In the name of God

Fall Detection Using Camera Feeds

Mansour Saffar Mehrjardi, Mohammadhadi Shadmehr, Mohsen Mohammadi

Introduction

Falls are a major issue among older adults. It is estimated that one out of every three older adults (those age 65 and over) falls each year. Of those who fall, many suffer serious injuries, such as hip fractures and head traumas, which reduce their mobility and independence, and lead to an increased risk of early death. The direct medical cost of falls among older adults in the U.S. in the year 2000 was more than $19 billion. This cost does not account for the decreased quality of life and other long term effects experienced by many after suffering a fall. Studies have also found an increased risk of physical and physiological complications associated with prolonged periods of lying on the floor following a fall, due to an inability to get up. Older adults living alone are at great risk of delayed assistance following a fall.

In this study we propose an event detection method for fall detection based on 3 features which are common in most cases of fall. We propose a method for reducing false alarms by incorporating time in our data to achieve a better correct classification rate by combining 50 consequent frames and then feeding these combined frames as data to our classifier. We used support vector machines classifier for classifying our data and detect fall.

Methods and implementation

Data acquisition:

In our project we used webcam for data acquisition. MATLAB data acquisition toolbox was used to acquire frames. We took about 10 seconds recording from the subject normal walking and also his fall. Acquired images were of size 720 in 1280 pixels and frame rate was 30 FPS.

A sample walking is shown below:

Fig1

Background extraction:

There are several methods to extract background from consequent frames. In our project we used median method to extract background from 300 frames. We took median of each pixel value along 300 frames and assigned each pixel to its median value. The formula expression for this method is show in Eq1.

Eq1: $MP_{x,y} = \text{median}(P_{1\ x,y} , P_{2\ x,y,} P_{3\ x,y,..,} P_{300\ x,y})$

The background extracted from frames using this method is shown below:

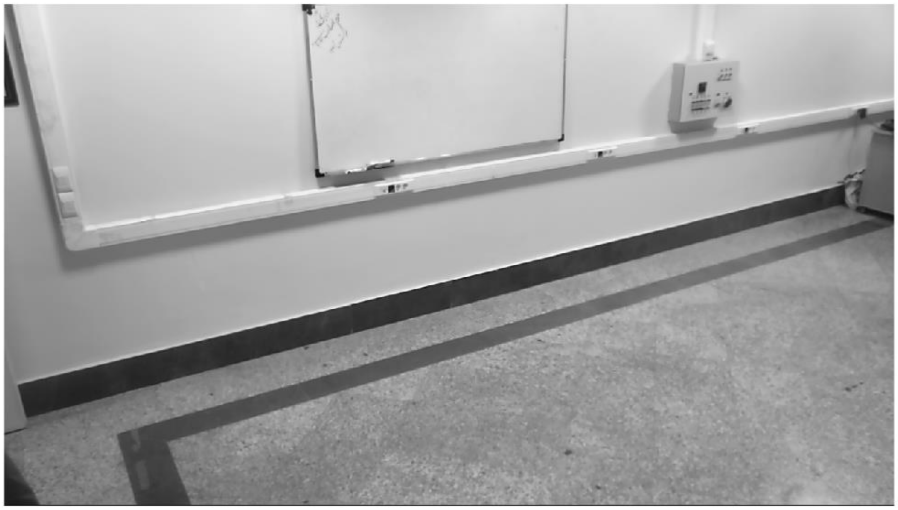

Fig 2- background

Silhouette extraction:

In order to extract silhouette in each frame we subtracted the background from each frame and then used thresholding to extract silhouette. Thresholding is a method to convert grayscale image into binary image and in our project we used thresholding to extract silhouette from frames. The value for thresholding was chosen arbitrarily and we examined several values to get the best result possible.

The results after subtraction are shown below:

Fig 3- frames after subtraction

The result after thresholding (threshold value=25):

Fig 4-frames after subtraction and thresholding

As it is seen in Fig 4, there are some white points in background scene and also a clear shape of silhouette is not achieved yet. To improve silhouette shape and also to remove background noise from frames we use morphological processing.

Morphological image processing covers a wide range of functions to manipulate the overall shape of objects in image, regardless of their pixel values. Here we use 2 methods called "opening" and "closing" images. In these methods we define a structuring element and then this element surfs the whole image pixels and assigns a value to the pixel according to a set of rules (hit or fit).

The result of Fig 4 after opening and then closing each frame is shown below:

Here structuring element is a 9 by 9 square.

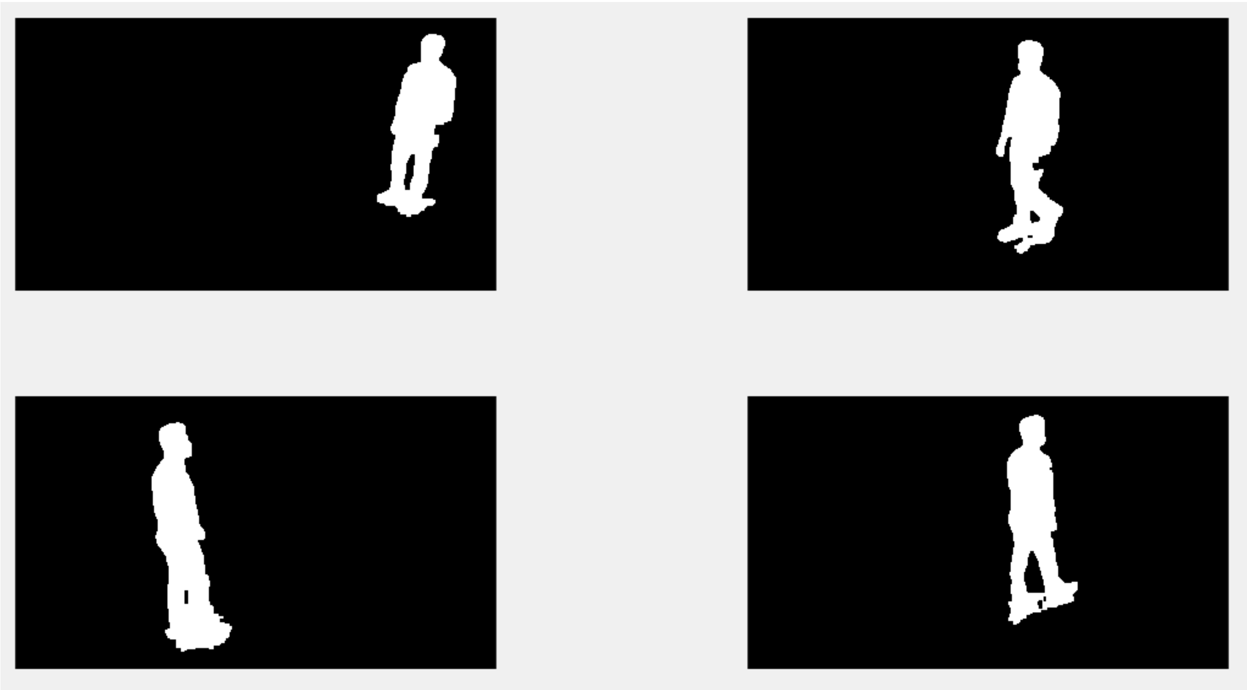

Fig 5- result after opening and closing

As you can see the overall shape of silhouette is improved except for the feet which is corrupted due to shadow.

Improvements:

As we in Fig 5 the silhouette has some distortion in subject's feet. A method to improve shadow effect is to increase threshold value. If we increase threshold value to 45 we can see that shadow effect is improved but the overall shape of silhouette is distorted.

Note that in our project we want to determine whether the subject has fallen or not and by exploiting this fact we can say that we do not need a clear shape of silhouette.

The result of opening and closing frames with threshold value of 45 is shown below:

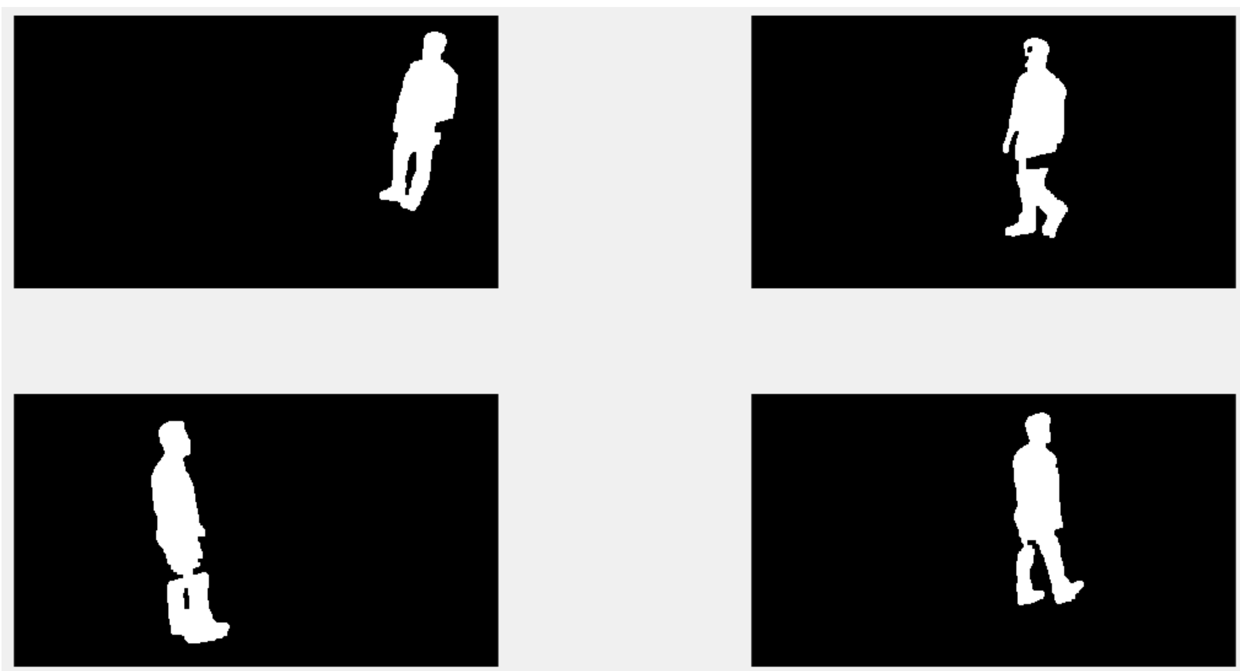

Fig 6 – result with threshold value=45

Feature selection

In our project we used 3 features to analyze whether the subject has fallen or not. We fit a square in silhouette of subject and located its centroid to compute subject's aspect ratio which is the proportion of his silhouette's height to its width. We also used silhouettes' height which is actually the height of the square we fit in our subject's silhouette as a feature. Finally we used the centroid's height as the third feature and extracted these features for every frame of our data.

A sample of centroid point which was extracted automatically is shown in Fig7.

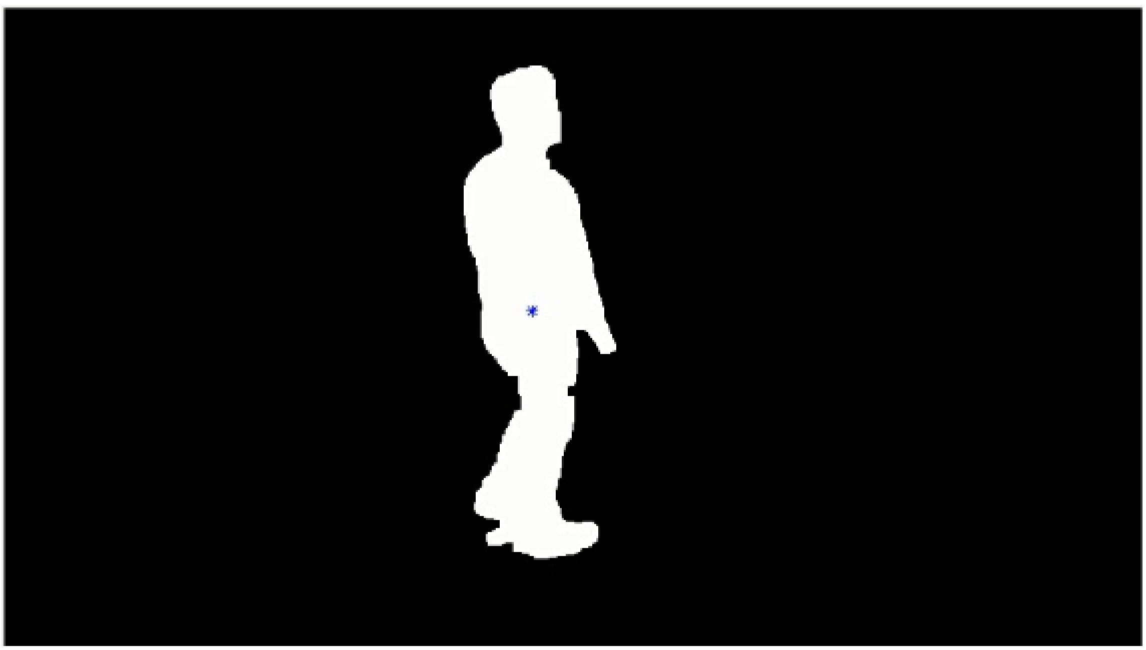

Fig 7- subject's centroid shown with blue star

Note that the reason we chose these 3 features was that in all cases in which a person falls, these 3 features change significantly compared to cases in which that person is standing. We also considered every 50 consequent frames and combined their features to make an event data which has 150 features.

By using 50 frames instead of only one frame for fall detection we actually used time as a feature to increase out classifiers CCR and also reduce false positive and false negative cases.

Classification

In order to detect fall we need to classify our data into two groups: fall or not fall. We used SVM (support vector machine) classifier in our project. SVM is a kernel based nonlinear classifier which was introduced by Russian scientist and researcher Vladimir Vapnik.

In this method, we are to find a hyper-plane that separates two class of data in space with dimension n from each other. In addition, we know that the best hyper-plane should have same distance from the points of two class in order to have better classification for new points.

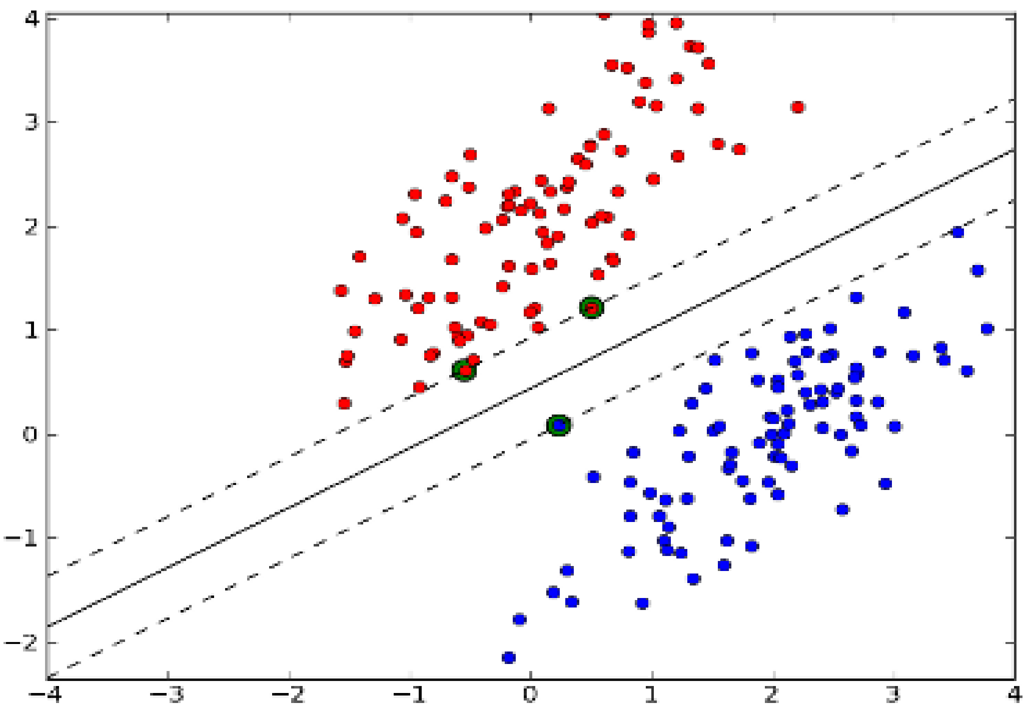

Fig 8- SVM classifier with support vectors in green

General formula that show a hyper-plane is:

$$W_0^T x + b_0 = 0$$

New problem is introduced:

- Minimize: $\emptyset = \frac{1}{2} W^T W$

- Subject to: $y_i(W^T x_i + b) \geq 1$ i = 1,2,…,l

This problem is solved using Lagrangian multipliers:

$$W_0 = \sum_{x_i \in \delta v} \lambda_i y_i x_i$$

In this summation λ_i is zero in non-support vectors and show we need just support vectors for our calculations.

So we have:

$$g(x) = W^T x + b = \sum \lambda_i y_i (x_i^T x) + b$$

In some cases, this is impossible to separate classes with hyper-plane in the dimension of our space. For example in Fig.9 in left space, it is impossible to separate blue and red dots by a simple line. Unfolding feature space and mapping data to a space with larger dimension is a usual approach.

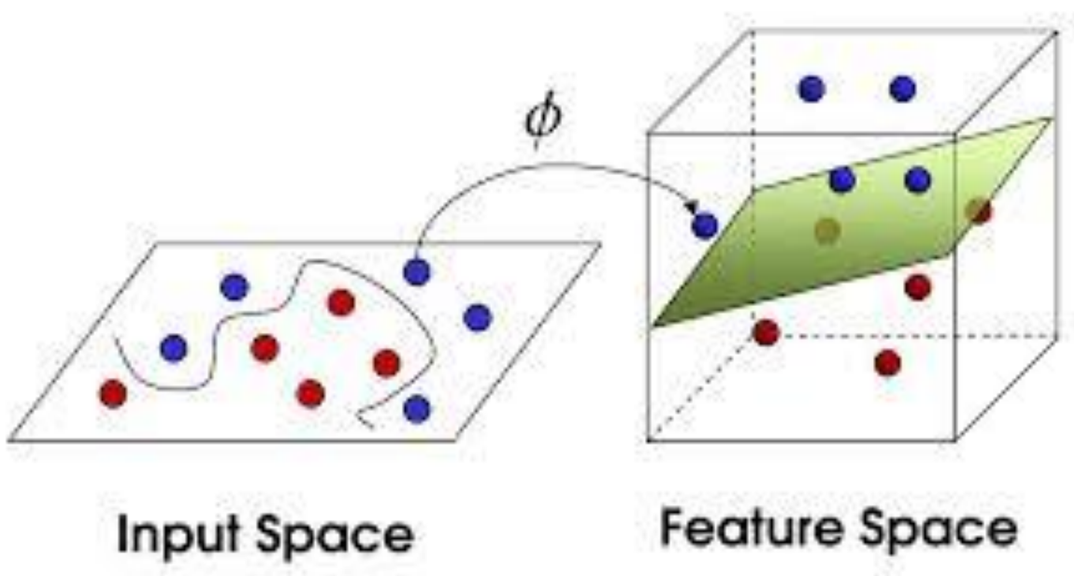

Fig 9- Mapping feature space to higher dimensions

Results

We used open source library *libSVM* that contain SVM function in different programming languages. We computed our results in MATLAB. We separated our 28 data which includes falling and not falling data into train and test data. 75% of our data was used to train SVM classifier and 25% of it was used as test data to assess our model. It’s also of importance to note that we chose our training data randomly to increase generalization of our classifier. The results for different kernel methods are shown below:

SVM type	Kernel type	Degree	Cost	Accuracy
C_SVC	Polynomial	5	10	%91.2
C_SVC	Polynomial	7	10	%91
C_SVC	Polynomial	20	10	%71.5
C_SVC	Linear	5	10	%97
C_SVC	Linear	7	10	%97
C_SVC	Linear	20	10	%97

References

I. *Kristian Haga Karstensan,"Silhouette extraction using Graphic's Processing Unit".*

II. *Mark S. Nixon, Alberto S. Aguado,"Feature Extraction and Image Processing for Computer Vision", Third edition, pp. 435-470.*

III. *Richard O.Duda, Peter E.Hart, David G.Stork."Pattern Classification" , Second edition.*

IV. *Erik E.Stone."Fall Detection In Homes of Adults Using the Microsoft Kinect".*