International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:9, 2008

Implementation of a Motion Detection System

Asif Ansari[1], T.C.Manjunath (Ph.D., IIT Bombay)[2], C.Ardil[3]

***Abstract*—**In today's competitive environment, the security concerns have grown tremendously. In the modern world, possession is known to be 9/10'ths of the law. Hence, it is imperative for one to be able to safeguard one's property from worldly harms such as thefts, destruction of property, people with malicious intent etc. Due to the advent of technology in the modern world, the methodologies used by thieves and robbers for stealing have been improving exponentially. Therefore, it is necessary for the surveillance techniques to also improve with the changing world. With the improvement in mass media and various forms of communication, it is now possible to monitor and control the environment to the advantage of the owners of the property. The latest technologies used in the fight against thefts and destruction are the video surveillance and monitoring. By using the technologies, it is possible to monitor and capture every inch and second of the area in interest. However, so far the technologies used are passive in nature, i.e., the monitoring systems only help in detecting the crime but do not actively participate in stopping or curbing the crime while it takes place. Therefore, we have developed a methodology to detect the motion in a video stream environment and this is an idea to ensure that the monitoring systems not only actively participate in stopping the crime, but do so while the crime is taking place. Hence, a system is used to detect any motion in a live streaming video and once motion has been detected in the live stream, the software will activate a warning system and capture the live streaming video.

***Keywords*—**Motion, Detection, System, Video, Crime, Matlab, Surveillance.

I. Introduction about vieo surveillance

IF you consider video in the simplest of terms, video surveillance began with simple closed circuit television monitoring (CCTV). As early as 1965, there were press reports in various countries across the world suggesting police use of surveillance cameras in public places. When video-cassette recorders hit the market, video surveillance became really popular.

Analog technology using taped video-cassette recordings meant surveillance could be preserved on tape as evidence. A complete analog video-surveillance system consisted of a camera, monitor, and VCR. The old tube camera was only useful in daylight, and the VCR could only store eight hours of footage at best. The drawback was that after a while, owners and employees of such a system would become complacent and not change the tapes daily or the tapes would wear out after months of being re-used. There was also the problem of recording at night or in low light. While the concept was good, the technology hadn't yet peaked. The next step was the Charged Coupled Device camera (CCD), which used microchip computer technology. In the 1990's video surveillance made great strides in practicality by the introduction of digital multiplexing. When digital multiplexer units became affordable, it revolutionized the surveillance industry by enabling recording on several cameras at once (more than a dozen at time in most cases) [1].

Three key factors brought on the popular use of the digital video recorder. They are

- The advancement in compression capability, allowing more information to be stored on a hard drive. (Round-the-clock surveillance produces a lot of information.)
- The cost of a hard drive, which has dropped dramatically in recent years.
- The storage capacity of a hard drive, which has increased dramatically in recent years.

Digital video surveillance made complete sense as the price of digital recording dropped with the computer revolution. Rather than changing tapes daily, the user could reliably record a month's worth of surveillance on hard drive. The images recorded digitally were so much clearer than the often grainy images recorded with analog that recognition was immediately improved for identification purposes. Digitally stored images can also be enhanced in various ways (add light, change colors, reverse black and white) to make crucial determinations. With videotape, what you see is what you get.

The paper is organized as follows [17]. A brief introduction to the surveillance system was presented in the previous paragraphs. The requirements of the video surveillance are depicted in brief in section 2. Motion detection in live video stream is presented in section 3, followed by the work specification in section 4. The study and analysis of the work is presented in section 5. Section 6 describes the motion detection algorithm. Various types of graphical user interfaces developed for this work is presented in section 7. Results and discussions is presented in section 8, followed by the conclusions in section 9 [17].

[1] Asif Ansari is currently, a Lecturer in the Dept. of Information Technology of Thakur College of Engg. & Tech., Kandivili, Mumbai, Maharashtra, India.
[2] T.C. Manjunath is currently, Professor & Head in Electronics and Communications Engineering Dept.
[3] C.Ardil is with the National Academy of Aviation, AZ 1056 Baku, Azerbaijan.

II. REQUIREMENT OF VIDEO SURVEILLANCE

While it is important to understand the various places video surveillance can be used it is also important to asses the risks involved in the protection of a certain item. In the recent years, as more and more items such as art are gaining importance, the prices of such things are also going through the roof. Therefore, technology has come in the forefront for protection and surveillance of such goods and items. When assessing risk one of your inputs should be theft statistics. The following are the statistics of thefts in places such as shops, residences and in public places in our country, India in a particular year. Of the 50 reported thefts in one year, the breakage of thefts can be shown as the following [2]:

- Display Cases 19
- Open displays 10
- Pictures 04
- Other displays 02
- At night 06
- From stores 02
- Long timescale 04
- Other 03

This means that even though technology has improved dramatically in the past few decades, it has still a long way to go. It is clearly seen from the statistics that although the focus of security surveillance is on gaining evidence against crimes and thefts, the thought process should change to stopping thefts and crimes while they are in progress.

III. MOTION DETECTION IN LIVE VIDEO STREAM

When all is said and done, surveillance systems should be a reflection of the real world we live in. As people become more and more security savvy, they will demand real protection for their property. The new digital video systems will have to raise that security to a new level. They should make the customers feel good. Scare off a few troublemakers. And those who do try to beat the system should face a far greater risk of getting caught. Hence, the new digital video surveillance systems should be able to provide a high sense of security. The peace of mind can only be achieved when the person is assured that he will be informed of any thefts of his property while they are in progress. He would also feel more secure if he can be guaranteed that the surveillance system that he uses will not only give him evidence against the perpetrators but also try to stop the thefts from taking place in the first place. Therefore, to achieve such kind of security Motion Detection in the live video stream is implemented. The motion detection systems will not only be monitoring the areas of interest but will also keep an active lookout for any motion being produced.

IV. WORK SPECIFICATION

Aim: In our project we have aimed to build such a surveillance system, which can not only detect motion, but will

a) Warn the user of the intrusion and
b) Record the footage of the video from the moment the motion was detected.

Coding Language: To fulfill our aim, we have used a strong computing software called Matlab 7 [3].

Advantage of Matlab: Basically the advantage of using Matlab is that Matlab is an interpreted language for numerical computation. It allows one to perform numerical calculations, and visualize the results without the need for complicated and time consuming programming. Matlab allows its users to accurately solve problems, produce graphics easily and produce code efficiently.

Disadvantage of Matlab: The only problem with Matlab is that since Matlab is an interpreted language, it can be slow, and poor programming practices can make it unacceptably slow. If the processing power of the computing machine is low the Matlab software takes time to load and execute any code making the code execute very slowly.

Reason for Selection: We used Matlab to develop our work, because Matlab provides Image Acquisition and Image Processing Toolboxes which facilitate us in creating a good GUI and an excellent code.

Approach: Using a video input object, live data is acquired and analyzed to calculate any motion between two adjacent image frames. Any motion in the image stream is plotted in a MATLAB figure window as shown in Fig. 2.

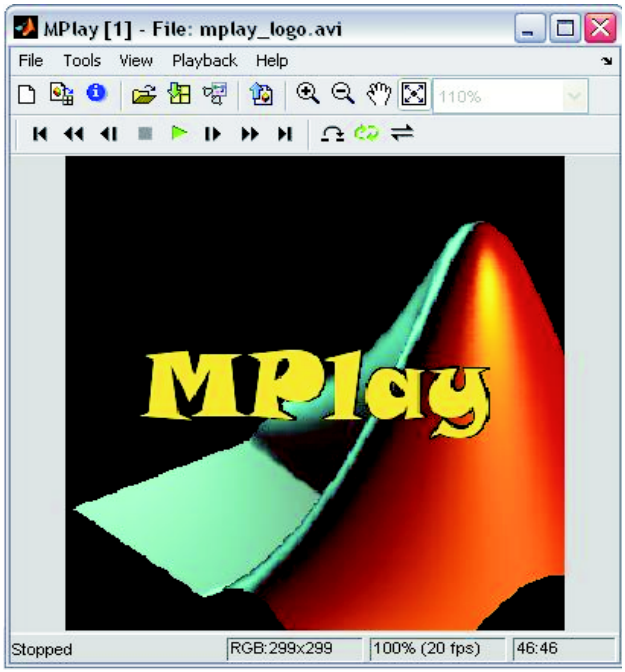

Fig. 1 M-Play figure file function

V. STUDY AND ANALYSIS

The objective of this work was to develop a surveillance system which would detect motion in a live video feed and if

motion is detected, then to activate a warning system and store the video feed for future reference and processing purposes. The activation of an alarm would help in nullifying a threat of security and storing of video provides a proof of such malicious activity. Keeping the work objective in mind, we firstly developed basic system architecture as shown in the Fig. 2.

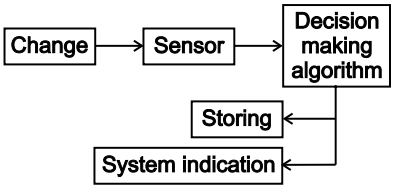

Fig. 2 A basic system architecture of our system

The system architecture, which we developed, describes how the system component interacts and work together to achieve the overall system goals. It describes the system operation, what each component of the system does and what information is exchange. The architecture was designed for basically getting an idea of how the actual system works and operates [4].

A. System architecture functioning

The system architecture is going to function in following way:

- Capturing the live video feed through a web cam : To detect motion we first have to capture live video frames of the area to be monitored and kept under surveillance this is done by using a web cam which continuously provides a sequence of video frames in a particular speed of FPS (frames per second).
- Comparing the current frames captured with previous frames to detect motion: For checking whether any motion is present in the live video feed, we compare the live video frames being provided by the web cam with each other so that we can detect changes in these frames and hence predict the occurrence of some motion..
- Storing the frames on the memory if motion is detected : If motion is being detected, we would require storing such motion so that the user can view it in the near future. This also helps the user in providing a legal proof of some inappropriate activity since a video coverage can be used as a proof in the court of law.
- Indicating through an alarm when the motion is detected : The user may want to be notified immediately that there has been some intrusion detected by the software, hence an alarm system is included in the software. This alarm system immediately activates a WAV file format audio alarm signal if any kind of motion is detected hence. This helps in preventing any kind of breach of security at that moment of time.

B. Selection criteria of the tasks

Our work is motion based change detection in .avi video format. Before beginning with the work, one of the important tasks was deciding the various tasks required to implement the work. Therefore, we performed a brain storming session and decided various important tasks which would be required in completion of the work such as:

- Analysis and study of the problem definition,
- Deciding the requirements of the system being developed,
- System architecture containing the following sub function:
  - Capturing,
  - Comparing
  - Storing and
  - Indication of motion
- Developing the code and
- Documentation.

After deciding the various important tasks in our work, we decided that the platform on which we are going to develop our code will be Matlab. We choose Matlab because various video acquisition and analysis functions are pre-defined in Matlab that would make the development of our work much easier. Finally, just before we started developing the code, we designed a rough GUI and created a design, which would suit our needs and perform all activities, which were desired by us and would be easier to use by anybody.

C. Motion detection

1) RATIONALE

The detection of motion essentially requires the user to perform two major steps. They are: foremost step is to setup the hardware for acquiring the video data in which the motion is to be detected and the later step is to actually device an algorithm by which the motion will be detected. The AVI video format is actually an interleave of Audio and Video. The video stream is stored or acquired as a series of frames occurring in an ordered sequence one after the other [5].

2) ACQUISITION SETUP

The Matlab programming language is used to store data in the form of matrices. Therefore Matlab can provide quick interface with data matrices. The software provides for frame acquisition from hardware devices such as web cams or digital cameras as long as the devices are correctly initialized by the programmer. Therefore, in order to allow quick setup with the image acquisition devices, Matlab Function directory provides a host of predefined functions by which the user can inquire about the various different devices currently connected and then setup the required device with Matlab so that it can acquire and store data at run time.

VI. MOTION DETECTION ALGORITHM

The Matlab interface allows the user to define the commands to be performed at the run time. Once the user setup of the video source is complete the algorithm comes into play. The algorithm is built to take advantage of the strength of Matlab i.e. to store data as a form of matrices. The frames acquired are stored in the Matlab directory as matrix in which each element of the matrix contains information about the pixel value of the image at a particular location. Therefore, the pixel values are stored in the workspace as a grid where every element of the matrix corresponds to an individual pixel

value [6].

Since Matlab considers each matrix as one large collection of values instead of a bunch of individual values it is significantly quicker in analyzing and processing the image data. The algorithm hence checks each frame being acquired by the device with the previously acquired frame and checks for the difference between the total values of each frame. A threshold level is set by the user with which the difference of values is compared. If the difference exceeds the threshold value the motion is said to be detected in the video stream. The various codes used in this work are shown below.

CODE :

```
% --- Outputs from this function are returned
to the command line.

function  varargout  =  New1_OutputFcn(hObject,
eventdata, handles)

varargout{1} = handles.output;

guidata(hObject, handles);
set(handles.Mstart,'enable','off');
set(handles.Mstop,'enable','on');
set(handles.Vstart,'enable','off');
set(handles.Vstop,'enable','off');
set(handles.Mstop,'UserData',0);

vid = videoinput('winvideo');
handles.vid = vid;
set(handles.vid,'FramesPerTrigger',50);
[filename, pathname] = uiputfile('*.avi');
aviobj = avifile(filename,'fps',25);
set(handles.vid,'TriggerRepeat',Inf);
triggerconfig(handles.vid, 'Manual');

guidata(hObject, handles);

global pr
if(pr==10)
    pr=1;
end

start(handles.vid);
trigger(handles.vid);

y = (getdata(handles.vid,1,'uint8'));

count = 0;
countbck = 0;
cntsnap = 1;
while 1
   trigger(handles.vid);

    yprev = y;

 if get(handles.Mstop,'UserData')
     global lp
     if lp==0
         lp = lp + 1;
         z= imread('nomot.tif');
         aviobj = addframe(aviobj,z);
     end
     aviobj = close(aviobj);
     stop(handles.vid);
     delete(vid);
     clear vid;
     break
 else

     y = (getdata(handles.vid,1,'uint8'));
     diff = abs(y-yprev);
     diff = abs(y-yprev);
     abs_img = mean(diff(:));
     axes(handles.axes1);
     subimage(y);

     if abs_img >
str2num(get(handles.editSens,'string'));
             count = count + 1;
     else
         countbck = countbck +1;
     end
     if count >= 3;
             global lp

             lp = lp + 1;
             for i=1:5
             z=getimage(handles.axes1);
             aviobj = addframe(aviobj,z);
             end
             global pr
             if pr == 1
                 global filename1
                 [t,Fs] = wavread(filename1);
                 player = audioplayer(t,Fs);
                 play(player);
                 global pr
                 pr = 10;
             end
     end
 end
end

axes(handles.axes1);
cla;
subimage(handles.S);
delete(handles.vid);

clear handles.vid;
imaqreset;
clear handles.axes1;
guidata(hObject, handles);

% --- Executes on button press in Mstop.

function Mstop_Callback(hObject, eventdata,
handles)

guidata(hObject, handles);
set(handles.Mstart,'enable','on');
set(handles.Mstop,'enable','off');
set(handles.Vstart,'enable','off');
set(handles.Vstop,'enable','off');

set(handles.Mstop,'UserData',1);

guidata(hObject, handles);
```

FUNCTION EXPLANATION

a) Function Name:

The Function name Mstart is executed as the Monitor button in the GUI is pressed by the user. It takes the value of the GUI from the user and updates it in the workspace.

b) Set:

The set command is used to change the looks and controls available to the user in the GUI. It is used to change the value of the buttons and is also used to prevent the user from pressing buttons which cannot logically occur again. As the number of buttons that can be pressed by the user reduces, the amount of confusion in the users mind will also reduce as the process will be self guiding thereby reducing the number of errors or bugs and to ensure that user's experience is hassle free [7].

c) Video Input:

The video input command is used to setup the video source for the rest of the program to be run.

```
% Syntax Checking

requestedDevice =
hwInfo.DeviceInfo(infoIndex);
deviceFileOK =
requestedDevice.DeviceFileSupported;
defaultFormat = requestedDevice.DefaultFormat;
supportedFormats =
requestedDevice.SupportedFormats;
checkSupported = strcmpi(formatType,
supportedFormats);

% Based on the syntax called, determine the
  format option and extract any PV pairs
  present.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [newDeviceFile, fileFlag] =
localAddPathToFile(deviceFile)
% Adds the path to the device file if none are
  present.

% Initilaize.

fileFlag = true;
newDeviceFile = deviceFile;

% Check to see if a path was already present

[pathStr, fileName, fileExt] =
fileparts(deviceFile);
if isempty(pathStr),

% If no file extension was provided either,
% must assume it's not a device file.

    if isempty(fileExt)
        fileFlag = false;
    end

% Try to add a path and extension (via WHICH).

If file is

try
    pathLocation = which(deviceFile);
    if ~isempty(findstr(pathLocation,
fullfile('matlab', 'ops'))),
        pathLocation = '';
        end
    catch
        pathLocation = '';
    end
    if ~isempty(pathLocation),
        newDeviceFile = pathLocation;
        fileFlag = true;
    end
end
```

The codes presented above are snippets of the processes performed by Matlab when the video input command is called into play. The initial four commands are given to inquire about the presence and status of the camera that has been stated in the program. The subsequent function is carried out if Matlab is able to connect and initialize the specified device. If the connection is successfully established Matlab just sets its input location to the device, which has been initialized, and then keeps on catching the frames as input data from the device [8].

d) Uiputfile:

The Uiputfile function is used to allow the user to define the name and storage space of the output file of the video. This function is essential for two major reasons:

i) It allows the Matlab to save the file exactly where the user specifies thereby ensuring the user can easily find the storage location.
ii) It allows the user to name the file thereby allowing him to keep a record of each and every file without the chances of any previous record being overwritten.

e) Aviobj:

The AVI object command is used to create an object file of the type AVI. The AVI file is a standard video format with a predefined method of encryption. Therefore, a class file is already present in Matlab and the AVI object file defines an instance to create the AVI file in which the motion is being stored. The object created is set to the filename specified by the user in the previous function.

f) Get:

The get function is used to interface with the GUI file. It checks the status and returns the current value of the GUI button as specified.

g) Start Vid:

The Start function is used to start the video acquisition device to get the frame from the device object.

h) Stop Vid:

The Stop function is very important in the video acquisition device. The Start function begins the video stream entering as input to Matlab. The stop function will stop this input. If the function is not used, the video stream continues in the path already started by Matlab. If the user will try to use it again, Matlab will not be able to start it again as the path will be busy. It will therefore stop the reusability of the program unless the whole of Matlab reinitializes [9].

i) Delete Vid:

The delete function is used to delete the temporary frames stored by Matlab in the object file. This function will free up the workspace as well as enable the function to reuse the pathname.

j) Imaqreset:

The imaqreset function is very important as it resets the acquisition device altogether. It ensures that the frame buffer in the object is free and completely new at the time the device restarts. It also resets the device therefore ensuring that there is no device present for acquisition and the device can be used for other uses.

A. VIDEO, AUDIO, HELP AND GUI:

This section describes about the further development of the video and the audio units along with the help and the graphical user interfaces.

1) VIDEO

The software produces an AVI video file as it monitors the area in view. Irrespective of the fact that most modern operating systems would provide various different software's to play video files in AVI format, the user should be able to view the file without having to switch programs and searching for it. Hence, it is of the utmost importance that there should be a video player that plays the video stream that has been produced.

CODE

```
% --- Executes on button press in Vstart.
function Vstart_Callback(hObject, eventdata,
handles)
% hObject handle to Vstart (see GCBO)
% eventdata reserved - to be defined in a
  future version of MATLAB
% handles structure with handles and user data
  (see GUIDATA)

guidata(hObject, handles);

set(handles.Mstart,'enable','off');
set(handles.Mstop,'enable','off');
set(handles.Vstart,'enable','off');
set(handles.Vstop,'enable','on');

handles.output = hObject;
[filename2] = uigetfile('*.*');
mplay(filename2);
```

Explanation of the video code :

a) Uigetfile:

The uigetfile function is used to retrieve a file from the hard drive. The function is used here to ensure that the user has the option : choose the movie file that the user wants to playback. Although it may seem illogical since only one file is created in each instance, the user may want to keep a track of other files in the record or may want to play a previously recorded video.

b) Mplay:

The mplay function is a built in video player in the Matlab library function. It can play a host of multimedia files as well as be called from the Matlab command prompt. It has a Graphical User Interface to play the video file thereby allowing the user to stop, play as well as forward and rewind the file. The mplay function is predefined and encrypted in Matlab to be run with the various programs created by the user. The source code for the player is protected [10].

2) AUDIO

In order for the software to act as a surveillance system it is important to provide a mechanism to raise an alarm in case motion is detected in the video stream. However, conversely, stealth may be required in a few cases where alarms may prove more harmful. Therefore, an alarm function is required which will allow the user to choose the audio function as per his requisites.

CODE

```
function Alarm_Callback(hObject, eventdata,
handles)

button_state = get(hObject,'Value');

if button_state == get(hObject,'Max')
    icon5=imread('Volume.png');
    set(handles.Alarm,'CData',icon5)
    global filename1
    [filename1] = uigetfile('*.wav');
    global pr
    pr=1;

elseif button_state == get(hObject,'Min')
    icon5=imread('RedVolume.png');
    set(handles.Alarm,'CData',icon5)
    global pr
    pr=0;
end

% --- Executes Alarm.
[t,Fs] = wavread(filename1);
player = audioplayer(t,Fs);
play(player);
```

Explanation of the audio code :

a) Get:

The get function is used to interface with the GUI file. It checks the status and returns the current value of the GUI button as specified.

b) Global:

The global function is used to specify that the global variable is being called into play. Matlab ensures that every variable is local only to its local function to reduce complexity and to reduce conflicts between similarly named variables. The global function makes the variable accessible to all functions in the program to allow different functions to change variables according to the necessity [11] .

c) Waveread:

The wav read function is a Matlab function to read and store audio files in the wave audio format. It can search and find the first RIFF chunk of data in the file. It then opens the

file for the wave player and searches for the next subsequent chunks of data. It does not open the subsequent chunks but just reads the type of chunk that is present and forwards it to the player.

d) Audioplayer:

The audio player function is an audio playing file, which can play a host of audio signals in Matlab. It essentially initializes the audio file sent to it by the waveread or auread function and creates an audio signal object which returns the number of bits each signal takes up [12].

e) Play:

The play function starts and runs the audio signal object created in the audio player function. It plays each audio sample as provided in the audio file as it enters in the audio player. The following are snippets of the play function of the audio player. It checks for any present errors and if not, plays the audio sample.

```
if ~isa(obj, 'audioplayer')

error('MATLAB:audioplayer:noAudioplayerObj',
...
audioplayererror('MATLAB:audioplayer:noAudiopl
ayerObj'));
end
 error(nargchk(1, 2, nargin, 'struct'));
if isempty(varargin)
    play(obj.internalObj);
else
    play(obj.internalObj,
double(varargin{:}));
end
```

3) HELP

The help function is a prerequisite for any good software to ensure that the user can use each and every function of the program. The help file of any software should be detailed with examples or instructions about using the software to improve user's interaction with the software.

CODE

```
% --- Executes on button press in Vstop.

function Vstop_Callback(hObject, eventdata,
andles)

guidata(hObject, handles);

handles.output = hObject;

S = imread('welcome.tif');
handles.S = S;
axes(handles.axes1);
subimage(S);
mdhelp

function varargout = mdhelp(varargin)

TEX('CALLBACK',hObject,eventData,handles,...)
calls the local function named CALLBACK in
TEX.M with the given input arguments.

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',
mfilename, ...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@tex_OpeningFcn, ...
                   'gui_OutputFcn',
@tex_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end
 if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% --- Executes just before tex is made
visible.

function tex_OpeningFcn(hObject, eventdata,
handles, varargin)

guidata(hObject, handles);

function varargout = tex_OutputFcn(hObject,
eventdata, handles)

varargout{1} = handles.output;
```

Explanation of the help :

Mdhelp:

The mdhelp is a completely new gui file which acts as a popup when called as a function. It contains a static text box that acts as a frame in which there are instructions stored about how to use the given software. The text frame contains a step-by-step guide to running and interfacing with the software to help the user make various decisions about the options he wants exercise.

GUI

The modern operating systems allow for almost every program to run using visual icons and Interfaces. Hence, most users would be put off from using software's that are completely text based to run. Matlab provides the programmer with Matlab GUIDE which is a tool for generating user interfaces for the programs [13].

CODE

```
function varargout = New1(varargin)

% Begin initialization code-DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
                       mfilename,...
```

```
                   'gui_Singleton',
                      gui_Singleton,...
                   'gui_OpeningFcn',
                      @New1_OpeningFcn,...
                   'gui_OutputFcn',
                      @New1_OutputFcn,...
                   'gui_LayoutFcn',[],...
                   'gui_Callback',[]);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
    str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
     gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% --- Executes just before New1 is made
   visible.

function New1_OpeningFcn(hObject, eventdata,
handles, varargin)

% Choose default command line output for New1

imaqreset
handles.output = hObject;

icon1=imread('movies.png');
icon2=imread('stop.png');
icon3=imread('imovie.png');
icon4=imread('help.png');
icon5=imread('RedVolume.png');

set(handles.Mstart,'CData',icon1)
set(handles.Mstop,'CData',icon2)
set(handles.Vstart,'CData',icon3)
set(handles.Vstop,'CData',icon4)
set(handles.Alarm,'CData',icon5)

S = imread('welcome.tif');
handles.S = S;
axes(handles.axes1);
subimage(S);
global lp pr qw
lp=0;
pr=0;
qw=0;
Mon_Callback(hObject, eventdata, handles);
guidata(hObject, handles);

function varargout = New1_OutputFcn(hObject,
eventdata, handles)

varargout{1} = handles.output;

% --- Executes on button press in Mon.

function Mon_Callback(hObject, eventdata,
handles)

state of Mon
imaqreset
handles.output = hObject;
clear vid;
clear log;

S = imread('welcome.tif');
handles.S = S;
axes(handles.axes1);
subimage(S);
set(handles.Mstart,'enable','on');
set(handles.Mstop,'enable','off');
set(handles.Vstart,'enable','off');
set(handles.Vstop,'enable','on');

% --- Executes on button press in Cap.

function Cap_Callback(hObject, eventdata,
handles)
state of Cap
set(handles.Mstart,'enable','off');
set(handles.Mstop,'enable','off');
set(handles.Vstart,'enable','on');
set(handles.Vstop,'enable','on');

function editSens_Callback(hObject, eventdata,
handles)

function editSens_CreateFcn(hObject,
eventdata, handles)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

VII. Developed Graphical User Interfaces

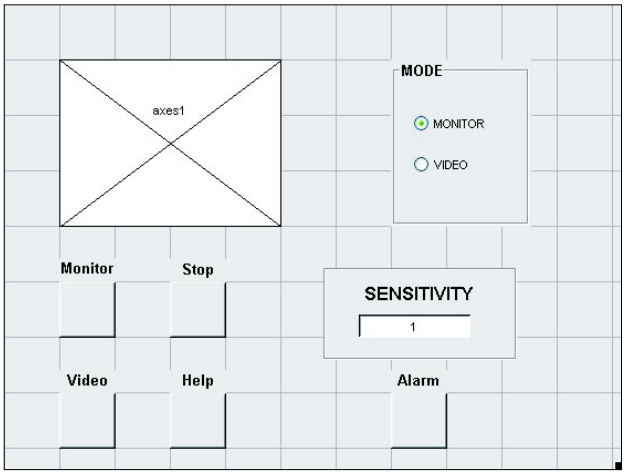

Fig. 3 Main GUI window used for monitoring purposes

FUNCTION EXPLANATION

a) Initialization:

The beginning lines are meant for initializing the program file before the function opens and runs. It sets the type of GUI by specifying if single or multiple instances can run at the same time [14].

b) Opening Function:

The opening function is run just before the GUI is made visible to the user. It contains the various instructions to initialize the figure file so that it can run as a cohesive unit with the program.

c) Figure File:
The figure file is a screenshot of the figure as created without the fringe icons and background. The initialization code ensures that the final interface looks better and is more appealing to the user.

d) Imread:
The Imread function is the Matlab function, which can read image files and store their values in a variable. The variables created are matrices containing the RGB pixel value of images.

e) Axes:
The axes function corresponds to the various axis figures present in the figure file. It sets up the axis with the corresponding output required.

f) Sub image:
Matlab provides the sub image function to output images. This function returns the image instead of the mathematical equivalent of the pixel values [15].

g) Callback Function:
The callback function is the auto-generated function that is called every time the user uses the corresponding interface. The callback function contains values that change every time a particular callback occurs. The callbacks are generated for buttons. Toogle buttons, Radio buttons, Dynamic strings etc.

SYSTEM MAINTENANCE & HOW IT WORKS

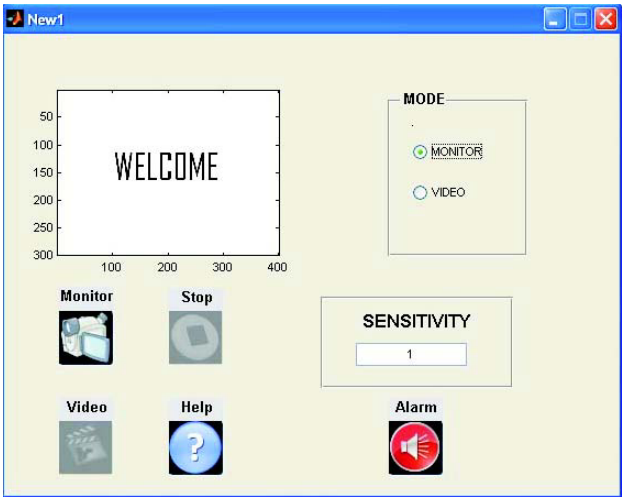

Fig. 4 GUI-1

Initially, on executing the code, the above GUI shown in Fig. 4 will get opened and displays the various icons, an axis and a sensitivity tool bar. This information displayed corresponds to the various features, which we have incorporated in our software. Each icon fulfills a specific feature the axis displays the video feed from the web cam an the sensitivity tool is used to manipulate the sensitivity of the software to detect motion.

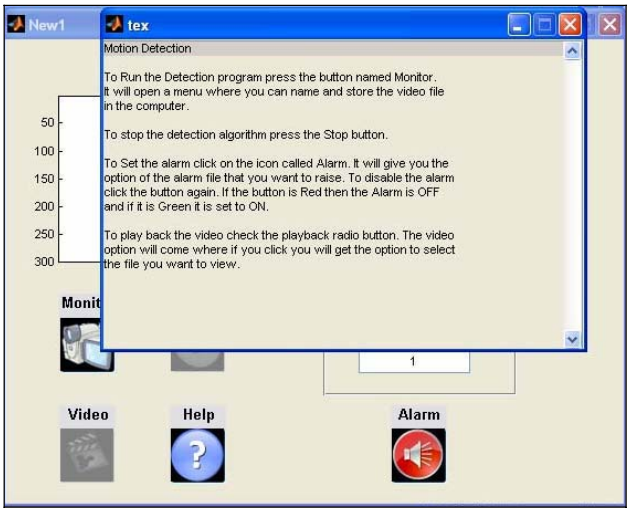

Fig. 5 GUI-2

On clicking the HELP icon a help menu as shown in Fig. 5 is displayed which gives general directions in a very simple language on how to use the GUI. We tried to use a very simple language so that any user even a layman can understand can use the software easily.

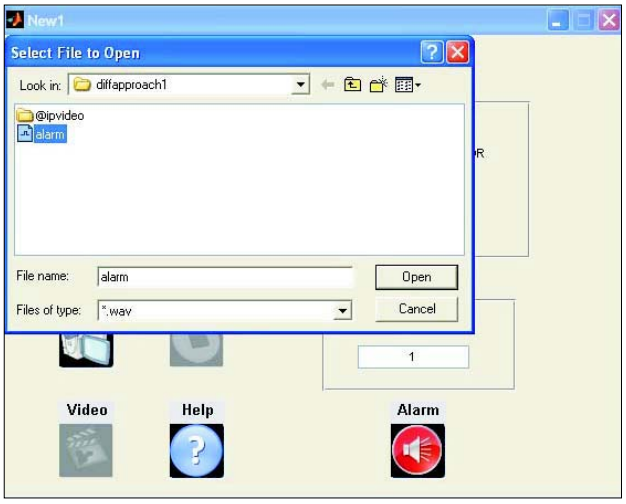

Fig. 6 GUI-3

We have given user the facility to use an alarm signal which will be triggered when a motion is detected as shown in the Fig. 6. This can be done by using the alarm icon on left bottom side of the GUI. On clicking this icon a user can direct the path to any .WAV file which he wants to hear as alarm. Once the alarm is set the icon turns into a green colour as shown in the Fig. 7.

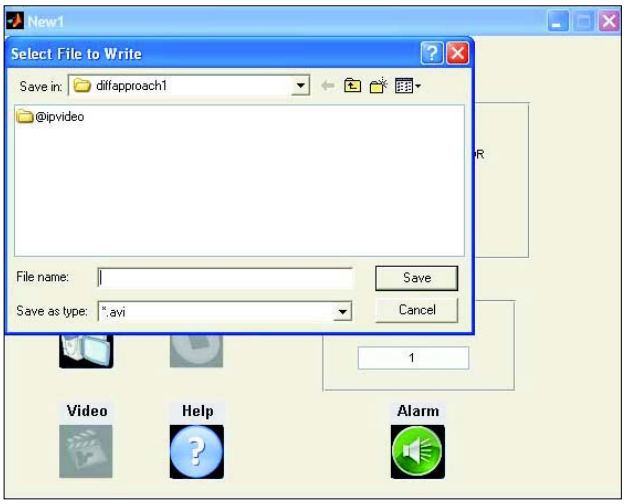

Fig. 7 GUI-4

To start monitoring the system we first click on the monitor check button on top left corner of the GUI (default is set on monitor). Now, we click the monitor icon on the middle right side of the GUI. A window as shown above will pop up on clicking this icon [16].

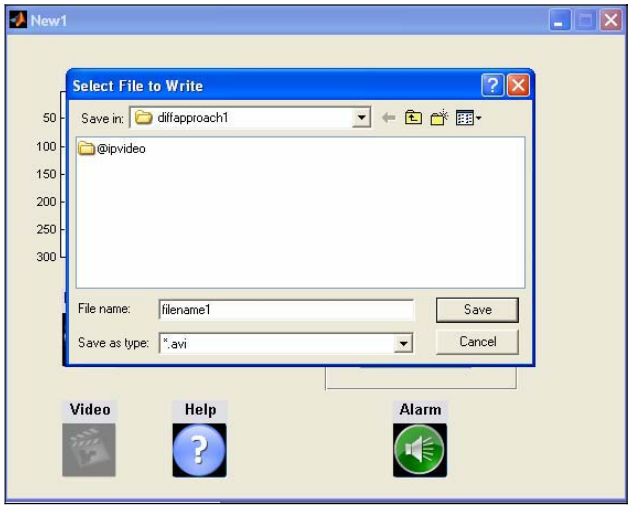

Fig. 8 GUI-5

When this window pops up on clicking the monitor icon, a user can give any filename by which he wants to save the video file press save after the filename is given as shown in the Fig. 5. If the motion is detected the video file will be saved under this filename and even if no motion is detected that information will be stored under this filename.

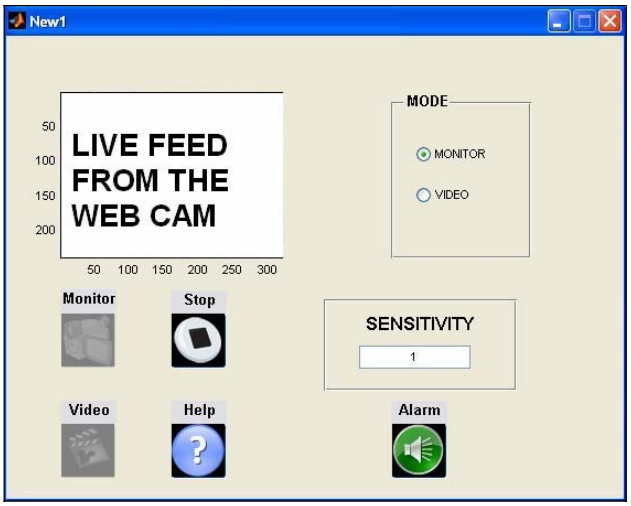

Fig. 9 GUI-6

After we have given the filename the monitoring starts and the video feed is shown from the web cam as shown in the Fig. 9. There will be no externally triggered activity till there is no motion once a motion is detected the software will trigger the alarm and start storing the video under the filename specified by the user. To stop the storing of the video the user has to click the stop icon in the right middle of the GUI as shown in the Fig. 9.

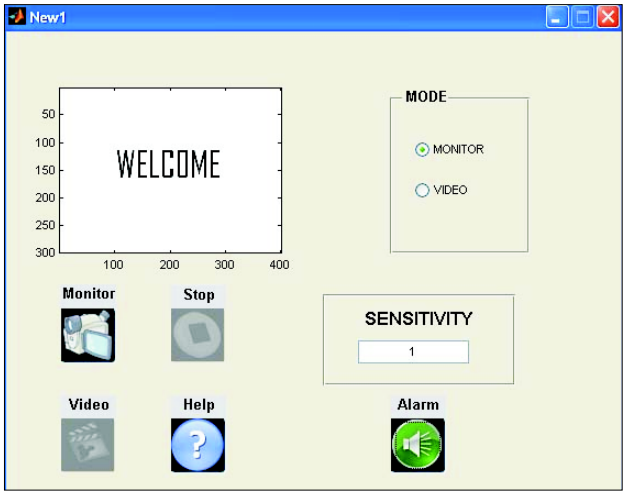

Fig. 10 GUI-7

After pressing the stop icon the GUI comes in the default view and the video display from the web cam is stopped. Now to view the video the user has to click the video check button on the left top of the GUI.

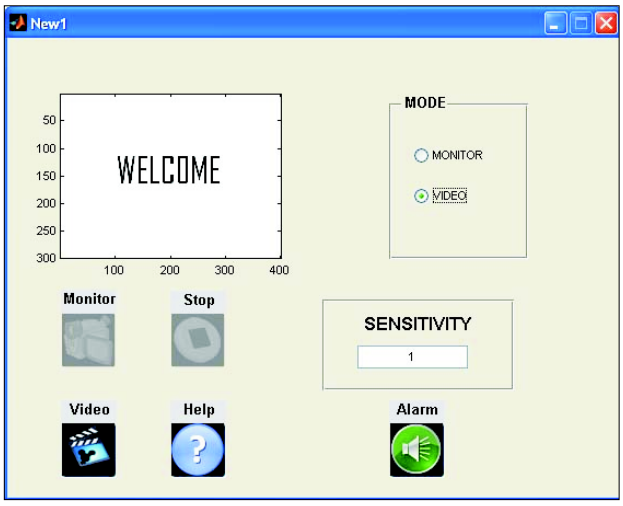

Fig. 11 GUI-8

After the user has clicked the video check button video icon on the right bottom side of the GUI is highlighted as shown in the Fig. 11. To view the video the user has to click this icon.

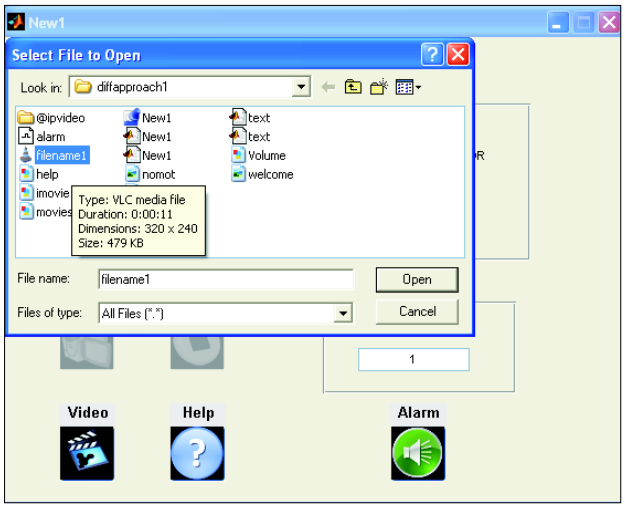

Fig. 12 GUI-9

On clicking the video icon the above window will pop up as shown in the Fig. 12 which will show the folder inside which the videos are saved.

The video which you want to view can be executed by clicking on that video file's filename and then clicking the open button of the pop up window.

If any motion had been detected the mplayer will pop up and the user can view the video of the surveillance by clicking the play (90 degrees turned triangle) button on the mplayer as shown in the Fig. 13.

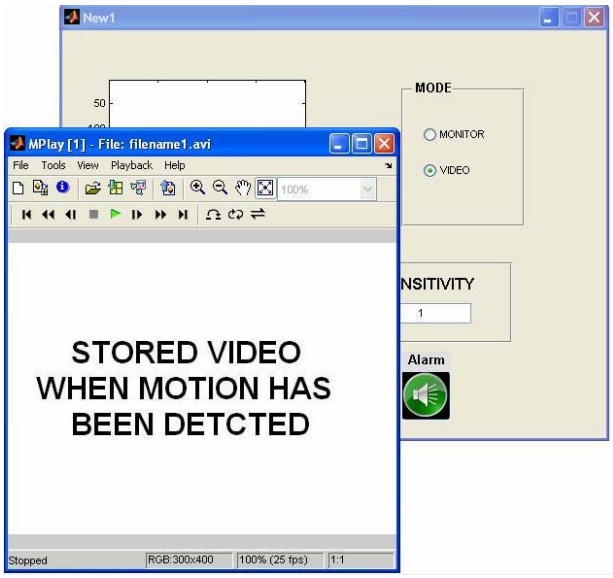

Fig. 13 GUI-10

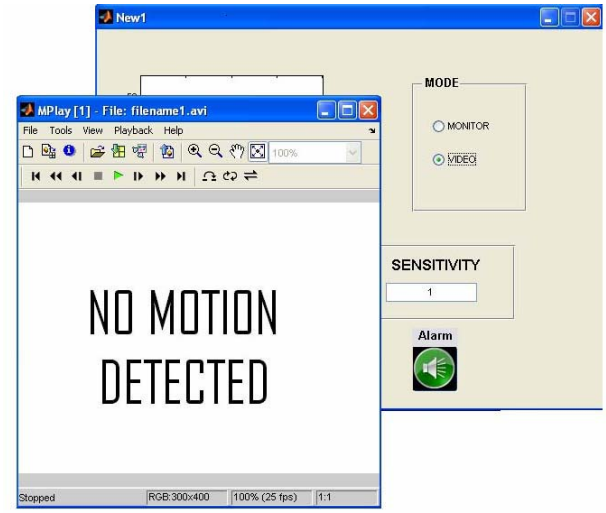

Fig. 14 GUI-11

If no motion has been detected, then the mplayer displays the above information as shown in the Fig. 14.

VIII. RESULTS AND DISCUSSION

The graphical user interfaces developed shows the effectiveness of the surveillance method in the work. This has got many features and of course some limitations also, which are discussed below.

Features included : We have included various important features in our work

- We have developed a GUI in our code which allows a user to use our software with ease and efficiently.
- Our software can be integrated and used with any company manufactured web cam.

- We have provided the user a facility to use any audio (.Wave) file as alarm signal. If the user wants he can use the software without the alarm audio signal.
- The user can store the recorded video after the motion has been detected on any place in the hard disk.
- We have used icons instead of usual buttons in our GUI to make a layman user more comfortable in using our software.
- Only one instance of our software can run at a single time hence reducing confusion due to multiple instances of same software running.
- The stored video is viewed in a Matlab media player which allows all the features of a media player and hence we can forward, rewind, pause etc a stored video.

Limitations:

- More features could have been added but due to time constraint they could not be added. But this is not an issue since the code can be directly extracted and manipulated to include new features.
- The code is dependent on Matlab compiler without which the code would not run but this is not a problem since acquiring Matlab is not difficult.

IX. CONCLUSION

A video monitoring & detection system was thus developed successfully in this paper. This system mainly provides an efficient method for surveillance purposes and is aimed to be highly beneficial for any person or organization. Thus, a motion based change detection in avi video format was completed and successfully implemented. The future scope of the work done could be as follows: the due course of time as we started to understand the minute details of our work, we significantly realized that our software would be tremendously important in the future world. Following changes or additions can be done on our work to include some new features.

- With the existing alarm system, advancement can be included and SMS can be sent to the user when motion is detected.
- The stored video can be automatically transferred to some email account so that an extra backup data can be used.
- A user_id and password can be given to a user so that unauthorized people don't have access to the software.
- A facility for the user can be given where he can mainly monitor only a small specific area in the range of the web cam.
- In the future, the user can be provided a remote access to this software from some remote PC through internet.
- Include an option to take snaps periodically, manually or automatically.
- Work could be done to make the system more users friendly for a layman user

REFERENCES

[1] Duane C. Hanselman and Bruce L. Littlefield, "Mastering Matlab 7".
[2] Google search.
[3] Yahoo search engine.
[4] www.w3schools.com.
[5] www.mathworks.com.
[6] www.matlab.com.
[7] Rozinet, O. and Z. Szabo, "Hand motion detection using Matlab software environment".
[8] Nehme, M.A.; Khoury, W.; Yameen, B.; Al-Alaoui, M.A., "Real time color based motion detection and tracking*", Proc. ISSPIT 2003, 3rd IEEE International Symposium on Signal Processing and Information Technology, 2003*, 14-17 Dec. 2003 , pp. 696 – 700, 14-17 Dec. 2003.
[9] Josué A. Hernández-García, Héctor Pérez-Meana and Mariko Nakano-Miyatake, "Video Motion Detection Using the Algorithm of Discrimination and the Hamming Distance", Lecture Notes in Computer Science, Springer-Verlag, Germany.
[10] H.A.M. El_Salamony, H.F. Ali, and A.A. Darweesh, "3D Human Body Motion Detection and Tracking in Video", Proc. Acta Press.
[11] Song, Y.,"A perceptual approach to human motion detection and labeling", *PhD thesis*, California Institute of Technology, 2003.
[12] Yilmaz, A., M. Shah, "Contour Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras", *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.
[13] Borst, A. and Egelhaaf, M., "Principles of visual motion detection", *Trends in* Neurocience, Vol. 12, pp. 297-305, 1989.
[14] Wachter, S. and H.H. Nagel, "Tracking persons in monocular image sequences," *Proc. Computer Vision and Image Understanding*, Vol. 74, pp. 174-192, 1999.
[15] Gavrila, D., "The visual analysis of human movement: A survey," *Proc. Computer Vision andImage Understanding*, Vol. 73, pp. 82-98, 1999.
[16] Motion detection with image acquisition toolbox, *Mathworks*, Matlab.
[17] Prasad Nadkarni, Abhinav Semwal, Vikas Singh, "Motion based change dectection in .avi format", *B.E. Thesis*, Thakur College of Engg. & Tech., Kandivili (E), Mumbai-101, Maharashtra, India, 2007.