Chapter 8

A Self Organizing Map Based Motion Classifier with an Extension to Fall Detection Problem and Its Implementation on a Smartphone

Wattanapong Kurdthongmee

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51002

1. Introduction

Automatic classification of human motions is very useful in many application areas. The characteristics of the motion types and patterns can be used as an indicator of one''s mobility level, latent chronic diseases and aging process [1]. The motion types can be employed to further make a decision if one is at risk; i.e. the motion type or the transition between the motion types may be risky and is likely to cause a fall. Alternatively, the motion types may be useful as an indication to request for a close observation/attention; i.e. a jogging is higher risk and requires a close attention than a normal walk especially for elderly people. The indication may be used to support the feature of a video surveillance system for monitoring elderly people. With respect to these examples of application areas in combination with the requirement to automate monitoring of elderly people as a result of ''aging society'', the demands for an automatic motion classification have been increased. To observe and make relation to motion types, either an acceleration sensor or video system has been widely accepted as useful.

In this chapter, we present the application of the self organizing map (SOM) to an automatic classification of basic, i.e. activities of daily live - ADL, human motions. To be specific, SOM is employed to solve the human motion classification problem. In our proposed approach, SOM is trained with motion parameters captured from a specific wearer and used to perform an adaptive motion types clustering and classification functions. This results in a codebook with different clusters of motion types whose parameters are similar. Instead of using a codebook alone, the algorithm is proposed in order to distinguish between different basic motion types whose motion parameters are similar and mapped to the same cluster as clear-

ly reflected by the codecook. In addition, the frequency of occurrences from different motion types resulted from SOM classifications are used to make decision for the most probable motion type. By mean of SOM, it makes the resulting application adaptable to wearers of different ages and motion conditions. With the success of classification of different motion types, we propose extending the algorithm to perform fall detection. In this case, SOM is trained and labeled with a majority of data from fall curves. As the characteristics of the fall curves is almost similar to the motion types with rapid transition of acceleration, the motion type is employed to support the differentiation between fall and other rapid transition motion types. It can be summarized that the contributions of our research are twofolds. Firstly, the motion types classification algorithm employing SOM is proposed and validated for its correctness with respect to the continuous waveform of mixed motion types. Secondly, the motion types is successfully used to distinguish between fall and other motion types.

The rest of this chapter is organized as follows. Section 2 presents related work which is focused on the class of sensor attachment to body for motion classification and fall detection algorithms. The background of SOM, data capturing and SOM training and labeling stages are detailed in Section 3. The proposed algorithm for motion type classification and fall detection are then explained in Section 4. In Section 5, the validation results from the implementation of the proposed algorithms are given and discussed. In addition, the extensions of the algorithm to the fall detection problem is then given. Eventually, the implementation of the algorithm on the platform of choice for practical use is presented in Section 6. Finally, the chapter is concluded in Section 7..

2. Related Work

In this section, the survey of the previously proposed motion classification and fall detection algorithms are given. In general, the previously proposed motion classification algorithms can be classified into two categories: acceleration sensor based detection and video processing based detection. Motion classification using acceleration sensor has been widely studied and resulted in two difference classification schemes which are the threshold-based and statistical classification schemes [1]. Threshold-based motion classification takes advantage of known knowledge and information about the movements to be classified. It uses a hierarchical algorithm structure, a decision tree like, to discriminate between activity states. A set of empirically-derived thresholds for each classification subclass are required. A systematic approach for motion classification based on a hierarchical decision tree is presented by [2]. A generic classification framework presented in [3] consists of a hierarchical binary tree for classifying postural transitions, falling, walking, and other movements using signals from a wearable triaxial acceleration sensor. This modular framework also allows modifying individual classification algorithm for particular purposes.

Tilt sensing is a basic function provided by acceleration sensors which respond to gravity or constant acceleration. Therefore, human postures, such as upright and lying, can be distinguished according to the magnitude of acceleration signals along sensitive axes from

http://dx.doi.org/10.5772/51002

only one acceleration sensor worn at the waist and torso [4, 6. However, the single-acceleration sensor approach has difficulty in distinguishing between standing and sitting as both are upright postures, although a simplified scheme with tilt threshold to distinguish standing and sitting has been proposed [4]. Standing and sitting postures can be distinguished by observing different orientations of body segments where multiple acceleration sensors are attached. For example, two acceleration sensors can be attached to the torso and thigh to distinguish standing and sitting postures from static activities [7, 8, 9]. Trunk tilt variation due to sit-stand postural transitions can be measured by integrating the signal from a gyroscope attached to the chest of the subject [5]. Sit-stand postural transitions can be identified according to the patterns of vertical acceleration from an acceleration sensor worn at the waist [6].

Acceleration signals can be used to determine walking in ambulatory movement. Walking can be identified by frequency-domain analysis [4, 10]. It is characterized by a variance of over 0.02 g in vertical acceleration and frequency peak within 1–3 Hz in the signal spectrum [10]. Discrete wavelet transform is used to distinguish walking on a level ground and walking on a stairway [11].

Motion classification using statistical schemes utilize a supervised machine learning procedure, which associates an observation (or features) of movement to possible movement states in terms of the probability of the observation. Those schemes include, for example, k-nearest neighbor (kNN) classification [8, 12], support vector machines (SVM) [13, 14], Naive Bayes classifier [15, 16], Gaussian mixture model (GMM) [17] and hidden Markov model (HMM) [18, 19]. Naive Bayes classifier determines activities according to the probabilities of the signal pattern of the activities. In GMM approach, the likelihood function is not a typical Gaussian distribution. The weights and parameters describing probability of activities are obtained by the expectation-maximization algorithm. Transitions between activities can be described as a Markov chain that represents the likelihood (probability) of transitions between possible activities (states). The HMM is applied to determine unknown states at any time according to observable activity features (extracted from accelerometry data) corresponding to the states. After the HMM is trained by example data, it can be used to determine possible activity state transitions.

From our point of view, there are several drawbacks of the previously proposed approaches. That is to say the differences in the collected data among different persons, or even within the same person but different time and sensor variable sampling rate, which are very common, are not taken into account. The previously proposed motion classifiers are almost all in the class of a pre-programmed system with the threshold for making decision from limited samples. This is in contrast to our proposed approach which relies on utilizing SOM to make it adaptable to a particular wearer. The movement nature of the wearer is taken into consideration and used for training and labeling and, in turn, used for fall detection and alert of the wearer. The details of our proposed algorithm are given in the next sections.

3. Background, Data Capturing and SOM Training and Labeling Stages

In this section, we briefly describe the background of SOM. The procedures for capturing data, the SOM training and labeling stages, and the proposed motion classification algorithm are then detailed.

3.1. A Brief Introduction to SOM

In general, SOM is one of the most prominent artificial neural network models adhering to the unsupervised learning paradigm [20]. It has been employed to solve problems in a wide variety of application domains. For the applications in engineering domain, it was elaborately surveyed and reported in [21]. Generally speaking, the SOM model consists of a number of neural processing elements (PEs) or "codebook". Each of the PE, *i*, which is called "codeword' is assigned an *n*-dimensional weight vector m_i where *n* is the dimension of an input data. During the training stage, the iteration *t* starts with the selection of one input data *p(t)*. *p(t)* is presented to SOM and each codeword determines its activation by means of the distance between *p(t)* and its own weight vector. The codeword with the lowest activation is referred to as the winner, m_c, or the best matching unit (BMU) at the learning iteration *t*, i.e.:

$$m_c(t) = min_i \quad p(t) - m_i(t) \quad . \tag{1}$$

The Euclidean distance (ED) is one of the most popular way to measure the distance between *p(t)* and a codeword"s weight vector $m_i(t)$. It is defined by the following equation:

$$d(p(t), m_i(t)) = \sqrt{(p(t)_1 - m_i(t)_1)^2 + (p(t)_2 - m_i(t)_2)^2 + \ldots + (p(t)_n - m_i(t)_n)^2} \tag{2}$$

Finally, the weight vector of the winner codeword as well as the weight vectors of selected codewords in the vicinity of the winner are adapted. This adaptation is implemented as a gradual reduction of the component-wise difference between the input data and weight vector of the codeword, i.e.:

$$m_i(t+1) = m_i(t) + a(t) \quad h_{ci}(t) \quad [p(t) - m_i(t)]. \tag{3}$$

Geometrically speaking, the weight vector of codewords of the adapted units are moved a bit towards the input data. The amount of weight vector movement is guided by a learning rate, α , decreasing with time. The number of codewords that are affected by this adaptation is determined by a neighborhood function, h_{ci} which also decreases with time. This movement makes the distance between these codewords decrease and, thus, the weight vector of the codewords become more similar to the input data. The respective codeword is more likely to be a winner at future presentations of this input data. The consequence of adapting not only the winner alone but also a number of codewords in the neighborhood of the winner leads to a spatial clustering of similar input patterns in neighboring parts of the SOM.

http://dx.doi.org/10.5772/51002

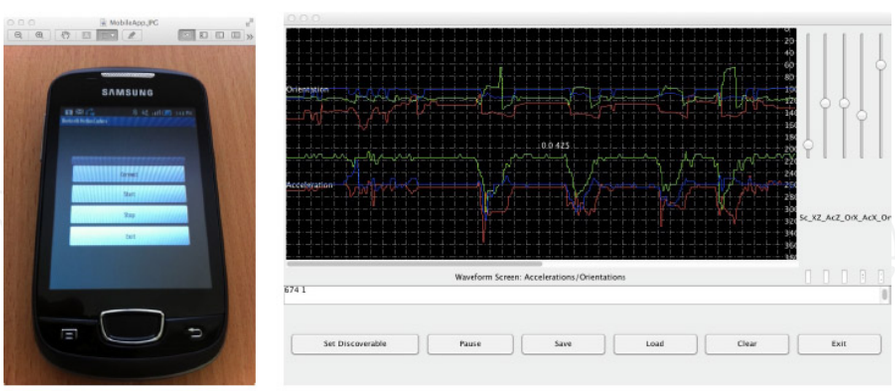
Thus, similarities between input data that are presented in the n-dimensional input space are mirrored within the two-dimensional output space of SOM or SOM map. The training stage is terminated after the final SOM map is labelled with some known conditions.

The classification stage is very similar to the learning stage with some exceptions. That is to say there is no need to perform adaptation to the winner codeword and its neighbours of the SOM map with respect to the input data. Instead, the label of the winner codeword corresponding to the input data is returned and used for further interpretation; i.e. if the input data is mapped to the codeword with jogging motion type label.

3.2 The Motion Data Capturing and Preparation Procedures

Our proposed algorithm relies on using motion data which is captured from different basic daily activity types for training SOM. To efficiently and economically get such data, an application was developed to be executed on a smartphone instead of relying on an embedded data acquisition with a built-in acceleration sensor. A smartphone was targeted from the point of view that it provides all the necessary hardware to serve our purposes. A smartphone with an Android operating system was selected as it is an open platform device 22. The platform, in general, provides support for interfacing with a built-in triaxial acceleration sensor via Application Program Interfaces (APIs). With respect to the developed application, the following roles are performed during the data capturing stage:

- Capture motion data from a triaxial acceleration sensor in as fixed a sampling period manner as possible,
- Wirelessly transmit the captured motion data to the computer server via Bluetooth.



(a) (b)

Figure 1. (a) The main screen of the application running on the mobile phone and (b) the screen captured of a personal computer application.

The Mobile Sensor Actuator Link (MSAL) API was employed to implement the applications running both on a smartphone and a personal computer with the required functionalities. The API is an open source and freely available from Ericsson Laboratory 23. Figure 1 shows the main screen of the application running on the smartphone and the screen captured of a

personal computer application. It is noted that the waveforms shown on the application screen in this case are three parameters along the *x, y* and *z*-axes from the acceleration sensors and the gyroscope. Only the parameters from the acceleration sensor are used in this experiment.

In the course of our experimentations, subjects were requested to perform the following motion types, which cover their basic daily activity types, while the smartphone with the installed application previously detailed was attached to their waist: {jogging (0), normal walk (1), walk upstairs and downstairs (1), stand-sit-stand (2), fall on the floor with support of bed (3), different types of fall on the arm chair (4), fast walk (5)}. The numbers in the parentheses represent the annotations of the motion types on the codebook resulted from the SOM labeling stage (to be described in Section 3.3. The first four and the last motion types were continuously captured for one minute and the fifth and sixth ones were requested to repeat 10 times each. The captured data for each motion type was saved by the server side application to a separate file. The captured motion data files for the first four and the last motion types required less preprocessing. This comes from the fact that these motion types could be performed almost continuously by subjects. As a result, the motion waveforms that represent these motion types have nearly continuous form. These data files were only required to convert from the raw data (separated a_x, a_y, and a_z) to be in the vector summed acceleration a form by applying Eq. 4. Doing this way created the data which are independent of the attachment orientation of the smartphone during data collection stage. Then, the time stamp t was attached to the vector summed acceleration data point by point in order to form the (a_i, t_i)-array. Figure 2 shows the motion waveforms in the vector summed acceleration form for 30 second periods of all captured motion types.

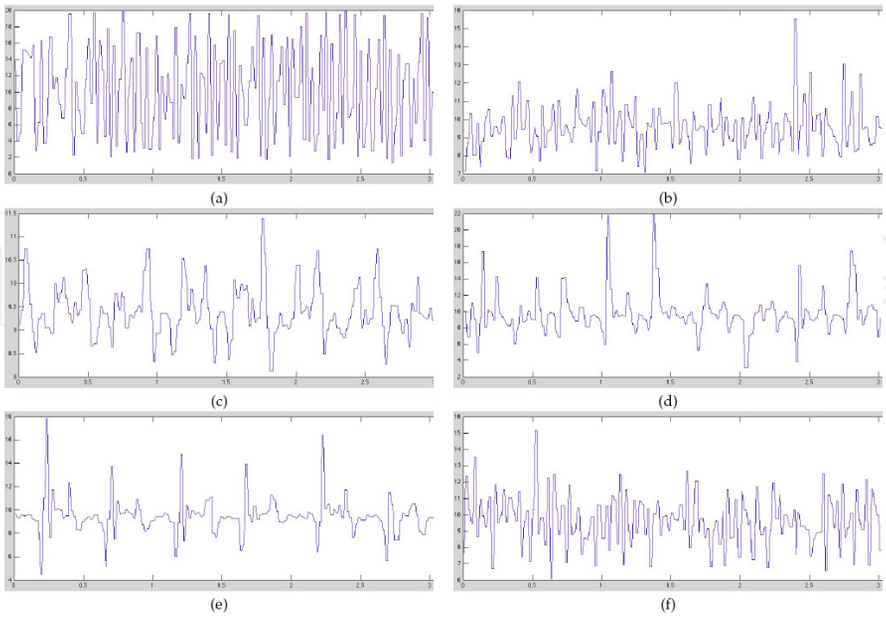


Figure 2. The motion waveforms in the vector summed acceleration form for 30 second periods of (a) jogging, (b) normal walk, (c) sit-stand-sit, (d) fall, (e) fall on the arm chair and (f) fast walk.

http://dx.doi.org/10.5772/51002

In contrast to the first group of captured data described earlier, the captured data files for the fifth and sixth motion types, which are the fall on the floor with support of bed and different types of fall on the arm chair, were required to preprocess. This was done in order to extract only motion waveforms, whose forms are (a_i, t_i), which were relevant to the falls.

$$a = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{4}$$

Followings are our findings from several experimentations during the training stage:

- the preprocessing stage for the motion types of "fall on the floor with support of bed" and "different types of fall on the arm chair" can be eliminated and replaced by using multiple set of training data from these two motion types.
- the combinations of following 5 motion parameters give rise to the highest degree of motion classification correctness: the origin of a segment (a_i), the endpoint of a segment (a_{i+1}), the endpoint of the adjacent segment (a_{i+2}), the slope of a segment, and the length of a segment. This can be represented by:

$$\left(a_i,\ a_{i+1},\ a_{i+2},\ \frac{(a_{i+1} - a_i)}{(t_{i+1} - t_i)},\ (a_{i+1} - a_i)\right) \tag{5}$$

All of the preprocessed data files were converted, from the vector summed acceleration and time stamp format, to be in the form defined by Eq. 5 on a segment by segment basis. At this point, the procedures taken to create the training data set ready for the training stage can be summarized as follows:

- Apply Eq. 4 to all elements of an array of raw acceleration data (a_x, a_y, a_z) in order to convert to an array of vector summed acceleration along with the time stamp format: (a_i, t_i). The process is applied to all arrays of raw acceleration data and gives rise to a set of T_m arrays where m is a motion type index.
- Create an array of $(S_j, S_{j+1}, S_{j+2}, O_j, m_j)$ by applying Eq. 5 to members at i, $i+1$, and $i+2$ of the T_m arrays. In addition, the motion type index, m, is added to all members of the T_m arrays.
- Merge all T_m arrays together by taking all data from the arrays whose motion types are jogging, normal walk, walk upstairs and downstairs, stand-sit-stand, and fast walk and using 3 repeats of fall on the floor with support of bed and different types of fall on the arm chair. All members of the merged arrays are then randomly rearranged. At this point, the training data with random mixture of motion data from all motion types, T, is obtained.

3.3. The SOM Training and Labeling Stages

In order to train and label SOM with the already prepared training data, the algorithms are implemented in the Matlab scripts. We keep the implementations to be as simple as possible as the algorithms will finally be exploited on the smartphone platform. Our training algorithm behaves by:

- single visiting to a member of the training data, T, of size k,
- using a simple learning function with linearly decayed learning rate. The learning rate is allowed to change 256 times with a linear decay rate of $\frac{1}{256}$. This is equivalent to changing the learning rate when $\frac{k}{256}$ members of training data is visited.
- using a simple neighborhood function of the form:

$$\beta(r, c) = a - \left(\sqrt{(r - r_{BMU})^2 + (c - c_{BMU})^2}\right)\frac{a}{2} \tag{6}$$

where $\beta(r, c)$ is the learning coefficient of the codeword at (r, c). α is the learning coefficient of the best matching unit (BMU) codeword whose index is at (r_{BMU}, c_{BMU}). Only the positive value of $\beta(r, c)$ is used to update the weight of the codeword.

For the labeling stage of the algorithm, the training data T is once again presented to the codebook resulted from the training stage. The BMU, whose index is at (r_{BMU}, c_{BMU}), is searched for a given input segment S_k which is a member of T. The matching frequency of the codeword at (r_{BMU}, c_{BMU}) is then incremented. Upon all S_k which is a member of T has been visited, the algorithm labels a codeword with the motion type whose frequency is the highest. It is noted that the matching frequency of a codeword is later used to make decision for the motion type of an unknown motion type segment. That is to say if an unknown motion type segment is found to match to the codeword at (r_m, c_m), its motion type is the motion type of the codeword at (r_m, c_m).

Figure 3 illustrates a codebook along with the projection maps on the training data planes for a configuration of 5×7 codewords. It can be clearly observed that the codewords are clustered very well. The separations between the clusters of slow transition motion types; normal walk (1), walk upstairs and downstairs (1), stand-sit-stand (2), and fast transition motion types; jogging (0), fall on the floor with support of bed (3), different types of fall on the arm chair (4), fast walk (5), are clearly noticeable. It, however, shows some imperfect clusters with mixed fast transition motion types; i.e. the cluster at the bottom left of the map which consists of these motion types: jogging (0), fall on the floor with support of bed (3), different types of fall on the arm chair (4) and fast walk (5). These can be interpreted that the fast transition motion types have some common motion parameters. This is confirmed by a section of comparative acceleration waveforms of the fast and slow transition motion types in Figure 4. At this point, it can be summarized that the codebook cannot be employed alone to the problem of motion classification. In the next section, we detail our proposed algorithm that makes use of the queried results from the SOM to improve the correctness of motion classification and to distinguish between each motion type in the fast transition group. The algorithm is also further designed to support detection of fall.

http://dx.doi.org/10.5772/51002

4. Our Proposed Algorithms

Our algorithm is proposed to compensate for the imperfect clustering of codebook mentioned earlier. It is illustrated in Algorithm 1. There are two inputs to the algorithm which are the codebook C and an unknown motion type segment S_t; i.e. the segment to be queried for its motion type with the form defined by Eq. 5. The codebook C is the one obtained from the training and labeling stages and assigned to the algorithm only once. It can, however, be changed to be specific to the motion conditions of users in practical usage. For a motion segment S_t, the algorithm finds the motion type, *segmentMotionType*, by querying the codebook. The set of query result is as follows: *{typeJogging, typeWalk, typeSitStandSit, typeFall, typeFallOnArmChair, typeFastWalk}*. Instead of using the query result right away which is very prone to error due to the imperfect clustering, the algorithm only registers the current segment motion type. This is performed by incrementing the counter at the corresponding index of the motion type in the *motionFrequency*-array. The algorithm keeps performing in this way for the new incoming segments without making decision for a period of *samplePeriod*. At the end of the Periodg, the motion type can be concluded that it is the one whose the occurrence frequency, or the value of the counter, within the *motionFrequency*-array is the highest. With respect to the algorithm, these processes are performed within the first if-condition. Upon finishing decision making of the motion type, the algorithm clears up the *motionFrequency*-array to ready it for the next period.

The algorithm described so far can be used to make decision between the motion types in the fast and slow transition groups. It, however, fails to distinguish between a jogging and a fall. The failure comes from the fact that these two motion types have fairly similar segment characteristics. These are confirmed by the codebooks (see Figure 3 in the region A, B and C) which show us that codewords of these two motion types seem to be close neighbors of each other. The algorithm is, therefore, incorporated with additional features to differentiate between these two motion types. Let's observe the second if-condition in Algorithm 1. First of all, the algorithm checks whether or not the segment under consideration is a falling edge whose slope is negative. If it is, the algorithm further uses the codebook query result, *segmentMotionType*, once again to check whether or not the segment is in the type of fall. Both testing results are used in combination with the current determination of the motion type result obtained from the first section of the algorithm. If the determination of the motion type result is of type jogging, it is likely that the segment under consideration is also of type jogging. Otherwise, the segment of type fall is detected alone and it is likely that a true fall is likely to occur under an additional condition that the endpoint of the segment, a_t, is less than a threshold.

In the next section, the experimental results after applying the proposed algorithm to the continuously captured motion data are presented.

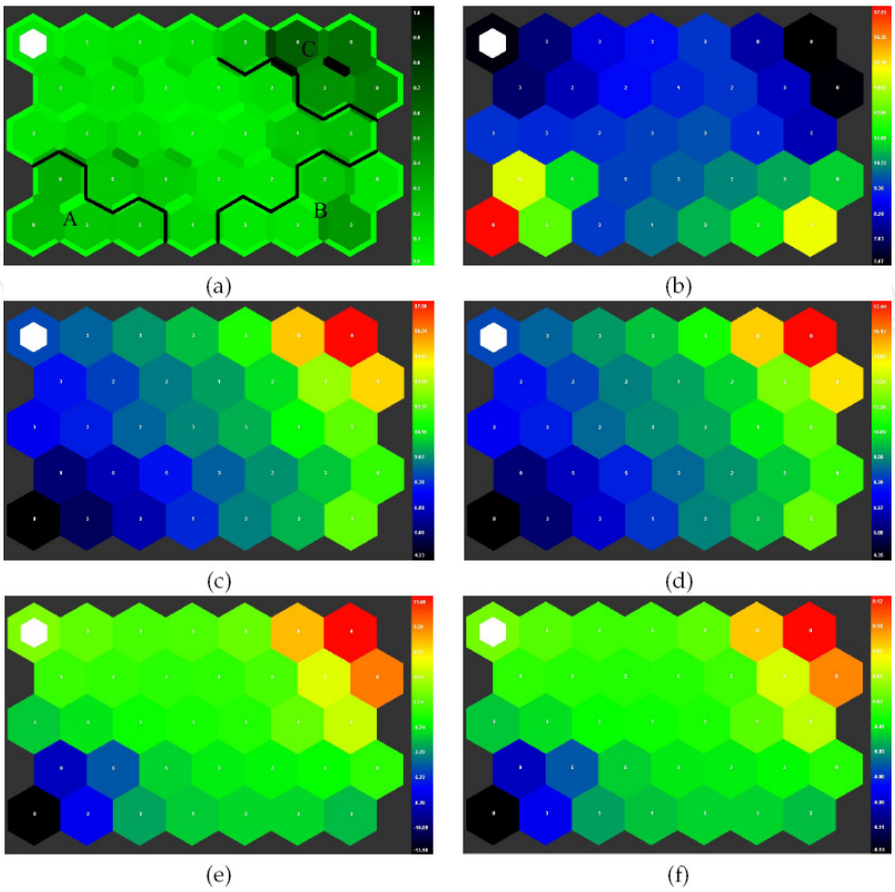


Figure 3. The codebook for a configuration of *5×7* codewords: (a) the original map, (b) - (f) the map projections on the origin of a segment (a_i), the endpoint of a segment (a_{i+1}), the endpoint of the adjacent segment (a_{i+2}), the slope of a segment, and the length of a segment, respectively. It is noted that the parameter value of a codeword is represented by a color whose value is shown on the right hand side bar.

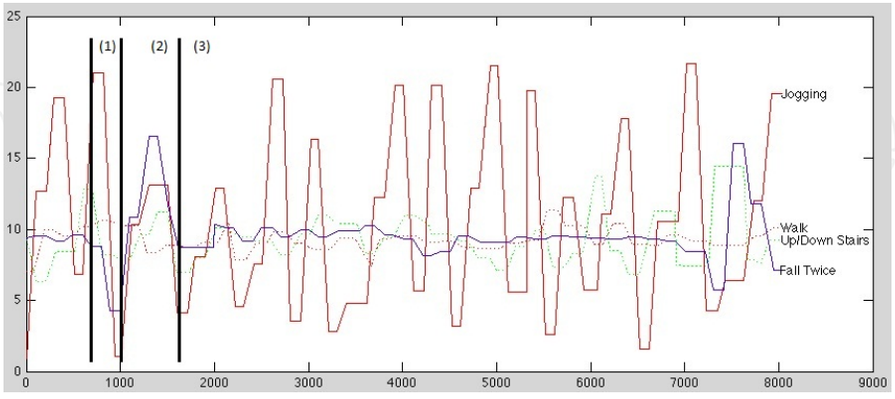


Figure 4. A section of comparative acceleration waveforms of the fast and slow transition motion types

http://dx.doi.org/10.5772/51002

5. Experimental Results and Discussions

During the course of our experimentations, the proposed algorithm was coded in Matlab script and verified its correctness with two sets of real and continuously captured motion data. The verification was performed on a personal computer. We avoided verifying the algorithm on a separate motion type waveform and reporting the false positive (FP) and fast negative (FN) results. It is true that this approach is commonly used in the previous publications. From our point of view, it makes more sense to perform verifications on the continuous motion data as it ensures that the proposed algorithm can be practically used. In addition, the weakness of almost all proposed algorithms in this application domain is that they do not point out clearly how to distinguish between fall and jogging or fast walk. They only focus on finding the representative, average and standard deviation, threshold slope of falls, from different people with different types of fall, in order to make a correct decision of fall. Our verification approach, however, has a weakness as it lacks a standard motion data to benchmark the proposed algorithm. This left us no choice apart from employing our captured motion data. Followings are the details of our motion data sets.

input : A codebook: C
input : a motion segment: $S_t = \{a_{t-1}, a_t, a_t, \frac{(a_t - a_{t-1})}{T_t - T_{t-1}}, (a_t - a_{t-1})\}$
output: A classification result of 'Yes', if fall is detected.
/* Query SOM for the motion type of S_t segment. The query result is one of the following set: {typeJogging, typeWalk, typeSitStandSit, typeFall, typeFallOnArmChair, typeFastWalk} */
$segmentMotionType$ = SOMQuery(C, S_t);
$motionFrequency(segmentMotionType) = motionFrequency(segmentMotionType) + 1$;
/* Check if a period is longer than the duration to make decision of motion type. */
if $stopWatch > samplePeriod$ **then**
  $stopWatch = 0$;
  /* Motion type is the highest frequency index in the motionFrequency-array. */
  $motionType = arg(max(motionFrequency))$;
  /* Reset $motionFrequency$ */
  $motionFrequency(0..end) = 0$;
end
/* Check if this is a falling edge segment. */
if $a_t - a_{t-1} < 0$ **then**
  if $(segmentMotionType == typeFall)$ *or* $(segmentMotionType == typeFastWalk)$ **then**
    /* Free-falling segment type is detected if the following case is true. */
    if $motionType = typeJogging$ **then**
      if $a_t < Threshold$ **then**
        return (Yes)
      else
        return (No)
      end
    else
      return (No)
    end
  end
end

Algorithm 1: A fall detection algorithm employing SOM and motion segments transition: Rising edge.

- Set 1: 78 seconds of normal walk, 41 seconds of fast walk, 37.5 seconds of jogging and 12 repeated falls with 10 seconds duration between a consecutive pair of falls.
- Set 2: Normal walk, fast walk, jogging, normal walk, fast walk, jogging, normal walk, fast walk, pause, fast walk, jogging and 10 repeated falls.

The experimental results with respect to these two sets of motion data are illustrated in Figure 5. In the figure, the green waveforms are the captured motion data in the vector summed acceleration form. The red waveforms are the motion types determined by the algorithm. The blue peaks are the positions of fall detected by the algorithm. The red waveforms and the blue peaks are automatically inserted by the algorithm during operation. The numbers labelled on the left bottom of the figures represent the motion type scale; i.e. the first motion type detected by the algorithm in Figure 5(a) is "1" which is a normal walk. It is noted that we avoid showing the motion type whose number is "0" (jogging) as it overlaps with the x-axis.

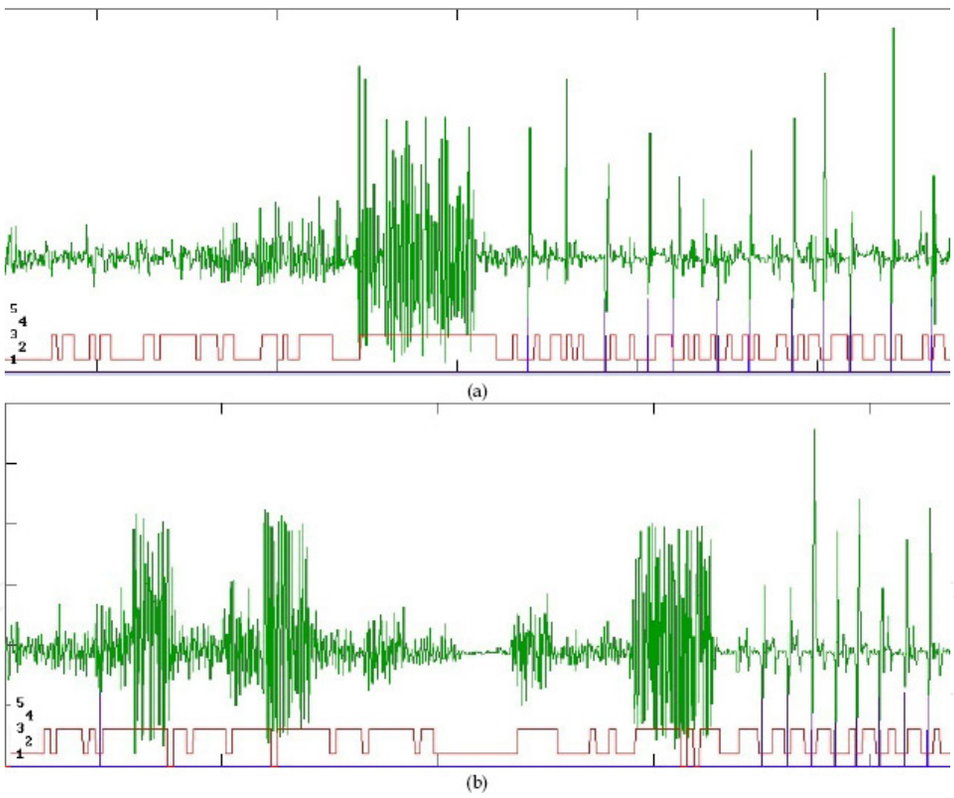


Figure 5. Experimental results after applying the algorithm to two different data: the green waveforms are the captured motion data, the red waveforms are the motion types and the blue peaks are the detected falls.

With respect to Figure 5, it can be seen that the algorithm, in conjunction with the configuration of codebook detailed in Section 3.3, is capable of detecting the changes in motion type. It is capable of detecting all fall curves in both motion data sets. It, however, cannot correctly handle the case of jogging. That is to say the algorithm reports the fast walk motion type

http://dx.doi.org/10.5772/51002

instead of jogging. This is quite obvious in the middle of Figure 5(a) where the jogging motion type is captured. The interesting point of the detection results is found after magnifying the detected parts of fall curves as illustrated in Figure 6. Clearly, it can be seen that the algorithm is able to make decisions if falls occur almost one segment ahead of the first impact. This is equivalent to approximately 133 mS with respect to the data sampling period used during the data capturing stage. It is very useful from the point of view that there exists a system to get a pre-impact alert signal and handle in the way to prevent severe impact. This is an open question for our future research project.

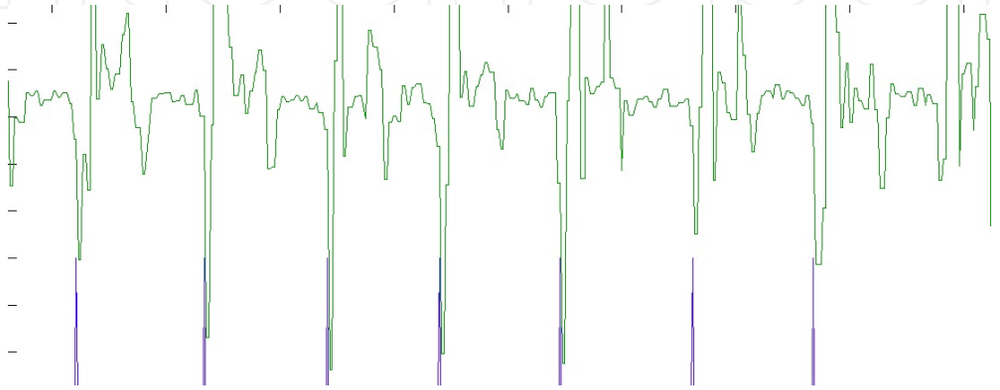


Figure 6. Magnifying the detected fall curves to show that the algorithm can make decision before the first impact.

After verifying the proposed algorithm with different configurations of SOM map, we found a configuration that gave rise to an acceptable correctness in term of motion classification result. This configuration has *8×8* codewords and the data preparation procedure differs from the previously described one in the way that it used only one, instead of three, data from the motion type of fall on the floor with support of bed. Figure 7 shows the classification results with respect to both motion data sets. The similarity between the verification results with respect to both data sets is that the algorithm can no longer detect fall. From Figure 7(a), the algorithm is capable of detection even a short period of normal walk with unusual waveforms which is the first peak of fast walk motion type. The fluctuations between normal and fast walks just a moment before jogging are also detected and correctly reported. In addition, the algorithm gives rise to a correct report during jogging motion type. The results in Figure 7(b) further confirms the correctness. The first two joggings are successfully detected. All motions ahead of the first jogging and in between the first and second and the second and the third joggings are correctly recognized and reported.

At this point, we are trying to give an answer to the question why two configurations of SOM map lead to different detection results. It can be said that the first configuration of training data is designed with the aim to make the abruptly changes of segment, fall segments - to be specific, detectable. In addition, the resulting SOM map must be capable of providing the current motion type to the algorithm. Because the motion type of either jogging or fast walk is used to reject the misjudgment of fall type. This is in contrast to the sec-

ond configuration which is designed to distinguish among the continuous types of motion. In the first configuration, we provide the SOM with the majority of data from multiple sets of fall curves. This results in the SOM map whose majority of codewords are labelled with fall type. On the other hand, the second configuration, the data set from fall motion type is minority and makes the resulting map with only a small number of codewords whose label are of fall type. At the same time, it increases the number of codewords whose label with another motion type. When it comes to querying a segment, the possibility of matching a segment to the codeword with fall type is therefore low.

Another explanation is that when the data from the motion type of fall on the floor with support of bed is minority, it also reduces the number of fall curves within the data. This comes from the fact that the data from the motion type of fall on the floor does not consist of fall curves only. It also has some parts of standing, preceding all fall curves, and sit to stand transition. The sit to stand transition could have similar segments to walking. This results in lowering the number of codewords with fall type and increasing the chance of matching a queried segment to other motion type.

6. The Implementation for Practical Usability

With the positive verification results of the proposed algorithm detailed in the previous section, we then move ahead to implement the algorithm on the suitable platform. The aim of this implementation is to make it applicable in real life. We avoid employing a self-developed embedded system as a platform of choice. The reasons are that it is fairly difficult to design an embedded system with the least eye-catching, less power consumption, small size and high processing power. We have found that a smartphone platform which has been used during the data collection stage of our research is the most appropriate platform as detailed earlier. The algorithm is, therefore, integrated with some parts previously developed which are the acceleration sensor interfacing and background mode sensor sampling. In the prototype, the algorithm is partitioned into two parts: the background mode acceleration sensor query process and the main user interface application with features to support training and labeling stages and system setting. The background mode is programmed to execute with a fixed interval of 100 mS. In reality, this is only the best case sampling period as the operating system could be busy to service the process. However, it does not matter since the algorithm is designed to tolerate for this limitation. Once the acceleration parameters has already been sampled, the process dispatches the parameters to the main user interface application via an inter-process communication. This is done even the main user interface application is forced to operate in the background mode. It is ensured that either the fall detection or motion classification can be performed.

http://dx.doi.org/10.5772/51002

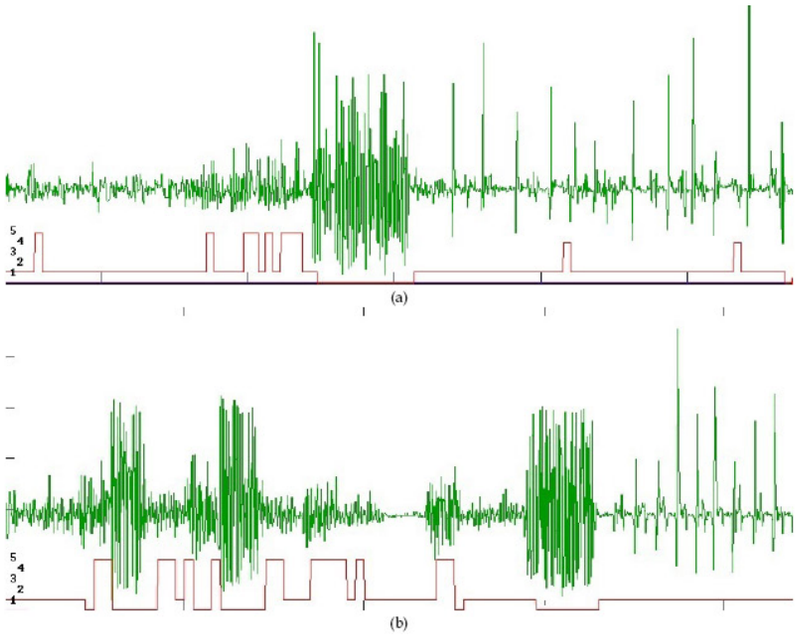


Figure 7. Experimental results after applying the algorithm to two different data: : the green waveforms are the captured motion data, the red waveforms are the motion types. In this case, the codebook is prepared to make the algorithm correctly classify motion types.

Figure 8 shows the screen captures of the main user interface application. In a normal operation mode, the screen of Figure 8(a) is displayed as a main user interface. The waveform shown on the screen is the real-time vector summed acceleration waveform. The button on the lower left is used to provide interaction with a user to change parameters setting (see 8(b)): a phone number to alert in case of fall and some parameters related to decision making of the algorithm. Figure 8(c) is an entry point to the training and labeling stages of the algorithm. It is activated as a result of pressing the "Train Mode"-button of the main user interface screen (Figure 8(a)).

In the training mode screen, the "Start Capturing"-button activates the application to enter data capturing for training and labeling stages. The application captures all motion types in order started from jogging and ended with fast walk with the capturing duration adjustable in the parameters setup screen. The application makes use of the internal data for all fall motion types without requiring a user to perform these risky motion types. When all motion types have already been captured, the application uses its own SOM implementation functions to perform training and labeling stages. The resulting codebook along with the labels can be visualized by use of a built-in feature of the application. Figure 8(d)-(f) shows the screen captures of the sample codebook. The codebook is displayed upon pressing a "Show Map"-button. It is noted that the codebook is useful as it provides a feedback to an operator in order to justify if the training and labeling stages result in a codebook with an acceptable quality. From our testing, we have found that the self training and labeling stages can be avoided and the default codebook can be used in most cases. The testing results for a bigger group of users is now under investigation.

Figure 8. Screen captures of the main user interface application on an Android smartphone.

7. Conclusions

In this chapter, the novel motion classification algorithm in the class of body attachment sensor is proposed. A single triaxial acceleration sensor is employed by the algorithm with a capability to tolerate for a variable sampling rate. In contrast to previously proposed algorithms in the similar category which rely on using threshold techniques, our algorithm Employs the self organizing map neuron network to perform an adaptive motion types clustering and classification functions. The motion data from a wearer is used to train the SOM in order to cluster motion parameters in relation to motion type of the wearer. Later, the motion type is obtained by querying the trained SOM given a motion segment on a real-time basis. With the success of classification of different motion types, we propose extending the algorithm to perform fall detection. In this case, SOM is trained and labeled with a majority of data from fall curves. As the characteristics of the fall curves is almost similar to other motion types with rapid transition of acceleration, the motion type is employed to support the differentiation between fall and other rapid transition motion types. It can be summarized that the contributions of our research are twofolds. Firstly, the motion types classification algorithm employing SOM is proposed and validated for its correctness with respect to the continuous waveform of mixed motion types. Secondly, the motion types is

http://dx.doi.org/10.5772/51002

successfully used to differentiate between fall and other motion types. The conducted experiments indicate that the algorithm gives rise to almost 100% of fall detection correctness with almost zero false for activities of daily living (ADLs). Above all, the algorithm can make decision if fall occurs one segment ahead of the first impact which is equivalent to approximately 133 mS. It is very useful from The point of view that there exists a system to get a pre-impact alert signal and handle immediately in the way to prevent severe impact. To make it applicable, the algorithm is successfully implemented on the smartphone platform based on an Android operating system.

Acknowledgements

This work was supported by National Research Council of Thailand under the project: Design and development of an elderly motion pattern classification and pre-fall alert system. The author would like to thank all anonymous reviewers for their comments in previous versions of this paper.

Author details

Wattanapong Kurdthongmee*

Address all correspondence to: kwattana@wu.ac.th

Walailak University, Thailand

References

[1] Yang, C. C., & Hsu, Y. L. (2010). A Review of Accelerometry-Based Wearable Motion Detectors for Physical Activity Monitoring. *Sensors*, 10(8), 7772-7788.

[2] Kiani, K., Snijders, C. J., & Gelsema, E. S. (1997). Computerized Analysis of Daily Life Motor Activity for Ambulatory Monitoring. *Technol. Health. Care 1997*, 5, 307-318.

[3] Mathie, M. J., Celler, B. G., Lovell, N. H., & Coster, A. C. F. (2004). Classification of Basic Daily Movements Using a Triaxial Accelerometer. *Med. Biol. Eng. Comput.*, 42, 679-687.

[4] Karantonis, D. M., Narayanan, M. R., Mathie, M., Lovell, N. H., & Celler, B. G. (2006). Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring. *IEEE. Trans. Inf. Technol. Biomed.*, 10, 156-167.

[5] Najafi, B., Aminian,, K, Loew,, F, Blanc,, Y, & Robert, P. A. (2002). Measurement of Stand-Sit and Sit-Stand Transitions Using a Miniature Gyroscope and Its Application

in Fall Risk Evaluation in the Elderly. *IEEE Trans. on Biomedical Engineering*, 49(8), 843-851.

[6] Yang, C. C., & Hsu, Y., L. (2009). Development of a Wearable Motion Detector for Telemonitoring and Real-Time Identification of Physical Activity. *Telemed. J. E. Health*, 15, 62-72.

[7] Veltink, P. H., Bussmann, B. J., de Vries, W., Martens, W. L., & van Lummel, R. C. (1996). Detection of Static and Dynamic Activities Using Uniaxial Accelerometers. *IEEE. Trans. Rehabil. Eng.*, 4, 375-385.

[8] Foerster, F., Smeja, M., & Fahrenberg,, J. (1999). Detection of Posture and Motion by Accelerometry: A Validation Study in Ambulatory Monitoring. *Comput. Human. Behav.*, 15, 571-583.

[9] Lyons, G. M., Culhane K., M., Hilton, D., Grace, P. A., & Lyons,, D. (2005). A Description of an Accelerometer-Based Mobility Monitoring Technique. *Med. Eng. Phys.*, 27, 497-504.

[10] Ohtaki, Y., Susumago, M., Suzuki, A., Sagawa, K., Nagatomi, R., & Inooka,, H. (2005). Automatic Classification of Ambulatory Movements and Evaluation of Energy Consumptions Utilizing Accelerometers and a Barometer. *Microsyst. Technol.*, 11, 1034-1040.

[11] Sekine, M., Tamura, T., Togawa, T., & Fukui, Y. (2000). Classification of Waist-Acceleration Signals in a Continuous Walking Record. *Med. Eng. Phys.*, 22, 285-291.

[12] Bussmann, H. B., Reuvekamp, P. J., Veltink, P. H., Martens, W. L., & Stam, H. J. (1998). Validity and Reliability of Measurements Obtained with an "Activity Monitor" in People with and without a Transtibial Amputation. *Phys. Ther.*, 78, 989-998.

[13] Lau, H., Y., Tong, K. Y., & Zhu, H. (2009). Support Vector Machine for Classification of Walking Conditions of Persons after Stroke with Dropped Foot. *Hum. Mov. Sci.*, 28, 504-514.

[14] Zhang, T., Wang, J., Xu, L., & Liu, P. (2006). Fall Detection by Wearable Sensor and One-Class SVM Algorithm. *Intel. Comput. Signal Process. Pattern Recognit.*, 345, 858-863.

[15] Huynh, T., & Schiele, B. (2006). Towards Less Supervision in Activity Recognition from Wearable Sensors. *Proceedings of the 10th IEEE International Symposium on Wearable Computers, Montreux, Switzerland*, 11-14 October 2006, 3-10.

[16] Long, X., Yin, B., & Aarts, R. M. (2009). Single-Accelerometer-Based Daily Physical Activity Classification. *Proceedings of the 31st Annual International Conference of the IEEE EMBS, Minneapolis, MN, USA*, 2-6 September 2009, 6107-6110.

[17] Allen, F. R., Ambikairajah, E., Lovell, N. H., & Celler, B. G. (2006). Classification of a Known Sequence of Motions and Postures from Accelerometry Data Using Adapted Gaussian Mixture Models. *Physiol. Meas.*, 27, 935-951.

http://dx.doi.org/10.5772/51002

[18] Mannini, A., & Sabatini, A. M. (2010). Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers. *Sensors*, 10, 1154-1175.

[19] Pober, D. M., Staudenmayer, J., Raphael, C., & Freedson, P. S. (2006). Development of Novel Techniques to Classify Physical Activity Mode Using Accelerometers. *Med. Sci. Sports Exerc.*, 38, 1626-1634.

[20] Kohonen, T. (1990). The Self-Organizing Map. *Proc. of IEEE*, 78(9), 1990, 1464-1480.

[21] Kohonen,, T., Oja, E., Simula, O., Visa, A., & Kangas, J. (2002). Engineering applications of the self-organizing map. *Proc. of IEEE*, 84(10), 2002, 1358-1384.

[22] Android Developer. (2011). Android Developer Guide, Available from:, http://developer.android.com/index.html, 21-Dec-2011.

[23] Ericsson Labs. (2011). Mobile Sensor Actuator Link. https://labs.ericsson.com/developer-community/blog/mobile-sensor-actuator-link, 21-Dec-2011.

IntechOpen