# Human Fall Detection System

*A report submitted in partial fulfilment of the requirements for the award of the degree of*

**Bachelor of Technology**

**in**

**Electronics and Communication Engineering**


By

**Akshay Kumar**

**ECB18037**

and

**Sourav Nath**

**ECB18027**




**Department of Electronics and Communication Engineering**

School of Engineering, Tezpur University

Tezpur-784028, Assam, India.


**2021-2022**

## ***Declaration by the Students***

We hereby declare that the project work presented in this report entitled "***Human Fall Detection System***", submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2021-2022, has been carried out by us and that it has not been submitted in part or whole to any institution for the award of any other degree or diploma.

Date: 22.06.2022

Place: Tezpur University

Students' name with signature

**1.**

**(Sourav Nath)**

**2.**

**(Akshay Kumar)**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**
**TEZPUR UNIVERSITY**

**Tezpur-784028, Assam, India**

**Dr. Riku Chutia** **Phone: 03712-275260**
**Assistant Professor** **email: riku@tezu.ernet.in**

# CERTIFICATE

This is to certify that the report entitled '***Human Fall Detection System***' submitted to the Department of Electronics and Communication Engineering, Tezpur University in partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering, is a record of project work carried out by Mr. Sourav Nath ***(ECB18027)*** and Mr. Akshay Kumar ***(ECB18037)*** under my supervision during the period from January 2022 to June 2022. All support received by them from various sources have been duly acknowledged. No part of this report has been submitted elsewhere for the award of any other degree or diploma.

Date: 22.06.2022 **Dr. Riku Chutia**

Place: Tezpur University ***Supervisor***

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**TEZPUR UNIVERSITY**

**Tezpur-784028, Assam, India**

**Prof. S. Sharma**
**Head of the Department**

**Phone:** +91-3712-275251
**Fax:** +91-3712-267005/6
**email:** sss@tezu.ernet.in

# CERTIFICATE

This is to certify that the report entitled "***Human Fall Detection System"*** is a bonafide record of project work carried out by Sourav Nath (ECB18027) and Akshay Kumar (ECB18037) submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2021-2022. They have carried out their project work under the supervision of **Dr. Riku Chutia**, **Assistant Professor, Electronics and Communication Department, Tezpur University.**

This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn as recorded in the report. It only signifies the acceptance of this report for the purpose for which it is submitted.

Date: 22.06.2022 (S. Sharma)

Place: Tezpur University

# ***Certificate by the Examiner***

This is to certify that the report entitled ***"Human fall detection system"*** submitted by Sourav Nath ***(ECB18027)*** and Akshay Kumar ***(ECB18037)*** in partial fulfilment of the requirements for the degree of Bachelor of Technology in Electronics and Communication Engineering has been examined by me and is found satisfactory for the award of the degree.

This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn as recorded in the report. It only signifies the acceptance of this report for the purpose for which it is submitted.

Date: 22.06.2022 (Examiner)

Place: Tezpur University

# ***Acknowledgement***

The real spirit of achieving a goal is through the way of excellence and austere discipline. We are truly indebted to many individuals who had helped and supported me throughout the period of my project work.

We avail this unique opportunity to express our gratitude and indebtedness to our project supervisor Dr. Riku Chutia, Assistant Professor, Department of Electronics and Communication Engineering, Tezpur University for his sustained interest, constant advices, perpetual encouragement and constructive criticism during the course of the project work and preparation of the manuscript.

We would like to thank Prof. S. Sharma, Professor and Head of the Department, Electronics and Communication Engineering, Tezpur University for providing necessary resources without which this project of mine would not have been possible.

And last but not the least; our heartiest thanks to all who wished us success especially our parents, whose support, sacrifices and care makes us stay on earth.

# TABLE OF CONTENTS

# List of figures

# CHAPTER 1

# INTRODUCTION AND IMPLEMENTATIONS

## 1.1. Introduction

Falls of the elderly always lead to serious health issues as the decline of their physical fitness. The falls may be caused by slipping on the floor, fainting or weakness. The situation may get worse if the elderly person is alone at home at the time of the fall as no-one will be there to call for immediate medical help. It is even possible the no-one comes to know about the person's fall until some time. The longer it takes for the medical help to arrive, the greater is the risk for the person.

The problem can be solved by using a *smart human fall detection system* that automatically detects a fall in real time and send emergency alerts to the person's relatives so that they might take the appropriate steps. Various kind of such fall detection systems are available which perform the detection using various sensing like visual based, audio based, acceleration and position based, vibration based, or combination of any of these. All these models have their own advantages and disadvantages. Due to relatively less work done in this field, the options for commercially available solutions are limited. The biggest limitations of these systems are their high cost, less reliability, limited compatibility and ease of application.

So, in this project work, a fall detection system has been designed which uses live video feed of the person to detect a fall. The objective is to design a real-time Human Fall Detection System based on the output from a camera mounted inside the room. This information can then be used to take the appropriate steps like triggering an alarm, sending alerts to the relatives or to the nearby medical centre.

The system uses an algorithm which distinguishes instances of falling from those of sitting and standing. Various techniques have been devised to extract important features like the angle of the human in the image and the approximate length of his/her projection on the floor. The algorithm keeps processing the live video from a camera and whenever a fall is detected, an alarm is triggered.

## 1.2. Literature Review

Most of the works done on fall detection systems can be broadly classified into three types based on the fall detection approach and hardware used. These are wearable/smartphone based, ambience based and vision based. Vision based systems are most popular these days because they can more accurately distinguish human falls from other activities.

The first paper proposes a fall detection system that has been designed and implemented to satisfy the requirements through a smartphone [1] which will be attached to the belt of the user. When a fall is detected it sends an alarm through application of the fall detection algorithm and continuously collects acceleration data through the accelerometer that is built-in; all this is done by an application with self-learning characteristics. The user and the application interact using the stock phone interface. Another paper uses used the k-nearest neighbour [2] (kNN) classifier (Duda et al., 2000) is a supervised machine learning approach for learning a function from training data that labels elements in the feature space based on the closest training data. The objects might be represented by position vectors in a multi - dimensional space to identify. The third paper presents a real-time algorithm based on the human 3D bounding box [3], which is given in global coordinates. The infrared (IR) signal from Kinect is used to get depth data, which is unaffected by varying illumination. To evaluate whether a certain behaviour is a fall or otherwise, their system uses a 3D bounding box to evaluate the first derivative (velocity) of width, height, and depth. Kinect relies on three different sensors: an RGB camera, an infrared camera, and an acoustic sensor, all of which were created by PrimeSense[4]. OpenNI [5] is an important Kinect development tool. PrimeSense also provides OpenNI, which is an open source software to access a human subject's depth data, calculate and monitor its articulate stance, and use it for human traceability, gestures, and motion detection.

There have been a lot of work going on in the field of Human Fall Detection Systems (FDS) lately. These systems are based on a variety of sensors which are to monitor the movement to estimate various postures and activities including falling. There are systems based on wearables and accessories. Smartphone based fall detection systems which use the device's built-in sensors like accelerometer and can also use the network to send alerts of the fall to various people concerned. However, these wearable and accessories based fall detection systems have the limitations that it is not practical to carry such a device all the time and it also needs regular recharging. An elderly person can forget to carry the device or can decide not to carry it for convenience. Slipping and falling in the bathroom is very common but people are not expected to carry their smartphone or wearable in the bathroom. Also, not all the elderly have smartphones with these sensors. Another family of fall detection systems is ambience and vision based fall detection systems. These use various physical quantities like pressure variations, infra-red radiations, sound variations, vibrations, or camera based visual signals. These need permanently installed setups in the rooms of the house which are connected to power supply and network connection which make them independent of any manual maintenance. Most of these systems, however, are very expensive and cannot be afforded by many.

Hence, there is a need for a human fall detection system which is rather simple in its working but highly accurate and reliable. It also should be low-cost so that it can be used by more and more elderly people to ensure emergency aid at time of falls. This was the main objective of this work.

# CHAPTER 2

# HUMAN BODY DETECTION

There are many different approaches used for detecting the human body in visual fall detection systems. Also, various pre-processing techniques are used to make the features easier to extract and making the detection process easier. In this work, a range of these techniques were studied and their suitability for the FDS was observed. Here are the techniques worked upon and the results that were obtained:

## 2.1 Haar Cascade algorithm

Haar cascade [6] is a machine learning based object detection algorithm. Initially it uses many positive and negative images and then uses them to train feature functions. The trained model is then ready to detect and localize objects in new unseen images. The number of features used for finding a complex object (a human, for example) is very large. In order to save computational time and resources, the features are grouped in various stages (cascaded) and are applied on images one after another. The most distinguishing features are checked in the beginning stages and the window goes to next stages only when a possibility of finding an object is found. Otherwise the window is discarded and the next window is checked. Its only when a window satisfies all the stages of the cascade, the object is said to be localized and is marked with a bounding box.

In this project a Haar Cascade model was trained to localize the position of human in the video frame. The frame is then sent to a Convolutional Neural Network (CNN) [7], both of which have been trained using a custom image dataset. This method is divided into various steps which include:

1. Creating the dataset
2. Training the Haar Cascade model
3. Constructing and training the CNN model
4. Testing the performance of the model on images and video clips of falling

## 2.2 Collecting the image dataset for training Haar Cascade and CNN models

The images for the dataset were extracted from the frames of video clips of falling and non-falling instances. A Mi A3 smartphone's wide angle camera was used to capture the videos. The camera has a resolution of 1920×1080 pixels and the video was taken at a frame rate of 30 frames per second. The frames were extracted from these videos using the OpenCV library functions.

During the extraction of these frames, the images are cropped to remove the portions of the frame which are not needed to detect a fall. For example, a person lying on a bed is assumed to be sleeping and hence needs not be considered by the fall detection algorithm. Also, the cropped frame has a dimension of 960×650 pixels. This is resized by a factor of 0.2 to convert

it into 192×130 pixels size. This makes the processing of these images faster and easier for the algorithms.

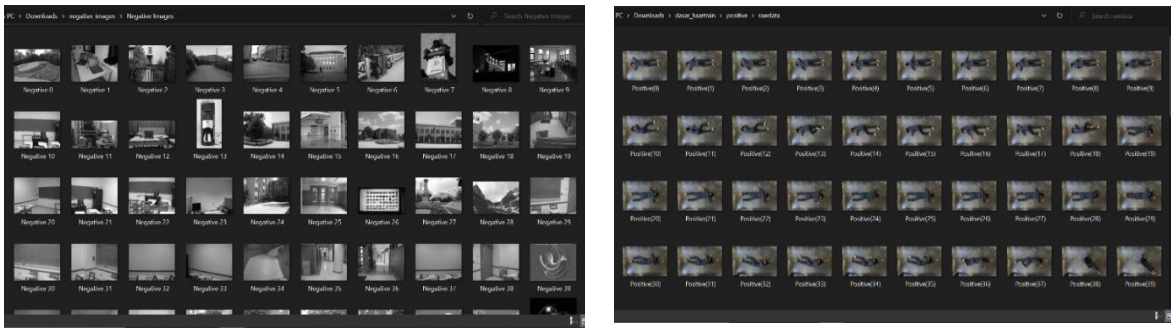
A total of 389 images were extracted from the video of duration 10 minutes and 49 seconds. Out of these 389 images, 270 images were selected for the dataset which showed distinct poses of human (fallen or not fallen) while the rest were discarded because they did not represent any relevant features.

## 2.3 Training the Haar Cascade

A Haar Cascade model was trained using a tool that was taken from a YouTube channel- "Dasaradh Makes". The 'dasar_haartrain.rar' file was downloaded and its contents were extracted. It consists of various executable files to select the Region of Interest (ROI), to create samples, to train the haar cascade and other necessary functions. The steps used in training the cascade using this tool are illustrated below:

Step 1:

Training the haar cascade requires two categories of images- 'Positive' and 'Negative'. The positive images are those ones in which there is a human in the frame. 139 such images are taken from our image dataset. The positive images must be in .BMP format. The negative images comprises of anything arbitrary which does not have a human. We took 1200 grayscale images from Kaggle Datasets negative images.

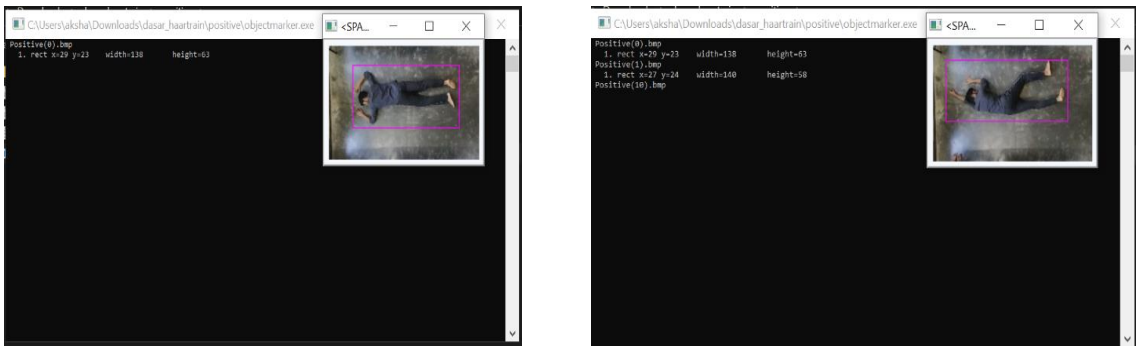


*(a)* *(b)*

Figure 2.1 (a) Negative images, Kaggle Dataset (b) Positive images

Step 2:

We need to select the region of interest (ROI) for each of the positive images using the application 'objectmarker'. We have trained the cascade to detect full human body. For this we



*(a)* *(b)*

Fig 2.2: Training haar cascade

need to enclose the human body in each image by dragging the cursor over the region. This gives the coordinates of the bounding box representing human in each image.

Step 3:

'01 samples_creation' application is run which prepares the images to be used for training. Then the cascade is trained using the '02 haarTraining' application. This application searches the images for all types of features and a set of features which best represents human body are selected. Multiple stages of haar cascade are trained depending on the quantity and complexity of the training images.

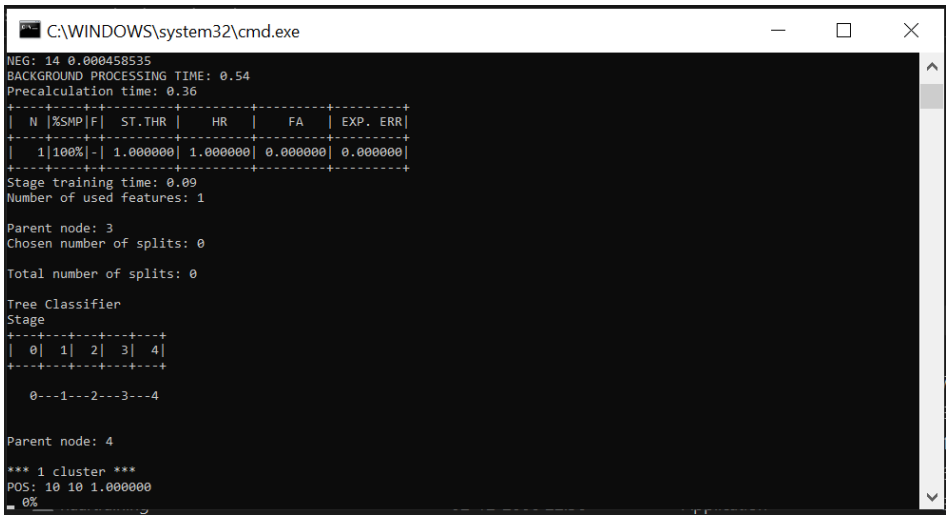



Fig 2.3: Cascade trainer application

After training is complete, a file called 'myhaar.xml' is created which contains the detailed parameters of all the stages of the Haar Cascade. This file will be required later on in the python program to detect and locate the human in a frame. Hence, the training of Haar Cascade is complete.

## 2.4 Results

Although Haar cascades are very simple to train and give fast detections, they have a major drawback of being less accurate especially for complex objects like human bodies. It means that Haar cascades lead to miss object of interest as well as detect undesired objects. It can be seen from the results which were obtained with a cascade classifier trained to detect full human bodies.

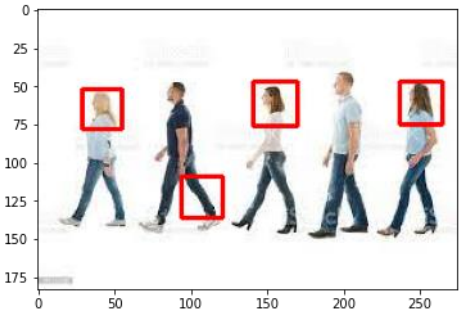



Fig 2.4 Haar cascade for full body

Not only they are missing the full bodies (by detecting only heads) but are also detect leg as full body which is completely incorrect. Due to these reasons, haar cascade classifier was found to be unreliable and hence, unsuitable for this project.

The CNN model consists of a number of layers including input layer, convolutional layers, batch normalization layers, maxpooling layers, fully-connected layers, and output layers. We tested different CNN modules including VGG-19, Xception, AlexNet [8]. We also designed some CNN modules ourselves by adding various layers from the TensorFlow library. We got the best prediction accuracy using the Xception CNN module. Xception uses depthwise seperable convolutional layers in addition to the regular layers used in other CNN models. So, we have used the Xception module for our Fall Detection System. The module can be divided into three parts- entry flow, middle flow and exit flow.

This convolutional base is followed by a flattening layer and some fully-connected layers. This gives the complete CNN model. The flattening layer converts the input image with dimensions (200,200,3) into a single long 1-dimensional feature vector that can be given to the fully-connected layers. The fully connected layers include two dense layers with 200 and 100 neurons, respectively, followed by a single neuron in the output layer for binary classification.

The image dataset for training the CNN model consists of total 270 images. Out of these, 135 represent falls and the other 135 represent no-falls. The dataset is divided into sub-categories for the training process:

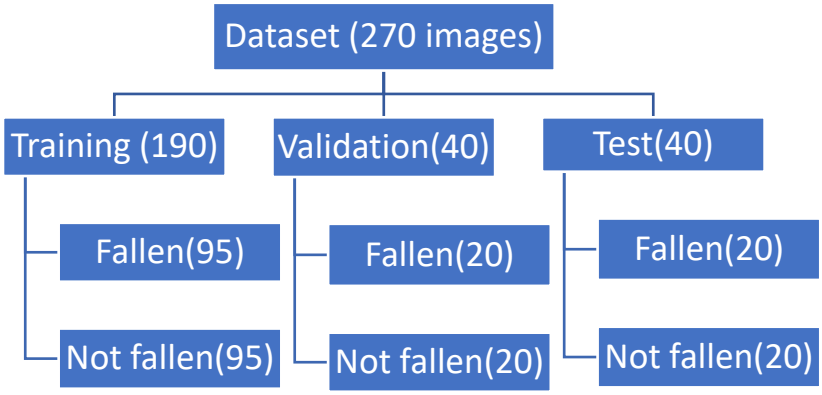



Fig 2.5: Dataset split into train, validation and test

These images are given to Tensorflow's Image data generator module. This module outputs the given set of images with a large number of variations like rescaling, rotating, height or width shifting, shearing, flipping and zooming. This makes the model more accurate and robust and also tends to prevent overfitting. The images are normalized by dividing all intensity values by 255. Train_generator, Validation_generator and Test_generator create such sets of images for train, validation and test purposes, respectively. Class_mode is taken as 'binary' because our model does binary classification i.e. fallen and not-fallen.

The Xception CNN module is imported from tensorflow.keras.applications with the parameters from ImageNet. The input shape is set as (200,200,3) which is same as that of our input images.

The layers of CNN model are added one by one. First of all, the convolutional base (conv_base) is added. conv_base is followed by a flattening layer the flattening layer is followed to two fully connected layers. The first and second hidden layers have 200 and 100 neurons, respectively. Rectified Linear Unit (ReLU) is used taken as the activation for both the layers. The output layer consists of a single neuron with sigmoid activation because it gives binary output.

Next, the model is compiled. The loss type is given as 'binary_crossentropy' and Stochastic Gradient Descent (SGD) optimizer is used. Finally, the model is training is done for 5 epochs which means that all the model training is done for 5 cycles. Also, each epoch is carried out in 19 steps because the batch size is 5 (95/5=19). Validation steps= 20/5=4. As it can be seen from the screenshot given below, the accuracy goes on increasing while the loss goes on decreasing with each epoch.

## 2.5 Observation

### 2.5.1 Accuracy

The accuracy of the model is determined using the test images which are unseen for the model. The accuracy on the test data was found to be almost 1 (100%) and loss was found to be 0.07449.

```
In [10]: loaded_model.evaluate(test_generator,steps=5)
         5/5 [==============================] - 4s 454ms/step - loss: 0.0745 - accuracy: 1.0000
Out[10]: [0.07449495792388916, 1.0]
```

### 2.5.2 Testing

Then we used the model to predict the output for some of the test images. The CNN model gives a fraction between 0 and 1 as the output. The values close to 0 correspond to 'not fallen' and values close to 1 represent 'fallen'. A threshold of 0.5 is taken to make the decision. The result of the prediction i.e. fallen or not fallen is given corresponding to each of the test case.

```
['not fallen', 'not fallen', 'not fallen', 'fallen', 'fallen']

<matplotlib.image.AxesImage at 0x1a35776d4c0>
```

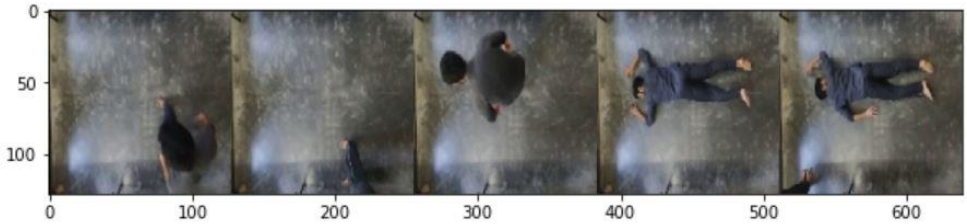



Fig 2.6 (a) Result on image dataset

Next, we tested the predictor on a video clip which has instances of fall as well as of not fall. The result of prediction is written on the video frame itself. Here are examples of the predictions:


Fig 2.6 (b): Results of the model for the test video

The classification of fallen and not fallen humans was done very accurately by the CNN model trained. However, it throws not light on the features of human body like orientation, angle, posture, aspect ratio, etc. To study the human body features which distinguish various postures including standing, sitting and falling, various techniques were used.

All the programming has been done in python programing language except the Haar cascade training module which uses C++. The platform used for python programming is Jupyter Notebook using Anaconda3 64-bit.

## 2.6 Image segmentation using thresholding

Image segmentation [9] means separating portions (segments) of an image from other segments of the image. It is mainly done in order to extract features from an image by keeping the useful information from the image and discarding the rest of the parts.

In thresholding, every pixel of the image is compared to a threshold value and if the pixel value is smaller than the threshold then the pixel is set to '0' else it is set to one. Thus, pixels of the whole image are divided into two regions- the ones with pixel value greater than the global threshold and those with pixel values less than the threshold. The OpenCV library has various functions for thresholding, the most common one being cv.threshold().

There are multiple ways of performing segmentation in images some of which are mentioned below:

- Supervised Segmentation by thresholding- Manual input
  In this method, the user has to manually give a threshold value and all the pixels in the image are segmented based on that single threshold value. Such a threshold is called Global Threshold.

- Adaptive thresholding
  Global thresholding might not be good in some cases especially when the lighting is not uniform throughout the image. This may give undesired segmentation results. To avoid this, adaptive thresholding is used. In adaptive thresholding, threshold are locally determined for small regions of the image and thresholding is done accordingly. This gives far better results than global thresholding in images with varying illumination.

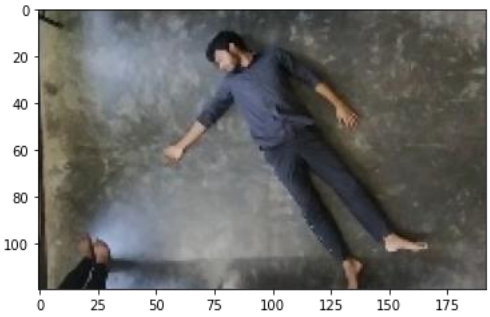
- Otsu's Binarization
  In Otsu's method, threshold is determined automatically unlike in global thresholding. Otsu's algorithm tries to find a threshold value which minimizes the weighted within-class variance.

In this project, it was tried to perform segmentation using manual thresholding on images in the attempt to separate the background floor area from the human in the images. Once the human is separated, one can perform the further steps of detection more easily and accurately.

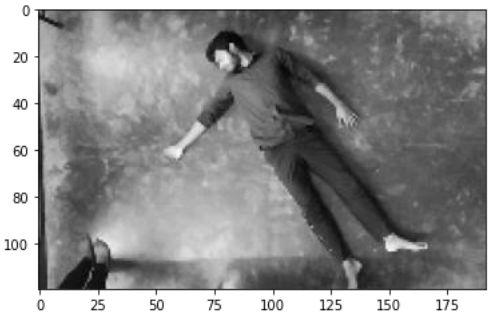
OpenCV function with a global threshold value of 50 was used in this project.

## 2.6.1 Method

**Step 1:** Conversion from BGR to GreyThe coloured BGR (Blue,Green,Red) image is converted so that the thresholding can be done with reference to a single grey channel instead of comparing all the three colour channels separately.



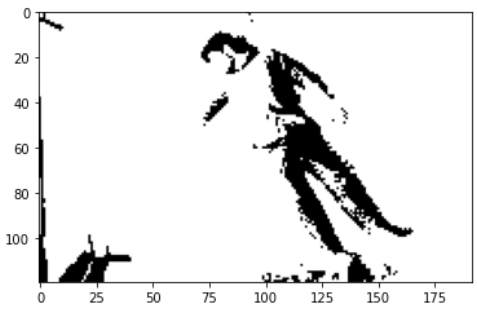

(a)




(b)

**Step 2:** Thresholding by using a global threshold of 50.

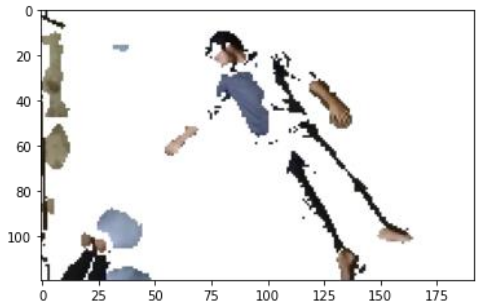
Various values of threshold were tried and the results were examined. The threshold of 50 was found to be best to separate the human from the background.




(c)

**Step 3:** Segmentation based on thresholding.

Now, the image is divided into two segments depending on the result of thresholding.




(d)

Fig 2.7 (a) Original image (b) Grayscale image (c) Binary image

(d) Segmented image to remove background

### 2.6.2 Results:

The result of segmentation was not as expected and didn't fulfil the requirement. The following disadvantages were found:

- The thresholding depends highly on the lighting conditions of the room. Also, different portions of the image did not receive equal light and shadows of objects and human made segmentation difficult.
- The colour of the garments that the human is wearing also matters a lot in thresholding. When the colour of the garment is similar to that of the floor, segmentation becomes very difficult.
- Different coloured floors will need different threshold values. Also, other objects with complex structures and colours can make the segmentation different.

Because of these limitations, segmentation of image to remove background from the image was not found reliable and was not used any further.

## 2.7 Body landmark detection

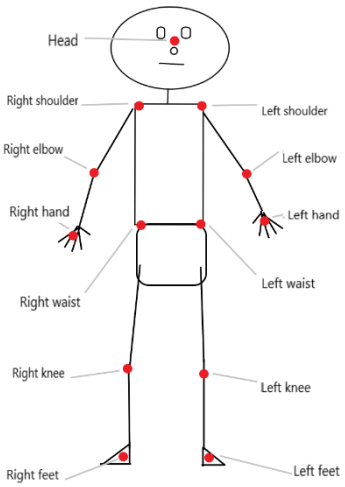
Human body has a number of landmarks with which one can easily perform pose estimation. These landmarks include major joints and features of the body including knee, elbow, hip, etc. This information can be used to detect if a person has fallen.

There are many body landmark detection models which have been proposed, one of which is Google's Mediapipe Pose [10] model (BlazePose GHUM 3D). BlazePose is a state-of-the-art pose estimation model which predicts the location of 33 body landmarks ranging from eyes, nose, hip, to toes. It is highly accurate and achieves real-time performance on most of modern devices including smartphones, desktops/laptops and even web.

## 2.7.1 Method

In order to study how the body landmarks were detected, a CNN model was trained with the objective to detect 13 body landmarks which used regression to estimate the x and y coordinates of respective landmarks for all the landmarks.

Here are the 13 body landmarks that were used for pose estimation:




1. Head
2. Left hand
3. Right hand
4. Left elbow
5. Right elbow
6. Left shoulder
7. Right shoulder
8. Left waist
9. Right waist
10. Left knee
11. Right knee
12. Left foot
13. Right foot

Fig 2.8 Body landmarks

The CNN was trained using a custom dataset consisting of 360 images with a resolution of 720×720 pixels. Every image contains human in different postures depicting various activities of a workout routine. The landmarks for all the images are collected manually one by one which consists of x and y coordinates of every landmark. The landmarks were then normalized by dividing every value by 720 so that all the values lie in between 0 and 1.

| | Head_x | Head_y | left_hand_x | left_hand_y | right_hand_x | right_hand_y | left_elbow_x | left_elbow_y | right_elbow_x | right_elbow_y | ... | right_waist_x | right_wa |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.4917 | 0.2458 | 0.4278 | 0.5806 | 0.5708 | 0.5806 | 0.4306 | 0.4569 | 0.5681 | 0.4597 | ... | 0.5292 | 0 |
| 1 | 0.4958 | 0.2708 | 0.3139 | 0.5514 | 0.6972 | 0.5458 | 0.3972 | 0.4319 | 0.6194 | 0.4514 | ... | 0.5514 | 0 |
| 2 | 0.5083 | 0.2667 | 0.2222 | 0.3639 | 0.7931 | 0.3486 | 0.3500 | 0.3764 | 0.6667 | 0.3597 | ... | 0.5472 | 0 |
| 3 | 0.4764 | 0.2431 | 0.2167 | 0.3611 | 0.7125 | 0.3250 | 0.3403 | 0.3736 | 0.6347 | 0.3583 | ... | 0.5236 | 0 |
| 4 | 0.4514 | 0.2514 | 0.4153 | 0.1583 | 0.4903 | 0.1403 | 0.3458 | 0.2583 | 0.5708 | 0.2125 | ... | 0.5028 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 355 | 0.8403 | 0.8778 | 0.5403 | 0.9028 | 0.5667 | 0.9500 | 0.6486 | 0.8861 | 0.6944 | 0.9403 | ... | 0.6083 | 0 |
| 356 | 0.8236 | 0.8750 | 0.5486 | 0.8986 | 0.5708 | 0.9514 | 0.6722 | 0.8875 | 0.6903 | 0.9361 | ... | 0.6167 | 0 |
| 357 | 0.8361 | 0.8750 | 0.5417 | 0.8972 | 0.5708 | 0.9514 | 0.6583 | 0.8958 | 0.6972 | 0.9403 | ... | 0.6389 | 0 |
| 358 | 0.5597 | 0.6278 | 0.4250 | 0.7458 | 0.6083 | 0.9111 | 0.4083 | 0.6639 | 0.6069 | 0.7681 | ... | 0.5000 | 0 |
| 359 | 0.7569 | 0.3403 | 0.6181 | 0.6278 | 0.8375 | 0.6028 | 0.6319 | 0.5000 | 0.8014 | 0.4931 | ... | 0.7861 | 0 |

360 rows × 26 columns

Fig 2.9 Normalized landmark positions

The CNN model consisted of a convolutional layer with a depth of 5 followed by a MaxPooling layer of dimension 2×2. Then the output of the MaxPooling layer is flattened and sent to dense ANN network. The ANN network consists of 3 layers. The first layer has 200 neurons, the second layer had of 100 neurons and the last layer, which is also the output layer, has 26

neurons, one for each coordinate value. Since the model is based on regression, all the dense layers have ReLU (Rectified Linear Activation) function.

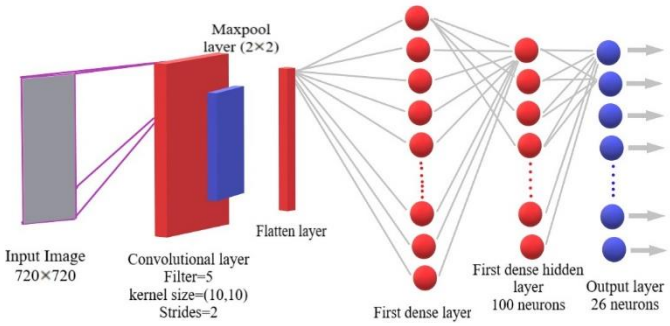



Fig 2.10 Convolutional Neural Network (CNN) model

For training, the 720×720 resolution images were given as input to the CNN layer while the corresponding outputs which are coordinates for the landmarks were also given for supervised learning.
The dataset, which consisted of 360 images and corresponding landmarks, was divided into train and test samples using Sci-kit learn's test-train split in a test to train ratio of 8:2. The model was trained for 5 epochs and the model was then tested on test images. The results found were as follow:

Fig 2.11 Results of Body Landmark detection

## 2.7.2 Conclusion

The model was tested on test input images and the accuracy was found to be very low. The estimated landmarks were not in agreement with the actual landmark positions.

The possible reason for the failure of the model can be the incorrect choice of hyper-parameters of the neural network i.e. structure and depth of the convolutional layers, size of the dense layers, the activation function, etc. Also, the task of landmark estimation may be too complex for this rather simple model and it might have led to underfitting. Further experimentation and study will be required to realize an accurate pose estimation model using body landmarks. Google's BlazePose is highly advanced pose estimation model which is fast as well as accurate.

## 2.8 Principle Component Analysis

Principle Component Analysis [11] (PCA) is a statistical method that is used to extract the most important features of a sample. PCA is very useful for dimensionality reduction which is the process of reducing the dimension of a dataset. The following is an example of dimensionality reduction:

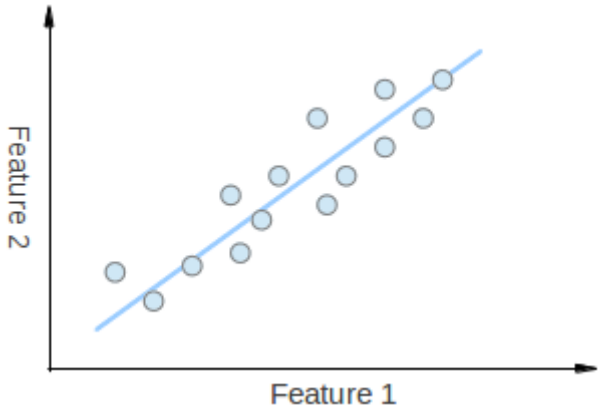



Fig 2.12 2-D dataset [11]

Here, we have a 2-dimensional dataset with points which vary along the two axes- feature 1 and feature 2. Although the points may seem random but from observation it can be noticed that the points are somewhat along a straight line which is represented by the blue line.

Moreover, knowing where a point lies on the blue line will give more information than knowing where it lies on any of the two axes. So, the information that otherwise needed two dimensions can is now somewhat given in one dimension too. This leads to dimensionality reduction.

PCA gives the direction along which the data varies the most. It does so by giving vectors which are most important (principle) features and are called eigenvectors. These eigenvectors contain multiple information regarding how the data is spread across the dimensions. The length of any eigenvector represents how much the data varies along its direction. Also, all the eigenvectors start from a common point which is the centre of all the points in the dataset.

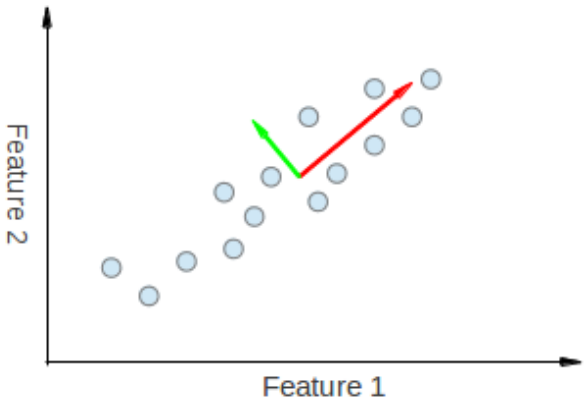



Fig. 2.13 Eigenvectors [11]

Principle component analysis can be used to find the orientation of objects in an image. The eigenvectors will give the directions of maximum spread of object and the direction perpendicular to it.

Here is an example on how PCA helps determining the orientation of objects in an image:

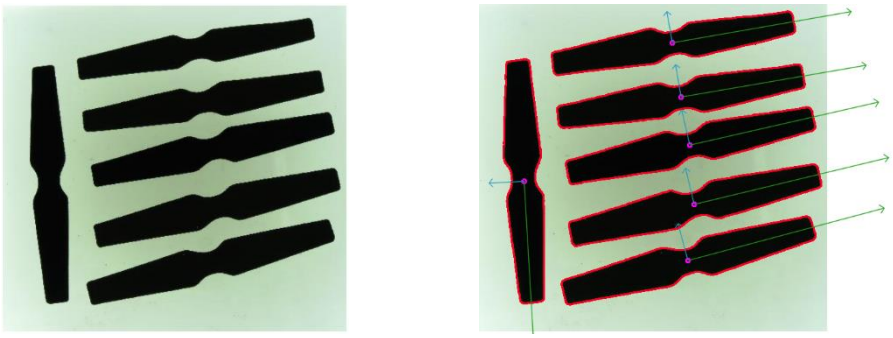


Fig 2.14 Determining object orientation through PCA [11]

In human fall detection, the orientation of the human in an image is a useful feature to determine the posture and detect falls. For this purpose PCA was tested on the database images and the results were observed.

Here are the steps required for finding object orientation from images:

1. First of all the region of interest is chosen by some object detection technique. In this project it was done using YOLO object detection which gives the bounding box enclosing the object which in this case is a human.

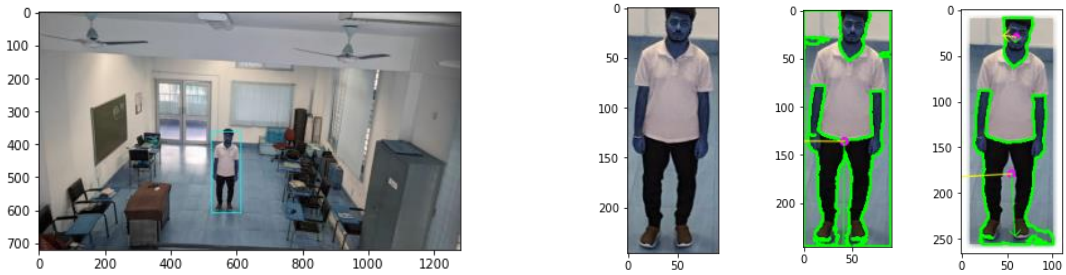


Fig. 2.15 Image segmentation process

   One of the challenges is that the edges of the image are also detected as contours which give false contours. To avoid this, the image is added with some padding on the edges and then the edges are blurred.

2. First of all the image is converted from BGR to grey. This is done for performing thresholding on the image with an appropriate threshold value. This gives out a binary image.

3. Next, contours are detected from the binary images. These contours are the boundaries of the objects which are differentiated by sudden gradient changes.

4. These contours are then tested for desired size. Contours enclosing a certain range of area are kept while those smaller or larger than those are discarded.

5. After the desired contours are extracted, PCA is performed over those contours to find the eigenvectors. These eigenvectors are then scaled properly to represent the variance of data and then these eigenvectors are superimposed on the original image for visualization.

### 2.8.1 Results

There were some issues with PCA which restrain its use for finding orientation of human body in an image-

The determination of contours depend on the gradient changes in the binary image, which in turn depends on the relative colours of the human, surrounding, clothes, and also on the lighting. Due to this, the human as a whole is not detected as one contour. Rather, different parts of the body and clothes are detected and processed as separate contours which don't serve the purpose of finding the orientation of the human body.

For this reason, orientation of the human was not determined using PCA and another algorithm had to be used which is discussed in a later part of this report.

## 2.9 YOU ONLY LOOK ONCE (YOLO)

You only look once (YOLO) [12] is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev.

YOLOv3 is extremely fast and accurate. In mAP measured at .5 IOU YOLOv3 is on par with Focal Loss but about 4x faster. Moreover, you can easily tradeoff between speed and accuracy simply by changing the size of the model, no retraining required.

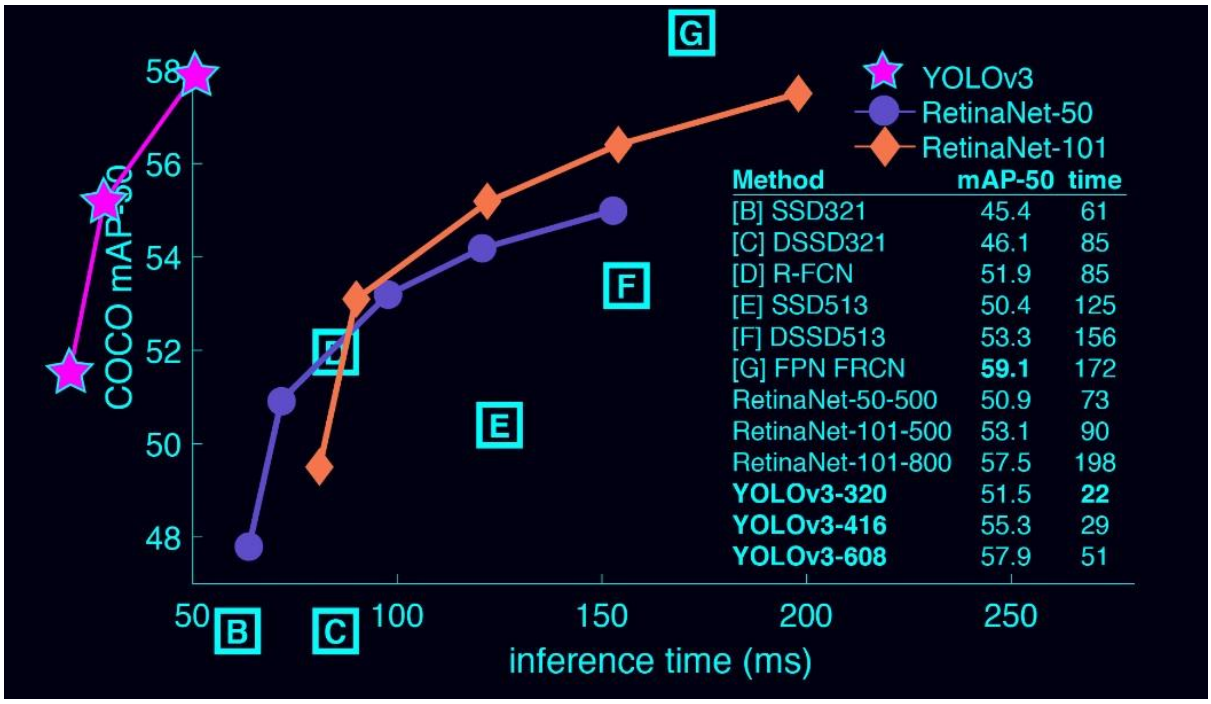



Fig 2.16 Comparison between detectors [12]

People can tell what items are in a picture, where they are, and how they interact just by looking at it. The human vision system is quick and precise, allowing us to do complicated activities like navigating with minimal thought. Fast, precise object identification algorithms would enable the user to drive automobiles without the requirement of specialised sensors, assistive gadgets to communicate real-time scene information to human users, and general-purpose, responsive robotic systems to emerge.

Classifiers are repurposed for detection in today's detection systems. Such systems use a classifier for an item to recognise it in a test picture and assess it at various locations and sizes. The classifier is performed at regularly spaced areas over the whole picture in systems like deformable components models (DPM).

Other modern algorithms, such as R-CNN, employ region proposal methods to construct possible bounding boxes in an image before applying a classifier to them. Post-processing can be used to improve the bounding boxes, reduce duplicate detections, and rescore the boxes depending on other items in the scene after categorization. Because each individual component must be trained independently, these complicated pipelines seem to be slow and difficult to optimise.

Moreover, YOLO has a mean average accuracy that is more than double that of comparable real-time systems. When developing predictions, YOLO uses a global approach to the image. YOLO, unlike sliding window and area proposal-based approaches, observes the full image during training and testing, implicitly encoding contextual information about classes as well as their appearance. When developing predictions, YOLO uses a global approach to the image. YOLO, unlike sliding window and area proposal-based approaches, observes the full image during training and testing, implicitly encoding contextual information about classes as well as their appearance. Because it cannot perceive the entire context, Fast R-CNN [14], a top detection approach, misidentifies background patches in an image as objects. When compared to Fast R-CNN, YOLO produces around half as many background mistakes.

Object representations that can be generalised are learned through YOLO. YOLO surpasses top detection algorithms like DPM and R-CNN when trained on natural photos and evaluated on artwork. If exposed to new domains or unexpected inputs, YOLO is much less likely to crash since it is highly generalizable. In precision, YOLO is still behind state-of-the-art detecting technologies. While it can swiftly recognise items in photos, it has trouble pinpointing the exact location of some things, especially little ones.

### 2.9.1 Detection Procedure

The many components of object detection are combined into a single neural network. To forecast each bounding box, our network uses characteristics from the whole image. It also calculates all bounding boxes for an image throughout all classes at the same time. This implies that our network considers the entire image as well as all of the objects inside it. End-to-end training and real-time speeds are possible because to the YOLO design, which maintains excellent average accuracy.

## 2.10 CONCLUSION

After working on all of the techniques and algorithms to detect human bodies from an image, is was found that YOLO object detection model was the most suitable method for our visual based fall detection system. It was found to be fast and more robust than the other techniques studied. The various features of the bounding boxes like position, aspect ratio and dimensions could be used to obtain important information about the posture of the person in an image. Also, it detects not only humans but a range of other objects which help in distinguishing accidental fall from voluntary activities like sleeping on the bed. So, the methods used in this work employ YOLO object detection and the use of the other techniques discussed earlier were discontinued.

# CHAPTER 3

# METHODOLOGY

## 3.1 INTRODUCTION

This fall detection system uses YOLO object detection for extracting the various features from the image. First of all, the bounding boxes are found for human body and other objects of interest using YOLO algorithm. These bounding boxes are used for finding the features like angle, length on floor and area common with furniture. The angle at which maximum aspect ratio is found is the actual angle of the human in the image. Similarly, the apparent length of human as seen in the image is found using an equation.

In this fall detection system, the objective is to categorize the posture of any human detected in the image frame as standing, sitting or fallen. The following python libraries were used:

- OpenCV
- matplotlib
- SciPy
- Numpy
- Pillow (PIL)
- Playsound

To do this, the following features were extracted from any image frame:

1. Body orientation (angle)
   The angle of a body with respect to the vertical axis in the image is very helpful in determining if the human is standing or fallen. The angle of a standing human will somewhere between -10° and 10°. The angle of a body fallen on the floor can be anything depending on the direction of fall.

2. Body length (apparent) over floor:
   This is another feature which removes the ambiguity thrown by the previous feature. For example, the angle of a standing human will be same as that of a human body fallen along the vertical axis of the image. This is where the apparent length of the body comes into play. When a human is lying on the floor, the length of the human body (as seen in the image) is lesser than that of a standing person. In other words, a standing person covers more floor length as compared to a person lying on the floor.

3. Common area with furniture:
   If a person has large area common with some furniture in the image then there are chances that the person is sitting or sleeping. For example, a person lying on the floor is most probably fallen while a person lying in the exactly same posture on a bed would certainly be sleeping. For distinguishing between them, the common area between the

bed and the human can play a crucial role. This can be found using the bounding boxes of human and the particular furniture of interest.

4. The uppermost point of bounding box:
   If the uppermost point of the human body in the image is above the floor area then he/she definitely cannot be lying on the floor. It only leaves the possibilities of sitting and standing. This is done easily by checking if the y-coordinate of the bounding box for the human is above the uppermost point of the floor in the image.

Now, the techniques for finding these features and the methodology are given in the upcoming sections of this report.

## 3.2 Finding the angle of human body

The angle of human body is found by utilizing the fact that the height to width ratio of a bounding box is largest when the human is upright (along the vertical). Keeping this in mind, the frame containing the human figure is rotated at all angles ranging from -90° and 90° with intervals of 10° and human is detected each time using the YOLO object detection model. The angle is 0° in the vertically upwards direction. The angle values are considered positive in the clockwise rotation and negative in anti-clockwise direction. The height to width ratio of the bounding boxes are calculated for each angle and the angle at which this ratio comes out to be maximum is the actual angle of the human in the image. Here are several images depicting the step by step process:

In this case, Maximum ratio found was 2.8803418803418803 at angle 0°. So, the human in the image is found by the algorithm to be at an angle of 0° which is also obvious form the image as the person is standing.

Fig 3.1 Original image

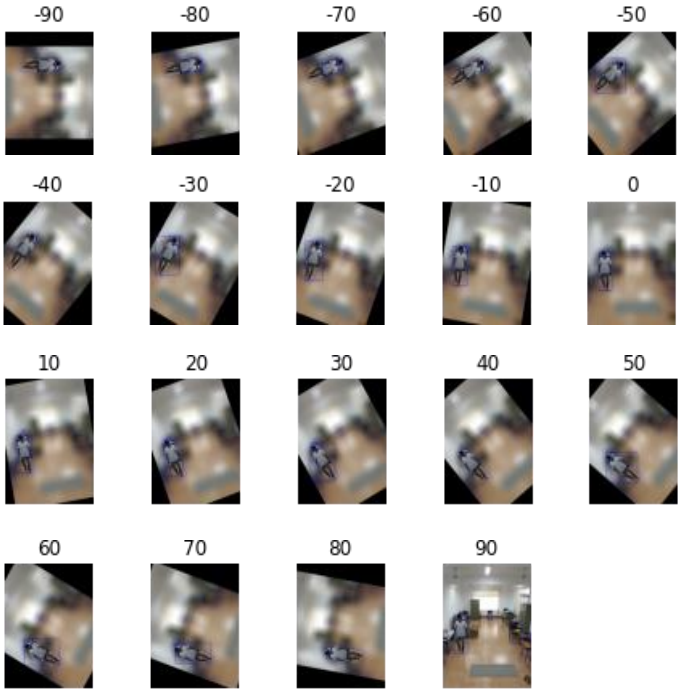



Fig 3.2 Image rotated at various angles

## 3.3 Finding the length of human (apparent) over the floor

To distinguish between a standing person and a person fallen in a direction along the vertical in the image, the floor length covered by him/her in the image was considered. If the floor length (apparent) comes out to be greater than a certain threshold then the human must be standing. It must be noted that the actual length of the person will obviously be constant. It only differs in the image due to perspective.

Now, the real challenge towards finding the actual floor length with help of the number of pixels covered in image is that the relationship is non-linear. It means that two far points separated by a certain distance cover less number of pixels than two points separated by the same distance but closer to the camera. To overcome this, a non-linear mapping was done to get the floor distance (in cm) per pixel for pixels at various positions in the image.

For this, markers were put on the floor at uniform intervals (2 feet) going away from the camera. The pixel number for each marker is found from a higher resolution (3264×2448 px) and the gaps between consecutive markers are also noted. Then all these readings are mapped for 1080p video frames by dividing them by an appropriate constant. Then the floor distance per pixel is approximated for every interval and a relation is derived after plotting the parameters in MS Excel.

$$Approximated\ cm\ per\ pixel = \frac{Actual\ gap\ between\ markers\ (cm)}{Pixel\ gap\ for\ same\ pair\ of\ markers\ in\ image\ (px)}$$


Fig 3.3 Arrangement for length

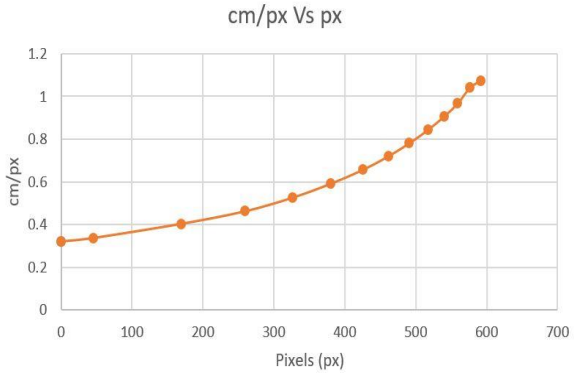



Fig 3.4 Non-linear relation between cm/px and px

Here is the equation that was found to satisfy this non-linear relation:

$$Y = 4 \times 10^{-15}X^5 + 10^{-12}X^4 - 10^{-9}X^2 + 10^{-6}X + 0.319$$

Here, X is the pixel number, counted from the bottom of the image and Y is the corresponding floor length for that particular pixel in centimeters.

To find the length of human in the image, the corresponding distances for all pixels (along vertical direction) are calculated and the accumulated sum of these value throughout the length of the bounding box gives the approximate length of the human on the floor. The threshold is kept as 7 feet (213.36 ≈ 320 cm) for maximum human length (in general).

Fig 3.5 Image of fallen person

Fig 3.6 Image of standing person

In both of the images shown above, the person is at an angle of zero degree with the vertical but the length of floor covered by the person in the first image is 212 cm while that in the second image is 523 cm. This distinguishes fallen (first image) from standing (second image).

## 3.4 Common area with the furniture

This feature takes the help of furniture located near the person to decide if he is sitting. For this, it is checked if the bounding box for the person coincides partially with the bounding box of any of the detected furniture. It is done by checking all the furniture one after another for having common area with the person in the image. If there is an area common then that area is computed. The process of finding common area is described below:

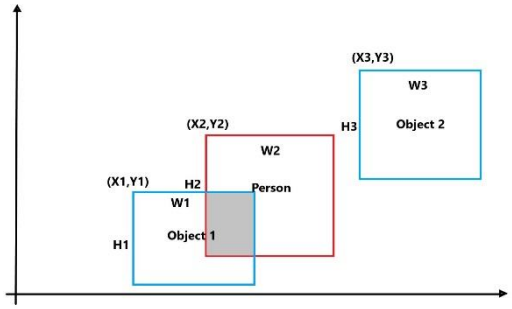



Fig 3.7 Common-area between human and person

$$H = \min(Y1 + H1, Y2 + H2) - \max(Y1, Y2)$$

$$W = \min(X2 + W2, X1 + W1) - \max(X1, X2)$$

$$Common\ area = H \times W$$

$$common\ area\ (\%) = \frac{Common\ area}{Human\ area} \times 100$$

Once the percentage of human area common with a chair is calculated, it is compared with a threshold to decide if the person is sitting on the chair. Similar concept can be used with bed to decide if the human is lying on the bed or not. These furniture items are detected by using the same YOLO algorithm which is used to detect human.

## 3.5 Working: Using these features to determine posture

1. First of all, the detection process is applied the frame and if a human is detected, the further steps are carried out otherwise the frame is skipped and next frame is processed. If a human is detected, the posture detection algorithm becomes active.

2. Initially it is checked if the head of the human (highest point) is above the floor level or not. If it is above floor area, then the posture can be either standing or sitting but cannot be fallen. If the highest point of the human in the image is lower than the floor then the next step is finding the angle of the human.

3. If the angle is not between -10° and 10° (inclusive) then the posture cannot be standing. In that case, the human's common area with any furniture (chair) is calculated and if it is above a threshold (20%) then it is labelled as "sitting" otherwise it is "fallen".

4. If the angle is between -10° and 10° (inclusive) then also it is checked for common area with any furniture to decide if the person is "sitting". If the person is not sitting then the length of the person's projection on the floor is checked.

5. If the length is found to be smaller than a threshold value then the person is labelled "fallen" otherwise he/she is labelled as "standing". The complete methodology is represented by the flowchart shown in the following page.

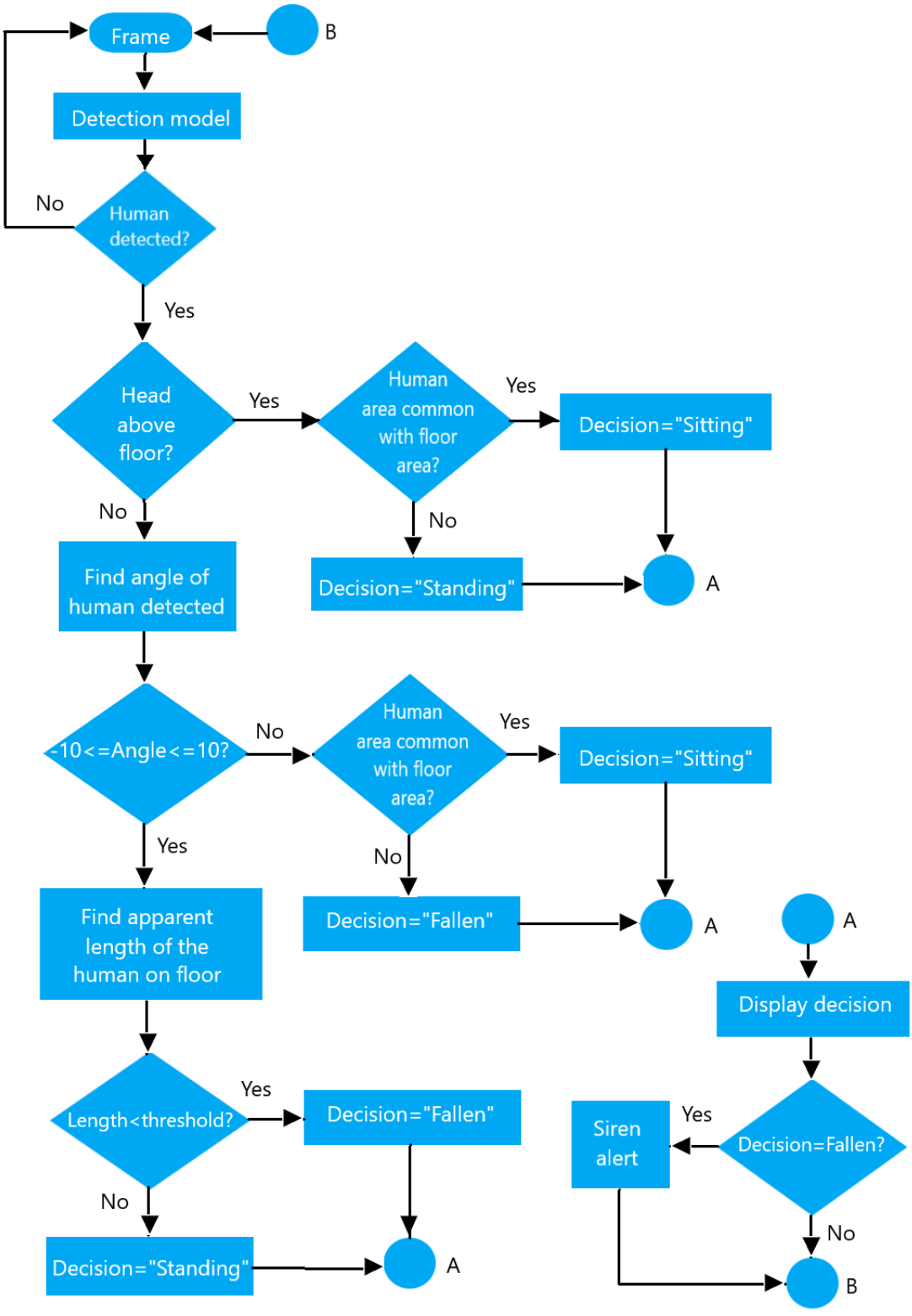



Fig 3.8. Working flowchart

## 3.6 Results and discussions:

The fall detection algorithm was applied to a video clip containing a variety of postures possible in a household. The results for posture estimation for various postures are discussed below:

### 3.6.1 Standing

The first criterion used for determining “Standing” posture is that the uppermost portion of the person should lie above the farthest point of the floor. This criterion is used in the Fig 4.1 to decide that the posture of the person is standing. The second criterion is that the angle of the person should be within the range of -10° to 10° (both inclusive) and the length of the human calculated according to the algorithm should be above the threshold (320 cm). This criterion gave the posture as “Standing” to the person detected in Fig 4.2.


Fig 3.9. Standing posture 1

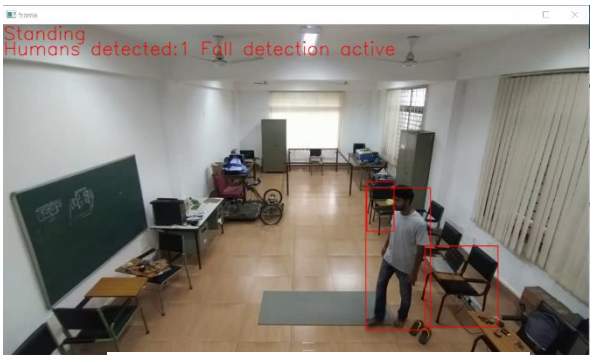



Fig 3.10. Standing posture 2

### 3.6.2 Sitting

The criterion used to decide the “Sitting” posture is that the bounding box for the human detected must have at least 20% of its area common with one of the furniture (chair). Fig 4.3 shown below represents a “Sitting” posture detection.


Fig 3.11. Sitting posture

### 3.6.3 Fallen

This is the most important posture for the Human Fall Detection System is “Fallen”. The criterion for detecting fallen feature is that either the angle of the human is outside the range of -10° to 10° or his/her approximated length calculated is less than the threshold of 320 cm (≈7 feet). In Fig 4.4, the person is detected to be at an angle of 90° with the vertical and so, is labelled as “Fallen”. In Fig 4.5, the angle is found to be 0° but the length was found to be lower than the threshold and so it was labelled as “Fallen”.


Fig 3.12. Fall detected 1


Fig 3.13. Fall detected 2

### 3.6.4 Limitations of the system

1. **Human body not detected**- Since the complete algorithm is based on the detection and localization of human bodies using YOLO, any failure to detect the person will lead to failure in detecting the fall. This occurs mostly in the cases in which the human is inverted at some angle in the image. Here are some instances where the YOLO object detection algorithm failed to detect the human:


Fig 3.14. Algorithm failure 1


Fig 3.15. Algorithm failure 2

2. **Sitting on floor is also labelled as fallen-** Since the algorithm does not distinguish between sitting on the floor and fallen, any activity that includes sitting on the floor (yoga, for example) will be detected as a fall which can cause undesired trouble. As per this algorithm, a person is sitting only when he/she is sitting on a chair.

# CHAPTER 4

## FUTURE SCOPE

- The algorithm can be modified to remove the limitation due to non-detection of human body by YOLO. For this, one can check three variants of an image for human bodies- original image and image rotated by angles 90 degree and 180 degree.
- The system can be converted to a stand-alone system by using a high resolution camera mounted to the wall and giving the live footage to the FDS algorithm that will be running on a Raspberry Pi. In addition to triggering the alarm, a GSM module can also be connected to send an alert of the fall to the people concerned.
- This system can be combined with some models using other types of sensing like microphone array, ambient sensor based, etc.
- We can use some different types of fall detection algorithms and merge them together to increase the accuracy and make the system more robust.

## CONCLUSION

This project, we tried to design a low cost Human Fall Detection System using camera for fall detection for elderly persons. We designed an algorithm to extract the relevant features from the video frames and use them to detect human postures including falls. The standalone wall-mountable system eliminates the limitations of some of the previously available systems based on wearables or accessories to detect the fall because it does not need any extra attention or maintenance (like regular charging for wearable systems).

This module can be easily implemented using a Raspberry pie, a high-resolution camera and some extra circuitry to be able to send the alerts for detected falls. This can also be implemented with various other predictive algorithms to improve the accuracy of the system. Also, other types of sensing like microphone arrays and Infrared cameras can be used parallel with the wall mounted cameras for better performance of the fall detection.

# 5. References


[1] Stefano Abbate Marco Avvenuti Francesco Bonatesta Guglielmo Cola Paolo Corsini Alessio Vecchio. *A smartphone-based fall detection system. A journal on ScienceDirect* <https://doi.org/10.1016/j.pmcj.2012.08.003> (2012)

[2] Chien-Liang Liu , Chia-Hoang Lee, Ping-Min Lin. *A fall detection system using k-nearest neighbour classifier. A journal on ScienceDirect.* <https://doi.org/10.1016/j.eswa.2010.04.014> (October 2010)

[3] Mastorakis, G., Makris, D. *Fall detection system using Kinect's infrared sensor*. *J Real-Time Image Proc* **9,** 635–646 (2014). <https://doi.org/10.1007/s11554-012-0246-9>

[4] Wikipedia contributors, *PrimeSense*, *Wikipedia, The Free Encyclopedia*, 7 February 2022, 21:06 UTC, <https://en.wikipedia.org/w/index.php?title=PrimeSense&oldid=1070511126> [accessed 19 June 2022]

[5] Wikipedia contributors, *OpenNI*, *Wikipedia, The Free Encyclopedia*, 25 April 2020, 05:38 UTC, <https://en.wikipedia.org/w/index.php?title=OpenNI&oldid=953005047> [accessed 19 June 2022]

[6] Wikipedia contributors, *Cascading classifiers*, *Wikipedia, The Free Encyclopedia*, 29 April 2022, 14:26 UTC, <https://en.wikipedia.org/w/index.php?title=Cascading_classifiers&oldid=1085275235> [accessed 19 June 2022]

[7] Chatterjee C. Chandra, *Basics of the Classic CNN*, 31 July 2019, <https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add>

[8] Tsang, Sik-Ho, *Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification),* 25 September 2018 <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568>

[9] K., Naveenraj. *Image segmentation using Python's scikit-image module.*(3rd December, 2021), <https://www.geeksforgeeks.org/image-segmentation-using-pythons-scikit-image-module >

[10] Google LLC. *Mediapipe pose. Pose landmark model(BlazePose GHUM 3D).*(2020), <*https://google.github.io/mediapipe/solutions/pose.html*>

[11] Intel. *Introduction to Principal Component Analysis (PCA).(1999)* <*https://docs.opencv.org/3.4/d1/dee/tutorial_introduction_to_pca.html*>

[12] Redmon, Joseph and Farhadi, Ali, *YOLOv3: An Incremental Improvement Redmon, arXiv*, (2018)