DYNAMIC SCENE ANALYSIS ON ABANDONED OBJECT DETECTION

By

SHIVAM NAGAR - 051234

DEEPAK SHARMA – 051242

SACHI MAHAJAN - 051259

MAY-2009

Submitted in partial fulfillment of the

Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, **"Dynamic Scene Analysis on Abandoned Object Detection"** submitted by **Deepak Sharma, Sachi Mahajan** and **Shivam Nagar** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science from Jaypee University of Information Technology has been carried out under my supervision. This work has been actually submitted wholly to Infosys Technologies Limited as part of the internship curriculum.

[illegible] 1/06/09

Satish Chandra
Computer Science & Information Technology
Jaypee University of Information Technology

ACKNOWLEDGMENT

We would first express our thanks to our university, Jaypee University of Information Technology along with Infosys Technologies Limited for giving us the opportunity to come and work on project and having faith in us. We are extremely thankful to our project lead, Dr. Mohan S, Technical Evangelist, E&R Mysore who came up with an incredible idea which is very useful for security surveillance. Without him, this project would not have been possible. We express our gratitude to him for mentoring and encouraging us, time to time. We would also like to take pleasure in thanking our technical associate, Sreekanth Hanumanthu, E&R Mysore for help and support throughout. Their feedback, comments and suggestions were helpful throughout the study.

Also, an important mention of Sr. Lecturer, Satish Chandra and S. P. Ghrera, Head of Department, Computer Science & Information Technology, Jaypee University of Information Technology for granting us all the required permissions and helping us in coordinating assistance, supervision and valuable input.

Lastly, special thanks to the complete Infosys family for their assistance during the preparatory phase and unselfish support in learning.

TABLE OF CONTENTS

1. LIST OF FIGURES

2. ABBREVATIONS

Open CV	Open Computer Vision
Img	Image
AOD	Abandoned Object Detection
RAM	Random Access Memory
HDD	Hard Disk Drive
DBMS	Data base Management System
SQL	Structured Query Language
N.A.	Not Applicable
GB	Giga Bytes
OS	Operating System
WWW	World Wide Web

3. ABSTRACT

This document provides the comprehensive architectural and design overview of the system, using a number of architectural views and design methodologies to depict different aspects of the system. It is intended to capture and convey the significant architectural designs which have been used to build the system. The overall design of the system has been made using the Top-Down design strategy, where the design starts with a top-level description of a system and then refine this view step by step. With each refinement, the system is decomposed into lower-level and smaller modules. Top-down decomposition requires identifying the major higher-level system requirements and functions, and then breaking them down in successive steps until function-specific modules can be designed.

The project is to identify and abstract an Abandoned Object from a scene. It can be applied in Video surveillance as security system to monitor the activities of the people. The goal is to notify a human operator who monitors the video surveillance system about potentially critical events such as unobserved objects (abandoned objects) placed in public areas covered by security cameras. The operator will then decide how to proceed based on the information provided by the system. The scope is limited to the videos with no camera motion and has no multiple abandoned objects, hence it is assumed that there is no motion in video capture device and there in only one abandoned object in the scene at any point of time.

4. MAIN BODY

4.1. HIGH LEVEL DESIGN ARCHITECTURE

4.1.1. Subsystems

This section describes the four main component modules or subsystem of the System. The system has the following sub modules:

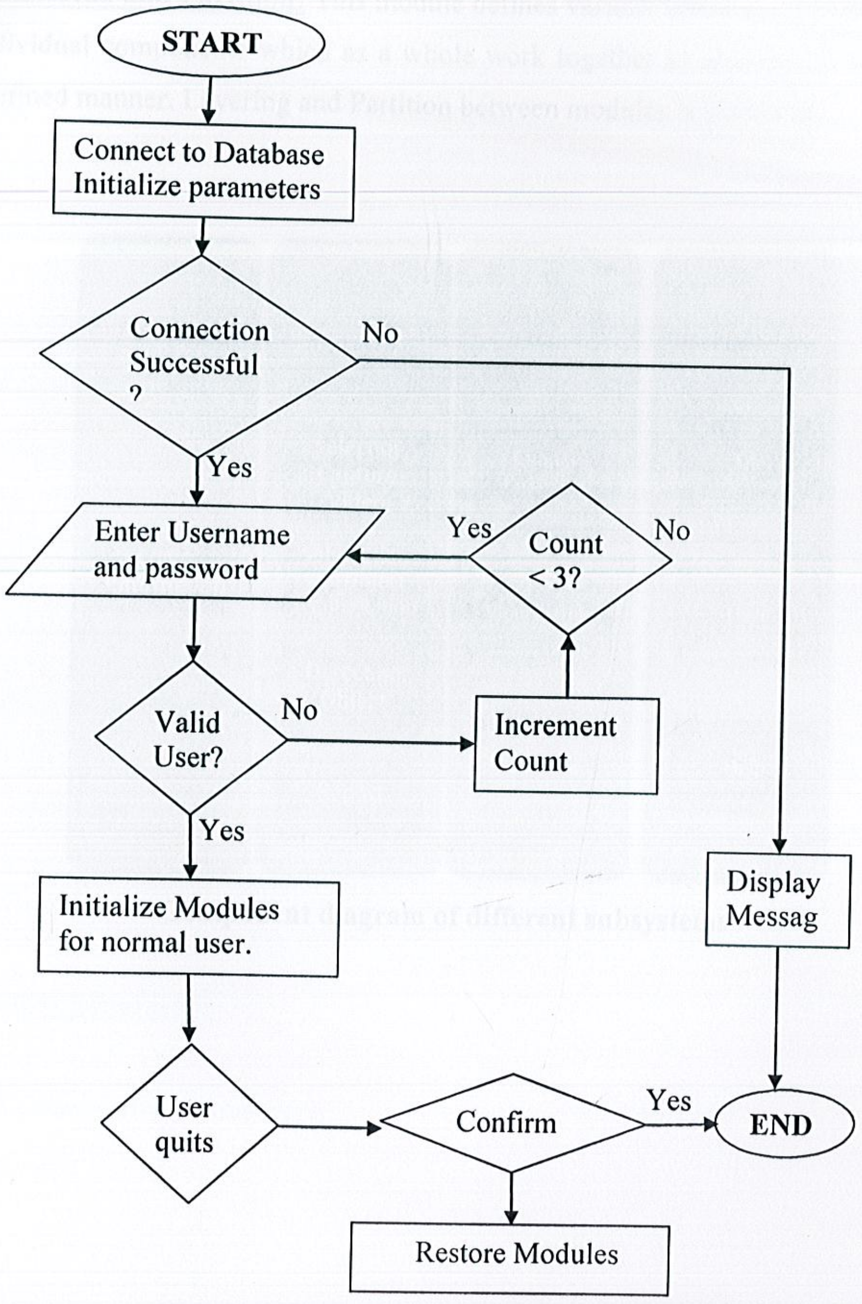
Profile: This module provides basic information like Name of the user who is currently logged in, number of abandoned objects reported and the number of times the user has logged in.

Video Processing Module: This module performs the entire tasks required for video processing. Sub modules, which are as under:

- **Video Capturing:** Capture the video stream from a capture device or file stored on disk.
- **Frame Selection:** Select currently captured frame for processing.
- **Subtracting scene Image from Frame:** Select only those pixels that exist exclusively in captured frame with respect to the scene.
- **Object Segmentation:** Defines whether the object introduced in video stream is to be classified as abandoned or not.
- **Database reporting Module:** Makes a database entry for details of each suspicious object, and the details of video source.
- **Frame Details:** If object is found abandoned its edges are defined, colored and overlapped with the actual video frame till object is in frame.

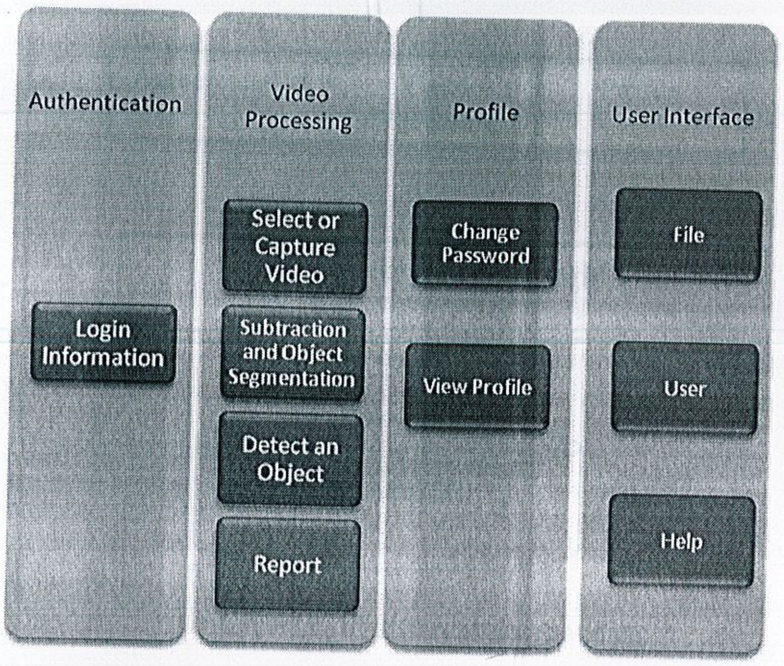
User Interface Module: This module describes the user interface of the application.

Authentication Module: This module prompts the user to enter a login ID and password. Starting up application requires user to authenticate him as a valid user, users actions will be logged into database. If an invalid username or password is provided by user they are requested again for three times. If user is unable to get him validated the application will shut down. On authentication user will initiate the camera or provide application with a video file to process upon. This module checks what options to provide to user, i.e. Configure the application with respect to the privileges granted to user account. This can be shown as under:

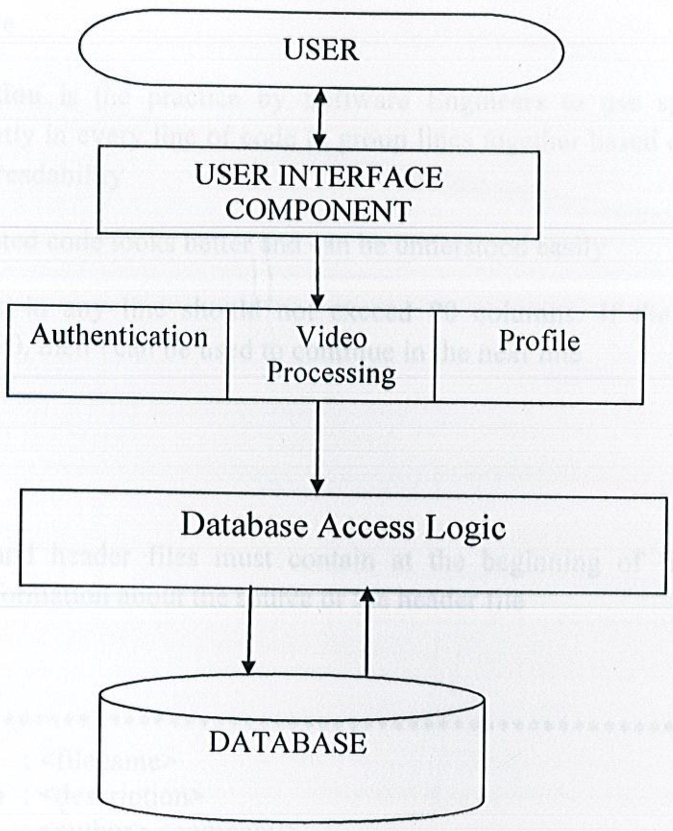


Activity Diagram for Authentication Subsystem

4.1.2. Layering and Partition: This module defines various layers and partition between the individual components which as a whole work together to accomplish the task in a well defined manner. Layering and Partition between modules is shown as:



Component diagram of different subsystems



Layering of different components

4.1.3. Coding Standards:

General Commenting Guidelines

- The ratio of code to comments should be 10 : 3
- Whenever there is a block of code which is doing something complex, sufficient amount of comments should be put in to explain
- Comment should not be in the same line as the code
- Use both C and C++ type comments but single line comments should not be used if Code is to complex and require multi line comment

Indentation of Code

- **Indentation** is the practice by Software Engineers to use spaces or tabs consistently in every line of code to group lines together based on their scope for easy readability
- An indented code looks better and can be understood easily
- The code in any line should not exceed 80 columns. If the column size exceeds 80, then \ can be used to continue in the next line

File Header Block

- All source and header files must contain at the beginning of file, a section providing information about the source or the header file
- **Format:**

```
/***********************************************************
* File          : <filename>
* Description   : <description>
* Author        : <author> <company>
* Version       : <version number>
* Date          : <Date>
***********************************************************/
```

File Footer Block

- All files should have this footer at the end of the file
- **Format:**

```
/***********************************************************
* End of <filename>
***********************************************************/
```

Function Header Block

- **Format:**

```
/*******************************************************
* Function: <Function Name>
* Description: <Overview of the function>
* Input Parameters:
*        <Parameter 1> – <brief description>
*        <Parameter 2> - <brief description>
*        ... ... ... ... ... ... ...
* Returns : <Return values both in case of success and
*   Error conditions if the function returns something>
*******************************************************/
```

4.1.4. Development Environment

Processor	: AMD Athlon 64 Dual Core 2.60 GHz
RAM/HDD	: 2 GB/160 GB
Operating System	: Windows Vista Enterprise .
Platform	: Windows.
Development tool	: Microsoft Visual Studio 2008, Open CV
DBMS	: Oracle

4.1.5. Deployment Environment

4.1.5.1. Hardware Requirement

Processor/RAM/HDD (Optimal)	: Windows NT Kernel/2 GB/160 GB
Web server	: N.A.
Database Server	: Oracle

4.1.5.2. Software Requirements

OS for Web server	: Windows
OS for Database Server	: Windows
DBMS	: SQL Server / Oracle
Third Party S/Ws	: Open CV

Function: DB_Error_Reporting ()

Parameters	char * cpUser, char * Error
Return Values	1 on success, 0 on connection error
Description	Makes entry to AOD_ACTIVITY table, maintaining a log of errors occurred so far.
Functions called by this function	None.

Function: DB_change_password ()

Parameters	char * cpUser, char *cpOldPass, char *cpNewPass
Return Values	1 on success, 0 on connection error, -1 on wrong userID, failure, -2 on wrong Password, -3 on any other error.
Description	Enables the user to change the password by updating AOD_Login table.
Functions called by this function	None.

Function: DB_NoOfUserLogin()

Parameters	char * cpUser
Return Values	1 on success, 0 on connection error
Description	the number of LOGIN on success, 0 on connection error, -1 on any other error
Functions called by this function	None.

Function: DB_NoOfAbonObj ()

Parameters	char * cpUser
Return Values	number of abandoned objects detected on success, 0 on connection error, -1 on any other error
Description	Calculates the total number of abandoned objects detected by a user
Functions called by this function	None.

Event: OnBnClickedOk()

Parameters	None
Return Values	None
Description	Contains the events that will occur when OK is pressed.
Functions called by this function	None.

Event: OnBnClickedLocation()

Parameters	None
Return Values	None
Description	Contains the events that will occur when "Location" is pressed.
Functions called by this function	None.

Event: OnBnClickedPreview()

Parameters	None
Return Values	None
Description	Contains the events that will occur when "Preview" is pressed.
Functions called by this function	None.

Event: OnBnClickedPlay()

Parameters	None
Return Values	None
Description	Contains the events that will occur when "Play" is pressed.
Functions called by this function	None.

Event: OnBnClickedPause()

Parameters	None
Return Values	None
Description	Contains the events that will occur when "Pause" is pressed.
Functions called by this function	None.

Event: OnFileLogOff()

Parameters	None
Return Values	None
Description	Logs off
Functions called by this function	None.

Function: LoadVideo()

Parameters	None
Return Values	None
Description	Function to refresh data on progress bar.
Functions called by this function	None.

Function: ShowVideo()

Parameters	Void * on
Return Values	0 – on success
Description	function for thread for video Preview
Functions called by this function	None.

Video Processing Module

Function: cvCaptureFromCAM()

Parameters	None
Return Values	Captured video stream. NULL on error.
Description	Initiates capturing of video stream from the default capture device.
Functions called by this function	cvCaptureFromCAM() : captures video from camera.

Function: cvCaptureFromFile()

Parameters	None
Return Values	Captured video stream. NULL on error.
Description	Initiates capturing of video stream from Hard Disk.
Functions called by this function	cvCaptureFromFile() OR cvCaptureFromAVI() : captures video from File.

Event: OnHelpAboutAOD()

Parameters	None
Return Values	None
Description	Shows the about AOD dialog.
Functions called by this function	-

Event: OnHelpAboutUs()

Parameters	None
Return Values	None
Description	Shows the about Us dialog.
Functions called by this function	-

Event: OnFileLogOff()

Parameters	None
Return Values	None
Description	Event to occur when "Logs off" button is pressed.
Functions called by this function	-

Function: fnLoadVideo()

Parameters	None
Return Values	None
Description	Loads the video and fetches its properties to the dialog.
Functions called by this function	None

Event: OnFileOpenVideoFile()

Parameters	None
Return Values	None
Description	Event to occur when "Open Video File" button is pressed in File Menu. It fetches path of file from file open dialog and calls fnLoadVideo to load file.
Functions called by this function	None

Event: OnFileCaptureFromCamera()

Parameters	None
Return Values	None
Description	Event to occur when "Capture from camera" button is pressed in File Menu. It sets path of file to camera stream and name as camera and calls fnLoadVideo to load file.
Functions called by this function	None

Function: UpdateData()

Parameters	None
Return Values	None
Description	Updates data when activity is reported
Functions called by this function	None

User Interface Module

Function: InitInstance()

Parameters	None
Return Values	True False to exit
Description	Initializes the instance of the application.
Functions called by this function	None.

Event: OnQueryDragIcon()

Parameters	None
Return Values	HCURSOR - type of cursor to display (default = NULL)
Description	The system calls this function to obtain the cursor to display while the user drags the minimized window.
Functions called by this function	None

Event: OnPaint()

Parameters	None
Return Values	None
Description	To add a minimize button to dialog, we require the code below to draw the icon. For MFC applications using the document/view model, this is automatically done by the framework.
Functions called by this function	None

Event: OnInitDialog()

Parameters	None
Return Values	True False to exit
Description	Message handlers for cFinalDlg.
Functions called by this function	None.

Function: Ipl2Bmp()

Parameters	IplImage * img - Pointer to an IplImage
Return Values	CBitmap * - Pointer to a CBitmap image
Description	converts IplImage to CBitmap for display in application
Functions called by this function	None.

Event: OnBnClickedStart()

Parameters	None
Return Values	None
Description	Event to occur when "Start Processing" button is pressed
Functions called by this function	None.

Event: OnBnClickedStop()

Parameters	None
Return Values	None
Description	Event to occur when "Stop Processing" button is pressed
Functions called by this function	None.

Event: OnBnClickedCancel()

Parameters	None
Return Values	None
Description	Event to occur when "Cancel" button is pressed
Functions called by this function	None.

Event: OnSysCommand()

Parameters	UINT nID, LPARAM lParam
Return Values	None
Description	To display the About box.
Functions called by this function	None

4.2.3. Database design

In database design, we have a login table AOD_LOGIN which contains the username and password. The username is unique and none of the fields can be null. Length of username as well as password is 15. Both the username as well as password can be either integer, alphanumeric or character. There is another table for recording activities AOD_ACTIVITY which contains an automatically generated activity id., employee name, action and time. Also the attribute action cannot be null while time contains the timestamp. The third table is a table to log the details of abandoned object AOD_LOG which keeps the log of all the activities. It contains the attributes like AOD Id., Time, X Coordinate, Y Coordinate, Filename and Username. The attribute AOD Id. contains the id. of that activity and time is the timestamp of the time when detection took place. X and Y coordinates of the object would help in locating the position of the object. The username can be 30 fields long and gives the info of the user who reported the activity. File name is the field that tells which file is being reported.

```
* Login Details Table:
create table AOD_Login
(
        UserName varchar2(15) not NULL primary key,
        Password varchar2(15) not NULL,
        Account  varchar2(5) constraint CHECK_AccType check(Account in
                ('Admin','User'))
);
```

* Activity Log Table:

```
create table AOD_Activity
(
 ActivityID number primary key,
 UserName varchar2(15) NOT NULL CONSTRAINT FK_UserName REFERENCES
AOD_Login(UserName),
 Action varchar2(70) constraint CHECK_ActionType check(Action in
  ('LogIn','LogOut','Recorded')),
 Time date
);
```

*Sequence to generate Activity Id.

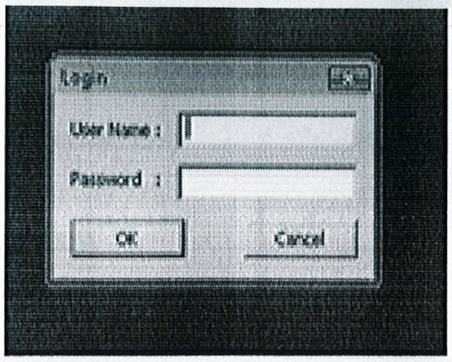
```
create sequence GetActivityId start with 1 increment by 1;
```

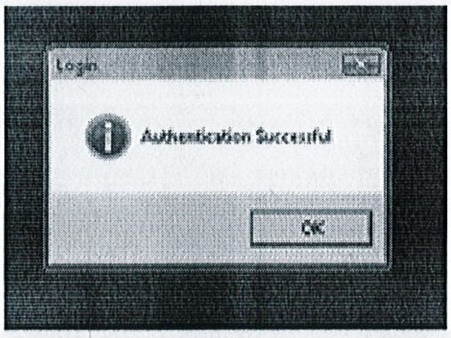
* Procedure to insert data in activity table.

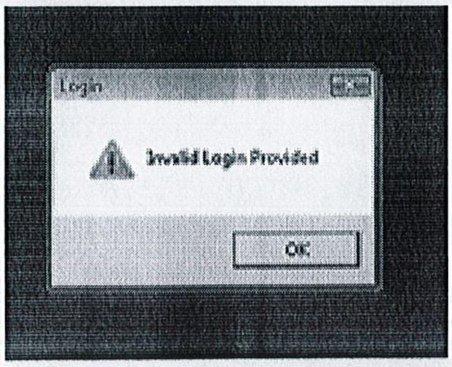
```
create or replace function fnInsertAction
(Action in AOD_Activity.Action%type) return Number
as
begin
        insert into AOD_Activity values( GetActivityId.nextval , Action , sysdate);
        commit;
        return 1;
exception
        when others then
                rollback;
                return 0;
end;
```

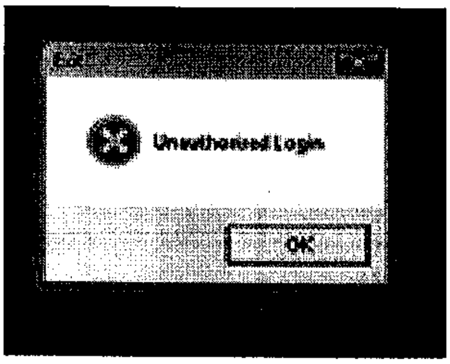
4.2.4. Interfaces design

Login Screens





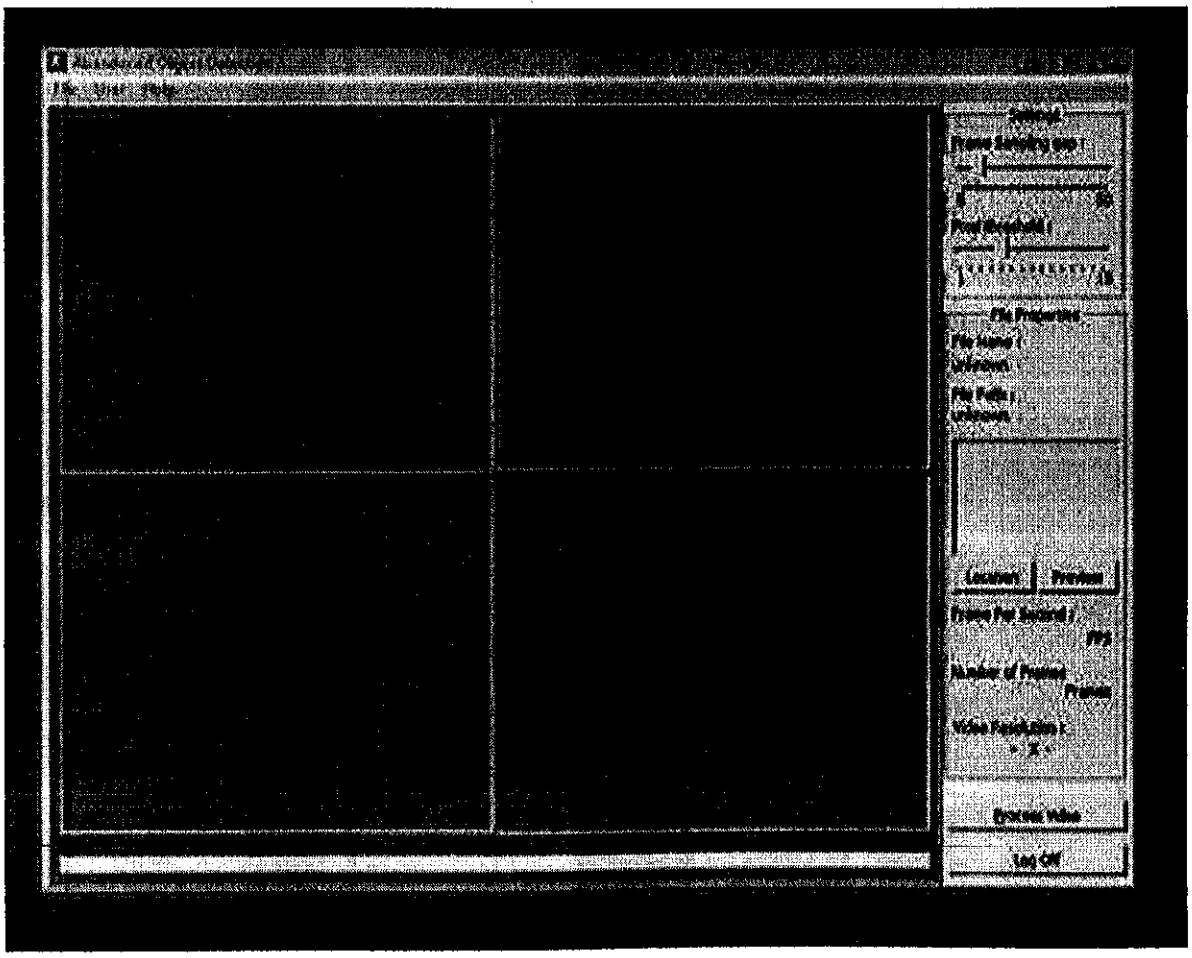




Unauthorised Login

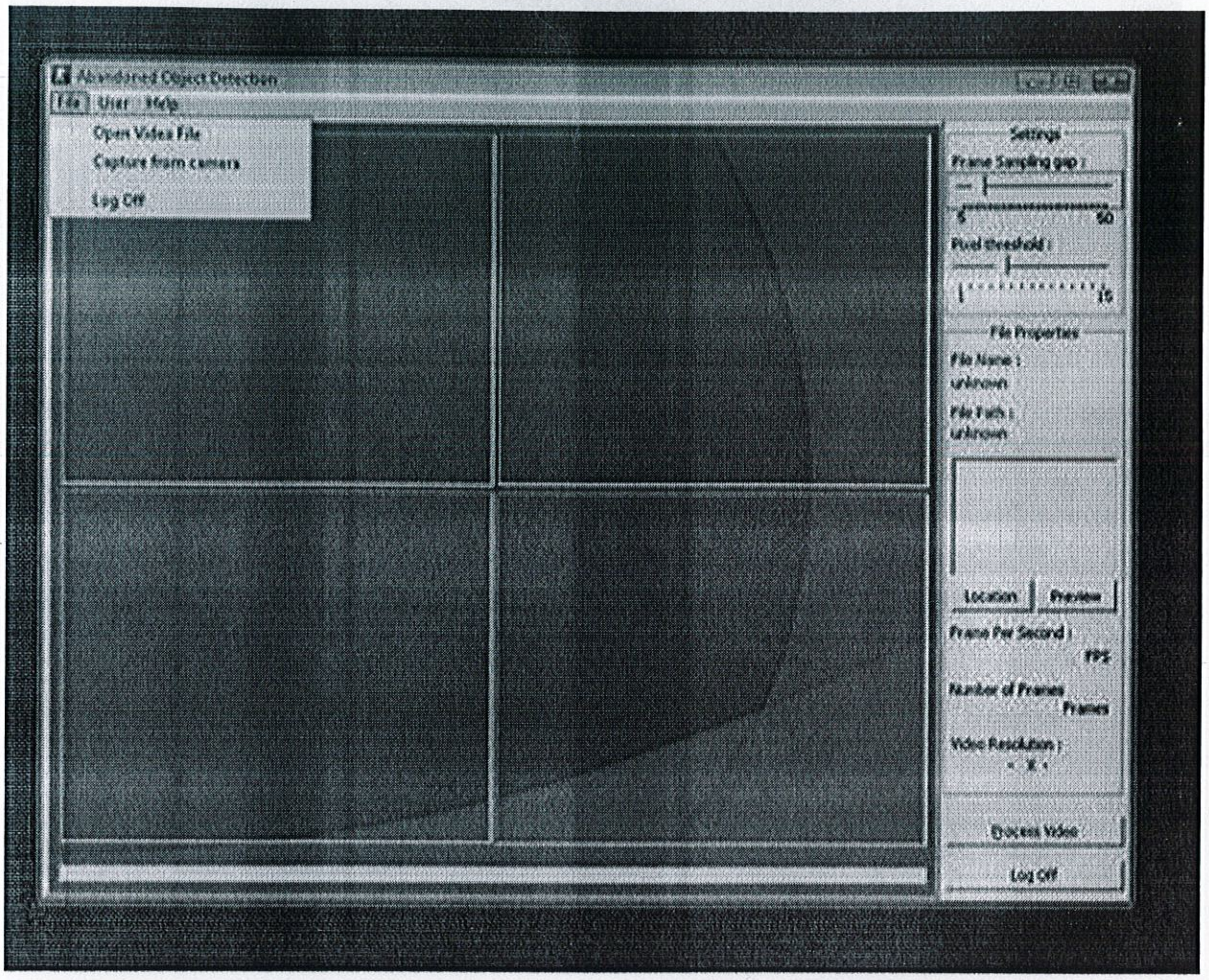
Menu Screens

Main Menu



The user interface of the application.

File



File Name : unknown

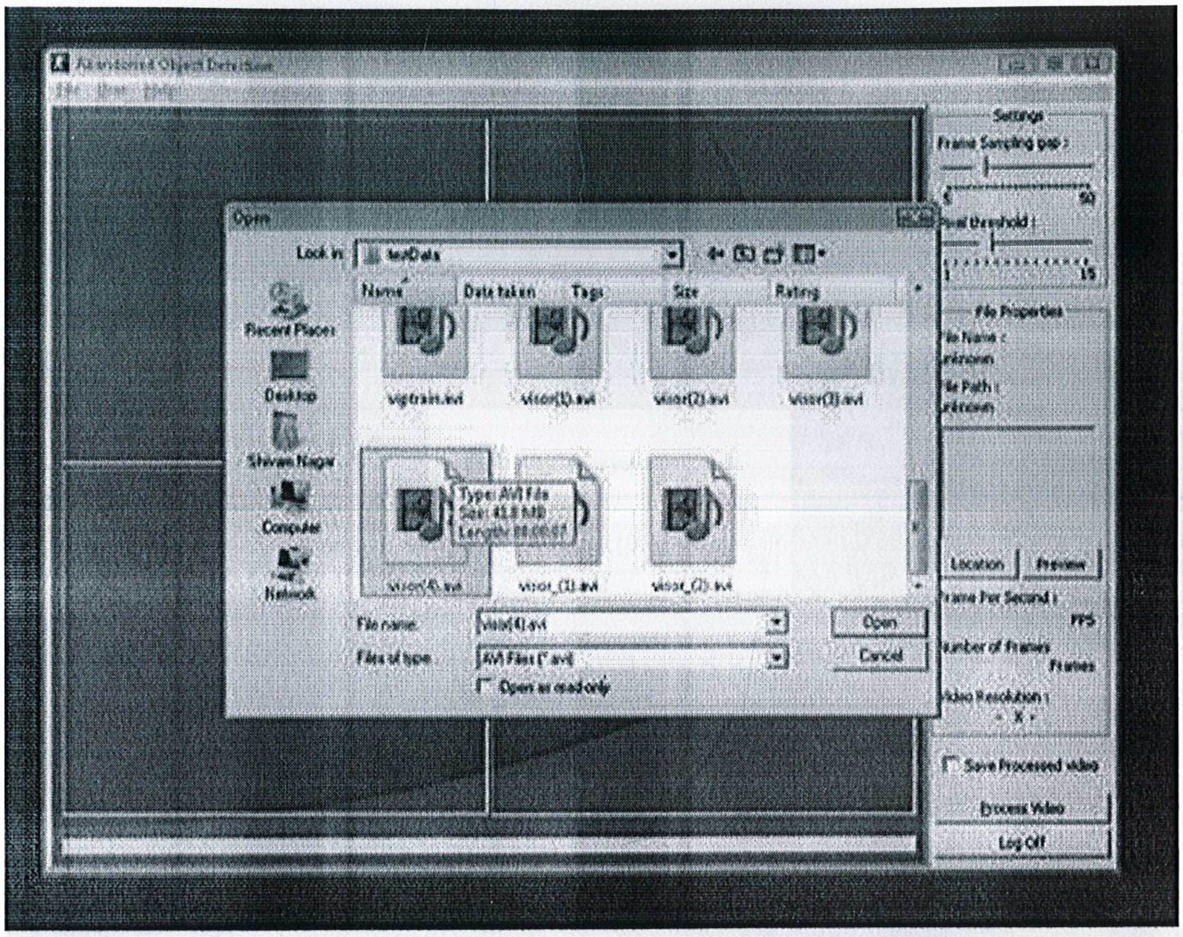
File Path : unknown

Frame Per Second : FPS

Number of Frames Frames

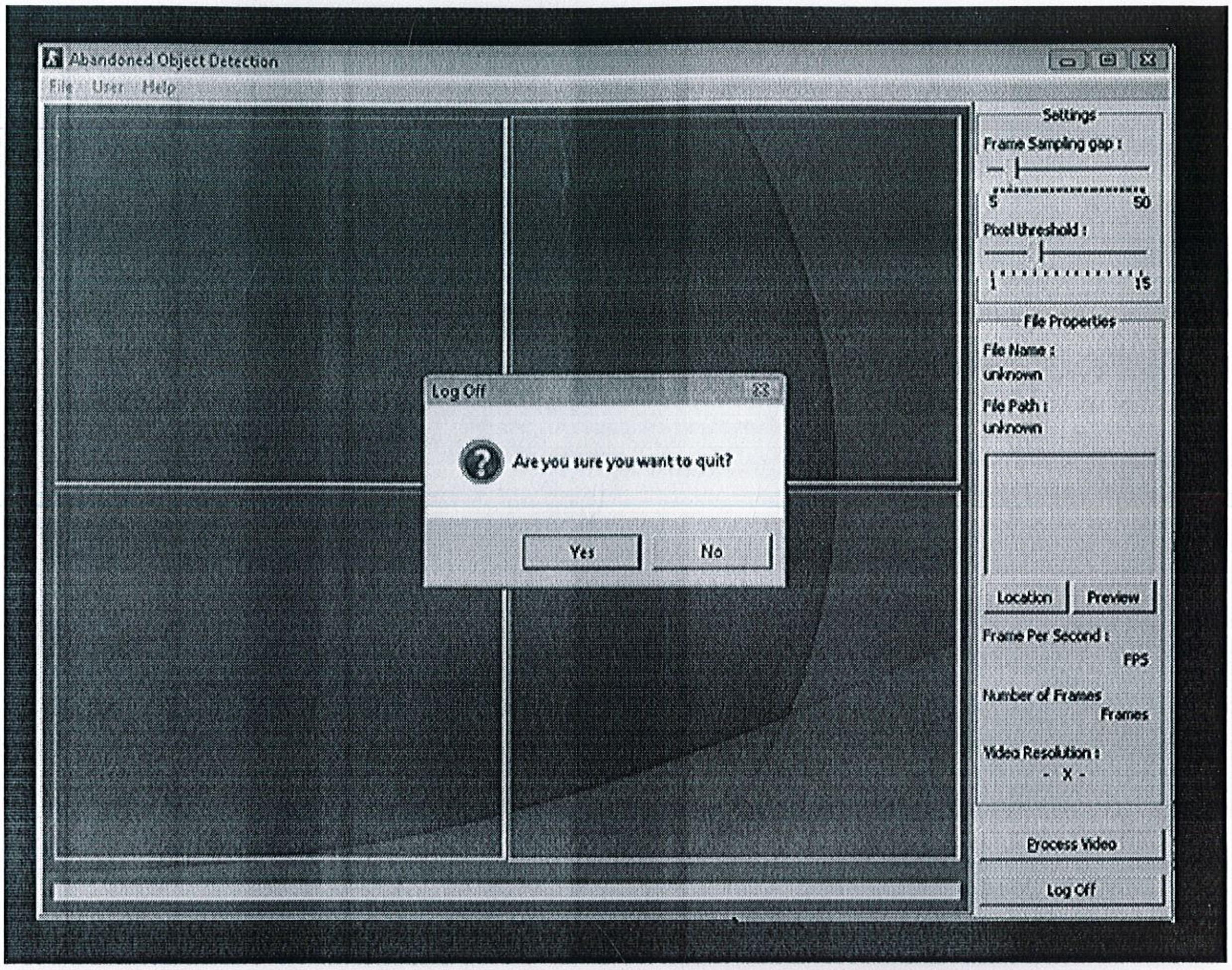
Video Resolution : - X -

Open Video File: Opens the video file and asks to browse the file



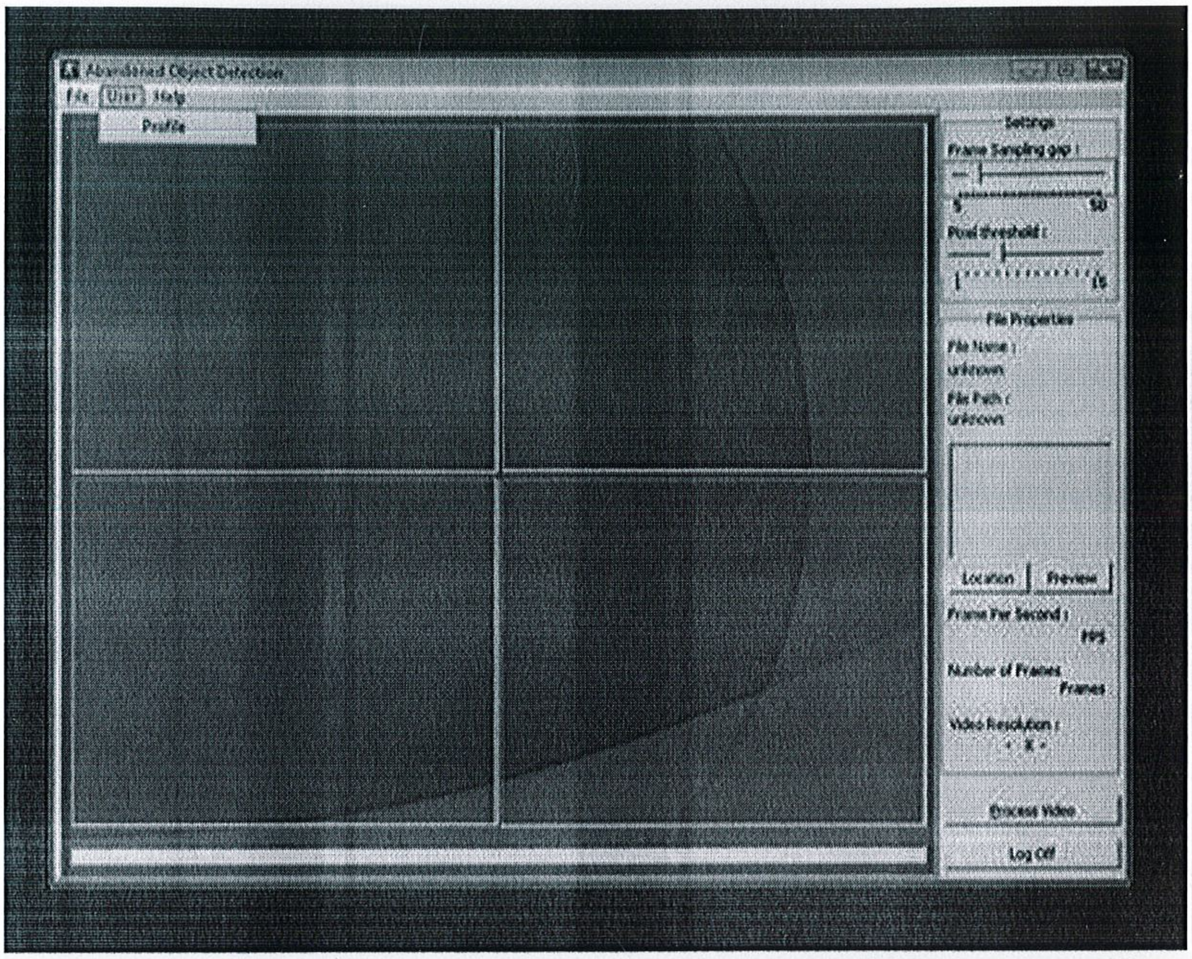
Capture from camera: looks for the camera and starts capturing the video.

Log Off: Logs off from the system after asking permission.



Are you sure you want to quit?

User Menu



Frame Sampling gap : 5 50

Pixel threshold : 1 15

File Properties

File Name :
unknown

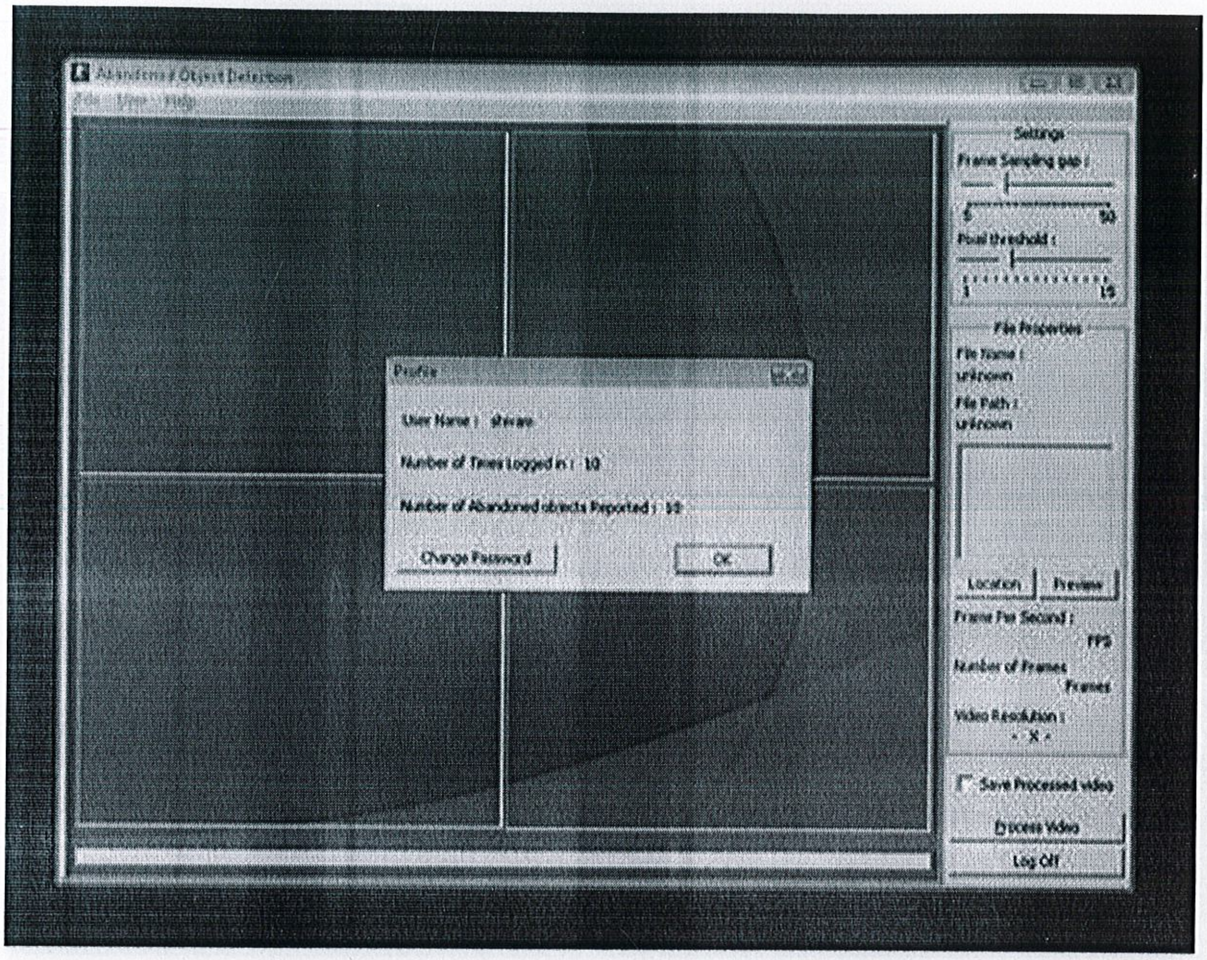
File Path :
unknown

Frame Per Second :
FPS

Number of Frames
Frames

Video Resolution :
- X -

Profile: Gives the details of the user.

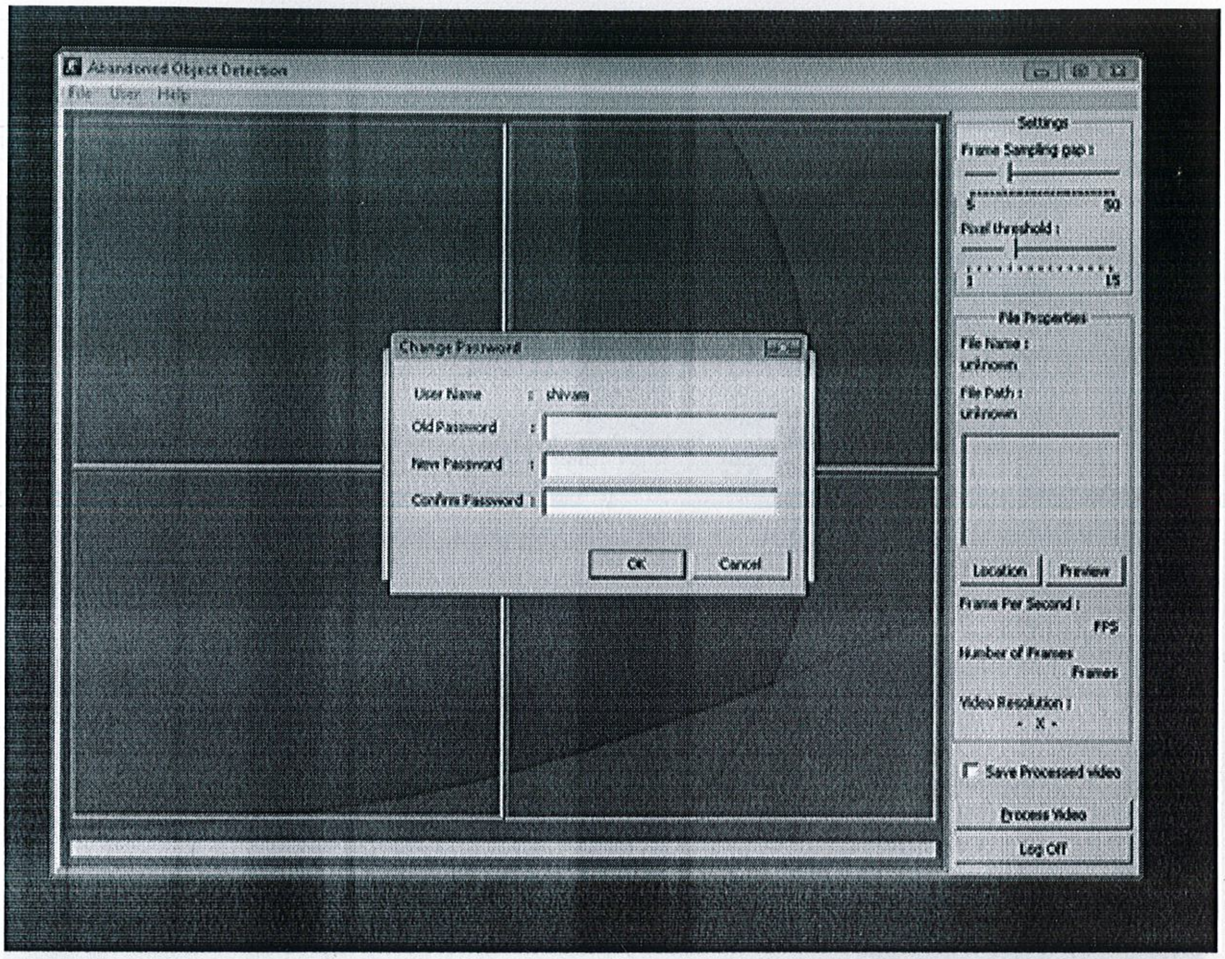

Profile

User Name : shivam

Number of Times Logged in : 10

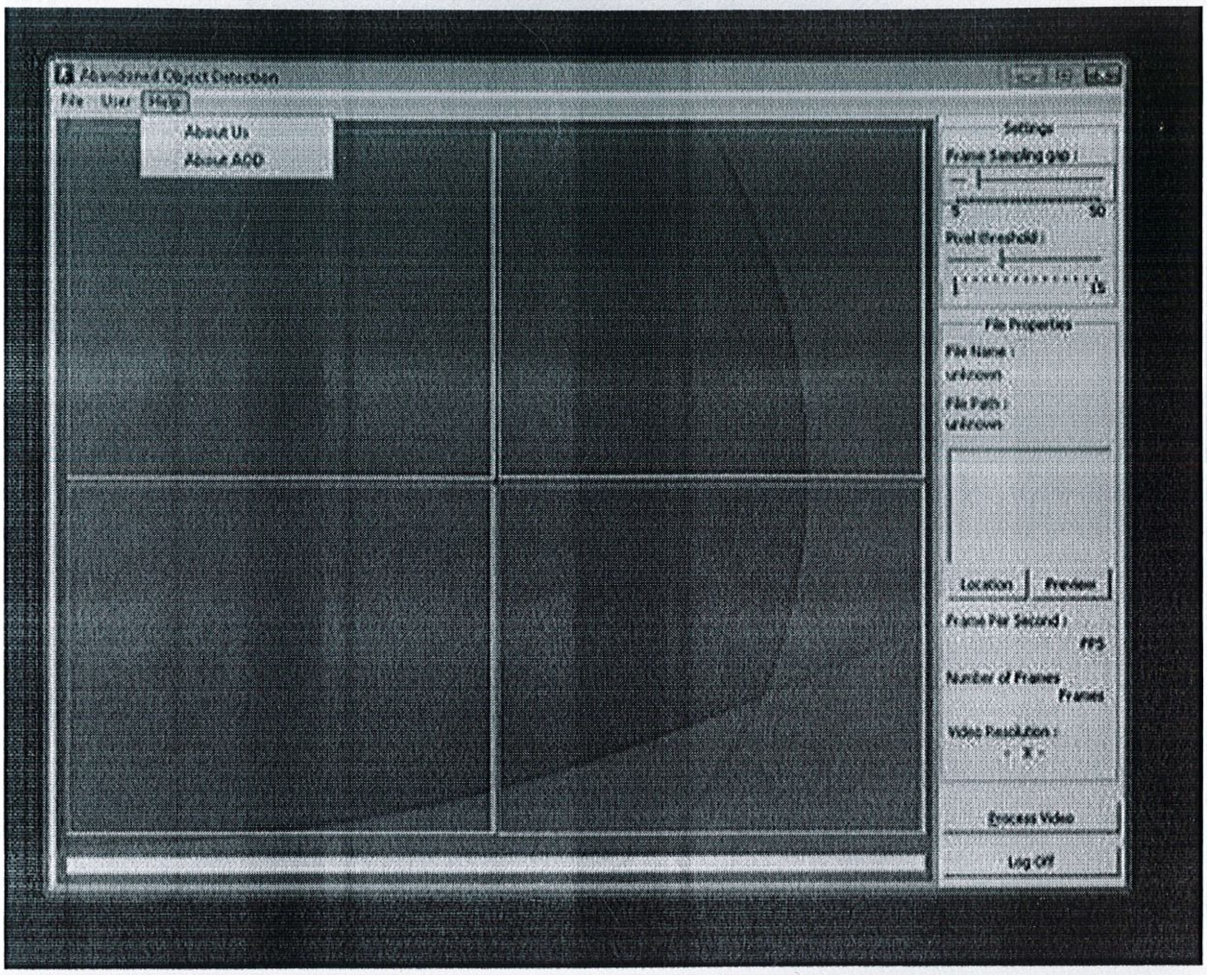
Number of Abandoned objects Reported : 10

Change Password: This option allows you to change your password.



User Name : shivam
Old Password :
New Password :
Confirm Password :

Help



File Name : unknown

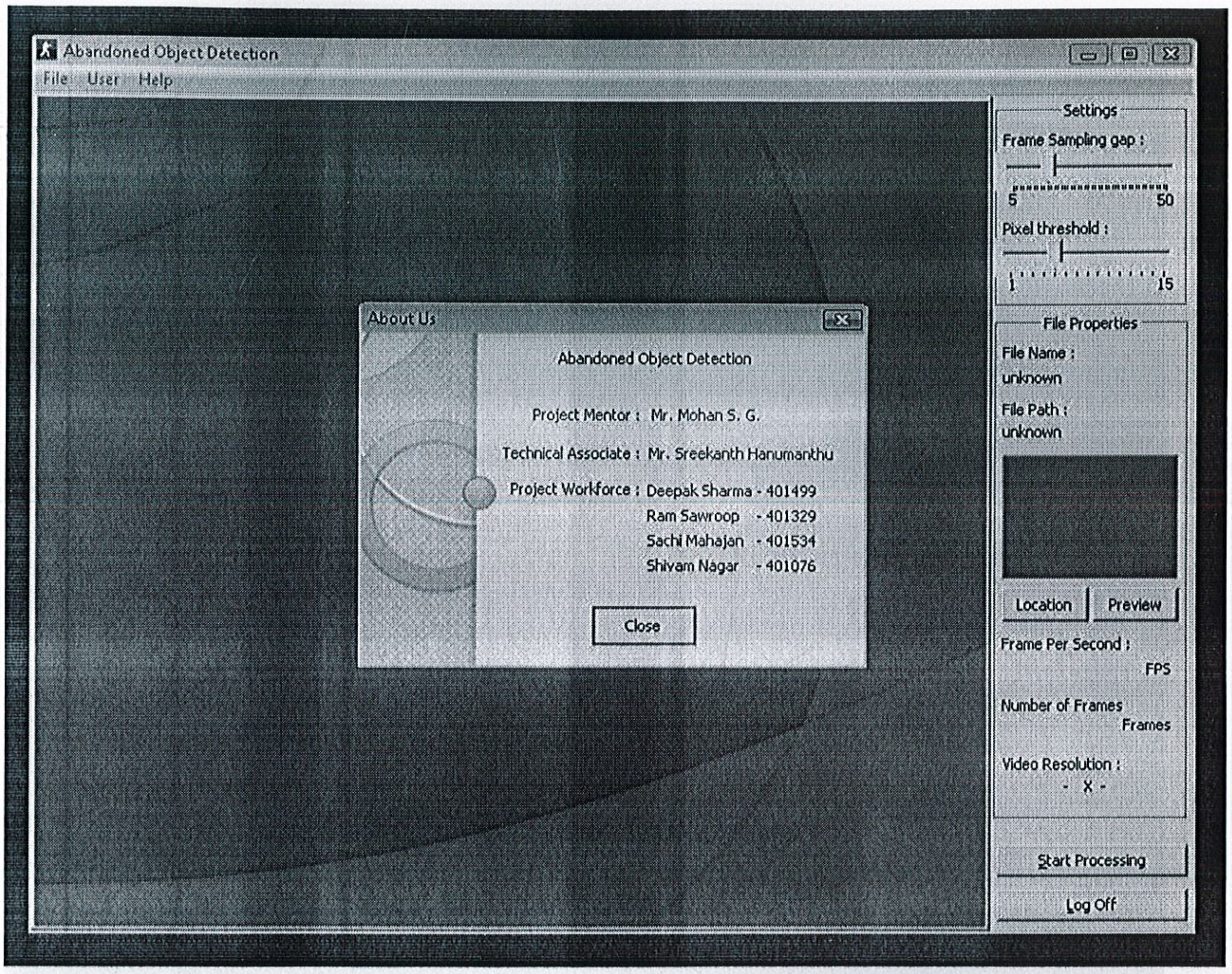
File Path : unknown

Frame Per Second : FPS

Number of Frames Frames

Video Resolution : X

About Us: About the people involved in the project



File Name : unknown

File Path : unknown

Frame Per Second : FPS

Number of Frames Frames

Video Resolution : - X -

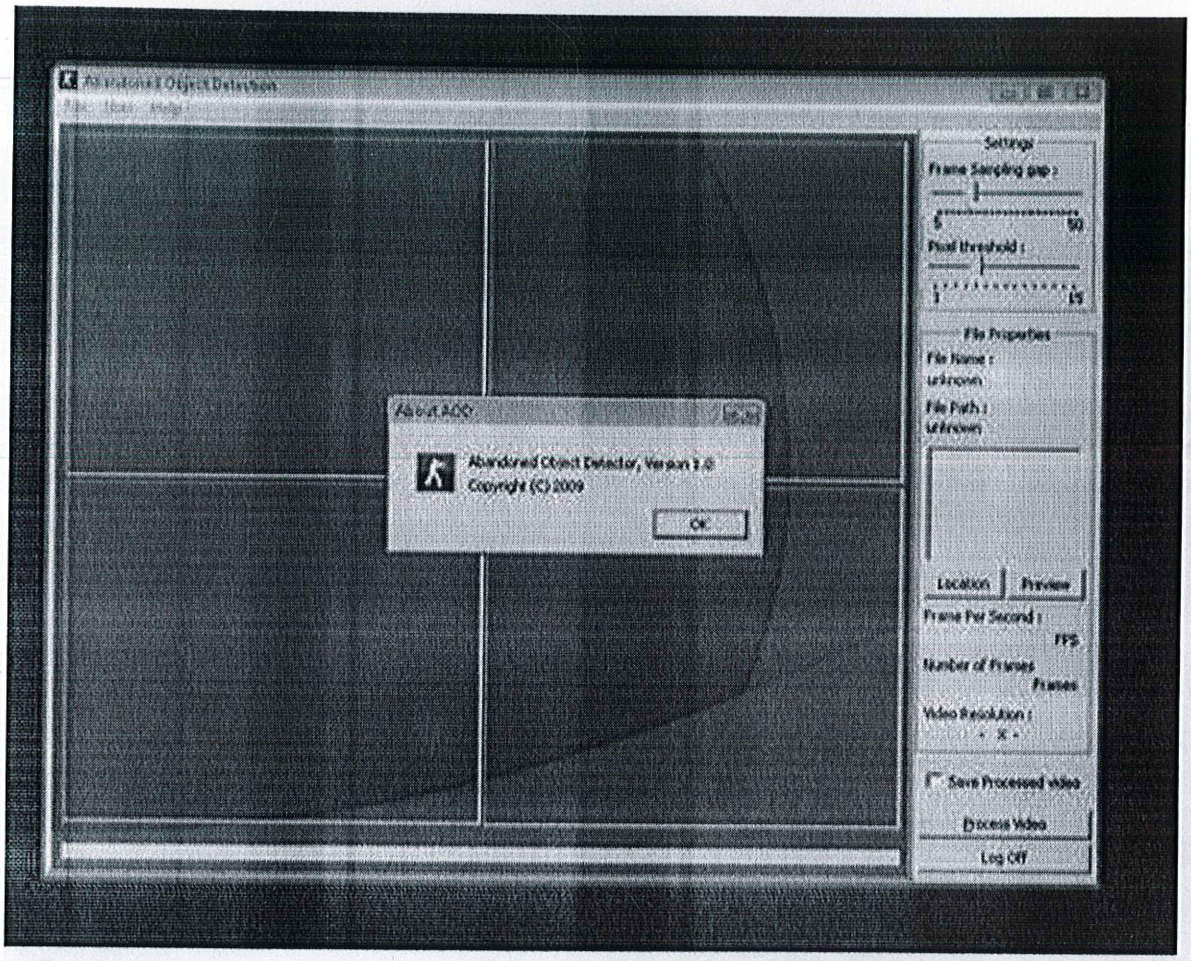
Abandoned Object Detection

Project Mentor : Mr. Mohan S. G.

Technical Associate : Mr. Sreekanth Hanumanthu

Project Workforce : Deepak Sharma - 401499
Ram Sawroop - 401329
Sachi Mahajan - 401534
Shivam Nagar - 401076

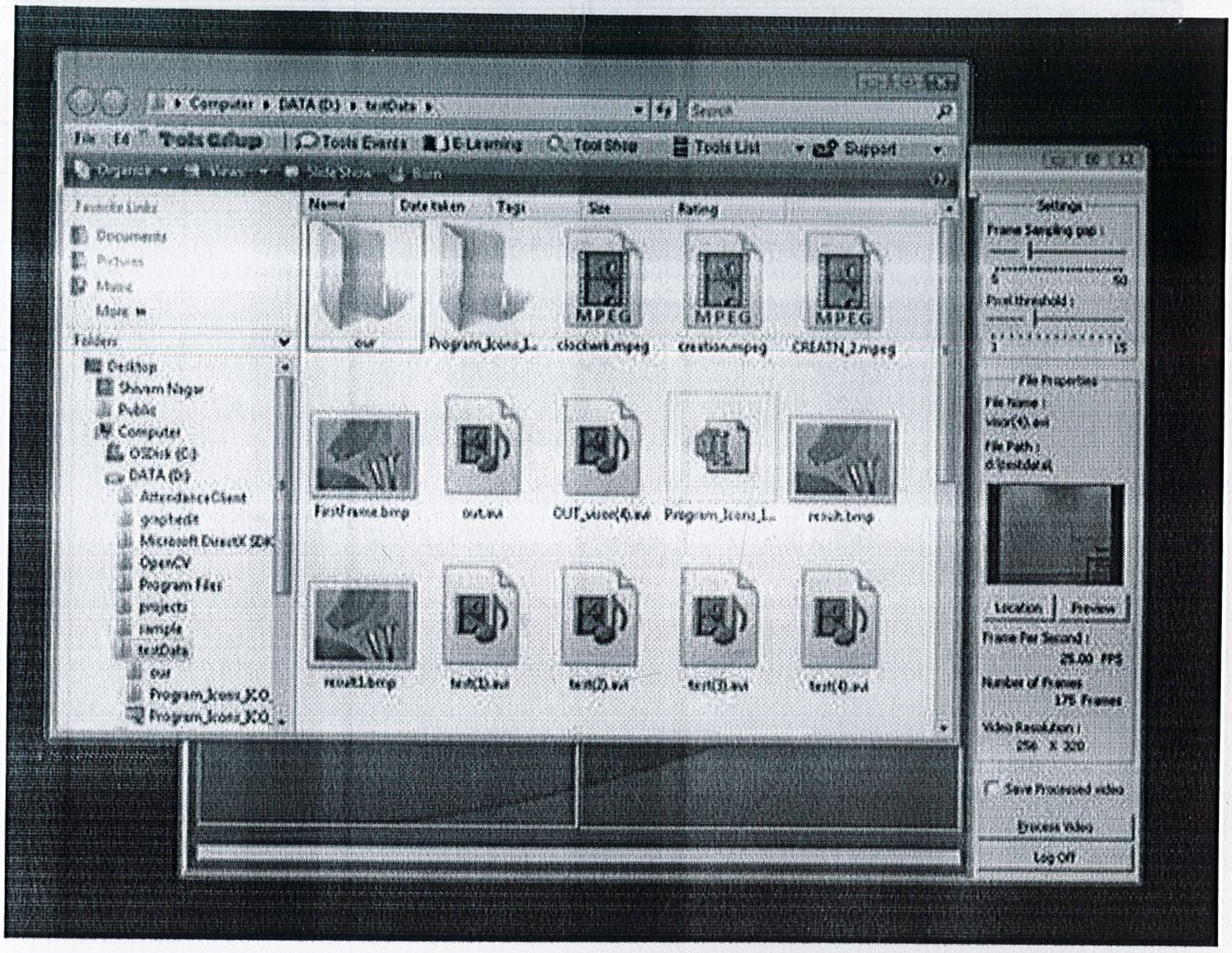
About AOD: About Abandoned Object Detection System


About AOD

Abandoned Object Detector, Version 1.0
Copyright (C) 2009

Interface Menu

Location: If the user needs to know the location of the file being used, the system shows the location.



Settings

Frame Sampling gap :

5 — 50

Pixel threshold :

3 — 15

File Properties

File Name :
visor(4).avi

File Path :
d:\testdata\

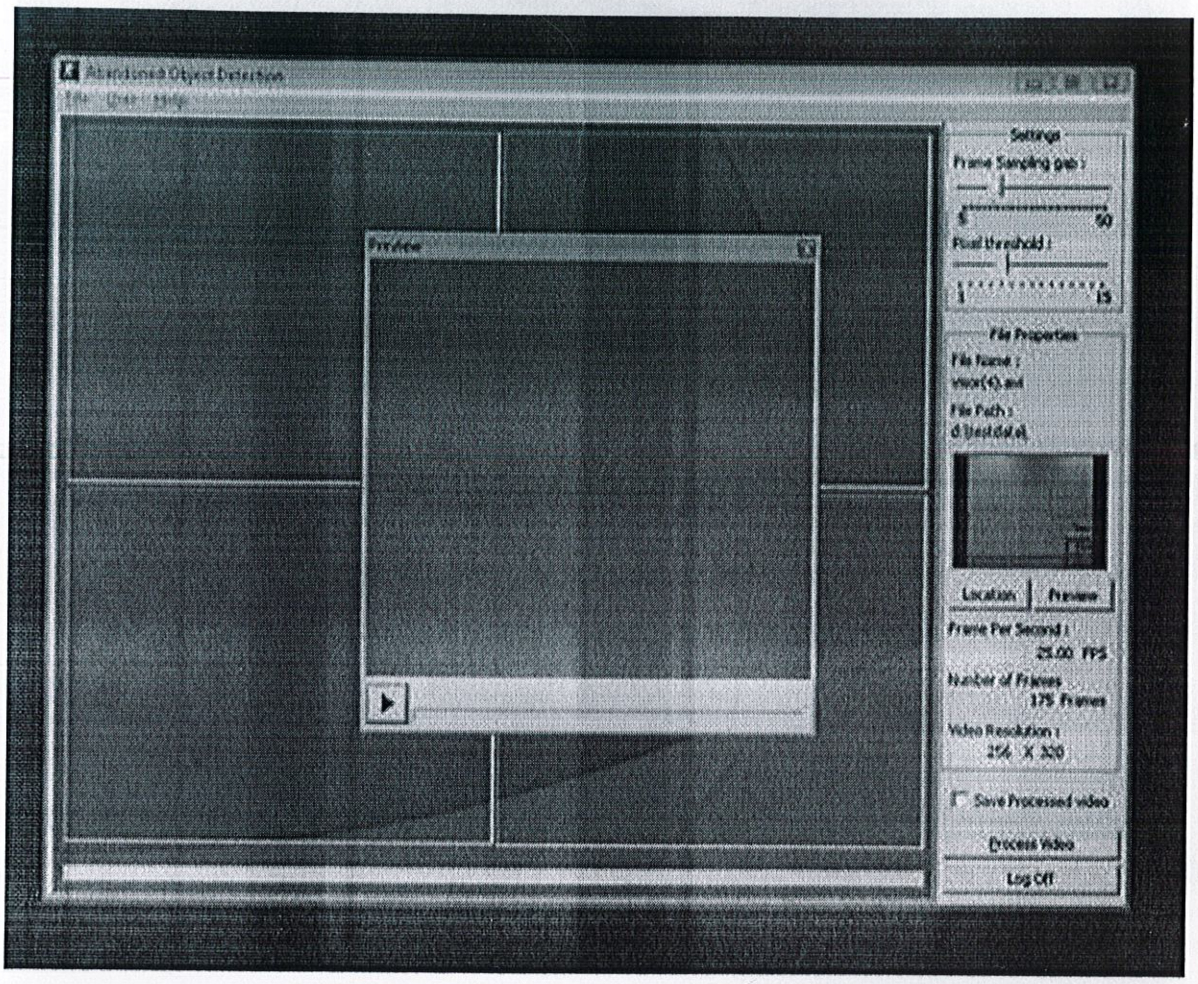
Frame Per Second :
25.00 FPS

Number of Frames
175 Frames

Video Resolution :
256 X 320

Save Processed video

Preview: Previews the loaded video.



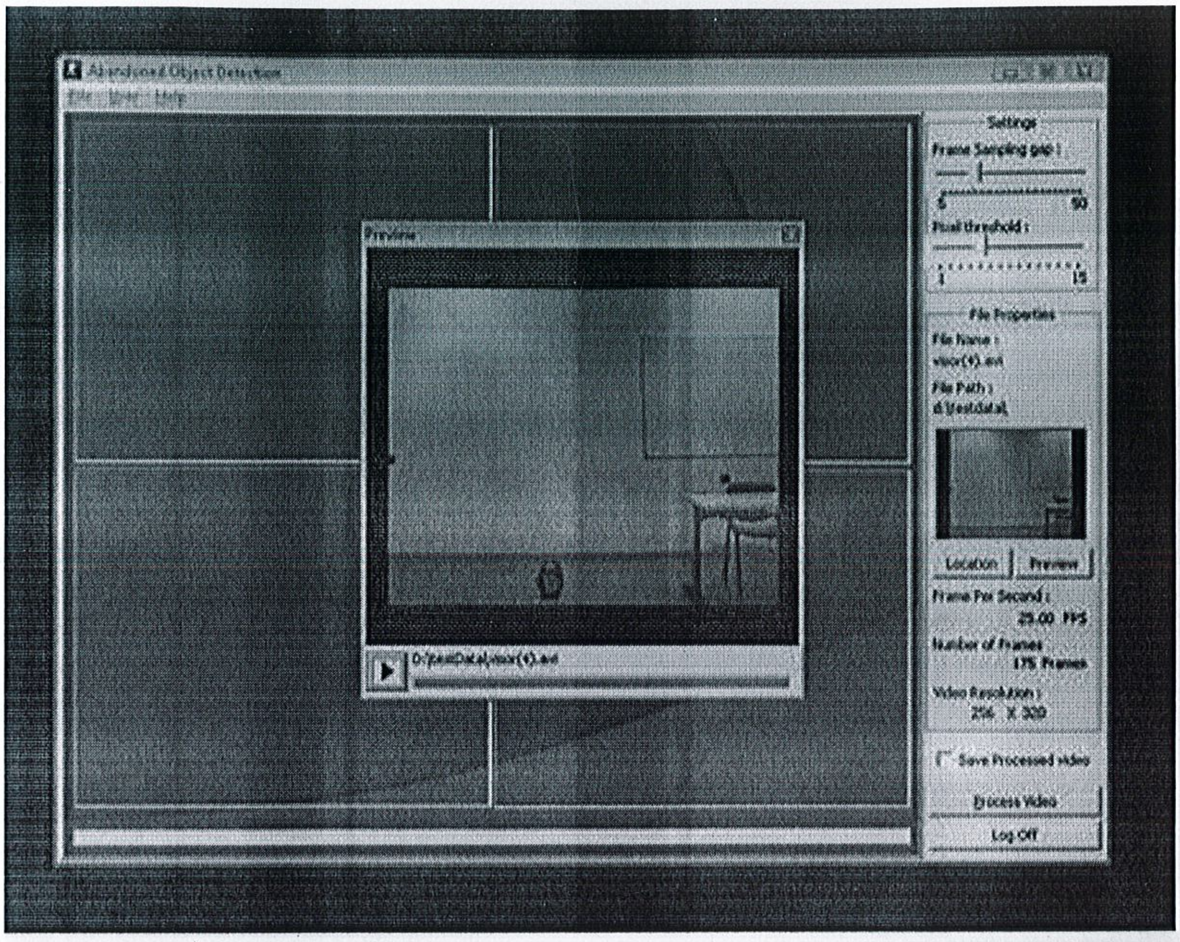
Settings
Frame Sampling gap :
5 50
Pixel threshold :
1 15

File Properties
File Name :
video(4).avi
File Path :
d:\test data\

Frame Per Second :
25.00 FPS
Number of Frames
175 Frames
Video Resolution :
256 X 320

Save Processed video

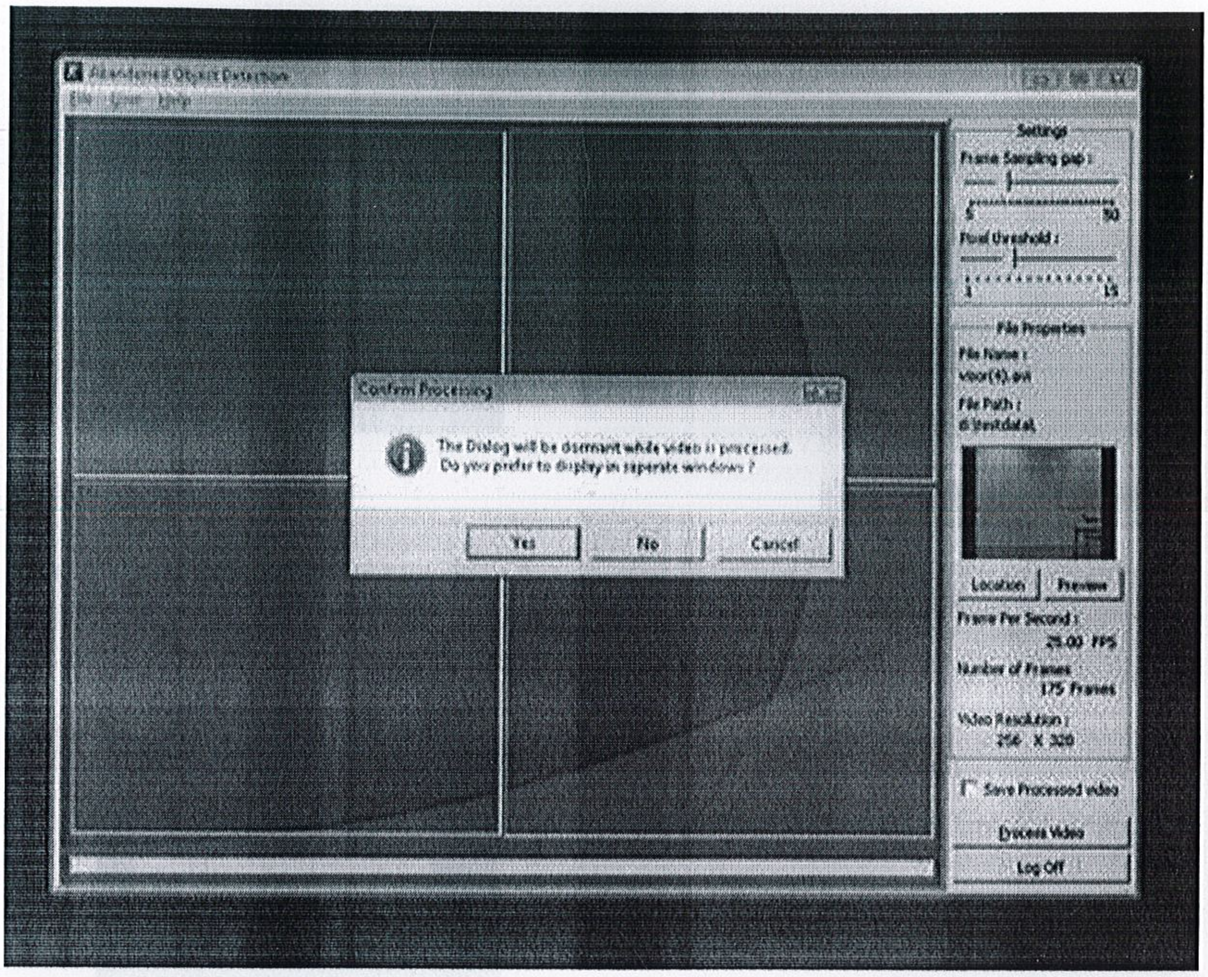
The system provides an option to display the processing in separate window .



Settings
Frame Sampling gap :
5 50
Pixel threshold :
1 15
File Properties
File Name :
visor(4).avi
File Path :
d:\testdata\
Frame Per Second :
25.00 FPS
Number of Frames
175 Frames
Video Resolution :
256 X 320
Save Processed video

D:\testData\visor(4).avi

The system provides an option to display the processing in separate window.


Abandoned Object Detection

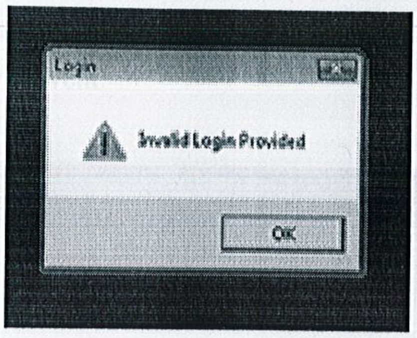
Confirm Processing

The Dialog will be dormant while video is processed.
Do you prefer to display in separate windows ?

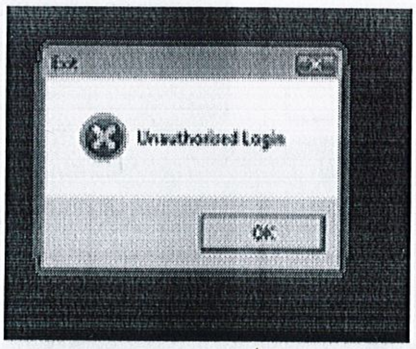
File Name :
vsor(4).avi
File Path :
d:\testdata\
Frame Per Second :
25.00 FPS
Number of Frames
175 Frames
Video Resolution :
256 X 320
Save Processed video

4.2.5. Error messages

In case of an invalid login is provided:

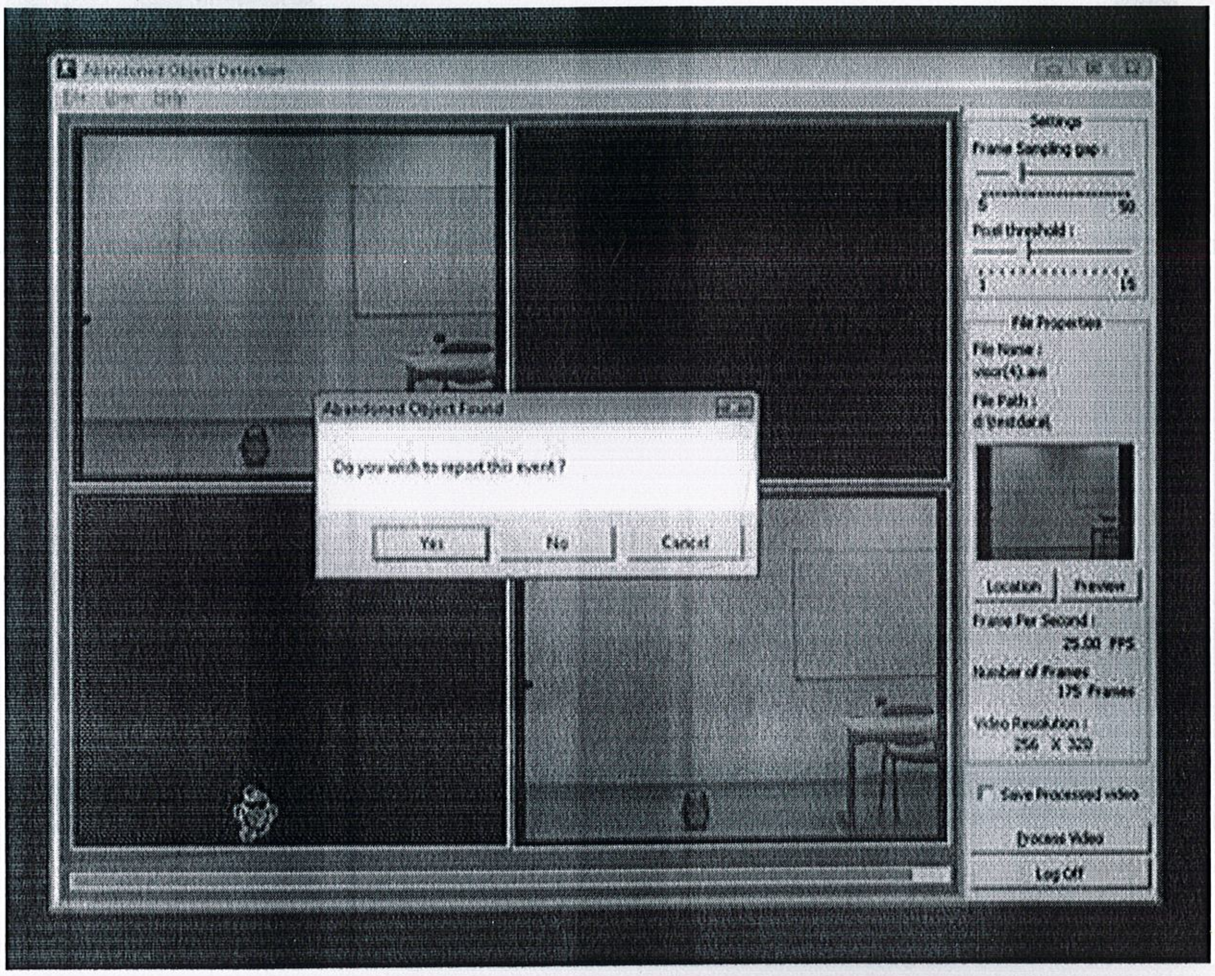

Login
Invalid Login Provided
OK

In case three invalid logins are provided:


Exit
Unauthorised Login
OK

4.2.6. <u>Report layout/ Screen layout</u>

As soon as an abandoned object is detected, the system asks if the user wants to report the detection event:

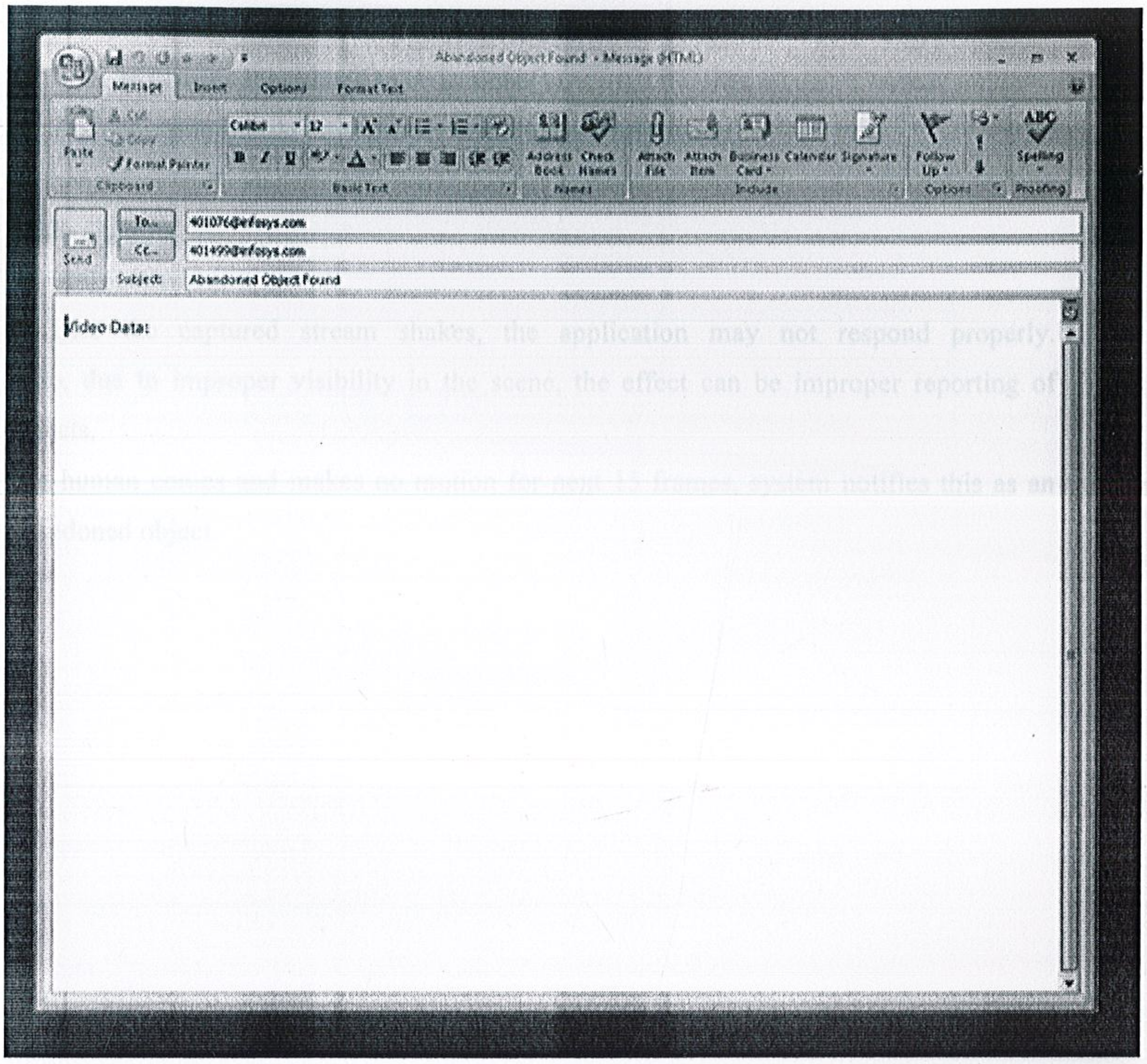


Do you wish to report this event ?

Settings
Frame Sampling gap : 5 – 50
Pixel threshold : 1 – 15

File Properties
File Name :
visor(4).avi
File Path :
d:\testdata\
Frame Per Second :
25.00 FPS
Number of Frames
175 Frames
Video Resolution :
256 X 320
Save Processed video

If the user selects to report the action, then an e-mail is sent.



To... 401076@infosys.com

Cc... 401499@infosys.com

Subject: Abandoned Object Found

Video Data:

4.2.7. Assumptions

- The camera is assumed to be stationery.
- The scene should have proper visibility and brightness.

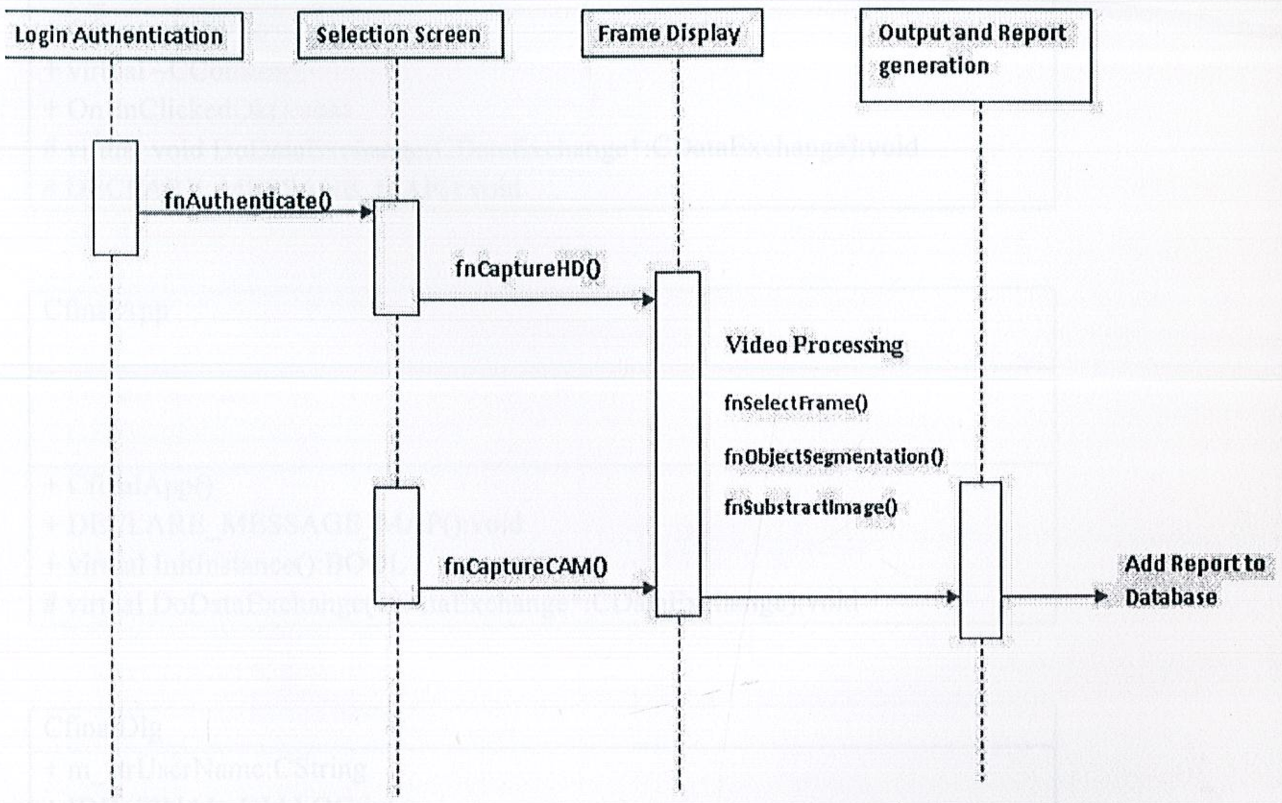
4.2.8. Limits

In case the captured stream shakes, the application may not respond properly. Also, due to improper visibility in the scene, the effect can be improper reporting of objects.

If a human comes and makes no motion for next 15 frames, system notifies this as an abandoned object.

4.3. Class Design Specification

4.3.1. Sequence diagram



4.3.2. Class Diagram

CAboutUs
+ IDD: int
+ CAboutUs(CWnd* pParent = NULL) + virtual ~CAboutUs() # virtual DoDataExchange(CDataExchange* :CDataExchange):Void

CControl

+ IDD_Login: int
+ m_strUser:Cstring
+ m_strPass:Cstring
+ m_editUserName:CEdit

+ CControl(CWnd*:CWnd)
+ virtual ~CControl()
+ OnBnClickedOk():void
# virtual void DoDataExchange(CDataExchange*:CDataExchange):void
# DECLARE_MESSAGE_MAP():void

CfinalApp

+ CfinalApp()
+ DECLARE_MESSAGE_MAP():void
+ virtual InitInstance():BOOL
# virtual DoDataExchange(CDataExchange*:CDataExchange):void

CfinalDlg

+ m_strUserName:CString
+ IDD_FINAL_DIALOG:int
# m_hIcon:HICON
+ m_stImage:CStatic
+ m_stPicture:CStatic
+ m_btButton:CButton
+ m_stText CStatic
+ m_stFrame:CStatic
+ m_stSub:CStatic
+ m_stObject:CStatic
+ m_stFinal:CStatic
+ m_prog:*CProgressCtrl*
+ m_btFile CButton
+ m_stButton:CStatic
+ m_strFPS:CString
+ m_strFrameX:CString

```
+ m_strFrameY:CString
+ m_iPixelThreshold:int
+ m_strFileName:CString
+ m_pixel: CSliderCtrl
+ m_Window:BOOL
+ m_NumberOfFrames:CString
+ m_slider:CSliderCtrl
+ m_FileName:CEdit
+ m_strPath:CString
+ m_btStop:CButton
+ m_btPrev:CButton
+ m_btLogoff:CButton
+ m_btLocation:CButton
--------------------------------------------------
# DECLARE_MESSAGE_MAP():void
# virtual InitDialog():BOOL
# OnSysCommand(UINT ,LPARAM ):void
# OnPaint():void
# OnQueryDragIcon():HCURSOR
# friend ThreadFunction(LPVOID):UINT
+ CfinalDlg(CWnd*:CWnd)
+ virtual  DoDataExchange(CDataExchange*:CDataExchange):void
+ OnBnClickedStart():void
+ OnStnClickedPicture():void
+ OnBnClickedFile():void
+ OnBnClickedCancel():void
+ LoadVideo()():void
+ OnBnClickedButton6():void
+ OnFileOpenvideofile():void
+ OnFileCapturefromcamera():void
+ OnFileLogoff():void
+ OnHelpDemo():void
+ OnHelpAboutAod():void
+ OnUserProfile():void
+ OnHelpAboutus():void
+ OnStnClickedBackground():void
+ OnBnClickedLocation():void
+ OnBnClickedPreview():void
+ OnBnClickedWindow():void
+ OnBnClickedStop():void
```

CLoader
+ IDD_Loader:int + m_progress :CProgressCtrl + m_strText:CString
virtual DoDataExchange(CDataExchange*:CDataExchange # DECLARE_MESSAGE_MAP():void + ~CLoader() + Cloader(CWnd*:CWnd) + Load(int ,char*):void + OnEnChangeEdit1():void

CAboutDlg
IDD_AboutDlg:int
virtual DoDataExchange(CDataExchange*:CDataExchange + CAboutDlg() # DECLARE_MESSAGE_MAP():void + OnFileLogoff():void

CPassword
+ IDD_ChangePassword:int + m_strUserName :CString + m_strOldPassword: CString + m_strNewPassword: CString + m_strConfirmPassword: CString
virtual DoDataExchange(CDataExchange*:CDataExchange # DECLARE_MESSAGE_MAP():void + virtual ~CPassword() + CPassword(CWnd*:CWnd) + OnBnClickedOk():void

CPreview
+ IDD_Preview:int
+ m_pcPriviewFile :Char*
+ m_stPreview: CStatic
+ m_strFilename: CString
+ m_btPause: CButton
+ m_Progress :CProgressCtrl
+ m_btPlay:CButton
virtual DoDataExchange(CDataExchange*:CDataExchange
DECLARE_MESSAGE_MAP():void
+ CPreview(CWnd*:CWnd)
+ OnBnClickPause():void
+ friend showVideo(void *:void):void
+ virtual onInitDialog():BOOL
+ virtual onCancel():void
+ virtual ~Cpreview()
+ OnBnClickPlay():void

CUser
+ IDD_Profile:int
+ m_iFound :int
+ m_iTimes:int
+ m_strUserName: CString
virtual DoDataExchange(CDataExchange*:CDataExchange
DECLARE_MESSAGE_MAP():void
+ virtual ~CUser()
+ CUser(CWnd*:CWnd)
+ OnBnClickedOk():void
+ OnBnClickedChange():void

4.3.3. Class Identification

Class Name	cAboutUs
Class Description	Contains the description of the people involved in the project.
Class Inheritance	cDialog
Classes Referenced	finalDlg
Sub System	None
Class Type	Complete
Change History	None

Class Name	cControl
Class Description	Contains the login information and controls the login of users.
Class Inheritance	cDialog
Classes Referenced	None
Sub System	None
Class Type	Complete
Change History	None

Class Name	cFinalApp
Class Description	Loads the final application by inheriting cWinApp and referencing the class cdialog
Class Inheritance	cWinApp
Classes Referenced	cDialog
Sub System	None
Class Type	Complete
Change History	None

Class Name	cfinalDlg
Class Description	Draws dialogs
Class Inheritance	cDialog
Classes Referenced	None
Sub System	None
Class Type	Complete
Change History	None

Class Name	cPassword
Class Description	To handle passwords
Class Inheritance	cDialog
Classes Referenced	None
Sub System	None
Class Type	Complete
Change History	None

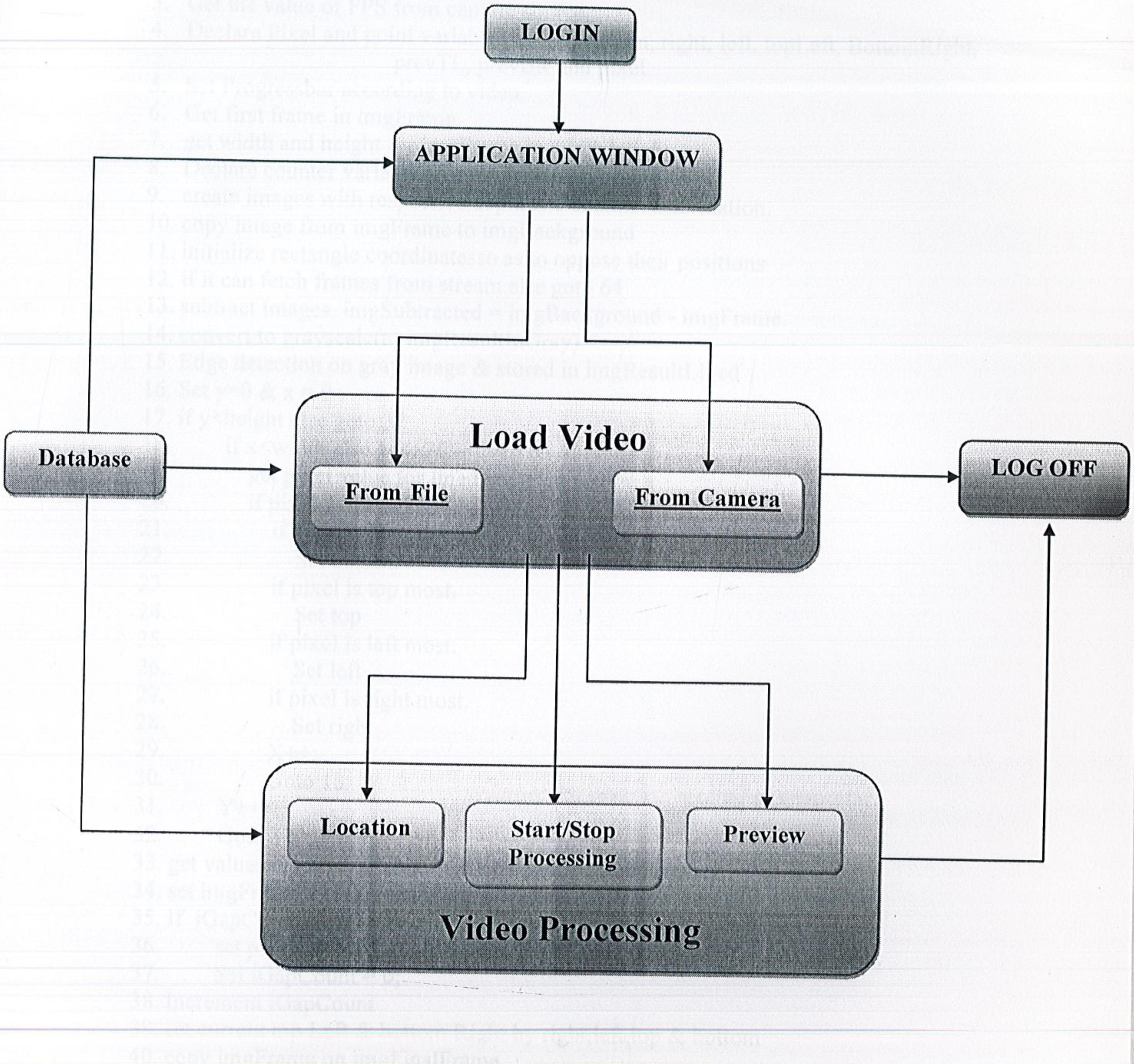
Class Name	cPreview
Class Description	Loads all the other classes indirectly by inheriting cDialog class
Class Inheritance	cDialog
Classes Referenced	None
Sub System	None
Class Type	Complete
Change History	None

Class Name	cUser
Class Description	Authenticates the user name and password
Class Inheritance	cDialog
Classes Referenced	None
Sub System	None
Class Type	Complete
Change History	None

Class Name	cDialog
Class Description	Loads dialog in all the other classes
Class Inheritance	None
Classes Referenced	cAboutUs, cControl, cfinalDlg, cPassword, cPreview, cUser
Sub System	None
Class Type	Abstract/Complete
Change History	None

5 Code Walkthrough

Pictorial representation of the Procedure



Algorithm:

```
1. Get the value of pixel approximation threshold entered by user.
2. Get the value of Frame Spacing entered by user.
3. Get the value of FPS from capture source.
4. Declare Pixel and point variables as top, bottom, right, left, topLeft, BottomRight,
                 prevTL, prevBR and pixel.
5. Set Progressbar according to video.
6. Get first frame in imgFrame.
7. get width and height for imgFrame in width,height.
8. Declare counter variables for movement in pixels
9. create images with respective depth and channel information.
10. copy image from imgFrame to imgBackground
11. initialize rectangle coordinatesso as to oppose their positions
12. if it can fetch frames from stream else goto 64
13. subtract images. imgSubtracted = imgBackground - imgFrame.
14. convert to grayscale(to imgResultInGray)
15. Edge detection on gray image & stored in imgResultLined
16. Set y=0 & x = 0
17. if y<height else goto 32
18.     if x<width else goto 30
19.         get pixel value for lined image at y,x
20.         if pixel value is 255 i.e. white
21.             if pixel is bottom most.
22.                 Set bottom
23.             if pixel is top most.
24.                 Set top
25.             if pixel is left most.
26.                 Set left
27.             if pixel is right most.
28.                 Set right
29.             X++;
30.             Goto 18
31.     Y++
32.     Goto 17
33. get values in prev variables if iGapCount between current
34. set imgFrame = iFramePadding.
35. If  iGapCount MOD iFramePadding is 0 else goto 38
36.     set previous top Left & bottom Right to current topleft & bottomright
37.     Set iGapCount = 0;
38. Increment iGapCount
39. set current top Left & bottom Right by right,left,top & bottom
40. copy imgFrame on imgFinalFrame
41. draw rectangle on imgFinalFrame in red color
42. if previous and current coordinates are within threshold else goto 54
43.     if SamplingType is CONTINUOUS and repete > iFramespace else goto 54
```

```
44.     if rectangle in not covering whole scene else goto 54
45.         draw rectangle on bounding box in blue color.
46.         reinitialize repete flag to zero
47.         reset iFramePadding = iFramespace
48.         Ask user to report
49.         If User Selects Yes then report else
50.         If User Selects no then continue else
51.         If User Selects cancel then exit
52.         object is found in descreet sampling set flags to perform continuous
                sampling.
53.     Goto 58
54.     Set iGapCount = 0;
55.     Set SamplingType = CONTINUOUS;
56.     Increment repete
57.     Set iFramePadding = 1;
58.     draw bounding edge point on lined image
59.     reset rectangle positions.
60.     display frames in GUI
61.     wait for next imgFrame
62.     move progress bar
63. Goto 12
64. release resourses
65. swap buttons.
66. restore state
67. Terminate thread
68. return 0 on success
```

Description of Algorithm:

The Algorithm basically is intended to detect objects which do not move in scene and are not there in base image. Algorithm sets 0th frame as base frame and subtracts it from frame obtained after frame space count of frames. Edge detection is performed on this image and the bounding points are obtained which cover the changed object. The coordinates of top, left, right, left are obtained and the coordinates of top left and bottom right points are calculated. Thereby providing bounded rectangular coordinates on object. As shown in figure (assuming frame spacing set to 25):

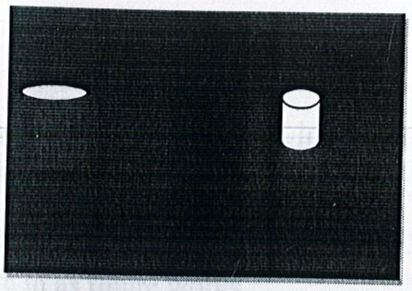


Figure1. First frame, taken as reference.

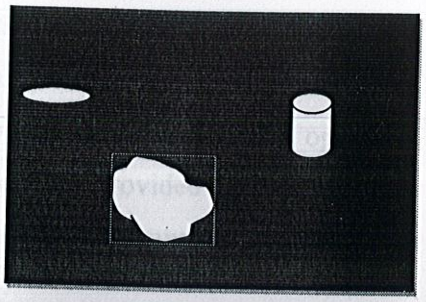
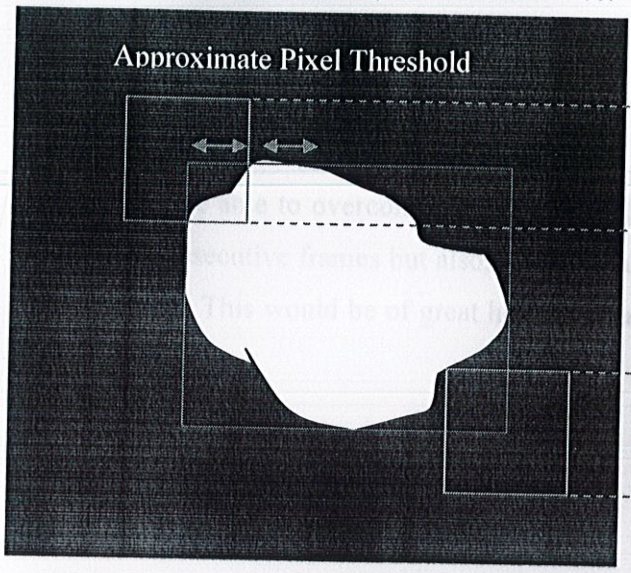


Figure2. 25th frame, Object found



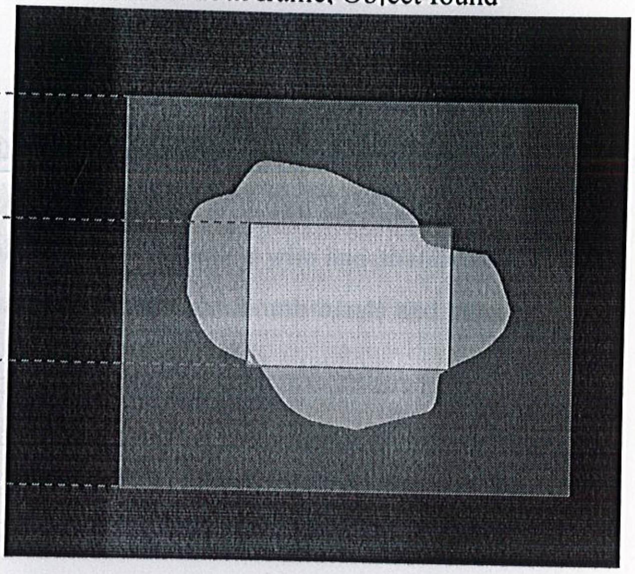


Figure3. Approximate pixel threshold boundaries around edge pixels

Figure4. Region defined by approximate pixel threshold where object should lie to be declared as Abandoned.

The next frame that will be compared will be defined by frame sampling gap. Same procedure will be applied on it to obtain bounding rectangle. If the rectangle lies within the region depicted in figure 4 above the sampling gap will be reduced to 1(continuous sampling) and same procedure is applied on each frame. No of time object found in same region is > sampling gap then it will be declared abandoned.

5. CONCLUSION

The goal of the project was to develop an application that could notify a human operator monitoring the video surveillance system about potentially critical events such as abandoned objects placed in public areas covered by security cameras. The operator will then decide how to proceed based on the information provided by the system. As of now the goal has been successfully achieved but a lot can be improved in it and this we take in as future works.

6. Future Works

In future, the application can be advanced by adding facial recognition which would not only be able to overcome the limitation of reporting a man who has been stationary for 15 consecutive frames but also, would be able to track the person who had abandoned the object. This would be of great help in reducing the crimes like bomb-blasts and hoax bomb calls.

7. BIBLIOGRAPHY

Web Pages

- http://mha.cs.umn.edu/proj_events.html
- http://www.robots.ox.ac.uk/~misard/condensation.html
- http://opencv.willowgarage.com/wiki/
- http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html
- http://ai.stanford.edu/~dstavens/cs223b/stavens_opencv_optical_flow.pdf

Books

- Bradski and Kaehler, Learning *OpenCV* Computer Vision with the *OpenCV*. Library, O'Reilly Press

Research Paper

- CAMSHIFT Tracker Design Experiments with Intel OpenCV and SAI, Alexandre R.J. Fran, Institute for Robotics and Intelligent Systems, University of Southern California
- Abandoned Object Detection Using Multi-Layer Motion Detection, Sridha Sridharan, Vinod Chandran Simon Denman, Image and Video Research Laboratory, Queensland University of Technology, Australia

[illegible]

Brig (Retd.) S.P. Ghrera
HOD (CSE & IT)