Pilani, Rajasthan 7th July, 2015

VISION BASED HUMAN MONITORING SYSTEM

Prepared by:

Aditya Sarma

Abhinav Mehta

Work carried out under the Guidance of

Dr. Jagdish L. Raheja

Sr. Principal Scientist,

Machine Vision Lab,

Digital Systems Group,

CSIR-CEERI, Pilani,

Rajasthan,333031.

Abstract:

Today’s Monitoring systems have the tendency and the ability to detect and thus prevent thefts, burglaries and accidents. Lately a lot of research has been going on in making smart video surveillance systems that can not only prevent thefts but can also take care of the family and the whole house. Following the same trend, we a have built a C# based monitoring system which can detect various activities including suspicious ones like falling. This application can be later installed in homes where old or sick people live alone, so that if some kind of miss happening takes place immediate help can be reached. In order to detect and track the object of interest we have used Background Subtraction and Ellipse formation using blob moments. Activity analysis is later on handled with Ellipse parameters like θ (orientation), lengths and ratio of the axis’s as well as Motion History Image. After doing some statistical analysis we are able to detect activities like walking, standing, sitting, bending, lying and falling.

Acknowledgements

First off, we are grateful to Dr. Jagdish L. Raheja, our mentor in the project for giving us the opportunity to work under him. Without his constant checking in on our progress and giving us advice and tips it would have been an impossible task.

We are greatly indebted to Dr. Bharti Khungar, our PS instructor and Dean Practice School Division Bits Pilani for providing us a opportunity to work in CEERI.

We would like to thank Mr.Dheeraj Sangwan from the Machine Vision Lab in CEERI, who took interest in knowing about our work and constantly stays updated about our work.

We would like to show our gratitude to Ms. Som Shukla and Ms. Aradhana who are researchers in the Machine Vision Lab for their constant caring and support.

Last but not the least, we would like to express our heartfelt thanks to our parents, teachers, friends and all our well-wishers for their co-operations and inspirations, which they were always ready to extend.

- Aditya & Abhinav.

Table of Contents

List of Figures

Section 1: Introduction

1.1 Background

According to the Census 2011, India has a population [1] of 1,210,193,422 making India the second most populous country in the world. Elderly people (age 60+) account for 8-9% (i.e. around 2 billion) of our population [2], and that share is predicted to increase to over 19% of our population by 2050 by a report by the United Nations Population Division [3]. With the increasing number of elderly people the number elderly people living alone at homes will also increase. According to report [4] around 2-3 % of elderly men live alone while 7-8% of elderly women live alone. Around 15 million [5] elderly people live alone.

There are several severe side effects when elderly people live alone. They may become a target of Social Isolation, Depression which are linked to long term illnesses. Apart from these there are Medication Management Issues, Nutrition Issues, Home hazards and Injury, Housekeeping Issues, Financial Issues and the potential danger of falling victim to crime.

1.2 Motivation

Our work is mostly focused on finding a solution to the Home Hazards and Injury aspect of several potential dangers aforementioned above. Elderly people are prone to more damage [6] to seemingly small accidents than younger people i.e. a simple fall can prove to be fatal. Immediate rescue is really important in such cases since absence of medical care at the right time can lead to severe consequences.

Our Project once completed properly would be a handy tool in this respect, analyzing the movement continuously and alert the concerned people (Nearest Hospital, Related people etc.) if it suspects something suspicious like the person being unconscious or if the person falls down.

There has been work going on in this respect and there are quite some methods to deal with the issue. For example, the elderly person can call for help by pushing a button [7] but this would be useless if the person is immobilized or unconscious after the incident. Automatic wearable devices are more interesting as no human intervention is required. There have been methods based on accelerometers [8] & [9] which detect the magnitude and direction of the acceleration. Others are based on gyroscopes [10] which measure body orientation. The major drawbacks of the above techniques are that they involve wearing of sensors, which require batteries to operate and need to be replaced regularly for proper functioning. Video Surveillance offers a new and promising solution for elderly people’s monitoring since no wearable devices are required nor human intervention. A simple camera placed in each room at an appropriate position would be sufficient.

1.3 Objective

Our main aim is to be able to monitor the elderly person living alone at all times, and report to concerned people in case of a suspected emergency so that appropriate action can be taken. That is our system will be a fully automatic human monitoring system.

1.4 Report Organization

Section 2 will explain in detail the techniques we have adopted in our project and also explain the terms we use in the rest of the report. The Researcher is advised to skip this section and dive straight into Section 3 which talks about our proposed method. The layman is recommended to read Section 2 completely and then proceed to Section 3.

Section 4 is concerned with the implementation aspects of the project i.e. the platform, library used and the programming aspects.

Section 5 discusses the experimental results we got while testing the project in our laboratory.

Section 6 is where we conclude and also talk about the future work that can be carried in this field, improvements that can be made etc.

Section 2: Essential Information

2.1 Background Subtraction

Background Subtraction (BS) is one of the most used techniques for motion detection the others being optical flow, tracking and feature tracking. BS is actually a simple technique. The idea is to have a Background Model modelling the background, this can be either static (in case of Non-Adaptive BS), or can modified at runtime according to the new frames being received (Adaptive BS).

Non-Adaptive BS is really simple, the first frame is selected as the background model and then on all the frames are received are compared with the background model to find out the regions of motion. On the background subtracted image an appropriate threshold filter is applied to differentiate areas of motion from other areas. The end result would a black and white binary image with white at the places where there has been a motion.

Adaptive BS is also simple but somewhat more involved than Non-Adaptive BS. There are some techniques [11] as to how to update the background model. One of the widely used technique is Running Gaussian Average Method. The others being Median Filtering, Mixture of Gaussians, Mean-Shift based estimation and Eigen Backgrounds. The Running Gaussian Average will be explained in detail.

EMGU CV has the following classes [12] that implement BS:

a. IFBGDetector.

b. BackgroundDetector.

c. BackgroundDetectorMOG.

d. BackgroundDetectorMOG2.

2.1.1 Running Gaussian Average [11]:

In [13] have proposed to model the background independently at each (i, j) pixel location. The model is based on ideally fitting a Gaussian probability density function on the last n frames. In order to avoid fitting at every step a running average is stored to make it easier for calculating the new averages. The following formula is used to compute the running gaussian average.

$$\mu t = \alpha * It + (1 - \alpha) * \mu_{t\text{-}1}$$

$\mu t -$ new average.

$It -$ current pixel value.

α- empirical weight chosen as a trade-off between stability and quick update.

$\mu_{t\text{-}1}$– old average.

2.2 Blob Analysis

Blob – A blob is defined as connected component with similar properties with respect to color, intensity etc.

Blob Analysis is a fundamental technique of machine vision based on analysis of consistent image regions (blobs). As such it is a tool of choice for applications in which the objects being inspected are clearly discernible from the background. Diverse set of Blob Analysis methods allows to create tailored solutions for a wide range of visual inspection problems.

EMGU CV has the following classes [12] with respect to Blob Analysis

1. Emgu.CV.Cvb.CvBlobs
2. Emgu.CV.Cvb.CvBlobDetector

We make use of the method "Detect" from the Emgu.CV.Cvb.CvBlobDetectorClass to detect blobs and Emgu.CV.Cvb.CvBlobs to store the resulting blobs.

2.3 Motion History Image

Bobick and Davis [14] proposed a method to detect motion using temporal templates. The outcome was the Motion History Image, which is an image which shows the recent motion in the scene, and is based on a binary image sequence D(x, y, t) derived from the original image sequence I(x, y, t) . Every pixel of the Motion History Image H_τ is a function of the temporal history at that point.

$$H_\tau(x, y, t) = \tau \qquad \text{if } D(x, y, t) = 1$$

$$H_\tau(x, y, t) = \max(0, H_\tau(x, y, t-1) - 1) \qquad \text{otherwise}$$

Emgu CV has a class MotionHistory with methods for making Motion History Images. It has a method Update (…) which is mainly responsible for the construction of the MHI. In our case, we did not make use of the Motion History Class from EMGU CV, instead we made use of the definition of the MHI and made manipulations on the images accordingly so as to construct an MHI.

2.4 Image Moments.

In Image Processing, Computer Vision and related fields, an image moment is a kind of weighted average of the pixel intensities or a function of such moments. In pure mathematics, for a 2D continuous function f(x, y), moment [15] of the order (p + q) is defined as

$$M_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x^p * y^q * f(x, y)\, dx\, dy)$$

For digital images, which are discrete entities we use the discrete version of the above formula i.e.

$$M_{pq} = \sum_{x=0}^{w} \sum_{y=0}^{l} x^p * y^q * f(x, y)$$

Image moments have a lot of applications in calculating the centroid, area of the blob etc.

Area of the blob is calculated by calculating its zeroth moment i.e.

$$\text{Area} = M_{00}$$

Centroid of the blob is calculated by using the following formulae for its x and y coordinates respectively

$$X_c = \frac{M10}{M00} \qquad Y_c = \frac{M01}{M00}$$

M_{00} – Zeroth Image Moment

M_{01} and M_{10} – First Order Image Moments

Section 3: Our Approach

3.1 Video fetch

IP Cameras will be placed at appropriate positions in all the rooms, all of which will be connected to the internet via Ethernet or Wireless. Once the program is started on the server, it keeps sending HTTP requests to the IP Camera. The camera will then send a response in accordance with its current status and if the response is positive a stream of frames is sent to the server by the camera. We chose MJPEG stream over JPEG stream since in the former the number of overall requests sent will be much less as compared to the latter.

3.2 Background Subtraction

Once the server starts receiving the frames it sets the first frame as the Background Model, and from then on each frame is compared to the Background Model to find out if there has been a motion. The background subtraction algorithm that we have is adaptive There are other options available like Gaussian Background Subtraction which is adaptive, or Median Filtering. What the background subtraction does is that it thresholds the

We made use of the IFBGDetector<Hsv, byte>class for background subtraction. We chose the Hue Saturation Value (HSV) model instead of Blue Green Red (BGR) model because HSV was better at detection as it separates luma (image intensity) from chroma (Color information).

Since the Background Subtraction is non-adaptive it is important that initially there are no people and also whenever there are big changes to the room (like moving an almirah, sofa set etc.) the system should be re-started. The end result of Background Subtraction is the thresholded foreground image which shows the regions where movement has taken place. Certain filters are applied to the foreground image to smoothen it a little bit and to remove any noise. Gaussian and Erode filters were used for this purpose.

3.3 Biggest blob identification

From the smoothened, noise reduced foreground image blobs are detected using the classes Emgu.CV.Cvb.CvBlobs and Emgu.CV.Cvb.CvBlobDetector. But we filtered out all the blobs with areas less than some prefixed value (say 5000) and among those filtered only the biggest blob is considered. For more information about how blob identification is achieved by the library functions, please refer Section 2.2.

3.4 Ellipse fitting

Once the blob has been finalized, we make use of the blob's moments (Section 2.4) to find the appropriate parameters of an ellipse that fits the blob perfectly. The following formulae [16] were used for calculating the ellipse parameters.

$$\mathrm{Xc} = \mathrm{m10/m00}$$

$$\mathrm{Yc} = \mathrm{m01/m00}$$

$$\theta = (1/2)\tan^{-1}(\frac{\mathrm{b}}{\mathrm{a-c}})$$

$$\mathrm{w} = \sqrt[2]{6 * (\mathrm{a} + \mathrm{c} - \sqrt[2]{\mathrm{b}^2 + (\mathrm{a}-\mathrm{c})^2}\,)}$$

$$\mathrm{l} = \sqrt[2]{6 * (\mathrm{a} + \mathrm{c} + \sqrt[2]{\mathrm{b}^2 + (\mathrm{a}-\mathrm{c})^2}\,)}$$

Where a, b and c are defined as:

$$\mathrm{a} = \frac{\mathrm{m20}}{\mathrm{m00}} - \mathrm{Xc}^2$$

$$\mathrm{b} = 2 * (\frac{\mathrm{m11}}{\mathrm{m00}} - \mathrm{Xc} * \mathrm{Yc})$$

$$\mathrm{c} = \frac{\mathrm{m02}}{\mathrm{m00}} - \mathrm{Yc}^2$$

X_c – X-Coordinate of the center of the ellipse

Y_c – Y-Coordinate of the center of the ellipse

w – Length of Minor axis of the ellipse

l – Length of Major axis of the ellipse

and m_{ij}- is $(i + j)^{th}$ degree image moment(explained in Section 2.4).

By using w, l, X_c, Y_c, and θ we can construct the ellipse correctly.

The above calculated parameters are used to calculate ratio of minor to major axis, height, width and orientation of the object of interest. In order to identify all the postures correctly we maintain queues for both θ as well as ratio. As a result mean and standard deviation of last n values of θ and ratio can be calculated in each frame. By making use of these values along with the motion coefficient (C_{motion} from Section 3.5) the activity can be identified.

3.5 Motion History Image

For calculating the motion history image [14], we did not make use of the existing library class MotionHistory. Instead, we calculated the motion history image on our own, by making use of the definition of Motion History Image. We considered all the previous frames for this purpose. From the MHI constructed we find the motion coefficient (C_{motion}) by using the following formulae.

$$\mathrm{Sum} = \sum_{\mathrm{Pixel}(x,y)€\mathrm{blob}} \mathrm{I}(x, y)$$

Where I(x, y) is the value of intensity of the pixel (x ,y).

For programming ease we considered the sum of intensities of all the pixels and not just the blob because in the MHI the other pixels account to 0 anyway.

$$\mathrm{Cmotion} = \mathrm{Sum/total\_pixels\_in\_MHI}$$

C_{motion} is used in combination with the various parameters calculated in Section 3.4 to predict appropriate activity of the person of interest.

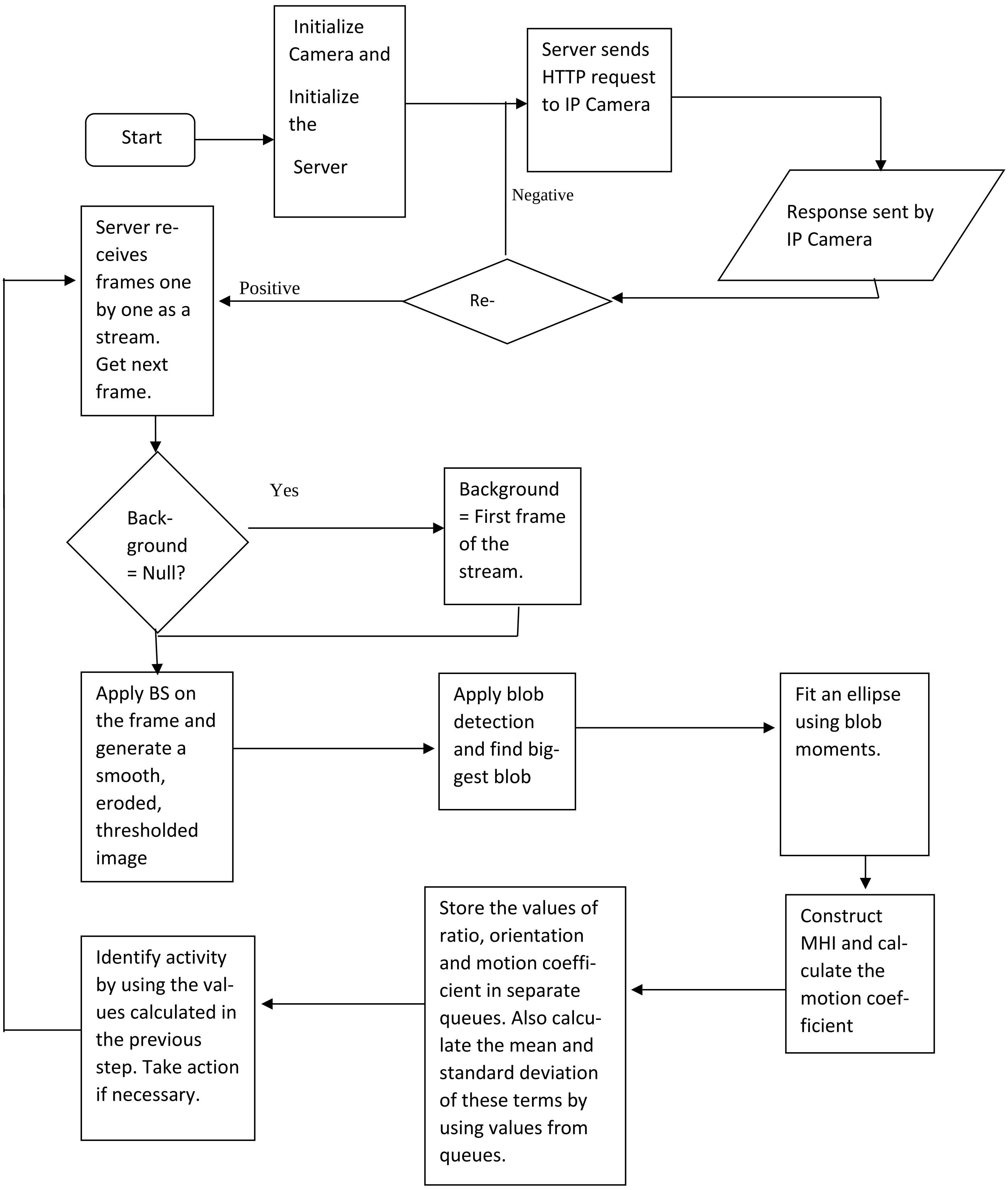
Start
Initialize Camera and Initialize the Server
Server sends HTTP request to IP Camera
Response sent by IP Camera
Re-
Negative
Positive
Server receives frames one by one as a stream. Get next frame.
Background = Null?
Yes
Background = First frame of the stream.
Apply BS on the frame and generate a smooth, eroded, thresholded image
Apply blob detection and find biggest blob
Fit an ellipse using blob moments.
Construct MHI and calculate the motion coefficient
Store the values of ratio, orientation and motion coefficient in separate queues. Also calculate the mean and standard deviation of these terms by using values from queues.
Identify activity by using the values calculated in the previous step. Take action if necessary.

Fig 1. Flow chart of the algorithm that we follow

3.6 Activity identification:

3.6.1 Standing

Standing is identified by an upright ellipse with no motion i.e. the motion coefficient will be very less and won't change by a large amount. The ratio of minor to major axis will be less, the orientation of the ellipse will be around 90^o and the motion coefficient will be less.

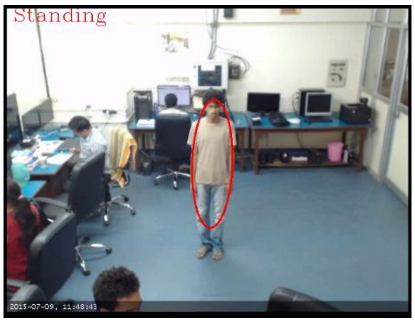

Fig2.Standing facing the camera

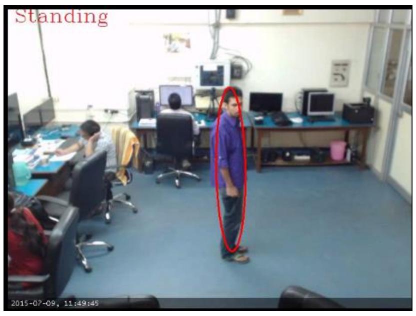

Fig 3. Standing perpendicular to the camera

3.6.2 Sitting

We have considered there will be not much motion after sitting, so sitting will be similar to standing in terms of motion coefficient. What changes is that the major axis of the ellipse decreases and correspondingly ratio of minor to major increases. These both facts are used together to find whether the person is sitting or not

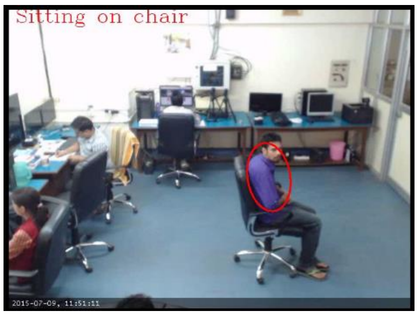

Fig 4. Sitting perpendicular to camera

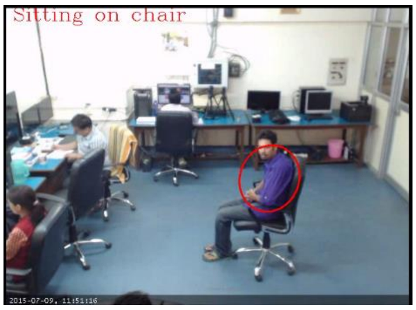

Fig 5. Sitting perpendicular to camera

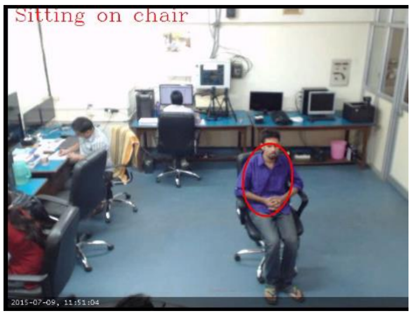

Fig 6. Sitting facing the camera

3.6.3 Moving

Moving is different from sitting or standing. The orientation of ellipse in Moving will be around 90^o just like standing but unlike standing the motion coefficient will be really high this time. In our scenario the motion coefficient value easily crosses 330, whenever there is any kind of motion.

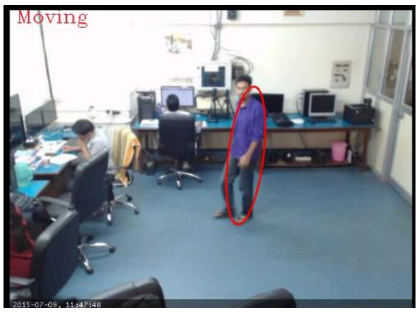

Fig 7. Moving

3.6.4 Falling

Falling is a bit different from all other activities. It is basically divided into two parts. Firstly the motion coefficient for a fall should be really high, something above 450 and secondly there should be a sudden and high change in orientation or the ratio of minor to major axis. Therefore when the standard deviation of either one of them shoots high and the motion coefficient is above 450 we can tell it is a fall.

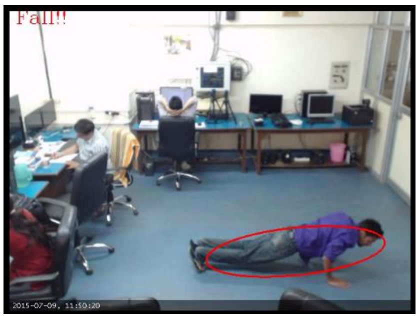

Fig 8. Fall perpendicular to camera

3.6.5 Lying

Lying is relatively easy to detect. Whenever the person is lying the orientation of the ellipse becomes horizontal, so if the value of the θ goes below 60^{o} we can safely assume that the person is lying. One more case we had to incorporate is when the person the lying in front of the camera, in that case the angel will does not change but the ratio does, as a result after some proper calibration we are able to detect lying in that case also.

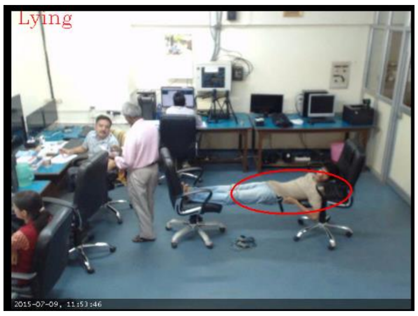

Fig 9. Lying perpendicular to camera

3.6.6 Bending

Bending is somewhat similar to sitting only. There in only one difference that although the ratio of minor to major axis increases the major axis of the ellipse almost remains same. So if there is not much change in the height and the ratio increase we detect it as Bending.

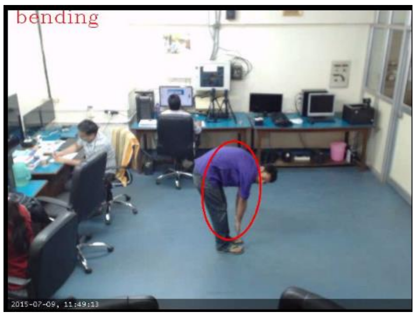

Fig 10. Bending perpendicular to camera

Section 4: Implementation

4.1 Platform

We implemented our program in C# by making use of the .NET Framework V4.0, the image processing library EMGU CV, and using the Visual Studio 2013 (Trial Version) as our Integrated Development Environment. In the beginning we used Aforge.NET Framework as our Image Processing Library but then shifted to EMGU CV because we found EMGU CV to be more robust. All our programs were run on a Dell Vostro 3800 Workstation equipped with an Intel i3 3.4GHz Processor, 8 Gb RAM with Windows 8 as the Operating System.

4.2 Tools used

4.2.1 Visual Studio

Microsoft Visual Studio [17] is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF applications, web applications, web sites and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver light.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer). Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++. 4.2.2 C – Sharp (C#)

4.2.2 C#

C# (pronounced "see sharp" or "C Sharp") is one of many .NET programming languages. It is object-oriented and allows you to build reusable components for a wide variety of application types Microsoft introduced C# on June 26th, 2000 and it became a v1.0 product on Feb 13th 2002.

C# is an evolution of the C and C++ family of languages. However, it borrows features from other programming languages, such as Delphi and Java. If you look at the most basic syntax of both C# and Java, the code looks very similar, but then again, the code looks a lot like C++ too, which is intentional. Developers often ask questions about why C# supports certain features or works in a certain way. The answer is often rooted in it's C++ heritage. Recent language features, such as Language Integrated Query (LINQ) and Asynchronous Programming (Async) are not necessarily unique to C#, but do add to it's uniqueness.

An important point is that C# is a "managed" language, meaning that it requires the .NET Common Language Runtime (CLR) to execute. Essentially, as an application that is written in C# executes, the CLR is managing memory, performing garbage collection, handling exceptions, and providing many more services that you, as a developer, don't have to write code for. The C# compiler produces Intermediate Language (IL), rather than machine language, and the CLR understands IL. When the CLR sees the IL, it Just In Time (JIT) compiles it, method by method, into compiled machine code in memory and executes it. As mentioned previously, the CLR manages the code as it executes.

Because C# requires the CLR, you must have the CLR installed on your system. All new Windows operating systems ship with a version of the CLR and it is available via Windows Update for older systems. The CLR is part of the .NET, so if you see updates for the .NET Framework Runtime, it contains the CLR and .NET Framework Class Library (FCL). It follows that if you copy your C# application to another machine, then that machine must have the CLR installed too.

Instead of a runtime library (such as APIs for file I/O, string handling, etc.) being dedicated to a single language, .NET ships with a .NET Framework Class Library (FCL), which includes literally tens of thousands of reusable objects. Since all .NET languages target the CLR with the same IL, all languages can use the FCL. This shortens the learning curve for any developer moving from one .NET language to another, but also means that Microsoft is able to add many more features because there is only one FCL, rather than a separate implementation for common features in every programming language. Similarly, 3rd party software vendors can write managed code that any .NET developer, regardless of language, can use. In addition to all of the services you would expect of a runtime library, such as collections, file I/O, networking, etc., the FCL includes the APIs for all of the other .NET technologies, such as for desktop and Web development.

4.2.3 .NET Framework

Microsoft .NET Framework is one of the most vital and emerging technological platform to any professional associated with Microsoft technologies. The .NET Framework is undergoing with rapid changes and enhancements to empower its developers to deliver world class software solutions. The .NET Framework is the first Microsoft development environment designed from the ground up for Internet development. Although .NET is not meant to be used exclusively for Internet development, its innovations were driven by the limitations of current Internet development tools and technology.

The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has following two main components:

Common Language Runtime (CLR)

The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness

Framework Class Library

The next component or layer of the .NET Framework is the .NET Framework base classes. The purpose of this layer is to provide the services and object models for data access and manipulation, data streams (input/output [I/O]), security, thread management, and more. In many respects the Windows API (Application Programming Interface) has been abstracted into these base classes. These libraries are an object-oriented collection of classes that can be reused and enhanced to enable rapid application development. The classes support the creation of applications that range from ASP.NET Web pages and Web Services to traditional Windows and command line applications.

4.2.4 EMGU CV

Emgu CV is a cross platform .NET wrapper to the Open CV image processing library. Allowing Open CV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc. The wrapper can be compiled by Visual Studio, Xamarin Studio and Unity, it can run on Windows, Linux, Mac OS X, iOS, Android and Windows Phone.

Emgu CV is written entirely in C#. The benefit is that it can be compiled in Mono and therefore is able to run on any platform Mono supports, including iOS, Android, Windows Phone, Mac OS X and Linux. A lot of efforts has been spent to have a pure C# implementation since the headers have to be ported, compared with managed C++ implementation where header files can simply be included. But it is well worth it if you see Emgu CV running on Fedora 10! Plus it always gives you the comfort knowing that your code is cross-platform

Section 5: Experimental Results

All our experimental testing was done in the Machine Vision Lab itself by both of us. Other employees from the lab also volunteered to help us in testing.

Standing, Moving and Fall were pretty accurately detected with around 90-95% accuracy, even lying was detected accurately with around 90-93% accuracy. We have had some problems with Sitting on Chair and Bending with respect to the parameters we chose i.e. the parameters we chose (C_{motion}, theta, ration etc.) were not enough to differentiate these tasks from the others. We are looking into other techniques to employ and improve the robustness of our program and make it somewhat near to 100% correct.

Section 6: Conclusion and Future Work

Conclusion

This application was built with an aim for making a monitoring system that can detect activities and respond whenever there is a suspicious movement. As of now we are able to detect walking, standing, lying and falling precisely. Sitting and Bending are still not so robust. We are also able to differentiate if the person is just lying or is lying after a fall. This way we can tell if the person is unconscious after the fall.

Future Work

In the future we plan to work on and improve our program so that we include methods so that detection can take place at night (with the help of infrared imaging etc.). We plan to include motion tracking in our code, so that when we make our Background Subtraction adaptive we adapt only the objects and not human beings.

The project can be extended to deal with multiple persons in the same room and give status of each of them

References

[1] – Official Census Report of India

http://censusindia.gov.in/2011-prov-results/indiaatglance.html

[2]-Official Census Report – Age-wise Statistics in India in the year 2011.

http://www.censusindia.gov.in/vital_statistics/SRS_Report/9Chap%202%20-%202011.pdf

[3] United Nations Population Division Report submitted in 2013.

http://www.un.org/esa/socdev/documents/ageing/Data/WorldPopulationAgeingRe-port2013.pdf & http://www.un.org/esa/population/publications/worldage-ing19502050/pdf/111india.pdf

[4] – Article about elderly living alone in India .

http://mospi.nic.in/mospi_new/upload/elderly_in_india.pdf

[5] –Newspaper article on the condition of elderly living alone in India along with some sta-tistics, October 2014.

http://timesofindia.indiatimes.com/india/15-million-elderly-Indians-live-all-alone-Census/ar-ticleshow/43948392.cms

[6] - http://nihseniorhealth.gov/falls/causesandriskfactors/01.html

[7]-Direct Alert (2010) Wireless emergency response system

URL: http://www.directalert.ca/emergency/help-button.php

[8] - Kangas, M., Konttila, A., Lindgren, P., Winblad, I. & Jämsä, T. (2008). Comparison of low-complexity fall detection algorithms for body attached accelerometers, Gait & Posture: 28(2):285–291.

[9] - Karantonis, D., Narayanan, M., Mathie, M., Lovell, N. & Celler, B. (2006). Implementa-tion of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring, IEEE Transactions on Information Technology in Biomedicine 10(1): 156–167.

[10] - Bourke, A. & Lyons, G. (2008). A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor, Medical Engineering & Physics 30(1): 84–90.

[11] – Piccardi, M. (n.d.). Background Subtraction techniques: A review. 2004 *IEEE Interna-tional Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583).*

[12] – EMGU CV Documentation

http://www.emgu.com/wiki/files/3.0.0/document/html/8dee1f02-8c8a-4e37-87f4-05e10c39f27d.htm

[13] - C. Wren, A. Azarhayejani, T. Darrell, and A.P. Pentland, "Pfinder: real-time tracking of the human body," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 78g785, 1997.

[14] - Bobick, A. & Davis, J. (2001). The recognition of human movement using temporal templates, IEEE Transactions on Pattern Analysis and Machine Intelligence 23(3): 257–267.

[15] – Wikipedia – Image Moments

https://en.wikipedia.org/wiki/Image_moment

[16] – Rocha, L., Velho, L., & Carvalhi, P. (n.d.). Image moments-based structuring and tracking of objects. *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing.*

[17] - Wikipedia - Visual Studio

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[18] - Emgu Wiki Page

http://www.emgu.com/wiki/index.php/Main_Page