].

Many studies have used data from inertial measurement units (IIMUs) for human posture recognition using machine learning models. Tahir [6] used a one-dimensional Hadamard wavelet transformation and a feature extraction method based on one-dimensional LBP (Local binary patterns) to compute valuable eigenvalues in acceleration and angular velocity signals, and used a sequential minimization algorithm and a stochastic forest method to classify activities in the USC-HAD (USC human activity dataset) and IMSB (IM-Sporting Behaviors) datasets. No IMU-based device was presented in this work. Ezio Preatoni [7] trained and tested different supervised machine learning models based on 14 participants, including k-NN (k-nearest neighbors) algorithm and SVM (Support Vector Machine), using accelerations and angular velocities of one sensor with an highest accuracy of 96.4% and of two sensors with 97.6% accuracy to classify four fitness sports. No exercise prescription variables, such as exercise sets, training capacity, workout capacity, training period, explosive power, etc. were considered. Many research papers have combined data from cell phones and wearable sensors to automatically capture features using deep learning algorithms, effectively saving manual feature processing [8]. Sui [9] mentioned that many papers use IMU data into a CNN (Convolutional Neural Network) to achieve posture recognition, and he proposed another CNN algorithm to calculate the pace trajectory. In his work, the developed IMU device was worn on a shoe for foot motion tracking. The literature [10] proposes a real-time segmentation and classification algorithm that can identify physical exercise and works well in both indoor and outdoor environments. This proposed algorithm achieved a 95% classification accuracy for five indoor and outdoor exercises. No exercise prescription variables were considered in [10]. Koskimäki and Siirtola [11] trained a machine learning model with accelerometer data to classify 36 types of fitness exercises. The data were collected using a GeneActiv 3D accelerometer, which can be used as a wrist-worn or be attached with straps to other body locations.

In many cases, IMUs have been used to develop multimodal algorithms, and Hausberger et al. [12] designed a smart fitness device that uses acceleration and angular velocity signals read by IMUs to perform automatic motion tracking and analyze weight training exercises. Their device was mounted on the dumb bell. The repetitions of dumb bell exercise were detected by a peak/valley detector. Other exercise prescription variables

were not considered in this work. Seong and Choi [13] used an IMU to implement a new type of computer interface operating system that performs hand motion tracking and click gesture recognition algorithms using Euler angles and acceleration data, respectively.

Chen et al. [14] proposed a deep learning framework based on graph convolutional network (GCN) to predict the position of each joint of human lower limbs during motion in 3D space using signals collected from five IMUs (above the ankles of both feet, above the knees of both feet, and at the waist). Abdelhady et al. [15] proposed a system of IMUs in the lower legs, IMUs in the thighs, and pressure sensors in both feet to measure the kinetics and kinematics of the lower limbs. Wang et al. [16] used an IMU on each leg and a motion camera on each calf to estimate the asymmetry of stride length and gait, and the root mean square error of stride length was 5.4 cm and asymmetry was 3.0% in healthy subjects, while the root mean square error of stride length was 9.6 cm and asymmetry was 14.7% in subjects with abnormal gait.

Zou et al. [17] proposed a low-cost smart glove equipped with an inertial unit. Their system could recognize 15 sets of training programs, detect three common nonstandard behaviors: case 1-the overall speed of a repetition is too fast or too slow, case 2-the speeds of outward and backward processes are not balanced, and case 3-the repetitions are not stable with noticeable shakes. A segmentation process was used in their system to detect the start point and end point of each repetition with the help of a double adaptive thresholds-based method. Then the half dynamic time warping (Half-DTW), which is not a deep learning approach, was proposed to measure the similarity between the unrecognized activity segmentation data (accelerometer and gyroscope readings) and the activity templates. The Half-DTW requires two steps. First, it applies an outward-backward segmentation operation to divide a repetition into outward and backward parts through locating the turning point (TP, stage changing point). Second, the Half-DTW applies the corresponding matching operation and outputs the identified activity type. Several exercise quality indices were calculated as well: standard degree and execution duration for the assessment of single repetition, and smoothness and continuity for the assessment of group repetitions. The average accuracy using Half-DTW for 15 exercise types is 96.07% with 103.76 ms average time delays. Rodriguez et al. [18] presented a wearable device using three inertial units with triaxial accelerometers, gyroscopes, and magnetometers to measure the orientation of three sections of the spine for low back pain therapy. By integrating the absolute and relative orientation from the sensors based on the so-called direction cosine matrix (DCM), it estimates the posture of the back in real-time and uses a fuzzy system to control a vibration unit to alert the user to correct the posture of his/her back.

From these mentioned literature in the above, most papers focus on recognition algorithms for exercises to improve prediction accuracy. The number of sensors used for collecting data and identifying exercise types is ranging from one to five or even more. Depending on the application purpose, some sensors are attached to lower limbs, and some are attached to arms. However, to the best of the authors' knowledge, very few papers extend the results of the IMU motion tracking to produce helpful exercise prescription variables for fitness lovers. This paper integrates hardware and software to develop an attitude and heading reference system (AHRS) device using only two IMUs (one on the wrist band and the other in the sleeve near the elbow) and an intelligent algorithm that performs motion tracking, posture recognition, and calculates exercise prescription variables for weight training. These exercise prescription variables, including exercise items, repetitions, sets, training capacity, workout capacity, and training period, etc., provide meaningful information for weight trainers/trainees. A fitness App with the function of recording exercise performance is also developed on the smartphone, so that the users can check their exercise performance and review the historical exercise records in the user interface in real time.

2. Materials and Methods

2.1. Overall Study Design

As shown in Figure 1, a wearable device was developed in which a microcontroller unit (MCU) was used to collect six-axis data and quaternion data from two IMUs (one on the wrist band and the other on the upper arm near the elbow) and heart rate data. In addition to the IMU and heart rate module, the MCU in the wrist band is equipped with a real time clock (RTC) and a Bluetooth Low Energy (BLE). The RTC provides the present time for MCU, and BLE is used to transmit IMU six-axis (6 degrees of freedom, 6-DoF) data, quaternion, heart rate, and time through data packaging to the smart phone. By integrating the existing techniques of motion tracking [19] and posture recognition [12], a weight training quantization algorithm is devised for smartphone to convert motion data into prescription variables. The smartphone App can communicate with the MCU through BLE using pre-defined commands. The App then parses the data received through Bluetooth (BT data packet) into IMU 6-DoF data, quaternion, and heart rate. The IMU 6-DoF data are used for posture recognition based on a pre-trained machine learning model (ML model) installed on the phone. This ML model is trained offline on a PC by feeding recorded IMU 6-DoF data, which are prepared in csv files. Once the ML model is trained, it can be installed on the smart phone for posture recognition to create posture labels. The parsed quaternion is used for motion tracking to calculate arm orientation vectors and to find out the elbow and wrist positions that are then fed to the Unity's Animation System (avatar, virtual humanoid) [20]. The parsed heart rate and the motion of the 3D virtual humanoid are displayed on the screen of the developed smartphone App. The posture label with the positions of elbow and wrist is input to the exercise state analysis, and the exercise event stream (including exercise state, rest state, repetitions, rest time between sets, and exercise event) is obtained for the exercise event process. The exercise prescription variables are then computed, based on the input dumb bell weight and the exercise event stream, by the exercise event process, shown on the phone screen of the App, and stored in SQLite database. Not only transmitted by BLE, the IMU 6-DoF, quaternion, and heart rate data collected by MCU can also be transmitted by wire and stored as csv files. These csv files can be stored in a cloud or in a PC.

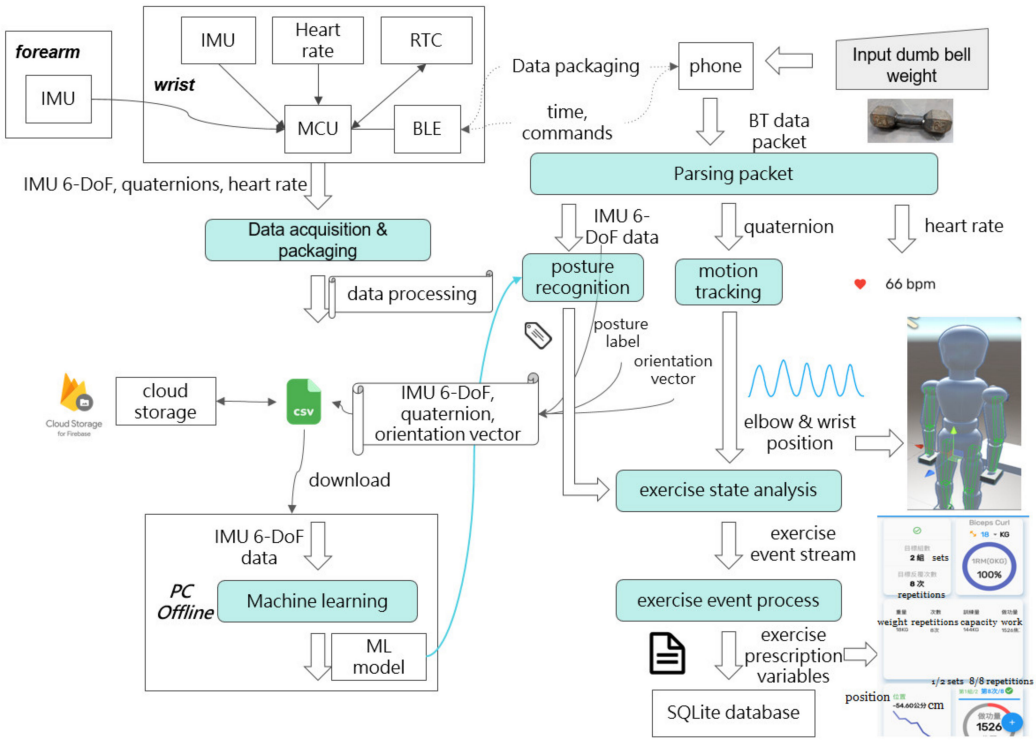


Figure 1. System architecture.

The proposed artificial intelligence weight training quantization algorithm converts the raw motion data into exercise prescription variables step by step based on mathematical and physical theories, as shown in Figure 2. This part has two procedures, one for posture recognition and one for motion tracking using IMU 6-axis data and quaternion data of the wrist and the arm to calculate the posture label type and the joint position of the arm. The outputs of the two procedures are fed first to the exercise state analysis for calculating repetitions, state, and event, and then to the exercise event process for calculating the exercise prescription variables.

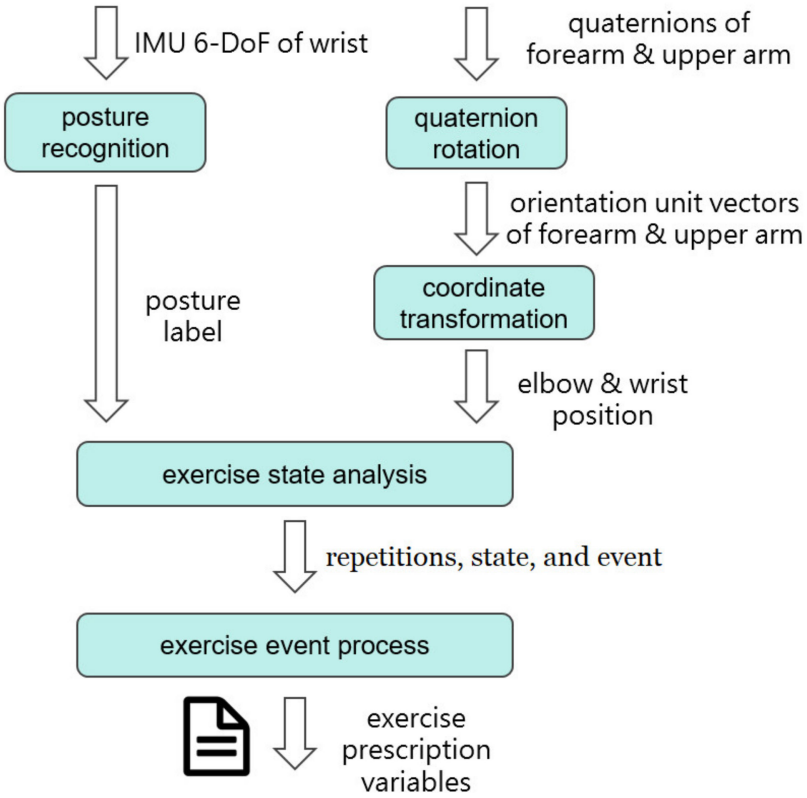


Figure 2. Flowchart of the presented algorithm.

2.2. Motion Tracking Algorithm

In this paper, we use two nine-axis sensors (Adafruit BNO055 Absolute Orientation Sensors), as shown in the top photo of Figure 3, one on the wrist and the other on the upper arm near the elbow (marked by red circles). The bottom left photo shows that the sensor coordinates transformed to world coordinates, and the bottom right photo shows the world coordinates transformed to Unity coordinates. Once the alignment with the user's orientation is done, the Unity would show the same movement as the real person does. In the built-in microcontroller of the sensor, a world coordinate with the Y-axis facing the north pole of the geomagnetic field and the Z-axis perpendicular to the earth plane is defined, and the corresponding quaternion $q_t = [\mathrm{w}, \mathrm{x}, \mathrm{y}, \mathrm{z}]$ is generated based on this world coordinate. q_t represents the action of rotating the sensor from the base state to the current position, which can be further expressed as Equation (1).

$$q_t = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\vec{\mathrm{u}}\right] \tag{1}$$

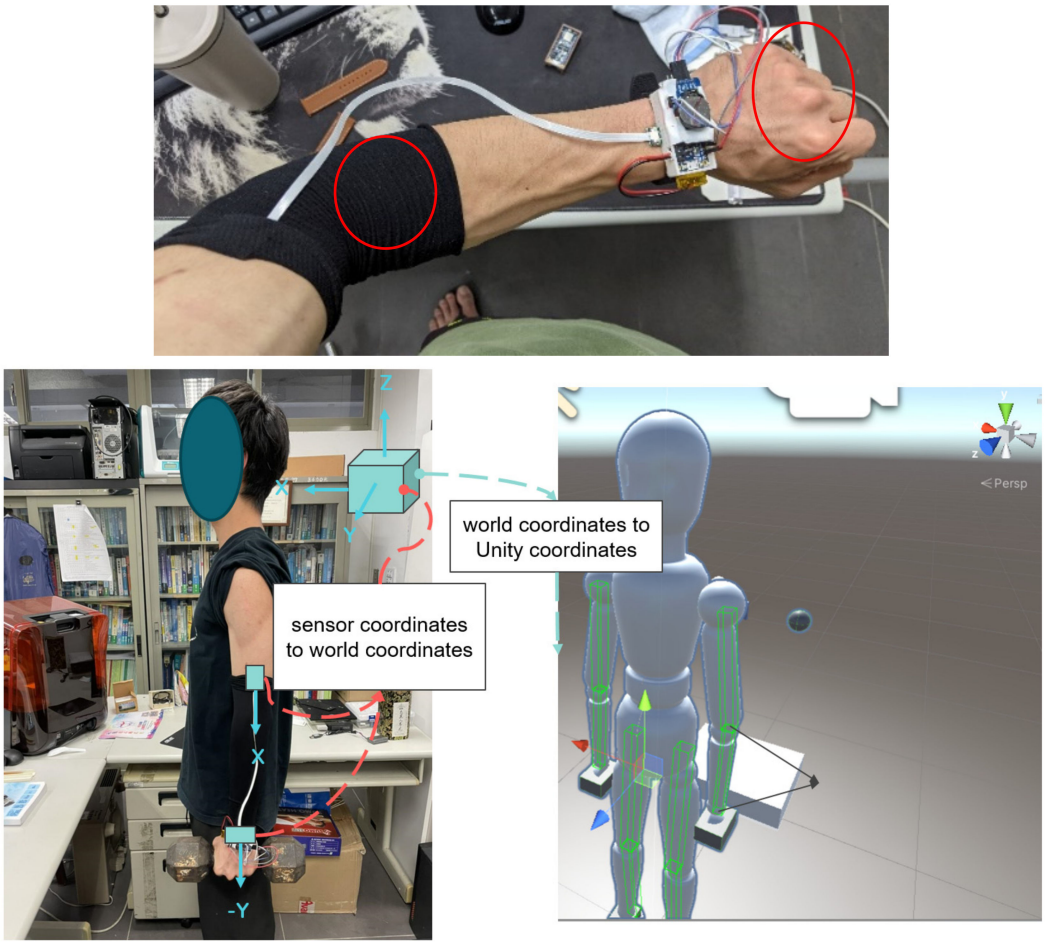


Figure 3. 3D coordinate conversion flowchart: (**top**) wearable device, (**bottom left**) sensor coordinates to world coordinates, (**bottom right**) world coordinates to Unity coordinates.

The physical meaning of $v' = qvq^* = qvq^{-1}$ is to rotate a certain three-dimensional vector $\vec{v}$ around a unit vector rotation axis $\vec{u}$ by θ degrees into a new vector $\vec{v}'$, where $\vec{v}$ is represented by the quaternion $v = \left[0, \vec{v}\right]$, and $\vec{v}'$ is represented by $v' = \left[0, \vec{v}'\right]$. When the quaternion at initial state, $q^{(0)}$, is $1 + 0i + 0j + 0k$, the object coincides with the world coordinate, which is $\vec{v}' = \vec{v}$. In Figure 3, the coordinates of the nine-axis sensor in the base state coincide with the world coordinates, and the attitude vectors of the upper arm and forearm $\vec{v}^S_{upper}$ and $\vec{v}^S_{forearm}$ align the X-axis and $-Y$-axis of the sensor coordinates, respectively, and can be expressed as quaternions as Equations (2) and (3).

$$v^S_{upper} = \left[0, \vec{v}^S_{upper}\right] = (0, 0, -1, 0) \tag{2}$$

$$v^S_{forearm} = \left[0, \vec{v}^S_{forearm}\right] = (0, 1, 0, 0) \tag{3}$$

The quaternions q_{upper} and $q_{forearm}$ of the upper arm and forearm are multiplied by the arm orientation vector as in Equations (4) and (5), respectively, to obtain the quaternions v^G_{upper} and $v^G_{forearm}$ of the arm after rotation.

$$v^G_{upper} = q_{upper} \cdot v^S_{upper} \cdot q^*_{upper} \tag{4}$$

$$v^G_{forearm} = q_{forearm} \cdot v^S_{forearm} \cdot q^*_{forearm} \tag{5}$$

where $v^G_{upper} = \left[0, \vec{\mathrm{v}}^G_{upper}\right]$, $v^G_{forearm} = \left[0, \vec{\mathrm{v}}^G_{forearm}\right]$, and the orientation vectors of the arms in the world coordinates are Equations (6) and (7).

$$\vec{\mathrm{v}}^G_{upper} = \left(x^G_{upper}, y^G_{upper}, z^G_{upper}\right) \tag{6}$$

$$\vec{\mathrm{v}}^G_{forearm} = \left(x^G_{forearm}, y^G_{forearm}, z^G_{forearm}\right) \tag{7}$$

The world coordinates are transformed to the virtual Unity coordinates. In the paper, the Y and Z axes of the world coordinates are aligned to the Unity coordinates, and Equations (8) and (9) are the arm pose vectors $\vec{\mathrm{v}}^U_{upper}$ and $\vec{\mathrm{v}}^U_{forearm}$ in the virtual three-dimensional space (virtual Unity).

$$\vec{\mathrm{v}}^U_{upper} = \left(x^G_{upper}, z^G_{upper}, y^G_{upper}\right) \tag{8}$$

$$\vec{\mathrm{v}}^U_{forearm} = \left(x^G_{forearm}, z^G_{forearm}, y^G_{forearm}\right) \tag{9}$$

There is no wearable sensor on the torso, so it is not possible to determine the orientation of the model in the Unity coordinates. Therefore, in this paper the arm movement trajectory in three dimensions is simulated under the assumption that the user's torso does not turn. To achieve this simulation, the orientation of the subject has to be corrected to the same direction as the front of the 3D avatar (virtual humanoid, Unity) before the movement. First, let the user straighten the arm towards the front of the body and calculate the horizontal angle γ between $\vec{\mathrm{v}}^U_{upper}$ and the Unity coordinate Z-axis (in front of the avatar), which is the angle between the user and the avatar's face, and simultaneously rotate $\vec{\mathrm{v}}^U_{upper}$ and $\vec{\mathrm{v}}^U_{forearm}$ in the ground plane (orthogonal to the Unity coordinates' Y-axis) by γ degrees to obtain the Unity coordinates' arm orientation vectors after aligning the real and virtual sides. Finally, based on the user's arm length l_{upper} and $l_{forearm}$, the positions of the elbow and wrist joints relative to the origin of the Unity coordinates, P^U_{elbow} and P^U_{wrist}, are calculated using Equations (10) and (11). The wrist position P^U_{wrist} is then used in the inverse kinematic model to update the 3D avatar's movement.

$$P^U_{elbow} = O(0,0,0) + l_{upperarm} \cdot \vec{\mathrm{v}}'^U_{upperarm} \tag{10}$$

$$P^U_{wrist} = P^U_{elbow} + l_{forearm} \cdot \vec{\mathrm{v}}'^U_{forearm} \tag{11}$$

2.3. Posture Recognition Algorithm

The algorithm is based on a supervised deep neural network model (including LSTM layer) to train a model for classifying four weight training posture types. Each sample contains six features (three-axis accelerations and angular velocities measured by the wrist IMU), and the sampling rate is 100 Hz. The output layer is a dense layer with softmax as the activation function. The output data size is (1, 4), so the model can classify four labels. The epoch is set to 30 and the batch size is set to 64. The python code for this model based on Keras + Tensorflow is as follows:

```
input_layer = tf.keras.layers.Input (shape = (128, 6))
hidden1 = tf.keras.layers.LSTM (128) (input_layer)
hidden2 = tf.keras.layers.Dropout (0.5) (hidden1)
hidden3 = tf.keras.layers.Dense (6, activation = 'relu') (hidden2)
output = tf.keras.layers.Dense (NUM_GESTURES, activation = 'softmax') (hidden3)
model = tf.keras.Model (inputs = fitness_dataset, outputs = output)
model.compile (optimizer = 'rmsprop', loss = 'mse', metrics = ['mae'])
```

history = model.fit (inputs_train, outputs_train, epochs = 30, batch_size = 64, verbose = 0, validation_data = (inputs_validate, outputs_validate))

After training, the pre-trained model is converted to a TensorFlow Lite model file (.tflite) with a file size of 412 KB. TensorFlow Lite is a machine-learning solution for mobile or embedded devices, with APIs available for Android and iOS mobile development platforms. Developers can convert TF (TensorFlow) models into TF Lite files and import them into their mobile projects to realize edge computing locally. In this paper, we use Flutter to develop Android mobile application, and use the TensorFlow Lite patch tflite_flutter (version: 0.8.0) of Flutter Package to import the model file into the development project. The six-axis data from the wrist IMU is used as input data to the pre-trained model of the local App for classifying the user's motion posture (three pre-defined dumbbell movements and rest state). The flowchart of the algorithm running in the App is shown in Figure 4.

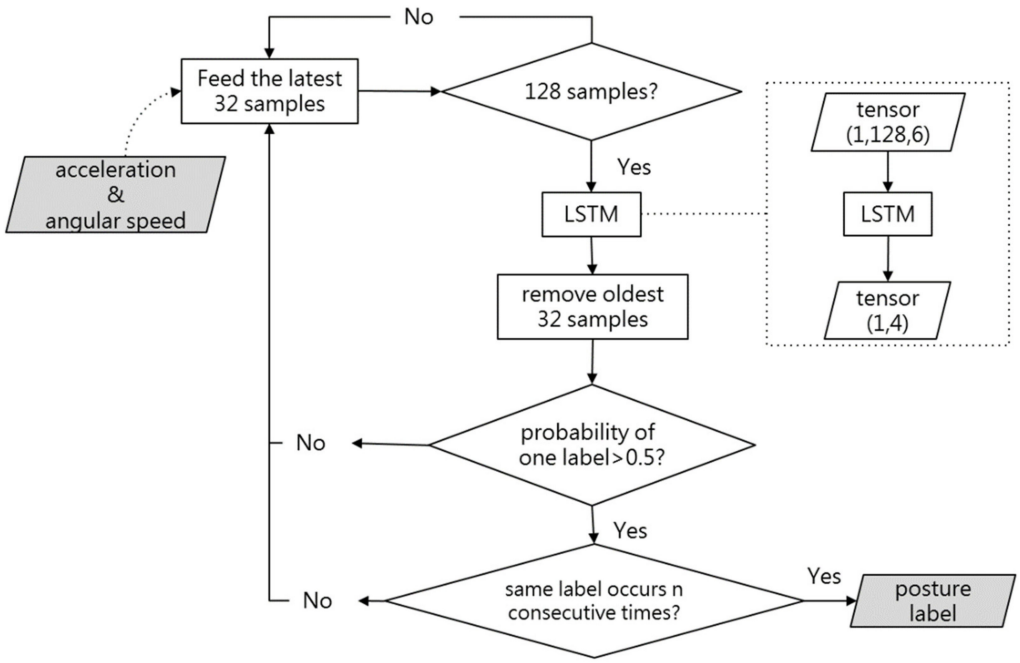


Figure 4. Flowchart of the posture recognition algorithm.

When the LSTM model completes the identification, the oldest 32 samples are removed, and the latest 32 samples are added into the input data (total 128 samples) for LSTM. The output of the LSTM model is a probability vector r with dimension (1,p), where p is the number of postures to be identified. The element with the largest probability (>0.5) in this vector is the most possible posture label at the moment. If the same label is identified for n consecutive times, this label will be used as the output result of the posture recognition algorithm.

In order to collect the motion data for training the neural network, the participants were divided into a training group (control group) and a test group. The datasets for training and testing were collected by an experiment instructor who instructed the participants to put on the wearable device developed in this paper, and the participants performed four postures (rest, bicep curl, lateral raise, and shoulder press) at their own pace. The instructor used the developed smartphone App to record the participants' exercise data (with a sampling rate of 100 Hz). The data were stored in the local database, converted to a csv file, and uploaded to the storage space of the cloud database at the end of sampling. Then the local computer downloaded the cloud data for training.

The dimension of the input data is (1,128,6), and the IMU six-axis data are packaged in 32 time steps, as shown in Figure 5. Each set of data has 128 consecutive samples. Once a set of data has been processed, the first (oldest) 32 samples in this set are discarded and

32 new samples are added to the end of the set to create another data set until the sample in the data is exhausted.

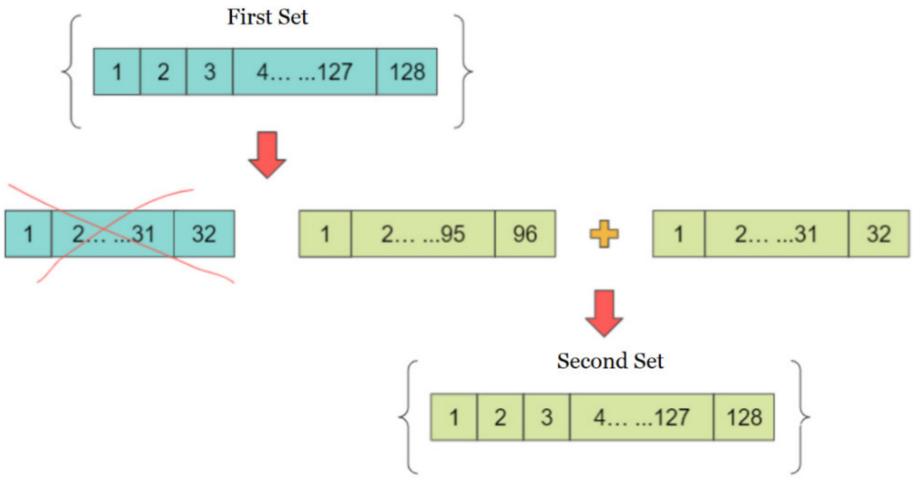


Figure 5. Data packaging flowchart.

2.4. Exercise Prescription Variables Algorithm

The algorithm integrates the wrist position and posture label to calculate the exercise state, repetitions, exercise period, and rest intervals between sets and to trigger the corresponding exercise events to calculate the exercise prescription variables. In addition to the weight training, it also can be used for the assessment of neuromuscular qualities and for a targeted resistance training. Picerno [21] provided the exercise operator with a list of good practice rules concerning the instrumented assessment of muscle strength during isoinertial resistance exercises, such as movement execution, load assignment, number of lift within a set, rest intervals between sets, etc. Weakley et al. [22] listed feedback variables and their effects on acute-training performance, such as frequency, quantitative vs. qualitative, conscientiousness, motivation and competitiveness, intrinsically vs. extrinsically motivated athletes, and encouragement. Good exercise prescription variables of weight training can provide correct feedback to the exercise operator and motivate him/her to keep on good exercise habit. As shown in Figure 6, the current state flag is determined by the state of the previous round and the current output of the posture label, where a flag of True means the user is in exercise state, otherwise it is in rest state. When in exercise state, movement event is triggered, and peak/valley detection algorithm is used to count the repetitions of this exercise. If no repetition is detected, repetition event is triggered; otherwise go to state flag identification. When in rest state, if a new set of exercise begins, data for the previous set are updated, rest time between sets is computed, and the start event is triggered.

In the exercise state, the movement event is triggered, and the wrist position P relative to the shoulder is input to the peak/valley detection algorithm to calculate the repetitions and the displacement in the vertical direction, as shown in Figure 7. The shoulder here is set as the origin, assuming that the length of the upper arm and forearm are 1. The current value of the wrist position P in the Z-axis coordinate perpendicular to the ground plane is between -2 and $+2$. In state 0, we store P into the array and determine whether P is greater than the lower threshold B1. If yes, find the minimum value P_{min} in the array, and go into state 1. In state 1, if P is greater than the upper threshold B2, enter into state 2. In state 2, record the value P in the array until P is less than the threshold B2. When P is less than B2, find the maximum value P_{max} in the array and enter into state 3. In state 3, if P is less than B1, it means that a repetitive motion is completed. The displacement of the current repetition in Z-axis $h = 2 \times (P_{max} - P_{min})$ and the time difference Dt from the last completed repetition to the current one (i.e., the time spent on the current repetition) are calculated, and therefore the work is computed (h multiplied by the acceleration of gravity g). After that, trigger the repetition event and return to state 0 from state 3. Different exercise modes

should use different B1 and B2 values, and different users have different thresholds for the same exercise item. Therefore, the values B1 and B2 should be adjusted according to the user's exercise mode and habit. Table 1 shows the thresholds for three exercise modes for one subject.

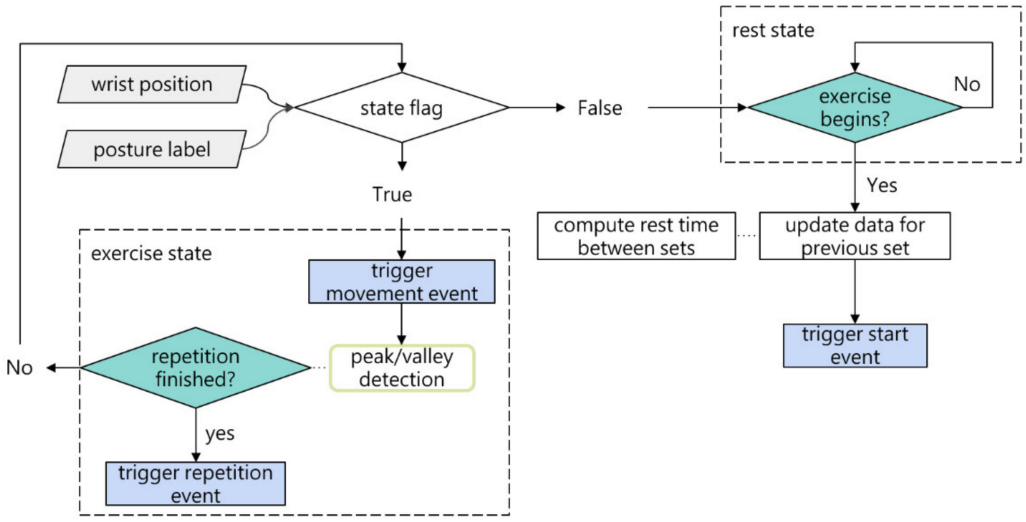


Figure 6. Flowchart of state analysis.

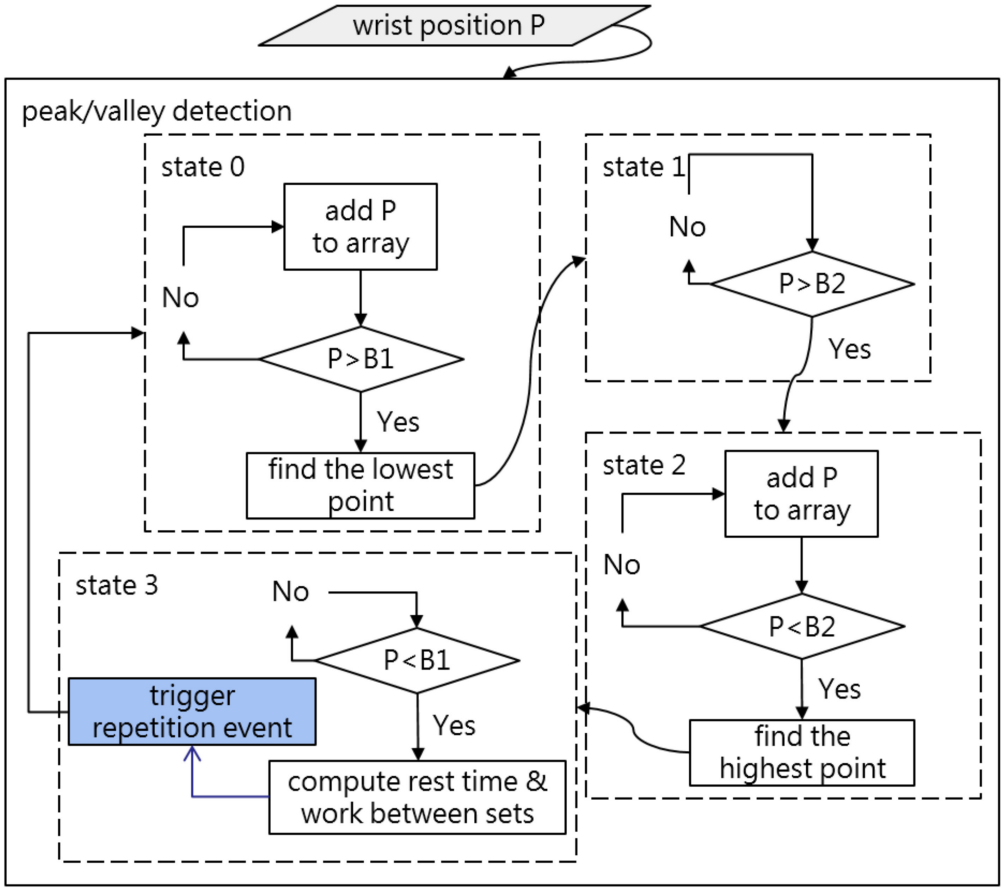


Figure 7. Peak/valley detection module.

Table 1. Thresholds for three exercise modes.

Modes \ Thresholds	B1	B2
Bicep curl	−1.6	−0.15
lateral raise	−1.65	−0.55
shoulder press	1	1.65

As shown in Figure 8, the exercise event enters into the application programming interface (API) of the event handler in a prescribed format. Depending on the type of event and the parameters it carries, the program executes the corresponding computation of the exercise prescription variables and refreshes the user interface.

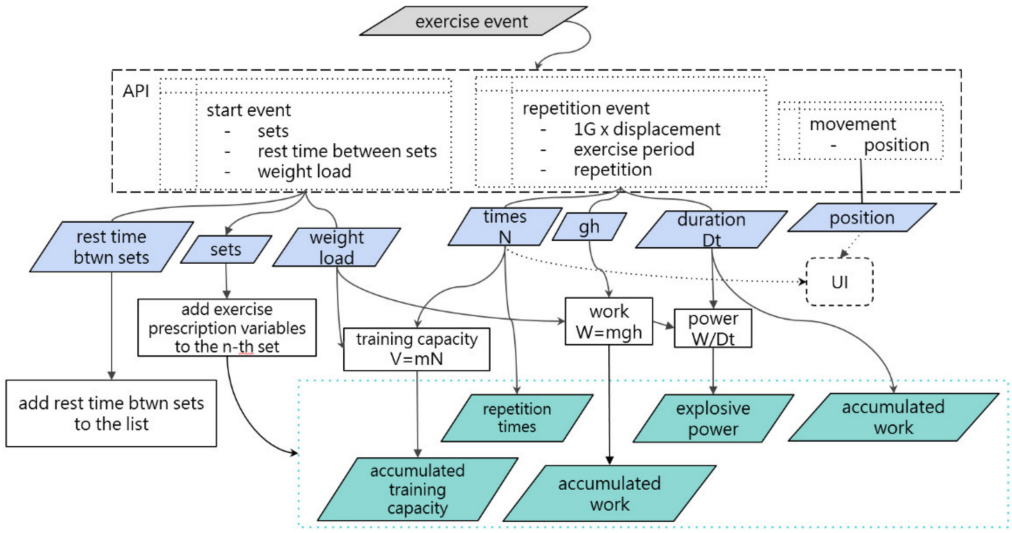


Figure 8. Exercise event handler process.

- When the start event occurs, update the number of sets (unit: times), the rest time between sets (unit: seconds), and the load weight of the set (mass of dumbbell input by the user, unit: kg).
- When a movement event occurs, the current wrist position (in centimeters) is updated and the user interface is refreshed.
- When a repetition event occurs, the training capacity, work, and power are calculated; the number of repetitions, cumulative repetition period (in seconds), cumulative training capacity, cumulative work, and maximum power (explosive power) are updated; and the user interface is refreshed.

Table 2 lists the source or formula of the exercise prescription variables, where the load weight is manually entered into the App by the user; the number of repetitions is calculated by the peak/valley detection algorithm and brought in by the repetition event; the training capacity is the load weight multiplied by the number of repetitions in the current set as Equation (12); the work is calculated by multiplying the load weight by the acceleration of gravity and the repetition displacement as Equation (13); the average power is the work in the current repetition divided by the time spent in the repetition D_{rep} as Equation (14); the burst power is the maximum average power in the current set; and the rest time between sets is the time interval from the previous set to the current set.

Table 2. List of exercise prescription variables.

Prescription Variables	Source/Equation
weight load m_{load}(kg)	data input from App
repetitions N_{reps}(times)	data from repetition event
training capacity V_{tra}(kg)	$V_{tra} = m_{load} \times N_{reps}$ (12)
work W_{rep}(J)	$W_{rep} = m_{load}gh$ (13)
average power P_{rep}(W)	$P_{rep} = W_{rep}/D_{rep}$ (14)
explosive P_{max}(W)	the maximum P_{rep} of a set
rest time between sets (s)	duration from previous set to current set

2.5. Verification Experiments

The designed system was proven by two types of verification tests. The first type is to control a stepper motor to rotate the designed IMU and to compare the rotation angle obtained from the IMU with the rotation angle of the controlled stepper motor. As shown in Figure 9, the nine-axis sensor was fixed above the center axle of a stepper motor before the experiment was conducted. The experiment used a microcontroller to output the control signal to the drive circuit of the motor, which drove the nine-axis sensor to rotate forwards from 0 to 270 degrees with an increment of 18 degrees and then backward from 270 back to 0 degrees with a decrement of 18 degrees ($0^\circ \leq \theta_{motor} \leq 270^\circ$). For every 18 degrees of motor rotation ($\Delta\theta_{motor} = 18^\circ$), the microcontroller records the quaternion q of the nine-axis sensor. Equations (15) and (16) are used to derive the angle value θ of the current nine-axis sensor rotation along the central axel u of the motor (the z-axis of the nine-axis sensor is parallel to u). The angle of rotation of the stepper motor θ_{motor} was used as the control group, and the angle value measured by the nine-axis sensor θ_{imu} was used as the experimental group. The mean absolute error was calculated between the two sets of samples.

$$q = \left[a, \vec{b}\right] = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)u\right] \quad (15)$$

$$\frac{\theta}{2} = \cos^{-1}(a) \Rightarrow \theta = 2\cos^{-1}(a)^\circ \quad (16)$$

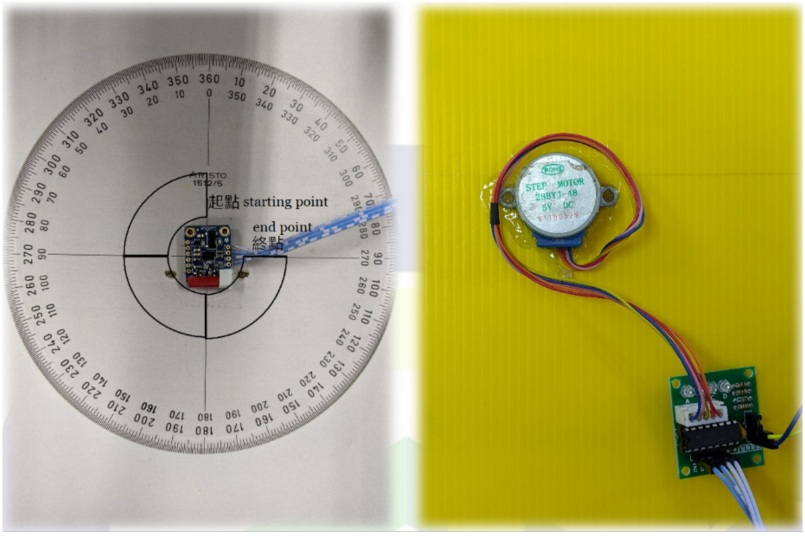
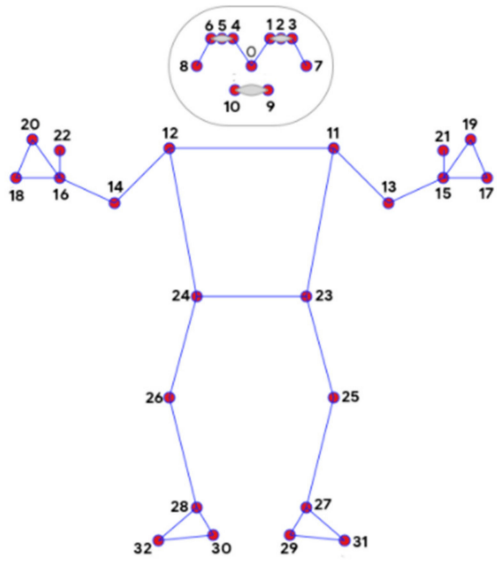


Figure 9. Experimental environment set-up—front (**left**, IMU is shown) and back (**right**, stepping motor & drive circuit is shown) of the acrylic panel.

The second type of verification test is to use a camera with an open source optical human body pose estimation system, Mediapipe Pose [23], to calculate the position of the wrist and the angles of upper arm and forearm between the Z-axis, and these calculated data are compared with the designed system. This is to verify the reliability of the motion tracking algorithm. MediaPipe Pose is a machine learning-based human pose tracking algorithm that can identify and mark 33 human body feature points from RGB images of the whole body, as shown in Figure 10. An experimental environment where a camera is placed facing a wall with a white board was setup. We connected a camera to the computer; ran the Python script in Anaconda environment, captured the RGB image of the human body through the camera; input the image to the Python API provided by MediaPipe Pose to find the left shoulder, left elbow, and left wrist coordinates of the human body ($p_{shoulder}$, p_{elbow}, and p_{wrist} in Figure 10 [4] at locations 11, 13, and 15); and mark these coordinates on the original image.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure 10. Feature points output map of human from MediaPipe Pose.

In the M system (MediaPipe Pose estimation system), a rectangular coordinate system is defined and the positions of each feature in the image are denoted as $f_{features}$, and the left shoulder, elbow, and wrist are denoted as $f_{shoulder}$, f_{elbow}, and f_{wrist}, respectively. The two-dimensional unit vectors $\vec{v}^{M}_{upperarm}$ and $\vec{v}^{M}_{forearm}$ of the forearm and upper arm were calculated using Equation (17), and the timestamp and arm orientation vector were saved as a csv file in a list at the end of each sampling.

$$\begin{cases} \vec{v}^{M}_{upper} = \frac{f_{elbow} - f_{shoulder}}{||f_{elbow} - f_{shoulder}||} \\ \vec{v}^{M}_{forearm} = \frac{f_{wrist} - f_{elbow}}{||f_{wrist} - f_{elbow}||} \end{cases} \quad (17)$$

In this paper, a new coordinate system will be constructed as the common coordinate system between the M system and the presented system, and these two coordinate systems will be aligned with the new coordinate system, and the M system will be used as a reference to observe the difference of the motion tracking algorithm of the presented system.

The left shoulder is set as the origin of the new coordinate system $O^{new}(0, 0, 0)$, and the arm orientation vectors of the two original coordinate systems are transferred from the old coordinate system to the new coordinate system (the upper arm is denoted as $\vec{v}_{upper}$ and the lower arm is denoted as $\vec{v}_{forearm}$), and the relative positions of the elbow and wrist joints in the new coordinate system, P_{elbow} and P_{wrist}, are derived by Equation (18).

$$\begin{cases} P_{elbow} = O^{new}(0,0,0) + l_{upper} \cdot \vec{v}_{upper} \\ P_{wrist} = P_{elbow} + l_{forearm} \cdot \vec{v}_{forearm} \end{cases} \quad (18)$$

The experiment is to use the two-dimensional plane (Y-Z plane) as the reference plane. The camera lens surface is parallel to the Y-Z plane. The subject's back is against the wall (Y-Z plane). During the experiment, the subject moves the arm in the frontal plane (Y-Z plane), and samples are taken simultaneously by both systems (sampling rate: 30 Hz for M system and 100 Hz for this system). Time (in ms) is recorded, and $\vec{v}_{upper}$, $\vec{v}_{forearm}$, P_{elbow}, P_{wrist}, α, and β, are calculated, where the angle between the left upper arm vector $\vec{v}_{upper}$ and the *Z*-axis is α, and the angle between the forearm vector $\vec{v}_{forearm}$ and the *Z*-axis is β. The direction from the user's back to the user's front is the –*X*-axis direction in the common coordinate system with the vector $(-1,0,0)$.

3. Results

3.1. Verification of Quaternion and Angle Reliability

Table 3 shows the results of one experiment for the first type of verification test mentioned in Section 2.4. The mean absolute error of 30 samples is 1.195 degrees for this experiment. The angles of the motor forward and backward passing 180 (samples #10 and #20) are different, which may be caused by the system device itself (motor gear gap). A total of 31 experiments were conducted. Table 4 shows the mean absolute error of each experiment, and the average mean absolute error of 1.485 degrees was obtained. Therefore, this proves the quaternion values obtained from the nine-axis sensor are within an acceptable engineering criterion.

Table 3. *Z*-axis rotation angle experimental results.

	Degree				Degree		
Sample #	θ_{motor}	θ_{imu}	**Absolute Error**	**Sample #**	θ_{motor}	θ_{imu}	**Absolute Error**
1	18	17.22	0.78	16	252	253.37	1.37
2	36	35.61	0.39	17	234	235.8	1.8
3	54	52.39	1.61	18	216	217.98	1.98
4	72	69.59	2.41	19	198	200.68	2.68
5	90	88.75	1.25	20	180	182.45	2.45
6	108	109.92	1.92	21	162	164.71	2.71
7	126	125.1	0.9	22	144	146.24	2.24
8	144	142.81	1.19	23	126	128.17	2.17
9	162	161.36	0.64	24	108	110.11	2.11
10	180	179.75	0.25	25	90	91.85	1.85
11	198	198.23	0.23	26	72	73.29	1.29
12	216	216.09	0.09	27	54	54.64	0.64
13	234	234.12	0.12	28	36	36.01	0.01
14	252	252.13	0.13	29	18	17.46	0.54
15	270	269.9	0.1	30	0	0.00	0.00
Mean absolute error					1.195		

Table 4. The average mean absolute error of the quaternions obtained from the nine-axis sensor.

	Degree		Degree
Experiment #	**Mean Absolute Error**	**Experiment #**	**Mean Absolute Error**
1	1.195	17	1.629
2	1.325	18	1.402
3	1.153	19	1.560
4	1.352	20	1.522
5	1.339	21	1.495
6	1.487	22	1.769
7	1.366	23	1.677
8	1.314	24	1.666
9	1.245	25	1.939
10	1.399	26	1.545
11	1.398	27	1.569
12	1.524	28	1.558
13	1.708	29	1.569
14	1.512	30	1.588
15	1.513	31	1.512
16	1.193		
average mean absolute error		1.485	

3.2. Motion Tracking Verification

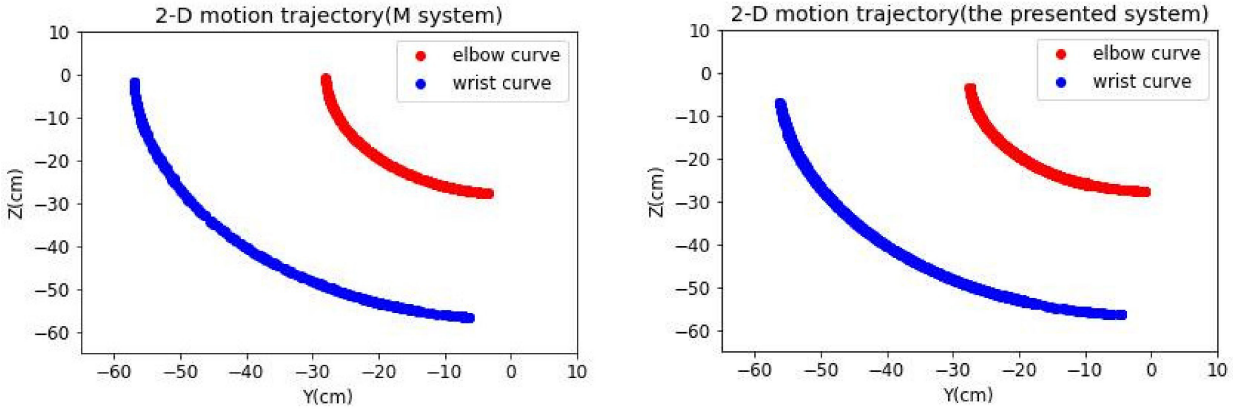
Figure 11 shows the curves of the parameters (P_{elbow}, $P_{wrist}(z)$, α, and β of the second type verification test mentioned in Section 2.4) obtained from the two systems (M system and the presented system). These curves show that the movement of the arm is slow, about more than 2 s per one cycle. The purpose for doing this is to eliminate the possible error caused by the low sampling rate of Mediapipe Pose. Finally, the root mean square errors of three parameters α, β, and $P_{wrist}(z)$ (the z-coordinate of the wrist joint) between two systems were calculated.

Based on the experiment with six repetitions of lateral raise on the frontal plane, the root mean square errors between the two systems are 2.434 cm for $P_{wrist}(z)$, 5.654° for α, and 4.385° for β.

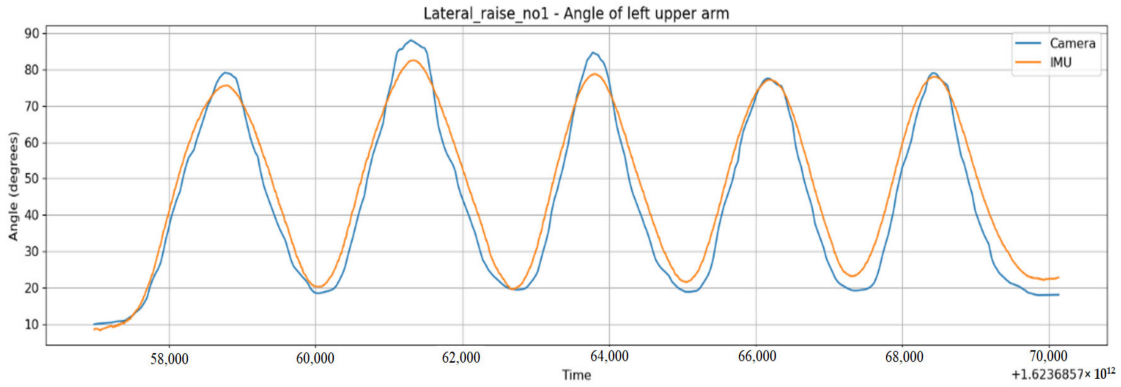
3.3. Posture Recognition Verification

Ten participants were in the training group, and their motion data were used to train the neural network model. Two participants were in the test group, and their data were used to verify the accuracy of the trained ML model for posture recognition.

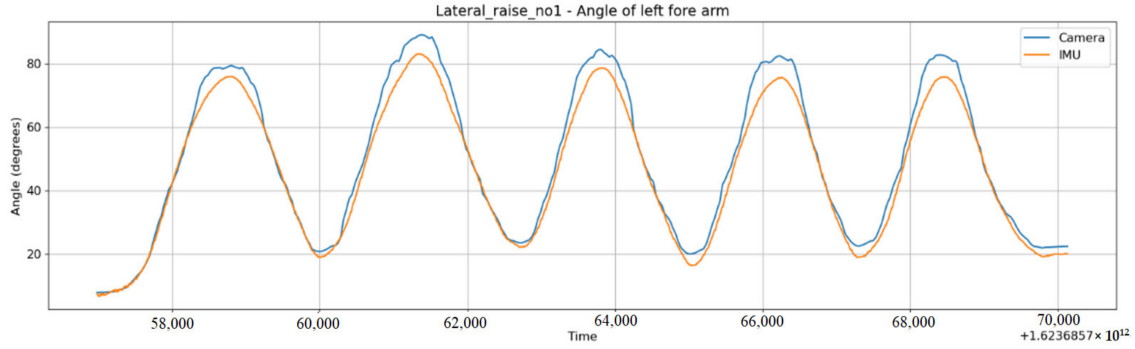
Among the data of 10 participants, 80% of the data was used for training (19,970 samples), and the remaining 20% was used for validation (4993 samples). The mean squared error loss function was used to evaluate the training model at each epoch and update the neural network parameters. Once the model was trained, the data in the test group were used to verify the accuracy of the posture recognition algorithm. The prediction results based on two participants in the test group (3359 samples) were evaluated using Python's scikit-learn library, as shown in Table 5. The evaluation metrics include Precision, Recall, and F1-score, and the accuracy is defined as the ratio of correctly predicted *samples* by the total *samples*. The last column of Table 5 indicates the number of samples for each different posture. The confusion matrix is shown in Figure 12. The accuracy is 98.96%. Among 19,970 samples of training data, there are 9141, 3244, 4122, and 3463 samples for rest (0), bicep curl (1), lateral raise (2), and shoulder press (3), respectively.



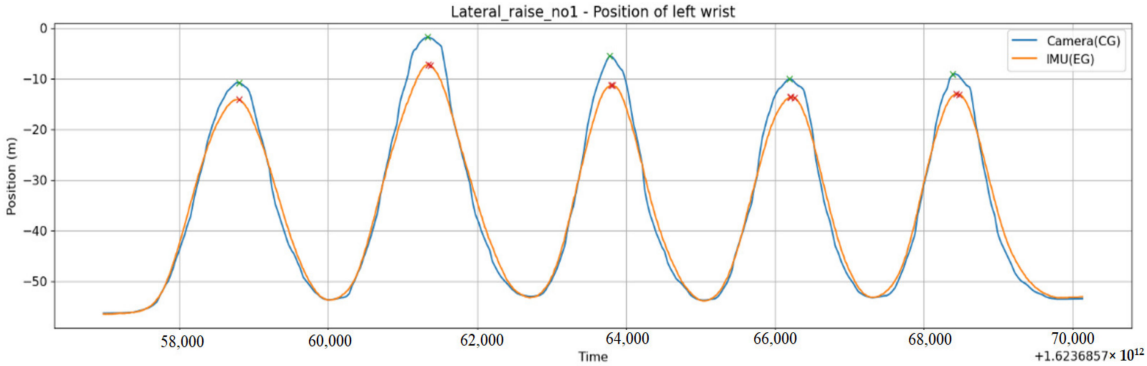
(**a**)



(**b**)



(**c**)



(**d**)

Figure 11. Comparison of the experimental results of the two motion tracking systems. (**a**) 2-D motion trajectory (**left**: M system, **right**: the presented system); (**b**) curves for α; (**c**) curves for β; (**d**) curves for wrist position.

Table 5. Evaluation metrics table of deep learning model.

Performance / Label	Precision	Recall	F1-Score	Data Samples
rest (0)	0.98	1.00	0.99	836
bicep curl (1)	1.00	0.96	0.98	878
lateral raise (2)	1.00	1.00	1.00	778
shoulder press (3)	0.98	1.00	0.99	867

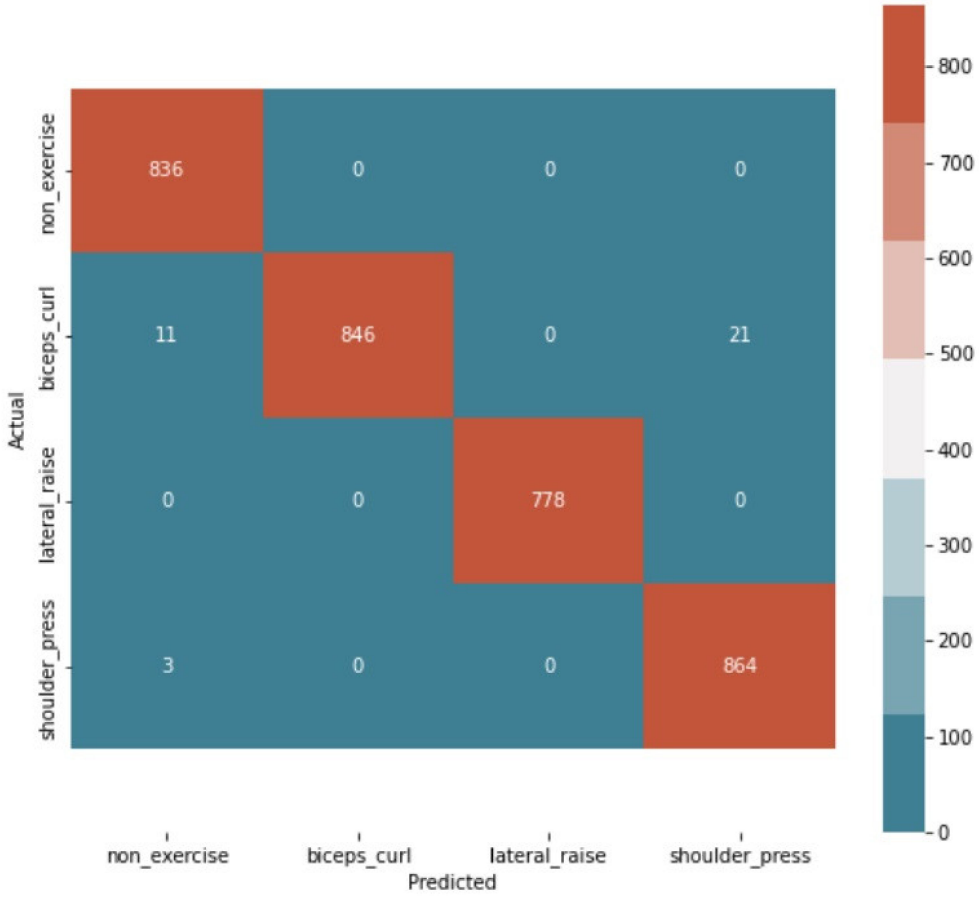


Figure 12. Confusion matrix of test data.

3.4. Results of Exercise Prescription Variables

After establishing a Bluetooth connection, aligning the user's face with the Unity avatar's face, and opening the 9-axis feature channel to listen to the notification data from the microcontroller, the motion data will be input to the mobile App algorithm to calculate the position of the elbow and wrist. The arms of the avatar will be synchronized with the user's movement as shown in Figure 3.

As shown in Figure 13, during the user's exercise, the App of the exercise prescription variables algorithm can automatically analyze the state, calculate the number of repetitions, identify the exercise modes, and trigger exercise events. When a repetitive event occurs, it can update each exercise prescription variable in the current set and refresh the interface. When the rest state ends, the next set of exercise starts and the start event is triggered; it can update the rest time between sets, add a new row (set) of prescription variables, and refresh the interface. The top figure in Figure 13 shows the exercise menu the user can choose and the API indicating wrist position curve, repetition times, accumulated work, exercise variables of each set, etc. The bottom figure of Figure 13 shows the changes of exercise information when a new set of exercise starts or the exercise ends.

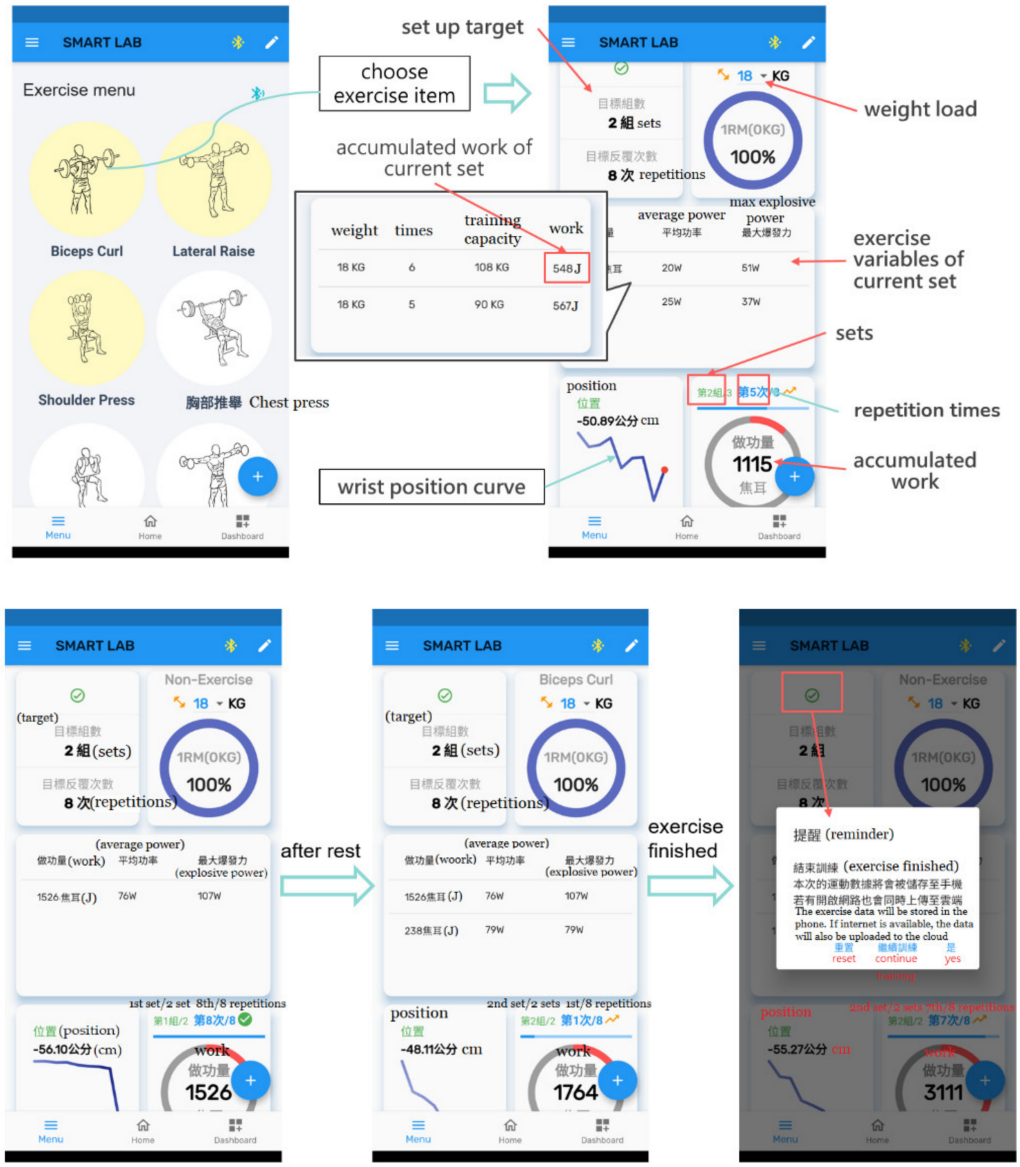


Figure 13. The exercise menu the user can choose and the API indicating wrist position curve, repetitions, accumulated work, exercise variables of each set, etc.

When an exercise is finished, the user can click the green circled check mark in the upper left portion of the API (as shown in the bottom figure of Figure 13) to save the exercise prescription variables in the SQLite database. Then, as shown in Figure 14, users can search all the past exercise records saved in the database, search the history by date or exercise item, and view the exercise prescription variables of a past exercise. The data can also be presented as graphical trends or bar charts, which allow users to analyze their progress and arrange exercise plans to match their needs.

The wearable wrist band can also detect heart rate. Figure 15 shows the dashboard page of the App, where the user can subscribe to the heart rate data. After establishing a Bluetooth connection, the user can turn on or off the heart rate data channel on the microcontroller by clicking the heart rate button at the top right (red circle) of the dashboard page. The first row on the left of the dashboard shows the output probability vector from the machine learning model. The second row shows the posture label with probability greater than 0.5 in the output probability vector, which is considered as the current recognition result (posture label). Other functions on this dashboard such as subscribing the RTC of MCU on the wrist band (light cyan circle in Figure 15) and setting up time are also available.

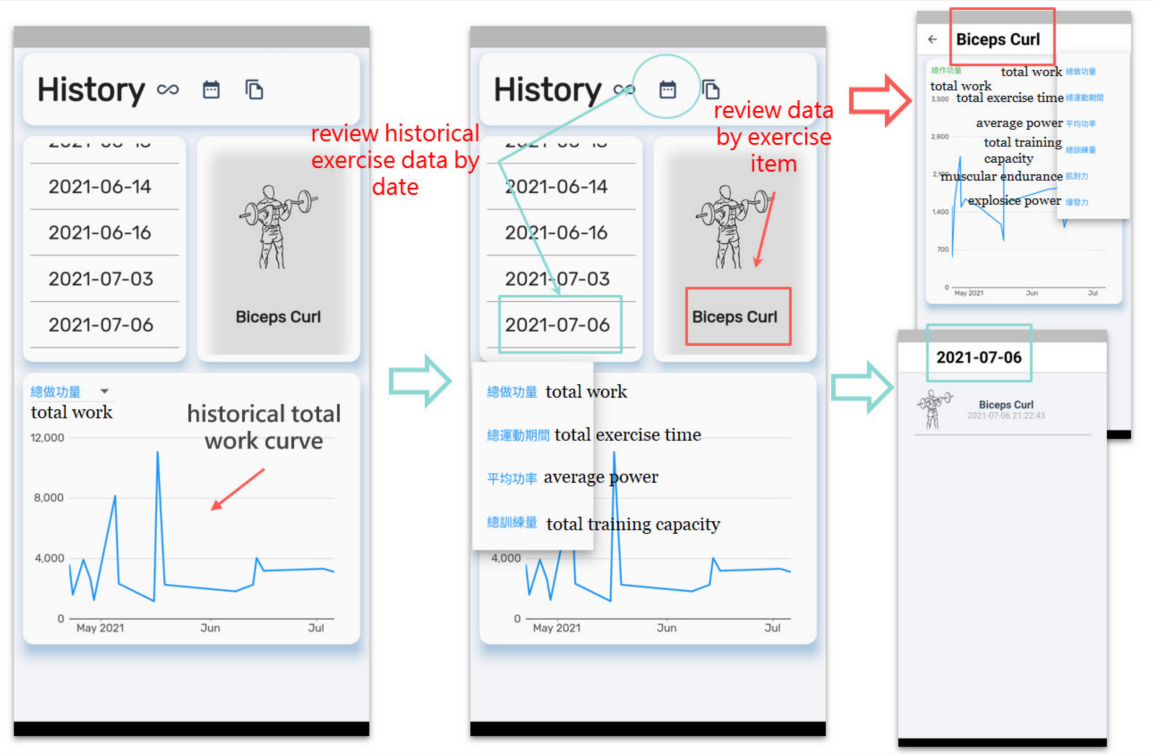


Figure 14. GUI of historical data.

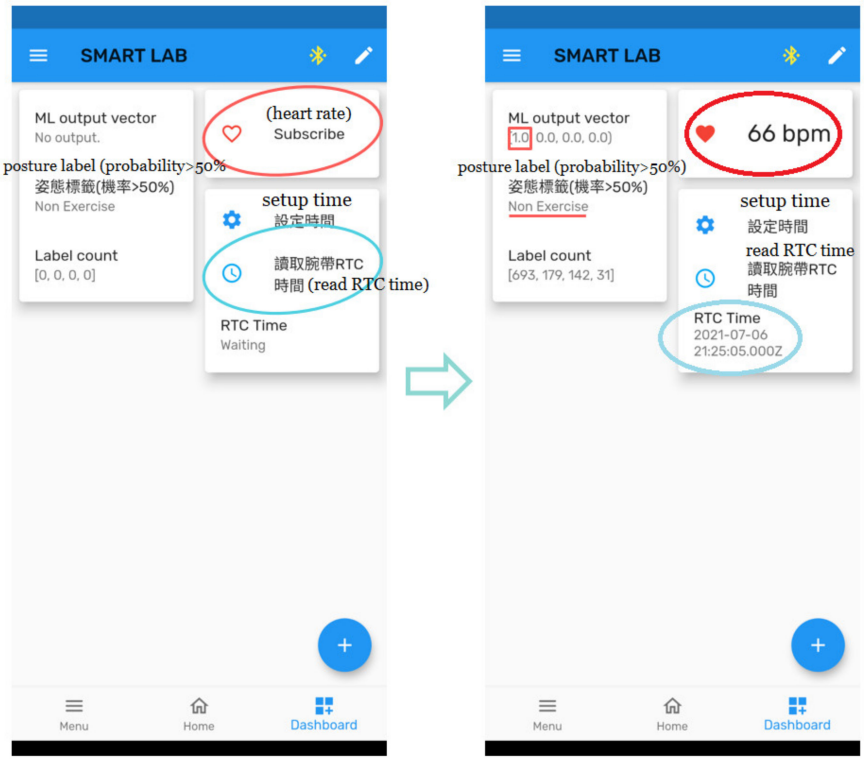


Figure 15. Dashboard to subscribe heart rate data.

Three sets of weight training were performed by one participant in each of the three weight exercises: bicep, side planks, and shoulder press. Table 6 shows an example

of training results, showing the exercise prescription variables and rest periods for a bicep curl.

Table 6. Exercise prescription variables for biceps curl.

Set \ Variables	$\mathbf{m}_{load}(\mathbf{kg})$	N_{reps}	$V_{tra}(\mathbf{kg})$	$W_{rep}(\mathbf{J})$	$P_{rep}(\mathbf{W})$	$P_{max}(\mathbf{W})$
1	18	8	144	743	37	52
rest			119 (s)			
2	18	6	108	645	32	55
rest			108 (s)			
3	18	4	72	463	22	39

4. Discussion

4.1. Motion Tracking Verification

Two types of verification tests were conducted to show the reliability of the presented IMU measurements. The rotation angle of the IMU was tested by a controlled stepper motor, and an average mean absolute error of 1.485 degrees was observed. The three parameters α, β, and $P_{wrist}(z)$ of the presented system were compared with the M system, and the root mean square errors of 5.654°, 4.385°, and 2.434 cm were found, respectively. A mean absolute error of 3.88 cm was reported in [24] in which the wrist positions of elbow flexion-extension (E_{FE}) measured by nine-axis sensors were compared with the optical motion capture (OMC) system. These results show that the measurements obtained both from the presented system and from [24] provide an acceptable motion tracking within an engineering standard. The waveforms between the M system and the presented system, as shown in Figure 11, have similar trends. However, there is a slight offset between the angular value and the vertical axis (z-axis), which might be caused by the inherent sensor configuration error. Because the size of each person's arm is different, when the user wears sensors on his/her arm, the physical offset occurs. The sensor spindle and the arm orientation vector ($\vec{v}_{upper}$ and $\vec{v}_{forearm}$) cannot be perfectly parallel to each other, and the error of this angle will affect the results of the subsequent computation. The muscle changes in the arm during exercise also cause the IMU data worn on the upper arm to drift dynamically. Moreover, it is still not possible to make the plane of arm movement exactly parallel to the lens surface (the M system is based on a two-dimensional plane), and the arm joint marker point of the M system will have uncontrollable dynamic shift during the movement. Therefore, to improve the reliability of the system, we can focus on eliminating the error of the angles α and β. The sensor configuration error can be eliminated by comparing the developed device with more reliable reference system (e.g., Vicon 3D motion capture system) and then calibrating it using computer software to improve the motion tracking algorithm. The experimental results show that there are inherent system errors in both systems. Although the experiments have been designed to minimize the impact of system errors on the experimental results, it is still impossible to completely eliminate the system errors.

4.2. Posture Recognition Verification

During the course of this study, we found that the developed one-layer LSTM is simpler and more accurate than the other LSTM-based variants: two-layer LSTM, Bidirectional LSTM (one layer and two layers), and Bidirectional LSTM with attention. The presented one-layer LSTM model was trained by 19,970 samples and validated by 4993 samples. The prediction accuracy of the trained LSTM was based on 3359 samples. An accuracy of 99% was achieved, which is good enough for our weight training posture recognition. On the other hand, the accuracy of two-layer LSTM is 97.02%, the accuracy of one-layer Bidirectional LSTM is 97.26%, the accuracy of two-layer of Bidirectional LSTM is 97.37%, and the accuracy of Bidirectional LSTM with Attention is 96.12%. Furthermore, when

compared with [7,10,12,17], the accuracy of one-layer LSTM presented in this paper is still promising. In [7], the highest accuracy of 96.4% was found based on one-sensor worn on one upper arm, and 97.6% accuracy based on two sensors on the upper arm and the thigh. In [10], 95% classification accuracy was achieved by using five sensors, including an IMU mounted on the chest, a smartwatch attached to the left wrist, a smartphone attached to the upper left arm, and the eSense earbud attached to the left ear, for five indoor and outdoor exercises. In [12], several classification approaches were used, based on 715 repetitions data from 10 users, for seven different dumbbell exercises with different performance: accuracy of 96.9% and F1 score of 0.923 for hidden Markov models, and accuracy of 99.7% and F1 score of 0.989 for the nearest centroid classifier. In [17], a non-machine-learning method was used to recognize 15 sets of training programs, and 96.07% accuracy was achieved. The presented one-layer LSTM still offers a promising prediction accuracy.

4.3. Exercise Prescription Variables

The developed App can process exercise data from IMU and provides exercise prescription variables: exercise items, repetitions, number of sets, training capacity, work, average power, explosive power, and rest time between sets. Although a sequential minimization algorithm and a stochastic forest method to classify activities in the USC-HAD was proposed in [6], neither IMU-based device nor exercise prescription variables were proposed. Several different supervised machine learning models were tested to classify four fitness sports in [7]; however, no exercise prescription variables were considered. Other papers such as [10,12] provided either no exercise prescription variables or only repetitions. As stated in [22], good exercise prescription variables of exercise training can provide correct feedback to the exercise operator and motivate him/her to become better. The developed App not only can provide real-time exercise prescription variables but also can offer the exercise history output information (exercise prescription variables) from the exercise state analysis and exercise event handling program. These variables can be used as a reference for individual's long-term training progress and transformed into a momentum of keeping exercise habit. The outcomes of exercise information from the designed App can be further shared to a social media group to receive social support from friends and relatives. The recorded avatar movement can be replayed for correcting any wrong-doing exercise movement. The heart rate module in the presented wearable device can also be used for fat burning heart rate calculation. Working together with InBody data [25] (including weight, muscle mass, body fat percentage, protein, bone mass, basal metabolic rate, visceral fat, and total body water content), the designed system can be used as a basis for exercise performance evaluation and provides recommendations for exercise and fitness goals. This system can be even applied in the future to medical rehabilitation or sports training where correctness of movement is of importance.

4.4. The Limitation of the Study

Although two types of verification tests were conducted to show the reliability of the presented IMU measurements, for more reliable motion capture comparisons, a 3D OMC (such as VICON motion analysis system) is required. Since such an OMC is expensive and beyond the affordable ability of this project; the Mediapipe Pose was resorted to as a comparison reference for motion tracking. Although Damindarov et al. [26] reported that the relative errors between the real size (1050×620 mm^2) and the computed size of the screen using Mediapipe Pose are 5.06% for the width and 1.11% for the height with a standard deviation in measurements within 10 mm (except for one point), these figures would just give us some idea about the performance of Mediapipe Pose.

5. Conclusions

This paper successfully developed an integrated system for evaluating and recording weight training exercise performance. The wearable devices based on two IMUs and a heart rate module along with a developed artificial intelligence algorithm and a Unity's

Animation System on the smartphone can provide weight-training lovers with helpful exercise prescription variables that can be used as progressive recommendations to improve the training effect. A simple yet effective LSTM model for posture recognition is also presented, providing a promising accuracy of 99%. The reliability of IMU measurements was also tested by two types of experiments. Test results show that measurements of the IMU are within the engineering acceptable range.

In the future, the presented system can be extended for lower limb exercises. Its applications can also be extended to medical rehabilitation and sports training.

6. Patents

The work presented in this paper will be filed for a patent in Taiwan.

Author Contributions: Conceptualization, Y.-C.W. and S.-X.L.; methodology, Y.-C.W. and S.-X.L.; software, S.-X.L.; validation, Y.-C.W., S.-X.L., C.-C.H. and C.-S.C.; formal analysis, S.-X.L.; investigation, Y.-C.W. and S.-X.L.; resources, Y.-C.W. and S.-X.L.; data curation, S.-X.L. and J.-X.J.; writing—original draft preparation, Y.-C.W. and S.-X.L.; writing—review and editing, Y.-C.W.; visualization, Y.-C.W. and S.-X.L.; supervision, Y.-C.W., J.-Y.L., C.-C.H. and C.-S.C.; project administration, Y.-C.W.; funding acquisition, Y.-C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, grant number 108-2622-E-239-006-CC3.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Wong, R.; Yang, L.; Szeto, W. Wearable Fitness Trackers and Smartphone Pedometer Apps: Their Effect on Transport Mode Choice in a Transit-oriented City. *Travel Behav. Soc.* **2021**, *22*, 244–251. [CrossRef]
2. Kong, X. INS algorithm using quaternion model for low cost IMU. *Robot. Auton. Syst.* **2004**, *46*, 221–246. [CrossRef]
3. Han, S.; Wang, J. A Novel Method to Integrate IMU and Magnetometers in Attitude and Heading Reference Systems. *J. Navig.* **2011**, *64*, 727–738. [CrossRef]
4. Manon, K.; Jeroen, D.H.; Thomas, B.S. Using Inertial Sensors for Position and Orientation Estimation. *Found. Trends Signal. Processing* **2017**, *11*, 1–153.
5. Bergamini, E.; Ligorio, G.; Summa, A.; Vannozzi, G.; Cappozzo, A.; Sabatini, A.M. Estimating Orientation Using Magnetic and Inertial Sensors and Different Sensor Fusion Approaches: Accuracy Assessment in Manual and Locomotion Tasks. *Sensors* **2014**, *14*, 18625–18649. [CrossRef] [PubMed]
6. Tahir, S.B.u.d.; Jalal, A.; Batool, M. Wearable Sensors for Activity Analysis using SMO-based Random Forest over Smart home and Sports Datasets. In Proceedings of the International Conference on Advancements in Computational Sciences, Lahor, Pakistan, 17–19 February 2020; pp. 1–6.
7. Preatoni, E.; Nodari, S.; Lopomo, N.F. Supervised Machine Learning Applied to Wearable Sensor Data Can Accurately Classify Functional Fitness Exercises Within a Continuous Workout. *Front. Bioeng Biotechnol.* **2020**, *8*, 664. [CrossRef] [PubMed]
8. Nwekeab, H.F.; Teh, Y.W.; Al-garadi, M.A.; Alo, U.R. Deep Learning Algorithms for Human Activity Recognition Using Mobile and Wearable Sensor Networks: State of The Art and Research Challenges. *Expert Syst. Appl.* **2018**, *105*, 233–261. [CrossRef]
9. Sui, J.D.; Chang, T.S. Deep Gait Tracking With Inertial Measurement Unit. *IEEE Sens. Lett.* **2019**, *3*, 7002404. [CrossRef]
10. Ishii, S.; Nkurikiyeyezu, K.; Yokokubo, A.; Lopez, Z. ExerSense: Real-Time Physical Exercise Segmentation, Classification, and Counting Algorithm Using an IMU Sensor. In *Activity and Behavior Computing. Smart Innovation, Systems and Technologies*; Ahad, M.A.R., Inoue, S., Roggen, D., Fujinami, K., Eds.; Springer: Singapore, 2020; Volume 204. [CrossRef]
11. Koskimäki, H.; Siirtola, P. Recognizing Gym Exercises Using Acceleration Data from Wearable Sensors. In Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, Orlando, FL, USA, 9–12 December 2014.
12. Hausberger, P.; Fernbach, A.; Kastner, W. IMU-based Smart Fitness Devices for Weight Training. In Proceedings of the Annual Conference of Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 5182–5189.

13. Seong, J.H.; Choi, Y. Design and Implementation of User Interface through Hand Movement Tracking and Gesture Recognition. In Proceedings of the International Conference on Information and Communication Technology Convergence, Jeju, Korea, 17–19 October 2018; pp. 552–555.
14. Chen, Y.L.; Yang, I.J.; Fu, L.C.; Lai, J.S.; Liang, H.W.; Lu, L. IMU-based Estimation of Lower Limb Motion Trajectory with Graph Convolution Network. *IEEE Sens. J.* **2021**, *21*, 24549–24557. [CrossRef]
15. Abdelhady, M.; van den Bogert, A.J.; Simon, D. A High-fidelity Wearable System for Measuring Lower-limb Kinetics and Kinematics. *IEEE Sens. J.* **2019**, *19*, 12482–12493. [CrossRef]
16. Wang, L.; Sun, Y.; Li, Q.; Liu, T. Estimation of Step Length and Gait Asymmetry Using Wearable Inertial Sensors. *IEEE Sens. J.* **2018**, *18*, 3844–3851. [CrossRef]
17. Zou, Y.; Wang, D.; Hong, S.; Ruby, R.; Zhang, D.; Wu, K. A Low-Cost Smart Glove System for Real-Time Fitness Coaching. *IEEE Internet Things J.* **2020**, *7*, 7377–7391. [CrossRef]
18. Rodriguez, A.; Rabunal, J.R.; Pazos, A.; Sotillo, A.R.; Ezquerra, N. Wearable Postural Control System for Low Back Pain Therapy. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 4003510. [CrossRef]
19. Pereira, A. 3D Arm Inertial Sensor-Based 3D upper Limb Motion Tracking and Trajectories Reconstruction. 2016. Available online: https://repositorio-aberto.up.pt/bitstream/10216/85094/2/139165.pdf (accessed on 26 June 2021).
20. Unity. 2022. Available online: https://unity.com (accessed on 1 November 2021).
21. Picerno, P. Good Practice Rules for the Assessment of the Force-Velocity Relationship in Isoinertial Resistance Exercises. *Asian J. Sports Med.* **2017**, *8*, e15590.
22. Weakley, J.; Mann, B.; Banyard, H.; McLaren, S.; Scott, T.; Garcia-Ramos, A. Velocity-Based Training: From Theory to Application. *Strength Cond. J.* **2021**, *43*, 31–49. [CrossRef]
23. MediaPipe Pose. 2020. Available online: https://google.github.io/mediapipe/solutions/pose.html (accessed on 27 June 2021).
24. Filippeschi, A.; Schmitz, N.; Miezal, M.; Bleser, G.; Ruffaldi, E.; Stricker, D. Survey of Motion Tracking Methods Based on Inertial Sensors: A Focus on Upper Limb Human Motion. *Sensors* **2017**, *17*, 1257. [CrossRef] [PubMed]
25. InBody. Available online: https://inbodyusa.com/ (accessed on 27 June 2021).
26. Damindarov, R.; Boby, C.A.; Fahim, M.; Klimchik, A.; Matsumaru, T. A depth camera-based system to enable touch-less interaction using hand gestures. In Proceedings of the International Conference on Nonlinearity, Information and Robotics, Innopolis, Russia, 26–29 August 2021.