Int. J. Advance Soft Compu. Appl, Vol. 10, No. 2, July 2018
ISSN 2074-8523

Developing an Accelerometer-based Mobile Application to Aid Knee Extension Exercise

Rong Phoophuangpairoj

Department of Computer Engineering, College of Engineering,
Rangsit University, Pathum Thani, Thailand
e-mail: rong.p@rsu.ac.th

Abstract

Knee pain is a medical condition that disturbs many individuals throughout the world. Physicians prescribe knee exercises to restore mobility and strength to painful and weakened knees. However, some sufferers may find these exercises dull, which might affect their motivation. Currently, smartphones are ubiquities and they come equipped with sensors such as accelerometers. In this research, smartphone accelerometer data are used to compute knee angles and Google's TensorFlow machine learning software library for machine intelligence is applied to derive a linear regression equation. The resulting algorithm is the basis of a mobile application, which was developed to count the number of times patients' legs are raised and lowered correctly, calculate the length of time the legs are held, count the number of times the legs are held correctly and incorrectly, compute the percentage of correct leg holds, warn patients when leg movement is outside acceptable thresholds and evaluate the overall effectiveness of the exercise by giving it a score. The mobile application allows patients to recover more rapidly because knee extension exercises can be monitored and adjusted through feedback to improve their efficiency.

Keywords*: accelerometer, Android application, knee extension, machine learning,* regression equation, TensorFlow.

1 Introduction

Accelerometers have been utilized by researchers to recognize actions and movement [1-10]. For example, mobile phone accelerometers have been applied to recognize activities such as walking, jogging, going upstairs and downstairs,

sitting, standing and falling [11][12]. Smartphones have been used to reduce prosthetic hip dislocation caused by improper posture [13]. Nowadays, billions of people throughout the world carry their smartphones with them throughout the day and both budget and flagship models come equipped with accelerometers making the technology accessible to almost all users.

A knee extension exercise is carried out by extending the knee while sitting in a chair with the back of the body straight. The knee is held in an extended position for predetermined seconds before it is lowered back to its original position, then the steps are repeated. The number of times the knee is extended and the exactness of the extended leg position are significant factors, which directly affect the usefulness of the exercise. TensorFlow [14] is an open-source software library for numerical computation and machine intelligence originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research. It is not a straightforward tool with clickable menus that requires only data and parameters to be entered, extensive prior knowledge of machine learning, artificial intelligence, and programming is required. In this research, TensorFlow was used to compute the equation parameters. The knee angles were determined from the accelerometer data. In addition, a mobile application was developed which, uses the equation to keep track of the angles when patients raise, hold and lower their legs. Consequently, knee extension exercises can be monitored so that patients are alerted when they move their legs improperly. Inappropriate exercises could result in knee pain taking longer to heal. The mobile application allows patients to recover faster from knee injuries because they can monitor the effectiveness of their exercises. In this research, the performance of the equations obtained from TensorFlow and linear regression to compute knee angles are compared. Then, the method used to develop the knee extension exercise mobile application is presented.

2 Related Work

Researchers have used accelerometers in smartphones to detect triaxial motions and identify human activities [1]. A smartphone with an accelerometer attached to a head-mounted display could detect walk, run, lower half, upper half, jump and full vertical activities [15]. However, research on using smartphone accelerometers to detect poor posture during rehabilitation has been limited to only a few papers. Patients who exercise when physiotherapists and physicians are unavailable to supervise the outcome could further damage their knees and delay the healing process. Accelerometer data has been used to estimate unsuitable postures for patients who had undergone hip surgery. Rules using values obtained from an accelerometer were used to identify unsuitable postures [13]. Knee extension exercises have also been studied [16]. In the research, stored triaxial accelerometer data were analyzed to determine the degree the leg was held at, the

length of time the leg was held for, and the angular velocity the leg was lowered and raised at. A mobile application which activates an alert and provides feedback in real time on the effectiveness of knee extension exercises could help those with injured knees to heal more rapidly.

Machine learning gives computer systems the ability to learn and iteratively enhance the performance of a specific task with data. A support vector machine (SVM), artificial neural networks, regression tree analysis and k-nearest neighbor (k-NN) were applied to recognize the characteristic features of falls [17][18]. Deep learning recently broke records in speech and image classification; however, it has not been fully studied as a potential approach to analyzing wearable sensor data. Deep learning using Google's TensorFlow has been applied to recognizing patients with idiopathic Parkinson's disease where its classifiers outperformed those using AdaBoost.M1, PART, k-NN and SVM [19]. Currently, due to the emergence of new technology, a mobile application to determine the effectiveness of pain relief exercises should be developed.

3 Materials and Method

3.1 Instruments

Fig. 1 shows the instruments used to collect the accelerometer data. They consisted of a digital protractor, a smartphone and the Android-based Sensor Fusion application created by Linköpins Universitet as described in [16]. First, a Samsung Galaxy S6 smartphone was attached with an armband to the digital protractor. Next, the top of the smartphone was aligned parallel to the floor.

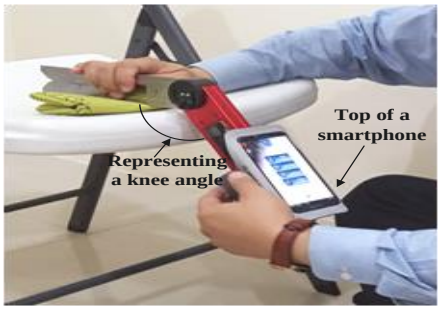

Fig. 1: Collecting data from the accelerometer

The angle of the digital protractor was varied and the Sensor Fusion application, which was downloaded from the Google Play Store, was used to record the sensor data. A Python program was written using TensorFlow and Microsoft Visual Studio Community 2017 to derive an equation, which could compute knee angles on mobile devices. Android Studio [20] and the Java SE Development Kit (JDK) [21] were used to develop the mobile application. Android Studio is the officially integrated development environment for Android and can be used with the JDK for building applications on Android devices.

3.2 Deriving an equation to compute the knee angle

The angle of the digital protractor (representing the knee angle) was incremented in steps of 5 degrees from 80 to 190 degrees. At each 5-degree step, the accelerometer data was recorded for 10 seconds. Then, for each degree, the average values of *X*, *Y* and *Z* were computed. Microsoft Excel and TensorFlow were used in the data analysis to derive equations that could predict the knee angle from the accelerometer data. However, the accelerometer *X* value did not affect the prediction performance. Therefore, only the accelerometer *Y* and *Z* values were used. Errors in the equations obtained from the TensorFlow and linear regression analysis were compared. A Python program was written, which used TensorFlow to find the equation parameters, as shown below.

```
#Import statements for the TensorFlow program
import numpy as np
import tensorflow as tf

# Model parameters (create weights, a bias, and initialize their values)
W1 = tf.Variable([.3], dtype=tf.float32)
W2 = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([.3], dtype=tf.float32)

# Model inputs and output
# Create placeholders for inputs and an output
# (inputs y (accelerometer y value) and z (accelerometer z value), output angle)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)
angle = tf.placeholder(tf.float32)

# Construct the model
# Scale y and z to the range of -1 and 1 by the maximum values of the accelerometer data
# obtained from accelerometer y and z values (10.54922853)
linear_model = W1*y/10.54922853 + W2*z/10.54922853 + b

# loss (use the square error as the loss function)
# angles were scaled to values of not more than 1 by dividing by 1,000
# When using the obtained equation, the predicted angle must be multiplied by 1,000 later
loss = tf.reduce_sum(tf.square(linear_model - angle/1000)) # sum of the squares

# optimizer (using gradient descent with a learning rate of 0.01 to minimize the loss)
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

# Training data (y_train, z_train and angle_train)
y_train = [-9.854918276, -9.935656452, -9.930376719, -9.876094528, -9.609847081, -9.400428557,
-9.188858565, -8.606071056, -8.281913096, -7.646664423, -7.144252027, -6.421738093, -5.947865058,
-5.36909638, -4.225071377, -3.933800704, -2.958510073, -2.402172669, -1.039182786, -0.526669963,
-0.015127189, 1.036411793, 2.432224464]
z_train = [-0.544567652, 0.425204723, 1.131641737, 1.809724306, 3.181905278, 3.890094311,
4.442849749, 5.612897496, 6.14114012, 6.920205372, 7.449510166, 8.248553807, 8.603829062,
9.018231482, 9.639302822, 9.770692485, 10.12989709, 10.27038887, 10.46382032, 10.53568769,
10.54922853, 10.49447203, 10.20460383]
angle_train = [80.0, 85.0, 90.0, 95.0, 100.0, 105.0, 110.0, 115.0, 120.0, 125.0, 130.0, 135.0, 140.0, 145.0,
150.0, 155.0, 160.0, 165.0, 170.0, 175.0, 180.0, 185.0, 190.0]
```

```
# Training loop (20000 epochs)
# Model parameters are stored in curr_W1, curr_W2 and curr_b
# Model loss is stored in curr_loss
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
for i in range(20000):
  sess.run(train, {y:y_train, z:z_train, angle:angle_train})

# Evaluate the training accuracy (from a value obtained from the loss function (curr_loss))
curr_W1, curr_W2, curr_b, curr_loss = sess.run([W1, W2, b, loss], {y:y_train, z:z_train, angle:angle_train})
print("W1: %s   W2: %s  b: %s loss: %s"%(curr_W1, curr_W2, curr_b, curr_loss))
```

TensorFlow was used to find the equation to compute the knee angles. The data were scaled between the range of -1 and 1 by dividing the values of *Y* and *Z* with the maximum value (magnitude). The loss from the model was calculated to find the equation parameters. The loss measured how far away its prediction was from the desired output, in other words, how poorly the model or equation had performed. To obtain the model, the loss value needed to be minimized or optimized. First, the training process attempted to find the best combination of weights and a bias to minimize the loss. Then, a gradient descent optimization algorithm was used to find optimal model variables that reduced the prediction error or its loss. Next, to minimize the loss, an optimizer used the computed gradients to find the model's variables. The model was adjusted iteratively by computing the loss and gradients at each step. The optimizer generated the weight and bias parameters of the equation to minimize the loss function 20,000 times. At the training stage, the output angles were scaled to values that were not more than 1 by dividing by 1,000. Subsequently, the obtained angle was multiplied by 1,000 to reflect the original value.

3.3 Developing an application to aid knee extension exercises

The mobile application was developed to give feedback on the knee extension exercise. The factors that affected the knee extension exercise were the knee angle, the number of correct leg-raise and leg-lower cycles, the number of correct leg holds, the number of incorrect leg holds and the percentage of correct leg holds, as shown below.

3.3.1 Computing the knee angle

Accelerometers report a sequence of three values, which are *X*, *Y* and *Z*. The results section shows that the knee angle (*KA*) could be approximated from the accelerometer data by the following equation, which was derived using TensorFlow.

$$KA = (W_1Y/10.54922853 + W_2Z/10.54922853 + b) * 1000 \quad (1)$$

Y : Accelerometer *Y* value *Z* : Accelerometer *Z* value

W_1 : 0.0529054 W_2 : 0.04322108 *b* : 0.13487998

Alternatively, the knee angle could also be computed from the accelerometer data using the following regression equation obtained by the method proposed in [16]. The errors obtained when using the TensorFlow and linear regression equations are compared in the results section.

$$KA = 5.0150259663567Y + 4.09716153510471Z + 134.879035721248 \quad (2)$$

Y : Accelerometer Y value Z : Accelerometer Z value

The KA value represented the position of the leg when it was raised and lowered, which was computed every time new data were received from the accelerometer. The angle could be less than 90° and greater than 180°. The KA value was used in the mobile application to estimate accurately the lower leg and thigh angles from 90° to 180°.

3.3.2 Alert when raised or lowered leg is out of range

When legs were over or under extended, the application notified the patient with an alert. The method for warning patients when legs are moved out of range is shown below:

Given
σ_{LW}: Leg- lower warning threshold (e.g. 85)
σ_{RW}: Leg- raise warning threshold (e.g. 185)
event.values[1] : accelerometer Y value
event.values[2] : accelerometer Z value

When new values of accelerometer data are received
// public void onSensorChanged(SensorEvent event)
{
    Compute KA by
    KA = (0.0529054*event.values[1]/10.54922853 +
    0.04322108*event.values[2]/10.54922853 + 0.13487998) ***** 1000**;**
      …
      if((KA <= σ_{LW}) OR (KA >= σ_{RW}))
          Generate beep sounds
      …
}

The predefined σ_{LW} and σ_{RW} can be set to values representing angles that are less than 90° and greater than 180°.

3.3.3 Determining leg-hold duration and the number of correct leg holds

The start and finish time of each leg hold was obtained from the system clock and each leg-hold duration was computed from the time difference. The method for determining the leg-hold duration is shown below:

Keep start time (in milliseconds from a reference time).

Do something.

Keep end time.

Compute the difference of time (in milliseconds).

The number of correct and incorrect leg holds was determined using the following method.

Given

σ_{LR} : Low angle threshold for a raised a leg (e.g. 170)
σ_{HR} : High angle threshold for a raised a leg (e.g. 190)
σ_{MCD} : Minimum threshold of correct leg-hold duration (e.g. 5000)
σ_{MID} : Minimum threshold of incorrect leg-hold duration (e.g. 2000)
(Duration (in milliseconds) that is shorter than σ_{MID} is not considered as a leg hold)
N_{CH} : the number of correct leg holds
N_{ICH} : the number of incorrect leg holds
LHD : Leg-hold duration (in milliseconds)
FirstHighDegreeFlag : Flag indicating that a leg has passed a high-degree angle for the first time (for each time of leg raised)
event.values[1] : accelerometer *Y* value
event.values[2] : accelerometer *Z* value
KA : Knee angle

N_{CH} = 0;
N_{ICH} = 0;
FirstHighDegreeFlag= 0;
When new values of accelerometer data are received
// public void onSensorChanged(SensorEvent event)
{ Compute *KA* by
KA = (0.0529054*event.values[1]/10.54922853 +
0.04322108*event.values[2]/10.54922853 + 0.13487998) ***** 1000**;**
if ((*KA* >= σ_{LR}) AND (*KA* <= σ_{HR})) { // A leg is passing a high-degree range or is hold.
Increment *FirstHighDegreeFlag* by 1 // Not 0 if a leg passes a high-degree angle
if(*FirstHighDegreeFlag* == 1) {
Keep start time
}
…
Keep the end time
Compute *LHD* (which is the leg-hold duration) from the difference of the kept start and finish time
}
…

```
        else {
            if (LHD >= σMCD) {
                Increment NCH by 1
                …
            }

            if ((LHD >= σMID) AND (LHD < σMCD)) {
                Increment NICH by 1
                …
            }

            LHD = 0;
            FirstHighDegreeFlag= 0;
            …
        }
        …
        Display the correctness and effectiveness of the knee extension exercise.
}
```

3.3.4 Determining the number of correct leg raise and lower cycles

The number of correct leg raise and lower cycles (N_{CRL}) can be found from the number of times that a patient moves their leg through the low and high thresholds of the raised leg (σ_{LR} and σ_{HR}) and the low and high thresholds of the lowered leg (σ_{LL} and σ_{HL}). The method used to determine the number of correct raise and lower cycles is shown below:

Given

σ_{LR} : Lower threshold of a raised leg (e.g. 170)
σ_{HR} : Higher threshold of a raised leg (e.g. 190)
σ_{LL} : Lower threshold of a lowered leg (e.g. 80)
σ_{HL} : Higher threshold of a lowered leg (e.g. 100)
N_{CRL} : the number of correct raise and lower cycles
LHD : Leg-hold duration (in milliseconds)
*PassedLowDegreeFlag***:** Flag indicating that a leg has passed a low-degree angle (for each time a leg is lowered)
FirstHighDegreeFlag : Flag indicating that a leg has passed a high-degree angle for the first time (for each time a leg is raised)
event.values[1] : accelerometer *Y* value
event.values[2] : accelerometer *Z* value
KA : Knee angle

PassedLowDegreeFlag = 0;
N_{CRL} = 0;

When the new values of accelerometer data are received
// public void onSensorChanged(SensorEvent event)
{
Compute *KA* by
KA = (0.0529054*event.values[1]/10.54922853 +
0.04322108*event.values[2]/10.54922853 + 0.13487998) ***** 1000**;**
if ((*KA* >= σ_{LR}) AND (*KA* <= σ_{HR})) { // A leg is passing a high-degree range or is held.
…
if (*PassedLowDegreeFlag* == 1){ // A leg has passed a low-degree range.
Increment N_{CRL} by 1
PassedLowDegreeFlag = 0;
}
…
}
else if ((*KA* >= σ_{LL}) AND (*KA* <= σ_{HL})) // A leg is passing a low-degree range.
{ // Can use the σ_{HL} only
PassedLowDegreeFlag = 1;
LHD = 0;
FirstHighDegreeFlag = 0;
}
…
Display the correctness and effectiveness of the knee extension exercise.
}

3.3.5 Computing the percentage of correct leg holds

From the leg-hold duration (*LHD*), the number of correct and incorrect leg holds (N_{CH} and N_{ICH}) were counted and the percentage of correct leg holds (P_{CH}) was computed, as shown below.

Given
N_{CH} : the number of correct leg holds
N_{ICH} : the number of incorrect leg holds

if ($N_{CH} + N_{ICH}$) > 0

$$P_{CH} = \frac{N_{CH}}{N_{CH} + N_{ICH}} x\ 100 \quad (3)$$

3.3.6 Computing the score of the knee extension exercise

After the N_{CH} and N_{CRL} values were obtained, an overall score (*Score*) for the exercise was computed using the equation below.

Given
W_{CH} : Weight of correct leg holds (e.g. 3)
W_{CRL} : Weight of the correct raise and lower cycles (e.g. 1)

N_{CH} : the number of correct leg holds
N_{CRL} : the number of correct raise and lower cycles

$$Score = W_{CH}N_{CH} + W_{CRL}N_{CRL} \quad (4)$$

W_{CH} and W_{CRL} are manually pre-defined weighting values, which represent the significance of the number of correct leg holds and the number of raise and lower cycles, respectively.

To analyze the knee extension exercises in the mobile application, the σ_{LW}, σ_{RW}, σ_{LR}, σ_{HR}, σ_{MCD}, σ_{MID}, σ_{LL}, σ_{HL} thresholds and the W_{CH} and W_{CRL} weighting values must be adjusted to suit physiotherapists' diagnosis.

4 Results

4.1 Equations to determine the knee angle

Table 1 shows the equations used to determine the knee angle from *X*, *Y* and *Z* obtained using regression data analysis in Microsoft Excel and TensorFlow machine learning.

Table 1: Equations to determine the knee angle from *X*, *Y* and *Z*

Method	Results
Regression using regression data analysis in Microsoft Excel	KA = -0.0329195890694764X + 5.0141242004229Y + 4.09999153486268Z + 134.859522862178 R^2 = 0.997536452
Machine learning using TensorFlow (Scale the data to the range of [-1, 1] using a maximum value of 10.54922853)	KA = (W_1X/10.54922853 + W_2Y/10.54922853 + W_3Z/10.54922853 + b) * 1000 W_1 : -0.00033007 W_2 : 0.05290307 W_3 : 0.04324205 b : 0.13486998 Maximum value in the data : 10.54922853 Error (loss) = 6.3227e-05

Table 2 shows the coefficients of *X*, *Y*, *Z*, the intercept value and the corresponding P-value computed in the regression analysis.

Table 2: Coefficients of *X*, *Y*, *Z*, the intercept value and their P-values

	Coefficients	P-value
Intercept	134.859522862178	4.21001E-22
X	-0.0329195890694764	0.976228404
Y	5.0141242004229	3.23129E-16
Z	4.09999153486268	2.34989E-13

The P-value of *X* (0.976228404) was greater than the common alpha level of 0.05, which indicated that it was not statistically significant. Generally, P-values are used to determine the variables that should be kept in the regression model. According to the P-values, *X* was considered for removal from the regression equation. After *X* was removed, linear regression and TensorFlow were applied to find the equations again. The obtained equations are shown in Table 3.

Table 3: Equations used to determine the knee angle from *Y* and *Z*

Method	Results
Regression using regression data analysis in Microsoft Excel	$KA = 5.0150259663567Y + 4.09716153510471Z + 134.879035721248$ $R^2 = 0.997536334$
Machine learning using TensorFlow (Scale the data to the range of [-1, 1] using a maximum value of 10.54922853)	$KA = (W_1Y/10.54922853 + W_2Z/10.54922853 + b) * 1000$ W_1 : 0.0529054 W_2 : 0.04322108 b : 0.13487998 Maximum value in the data : 10.54922853 Error (loss) = 6.23309e-05

Table 4 shows the coefficients of *Y*, *Z*, the intercept value and the corresponding P-value computed in the regression analysis. The obtained P-values ensured that *Y*, *Z* and the intercept value were kept in the regression equation.

Table 4: Coefficients of *Y*, *Z*, the intercept value and their P-values

	Coefficients	P-value
Intercept	134.879035721248	1.73E-23
Y	5.0150259663567	4.17E-17
Z	4.09716153510471	9.6E-15

The linear regression equation obtained shown in Table 4 was used to compare the knee angle computation errors derived from TensorFlow. The results are shown in the section below.

4.2 Knee angle computation errors

Table 5 shows that errors occurred when the equations obtained from TensorFlow and linear regression were used to compute the knee angle from *Y* and Z. The results show that the average differences when using TensorFlow machine learning and regression analysis were 1.273596193° and 1.273601303°, respectively. TensorFlow machine learning was used to efficiently find an equation that could compute the knee angle. The error of the angle determined

from *Y* and *Z* using TensorFlow machine learning was about 1.27360°. Since the errors obtained from TensorFlow machine learning and regression analysis were not significantly different, both equations could be used to compute the knee angles from the accelerometer data.

Table 5: Errors when using equations obtained from TensorFlow and linear regression

Actual knee angle (*KA*) (degree)	*Y*	*Z*	*KA* using TensorFlow machine learning) (degree)	*KA* (Regression) (degree)	Difference (using TensorFlow machine learning) (degree)	Difference (Regression) (degree)
80	-9.854918276	-0.544567652	83.22547333	83.22518303	3.225473334	3.225183032
85	-9.935656452	0.425204723	86.79380289	86.79359306	1.793802886	1.793593057
90	-9.930376719	1.131641737	89.71461335	89.71445762	0.285386646	0.28554238
95	-9.876094528	1.809724306	92.76500532	92.76489803	2.23499468	2.235101967
100	-9.609847081	3.181905278	99.72220328	99.72218299	0.277796716	0.277817009
105	-9.400428557	3.890094311	103.6739677	103.6739872	1.326032329	1.326012809
110	-9.188858565	4.442849749	106.9996973	106.9997445	3.000302682	3.000255485
115	-8.606071056	5.612897496	114.7162174	114.7163136	0.283782623	0.283686371
120	-8.281913096	6.14114012	118.5061555	118.5062696	1.493844513	1.493730426
125	-7.646664423	6.920205372	124.8838851	124.8840143	0.116114944	0.115985651
130	-7.144252027	7.449510166	129.5721382	129.5722728	0.427861817	0.427727195
135	-6.421738093	8.248553807	136.4693646	136.4695098	1.469364609	1.469509813
140	-5.947865058	8.603829062	140.3014764	140.3016155	0.301476368	0.301615499
145	-5.36909638	9.018231482	144.9018989	144.9020291	0.098101106	0.097970898
150	-4.225071377	9.639302822	153.1838765	153.1839738	3.183876499	3.183973801
155	-3.933800704	9.770692485	155.1829416	155.1830285	0.182941604	0.183028466
160	-2.958510073	10.12989709	161.5458099	161.5458556	1.545809913	1.545855595
165	-2.402172669	10.27038887	164.9115024	164.9115196	0.088497586	0.088480362
170	-1.039182786	10.46382032	172.5395333	172.5394692	2.539533309	2.539469192
175	-0.526669963	10.53568769	175.4042813	175.4041865	0.404281296	0.404186525
180	-0.015127189	10.54922853	178.0251957	178.0250658	1.974804322	1.974934175
185	1.036411793	10.49447203	183.0744232	183.0742149	1.92557676	1.925785095
190	2.432224464	10.20460383	188.8869441	188.8866148	1.113055897	1.113385155
				Sum of difference (degree)	29.29271244	29.29282996
				Average of difference (degree)	1.273596193	1.273601303

Remark: The actual knee angle was the value obtained from the digital protractor.

4.3 Android application to aid knee extension exercises

Fig. 2 shows an android application, which was developed as an aid to knee extension exercises. The application reports current accelerometer values (*X*, *Y* and *Z*), the current angle, the number of cycles of correct raise and lower

movements, current leg-hold time, the number of correct and incorrect leg holds, the percentage of correct leg holds, and the overall score of the exercise.

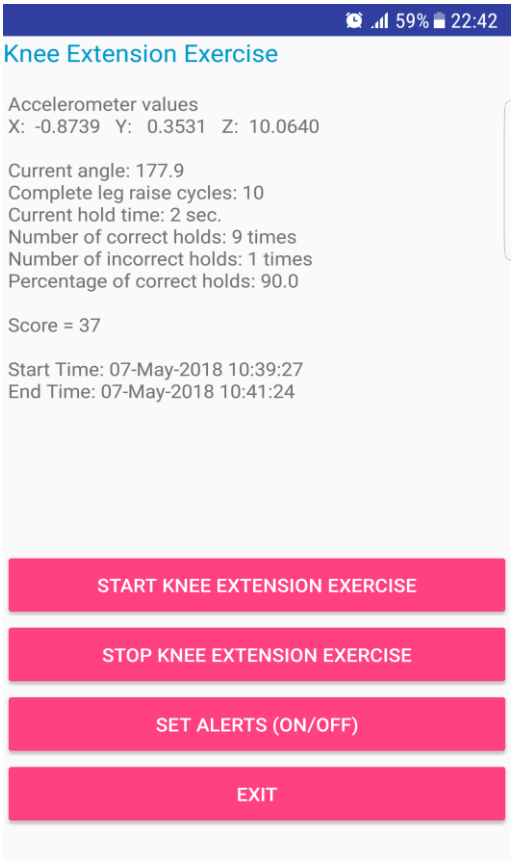

Fig. 2: Android application to aid knee extension exercises

To use the application, first, attach a smartphone with an accelerometer to an armband. Then, adjust the armband to suit the diameter of the leg. Next, fasten the smartphone and the armband to a point on the shin below the injured knee while making sure the top of the smartphone is aligned parallel with the foot, as shown in Fig. 3.

Fig. 3: A smartphone attached to an armband fastened on a leg

Start the application by pressing the START KNEE EXTENSION EXERCISE button on the screen, to begin exercising raise the leg up, hold it in the extended position then lower it back down. During the exercises, patients can set alerts that will notify them when the leg is moved out of the preset range by pressing the SET ALERTS (ON/OFF) button. Finally, pressing the STOP KNEE EXTENSION EXERCISE button will end the program and give the patient feedback on the outcome of the exercise. They can use the feedback from the application to determine whether the exercise was done correctly or not. In addition, smartphone screens can easily be mirrored to smart TVs allowing patients to get feedback in a more enjoyable way.

5 Conclusion

The aim of this research was to develop a mobile application that could be used to make knee extension exercises more effective and enjoyable. Knee pain sufferers may find exercises boring due to their repetitive nature. The mobile application developed in this research encourages outpatients to exercise more appropriately and it may motivate them to restore strength and mobility to their injured knees because the exercises are more enjoyable. Furthermore, they can evaluate and adjust their exercises through feedback. Linear regression and TensorFlow machine learning were investigated to compute knee angles. Both methods could be applied to find equations that computed the knee angle from the accelerometer data with an average error of about 1.27°. TensorFlow machine learning approach is more beneficial than the traditional regression method because it is more flexible and applicable to a wider variety of algorithms such as deep neural network models. In addition, TensorFlow is not limited to only linear regression. It can be used to find the coefficients of various other types of equations. In the future, it is hoped the mobile application will be further developed so that it could be applied to a variety of other exercises such as sit-ups.

References

[1] Bayat, A., Pomplun, M., & Tran, D. A. (2014). A study on human activity recognition using accelerometer data from smartphones. In *Procedia Computer Science*, *34*, 450-457.

[2] Brezmes T., Gorricho J-L., & Cotrina J. (2009). Activity Recognition from Accelerometer Data on a Mobile Phone. In *Lecture Notes in Computer Science*, 5518, 796-799, Springer, Berlin, Heidelberg.

[3] Cho, Y., Nam, Y., Choi, Y-J., & Cho, W-D. (2008). SmartBuckle: human activity recognition using a 3-axis accelerometer and a wearable camera. In *Proceedings the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*.

[4] Krishnan, N. C., & Panchanathan, S. (2008). Analysis of low resolution accelerometer data for continuous human activity recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (pp. 3337-3340).

[5] Lester J., Choudhury T., & Borriello G. (2006). A Practical Approach to Recognizing Physical Activities. In *Lecture Notes in Computer Science*, 3968, 1-16, Springer, Berlin, Heidelberg.

[6] Ravi, N., & Dandekar, N. (2005). Activity recognition from accelerometer data. In *Proceedings 17th Conference on Innovative Applications of Artificial Intelligence (IAAI), 3*, (pp. 1541-1546).

[7] Tapia, E. M., Intille, S. S., et al. (2007). Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proceedings 11th IEEE International Symposium on Wearable Computers*, (pp. 1-4).

[8] Long, X., Yin, B., & Aarts, R. M. (2009). Single accelerometer-based daily physical activity classification. In *Proceedings 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, (pp. 6107-6110).

[9] Mannini, A., & Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, *10*(2), 1154-1175.

[10] Győrbíró, N., Fábián, Á., & Hományi, G. (2009). An activity recognition system for mobile phones. *Mobile Networks and Applications*, *14*(1), 82-91.

[11] Kwapisz, J. R., Weiss, G. M., & Moore S. A. (2010). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, *12*(2), 74-82.

[12] Mathie, M., Celler, B., Lovell, N., & Coster, A. (2004). Classification of basic daily movements using a triaxial accelerometer. *Medical & Biological Engineering and Computing*, *42*, 679-687.

[13] Chutatape, O., Naonueng, K., & Phoophuangpairoj, R. (2017). Detection of improper postures leading to dislocation of hip prosthesis by a smartphone. In *Proceedings 14th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*.

[14] TensorFlow, An open-source software library for machine intelligence, Retrieved from https://www.tensorflow.org/

[15] Huda, F. A., Tolle, H., & Putra, K. P. (2017). Human activity recognition using single accelerometer on smartphone put on user's head with head-mounted display, *Int. J. Advance Soft Compu. Appl*, 9(*3*), 239-249.

[16] Phoophuangpairoj, R. (2016). Frame-based analysis of knee extension exercises using a smartphone accelerometer, In *Proceedings 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*.

[17] Vallejo, M., Isaza, C, & Lopez, J. (2013). Artificial neural networks as an alternative to traditional fall detection methods. In *Proceedings of 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, (pp. 1648-1651).

[18] Putra, I. P. R. S, Brusey, J., Gaura, E, & Vesilo R. (2017). An event-triggered machine learning approach for accelerometer-based fall detection, *Sensors, 18*(20), 1-18.

[19] Eskofier, B. M. (2016). Recent machine learning advancements in sensor-based mobility analysis: Deep learning for Parkinson's disease assessment, In *Proceedings 38th Annual International Conference of IEEE Engineering in Medicine and Biology Society (EMBC)*, Orlando, FL, USA.

[20] Android Studio, Retrieved from https://developer.android.com/studio/index.html

[21] Java SE development kit, Retrieved from http://www.oracle.com/technetwork/java/javase/downloads