DIVYADRISTI - A Real-time Hazard Detection System using Image Analysis

Anil Kumar Pulipaka
Junior Scientist
Central Food Technological Research Institute, Mysore

Krishna Rao S.N
Chief Scientist
Central Food Technological Research Institute, Mysore

Shilpa B.V
Assistant Professor
Sri Jayachamarajendra College of Engineering, Mysore

ABSTRACT
Divyadristi, a Linux-based software solution is the first of its kind differs from existing hazard detection system with no significant sensor-based circuitry and alarm-based trigger. Divyadristi offers a reactive hazard detection mechanism, extremely sensitive to both manned and unmanned hazards in the area of surveillance and features a fast trigger mechanism that quickly reports the hazard through a text message informing the level of the hazard in the area of surveillance and a multimedia e-mail composed of images that depict the state of the room only to the authorized personnel.

Divyadristi is optimally suited to places with static environments that are not prone to much movement; such environments include server rooms, storage rooms, lockers, radioactive material storage centres, and offices during non-working time and research centres.

General Terms
Hazard Detection, Image Analysis, Image Processing.

Keywords
Hazard Detection, Image Analysis, Motion Detection, Image Processing, Image Segmentation, Message, Alert

1. INTRODUCTION
Real-time Hazard Detection System using Image Analysis (Divyadristi) is an automatic hazard detection system and a surveillance system developed for CFTRI. Divyadristi basically needs an IP-based surveillance camera configured in such a way that the camera captures the state of the area of surveillance and sends the snapshots to a local computer using FTP. This system is dependent on tools like Gammu and Octave work in collaboration with JAVA-J2EE only on a Linux operating system preferably Ubuntu.

Divyadristi is dynamic and depends on the camera feed. The system detects manned intrusions and hazards like fire and smoke in the area of surveillance continuously at a preset time, which in contrast can out-do a smoke detector with the accuracy factor.

Divyadristi is optimally suited to places with static environments that are not prone to much movement. They can be used for the following circumstances to detect : Hazard due to short circuit in Computer Centre Server Rooms, Hazards in storage places, Hazards in bank lockers and normal lockers, Hazard due to radiation leakage in radioactive material storage centres and nuclear reactor rooms, Hazards in museums, offices and palaces, even houses during absence.

2. SYSTEM DESIGN
The conceptual view of Divyadristi is shown in Fig 1.

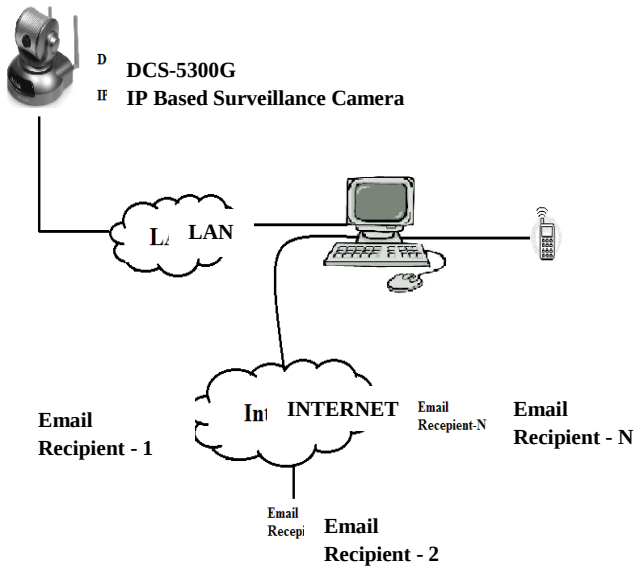

Fig 1: Conceptual View of Divyadristi

- The IP-based camera clicks snapshots and sends them to the computer through a LAN using File Transfer Protocol.
- The computer processes these images and notifies if there is a potential hazard.
- The notification takes place through a MIME formatted email and a text message through a mobile connected through that computer shown in Figure 1.

When the application is run for the first time, it asks the user to set a password. Once the user sets, becomes exclusive user of Divyadristi. If Divyadristi is not run for the first time then the control of the module is directly passed to the main GUI module of Divyadristi.

Divyadristi is basically a GUI that contains text-boxes for image-path, contacts, mail-ids, Spatiogram Similarity Level, Fault frequency, Password and also buttons for browse, set, start and pause operations, resume and stop operations, save-settings and load-settings operations, changing password and an add button for adding a new camera thread.

The Image-processor module invokes a set of image processing programs present as different source programs

which calculate spatiogram similarity level. These image processing programs calculate the histogram values based on those, spatiograms are formed. The spatiograms of the two images are compared and the magnitude of the similarity between the two images is recorded based upon the spatiogram values of both. In case of any intrusions or hazards i.e., if the current spatiogram similarity value is lesser than the maximum spatiogram similarity level, the two images are segmented using Euler's thresholding and stored in the folder of the camera's feed with its faulty image source. The folder name will have the current date and the camera position for record purposes.

Octave processor is a virtual module. It can be considered a sub-module of the image-processor. The octave programs access the images calculate the spatiogram similarity values and print them in the file 'Similarity.txt' and the control goes back to the Image-processor.

Euler thresholding is a non-parametric dynamic thresholding method, useful for many computer vision tasks. It requires the calculation of a graph relating thresholds to Euler numbers and performing a Rosin Unimodal threshold calculation on the graph. This can be done in time complexity O (N +T) where N is the number of pixels in the image and T is the number of thresholds used. The differential pixel values less than the Euler threshold will be whitened and the other parts will be blackened, thereby presenting the difference between the images accurately.

The Notify-by-SMS Module, in case of any hazards detected i.e., if the spatiogram similarity value is less than the maximum spatiogram similarity level, an alert is raised. This alert module has two actions. First one is the notification by an SMS containing the hazard level and the second one is notification by MIME (Multimedia Email). This module performs notification by SMS.

In the Notify-by-Delay Module, a user-defined API is invoked to create a text message containing the delay-information and reports about the potential breakdown in the camera's working and send it to the authorised personnel.

In the Notify-by-Resume Module, a text message about camera's working subsequent to its potential breakdown is sent to the authorised personnel.

3. SYSTEM IMPLEMENTATION

The implementation of Divyadristi consists of the following modules.

- Image-Processing using Octave
- Notification by SMS
- Notification by Mail
- Graphical User Interface

The Image-Processing using Octave includes the following algorithms with matlab and octave compatibility:

- Outer Spatiogram Module
- Spatiogram Patch Creation
- Spatiogram Comparison
- Segmentation Main Module
- Euler Threshold Calculation using Rosin's Algorithm

The Outer Spatiogram module calculates the spatiogram similarity values to detect hazards based on the maximum spatiogram similarity level.

The Spatiogram Patch Creation module creates a spatiogram and outputs a 3-D array to the Outer Spatiogram Module.

The Spatiogram Comparision Module compares and calculates the differential probability of two spatiogram patches in the 3-D array, which is called the spatiogram differential value (0 – 1).

1 – (spatiogram differential value) = spatiogram similarity value (0 – 1).

The Segmentation Main module is the outer module that invokes Euler Thresholding to produce a black image with white spots when there is a suspected intrusion. The process is explained as follows.

- Read the reference image and the faulty image into matrices A and B.
- Find (A-B) / 2, which is the matrix difference.
- Find the Euler threshold using Rosin's Algorithm, and
- If the matrix difference is greater than the Euler's Threshold, then the grey difference of the image is set to 0, which becomes the white part in the segmented image.

The Notification by SMS module is very important for the implementation of three high-level modules. They are:

- Notify by SMS.
- Notify Delay, and
- Notify Resume.

For implementing this, a common user-defined API is needed. The language used for the main module 'Divyadristi' is 'JAVA-J2EE'. Since the platform is UNIX, there exists a C-level package called 'Gammu' that acts as a PC-Suite for mobiles like Nokia and Alcatel mobiles in lieu with using J2ME. Using C one can implement a Gammu-based program and make the notify-by-SMS module to invoke that C-program to send the message to the authorised personnel.

The conceptual view of this module is shown in Figure 2.

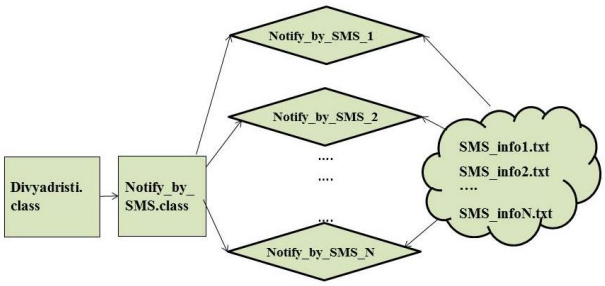

Fig 2: Conceptual View of SMS Notification Module

The 'Notify-by-SMS.class' invoke with proper synchronizations using file-based semaphores a set of C-executables purely based upon the value of 'N – Number of cameras running'. Basically Notify-by-SMS.class creates an information file called Sms_info with Camera_ID concatenated to it. The Sms_info file contains information regarding the camera's location and the hazard level.

There are two major tasks involved in implementing this module:

- Configuring the Gammu program to send the notification through a text message

- Invoking the Gammu-based C executable using JAVA without using J2ME

Sending SMS through Gammu from Command Line : With a configured connection, there is a utility of gammu, where one can send SMS using gammu through the command line.

The command, when pressed enter, sends an AT command to the mobile, which in turn sends the SMS to the actual recipients.

This command is used in the C executables. The C executable form a correct message in string format using the details in the Sms_info file and sends the AT commands to the mobile using the 'system ( )' function.

The above mechanisms are implemented in a C file called notify_by_sms.c and converted into an executable with the same name thereby creating a user-defined API. This is invoked by notifybysms.class module using Java System Runtime Command – mode Execution.

The main module 'Divyadristi' sends details about the room, paths of the images responsible for notification and the hazard level to this module. This module is implemented in Java Enterprise Edition. 'Javax.mail.jar' plays a big role in this module.

The GUI is designed keeping in mind all the high-level design aspects. The GUI is implemented using the 'JFrame' class in the JAVA-GUI language. The GUI contains components like JFrame, JTextArea, JPasswordField, JPanel, JDialogBox, JWindowPane and JButton. Threads are used to inter-link all the modules to the GUI.

4. RESULTS

Divyadristi when started, the terminal shows it is processing images. The current directory is checked whether the input file 'paths.txt' contained the correct path of the images and the output file 'similarity.txt' contained the spatiogram similarity values of all the images currently compared. All images are stored in a folder with current date and current camera-ID as its name.

When there is an intrusion or hazard, small variations are observed in the spatiogram similarity values thereby making it easy for studying the stability of the camera environment and the most preferable threshold for detecting hazards. As soon as intrusion is detected, it confirmed whether the reference image had dynamically updated itself or not.

As soon as hazards occurred, a folder called 'intrusions' is created and the images responsible for the invocation are copied into it. The folder called 'intrusions' contained the segmented black – white images, which ensure the images are being segmented whenever intrusion is detected, that is whenever the Current Spatiogram Similarity Level (CSSL) is less than Maximum Spatiogram Similarity Level (MSSL).

As soon as the hazards are detected, based upon the fault – frequencies of the SSL values, the creation of the sms_info file is confirmed. This ensure that the notification data is created correctly. Delays in the image reception produced delay_info file and Resume event produced resume_info file. All authorised personnel successfully received the notification message as shown in Figure 3.

In addition to this, all the authorised personnel are asked to check their email inboxes, which apparently has all mails from the Divyadristi Computer's server with successful attachments of the reference, faulty and segmented images in their inbox.

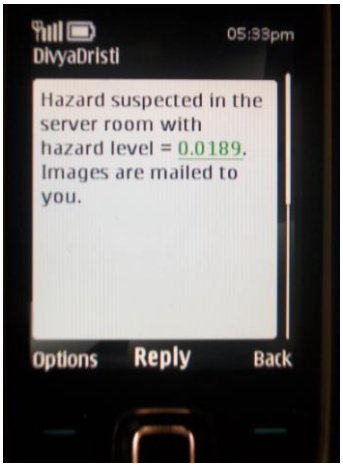

Fig 3: Notification as seen in the cell phone

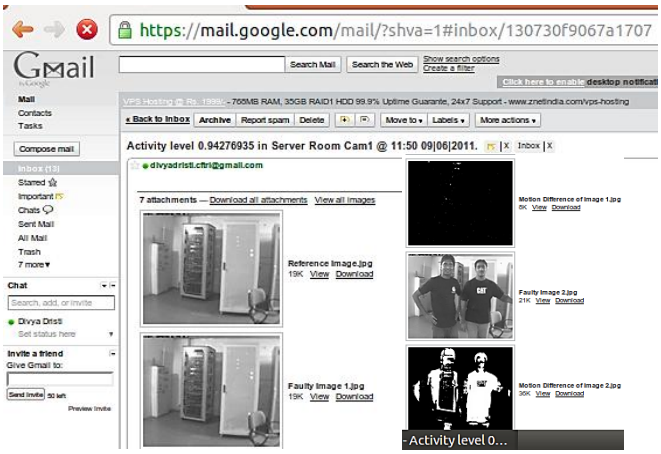

Fig 4: Email Notification as seen in the Inbox

For Divyadristi to detect hazards it needs to have a reference value to compare with. That reference value is the Maximum – Spatiogram Similarity Level (MSSL) or threshold. This value is very critical for running Divyadristi, since MSSL is the value with which Current – Spatiogram Similarity Value (CSSV) is compared with.

If MSSL is actually very high to the scale of the environment within the range of [0, 1], then even a small intensity variation or an LED blink would create a notification and send the SMS and E-mails to the authorised personnel. Obviously the authorised personnel would not want notifications for such small variations or reasons.

If MSSL is too less to the scale of the environment within the range of [0, 1], Divyadristi might ignore some potential hazards, since they would not be detected as a hazard.

Hence selecting the correct MSSL is very critical for image-processing and intelligent – notifications.

In Figures 5 and 8, the stability of the camera environment in order to speculate the most suitable MSSL for Divyadristi has been studied and speculated. Basically these graphs are the behaviour of the CSSV of two cameras covering the same

vicinity of surveillance. Following the graphs, images depict the actual event creating disturbance in the camera environment, thereby supporting the graphical representations of image variances in different scenarios.

The details of two test cases are as follows:

Case – 1: Behaviour of CSSV with a probable smoke in the camera environment for 60 seconds.

Case – 2: Behaviour of CSSV with a small-time human intrusion in the camera environment for 60 seconds.

Case – 1:. With intrusion of smoke, it can be seen that the CSSV values are dropped very low towards 0.9995. After the detection of smoke, since there is no state change again, the reference image automatically updated. Hence the stability went near equilibrium.

From the graph shown in Figure 5, it is observed that, MSSL suitably lies between (0.9995 – 0.9999) since the smoke is detected in the 14th second.

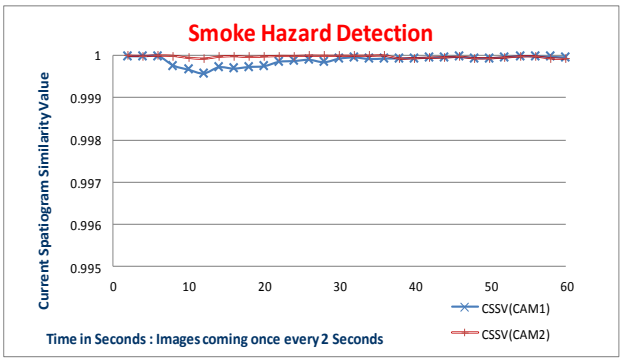

Fig 5: Graph during Smoke hazard Detection

The images shown in Figures 6 and 7 indicate the state change due to smoke detection in the camera environment. The images include the reference image, the faulty image indicating the state change and the segmented image.

The segmented image indicates the exact spot of the smoke in form of a white patch against black background. The smoke is identified as a hazard by the Camera-1 due to area of coverage being large in case of Camera-1. The smoke identified as a hazard has been clearly indicated as a white patch in the segmented black and white image shown in Figure 7

Fig 6: Faulty image indicating the smoke intrusion

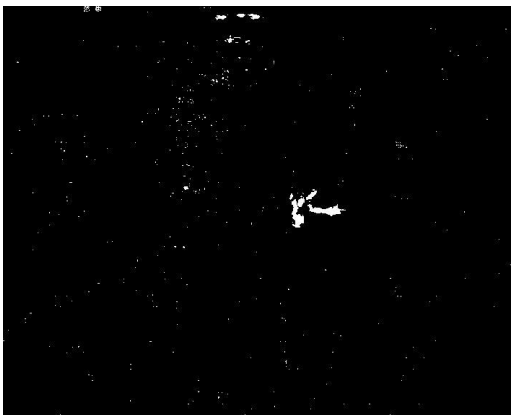

Fig 7: Segmented image indicating the area of smoke intrusion detection

Case – 2:. Human Intrusion is the most obvious hazard that can be detected very easily due to the mass of the human body. As portrayed in Figure 8, when the human came at the 9th second the threshold or CSSV had dropped to 0.99 in the case of Camera-1 and 0.9935 in the case of Camera-2.

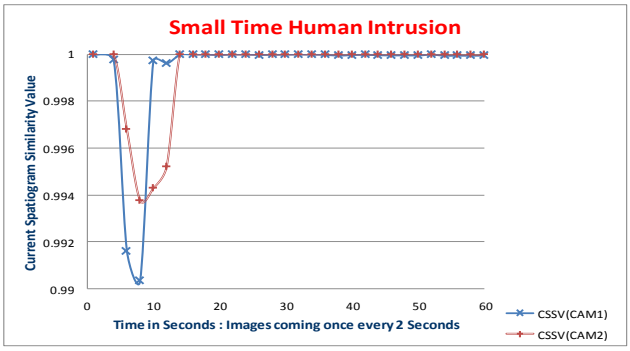

Fig 8: Graph indicating small time human intrusion

Since this happened to be a very small time intrusion and the human disappeared from the scene, the CSSVs stabilised further. But the notifications are sent as usual.

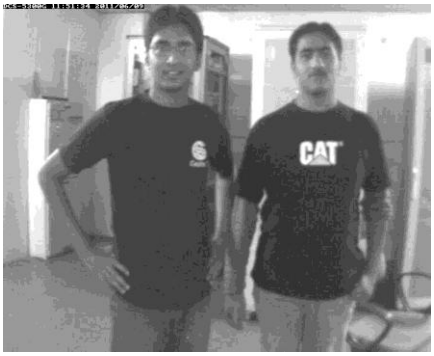

Fig 9: Image indicating small time human intrusion

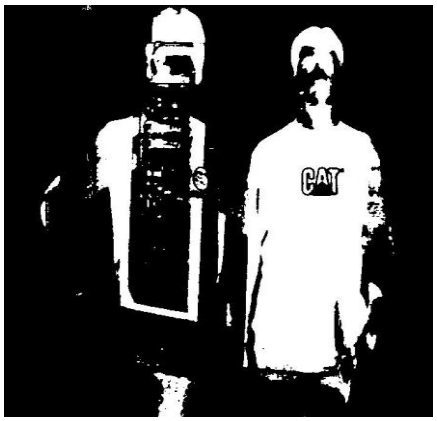

Fig 10: Segmented Image indicating the area of human intrusion detection

The images shown in Figure 9 and Figure 10 indicate the faulty image indicating the human intrusion and the segmented image.

The purpose of a real-time image-analysis-based hazard detection system is a successful idea, executed and tested successfully.

Divyadristi is completely dependent on the camera feed. If at all, the camera stops feeding the images, the program anyway would notify the breakdown. The intruder cannot afford to make any mistake when he enters this environment. Even if he touches a wire or moves a small pin, the variation is caught as an intrusion. Divyadristi is very sensitive and limited to static environments.

5. CONCLUSION

Divyadristi is a potential Linux-based software solution, that can do the hazard detection in real-time and prevent potential hazards or intrusions and most importantly record the details of the source of the intrusion or the hazard at an early time.

6. ACKNOWLEDGEMENTS

The authors would like to thank the Director, CFTRI for providing an opportunity to carryout this innovative research work and encouragement for successful completion and implementation. The authors like to thank Mr. Manas HR and Mr. Atul R Pai for their support in development of codes during this research work.

7. REFERENCES

[1] P.L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection", 6th British Machine Vision Conf., Birmingham, pp. 347-356 1995.

[2] P.L. Rosin, Unimodal Thresholding, Pattern Recognition, 34(11):2083-2096,2001.

[3] Max Bajracharya, Jet Propulsion Laboratory, California Institute of Technology,Pasadena, CA, USA, Single Image Based Hazard Detection for a Planetary Lander : Proceedings of the 5th Biannual World Automation Congress, 2002.

[4] Ciar´an ´O Conaire Centre for Digital Video Processing, Dublin City University, Ireland, Efficient Euler-Number Thresholding: Technical Report, Aug 2005.

[5] Ó Conaire, Ciarán and O'Connor, Noel E. and Smeaton, Alan F. (2007) An improved spatiogram similarity measure for robust object localisation. In: ICASSP 2007 - IEEE International Conference on Acoustics, Speech, and Signal Processing, 15-20 March 2007, Honolulu, Hawaii.

[6] Girish Singh Rajput, Zia-ur Rahman, Old Dominion University, Electrical and Computer Engineering Department, Norfolk, Virginia 23529, Hazard Detection on Runways using Image processing techniques: Proceedings of SPIE, the International Society for Optical Engineering, ETATS-UNIS 2008.

[7] http://www.dlink.com/products/?pid=342/

[8] http://en.wikipedia.org/wiki/Color_histogram/

[9] http://www.vipan.com/htdocs/javamail.html/

[10] http://en.wikipedia.org/wiki/GNU_Octave/

[11] http://en.doc.centreon.com/HowToSendSMSWithGammu/

[12] http://www.vipan.com/htdocs/javamail.html/

[13] http://www.control.com.sg/at_commands_sms.aspx

[14] http://www.cftri.com/

[15] http://en.wikipedia.org/wiki/Smoke_detector/

[16] http://en.wikipedia.org/wiki/Burglar_alarm/