A DEEP-LEARNING-BASED FALL-DETECTION SYSTEM TO SUPPORT AGING-IN-PLACE

A Thesis

by

HEND ALKITTAWI

BS, University of Jordan, Jordan, 2013

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

May 2017

A DEEP-LEARNING-BASED FALL-DETECTION SYSTEM TO SUPPORT AGING-IN-PLACE

A Thesis

by

HEND ALKITTAWI

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Maryam Rahnemoonfar, PhD
Chair

Ahmed Mahdy, PhD
Committee Member

Elizabeth Sefcik, PhD
Committee Member

May 2017

ABSTRACT

Emergency departments treat around 2.5 million older people for fall injuries each year. Serious head and broken bones injuries occur in 20% of falls. Fall injuries, adjusted for inflation, has direct medical costs of $34 billion a year. Taking into account that people 65 and older are expected to comprise 21.7% of the U.S population in 2040, compared to 14.4% in year 2013, the numbers presented in the statistics will dramatically increase as well.

Preserving the elderlys' right of aging in a home of their own choice is mandatory in today's world, as more elderly people are willing to live independently. But, with the statistics showing that falling is a major health problem that has a huge non-desirable impact on elderly lives, fall detection systems become a necessity.

Different approaches have been used to design fall detection systems. One approach depends on wearable sensors that measure different physical parameters of a human body or the environment around it, such as the body acceleration or its pressure on the floor. A second approach depends on sensors employed in the environment. These sensors mainly include wide-angle cameras, depth cameras, and microphones. Different approaches used different classifiers for training the system to detect falls. Despite these efforts to detect falls, it is possible that other naturally occurring falls trigger false alarms. Thus, the current implementations of fall detection systems need to be improved. Most recently, computer vision based approaches using depth cameras are the mostly used for such improvement.

Using deep neural networks to learn features from video frames have a potential to improve the fall-detection accuracy and reduce triggering false alarms. In this study, a more robust and deep fall-detection system was designed. This approach extends deep convolutional neural networks in time. This extension allows capturing

the spatial and temporal information presented through successive video frames. The result of the new approach can be used to implement a reliable surveillance system in a real-world environment.

ACKNOWLEDGMENTS

I would like to thank the members of my committee: Dr. Maryam Rahnemoonfar, Dr. Ahmed Mahdy, and Dr. Elizabeth Sefcik, for their helpful comments and suggestions. I also would like to thank Dr. Alaa Sheta, Akash Ashapure, and the members of Bina lab and Pixel Island lab specially Clay Sheppard, Vinay Pinnaka, and Dang Huynh for the helpful discussions.

TABLE OF CONTENTS

CONTENTS PAGE

LIST OF TABLES

LIST OF FIGURES

FIGURES PAGE

CHAPTER 1

INTRODUCTION

1.1 Motivation

There is nothing worthier than working on research that has a great impact on people's lives. Preserving the elderly's right of aging in a home of their own choice is a must. But, with the statistics showing that falling is a major health problem of a huge non-desirable impact on elderly lives, the security of elderly becomes a concern.

It has been reported that emergency departments treat 2.8 million older people for fall injuries every year [14]. Serious head and broken bones injuries occur in 20% of falls [14]. Fall injuries, adjusted for inflation, have direct medical costs of $31 billion [14]. Taking into account that people 65 and older are expected to comprise 21.7% of the U.S population in 2040, compared to 14.4% in year 2013 [13], the numbers presented in the statistics will dramatically increase as well. Therefore, fall detection systems become a necessity.

1.2 Goal and Objective

Current implementations of fall detection systems lack accuracy. Despite efforts to detect elderly falls, it is possible that daily life activities, such as lying down, trigger false alarms [47]. The goal of this study is to build a more robust fall-detection system by using deep learning algorithms. This approach uses a 3D convolutional neural network to capture both the spatial information available in video frames, and the temporal information presented through successive video frames.

1.3 Contributions

In general, the research on fall-detection systems focusses on extracting the best features that can correctly identify fall events. In this research we investigate using deep convolutional neural networks to describe the overall space-time appearance pattern of a fall-event. It is one of few approaches that apply deep convolutional neural networks to video frames of a relatively small dataset. Our approach focusses on finding the best representation of video frames that would allow automatic learning of space-time features presented in surveillance video frames. Our contributions in this research are:

1. Conducting an extensive study on data representation in order to get the system to perform well. This includes testing with depth images, and binary images. It also includes considering different number of frames for action representation.

2. Utilizing a 3D convolutional neural network architecture and adjusting it to fit the goal of this research. This includes finding the sizes of the convolutional and pooling filters, in the spatial and temporal domains.

3. Improving fall-detection systems. This has a great influence on lowering the false alarms ratio, and consequently, improves the system's accuracy and real-time response.

4. Extending the system obtained to process an online data-stream. The result of the new approach can be used to implement a reliable surveillance system in a real-world environment.

CHAPTER 2

LITERATURE REVIEW

2.1 Fall Detection Systems

A variety of systems have been developed to detect falls [21][36]. Technologies and algorithms used have changed over time, but the main objective of a fall-detection system remains the same; it needs to distinguish between fall events and the activities of daily life. The metrics used for measuring the system performance are: sensitivity, being able to classify a fall correctly, and specificity, being able to classify daily activity as daily activity. Having a robust system, with a high degree of sensitivity and specificity, is the ultimate goal of the research done in this field.

Fall detection systems can be generally categorized under the following types: context-aware systems, wearable devices, and cellphone based systems [21]. In context-aware systems, sensors are deployed in the environment to detect falls. The most commonly used sensors, in more recent research, are wide angle cameras [5][40], and depth cameras [31][47][2][34]. Installing cameras with other sensors such as microphones [33] and pressure sensors [54] is also a common approach. Radars are another type of sensor that is used in context-aware systems to sense motions based on the wavelet transform [48]. Wearable devices can be defined as miniature electronic sensor-based devices that are worn by the bearer, under, with, or on top of clothing. For these systems accelerometers are mainly used [53][8][32]. Approaches that uses cellphones for fall-detection take advantage of the accelerometers embedded in mobile devices. These approaches also use the built-in communication functionality in cellphones for designing fall-detection systems [11][1][18].

Several advantages and disadvantages are associated with the different systems.

While implementations which use cellphones are portable and self-contained, they face problems associated with the low battery life and the stability of the accelerometer used in a cellphone. In addition, they use the operating system of a cellphone which was not designed for this goal. The algorithms used with accelerometer-based systems are less complex than those used with vision-based systems [21], but having them worn by the user introduces a considerable limitation on such systems.

Context-aware systems in general are superior over wearable devices in terms of reliability as there are no chances of forgetting or losing them. Microphones are an example of context-aware fall-detection systems. The work presented in [33], uses an eight-microphone circular array is used to track a person. Tracking is based on beamforming techniques that allow performing under noisy environment. By determining the position of the sound source, the height of a person can be measured. A sound that comes from the ground represents a potential fall and triggers more processing of the sound signal in order to verify the fall-event. The main disadvantage associated with context-aware systems is that they can only function in the areas where they are installed. In more recent systems, wide angle cameras are used to allow monitoring large areas without having to install several cameras in that location, but the camera lens introduces some distortion that needs to be corrected [5]. The advantage that depth cameras provide is preserving the privacy of people under surveillance, since color images are not used. Occlusion represents a considerable constraint for systems based on computer vision [47].

Fusing cameras with other context-aware sensors is a common approach to minimize the amount of computations performed as the non-camera sensor is used to trigger the processing of video frames, instead of continuously analyzing the surveillance videos. A Pressure sensor is an example of such sensors [54], where the human

pressure on the floor is measured and compared against a threshold. If the pressure value is greater than the threshold value, video frames are processed in order to verify a fall-event. Employing context-aware sensors and wearable sensors in one system in order to overcome the limitations of the independent systems was also proposed. The work presented in [31] uses an accelerometer and a depth camera. The accelerometer continuously measures the acceleration of the wearer, which is compared against a threshold signal. An acceleration that is greater than a threshold triggers video frames extraction and processing so that a fall-event is identified.

The reported accuracy for the current implementations varies. The performance of the systems is highly affected by the conditions under which they were tested: the number and type of falls included in the dataset, the ages of people involved in testing, and the environment conditions under which the systems were tested. Therefore, the actual accuracy might be lower than the declared performance, and the findings of different studies may not be generalized.

The datasets used have real [47] or simulated [31] fall and non-fall events, which is collected and recorded using different sensors in order to process and classify them. To classify the events as falls or non-falls, techniques such as Decision Trees [47], Support Vector Machine [5], Kalman Filtering [40], Thresholding Techniques [31], Nearest-neighbor Rule [31], Gaussian Mixture Model [43], Rule-based Techniques [57], Multi-frame Gaussian Classifier [19], Gaussian Distribution of Clustered Knowledge [59], Bayesian Filtering [41], Hidden Markov Models [10], and Fuzzy Logic [39] are used.

These classification techniques are well known in the area of machine learning. Decision trees are used to model decisions and their consequences as a graph. In the graph, each node represents an attribute test. While branch represents the test

outcome. For a classification problem, the class label is represented by a leaf node. In support vector machines, vectors or hyperplanes are used to classify data with the maximum possible margin separation. The larger this margin is, the lower the generalization error is. Therefore, support vector machine classifiers can achieve high accuracies. Kalman filtering is an estimation technique that uses measurements captured over time to predict a class label. Thresholding techniques are low-complexity algorithms that produces a class label based on a signal exceeding a threshold. The threshold is usually determined heuristically. The nearest neighbor rule produces a class label for a new input using votes from k training examples neighbors. Voting is based on a metric distance, such as the Euclidean distance.

Gaussian mixture models are a class of mixture models where the data points are assumed to be generated from a mixture of a finite number of Gaussian distributions. A new data entry is assigned a label that maximizes the likelihood of the data being under a specific class, based on the parameters learned during the training phase. Hidden Mrakov models represents the system as a Markov process with hidden states. They are a generalization of mixture models where the hidden variables are related through a Markov process. In a rule-based system, features are used as rules sets. The rules are applied to infer a class label for a new data entry. Bayesian filtering uses the bag-of-words representation and the Bayes rule to produce a class label. The idea behind fuzzy logic classification is to define a class membership based on the relative importance of precision.

Overall, the selection of a classification technique often depends on the set of features obtained to map an input to an output. For fall-detection systems that are based on computer vision approaches, these features can be as simple as the ratio between the width and height of the bounding box surrounding a human [43], and

as complicated as the distance of the points in a human point cloud to the floor [31]. Some other features include extracting the patterns of change in human curvature [34], or human silhouette orientation [2] during a fall event. The focus of the research in this area is to design the best features that would correctly identify fall events, with the minimum false alarms ratio.

2.2 Human Action Recognition Using Deep Learning

Deep learning algorithms have been widely used for solving a variety of problems in different domains. Convolutional neural networks are extensively used for image classification [30][20], object detection [51], and scene recognition [60] tasks. Recurrent neural networks are also applied to image classification tasks [35], but they are more associated with sequences processing. Machine translation [7][49][50], and natural language processing [22][52][17] are some of the most common applications of recurrent neural networks. Sequence generation is another interesting application of such networks where machines can generate text [16], code [26], and music [9].

Due to the magnificent performance of these algorithms, human action recognition has received extensive interest by deep learning researchers. More recently, deep learning techniques have been used to recognize actions in videos. These applications are flourishing with the increased processing capabilities available through GPUs, as well as the large video datasets that were made available for the research community. The most commonly used video datasets are UCF-101, HMDB-51, and Sports-1M. These datasets contain RGB videos for a variety of human actions.

Deep learning algorithms allow the system to learn the best features to recognize human actions in videos. One class for human action recognition using deep learning relies on applying convolutional neural networks to process video frames [27][44][23].

The work presented in [27] uses Sports-1M dataset with one million videos representing 487 human action classes. In their work, the authors stack the video frames and feed them to a multi-resolution CNN. The multi-resolution network has two streams for processing; a low-resolution stream and a high-resolution stream. The authors also have investigated multiple approaches for fusing temporal data in the convolutional neural network. The approaches include a single-frame model, early-fusion, late-fusion and slow-fusion. In [44], the authors use two-stream convolutional neural network. One stream is used to capture the spatial information by processing single frames. The other stream is used to capture the temporal information by processing multi-frame optical flow representations. A 3D convolutional neural network was used in [23] to extract the spatial and temporal information encoded in successive video frames. The datasets used to test this 3D architecture are the TRECVID 2008 and the KTH with more than 49-hour videos. Bounding boxes were drawn around humans in the scene to keep track of them. Video frames were fed into the network which consists of two 3D convolutional layers, two pooling layers, one 2D convolutional layer, and one fully connected layer. In most cases, the classification based on the CNN approach achieves good results. For RGB video frames, these results can be due to the association of human actions with the presence of certain objects in the scene. For example, for a human action to be classified as swimming, a water surface needs to be present in the scene.

Combining convolutional neural networks and recurrent neural networks is used for image captioning tasks [56]. As videos consist of a sequence of frames that are related in time, a second class of recognizing human actions, using deep learning, relies on applying a combination of convolutional neural networks, and recurrent neural networks to process video sequences. The work presented in [12] uses this

combination to classify human actions. This combination can also be used to generate textual descriptions for the actions presented in videos [55]. The idea of the work presented in [12] is to learn the temporal information from videos by passing a sequence of spatial features learned by a convolutional neural network to a recurrent neural network. The later network is capable of extracting the temporal features to achieve the video recognition task. Similarly, [55] feed RGB frames to a convolutional neural network and use a sequence to sequence recurrent neural network to describe the action in videos.

The work presented here adapts deep learning techniques used for human action recognition. These techniques are employed for recognizing fall events captured in depth videos. This approach is different from other other fall-detection approaches in that it automatically learns the fall patterns in space and time using a deep 3D convolutional neural network. The following section gives an insight on how deep neural networks can learn those features.

2.3 Theoretical Background

2.3.1 Artificial Neural Networks/ Feedforward Neural Networks

Artificial neural networks, (ANNs), and their variants, are a class of machine learning techniques that have been proven to be powerful throughout many applications. These networks shine in the areas where there are lots of data and complex problems to solve through "Deep Learning". They have been applied to image and video classification, image and video captioning, machine translation, and many other fields.

ANNs were inspired by the neuroscience, thus, the building block of an ANN is called a neuron. A basic neural network is shown in figure 1. It consists of an input layer, one or more hidden layers, and an output layer. Each layer consists of one

or more neuron. Inputs are fed into the neurons that compute some output values based on the weights and biases associated with them [37]. These outputs are also referred to as the activations of neurons. Some functions are used more often to compute such activations. The Sigmoid Function is one of these functions.

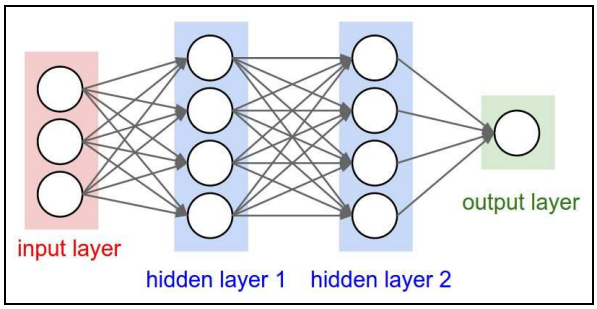

Figure 1. The general structure of an artificial neural network. [46]

The learning algorithms behind ANNs can automatically produce some desired output based on the given input by adjusting the weights and biases along the network. This adjustment aims to minimize the difference between the computed output and the actual output value, and can be automated using the Backpropagation and Gradient Descent algorithms during the training phase. Therefore, these steps are followed in order to train a neural network:

1. Design a network architecture.
2. Randomly initialize weights.
3. Implement forward propagation to compute the output(s).
4. Implement the cost function.
5. Implement back propagation to compute partial derivatives.
6. Use gradient descent and backpropagation to minimize the cost function as a function of the weights and biases.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks, (CNNs), are a subclass of feedforward neural networks that are mainly used for processing images. These networks can learn a hierarchy of features which can be used for image classification [30][20], object detection [51], scene recognition [60], and more recently for video classification [27][44][23]. A CNN consists of several layers as shown in figure 2. Different architectures are built for specific applications using these layers.

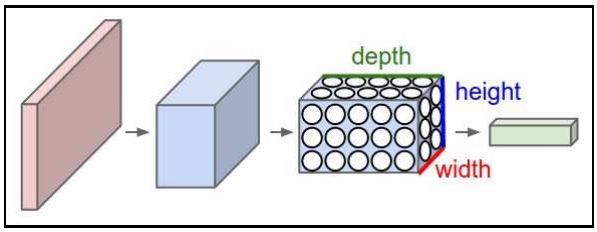

Figure 2. The general architecture of a convolutional neural network. An activation volume is produced after convolving the input image with a set of filters. Subsampling is performed in the pooling layer. The output representing a class prediction is produced after the fully connected layer. [46]

The building block of a CNN is the Convolutional Layer. This layer has neurons connected to local regions in the input image and it is where most of the computations are performed. The convolutional layer represents filters whose parameters are learned by the network. These filters in turn, detect features across the input image by being convolved with images in the dataset. The filters are being slid across the image in order to detect similar features across it. Since these filters are shared across the input image, the parameters of the network are highly reduced and optimized compared to a regular feedforward neural network. At this layer, padding the frames is a common practice to preserve the spatial extent of the input image. The output

of this layer is passed through an activation function such as the Rectified Linear Unit function (RELU), defined in equation 2.1. The result of passing the output through the activation function is an activation volume that extends along the input depth.

$$f(x) = max(0, x) \tag{2.1}$$

The Pooling Layer of the CNN is used to down sample the activation volume that results from the convolutional layer. This has an important role for reducing the amount of computations. Max pooling is the most commonly used approach in CNN compared to average or L2-norm pooling. It uses a window of a small size, 2 or 3, and takes the maximum value of the image in that window, sliding spatially across the entire image to produce its output.

For image classification problems, the Fully Connected Layer is used to compute the scores of classes. The neurons of this layer are connected to all activations in the previous layer. The output of this layer is a vector holding the class prediction for an input image. To turn these predictions into class probabilities, a softmax function is applied.

2.3.3 Recurrent Neural Networks

Recurrent Neural Networks, RNNs, can be used to estimate different functions with different input-output mapping. The general idea behind an RNN cell, is the recurrence application of the same set of functions on a sequence of input vectors at each timestep, which allows the persistence of information in the recurrent network. It can be thought of as a neural network with a series of repeating modules.

RNNs are capable of mapping a single input to a single output. Such mapping

is used in an image classification task [35]. RNNs are also used to map a sequence of inputs to a single output. An application of this mapping would be to determine the sentiment [22], being positive or negative, for a sequence of words i.e. a sentence. Lastly, an RNN can map a sequence of inputs to a sequence of outputs. This is widely used for language translation, where the input is a sequence of text in some language and the output is the corresponding translation in another language [50][49]. These mappings are shown in figure 3.

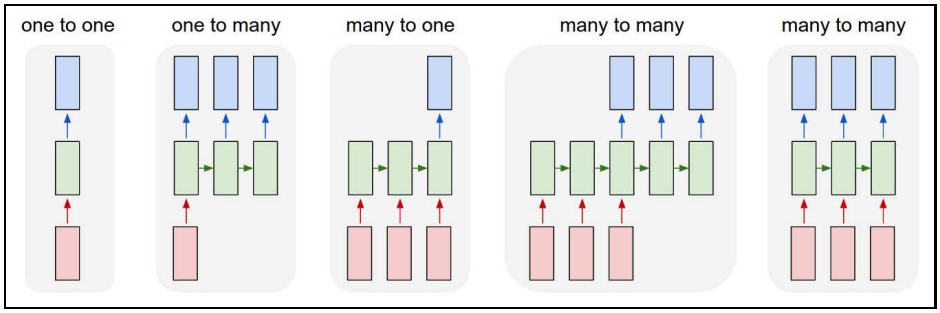

Figure 3. The different input-output mapping used in a recurrent neural network.[25]

There are different variations of recurrent neural networks including the Vanilla Recurrent Neural Networks [4], Long Short-Term Memory Cells [4], Gated Recurrent Units [4], Depth Gated Recurrent Neural Network [58], and Clockwork Neural Network [29]. The details of the most dominant variations are listed below.

2.3.3.1 Vanilla RNN

Vanilla RNN is the basic version of RNN, and is shown in figure 4. It calculates the hidden state based on some function f with parameters W. The recurrence function, most dominantly a *tanh* function, uses the current cell input and the hidden state in a previous time step to calculate the current hidden state. This hidden state, in

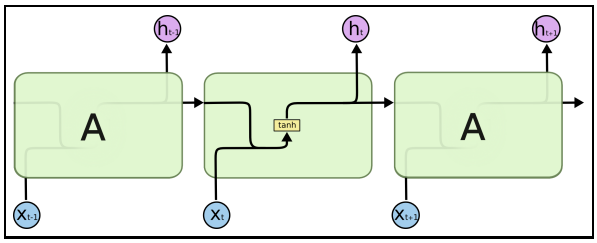

Figure 4. The Vanilla Recurrent Neural Network. [38]

turn, is used to calculate the output at that specific time step. This process is shown in equations 2.2 to 2.4.

$$h_t = f_W(h_{t-1}, x_t) \tag{2.2}$$

$$h_t = tanh(W_{hh}.h_{t-1} + W_{xh}.x_t) \tag{2.3}$$

$$y_t = W_{hy}.h_t \tag{2.4}$$

Vanilla RNN perform well for short sequences, where the time difference between the relevant information and the place where they are needed is small. For long dependences, such networks suffer from vanishing/exploding gradients. Gradients can rapidly reach a value of zero for sequences of only length 10 or 20 [4]. This occurs due to long backpropagation iterations, and prevents the network from properly learning how to model the problem.

2.3.3.2 Long Short-Term Memory Cells

To solve the problem of vanishing/exploding gradients, Long Short-Term Memory architecture was proposed. An LSTM cell is capable of learning long- term dependencies. In addition to the hidden state, an LSTM maintains a cell state, which is adjusted by structures called gates. The gate simply represents an application of a

sigmoid function on some input. An LSTM cell maintains three kinds of gates: the input gate, the forget gate, and the output gate. The architecture of an LSTM cell is shown in figure 5.

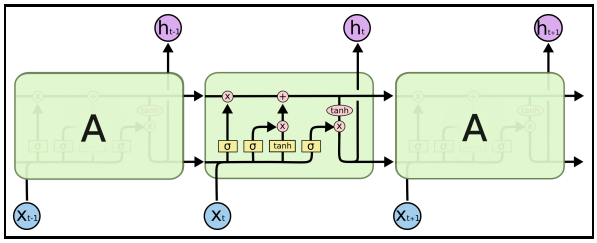

Figure 5. The Long Short-Term memory cell. [38]

The input gate chooses what new information needs to be stored in the cell state, this is shown in equation 2.4 and 2.5, where i_t is the input gate layer output and c_t is the cell state update. The forget gate decides what existing information in cell state needs to be thrown away, this is shown in equation 2.6, where c_t is again the update of the cell state. Finally, the output gate filters the output and determines the final cell output. This can be seen through equations 2.8 and 2.9, where o_t is the output-gate layer output and h_t is the resulting hidden state for the given input.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.5}$$

$$c_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.6}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.7}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.8}$$

$$h_t = o_t.tanh(c_t) \tag{2.9}$$

There are very interesting applications that use LSTMs; text [16], code [26], and music generation [9] to mention a few.

2.3.3.3 Gated Recurrent Units

This is a simpler model which is based on LSTMs, and has been used quite frequently in literature [58][52]. It builds an update gate using the input and forget gates. It also fuses the hidden and cell states.

CHAPTER 3

METHODOLOGY AND SYSTEM DESIGN

Due to the conditions under which fall-detection systems are to be employed, depth videos are preferred over RGB videos. Depth video streaming cuts down the elderly's concerns about their privacy. In addition, the use of IR sensors to capture these videos makes the system more robust to lighting condition changes and allows it to operate under low light conditions [2][34][47]. This study uses the SDUFall dataset[1] which contains 6 actions performed by twenty young people. These action are: bending, falling down, lying, squatting, sitting and walking. Men and women participants performed each action multiple times. The conditions under which these actions were captured are different based on the lighting conditions, and the direction and position of the stunt relative to the camera. The Microsoft Kinect sensor was installed at 1.5m height. The videos were recorded at 30 frames per second, with a 640×480 frame size. On average, the length of a video is 5.6 seconds.

There are a few approaches for human action recognition using deep learning algorithms. The choice of the deep neural network architecture for this research was governed by many factors. First, the temporal information that is available in videos gives a clue that cannot be ignored for fall-detection. While some approaches use multiple 2D convolutional neural networks to process the spatial and temporal data presented in videos, we are unable to feeds videos to such networks due to memory limitations. Other approaches feed sequences of features extracted using 2D convolutional neural networks to LSTM cells. Besides the fact that having multiple networks is computationally expensive, LSTMs are capable of capturing long tem-

[1] http://www.sucro.org/homepage/wanghaibo/SDUFall.html

poral dependencies using feedback loops along with the feedforward loops. These dependencies can be for sequences of hundreds of time-steps [4]. During a fall-event, video frames have short term temporal dependencies. Therefore, we made the choice of using 3D convolutional neural networks, taking into account that convolutional neural networks allow capturing short temporal dependencies, if extended in time.

The architecture used extends convolutional neural networks in time, to allow capturing both temporal and spatial information presented in successive video frames. The network architecture for the final implementation is shown in figure 6.

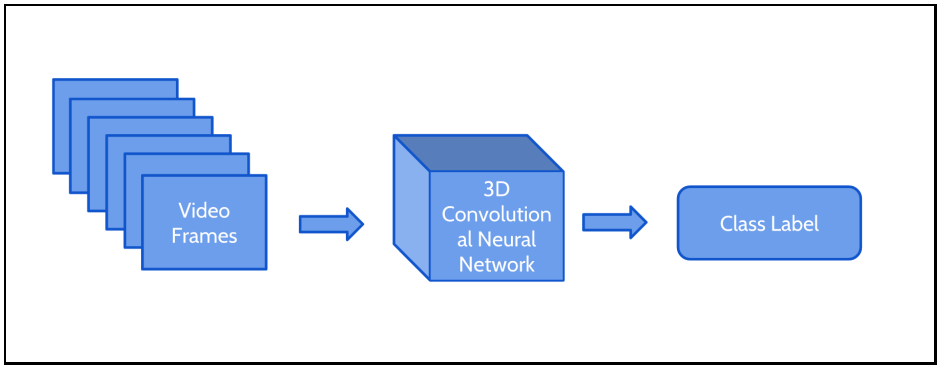

Figure 6. The model architecture.

The algorithms used by deep neural networks allow learning the parameters of the functions that best classify the data. This minimizes the need for hand-crafted features in order to classify data, but introduces a new challenge for machine learning engineers. That is, we need to find the best data representation that would allow the network to learn those features. The following section shows more details on data preprocessing and preparation, that cumulatively gave insight into the optimal data representation for our goal.

3.1 Data preprocessing and data representation

The dataset's video frames have 640×480 spatial resolution. All frames were resized to 160×120 pixels. By means of visual interpretation, we could determine that the size reduction lowered the amount of processing required, without loosing the spatial information. A sample of frames for the different actions is shown in figure 7.

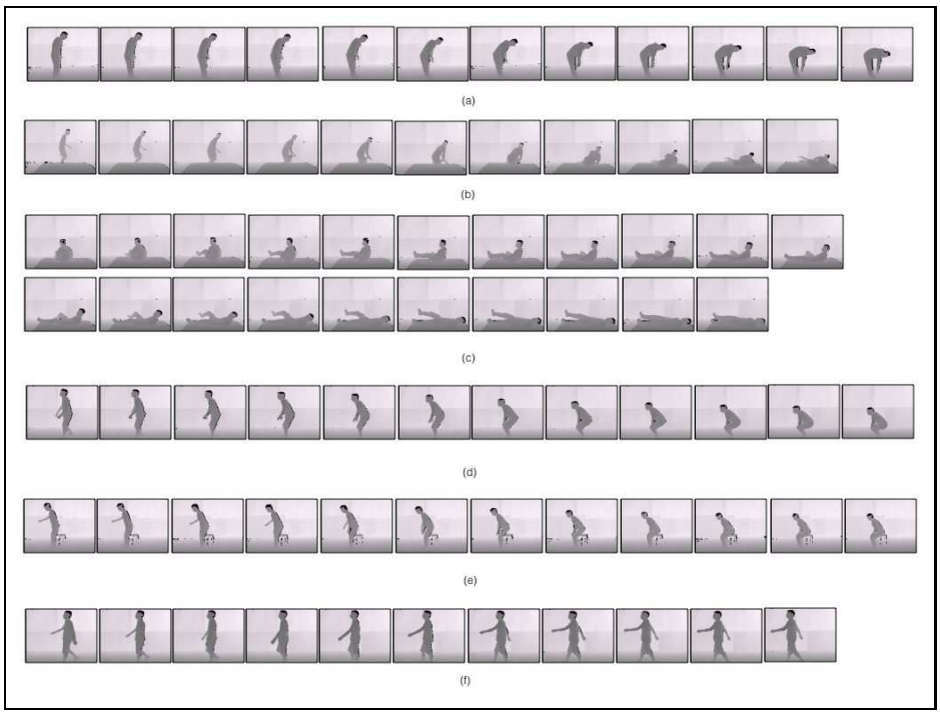

Figure 7. Sample video frames. (a) Bending frames. (b) Falling frames. (c) Lying frames. (d) Squatting frames. (e) Sitting frames. (f) Walking frames.

The videos in the original dataset are of different lengths. The number of frames in a video ranges between 69 and 502. The histogram in figure 8 shows the distribution of the number of frames in each video.

To be able to feed batches of videos to the 3D convolutional neural network, the dimensions of the input needs to be consistent. That is, all inputs must be of

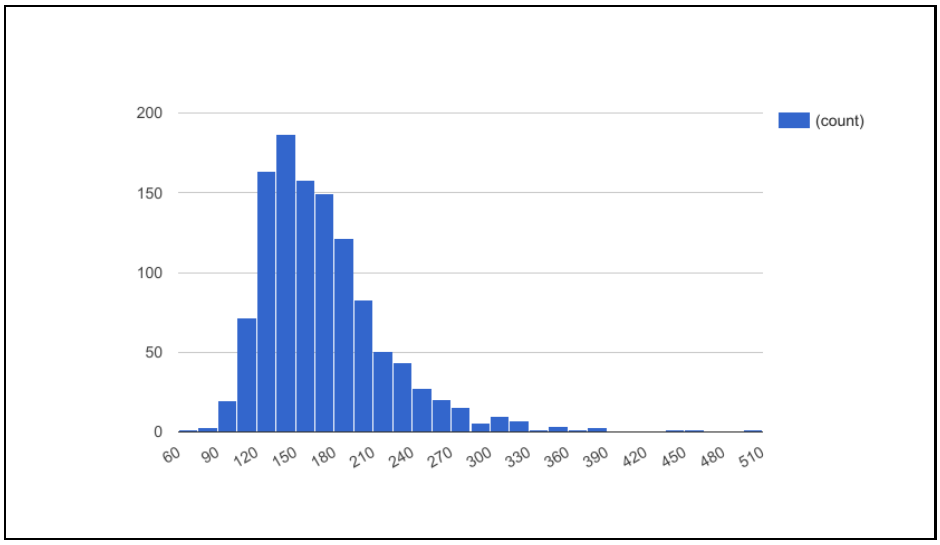

Figure 8. Video Length Distribution

the same spatial and temporal size. For the spatial domain, all video frames are of the same size and there are no inconsistencies. But for the temporal domain, videos are of different lengths. In order to achieve consistency in the temporal domain, two approaches to prepare the dataset were considered. One approach is based on creating the depth map motion, the other is based on using fixed number of video frames. The details for these approaches are in the following subsections.

3.1.1 Depth map motion

This approach generates the depth map motion similar to the work presented in [6]. That is, the absolute difference between two consecutive frames is projected and accumulated through the video sequence as shown in equation 3.1, where N is the total number of frames in the video. This would result in one image that captures the action sequence.

$$Depth\ Map\ Motion = \sum_{i=2}^{i=N} \mid frame_i - frame_{i-1} \mid \tag{3.1}$$

A sample of the results obtained is shown in figure 9.

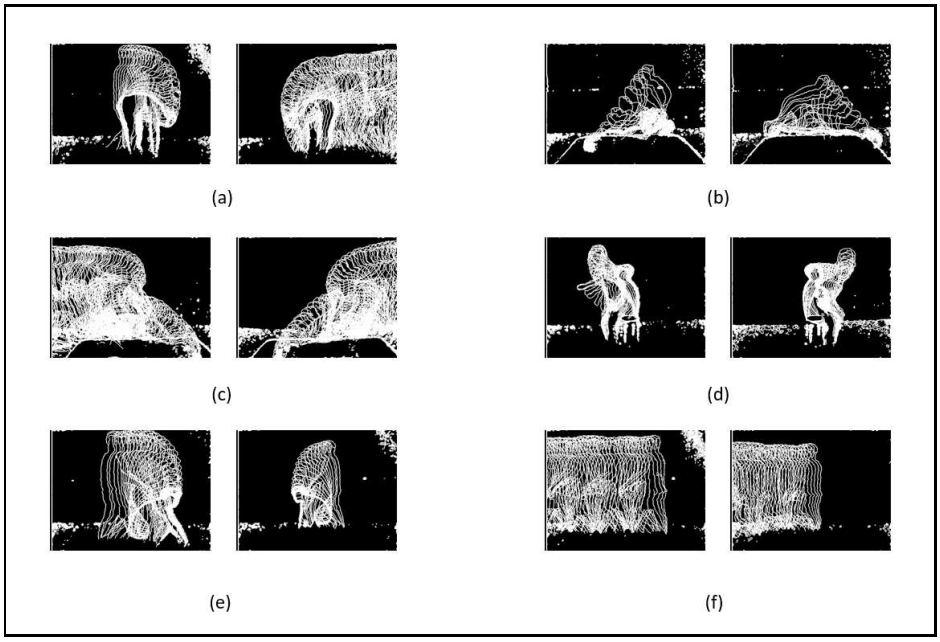

Figure 9. Samples of the depth map motions obtained for different classes. (a) Bending. (b) Falling. (c) Lying. (d) Squatting. (e) Sitting. (f) Walking.

The results obtained on our dataset's videos where not very well representative of the actions. Therefore, no further work was considered using this approach. Alternatively, an approach that uses a fixed number of video frames was adopted.

3.1.2 Sequences of frames

For this approach, it was important to maintain the same number of frames for all videos. This step is important for training a neural network as it allows creating batches of the data. Batching data minimizes the training signal noise. This means that batching eliminates continuous updates of the network parameters, and reduces

the computational overhead. Since the optimizer updates the network weights and biasses only after iterating over multiple samples included in the batch, the training performance is improved. Different experiments were used to analyze this approach as following.

3.1.2.1 Use the minimum number of video frames

This is an intuitive approach where the minimum number of frames for all videos was used as a starting point. All videos were trimmed to 69 frames. Thus, depending on the video length, all frames except the last 69 frames were discarded. Although some videos lost the complete representation of the human action, this approach showed promising results as the overall accuracy for action recognition was 84.3%. But as the action representation was lost on some videos, a better way for data representation needs to be considered.

3.1.2.2 Pad video frames

Since around 96% of the videos have a number of frames that falls in the range [150 - 300]. Only videos that fall in this range were considered in this approach. All videos of length greater than 150 and less than 300 were padded with frames of zeros. The dataset is relatively small which eliminates the option of further reducing this range by discarding more videos. Following this approach dropped the overall action classification accuracy to 27%. Padding the videos have created an association between the number of padding frames and the action presented in the video. Using this approach, a few falling videos were discarded since they do not fall in that range. Thus, this approach was abandoned.

3.1.2.3 Focus on falling frames

A considerable number of frames in the longer videos has no human present, and therefore does not provide any spatial or temporal information that affects the classification process. The minimum number of frames representing a fall-event is 99, and the maximum number of frames is 273. In consideration of that the focus of this study is on correctly recognizing fall-events, having a properly presented training and testing fall-events data is mandatory. Consequently, all videos from all actions that fall out of this range were excluded. Hence, we used about 95% of the videos in the dataset.

All videos whose number of frames fall in the range [99 - 273] were trimmed to 99 frames. The initial frames were discarded as they mostly are empty frames with no human present. The last 99 frames are kept as the human action representation is more stable towards the end of the action recorded. This way of trimming the videos preserved the human action representation. Trimming also randomized the initial location of the human in the videos. The randomization has a good effect on the learning algorithm, as it breaks the association of the initial location of the human and the actions performed in videos. Figure 10 gives more insight to this process.

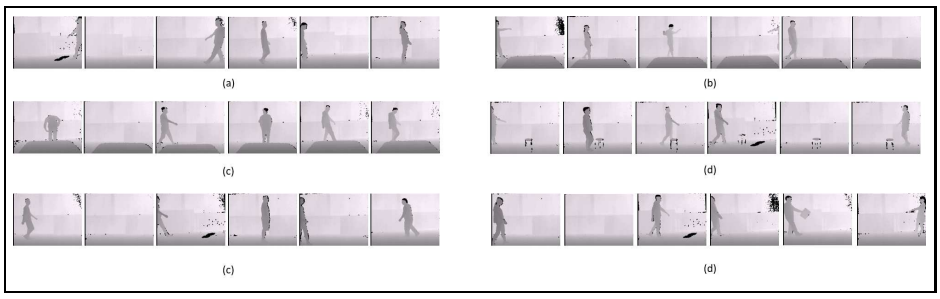

Figure 10. Sample of the first frame in different videos for different actions. (a) Bending. (b) Falling. (c) Lying. (d) Sitting. (e) Squatting. (f) Walking.

Using 99 video frames we could capture the full representation of the human action without redundant video frames. Thus, all the experiments conducted on training and testing the performance of the deep learning algorithm was based on using 99 video frames. The pseudocode for extracting these frames is:

1. *For i in range (videoLength - 99):*

   Discard Frame(i)

2. *For j in range (99):*

   Save Frame(j)

The details of the classification approaches, i.e. training and testing the neural network, are provided in the next section.

3.2 Classification approaches

The data representation approach described in section 3.1.2.3 was found to be the best fit for the goal of this research. Therefore, two classification approaches were considered; depth video classification, and binary video classification. In the following subsections, a description of these approaches is provided.

3.2.1 Depth videos classification

The videos in the dataset has frames of three channels. The depth dimension captured using the Microsoft Kinect sensor is sampled and transformed to three different channels for visualization. Two classification methods were considered using this approach; falling versus bending, lying, sitting, squatting, and walking actions, and falling versus non-falling actions. More details on these methods are provided in the following subsections.

3.2.1.1 Six-class classification

Here, 99 frames were extracted from each video whose length falls in the range [99 - 273]. All video frames were given one label based on the action in the video. The labels given are: bending, falling, lying-down, sitting, squatting, and walking. The dataset was split into two sets; a training set and a testing set. The training set contained 80% of the videos, and the testing set contained 20% of the videos. A sample of a fall-event video frames that were used to train the network are shown in figures 11.

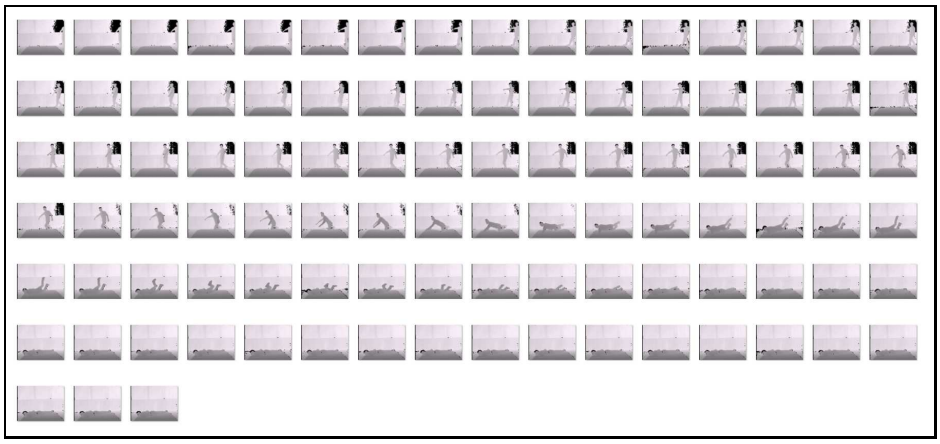

Figure 11. Sample of the falling frames used for training in the 6-class classification method.

Samples for the training frames of other actions are shown in appendix A.

3.2.1.2 Fall versus non-fall classification

The goal of this research is to build a fall-detection system that has high sensitivity and high specificity. The focus should be on classifying a fall event as a fall event and a non-fall event as a non-fall event, and how to possibly prevent falls on future by studying patterns of falling. Therefore, videos in the dataset whose length falls

in the range [99 - 273] were labeled as falling videos, or non-falling videos. Multiple experiments where conducted to test the fall versus non-fall approach. The details of these experiments are as follows:

1. Use all available training set videos and all available testing set videos.

For this approach, all the videos in the training and testing datasets where labeled as fall videos or non-fall videos. Thus, 810 videos were used for training, out of which 135 videos had a fall event. And 265 videos were used for testing, out of which 55 videos had fall events.

2. Use a subset of available training videos and a subset of available testing videos.

Using this approach, an equal number of videos were used for fall and non-fall events in both the training dataset and the testing dataset. That is, 135 fall videos and 135 randomly chosen non-fall videos were used for training. And 55 fall videos and 55 randomly chosen non-fall videos were used for training. Although the videos for non-fall events were randomly chosen, they had an equal number of videos from different classes from the original dataset.

3. Use a subset of available training videos and all available testing set videos.

This approach was used to insure that the approach described in the first experiment can generalize well for other videos in the test dataset. Here, the training dataset used is the one described in the second experiment. Whereas, the test dataset was composed of all test videos available in the dataset. More specifically, 135 fall videos and 135 randomly chosen non-fall videos were used for training. And 265 videos were used for testing, out of which 55 videos had fall events.

It should be noted that following the depth classification approach is not the final goal of this research. The ultimate goal is to be able to classify a human fall event

correctly irrespective of the surrounding objects. This generalization of fall-detection can be achieved on binary videos were the background is subtracted from the scene. There has not been much work done in literature on binary video classification using deep neural networks. So, depth video classification was conducted to verify that the system can learn to detect falls using a relatively small training dataset. The results were then taken one step further by trying to detect fall events in binary videos. The details for using binary videos in order to detect fall events are shown next.

3.2.2 Binary videos classification

This approach was followed for two main reasons. One, is to verify that the system can be generalized for classifying human actions under real life conditions. The goal was to extract the foreground object, which is the human, with the minimum background noise. Second, is for the sake of comparison with other available fall-detection algorithms that uses the dataset used in this research. These algorithms require processing binary video frames, where videos are split into training and testing datasets with 70% and 30% ratios. Therefore, these conditions were duplicated and tested using our algorithm.

For background subtraction, two algorithms were tested. The first experiment was based on the work in [61]. While the second experiment is based on the work presented in [24]. Both algorithms use Gaussian Mixture-based Background/Foreground Segmentation. The first algorithm uses K gaussian distributions for each pixel throughout the algorithm. Whereas, the second algorithm selects the appropriate number of gaussian distribution for each pixel. This selection allows the algorithm to adapt well to changes, such as illumination changes, in the scene. After the background subtraction, morphological operations; erosion and dilation, were performed

to reduce the noise. Several sizes of kernels were tested. It was found that the best background noise reduction is achieved using the second background subtraction algorithm with kernel size of 9 for both erosion and dilation operations. Figure 12 shows a sample of the result obtained using the first background subtraction algorithm. Figure 13 shows the results obtained using the second background subtraction algorithm.

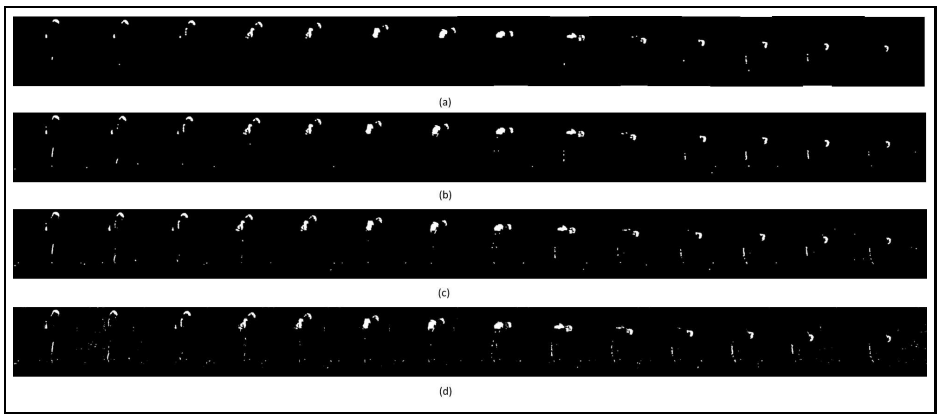

Figure 12. Samples of the frames obtained using the first background subtraction algorithm for a bending action. (a) using kernel size of 9. (b) using kernel size of 7. (c) using kernel size of 5. (d) using kernel size of 3.

Based on the experience gained through the previous classification approaches, discussed in section 3.2.1.1, better choices were made for training the network using the binary videos. First, we have learned that the network is more biased to data that dominate during the training phase. Therefore, a sample pool for training the network was created. This pool has equal number of fall and non-fall events. The data in the pool has all fall events, and a randomly sampled non-fall events from the original dataset. Four experiments were conducted using this pool; in each experiment 70% of the data was used for training, and 30% was used for testing. The

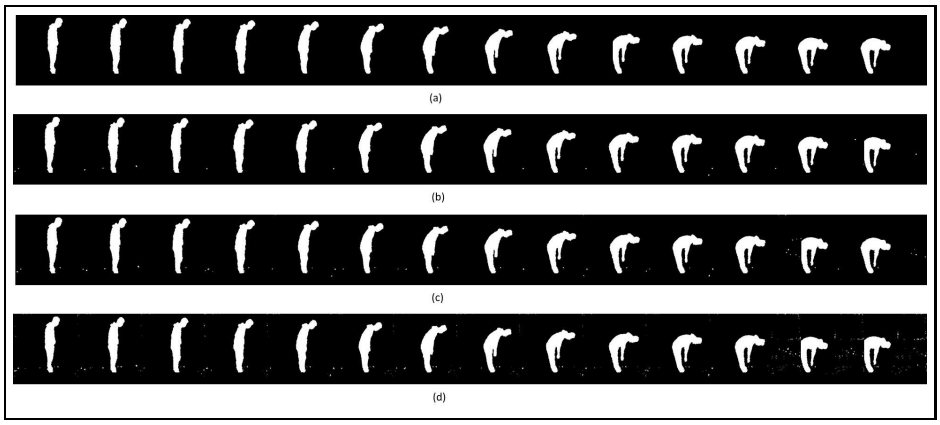

Figure 13. Samples of the frames obtained using the second background subtraction algorithm for a bending action. (a) using kernel size of 9. (b) using kernel size of 7. (c) using kernel size of 5. (d) using kernel size of 3.

videos for each experiment were chosen randomly. Table I show the data distribution for the four experiments.

Second, we have learned the importance of batching data and its effect on the learning progress of the network. A video whose frames are of size 240×320 were fed to the network. Due to memory constraints, the maximum batch size we could use was 2. The results during training were pretty random, and the network could not adjust its parameters represent the data. Therefore, all frames sizes were reduced to 160×120, and batches of 19 videos were fed to the network.

Samples of the fall-event video frames that were used to train the network during the different experiments are shown in figures 14 to 17.

More samples on the training video frames used for this approach are shown in appendix A.

	Training	Testing
Non-fall	131	59
Fall	135	55

(a) experiment 1

	Training	Testing
Non-fall	179	11
Fall	87	103

(b) experiment 2

	Training	Testing
Non-fall	160	30
Fall	106	84

(c) experiment 3

	Training	Testing
Non-fall	150	40
Fall	116	74

(d) experiment 4

Table I. The distribution of the data used for (a) experiment 1 (b) experiment 2 (c) experiment 3 (d) experiment 4.

3.3 Stream data classification

The system is also setup for processing data stream. A buffer that fits 99 frames is used. The system reads a new frame and adds it to the buffer after discarding the oldest frame. A label is produced with each frame added to the buffer. For the online data stream classification to function properly, the human action should be separated from the background. In other words, it is important to verify that the system works properly regardless of what the human surroundings are. Therefore, the binary videos classifier generated based on experiment 4 in section 3.2.2, were used for video stream processing. This classifier was chosen as it has the highest fall-detection accuracy. By classifier we are referring to the set of parameters generated by the 3D convolutional neural network.

Figure 14. Samples of the frames used for training in the fall versus non-fall classification using binary frames method experiment 1.

3.4 System Specifications

To train and test the model, TensorFlow library [15] was used on a Ubuntu 14.04 system that two GeForce GTX 980 Ti GPUs. Each GPU has 2816 CUDA cores. The open source library, TensorFlow, was developed by a team of engineers and researchers from Google in order to process machine learning and deep neural networks algorithms. TensorFlow uses graphs to represent numerical computations. Mathematical operations are represented by graph nodes. Data tensors communication between nodes is represented by edges in the graph. The following section describes in more details how the deep neural network used in this research was built.

3.5 Building the Network

While the 3D convolutional neural network approach was considered as the best fit for our problem, other experiments were done using a recurrent neural network to confirm that this choice was the best. Among other approaches, using a network of 2

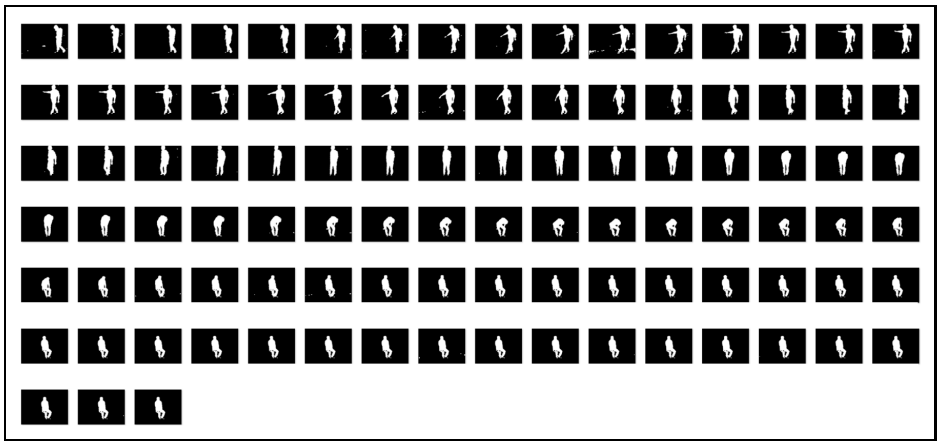

Figure 15. Samples of the frames used for training in the fall versus non-fall classification using binary frames method experiment 2.

LSTM cells was the most affordable in terms of the computational complexity. The following subsections describe the architectures of both networks.

3.5.1 Recurrent Neural Network

An implementation of the fall-detection system using a recurrent neural network with 2 LSTM layers was tested. Figure 18 shows an overview of the network used. Sequences of video frames were fed into this network, and a label to identify the action in that video is produced. The network has 20 hidden units, and processed a 99-frame video sequence. Adam optimizer was used with learning rate of 1e-4.

The network considers all the pixels in the sequences of frames A large portion of a video frame is raw video frames to the recurrent neural network have T The overall accuracy using this approach was below 50%. Therefore, no further experiments were conducted using this approach.

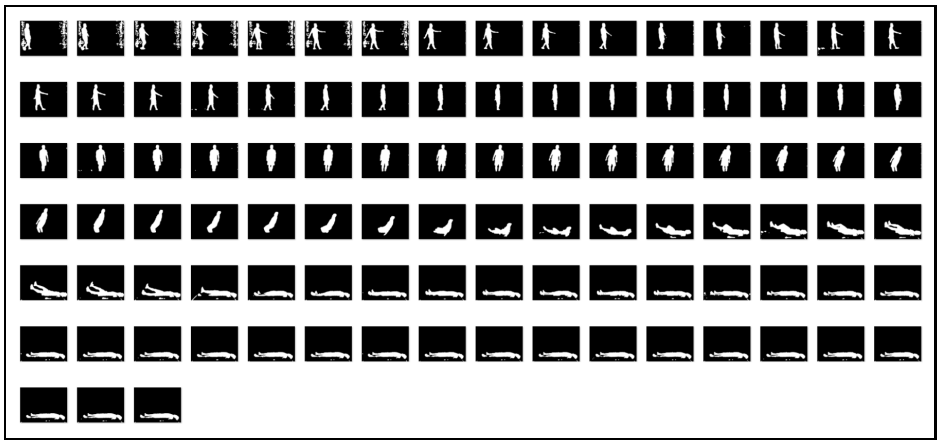

Figure 16. Samples of the frames used for training in the fall versus non-fall classification using binary frames method experiment 3.

3.5.2 3D Convolutional Neural Network

This study uses a 3D convolutional neural network for fall-detection. A video was fed to the network, and a class label is generated to determine whether a fall event has occurred. A detailed network architecture for the implementation is shown in figure 19.

The input to the network is a video of size $99 \times 160 \times 120 \times 3$. The video is then goes through the first 3D convolutional layer. This layer has 96 filters each of size $25 \times 11 \times 11$, where the filter spans the videos with a stride of 3 in the spatial domain and a stride of 3 in the temporal domain. The size of the filter in the temporal domain was chosen such that the network looks at sufficient number of frames before updating it's parameters. So, based on the conducted experiments using around one-fourth the number of video frames served this purpose. Next, we have a max-pooling layer that down-samples the resulting activation volume by 2 in every dimension. Following this pooling layer we have a second convolutional layer of 256 filters each of size $15 \times 5 \times 5$ that span the video with a stride of 2 in both spatial and temporal

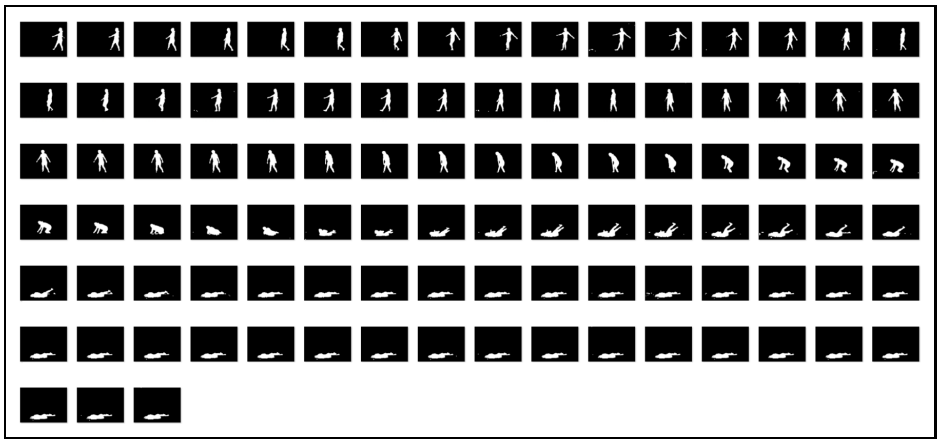

Figure 17. Samples of the frames used for training in the fall versus non-fall classification using binary frames method experiment 4.

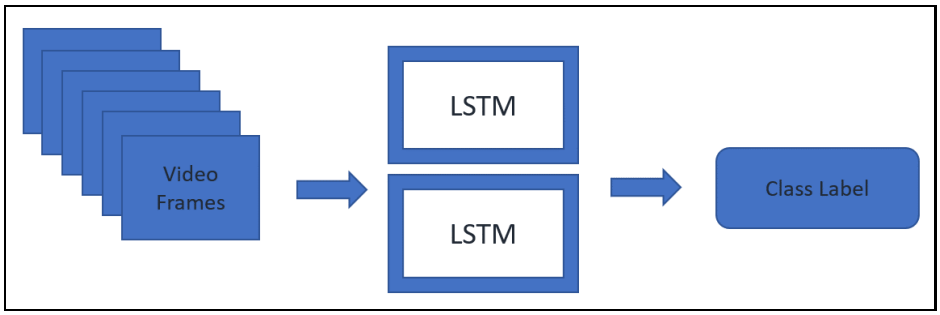

Figure 18. The recurrent neural network used for 6-class classification.

domains. Again, a pooling layer, with the same properties as the first pooling layer, follows this convolutional layer. We then have three convolutional layers of sizes $5 \times 3 \times 3$, $1 \times 3 \times 3$ and $1 \times 3 \times 3$ respectively. These convolutional layers are followed by a pooling layer. A summary of the filter sizes and strides for each layer is listed in table II. Finally, a fully-connected layer of 128 neurons is followed by another fully connected layer whose number of neurons is associated with the number of classes in the problem are used. A softmax function is applied to the output of the last fully-connected layer to compute the classes' probabilities.

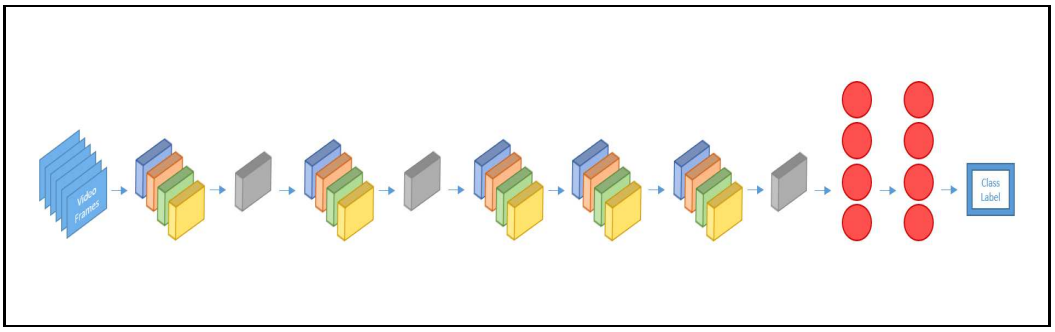

Figure 19. Detailed model architecture where video frames are mapped to class labels. The filters of 3D convolutional layers are shown in yellow, green, orange, and blue. The pooling layer filters are shown in grey. Fully connected layers are shown in red.

To train the network, Adam Optimizer algorithm was used. This is a gradient-based algorithm which requires little memory and is very efficient in terms of computations. The use of Adam optimizer fits problems of large data and parameters. Practically, this optimizer works well compared to other stochastic optimization methods [28]. The loss function used is the cross-entropy function given by equation 3.2. Using this function the optimizer minimizes the sum of the difference between labels and predictions of all samples in a batch.

$$Cost \ = -\sum_{i=2}^{i=N} Label_i - log(Prediction_i) \tag{3.2}$$

After experimenting with different hyper-parameters, the hyper-parameters of the network were set as follows: 0.7 for the dropout rate, 10 for the batch size, and 1e-4 for the learning rate. Two values for the dropout rate were evaluated; 0.5 and 0.7. Using the dropout technique means that the network randomly and temporarily removes some of the network units and their connections. Thus, the higher dropout rate is, the lower overfitting the network has [45]. Two values for

Layer	Spatial Filter		Temporal Filter	
	Size	Stride	Size	Stride
Conv1	11	3	25	3
Pool1	2	2	2	2
Conv2	5	2	15	2
Pool2	2	2	2	2
Conv3	3	1	5	3
Conv4	3	1	1	2
Conv5	3	1	1	2
Pool3	2	2	1	2

Table II. A summary of the filter sizes and strides for each layer in the deep fall-detection system.

the learning rate were also used for performance comparison, 1e-4 and 1e-8. The learning rate is used by the optimizer to adjust the weights and biases of the network. Different combinations of these values were tested, with accuracies below 70% for all combinations other than 0.7 for dropout rate, and 1e-4 for the learning rate. Several values for the batch size were tested. Figure 20 shows the accuracy changes across different values for the batch size. These sizes where used while training the network. For testing, all batch sizes where set to 5. This is because the batch size needs to be a divisor for the data size. The batch size cannot exceed 40, as this means loading a larger amount of data than what the system memory can handle. Therefore, the values of the batch size with a training set of 810 videos were: 10, 15, 18, and 30. The values of the batch size for a testing set of 265 videos was set to 5.

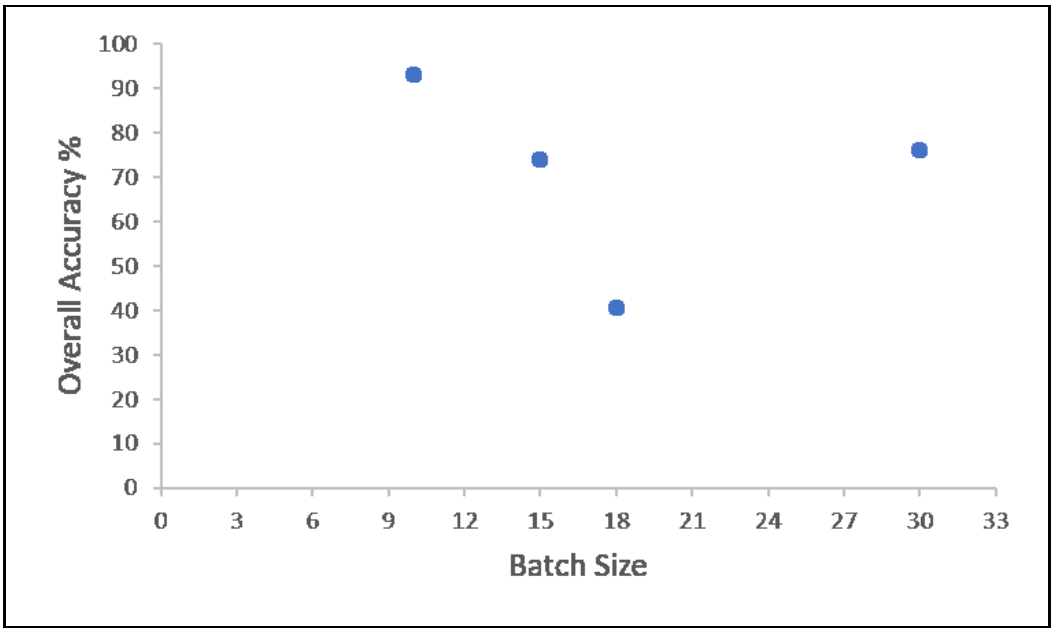

Figure 20. The overall six-classes classification accuracy using different batch sizes

CHAPTER 4

RESULTS AND DISCUSSION

The following sections describe in details the results of the two approaches used for training and testing the designed architecture. One approach was to detect fall events using depth videos, the other approach was based on binary videos. Based on the data preprocessing and preparation experiments, the best results for the 6-class classification on depth videos were obtained using 99 video frames. This number of frames was found to be optimum as shown in table III. Thus, all approaches use videos of 99 frames.

	Minimum no. of frames	Padding video frames	99 video frames
Accuracy	84.50%	27%	93.20%

Table III. Comparison of the different data preprocessing approaches.

The overall accuracy reported for all approaches is calculated using the equation:

$$Overall\ accuracy = \frac{Number\ of\ correctly\ classified\ videos}{Total\ number\ of\ videos\ in\ the\ test\ dataset} \times 100\% \quad (4.1)$$

Similarly, the accuracy reported for a class is calculated using the equation:

$$Class\ accuracy = \frac{Number\ of\ correctly\ classified\ class\ videos}{Total\ number\ of\ class\ videos\ in\ the\ test\ dataset} \times 100\% \quad (4.2)$$

All results are reported through a confusion matrix, where a row represents a class label, and a column represents a prediction. From the confusion matrix, other metrics such as the precision, recall, and F1-scores can be inferred. The precision can be expressed in terms of the relationship between the true positive and the true

negative results. It can be expressed by the equation:

$$Precision = \frac{True\ Positives}{True\ Positive\ +\ True\ Positive} \times 100\% \tag{4.3}$$

The recall metric is given by the equation:

$$Recall = \frac{True\ Positives}{True\ Positive\ +\ True\ Negative} \times 100\% \tag{4.4}$$

And the F1-score, which is one way of combining the precision and recall metrics, is given by:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision +\ Recall} \times 100\% \tag{4.5}$$

The results obtained for video stream are also listed. More samples of the correctly classified and miss-classified video frames are shown in appendix B and appendix C, respectively.

4.1 Depth videos classification

The results for depth videos classification were obtained based on two methods. One method distinguishes between all six classes presented in the dataset. The other classifies fall events among non-fall events. Following are the results in details.

4.1.1 Six-Class Classification

In this method, each video was given a label as bending, falling, lying-down, sitting, squatting, or walking. For the training phase, 135 videos were used per each action. The overall training time was around 32 minutes. A total of 265 videos were used for testing. The number of videos used for testing per class is as follows: 34 videos

for bending, 55 videos for falling, 35 videos for lying-down, 57 videos for sitting, 46 videos for squatting, and 38 videos for walking.

The estimated time for classifying one video is around 23 seconds. So, the system could achieve real time performance. Table IV shows the number of correctly classified videos, as well as the miss classified videos among all videos in the test dataset. The confusion matrix for the 6-class action recognition is shown in table V.

No. of videos		Bending	Falling	Lying-Down	Sitting	Squatting	Walking
34	Bending	31	0	0	0	3	0
55	Falling	0	48	7	0	0	0
35	Lying-Down	0	0	35	0	0	0
57	Sitting	2	0	0	54	0	1
46	Squatting	4	0	0	0	42	0
38	Walking	1	0	0	0	0	37

Table IV. The confusion matrix for the 6-class action recognition. It shows the numbers of correctly classified and miss classified videos.

The system could achieve 87.28% accuracy on fall events classification. The system could also achieve high specificity by correctly identifying daily life activities as non-falls. Interestingly, all lying-down events were correctly classified. The majority of miss classified falling events are given a lying-down label. This can be due to the fact that the dataset have simulated falls. Therefore, some of the fall events were performed more like lying-down events rather than falling events.

Samples of the correctly classified and miss classified falling video frames are shown in figures 73 and 22, respectively.

It can be seen that the miss-classified video frames share some patterns with a

93.2%	Bending	Falling	Lying-Down	Sitting	Squatting	Walking
Bending	91.18	0	0	0	8.82	0
Falling	0	87.28	12.72	0	0	0
Lying-Down	0	0	100	0	0	0
Sitting	3.51	0	0	94.74	0	1.75
Squatting	8.70	0	0	0	91.30	0
Walking	2.63	0	0	0	0	97.37

Table V. The confusion matrix for the 6-class action recognition. It shows the percentages for accuracies.

lying-down action, where the human bends towards a mattress.

4.1.2 Fall versus Non-Fall Classification

Since the goal of this research is to correctly classify fall-events as fall-events, otherwise, events should be classified as non-fall events. Several experiments were conducted for fall versus non-fall recognition. Using this approach, each video was given a label, either fall or non-fall. More details on these experiments are provided in the following subsections.

4.1.2.1 Use all available training set videos and all available testing set videos

For this classification approach, all videos available in the training set and testing set were used. Thus, for the training dataset, 275 videos were labeled as non-fall videos, and 135 videos were labeled as fall videos. For the testing dataset, 210 videos were labeled as non-fall videos, and 55 videos were labeled as fall videos. The confusion matrix using this approach is shown in table VI.

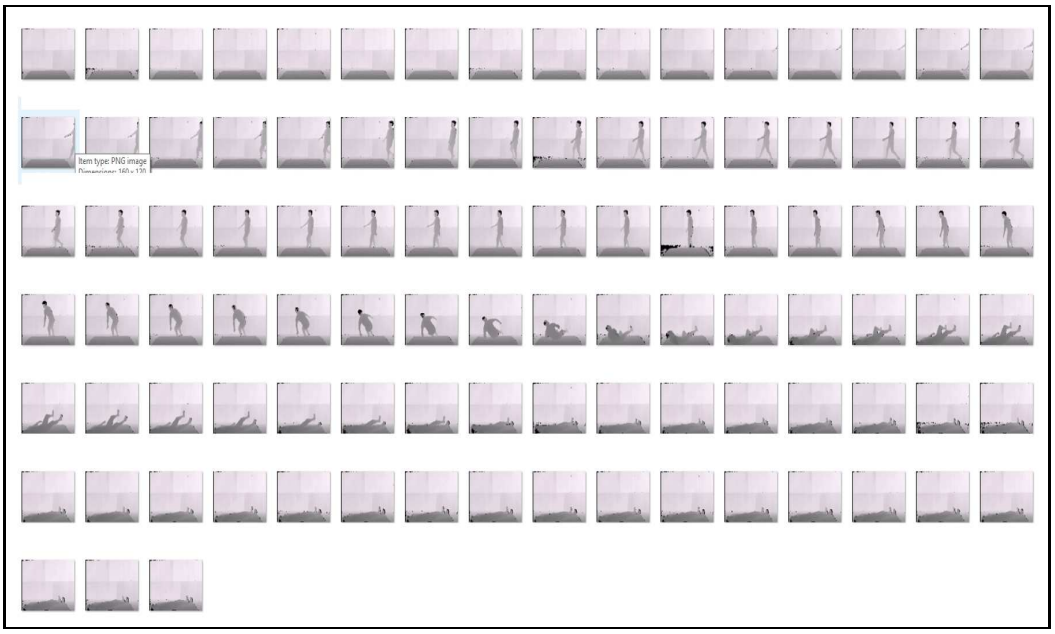

Figure 21. Sample of a correctly classified falling video frames using the six-class classification method.

The overall system accuracy using this approach is lower than the previously obtained accuracy. Both the precision and recall metrics of the system are 0%. This means that the network has learned to classify all videos as non-fall videos. The low precision and recall values are obtained due to the fact that the number of non-fall training videos is much larger the fall training videos. The unbalanced data distribution during the training phase makes the network more biased towards non-fall events.

4.1.3 Use a subset of available training videos and a subset of available testing videos

To overcome the problem of having the network biased towards non-fall events, a subset of the available training videos were used. Since 135 is the total number of

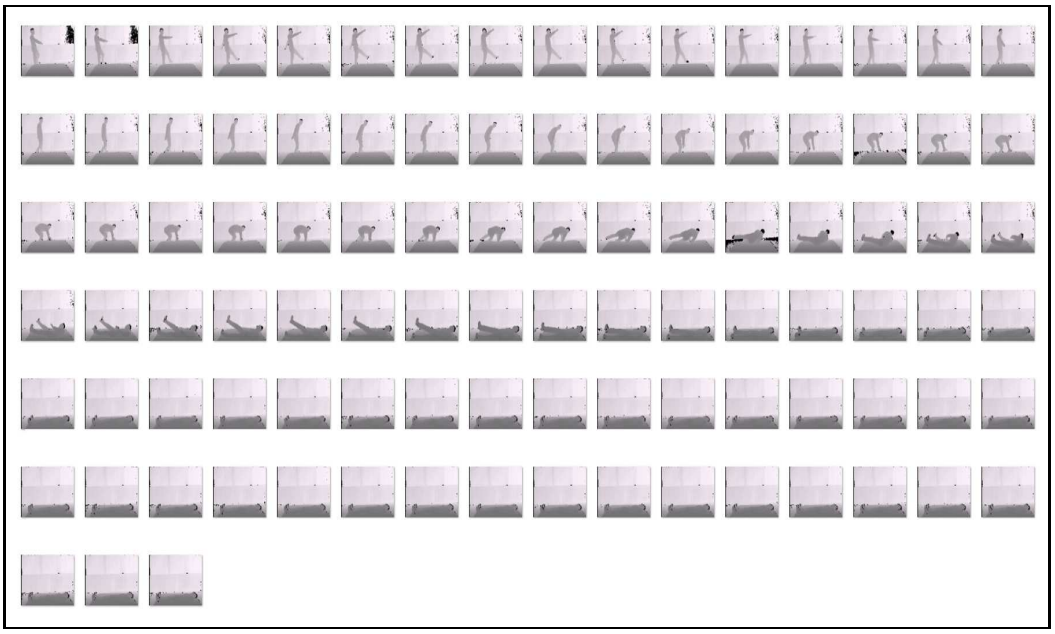

Figure 22. Sample of a miss-classified falling video frames using the six-class classification method.

fall events in the training dataset, 135 videos were used for falling and a similar number was used for non-falling events. A total of 110 videos were used for testing. The number of videos used for testing is 55 for falling, as well as non-falling events. The confusion matrix using this approach is shown in table VII.

The overall system accuracy is 97.2%. In order to ensure that this way of training the network and testing it can generalize well a more general approach was considered. This general approach uses a subset of the dataset for training, and all the test dataset. The details of this approach are explained in the following section.

4.1.4 Use a subset of available training videos and all available testing set videos

Using this approach, 135 fall videos and 135 non-fall videos were used for training the network. For the testing phase, 55 videos were used for fall events, and 210 videos

No. of videos		Non-Falling	Falling
210	Non-Falling	210	0
55	Falling	55	0

(a) The number of correctly classified and miss classified videos

79.2	Non-Falling	Falling
Non-Falling	100	0
Falling	100	0

(b) The percentages for accuracies

Table VI. The confusion matrix for the fall vs. non-fall action recognition using all training data and all testing data. (a) The number of correctly classified and miss classified videos. (b) The percentages for accuracies.

No. of videos		Non-Falling	Falling
55	Non-Falling	53	2
55	Falling	1	54

(a) The number of correctly classified and miss classified videos

97.2	Non-Falling	Falling
Non-Falling	96.3	3.7
Falling	1.9	98.1

(b) The percentages for accuracies

Table VII. The confusion matrix for the fall vs. non-fall action recognition using a subset of training data and a subset of testing data. (a) The number of correctly classified and miss classified videos. (b) The percentages for accuracies.

were used for non-fall events. The confusion matrix using this approach is shown in table VIII.

With 97% overall system accuracy, 98.18% sensitivity, and 96.7% specificity, the system could achieve better performance using a statistically balanced dataset during the training phase. The system could generalize well to all videos in the test dataset. The precision is found to be 88.5%, the recall is 98.1%, and the F1-score is 94.6%.

No. of videos		Non-Falling	Falling
55	Non-Falling	53	2
55	Falling	1	54

(a) The number of correctly classified and miss classified videos

97.20	Non-Falling	Falling
Non-Falling	96.30	3.70
Falling	1.90	98.10

(b) The percentages for accuracies

Table VIII. The confusion matrix for the fall vs. non-fall action recognition using a subset of training data and a subset of testing data. (a) The number of correctly classified and miss classified videos. (b) The percentages for accuracies.

4.2 Binary video classification

In this approach, we stick to the fall versus non-fall classification approach as we aim to achieve high fall-detection accuracy, rather than identifying daily life activities. Initially, video frames were resized to 320×240. Training the network on these images were not possible due to the computer's memory limitations. Using large images, we were able to feed a batch of size 2 to the network. The small batch size prevented the stabilization of the network's parameters. Thus the results were random. To address this problem, all video frames were resized to 160×120. The smaller frame size allowed creating batches of 19 videos. The updates of the network parameters were more stable. Therefore, we could achieve better results.

For the experiments using binary videos, we have created a subset of the original dataset. This subset contained all fall videos, and an equal number of randomly chosen non-fall videos from the original dataset. The subset was then quadruplicated. In each copy of the subset, videos were split to 70% training set, and 30% testing set. The videos in the training and testing sets were also chosen randomly. The numbers of fall and non-fall videos for each experiment are listed in table I. The confusion

matrix using the four experiments is shown in table IX.

98.2	Non-Falling	Falling
Non-Falling	98.3	1.7
Falling	1.9	98.1

(a) experiment 1

96.5	Non-Falling	Falling
Non-Falling	100	0
Falling	3.9	96.1

(b) experiment 2

97.4	Non-Falling	Falling
Non-Falling	96	4
Fall	2.4	97.6

(c) experiment 3

98.2	Non-Falling	Falling
Non-Falling	97.5	2.5
Falling	1.4	98.6

(d) experiment 4

Table IX. The confusion matrix for the fall vs. non-fall action recognition using binary videos. It shows the percentages for accuracies. (a) experiment 1 (b) experiment 2 (c) experiment 3 (d) experiment 4. It shows the percentages for accuracies.

For the first and fourth experiments, the sytem's precision, recall, and F1-score are identical; for experiment one all values are 98.1%, and for experiment four all values are 98.6%. For the second experiment the precision is found to be 100%, the recall is 96.1%, and the F1-score is 98%. Whereas, for the third experiment the precision is 98.8%, the recall is 97.6%, and the F1-score is 98.2%.

While it could be expected that using low resolution frames, the accuracy of the system would drop, the average accuracy of the system is 97.58%. To the best of our knowledge, this is the highest accuracy reported for fall-event recognition using this dataset. In other work published on the dataset used in this research described in [2][3][34], different authors disagree on the average video length. This leaves an open question for a clarification on handling the variations of video lengths. It is

also noted that there is no mention of the number of fall-event videos included in the test dataset. The ration of fall events to non-fall events in the test dataset is important as it highly affects the reported accuracy. Though, better performance than the approaches which uses hand-crafted features such as the orientation volume and the curvature scale space of human silhouettes described in [2][34] respectively. The results on the SDUFall dataset were further generalized and tested on online data stream.

Table X, shows a comparison of our approach to approaches which uses hand-crafted features and shallow classifiers.

	BoW-VPSO- ELM[34]	FV-SVM[3]	BoSOV-Bayes[2]	Proposed method
Accuracy	86.83	88.83	91.89	97.58

Table X. Comparison of the performance of the fall vs. non-fall action recognition. It shows the percentages for accuracies.

Samples of the correctly classified and miss classified falling video frames for the four experiments are shown in figures 23 to 30.

Unlike other videos in the test dataset, the miss-classified video frames in figure 26 has lots of noise. It could be that due to the huge noise that the video was miss-classified. For the video frames in figures 24, 28, and 30, a human can be easily deceived by the action performed, since it shows minimum similarities with an actual fall-event. Thus, it is expected that the machine would also miss-classify the actions in these videos.

We can see that due to the fact that the dataset has falls performed by stunt actors, some of the fall actions share lots of the characteristics of a lying-down or bending actions. Other fall events that are performed in a more realistic way are

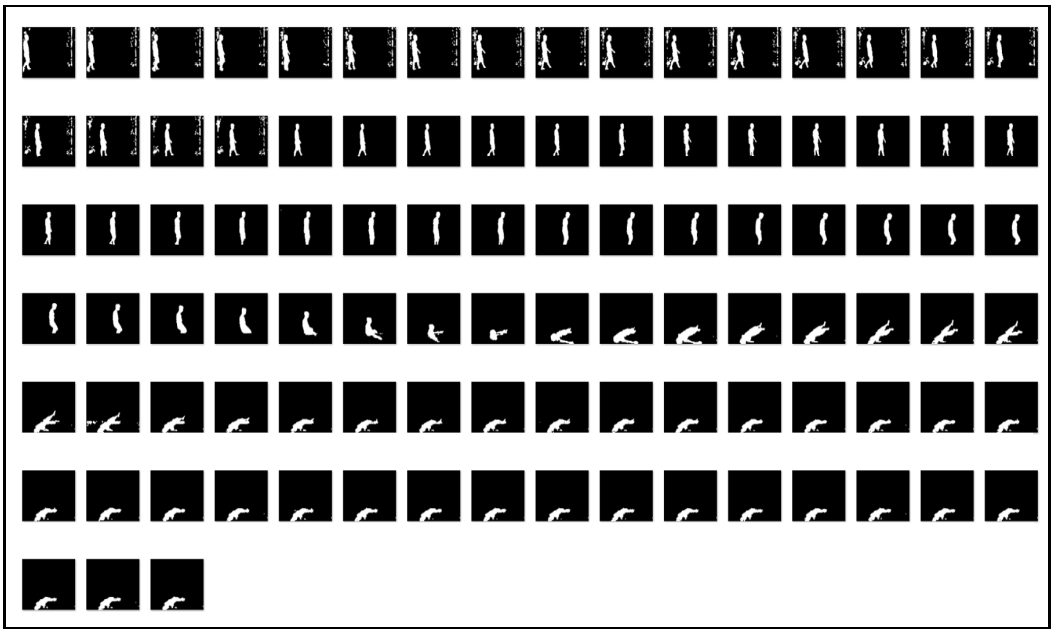

Figure 23. Sample of correctly classified falling video frames in the first experiment of fall versus non-fall binary video classification.

always correctly classified.

4.3 Data stream classification

For the data stream classification, a buffer was setup to fit 99 video frames. At each time step, a new frame is read from a camera, and the oldest video frame in the buffer is discarded. A class label is produced with each frame added to the buffer. The experiments were conducted in the computer vision lab, Bina, using a regular web camera. The camera was placed at a height of 1.5m. Background subtraction was performed to binarize all video frames.

Having a fall-detection system with high recall is more important than having a system with high precision. Higher recall means the system has higher ability of identifying fall events correctly. Higher precision means that the system generates

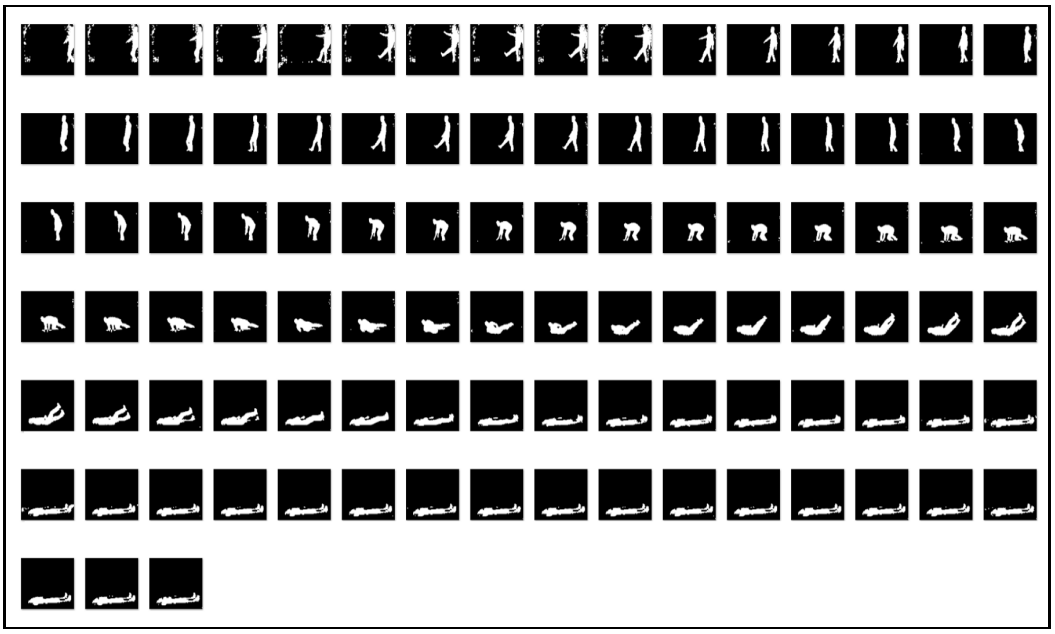

Figure 24. Sample of miss-classified falling video frames in the first experiment of fall versus non-fall binary video classification.

less false alarms. Therefore, the recall metric is more critical for our application. The classifier generated using binary video frames as described in experiment four in section 3.2.2 was used. This classifier could achieve the highest fall-detection accuracy on a test dataset. This classifier also has the highest recall percentage among the other classifiers.

In daily life, different human actions have similarities in video frame sequences. To avoid high false alarm ration, due to these similarities, a threshold was used. As a new video frame is read and an old frame is discarded, the system produces a class label. The label can be fall, or non-fall. For a threshold of value n the system would generate an alarm signal after n consecutive fall labels. The system was tested using several values of n. The actions performed are: sitting, bending, squatting, lying-down, and throwing an object. The low threshold values, 10 and 15, produced

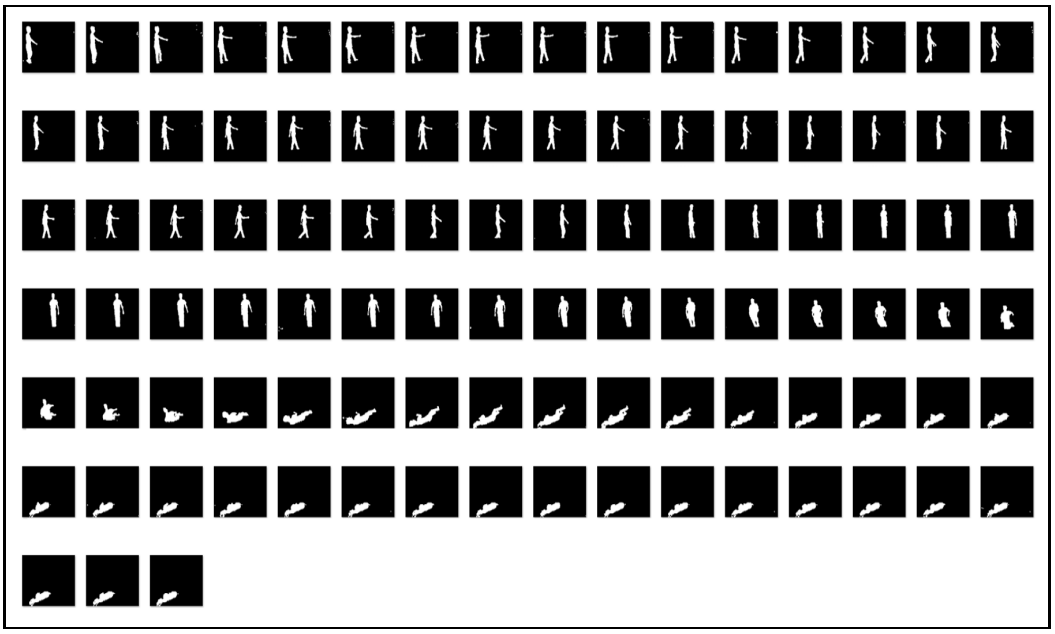

Figure 25. Sample of correctly classified falling video frames in the second experiment of fall versus non-fall binary video classification.

too many false alarms. Higher threshold values, 60 and 70, did not produce any false alarms even for lying-down events. Samples of the video frames for the actions performed are shown in figures 31 to 36.

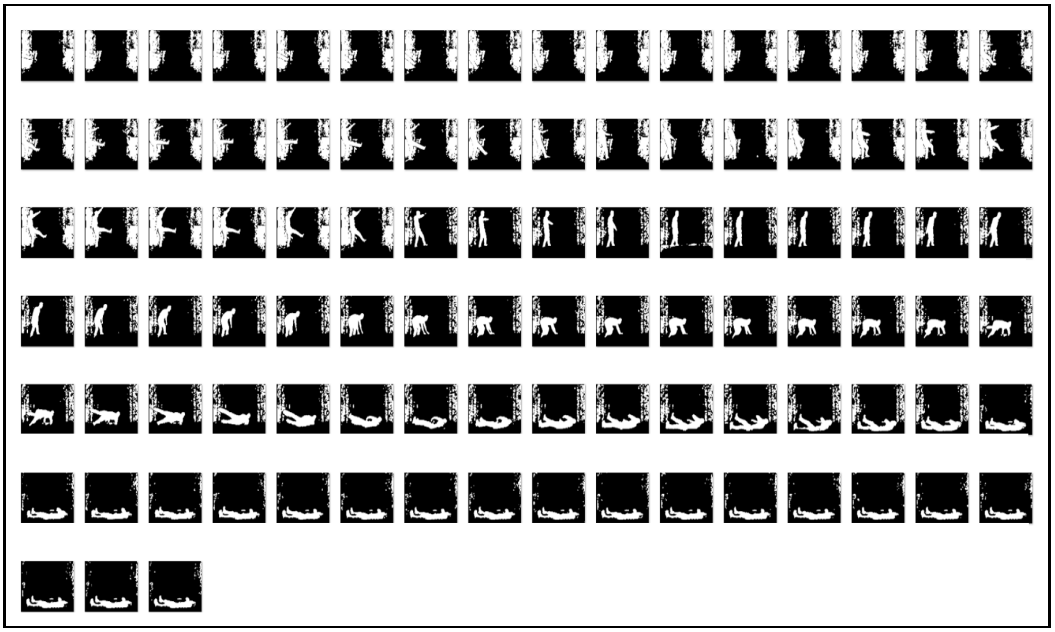

Figure 26. Sample of miss-classified falling video frames in the second experiment of fall versus non-fall binary video classification.

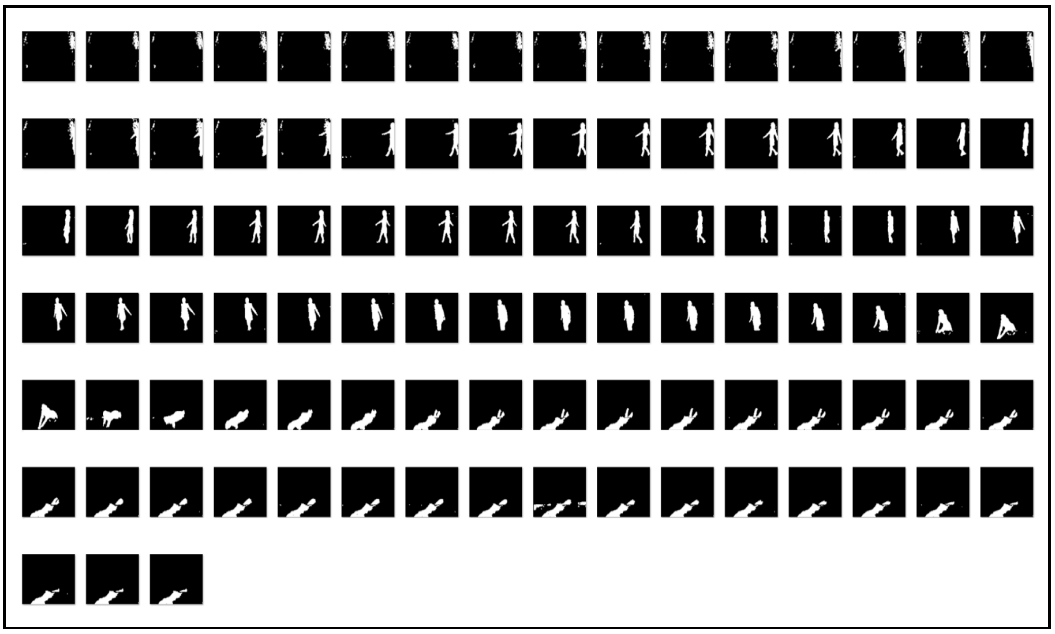

Figure 27. Sample of correctly classified falling video frames in the thirds experiment of fall versus non-fall binary video classification.

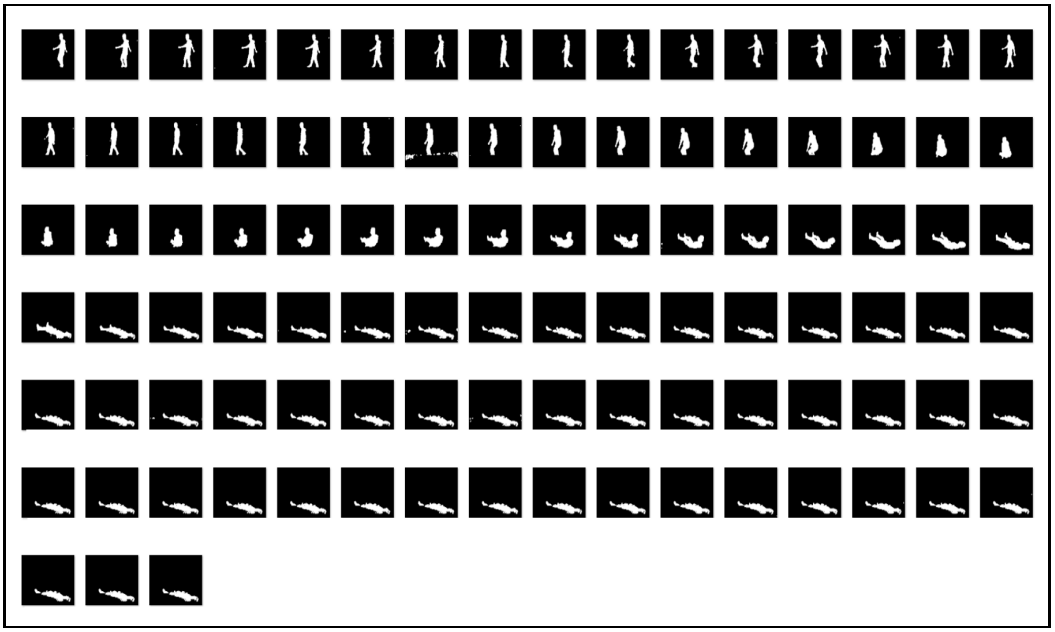

Figure 28. Sample of miss-classified falling video frames in the third experiment of fall versus non-fall binary video classification.

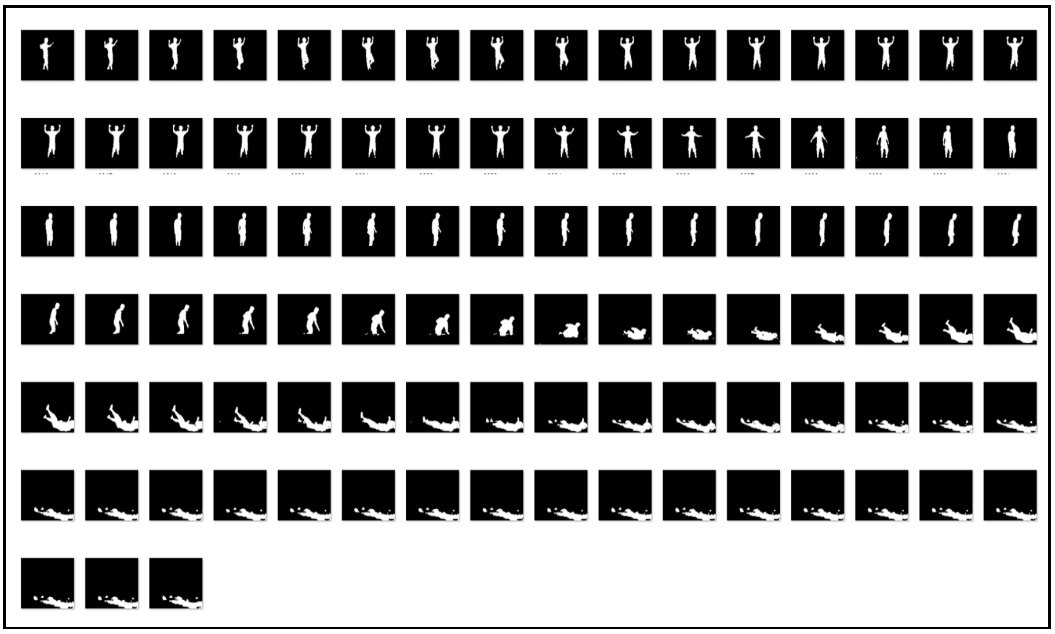

Figure 29. Sample of correctly classified falling video frames in the fourth experiment of fall versus non-fall binary video classification.

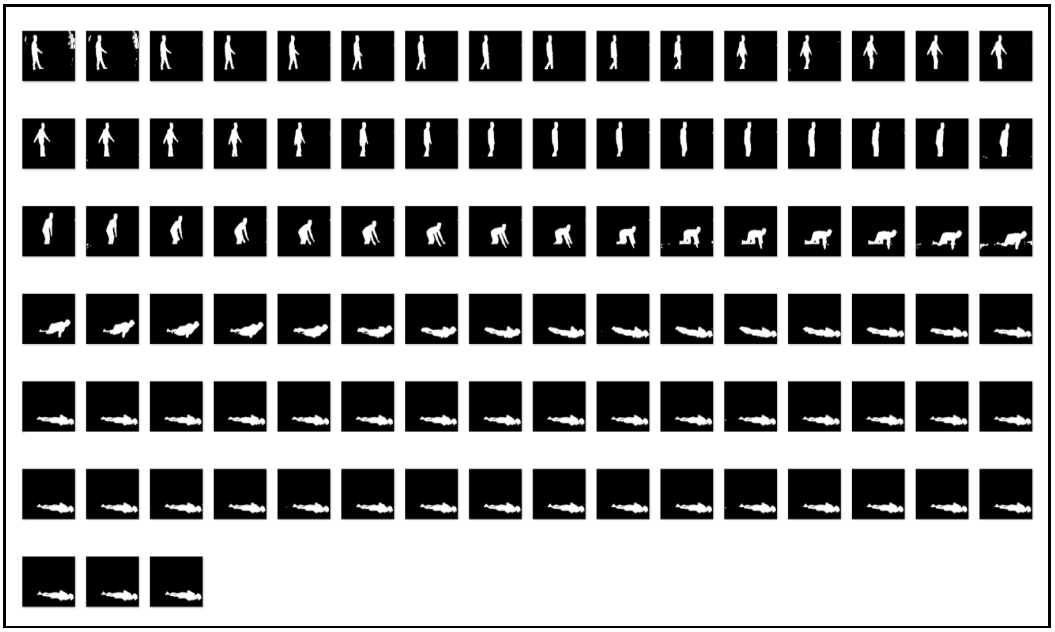

Figure 30. Sample of miss-classified falling video frames in the fourth experiment of fall versus non-fall binary video classification.

Figure 31. Sample of the walking action performed for the online data stream processing.

Figure 32. Sample of the bending action performed for the online data stream processing.

Figure 33. Sample of the throw object action performed for the online data stream processing.

Figure 34. Sample of the squatting action performed for the online data stream processing.

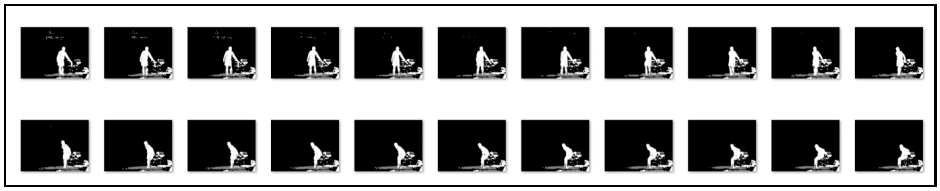

Figure 35. Sample of the sitting action performed for the online data stream processing.

Figure 36. Sample of the lying-down action performed for the online data stream processing.

CHAPTER 5

CONCLUSION

In conclusion, the research presented in this study shows that deep learning based algorithms are suitable for recognizing fall-event patterns. It is observed that the deep learning based pattern representations help increasing the accuracy for fall-detection systems as compared to traditional pattern representation techniques such as the orientation volume and the curvature scale space of human silhouettes described in [2][34] respectively. The main idea in this research is to identify the aforementioned patterns using spatio-temporal features. These spatio-temporal features were automatically learned and extracted using 3D convolution neural network.

The main contribution of this research lies in presenting an extensive effort to come up with a best representation of videos which would allow the system to detect fall events with high accuracy. The classification approaches of the experiments conducted in this study are twofold: the first is based on depth video classification, and the second is based on binary video classification.

The experimental results reveal that the presented approach improved the overall accuracy of the system. Since the results are encouraging, the system was setup for online data stream processing. This system can be employed under real life conditions.

As a future work, we would like to evaluate the performance of the overall system by incorporating the algorithm we implemented with the communication system to inform a health care provider about the occurance of a fall-event. Moreover, we also would like to consider more specific fall events. As described in [42], a considerable amount of falls among the elderly committee occurs while using wheel chairs or

walkers. More tests can be done to verify the ability of the system to detect such falls. More data to simulate these specific fall events and train the system to correctly classify them can be added as necessary.

REFERENCES

[1] Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., and Vecchio, A. A smartphone-based fall detection system. *Pervasive and Mobile Computing 8*, 6 (2012), 883–899.

[2] Akagunduz, E., Aslan, M., Sengur, A., Wang, H., and Ince, M. Silhouette orientation volumes for efficient fall detection in depth videos. *IEEE journal of biomedical and health informatics* (2016).

[3] Aslan, M., Sengur, A., Xiao, Y., Wang, H., Ince, M. C., and Ma, X. Shape feature encoding via fisher vector for efficient fall detection in depth-videos. *Applied Soft Computing 37* (2015), 1023–1028.

[4] Bengio, Y., Goodfellow, I. J., and Courville, A. Deep learning. *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook* (2015).

[5] Bosch-Jorge, M., Sánchez-Salmerón, A.-J., Valera, Á., and Ricolfe-Viala, C. Fall detection based on the gravity vector using a wide-angle camera. *Expert Systems with Applications 41*, 17 (2014), 7980–7986.

[6] Chen, C., Liu, K., and Kehtarnavaz, N. Real-time human action recognition based on depth motion maps. *Journal of real-time image processing 12*, 1 (2016), 155–163.

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Choi, Y., Ralhan, A., and Ko, S. A study on machine learning algorithms for fall detection and movement classification. In *Information Science and Applications (ICISA), 2011 International Conference on* (2011), IEEE, pp. 1–8.

[9] Chu, H., Urtasun, R., and Fidler, S. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477* (2016).

[10] Cucchiara, R., Prati, A., and Vezzani, R. A multi-camera vision system for fall detection and alarm generation. *Expert Systems 24*, 5 (2007), 334–345.

[11] Dai, J., Bai, X., Yang, Z., Shen, Z., and Xuan, D. Perfalld: A pervasive fall detection system using mobile phones. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on* (2010), IEEE, pp. 292–297.

[12] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2625–2634.

[13] for Community Living, A. Administration for community living, 2016.

[14] for Disease, C., and Prevention. Centers for disease and prevention, 2016.

[15] Goolge. Tensorflow, 2016.

[16] Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[17] Graves, A., and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In *ICML* (2014), vol. 14, pp. 1764–1772.

[18] Habib, M. A., Mohktar, M. S., Kamaruzzaman, S. B., Lim, K. S., Pin, T. M., and Ibrahim, F. Smartphone-based solutions for fall detection and prevention: challenges and open issues. *Sensors 14*, 4 (2014), 7181–7208.

[19] Hazelhoff, L., Han, J., et al. Video-based fall detection in the home using principal component analysis. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (2008), Springer, pp. 298–309.

[20] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[21] Igual, R., Medrano, C., and Plaza, I. Challenges, issues and trends in fall detection systems. *Biomedical engineering online 12*, 1 (2013), 1.

[22] Irsoy, O., and Cardie, C. Opinion mining with deep recurrent neural networks. In *EMNLP* (2014), pp. 720–728.

[23] Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence 35*, 1 (2013), 221–231.

[24] KaewTraKulPong, P., and Bowden, R. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*. Springer, 2002, pp. 135–144.

[25] Karpathy, A. Andrej karpathy blog, 2016.

[26] Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[27] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2014), pp. 1725–1732.

[28] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[29] Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. A clockwork rnn. arxiv preprint. *arXiv 1402* (2014).

[30] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[31] Kwolek, B., and Kepski, M. Improving fall detection by the use of depth sensor and accelerometer. *Neurocomputing 168* (2015), 637–645.

[32] Li, H., and Yang, Y.-L. Research of elderly fall detection based on dynamic time warping algorithm. In *Control Conference (CCC), 2016 35th Chinese* (2016), IEEE, pp. 5190–5194.

[33] Li, Y., Ho, K., and Popescu, M. A microphone array system for automatic fall detection. *IEEE Transactions on Biomedical Engineering 59*, 5 (2012), 1291–1301.

[34] Ma, X., Wang, H., Xue, B., Zhou, M., Ji, B., and Li, Y. Depth-based human fall detection via shape features and improved extreme learning machine. *IEEE journal of biomedical and health informatics 18*, 6 (2014), 1915–1922.

[35] Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. In *Advances in neural information processing systems* (2014), pp. 2204–2212.

[36] Mubashir, M., Shao, L., and Seed, L. A survey on fall detection: Principles and approaches. *Neurocomputing 100* (2013), 144–152.

[37] Nielsen, M. A. Neural networks and deep learning, 2016.

[38] Olah, C. Colahs blog, 2016.

[39] Planinc, R., and Kampel, M. Robust fall detection by combining 3d data and fuzzy logic. In *Asian Conference on Computer Vision* (2012), Springer, pp. 121–132.

[40] Rezaee, K., Haddadnia, J., and Delbari, A. Modeling abnormal walking of the elderly to predict risk of the falls using kalman filter and motion estimation approach. *Computers & Electrical Engineering 46* (2015), 471–486.

[41] Rimminen, H., Lindström, J., Linnavuo, M., and Sepponen, R. Detection of falls among the elderly by a floor sensor using the electric near field. *IEEE transactions on information technology in biomedicine: a publication of the IEEE Engineering in Medicine and Biology Society 14*, 6 (2010), 1475–1476.

[42] Robinovitch, S. N., Feldman, F., Yang, Y., Schonnop, R., Leung, P. M., Sarraf, T., Sims-Gould, J., and Loughin, M. Video capture

of the circumstances of falls in elderly people residing in long-term care: an observational study. *The Lancet 381*, 9860 (2013), 47–54.

[43] Rougier, C., Meunier, J., St-Arnaud, A., and Rousseau, J. Robust video surveillance for fall detection based on human shape deformation. *IEEE Transactions on Circuits and Systems for Video Technology 21*, 5 (2011), 611–622.

[44] Simonyan, K., and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems* (2014), pp. 568–576.

[45] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[46] Stanford. Cs231n convolutional neural networks for visual recognition, 2016.

[47] Stone, E. E., and Skubic, M. Fall detection in homes of older adults using the microsoft kinect. *IEEE journal of biomedical and health informatics 19*, 1 (2015), 290–301.

[48] Su, B. Y., Ho, K., Rantz, M. J., and Skubic, M. Doppler radar fall activity detection using the wavelet transform. *IEEE Transactions on Biomedical Engineering 62*, 3 (2015), 865–875.

[49] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (2014), pp. 3104–3112.

[50] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112.

[51] Szegedy, C., Toshev, A., and Erhan, D. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems* (2013), pp. 2553–2561.

[52] Tang, D., Qin, B., and Liu, T. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP* (2015), pp. 1422–1432.

[53] Tong, L., Song, Q., Ge, Y., and Liu, M. Hmm-based human fall detection and prediction method using tri-axial accelerometer. *IEEE Sensors Journal 13*, 5 (2013), 1849–1856.

[54] Tzeng, H.-W., Chen, M.-Y., and Chen, J.-Y. Design of fall detection system with floor pressure and infrared image. In *2010 International Conference on System Science and Engineering* (2010), IEEE, pp. 131–135.

[55] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 4534–4542.

[56] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3156–3164.

[57] Vishwakarma, V., Mandal, C., and Sural, S. Automatic detection of human fall in video. In *International conference on pattern recognition and machine intelligence* (2007), Springer, pp. 616–623.

[58] Yao, K., Cohn, T., Vylomova, K., Duh, K., and Dyer, C. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790* (2015).

[59] Yuwono, M., Moulton, B. D., Su, S. W., Celler, B. G., and Nguyen, H. T. Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *Biomedical engineering online 11*, 1 (2012), 1.

[60] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems* (2014), pp. 487–495.

[61] Zivkovic, Z., and Van Der Heijden, F. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters 27*, 7 (2006), 773–780.

Appendices

Appendix A

SAMPLE TRAINING VIDEO FRAMES

This chapter includes samples of the video frames used for training the deep neural network in different experiments.

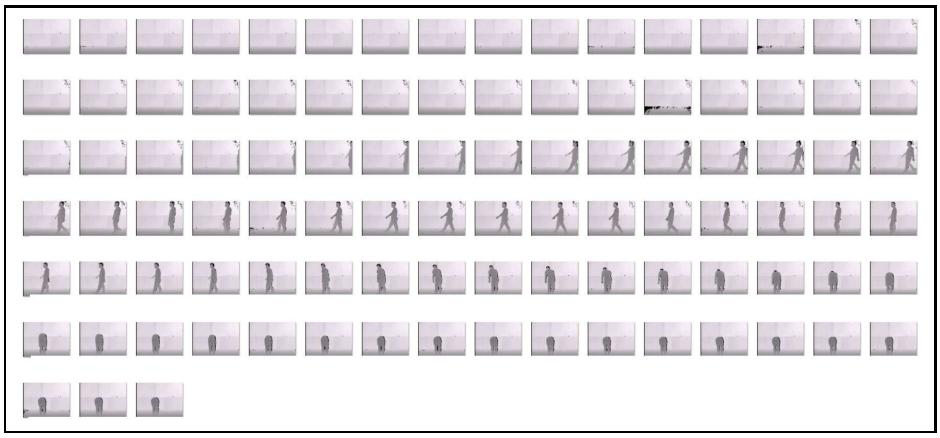

Figure 37. Sample of the bending frames used for training in the 6-class classification method.

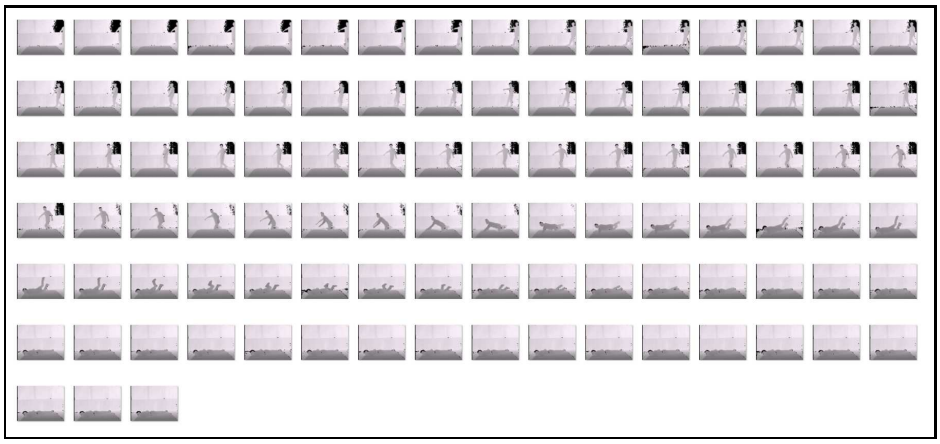

Figure 38. Sample of the falling frames used for training in the 6-class classification method.

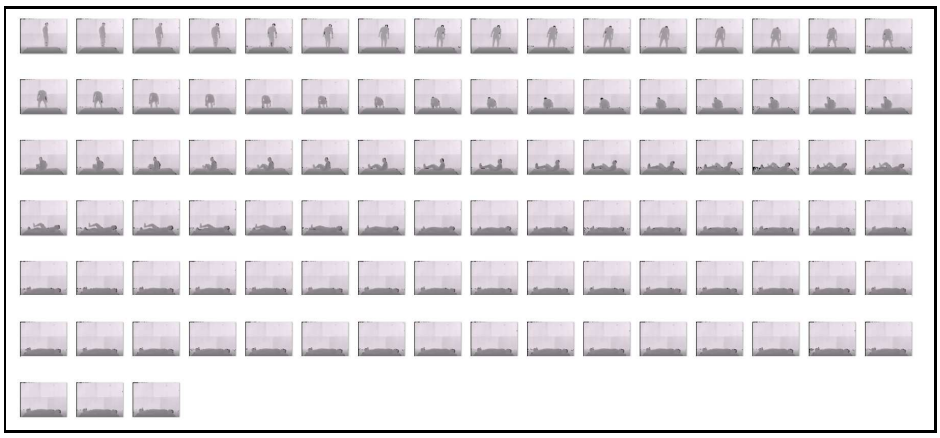

Figure 39. Sample of the lying-down frames used for training in the 6-class classification method.

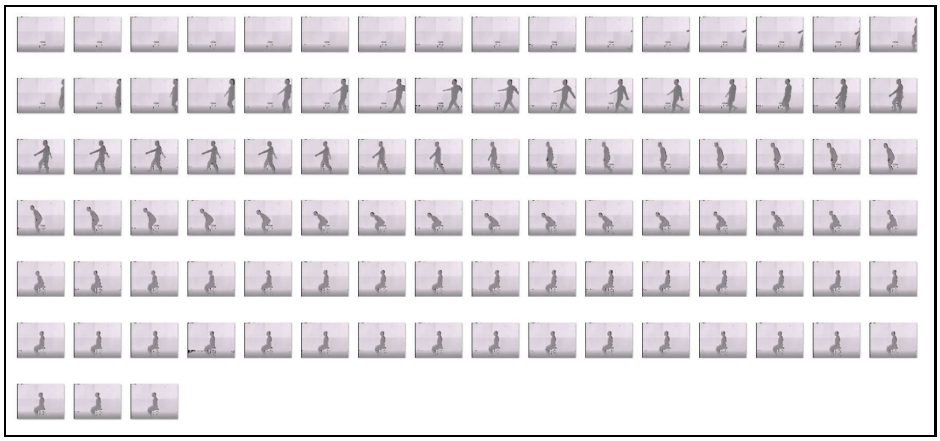

Figure 40. Sample of the bending frames used for training in the 6-class classification method.

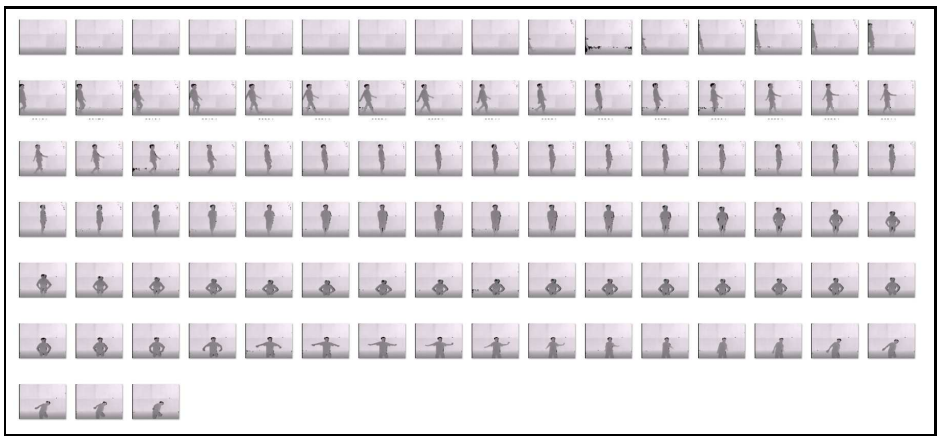

Figure 41. Sample of the bending frames used for training in the 6-class classification method.

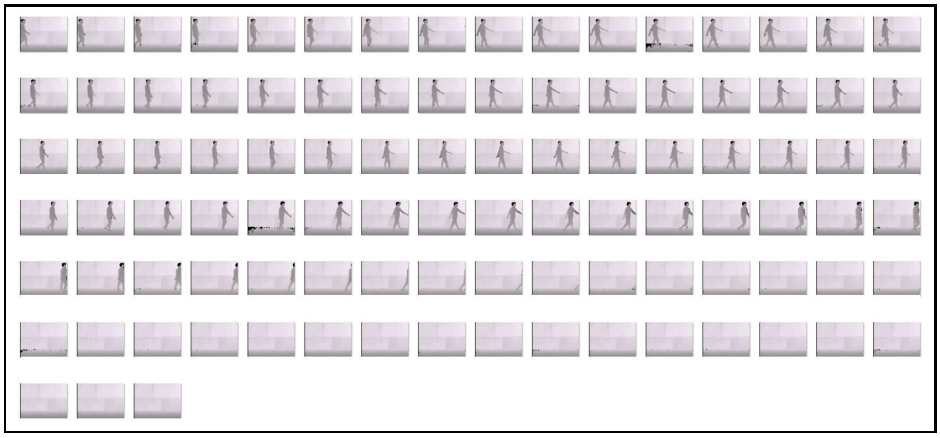

Figure 42. Sample of the bending frames used for training in the 6-class classification method.

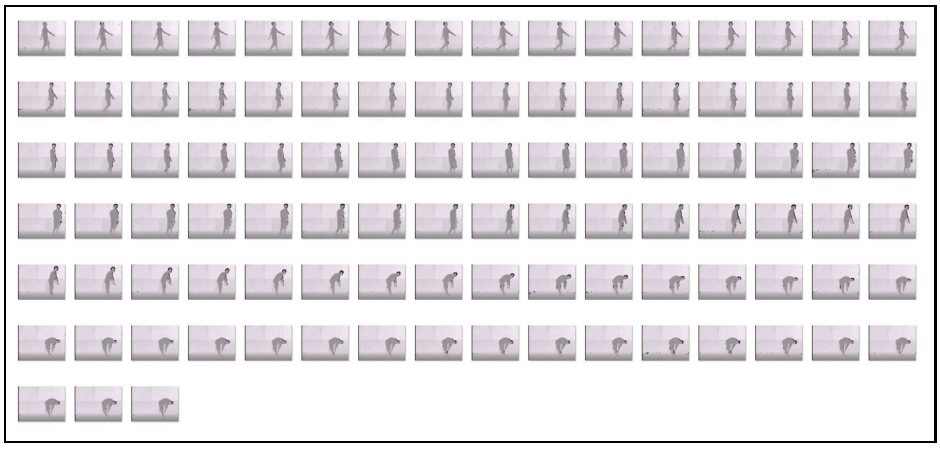

Figure 43. Sample of the non-fall frames used for training in the fall versus non-fall classification method.

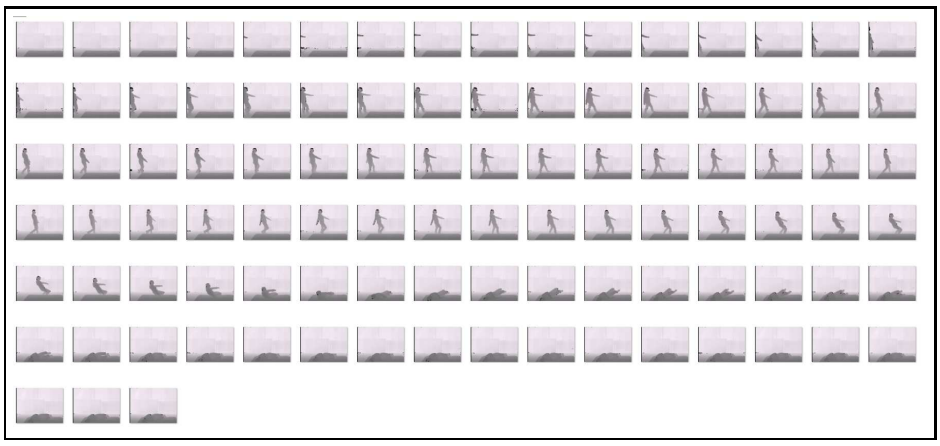

Figure 44. Sample of the fall frames used for training in the the fall versus non-fall classification method.

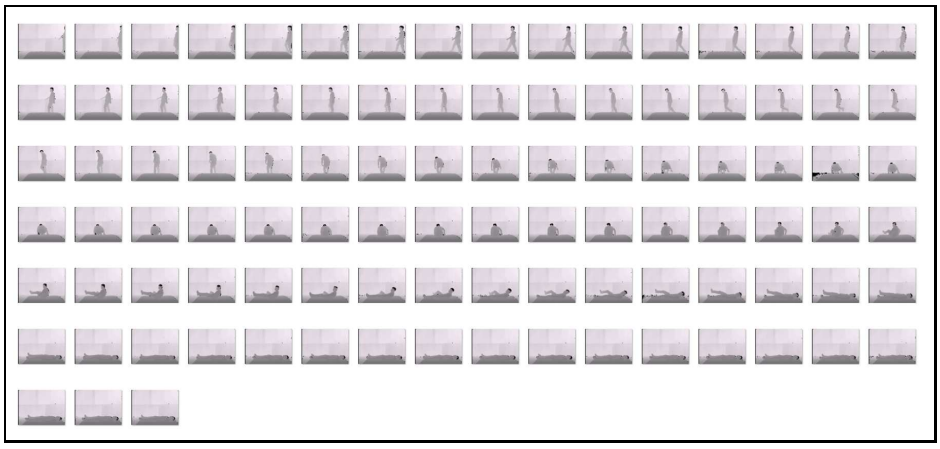

Figure 45. Sample of the non-fall frames used for training in the the fall versus non-fall classification method.

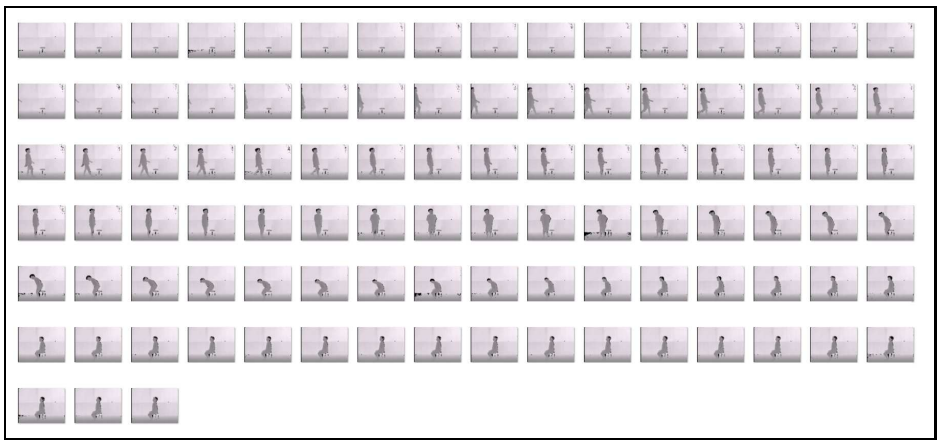

Figure 46. Sample of the non-fall frames used for training in the the fall versus non-fall classification method.

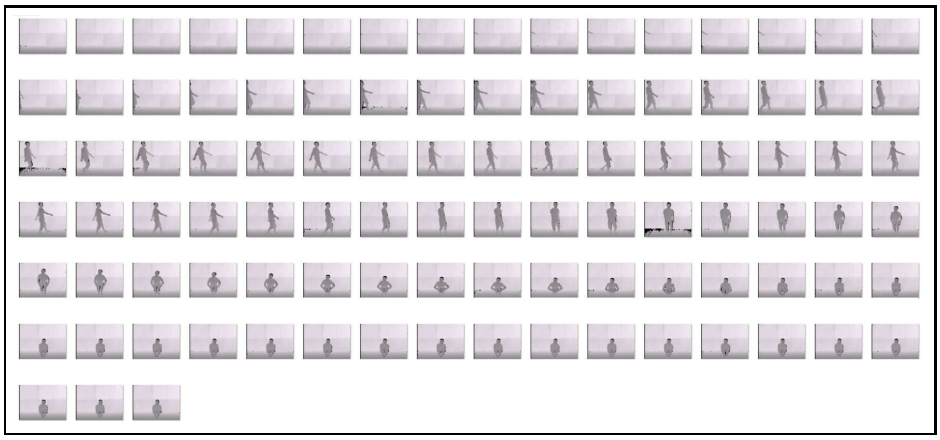

Figure 47. Sample of the non-fall frames used for training in the the fall versus non-fall classification method.

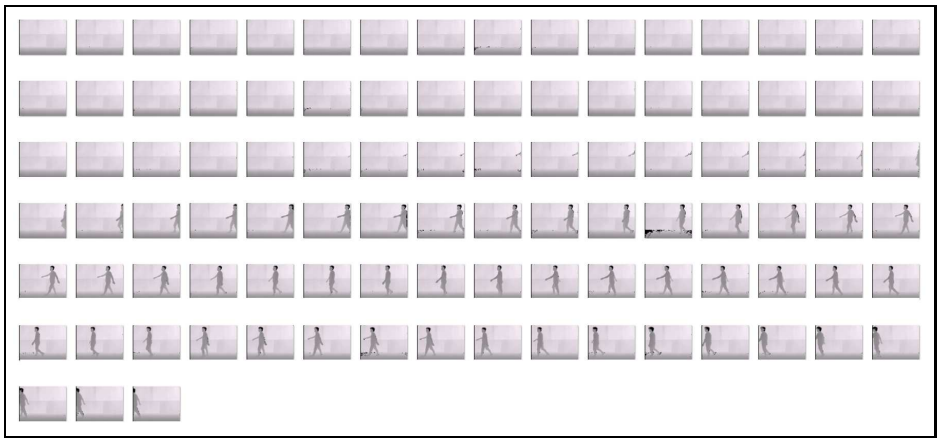

Figure 48. Sample of the non-fall frames used for training in the the fall versus non-fall classification method.

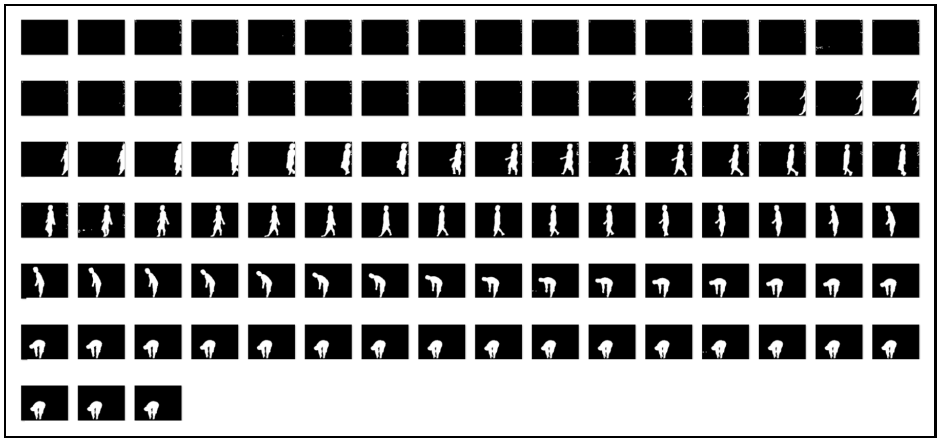

Figure 49. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 1.

Figure 50. Sample of the falling frames used for training in the the fall versus non-fall classification method in experiment 1.

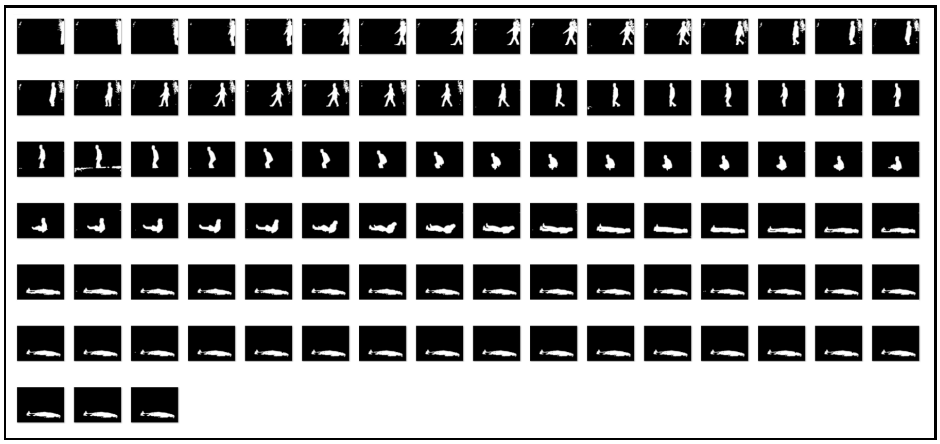

Figure 51. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 1.

Figure 52. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 1.

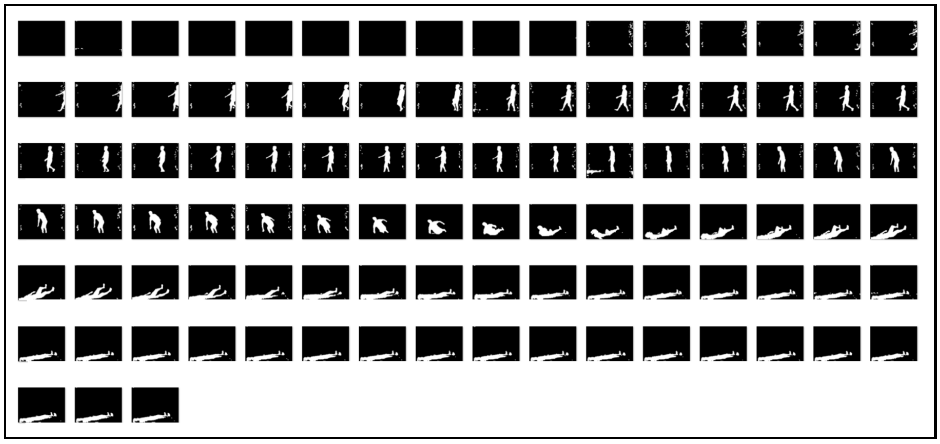

Figure 53. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 1.

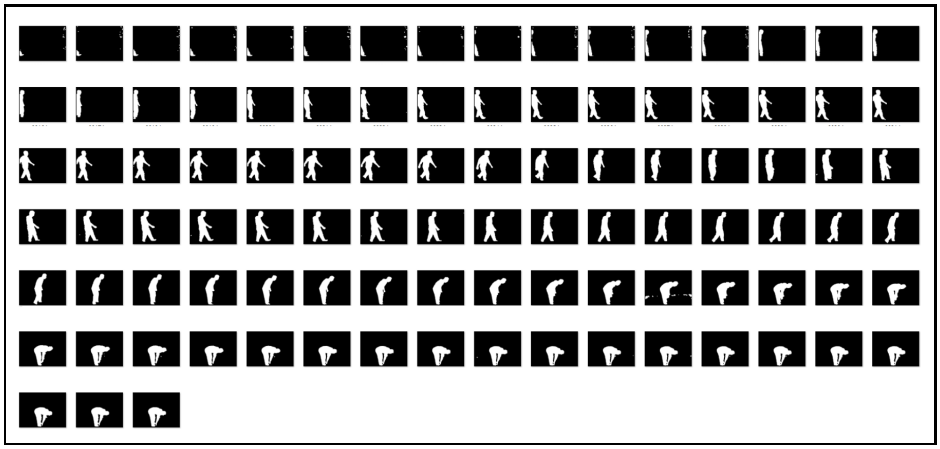

Figure 54. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 2.

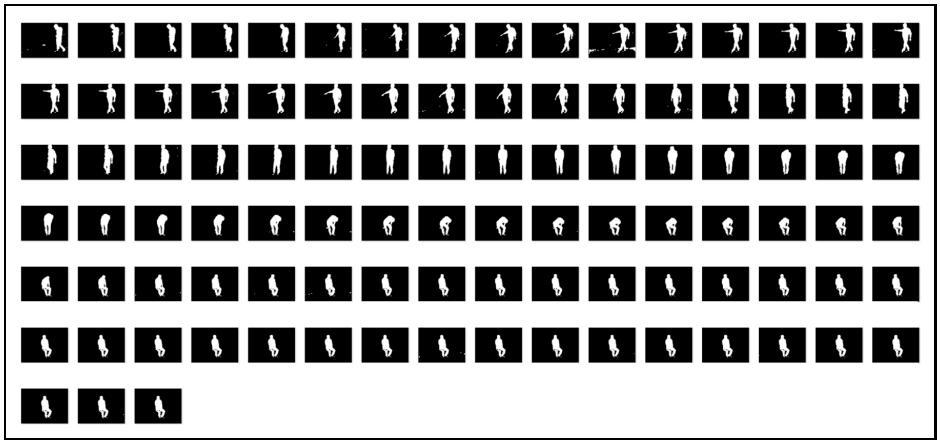

Figure 55. Sample of the falling frames used for training in the the fall versus non-fall classification method in experiment 2.

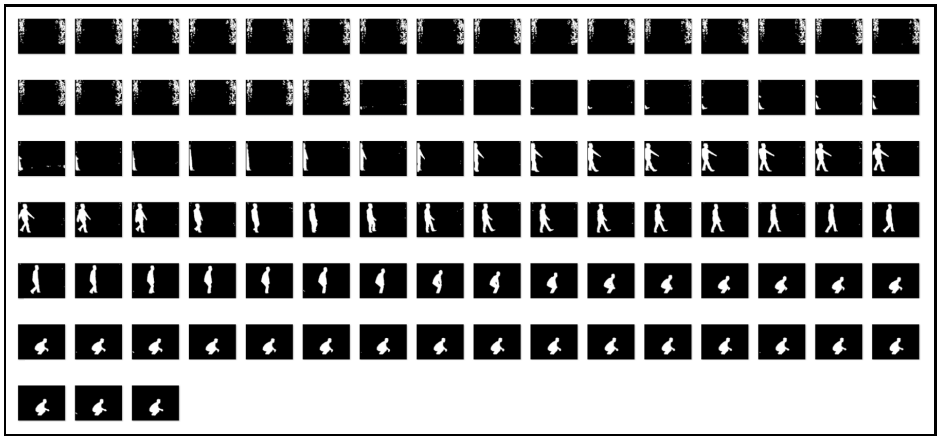

Figure 56. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 2.

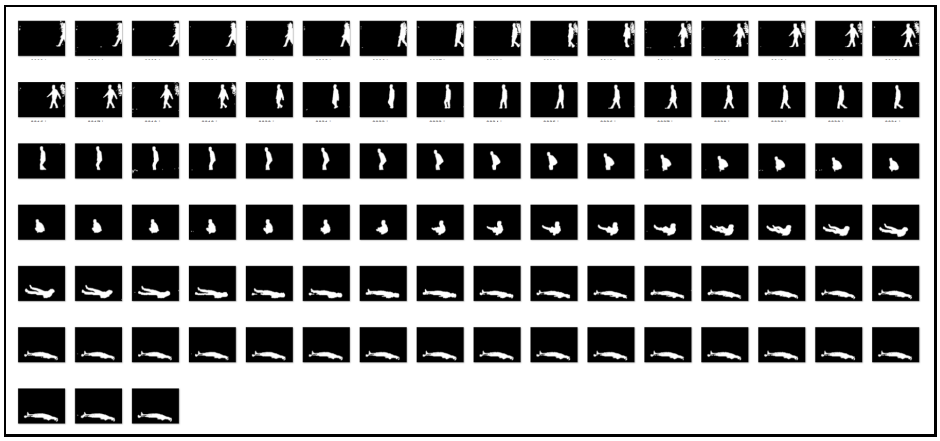

Figure 57. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 2.

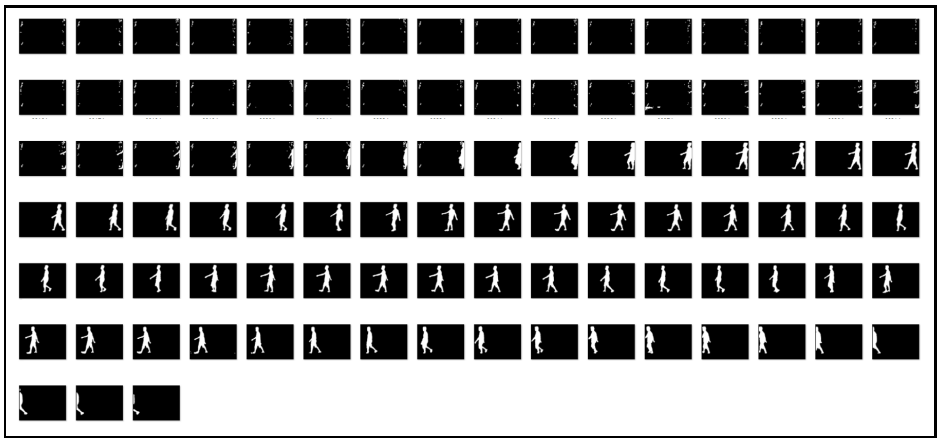

Figure 58. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 2.

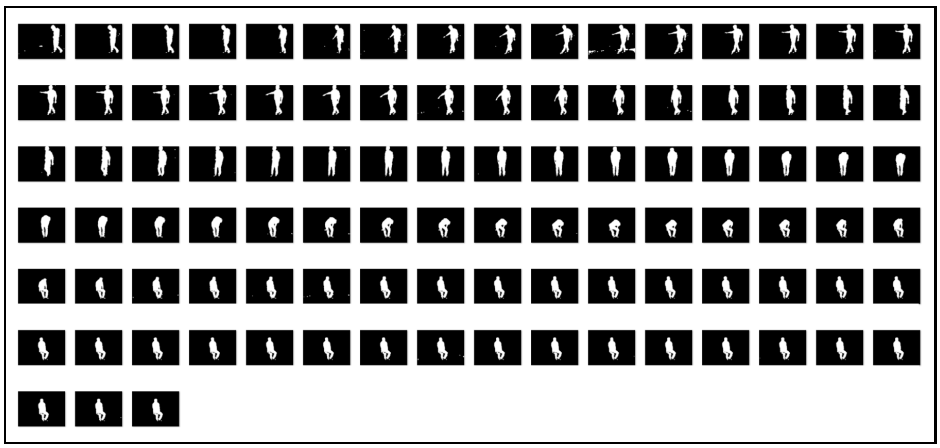

Figure 59. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 2.

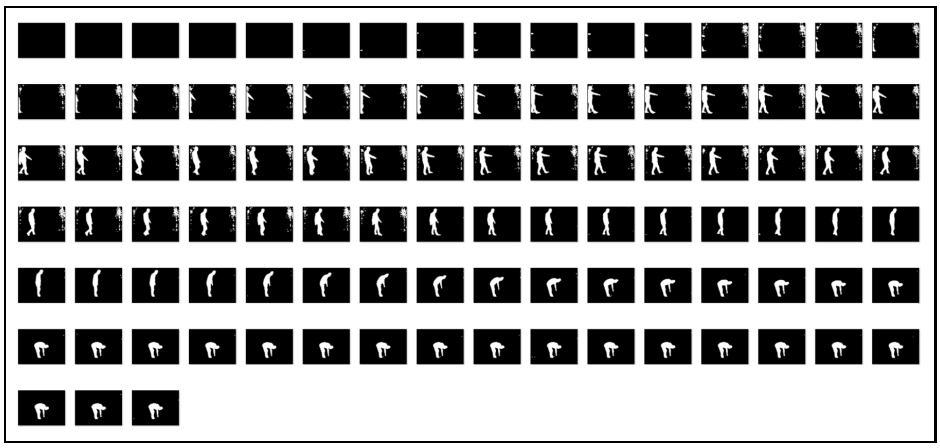

Figure 60. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 3.

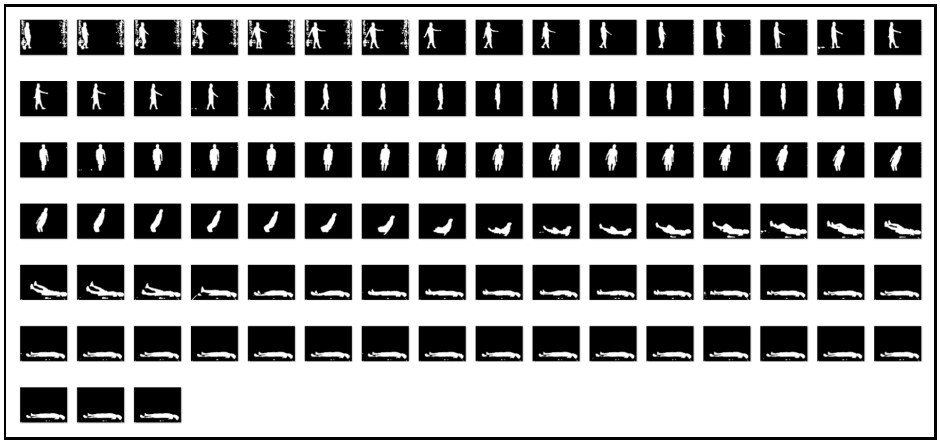

Figure 61. Sample of the falling frames used for training in the the fall versus non-fall classification method in experiment 3.

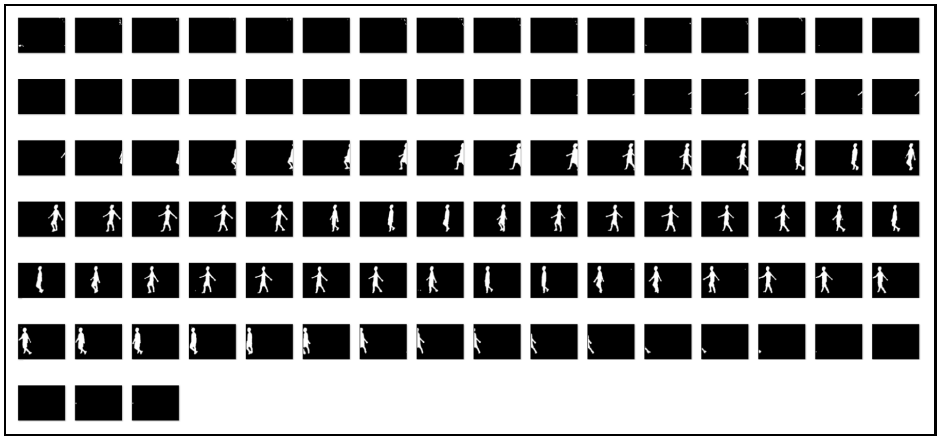

Figure 62. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 3.

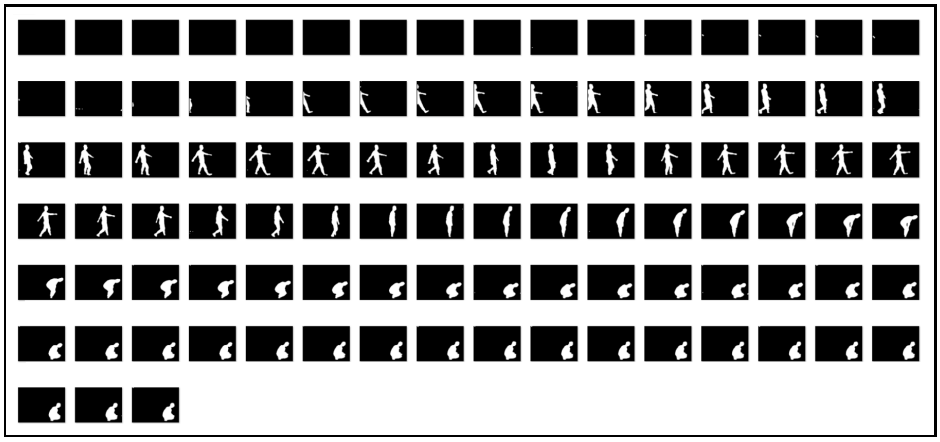

Figure 63. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 3.

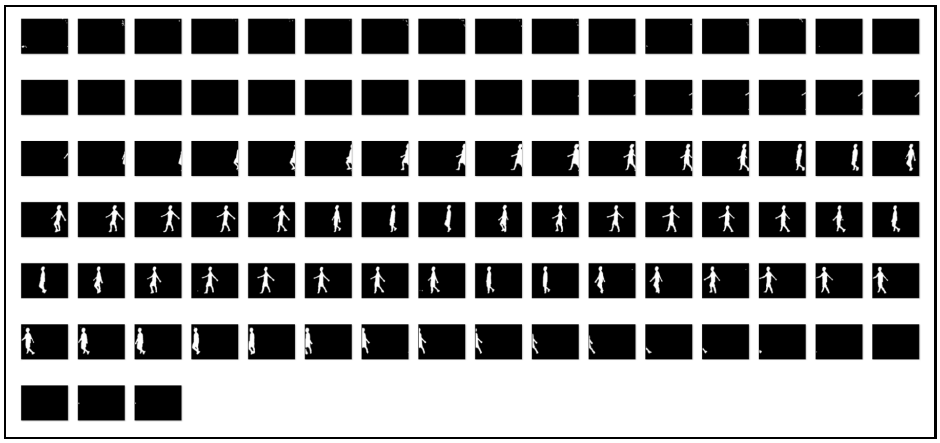

Figure 64. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 3.

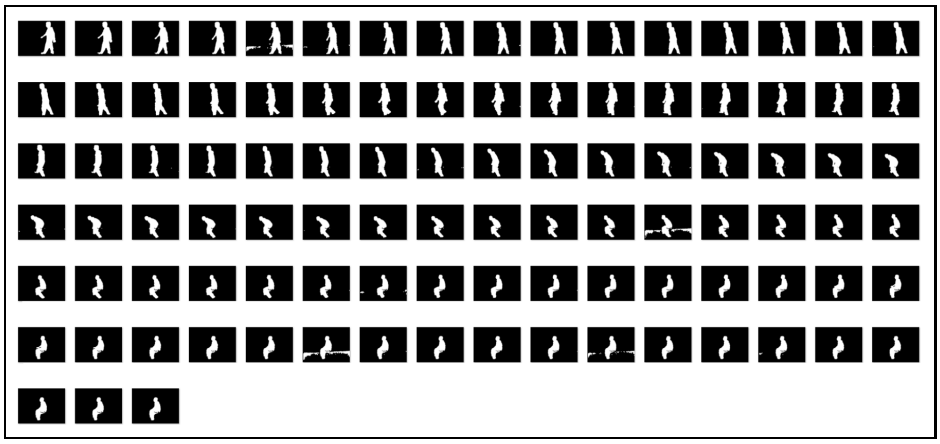

Figure 65. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 3.

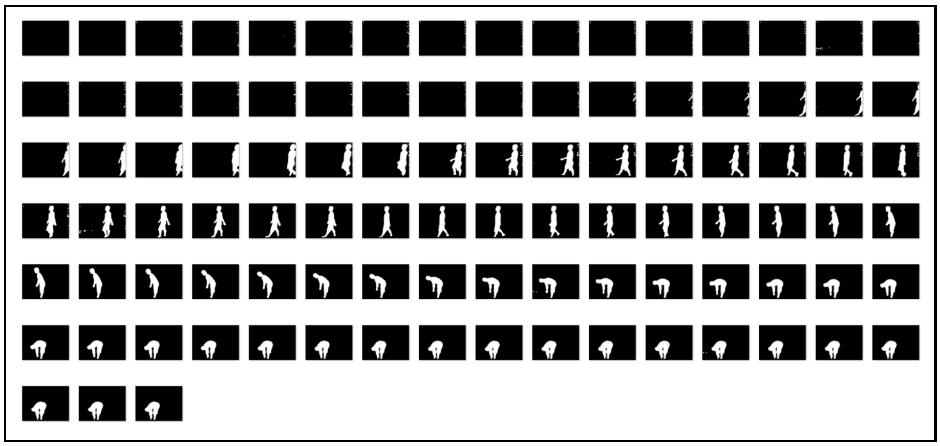

Figure 66. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 4.

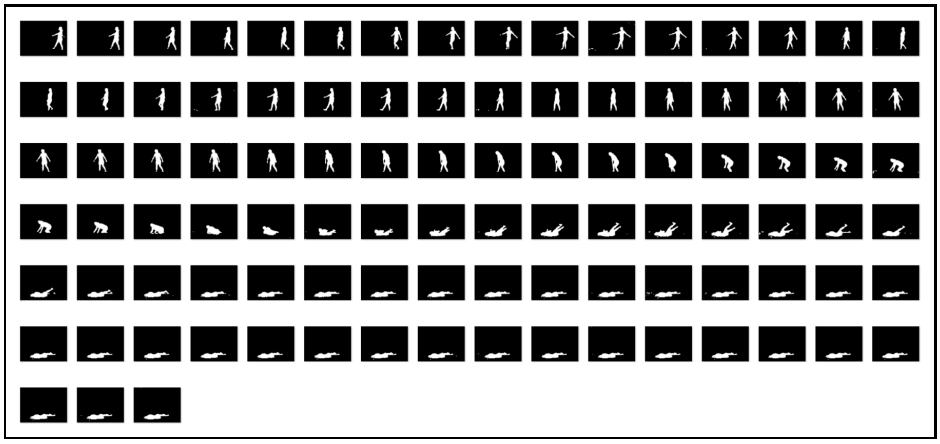

Figure 67. Sample of the falling frames used for training in the the fall versus non-fall classification method in experiment 4.

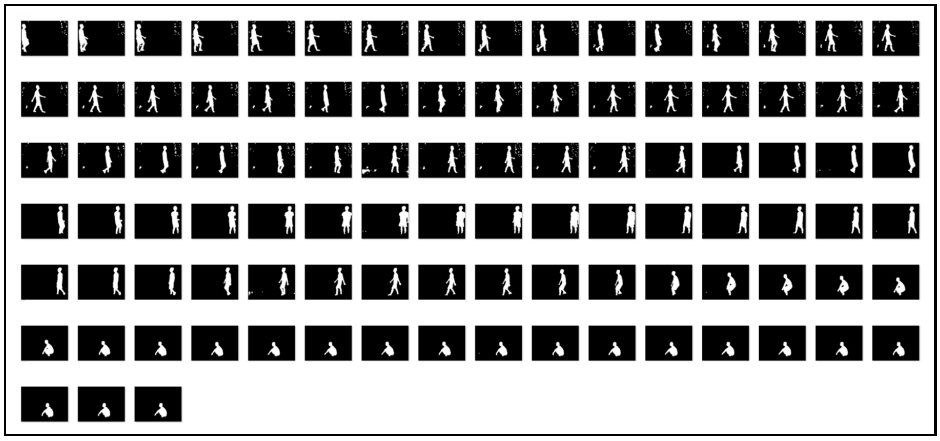

Figure 68. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 4.

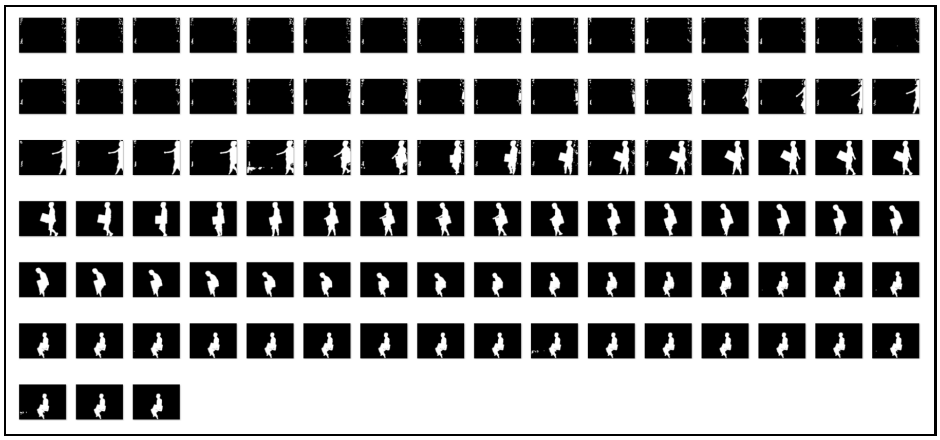

Figure 69. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 4.

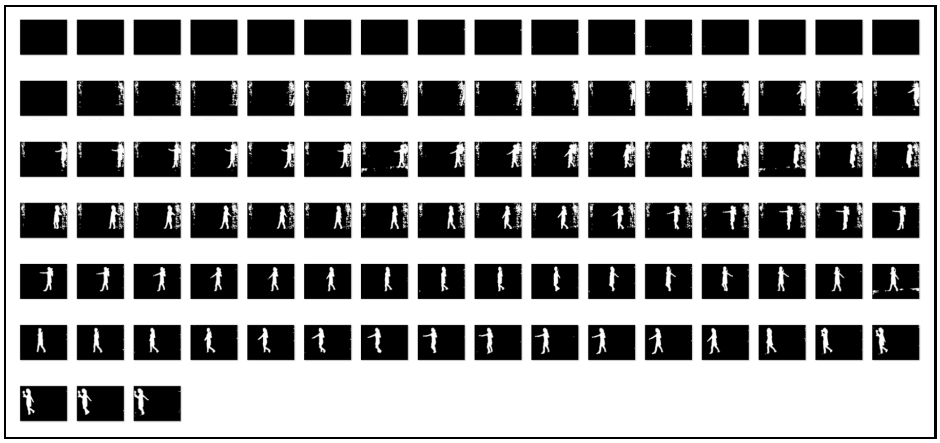

Figure 70. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 4.

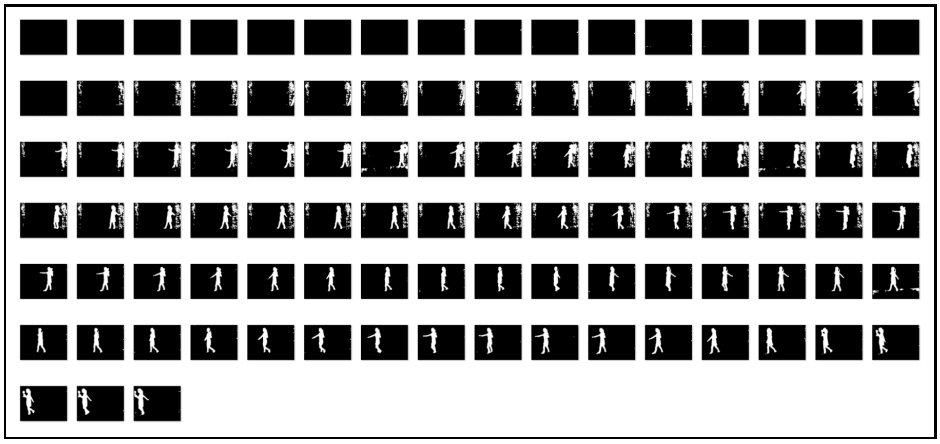

Figure 71. Sample of the non-fall frames used for training in the the fall versus non-fall classification method in experiment 4.

Appendix B

CORRECTLY CLASSIFIED VIDEO FRAMES

In the following sections we show samples of the correctly classified video frames using the two classification approaches followed in this research; depth video classification, and binary video classification.

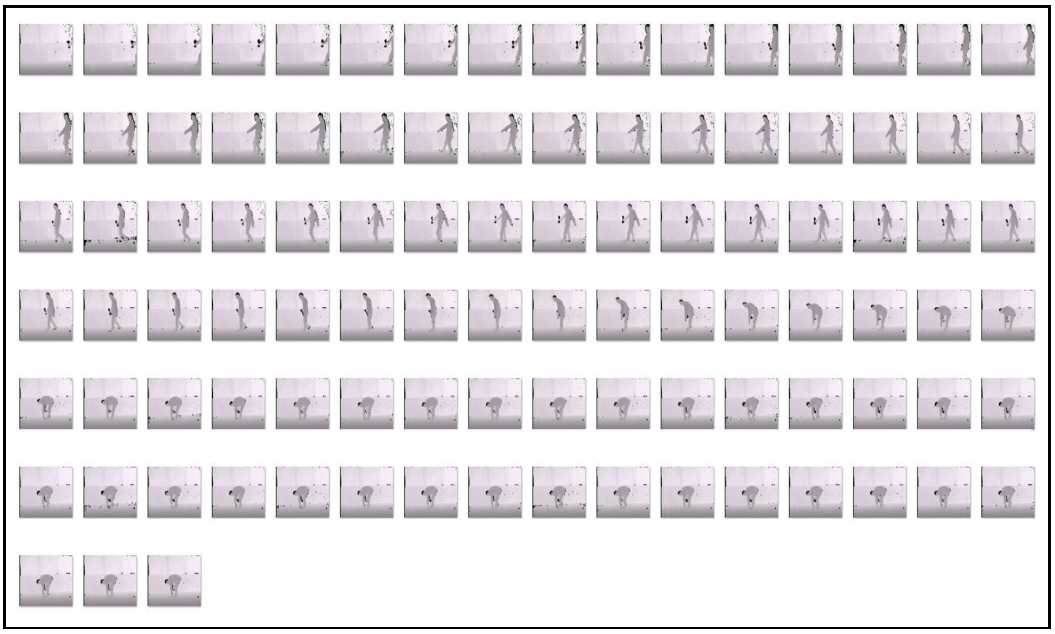

Figure 72. Sample of a correctly classified bending video frames using the six-class classification method.

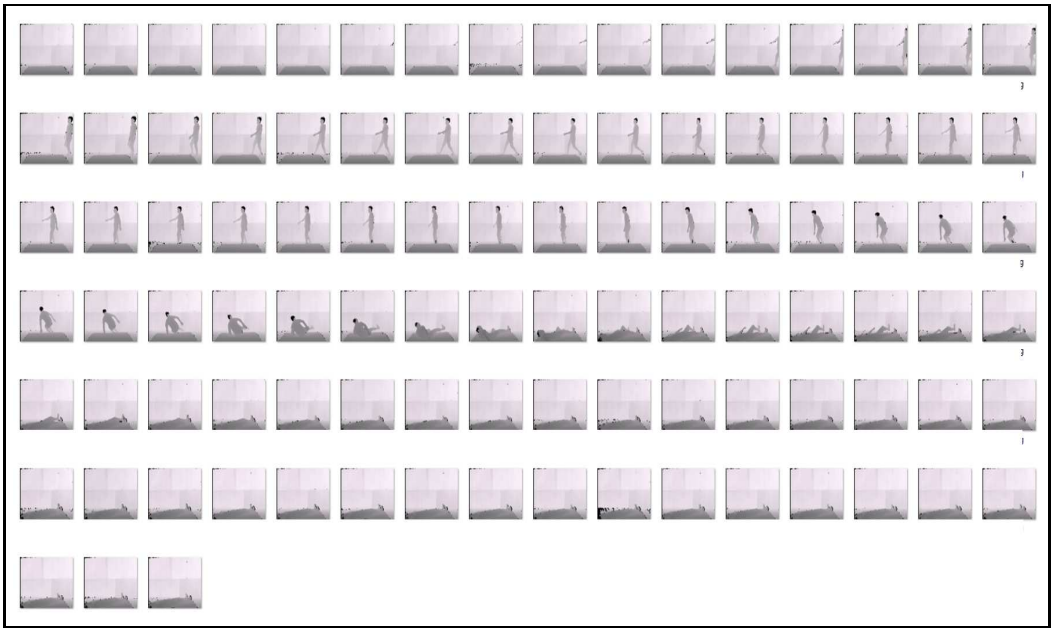

Figure 73. Sample of a correctly classified falling video frames using the six-class classification method.

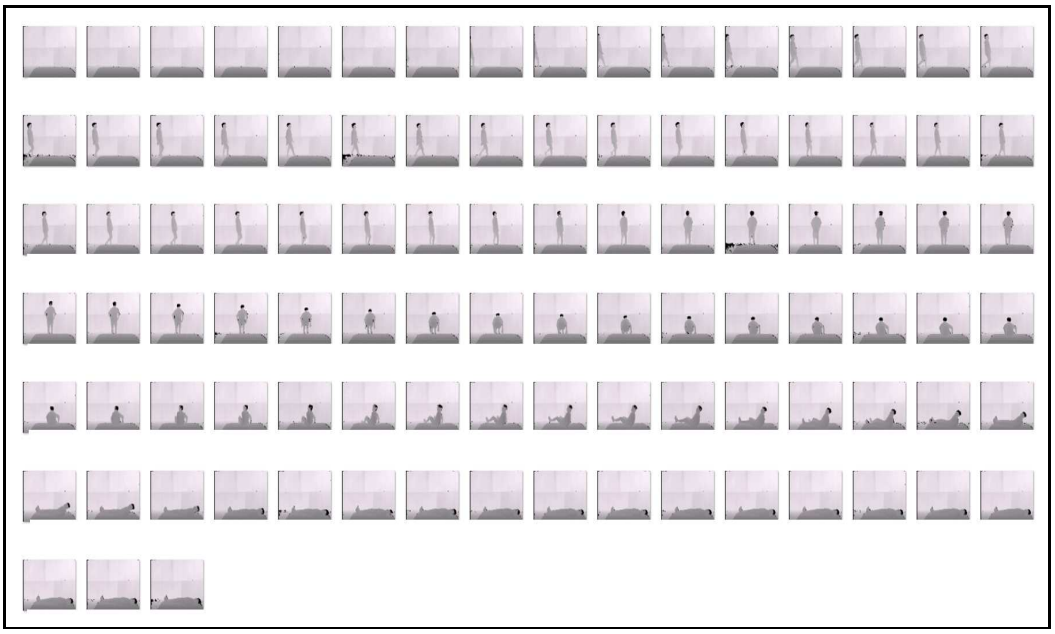

Figure 74. Sample of a correctly classified lying-down video frames using the six-class classification method.

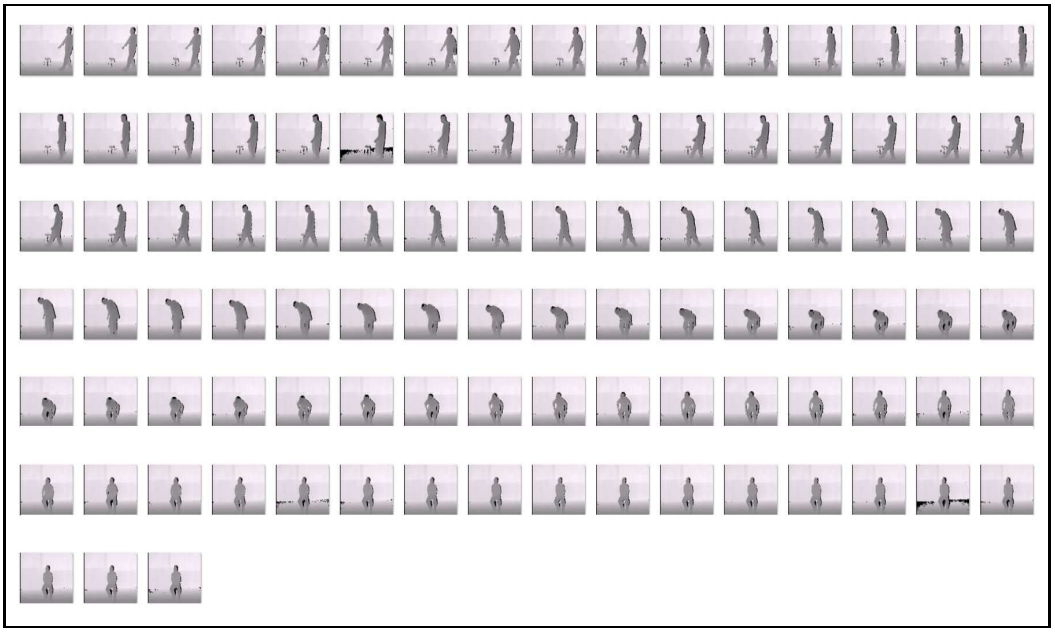

Figure 75. Sample of a correctly classified sitting video frames using the six-class classification method.

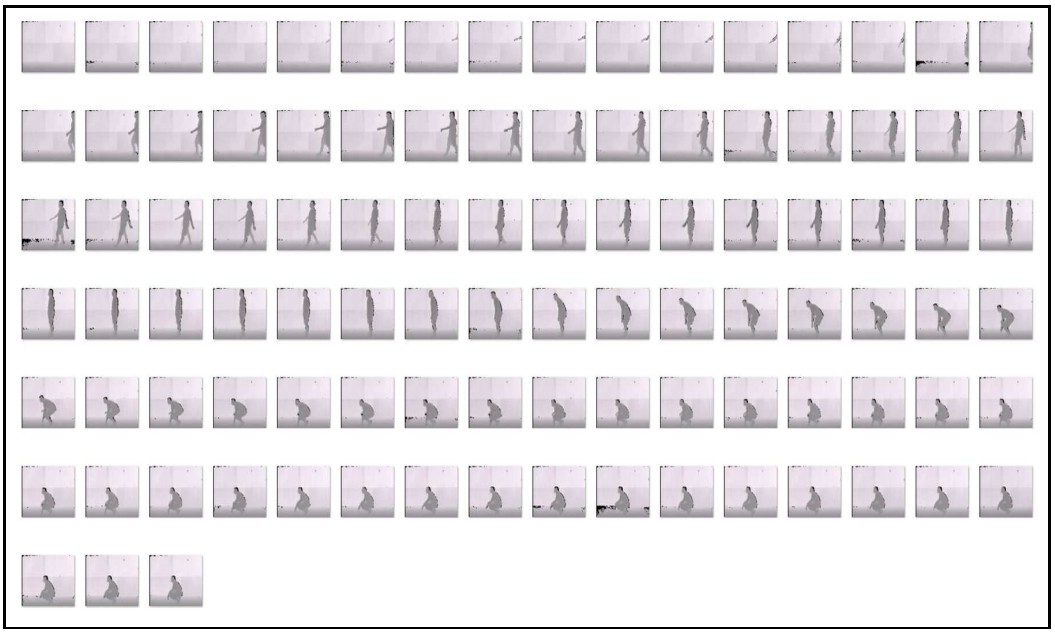

Figure 76. Sample of a correctly classified squatting video frames using the six-class classification method.

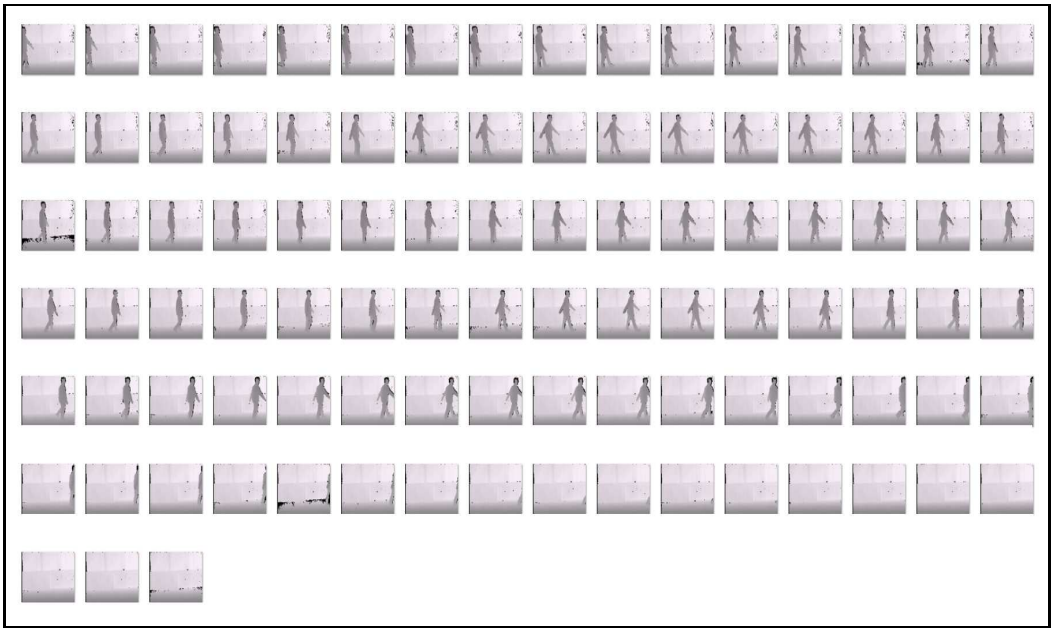

Figure 77. Sample of a correctly classified walking video frames using the six-class classification method.

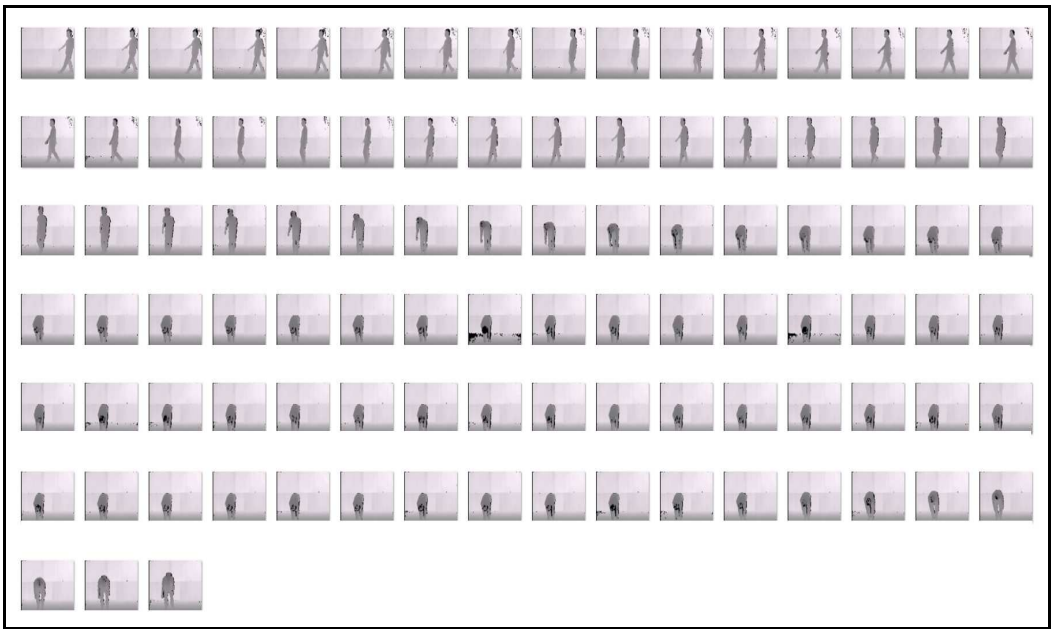

Figure 78. Sample of a correctly classified non-fall video frames using the fall versus non-fall classification method.

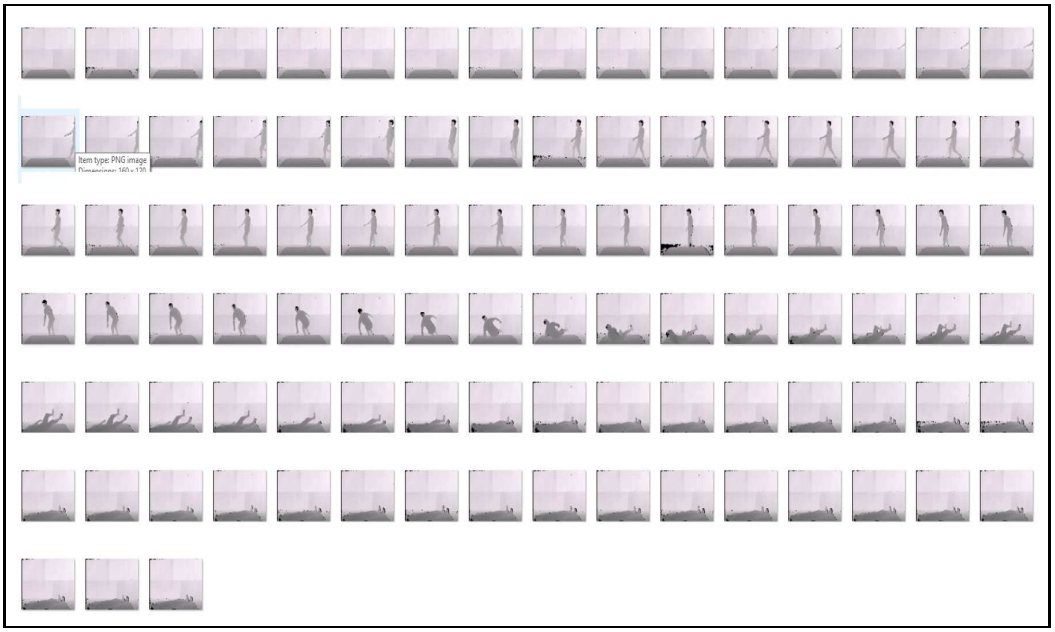

Figure 79. Sample of a correctly classified falling video frames using the fall versus non-fall classification method.

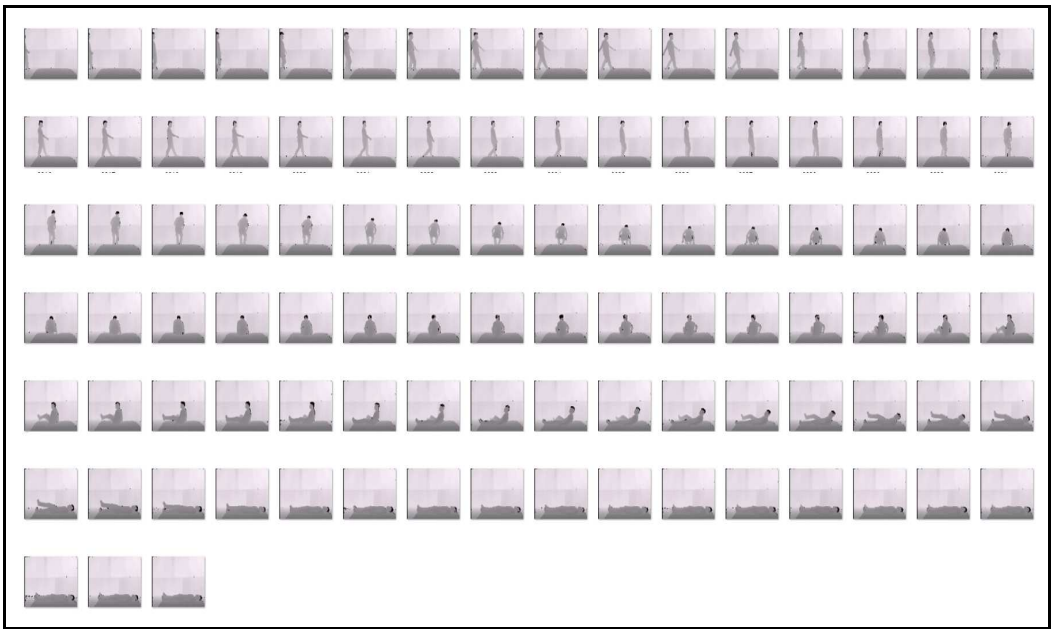

Figure 80. Sample of a correctly classified non-fall video frames using the fall versus non-fall classification method.

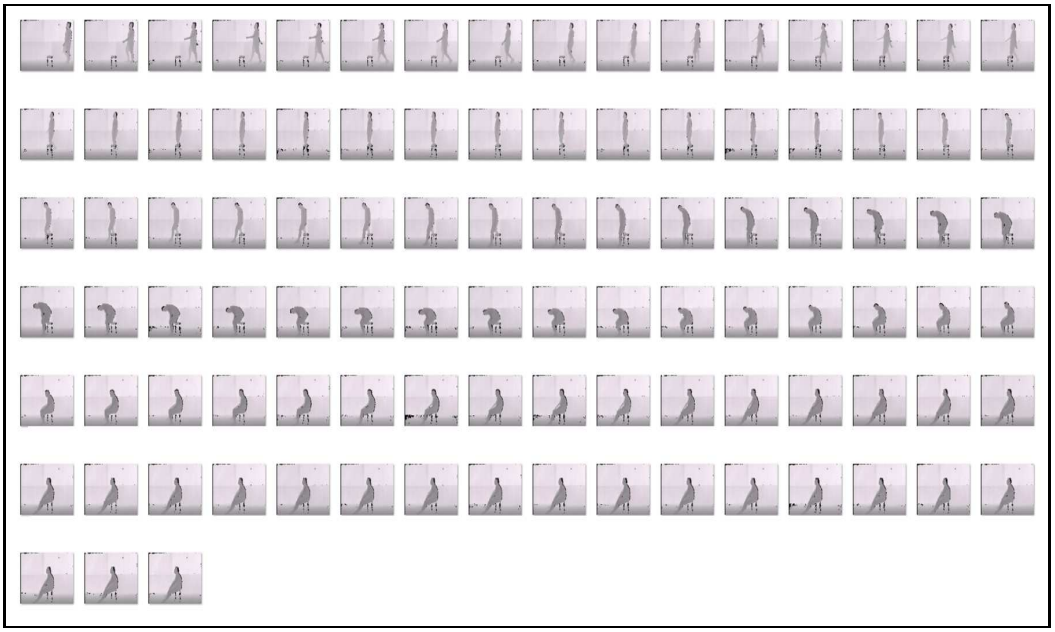

Figure 81. Sample of a correctly classified non-fall video frames using the fall versus non-fall classification method.

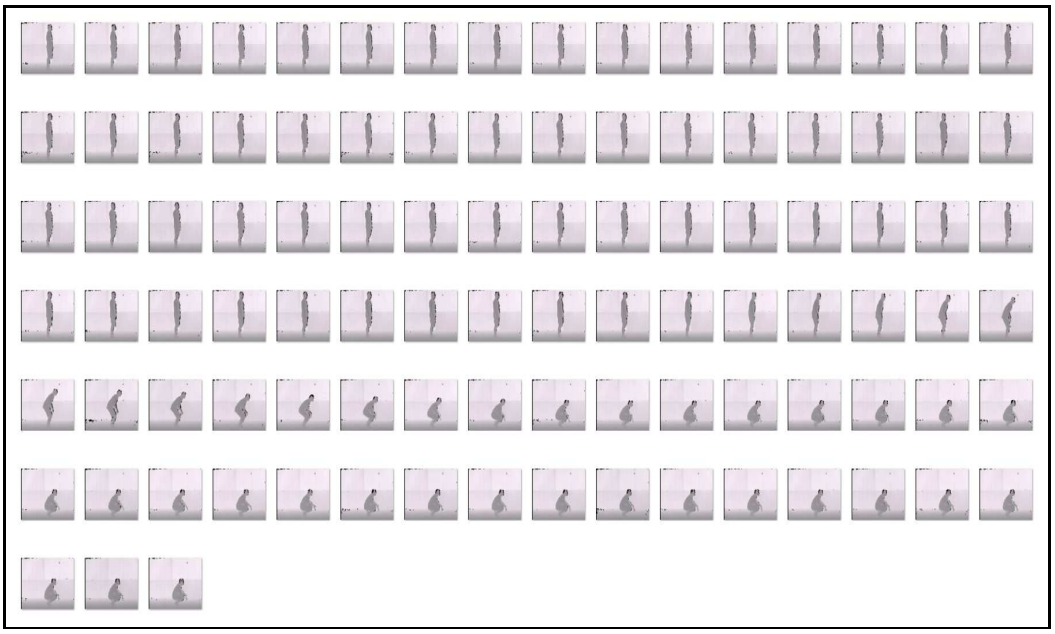

Figure 82. Sample of a correctly classified non-fall video frames using the fall versus non-fall classification method.

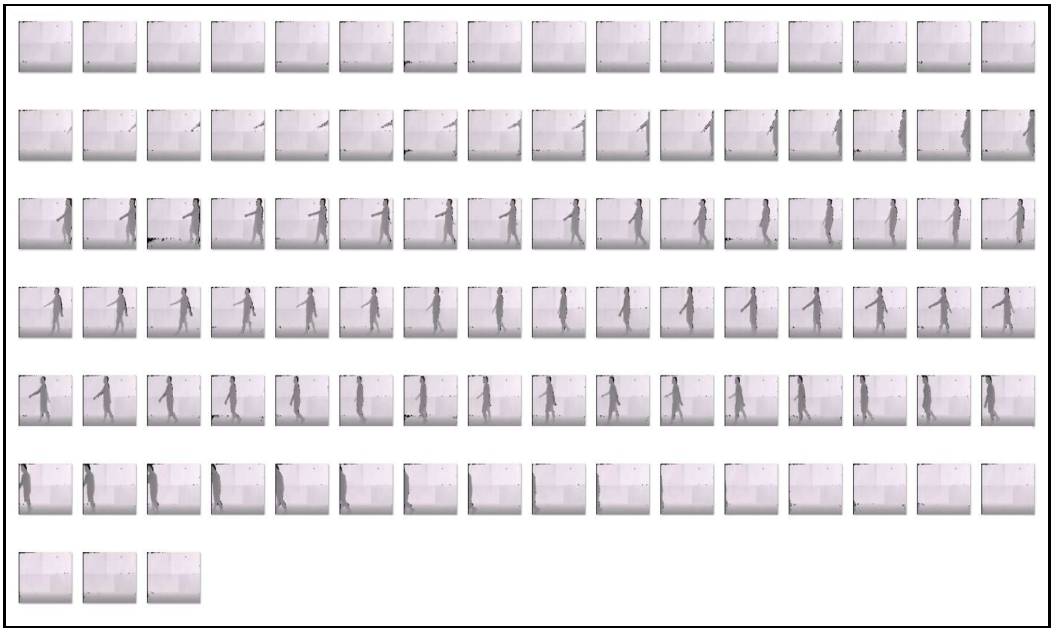

Figure 83. Sample of a correctly classified non-fall video frames using the fall versus non-fall classification method.

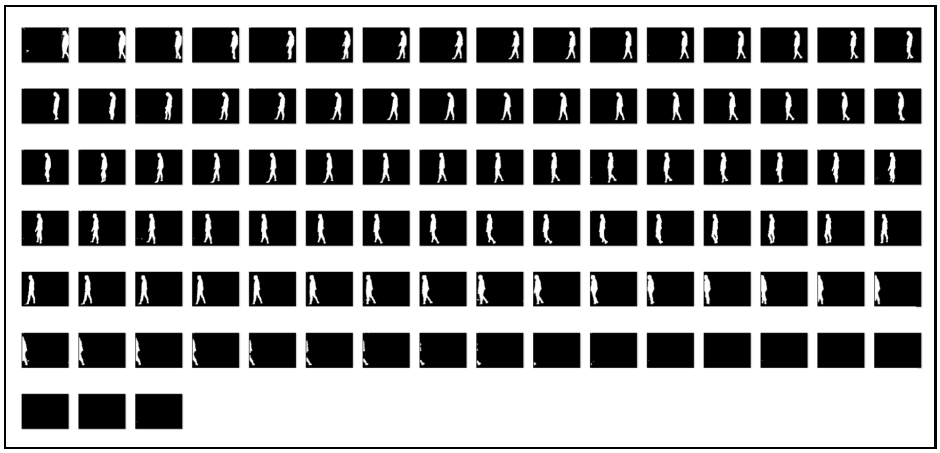

Figure 84. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 1.

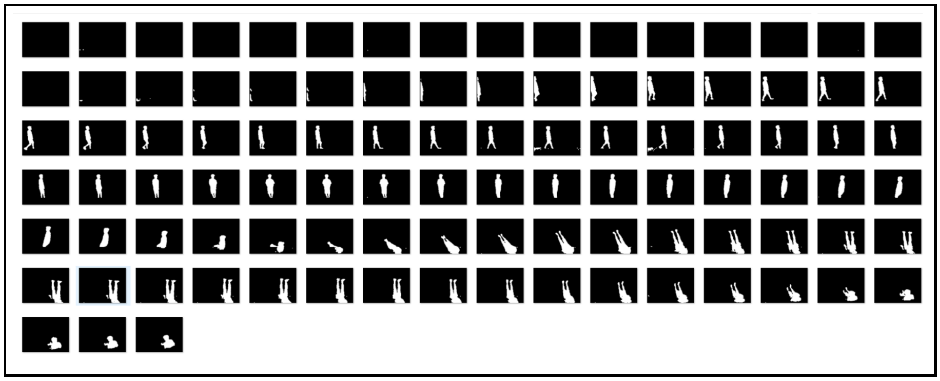

Figure 85. Sample of the correctly classified falling video frames for the fall versus non-fall classification method in experiment 1.

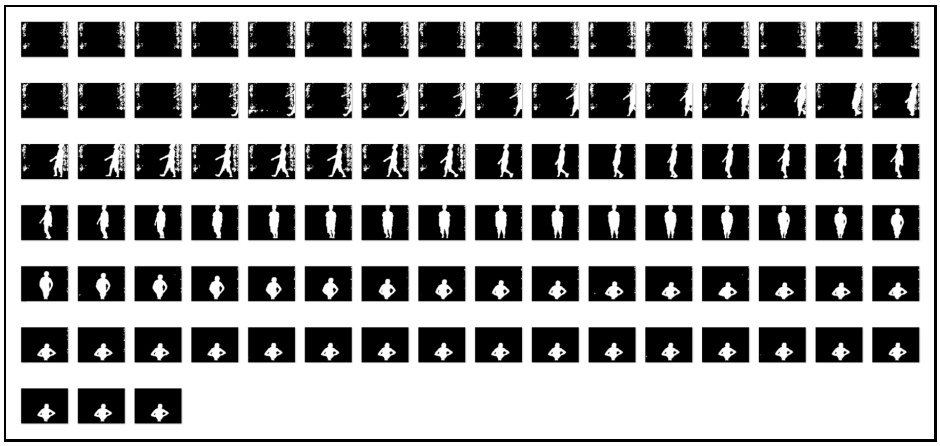

Figure 86. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 1.

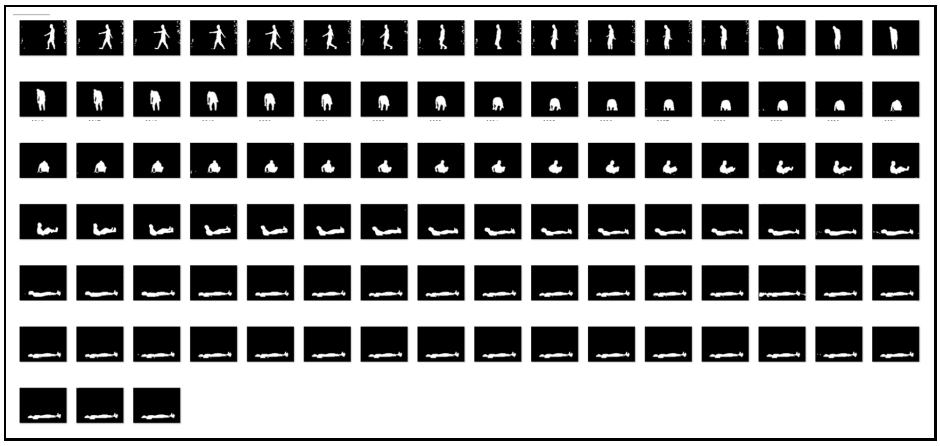

Figure 87. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 1.

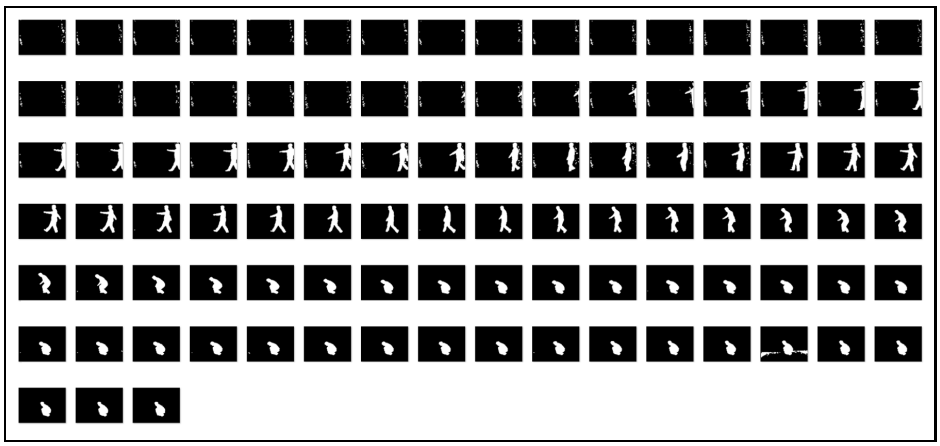

Figure 88. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 1.

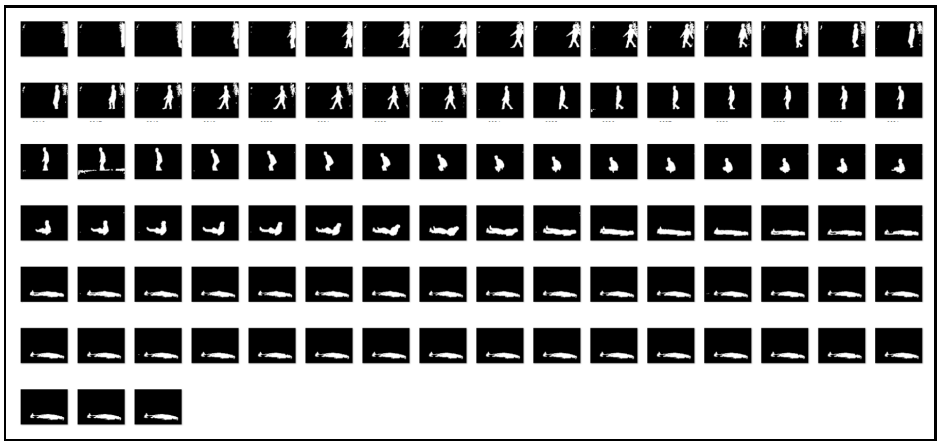

Figure 89. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 2.

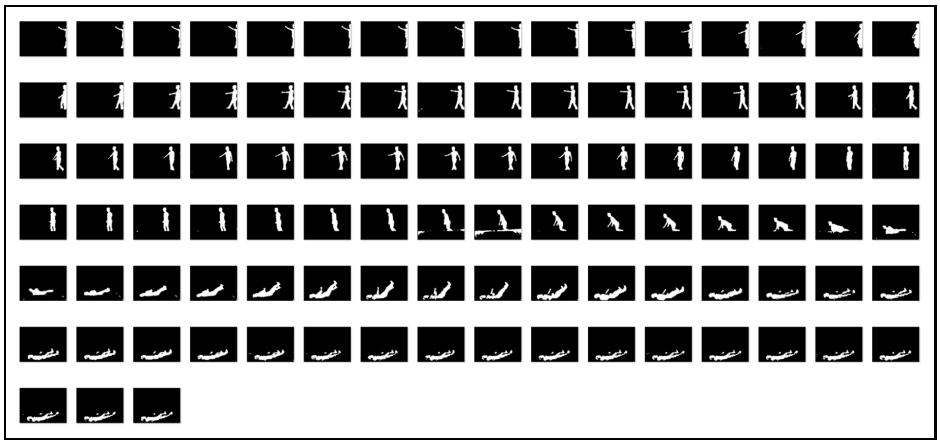

Figure 90. Sample of the correctly classified falling video frames for the fall versus non-fall classification method in experiment 2.

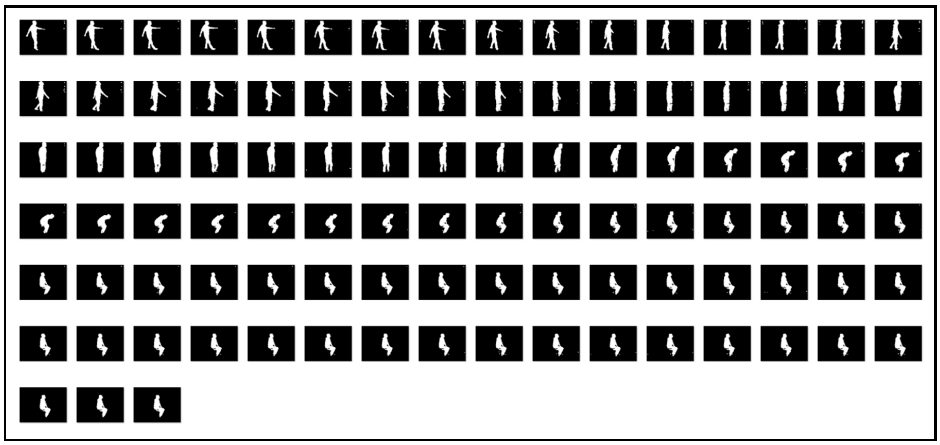

Figure 91. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 2.

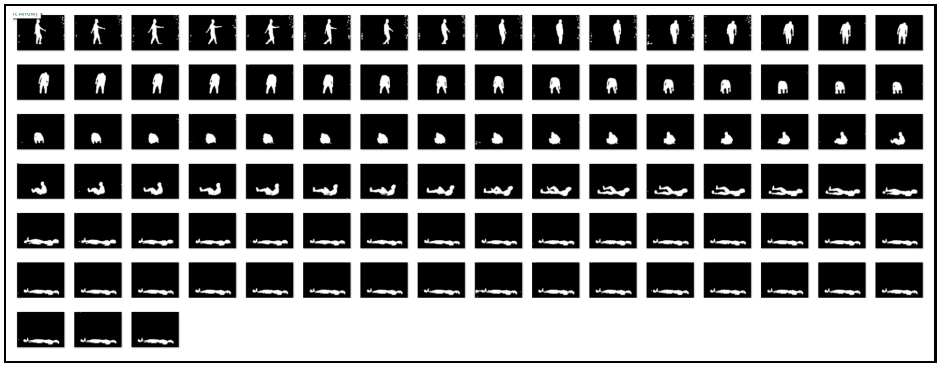

Figure 92. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 2.

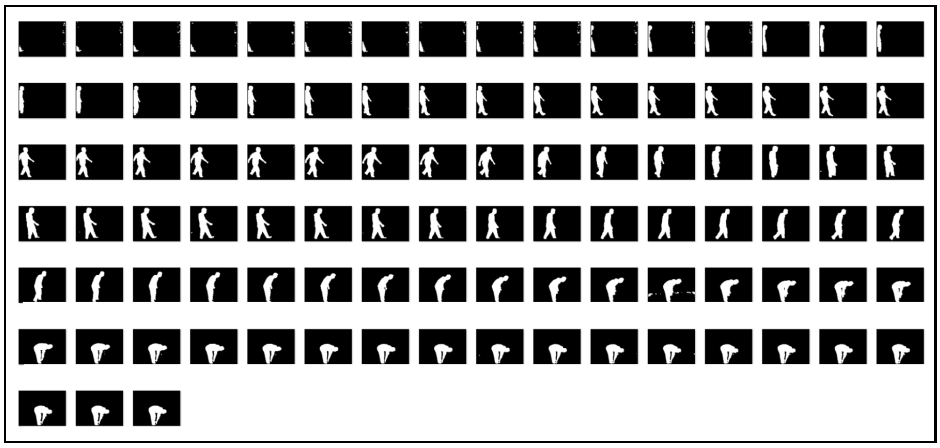

Figure 93. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

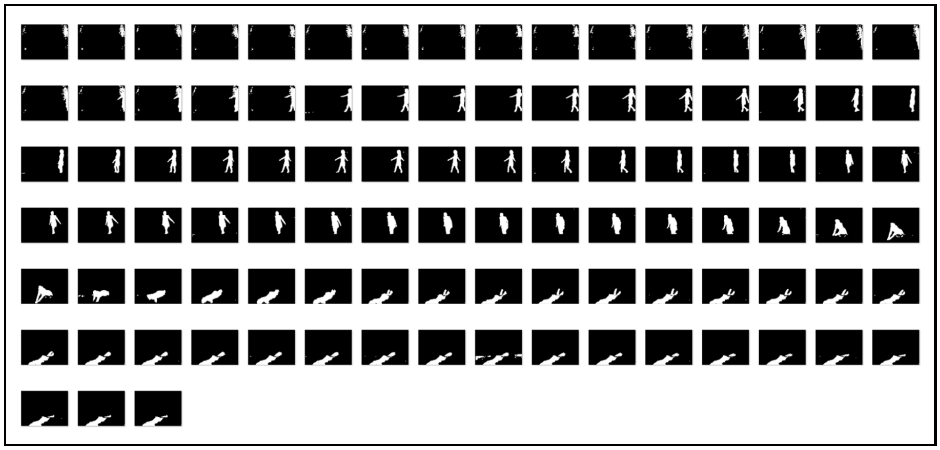

Figure 94. Sample of the correctly classified falling video frames for the fall versus non-fall classification method in experiment 3.

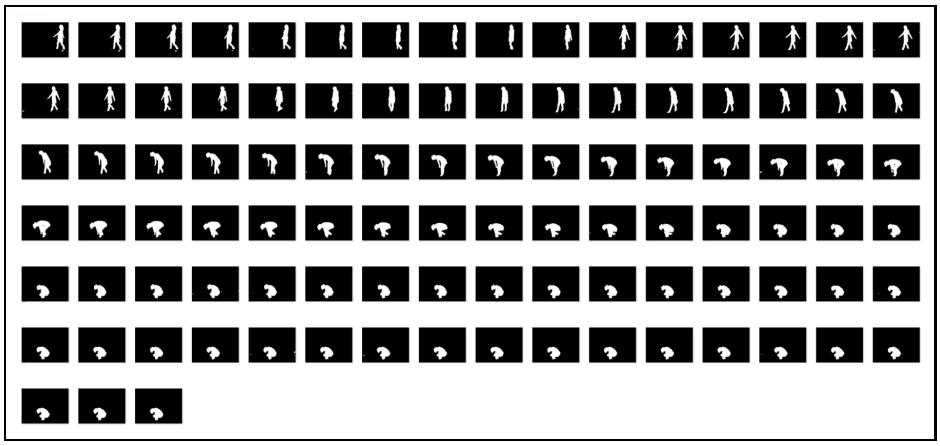

Figure 95. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

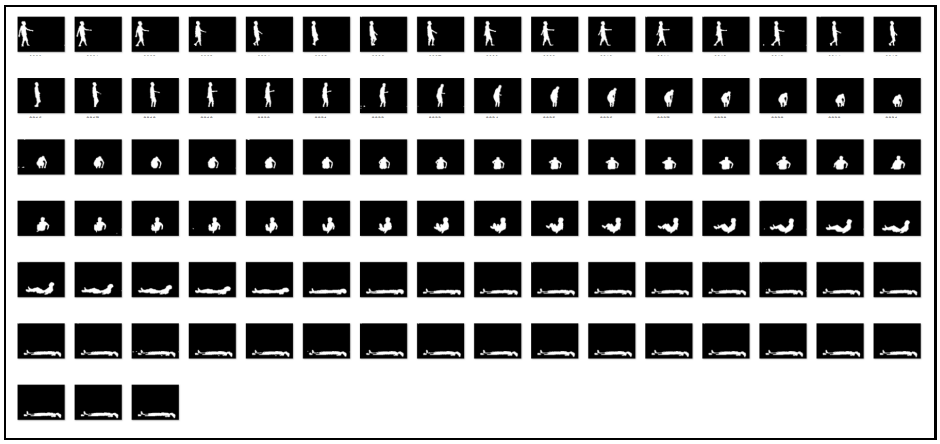

Figure 96. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

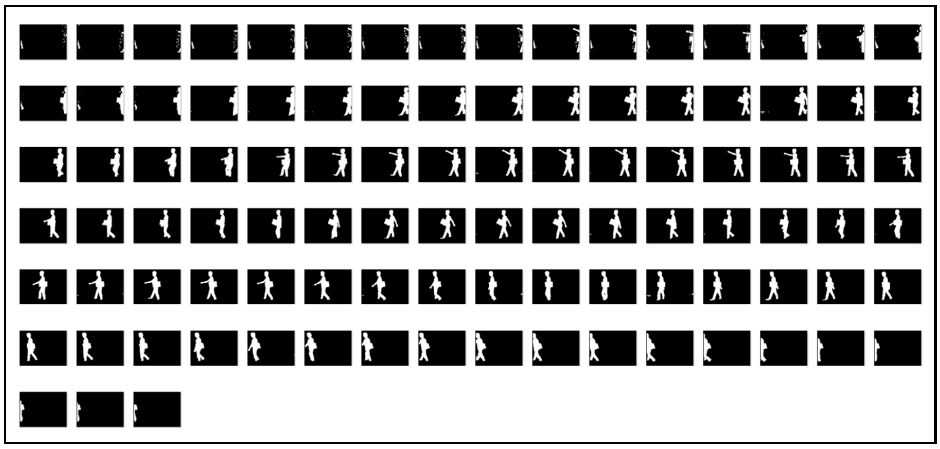

Figure 97. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

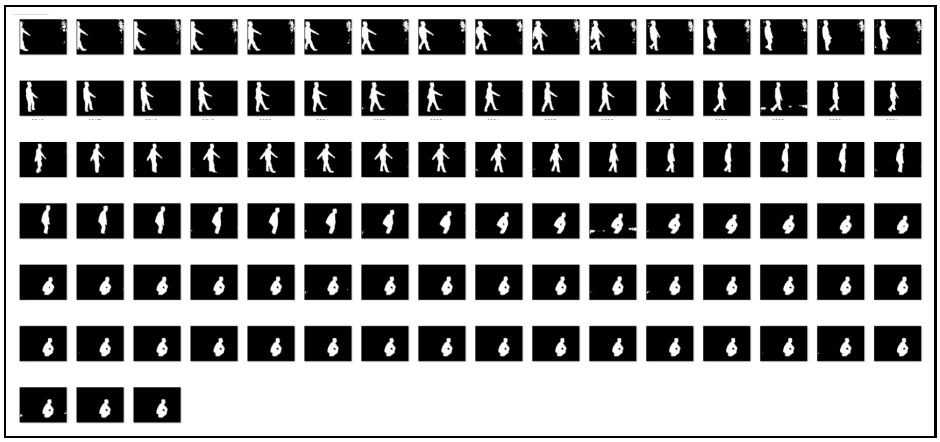

Figure 98. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

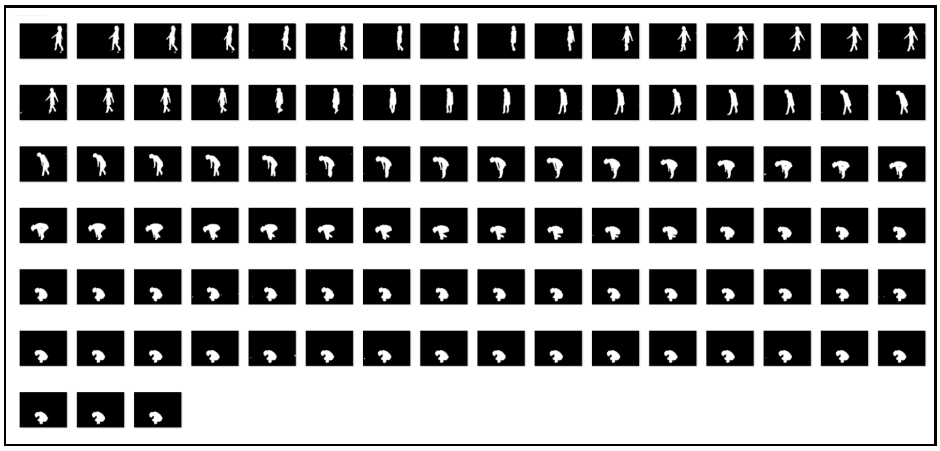

Figure 99. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 3.

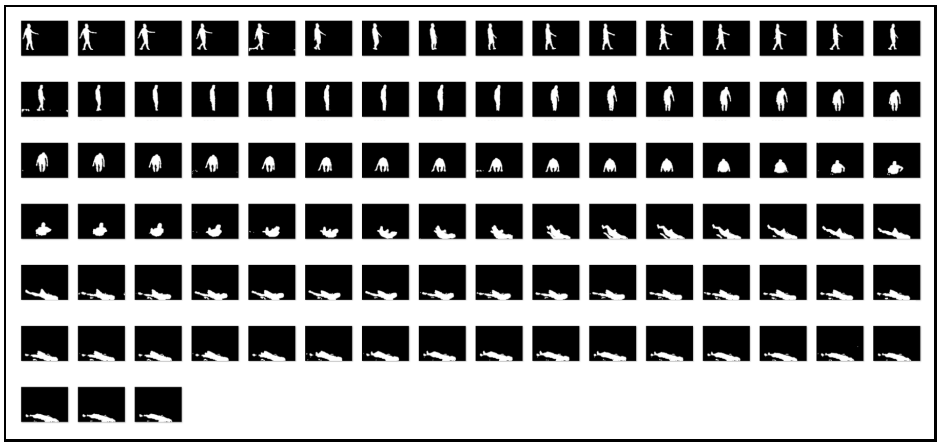

Figure 100. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 4.

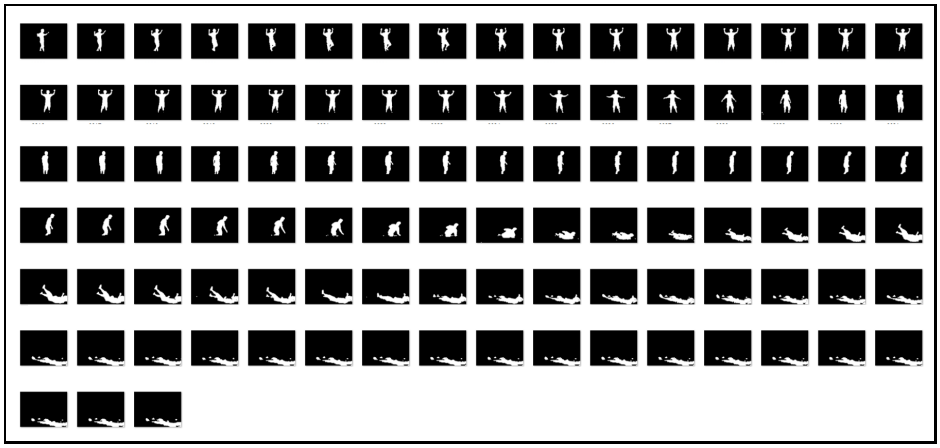

Figure 101. Sample of the correctly classified falling video frames for the fall versus non-fall classification method in experiment 4.

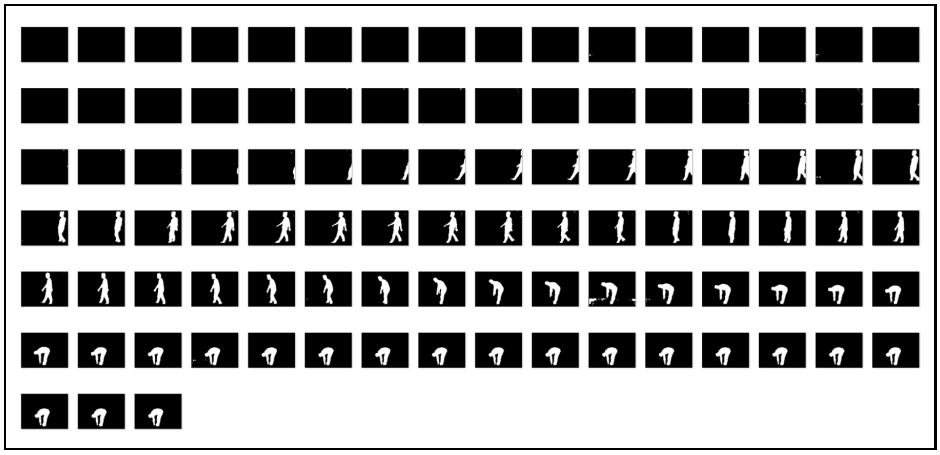

Figure 102. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 4.

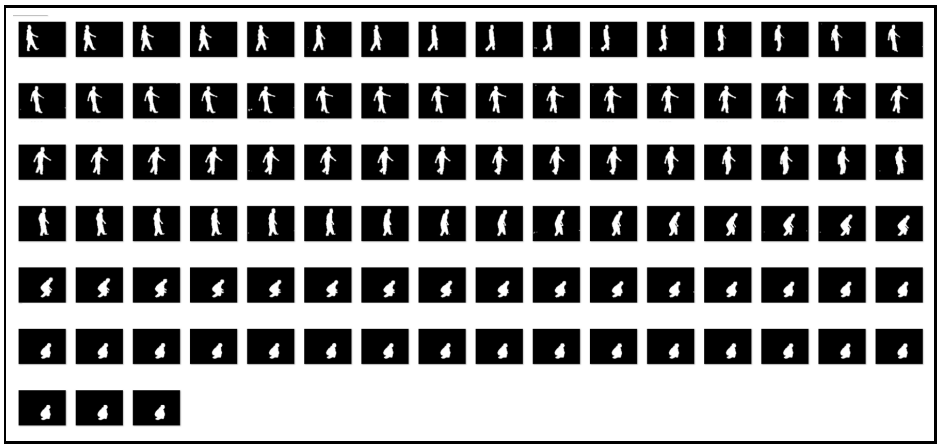

Figure 103. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 4.

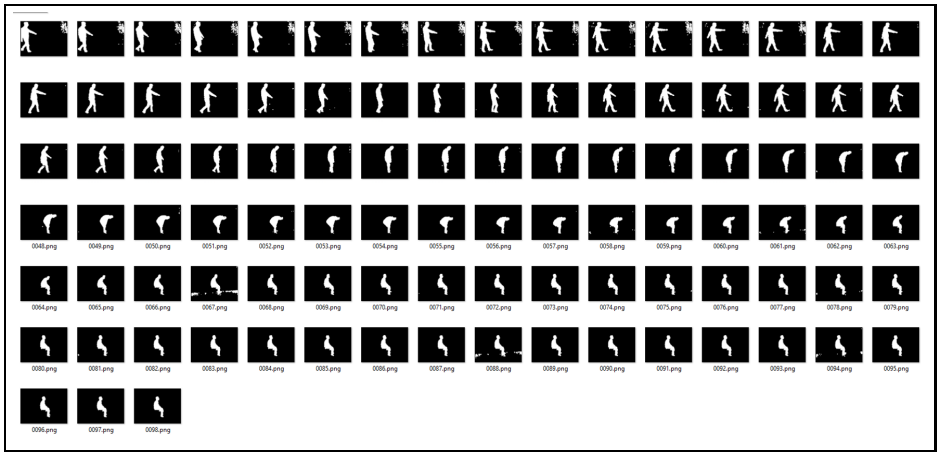

Figure 104. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 4.

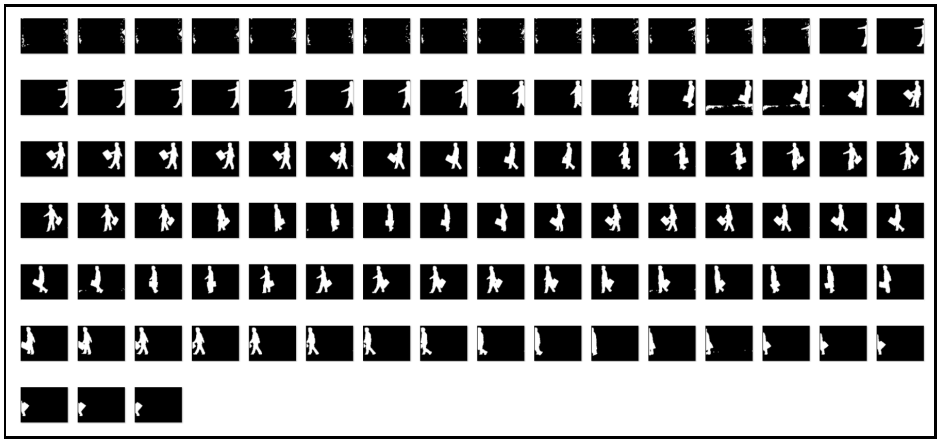

Figure 105. Sample of the correctly classified non-fall video frames for the fall versus non-fall classification method in experiment 4.

Appendix C

MISS CLASSIFIED VIDEO FRAMES

In the following sections we show samples of the miss-classified video frames using the two classification approaches followed in this research; depth video classification, and binary video classification.

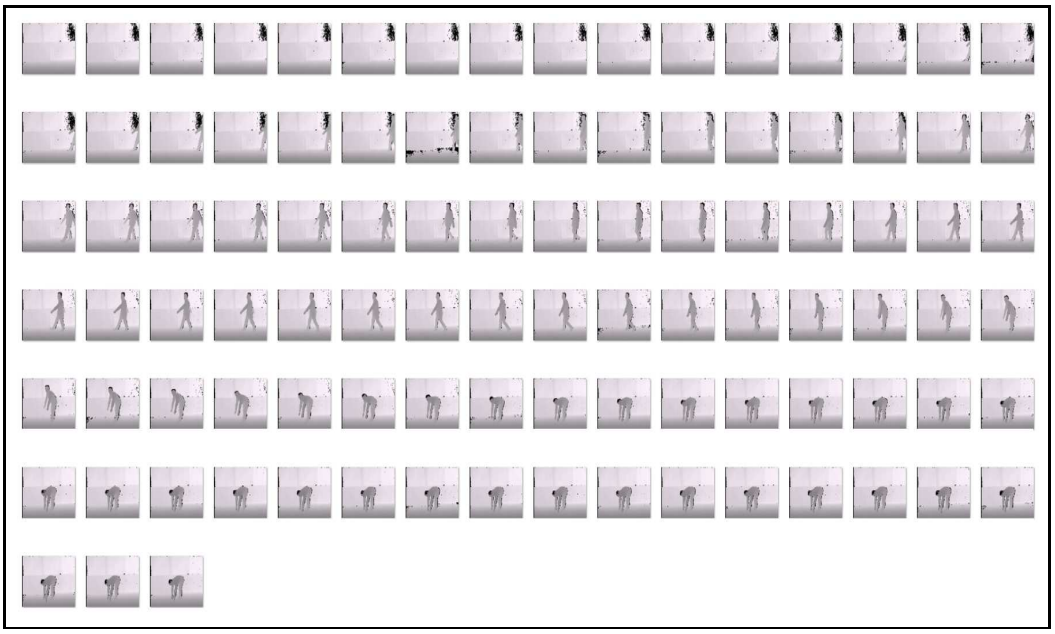

Figure 106. Sample of a miss-classified bending video frames using the six-class classification method.

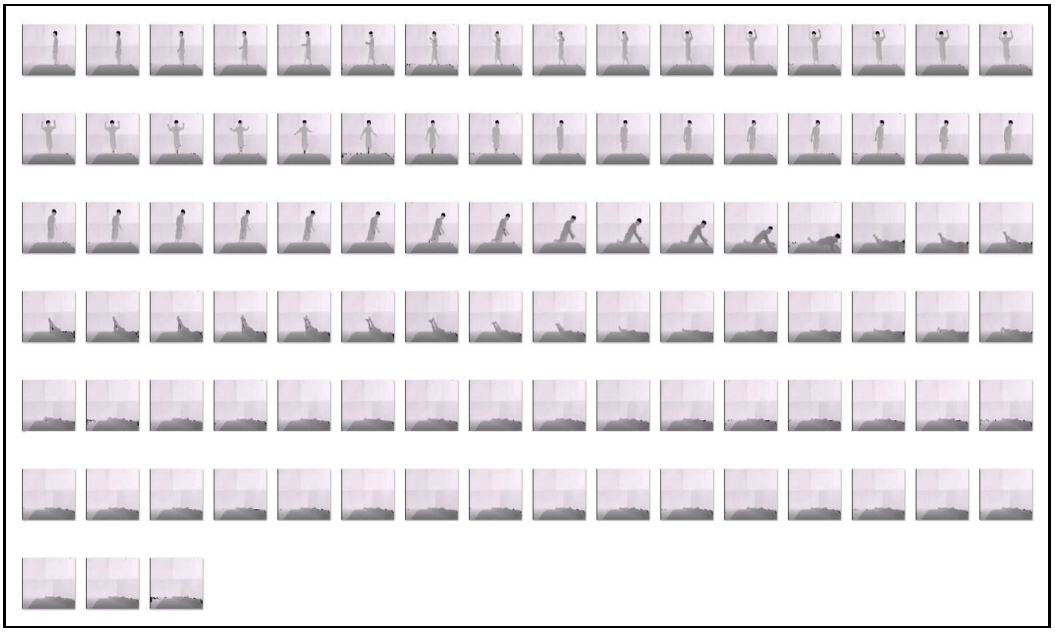

Figure 107. Sample of a miss-classified falling video frames using the six-class classification method.

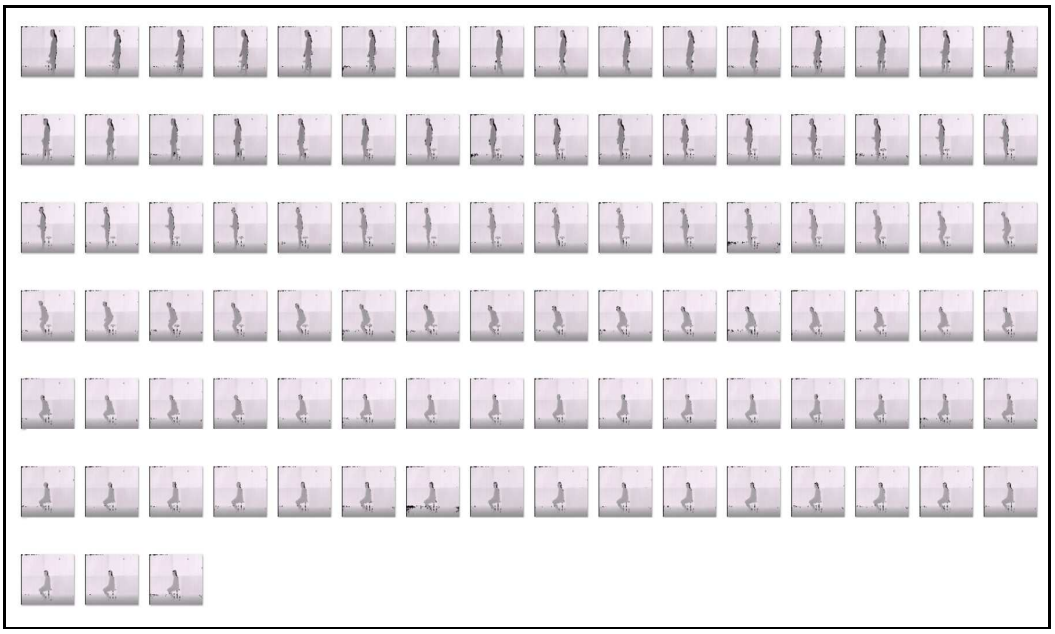

Figure 108. Sample of a miss-classified sitting video frames using the six-class classification method.

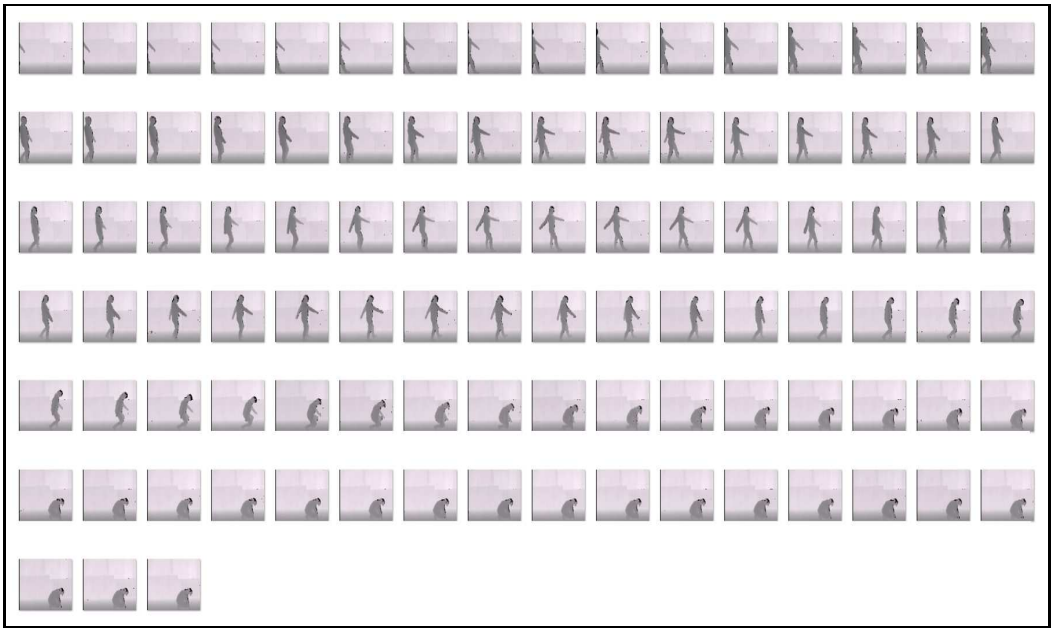

Figure 109. Sample of a miss-classified squatting video frames using the six-class classification method.

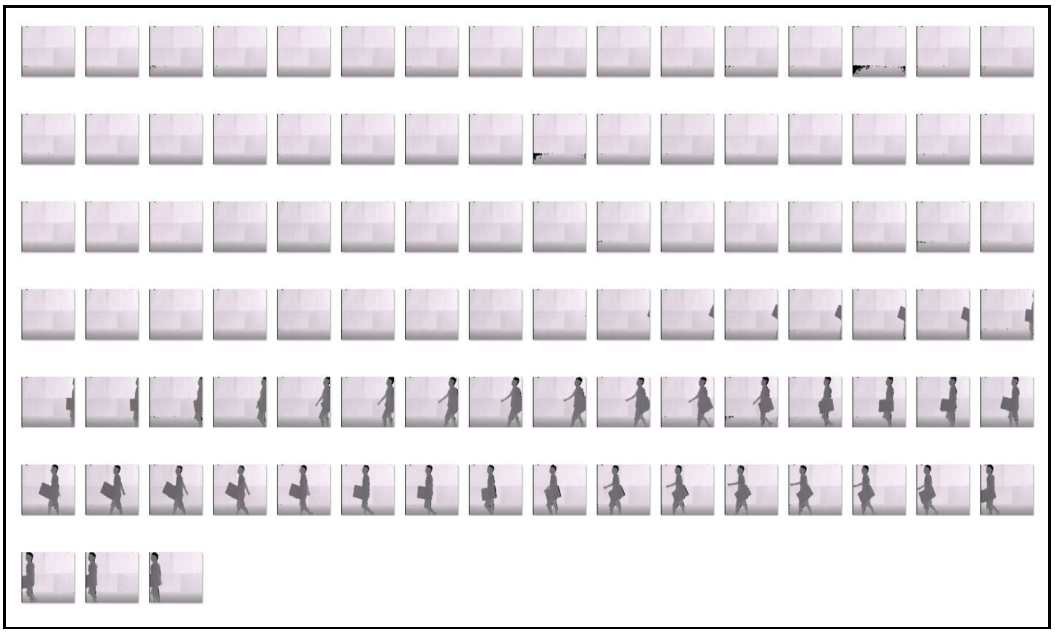

Figure 110. Sample of a miss-classified walking video frames using the six-class classification method.

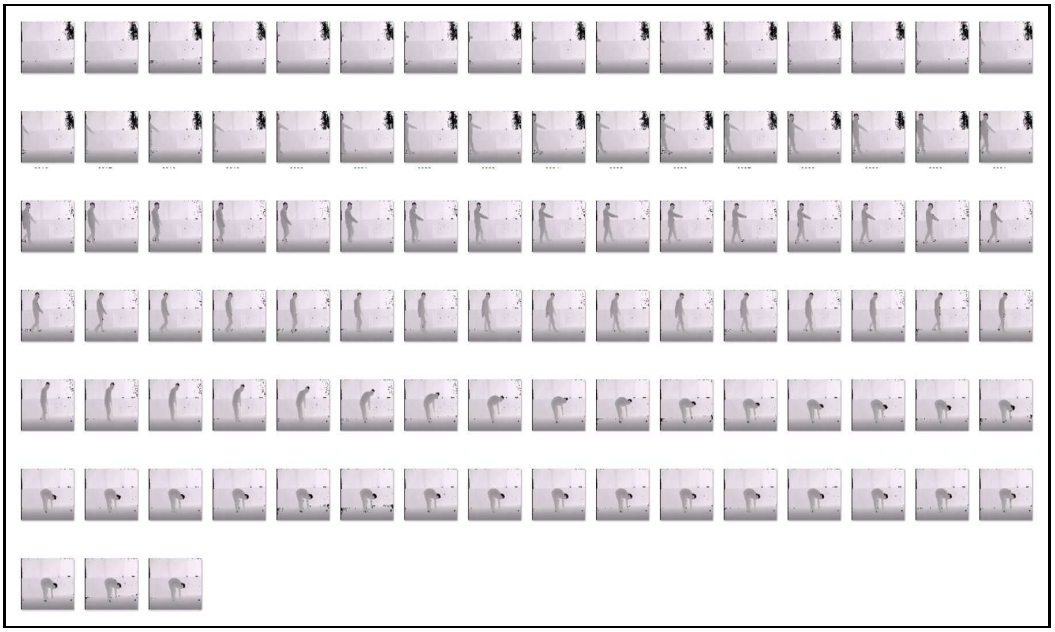

Figure 111. Sample of a miss-classified non-fall video frames using the fall versus non-fall classification method.

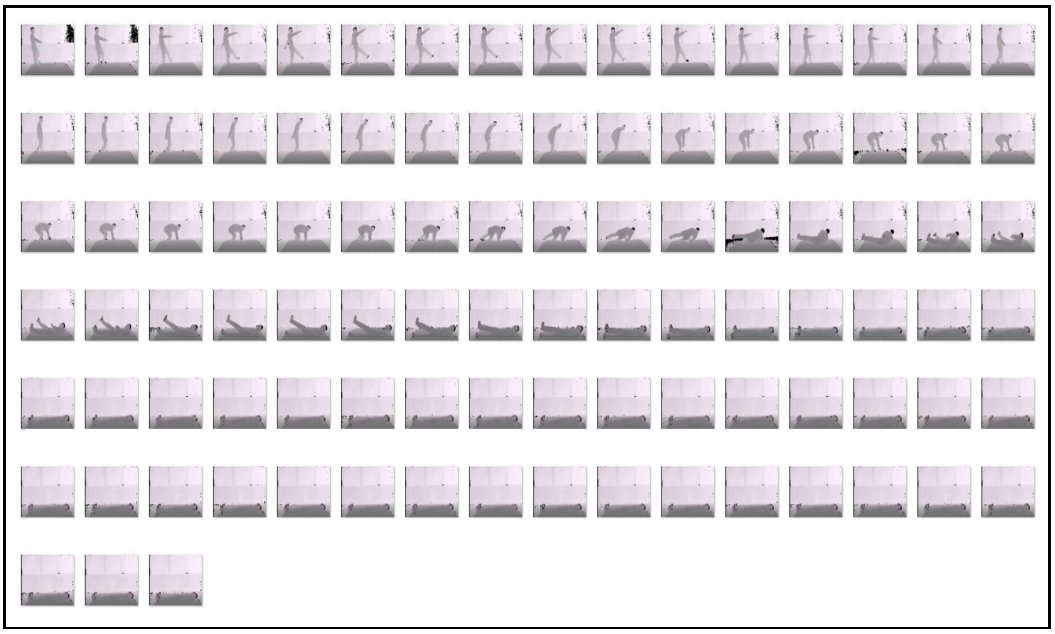

Figure 112. Sample of a miss-classified falling video frames using the fall versus non-fall classification method.

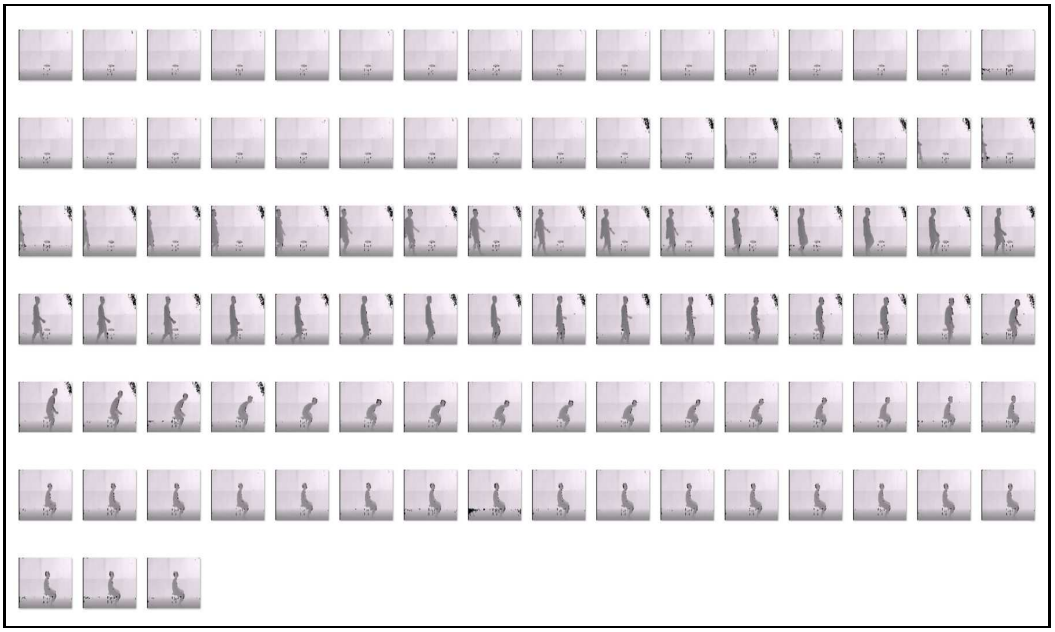

Figure 113. Sample of a miss-classified non-fall video frames using the fall versus non-fall classification method.

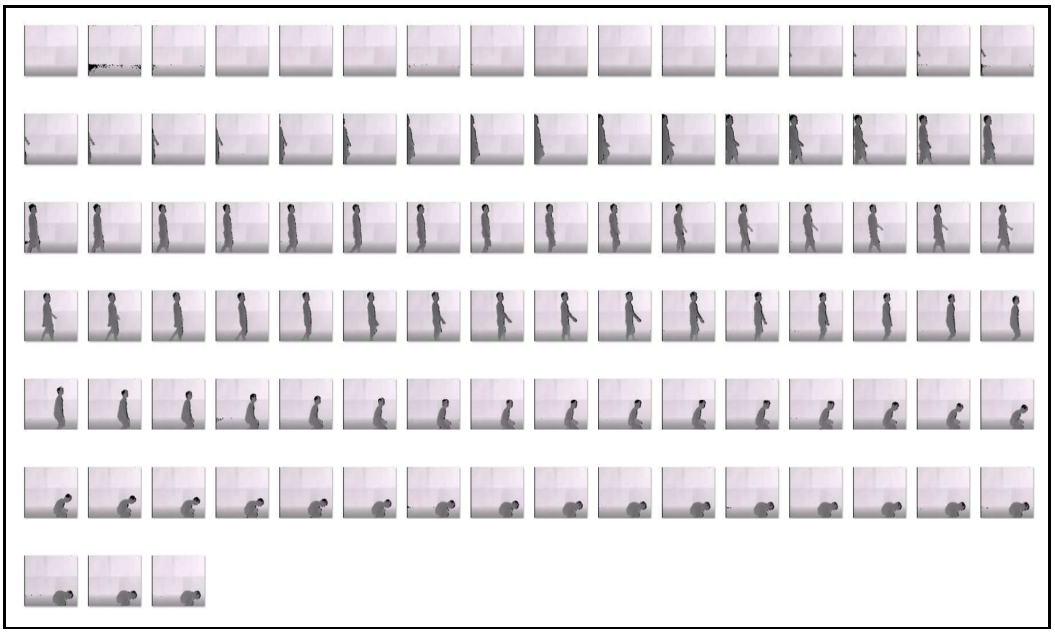

Figure 114. Sample of a miss-classified non-fall video frames using the fall versus non-fall classification method.

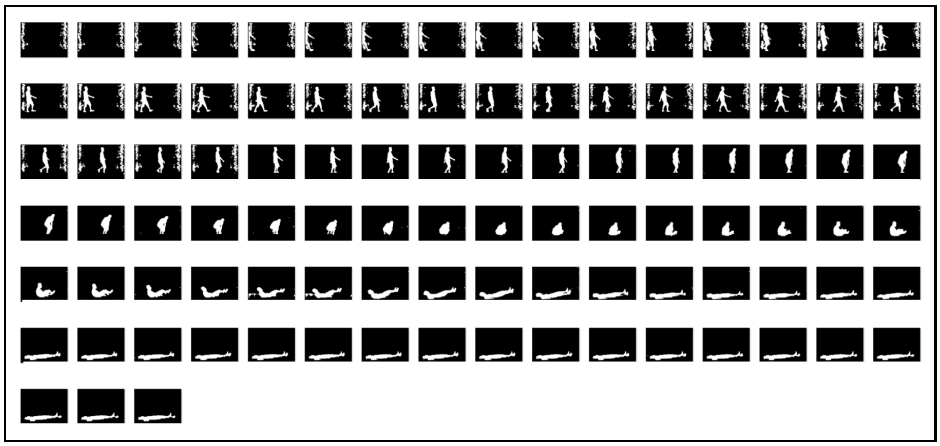

Figure 115. Sample of a miss-classified non-fall video frames using binary frames approach in experiment 1.

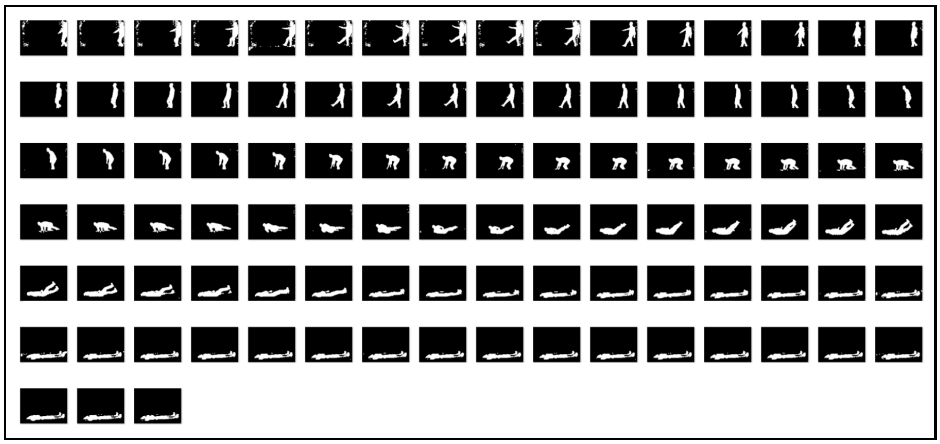

Figure 116. Sample of a miss-classified falling video frames using binary frames approach in experiment 1.

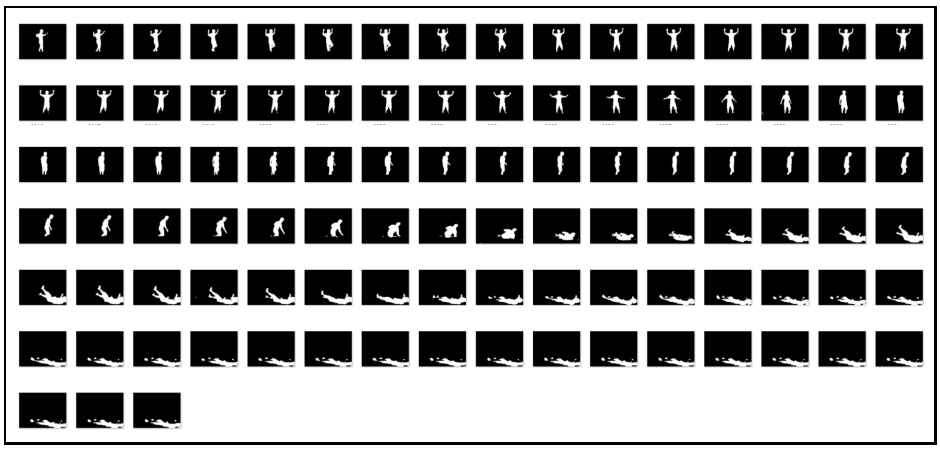

Figure 117. Sample of a miss-classified falling video frames using binary frames approach in experiment 2.

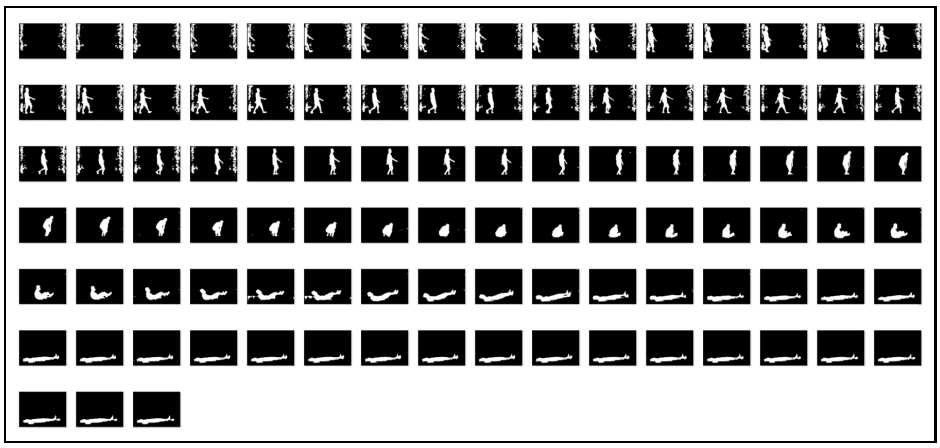

Figure 118. Sample of a miss-classified non-fall video frames using binary frames approach in experiment 3.

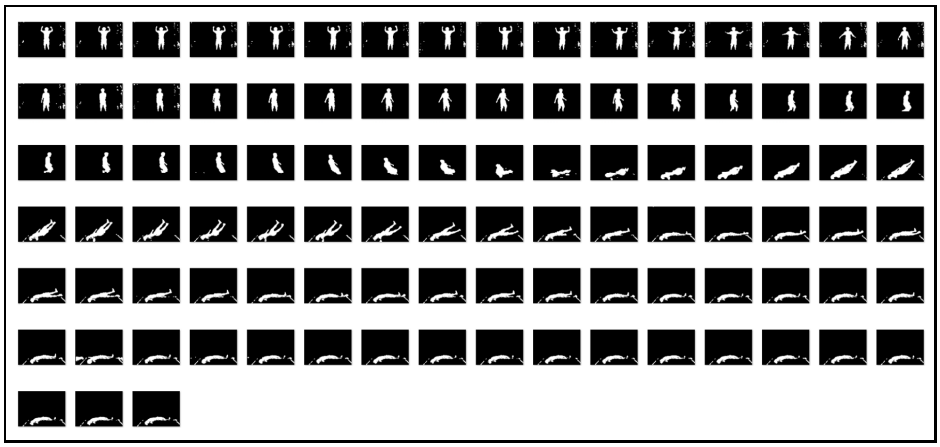

Figure 119. Sample of a miss-classified falling video frames using binary frames approach in experiment 3.

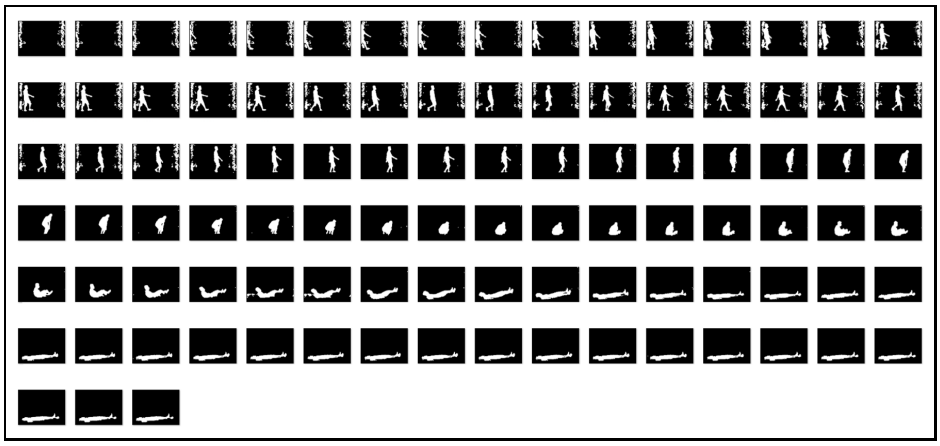

Figure 120. Sample of a miss-classified non-fall video frames using binary frames approach in experiment 4.

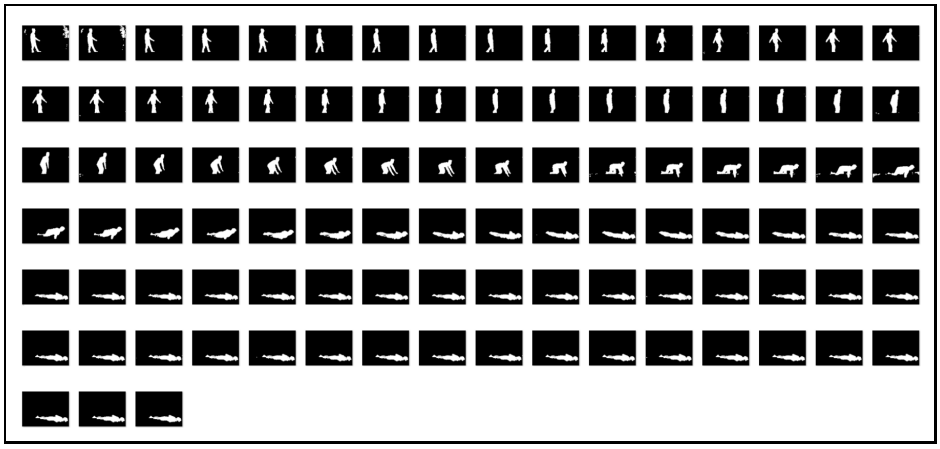

Figure 121. Sample of a miss-classified falling video frames using binary frames approach in experiment 4.