A Major Project Report

on

A REAL TIME FIRE DETECTION AND VIDEO ALERTING

submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

CHINTAPALLI GANGADHAR (17211A0549)

BHEEMA PRAVEEN KUMAR (17211A0535)

BIRADAR TULSIDAS (17211A0539)

BOGGULA ANAND (18215A0506)

Under the guidance of

Dr. AMJAN SHAIK, PhD

Professor, Associate HOD

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.V.RAJU INSTITUTE OF TECHNOLOGY

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narspur, Medak(Dist.), Telangana State, India - 502313

2020 - 2021

B. V. Raju Institute of Technology

(UGC Autonomous, Accredited By NBA & NAAC)
Vishnupur, Narspur, Medak (Dist.),
Telangana State, India – 502313

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project entitled **"A REAL TIME FIRE DETECTION AND VIDEO ALERTING"**, being submitted by

CHINTAPALLI GANGADHAR (17211A0549)

BHEEMA PRAVEEN KUMAR (17211A0535)

BIRADAR TULSIDAS (17211A0539)

BOGGULA ANAND (18215A0506)

In partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING to B.V.RAJU INSTITUTE OF TECHNOLOGY is a record of bonafide work carried out during a period from May 2020 to July 2021 by them under the guidance of **Dr. AMJAN SHAIK**, Professor/Associate HOD, CSE Department.

This is to certify that the above statement made by the students is/are correct to the best of my knowledge.

Dr. AMJAN SHAIK
Professor/Associate HOD

The Project Viva-Voce Examination of this team has been held on ____________________.

Dr. Ch. Madhu Babu
Professor & HoD-CSE

EXTERNAL EXAMINER

B. V. Raju Institute of Technology
(UGC Autonomous, Accredited By NBA & NAAC)
Vishnupur, Narspur, Medak (Dist.),
Telangana State, India – 502313

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"A REAL TIME FIRE DETECTION AND VIDEO ALERTING"** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology and submitted in the Department of Computer Science and Engineering, B. V. Raju Institute of Technology, Narsapur is an authentic record of our own work carried out during a period from May 2020 to July 2021 under the guidance of **Dr. AMJAN SHAIK**, Professor/Associate HOD. The work presented in this project report has not been submitted by us for the award of any other degree of this or any other Institute/University.

CHINTAPALLI GANGADHAR (17211A0549)
BHEEMA PRAVEEN KUMAR (17211A0535)
BIRADAR TULSIDAS (17211A0539)
BOGGULA ANAND (18215A0506)

<u>ACKNOWLEDGEMENT</u>

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion. Whatever we have done is due to such guidance and assistance. We would not forget to thank them.

We thank **Dr. AMJAN SHAIK** for guiding us and providing all the support in completing this project. We are thankful to **Mrs. SREEDEVI**, our section project coordinator for supporting us in doing this project. We thank the person who has our utmost gratitude is Dr**. Ch. Madhu Babu**, Head of CSE Department.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the staff members of CSE Department.

CHINTAPALLI GANGADHAR (17211A0549)
BHEEMA PRAVEEN KUMAR (17211A0535)
BIRADAR TULSIDAS (17211A0539)
BOGGULA ANAND (18215A0506)

A REAL TIME FIRE DETECTION AND VIDEO ALERTING

ABSTRACT

With advances in computing and telecommunications technologies, digital images and video are playing key roles in the security and surveillance. Most of the fire detection are performed by sensor-based systems which have perceived the temperature and smoke by themselves and utilized in various type of industry after combining with the fuzzy theory. The computer vision community has expended a great amount of effort in recent years towards the goal of tracking fire in images. Much more recently, algorithms have been developed to track in real-time. The goal of this project is to implement a system based on one of those algorithms, in order to the people in a database of image footage. Due to several constraints and performance issues, however, a more straightforward algorithm based on background subtraction is implemented and shows acceptable performance levels. Fire Detection is an end user application which detects the fire in real time video that is being given to the system. The purpose of the present project was to develop an intelligent system for detecting fire in real time. We show the relative performance achieved against prior work used before to illustrate maximally robust real-time fire region detection.

Key Words: Haar Classifier, Image processing, OpenCV.

TEAM MEMBERS:

1. CHINTAPALLI GANGADHAR – 17211A0549
2. BHEEMA PRAVEEN KUMAR – 17211A0535
3. BIRADAR TULSIDAS – 17211A0539
4. BOGGULA ANAND – 18215A0506

GUIDE:

Dr. AMJAN SHAIK, Professor & Associate HOD

CONTENTS

1. INTRODUCTION

Application of fire detection as tool has increase to due to the frequent occurrence of extended fire with consequences on human health and security. This current detection methods which are based on electronic sensors are usually depend on heat and pressure sensors. However those methods has a fatal flaw where they will only work when a certain condition has been reach. In the worst case scenario is the sensors are damaged or not being configure properly can cause heavy casualty incase of real fire. To solve these problems in electronics surveillance cameras being installed. Due to this there is an increase of need for fire detection based on computer vision for such devices. Such devices include a wide range of CCTV, wireless camera even to UAVS.

These type of systems offer several distinguish advantages over those traditional detection methods. For example the cost of using this type of detection is cheaper and the implementation of this type system is greatly simpler compare to those traditional methods. Secondly the response time of fire detection system is faster compare to any other traditional detection methods since a vision sensorbased fire detection system does not required any type conditions to trigger the sensors and it has the ability to monitor a large area depends on the camera used.

To develop the economy, the number of large high buildings is increasing. Generally, we can find fire detection as a difficult technique. Used in industries, commercial. It has more advantages towards Security. This can be reduced through image processing and IOT sensors. Proposed system usually would include the use of open-source technologies and helps in security and surveillance.

1.1 MOTIVATION

To predict “Fire detection” is the main motive of this project. The technique is demonstrated by developing a system that locates fire in cluttered scenes. In particular, the system detects the components of a fire body in an image. Whenever the user gives real time video streaming, it is taken as input , he/she would be getting the fire detection in an image as output. Results are of highest accuracy. Conclusion is made on the based on HAAR classifier for fire detection. This work can be used in Computer science, security and surveillance .

1.2 PROBLEM STATEMENT

Fire detection is an end user application which detects the Fire in the real time image being given to the system. The relevant detection in the image is shown on the screen which alerts the security team with the sound system. We have implemented HAAR classifier for fire detection.

1.3 OBJECTIVE

- Our main objective is to investigate Fire detection in real time and understand it.
- To detect fire using fire detection and give the relevant output. HAAR are used for fire detection respectively.

2. LITERATURE SURVEY

Fire and Smoke Detection without Sensors: Image Processing Based Approach:

There square measure scores of hearth detection systems within which the color data is employed as a pre-processing step. They used color predicate data record and also the temporal variation of little set of pictures to understand hearth in video sequences. A manually divided fire place set is employed to coach a system that understand recognizes fire like color pixels. He training set is used to form a look-up table for the fire detection system. This offer the use of generic lookup table if the training set is not available. They used chromatic and dynamic features to extract real fire and smoke in video sequences. They employ a moving object detection algorithm in the pre-processing section. The moving objects are filtered with fire and smoke clear out to raise an alarm for feasible fire in video. They used a generic fire and smoke model assemble the corresponding filter. They proposed a real-time algorithm for fire detection in video sequences. They combined motion and color clues with fire flicker analysis on wavelet domain to detect fire. The paper author have used a mixture of ten three dimensional Gaussians in RGB color space to model a fire pixel using a training set. They have employed the fire color model developed. Later on they have employed the same fire detection strategy to detect possible smoke samples which is used as early alarm for fire detection.

Flame Detection using Image Processing Techniques:

The fire is characterized using efficient features and detection of the equal uses of an appropriate processing. From every image the pixel is checked for the presence or absence of fire using color detection and periodic behavior in lire regions is likewise analyzed. They use combined approach of color detection, motion detection and area dispersion to detect fire in video data. Firstly the algorithm locates desired color regions in video frames. And then determines the region in the video where there is any movement, and in the last step they calculate the pixel arm of the frame. The combination Automatic of color, motion and area clues is used to detect fire in the Video.

Automation fire detection systems use physical sensors to detect and response of fire. The physical sensor utilizes the substance properties noticeable all around are gained by sensor and use by the recognition framework to raise a caution. This can likewise cause false cautions, the physical sensors are additionally not relevant holders, for open air condition and in substantial framework settings, for example, aircraft large tunnels. Due to the fast development of digital camera technology and superior content of high quality pixel based photo and video processing, there is a major trend to switch traditional fire detection system with computer vision based system.

Automated Fire Extinguishing System With Gsm Alarm:

Specifications, layout problems and solutions for the fire extinguishing- system project fulfilling the necessities. A fire lighter can be able to extinguish fire quickly, averting the damages and reduce losses. Technology has joined the gap between firefighting and machines using some effective method the purpose of this are to establish a system. It in the most limited time subject to a few effective factors. In the system aims to put out the fire before it spreads increasing the security of home, laboratory, office, factory and building that is important to human life.

3. ANALYSIS

The aim in this phase is studying the existing system and later is to interpret the necessities, requirements and domain of the new system. The mentioned both activities are balancingly salient, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the effects and essentials of a new system is more difficult and requires creative thinking and understanding of existing running system is also strenuous, improper understanding of present system can lead deflection from solution.

3.1 EXISTING SYSTEM

Identification of the Fire detection is made possible by various techniques, but these usually don't always generate the desired results. Therefore, pointed out the need for a general application which predicts the most accurate results. The existing system is deals with sensors. The system provides fire detection in real time capturing.

DISADVANTAGES

- Most of these approaches are costly to implement through sensors.
- The results produced are also less accurate, i.e., through image processing techniques .
- Complexity in code for building.
- Hardware connections may lose sometimes.

3.2 PROPOSED SYSTEM

The proposed system is an application where in the user can predict the fire detection in real time capturing. The time consumed is less in this and also the results obtained are comparatively more accurate to the existing system. Proposed system usually would include the use of open source technologies that would help out in many aspects. Open source technology OpenCv is used to detect the fire and detect it.

ADVANTAGES:

- Sound alerting the security team.
- Accurate and less time consuming.
- The project is very useful in Security surveillance and fire detection.
- No need of separate hardware devices(i.e. sensors).

3.3 SOFTWARE REQUIREMENT SPECIFICATION.

3.3.1.PURPOSE

A system with any operating system must have python (>= 3.3), OpenCV (python3-version) and PyCharm installed as Functional Requirement. As a part of Nonfunctional requirements, or performance requirements or quality of service requirements, the system must be usable, available, reliable, supportable, testable and maintainable.

3.3.2.SCOPE

The fundamental extension is to increase the dataset. Also the time taken to generate the result is very less. The accuracy of the project may be increased with the by providing the dataset. At present, the project is detecting fire further analysis can be added for increased performance. Robust spontaneous fire recognizer can be developed and deployed in real-time systems. This is going to have an impact in Security surveillance for detecting the fire.

3.3.3 OVERALL DESCRIPTION

Fire detection is a crucial and challenging problem in visual surveillance. Automatic monitoring of the fire detection areas is important for safety control and urban planning. For this purpose many techniques and methods have been proposed. These techniques are not producing high performance and better accuracy for complicated scenes. Recently Foreground Extraction and Background substraction based methods are proposed, which provides a better accurate solution for Fire detection. This literature survey discusses some of the existing methods and their performance.

Fire detection is an end user application which detects the Fire from the real time capturing being given to the system.

3.4 ECONOMIC FEASIBILITY

The system is economically feasible. It does not require any addition hardware or software besides the normal working system with Python (>= 3.3),OpenCV (python3-version), Pycharm GUI. Since the interface for this system is developed using the existing resources and technologies available online. There is no expenditure for certain.

3.5 SOFTWARE REQUIREMENTS

These software requirements are the specification of the system.. The software requirements provide a basis for creating the software requirements specification. These are useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the teams programs throughout the development activity.

Operating System	:	Windows XP
Coding Language	:	Python
Front End	:	PyCharm GUI

3.6 HARDWARE REQUIREMENTS

These hardware requirements serve as the basis for a contract for the implementation of the system and hence are a complete and consistent specification of the whole system. These may be used by software engineers as the starting point for the system design. These specify what the system does and not how it should be implemented.

Processor	-	Pentium –IV
Speed	-	1.1 Ghz
Ram	-	256 Mb
Hard Disk	-	20 Gb
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse

4. IMPLEMENTATION

The application is based on fire detection. A real time approach to detect the fire is used in this project. We implement HAAR Classifier to analyze the fire from the given video streaming. The frames are divided and background subtraction is applied to detect the fire from the divided frames. This ensures the detection of the fire and gives the sound alerting to the security and surveillance team to respond accordingly. Fire recognition application will be able to connect and send a photo to a server. The server will detect fire in that image, perform fire recognition and, using the recognized images dataset can be implemented. The relevant output to that image will be displayed. Image fire detection and recognition software which will be used to generate fire images and to test recognition performances.

Modules

The following **modules** are developed:

1. **The Fire Detection module** will locate fire in a given image and separate them from the scene. The extracted fire detected will be further transformed using processing techniques like gray scale transformation (fire detected used in recognition will be gray scale images because are less sensitive in background changes and more computationally efficient), histogram equalization (for consistent brightness and contrast), resizing (to have the same dimension as images in the training database). Histogram equalization and resizing are used to reduce the recognition module's lighting and scaling sensitivity.
2. **The Fire Recognition module** will recognize fire provided by the detection module. It will use the HAAR Classifier for recognition and Principal Component Analysis for dimensionality reduction. This phase divides the video into frames and uses the background subtraction for the recognition module to be implemented. From the original video then optical flow video is obtained through detection of fire and finally alarm would be generated. The recognition module will return a list of detected images with IDs, IDs used to extract information from the MySQL database.

3. **Application module** will be developed using Java Micro Edition for platform independence. The user will be able to choose server's address and the Fire detection will be alerted. Using a file browser, the user will provide real time video capturing and send it to the server waiting for results. Finally, the application will display the if any fire is detected and alert the system.

4. **Capturing module:** will perform a fire recognition. It will integrate
detection, recognition and database modules. Running in training mode, the software will display an interactive console. After that, it will detect fire and alert the particular users were it has been implemented

The fire detection and recognition modules are written in C++ using Object Oriented style to provide more clarity and re-usability. OpenCV it is used for implementing fire detection and recognition algorithms and for advanced image processing techniques. More information on the implementation of the project can be found on the wiki page of each page.

4.1 TECHNOLOGIES USED

This Fire detection is implemented using Python technology. Python language is one of the most flexible languages and can be used for various purposes. It is one of the best languages which is used to implement machine learning techniques as it contains a lots of special libraries for machine learning namely scipy and numpy that can be used for linear algebra and getting to know kernel methods of machine learning. The language is considerable to use when waged with machine learning algorithms and has uncomplicated syntax relatively.

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself. As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned. Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

Python Tools

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas Numpy intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time and 3D walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team.

In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months] and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site. On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Andriod and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL.

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

PYCHARM (IDE)

Pycharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm Features

- **Project Code Navigation -** Instantly navigate from one file to another, from method to its declaration or usages, and through classes hierarchy. Learn keyboard shortcuts to be even more productive
- **Code Analysis -** Take advantage of on-the-fly code syntax, error highlighting, intelligent inspections and one-click quick-fix suggestions to make code better
- **Python Refactoring -** Make project-wide code changes painlessly with rename, extract method/superclass, introduce field/variable/constant, move and pull up/push down refactorings
- **Web Development with Django -** Even more rapid Web development with Django framework backed up with excellent HTML, CSS and JavaScript editors. Also with CoffeeScript, Mako and Jinja2 support
- **Google App Engine Support -** Develop applications for Google App Engine and delegate routine deployment tasks to the IDE. Choose between Python 2.5 or 2.7 runtime
- **Version Control Integration -** Check in, check-out, view diffs, merge all in the unified VCS user interface for Mercurial, Subversion, Git, Perforce and other SCMs
- **Graphical Debugger -** Fine tune Python or Django applications and unit tests using a full-featured debugger with breakpoints, stepping, frames view, watches and evaluate expressions
- **Integrated Unit Testing -** Run a test file, a single test class, a method, or all tests in a folder. Observe results in graphical test runner with execution statistics
- **Customizable & Extensible -** Bundled Textmate, NetBeans, Eclipse & Emacs keyboard schemes, and Vi/Vim emulation plugin

4.1 UML DESIGN

4.1.1 UML Diagram

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

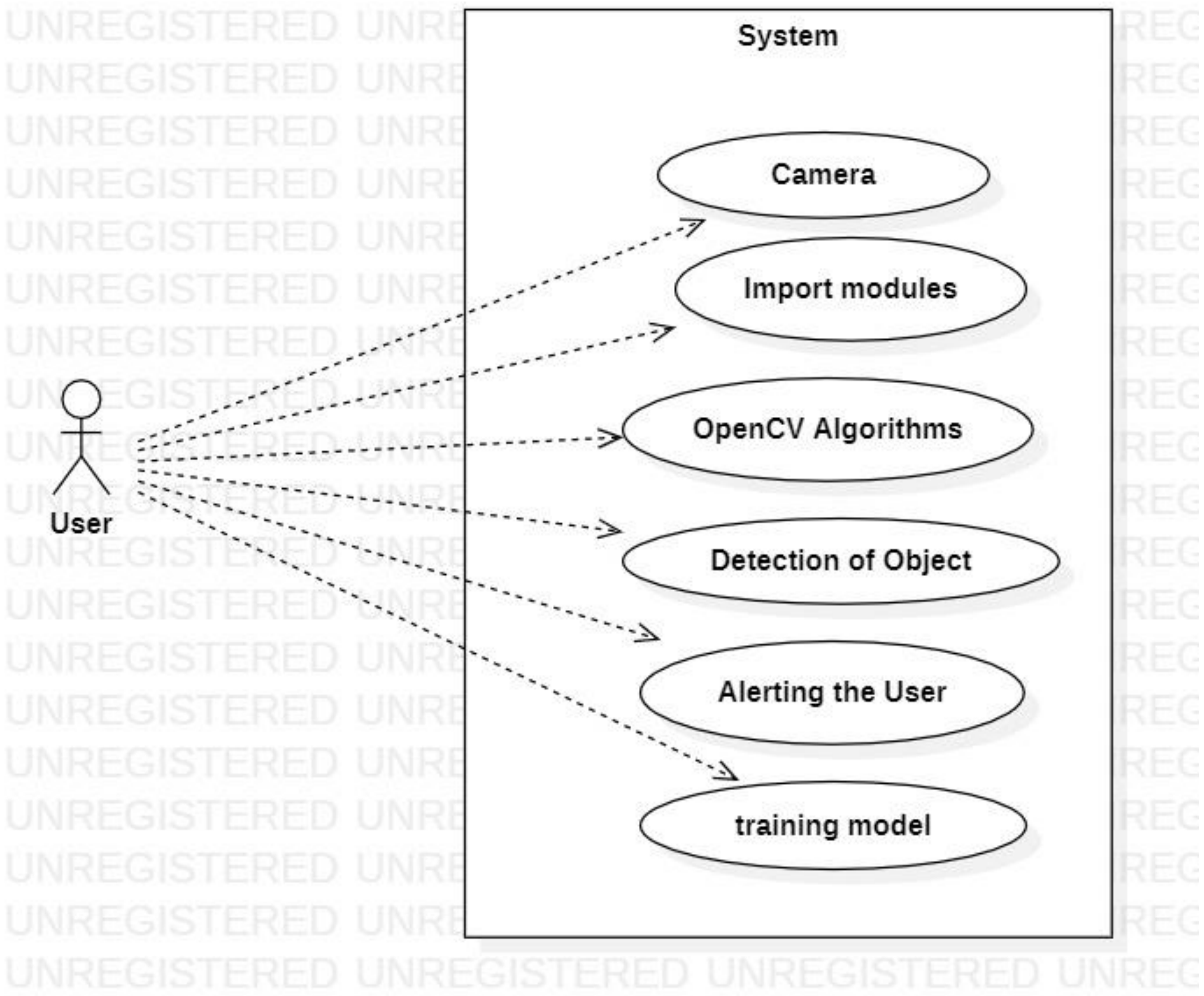
GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

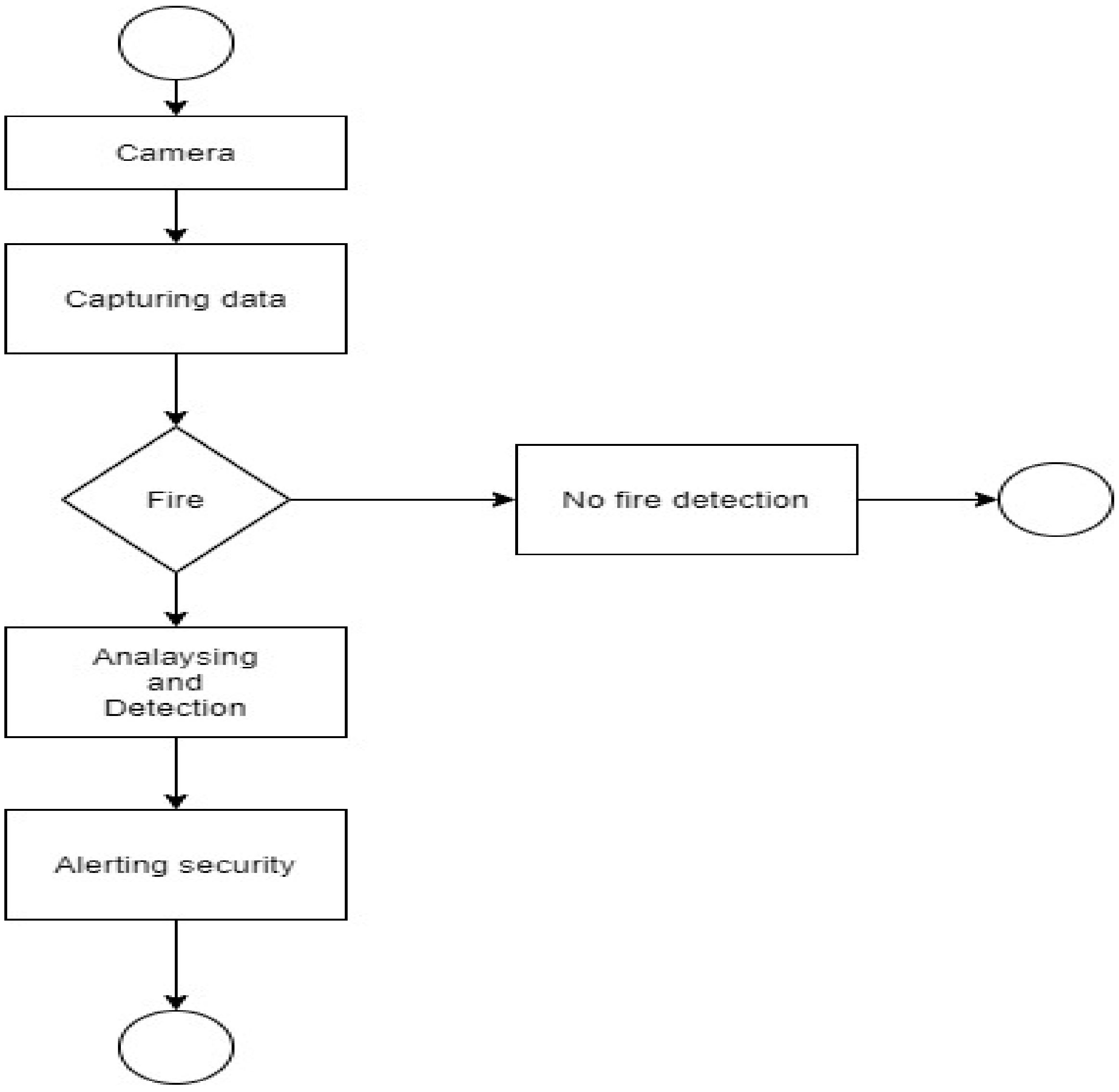
4.2.1. Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



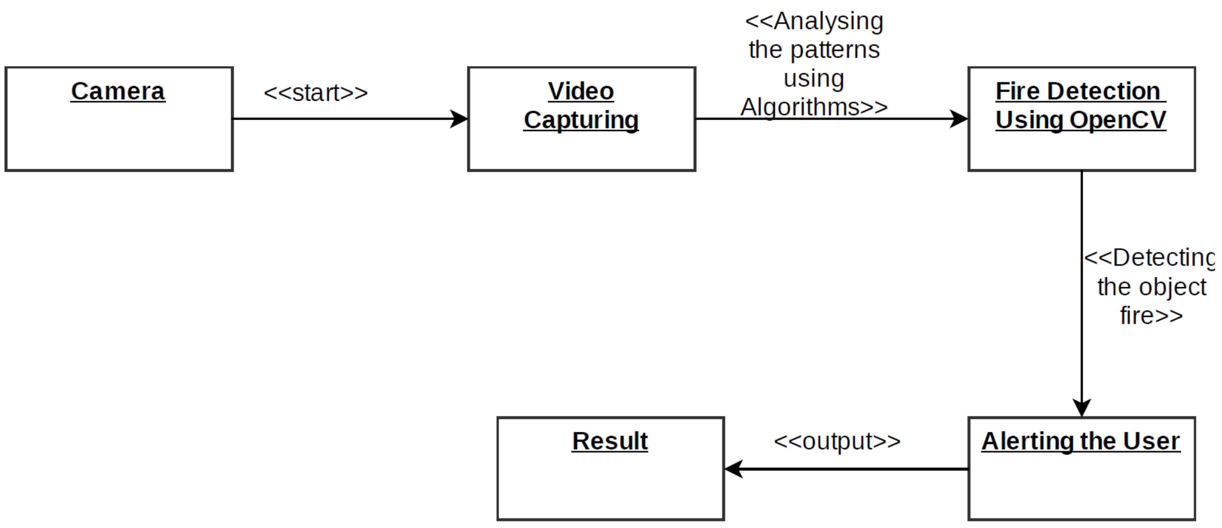
4.2.2Activity diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows.



4.2.3 Collaboration Diagram

Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task. Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes. Collaboration between objects working together provides emergent desirable functionalities in Object-Oriented systems. Each object (responsibility) partially supports emergent functionalities. Objects are able to produce (usable) high-level functionalities by working together. Objects collaborate by communicating (passing messages) with one another in order to work together.



Camera
<<start>>
Video Capturing
<<Analysing the patterns using Algorithms>>
Fire Detection Using OpenCV
<<Detecting the object fire>>
Alerting the User
<<output>>
Result

4.2.4 Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

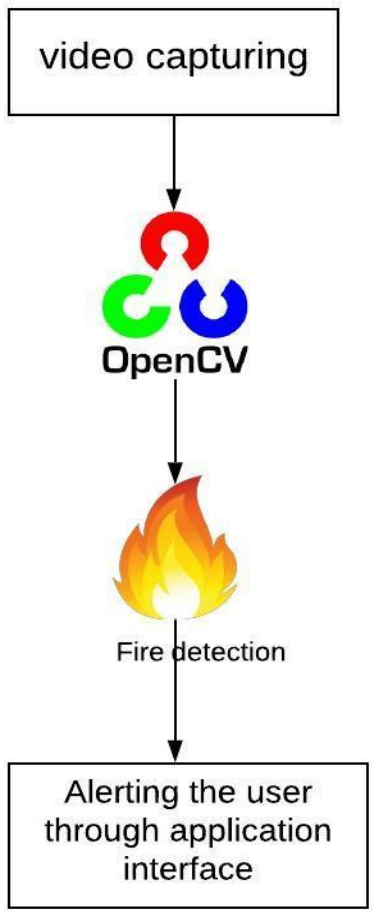


Figure 4.1.4 COLLABORATION DIAGRAM

Architecture design of the Project

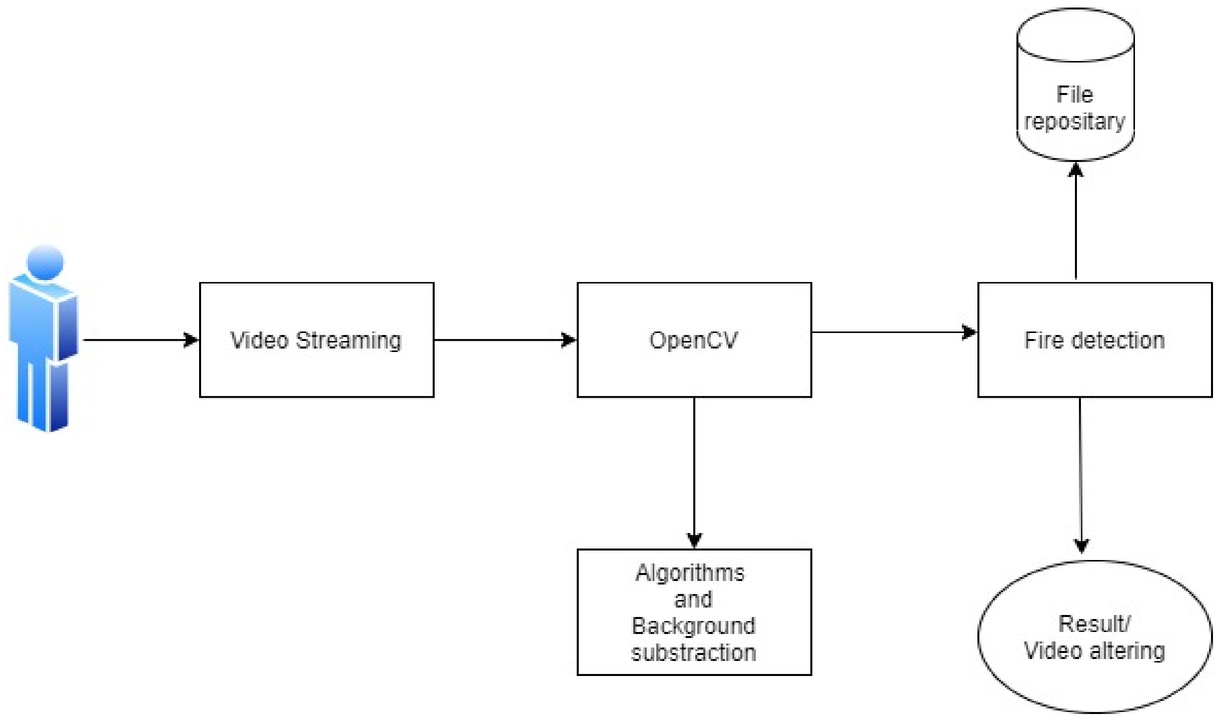


Figure: 4.1.3 Architecture design

4.2 ALGORITHM

HAAR classifier

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has fourstages:

1. HAAR Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object. Lets take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it. First step is to collect the HAAR Features. A HAAR feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

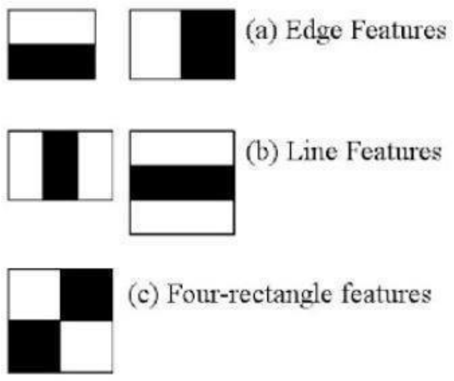


Fig. 4.3: Features of HAAR cascade classifier.

Integral Images are used to make this super fast.But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the propertythat the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.

This is accomplished using a concept called Adaboost which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier.

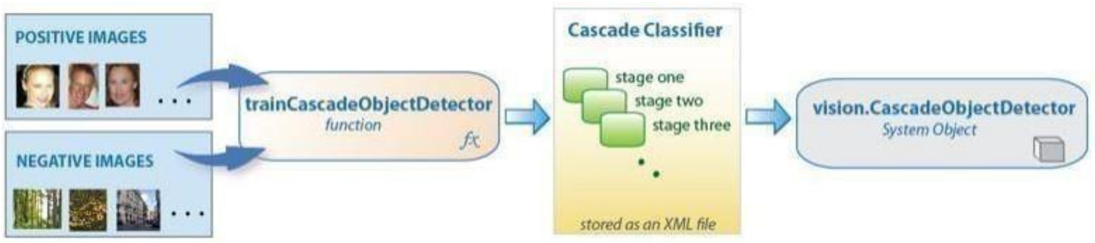


Fig. 4.4:Stages of Cascade classifier.

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage.

The detector reports an object found at the current window location when the final stage classifies the region as positive. The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest.

Machine Learning is a rapidly growing Haar Classifier is a supervised classifier, it has mainly been used for facial detection but it can also be trained to detect other objects. Computer vision is such a growing field that there are so many resources available for your own personal use. OpenCV provides a lot of functionality for machine learning techniques and the Haar Classifier is one of them.

The Haar Feature supported by OpenCV is an object detector initially proposed by Paul Viola and Michael Jones in Rapid Object Detection using a Boosted Cascade of Simple Features

Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the black rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

This technique harbors on the concept of using features rather than pixels directly. Which can then be easier to process and gain domain knowledge from correlating feature sets. It has also proven to be exceptionally faster.

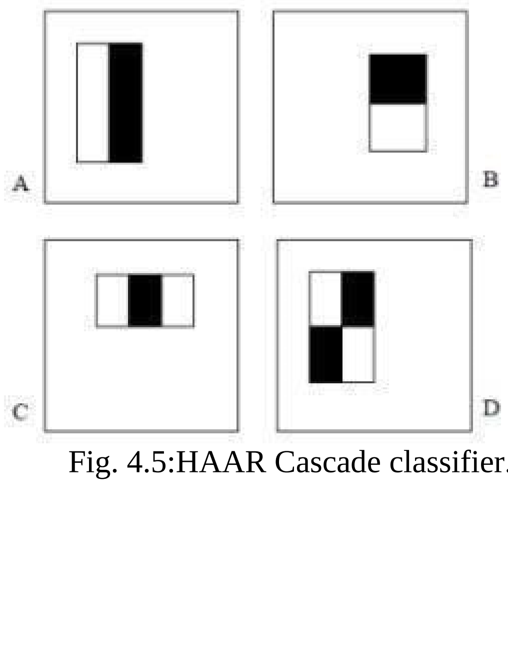


Fig. 4.5:HAAR Cascade classifier.

The two features are shown in the top row and then overlayed on a typical training cavity in the bottom row. The first feature measures the difference in intensity between the region of the top portion with the region right below. The feature capitalizes on the observation that the top region is often darker than lower parts of the cavity (a gradient). The second feature compares the intensities in the middle regions to the intensity across the outer edges of the cavity.

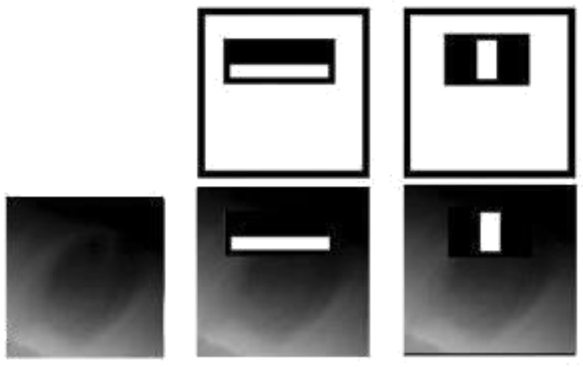


Fig. 4.6:HAAR Cascade

A Haar Classifier is really a cascade of boosted classifiers working with haar-like features. Haar-like features are specific adjacent rectangular regions at a specific location in a window (as shown in the first image above). The difference is then used to categorize subsections of an image and separates the non-objects from objects. Due to this distinction it is more of a weak learner, where you must use a large number positives to get as many Haar-like features as possible to accurately describe an object with some type of correlation and accuracy. So with all of these essentially weak classifiers we combine them into our classifier cascade to hopefully form a strong learner, by way of boosting.

Boosting is the concept of creating a set of weak learners and combining them to make a single strong learner. In the implementation of detecting solar cavities I used the AdaBoosting algorithm (aka Adaptive Boosting). This algorithm was initially proposed by Yoav Freund and Robert Schapire in A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. It is used to improve a learning algorithms performance, by calling weak classifiers repeatedly and updating weights.

4.3 SAMPLE CODE

Python Code :

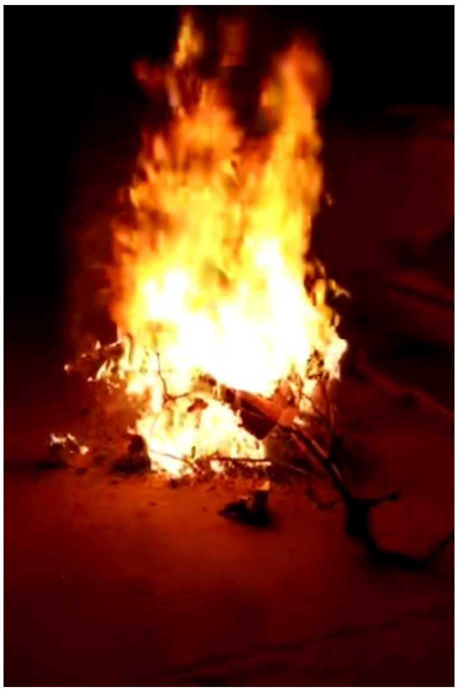
```
#import the necessery packages
import numpy as np
import cv2
import time
import winsound
duration = 2000
freq = 440
#fire_detection.xml file & this code should be in the same folder
while running the code
fire_cascade = cv2.CascadeClassifier('fire_detection.xml')
cap = cv2.VideoCapture(0)
ret, frame1 = cap.read()
prvs = cv2.cvtColor(frame1,cv2.COLOR_BGR2GRAY)
hsv = np.zeros_like(frame1)
hsv[...,1] = 255
while 1:
    #seperating frames from the video
    ret, img = cap.read()
    #implementing optical flow algorithm
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    next = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    flow = cv2.calcOpticalFlowFarneback(prvs,next, None, 0.5, 3,
15, 3, 5, 1.2,0)
    mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
    hsv[...,0] = ang*180/np.pi/2
    n=hsv[...,0]
    #normalization of hsv conversion
    hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
    #converting to BGR imge
    bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
    cv2.imshow('orginial video',img)
    cv2.imshow('optical flow video',bgr)
    #implementing fire detection
    fire = fire_cascade.detectMultiScale(img, 1.2, 5)
    #background subtraction
    blur = cv2.GaussianBlur(img, (21, 21), 0)
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
    lower = [18, 50, 50]
    upper = [35, 255, 255]
    lower = np.array(lower, dtype="uint8")
    upper = np.array(upper, dtype="uint8")
    mask = cv2.inRange(hsv, lower, upper)
    output = cv2.bitwise_and(img, hsv, mask=mask)
    res = cv2.bitwise_and(img,img, mask= mask)
    for (x,y,w,h) in fire:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        print ('Fire is detected..!')
        winsound.Beep(freq, duration)
        time.sleep(0.2)
    cv2.imshow('img',img)
    cv2.imshow('back ground iamge',res)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

5. TEST CASES

Test Id	TestCase Name	TestCase Description	Expected	Test Case Priority	Status
1.	Output of Application	Detection for the Fire in the low Light video Streaming	Fire Detection	High	Pass
2.	Output of Application	Detection for the Fire in the normal video Streaming	Fire Detection	Low	Pass
3.	Output of Application	No Detection of the fire	No Detection	Medium	Pass
4.	Output of Application	Detection for the Fire in the high Light video Streaming	Fire Detection	Medium	Fail
5.	Output of Application	No Detection of the fire In open places	Fire Detection	High	Pass

Table.5.1 Test Cases.

6.OUTPUT



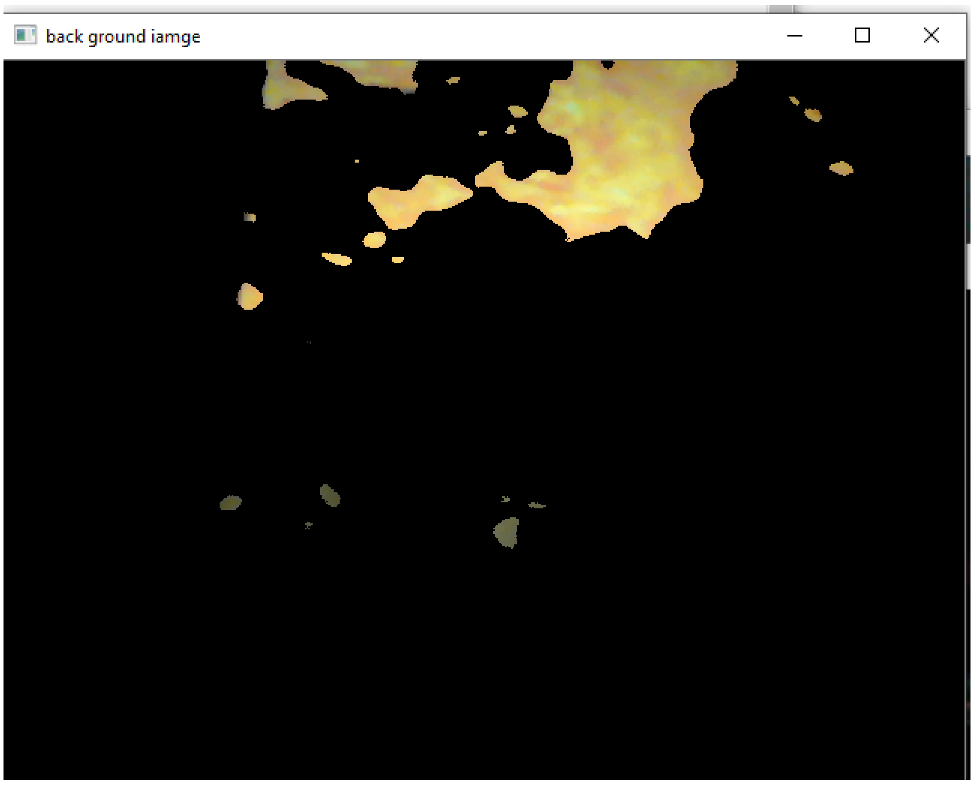
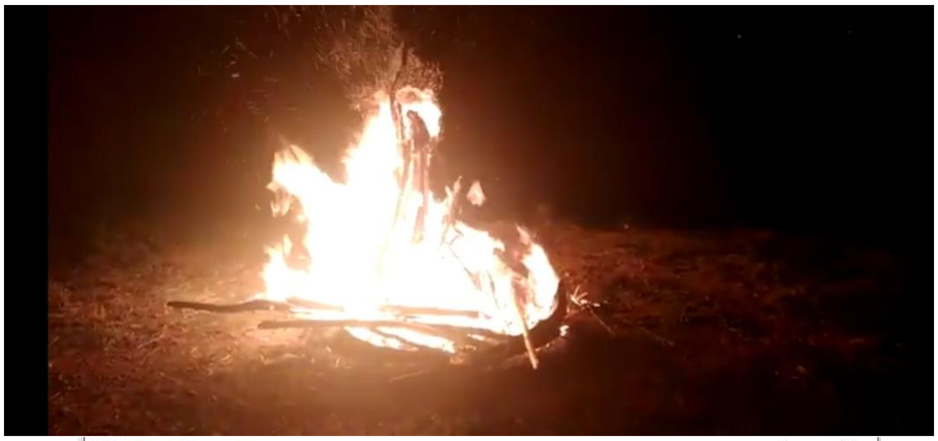


Fig.6.1:Output for Fire Detection.



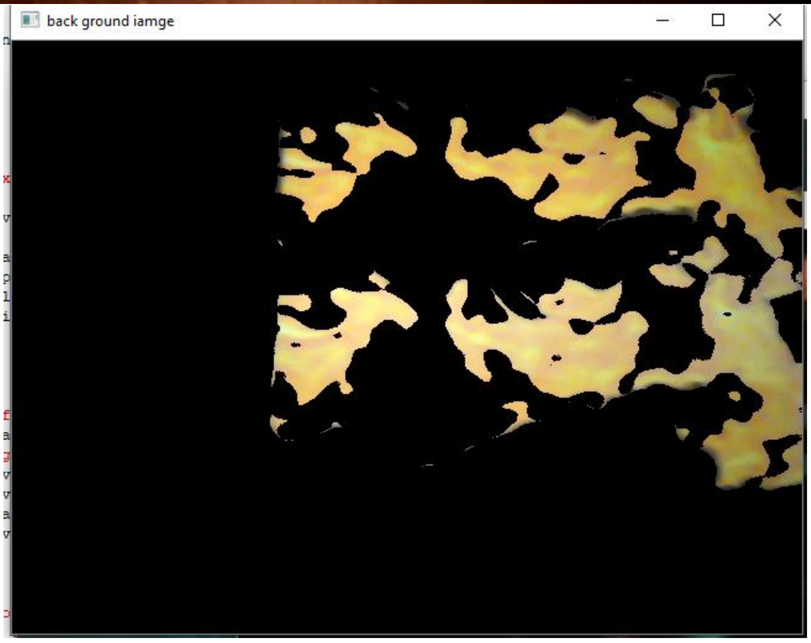


Fig.6.1:Output for Fire Detection

In the above figures, i.e., Sample output is the window where the fire detected and alarm sound from the system is generated when fire is detected. On detecting the fire, respective detection is shown on the screen. These changes with the change in the video streaming. Hence this real time application is very beneficial in various fields like Security and surveillance, computer science.

7.CONCLUSION

Proposed is a Fire Detection using Opencv to predict the Fire. This real time application is very beneficial in various fields of Security and surveillance. Early detection of fire is very important to disaster management systems for which several CNN based fire detection methods using edge intelligence arepresented to date. These methods have reasonable accuracy and execution time and are applicable to only certain environment. In case of uncertain environment having fog, smoke, and snow, their performance is limited. In addition, it is difficult to deploy computationally expensive fire detection models on resource constrained devices.

8. FUTURE ENHANCEMENT

The fundamental extension is to add the dataset. Also the time taken to generate the result is very less. The fire symptom detection will be built in real hardware system and it can be monitored from mobile and PC. The new system will be added with alarm system. The accuracy of the project may be increased with the increase in the dataset. At present, the project is detecting fire from through real time capturing in a video from the particular distance this can be improved to be used in the open areas such as forest where fire can be detected. Robust spontaneous detection and Alerting can be increased with the help of machine learning and deep learning. This is going to have an impact on our day to day life by enhancing the way we interact with computers or in general, our surrounding living and work spaces. High correct recognition rate , significant performance improvements in our system. Promising results are obtained under fire registration errors, fast processing time. System is fully automatic and has the capability to work with video feeds as well as images. It is able to recognize spontaneous Fire if detected. Our system can be used in Digital Cameras where in the image is captured and provide the results for the fire detection in open areas. In security systems which can detect the fire and alert the system. This project can be implemented in huge open areas with wide cameras so that it can detect the fire from the far distance too. Police and security can use the system in order for security and surveillance.

9. REFERENCES

[1]. “Efficient Fire Detection for Uncertain Surveillance Environment” by Khan Muhammad and Hassan Ahmed IEEE,FEB-2019.

[2]. R. Di Lascio, A. Greco, A. Saggese, and M. Vento, "Improving fire detection reliability by a combination of videoanalytics," in International Conference Image Analysis and Recognition, 2014, pp. 477-484.

[3]. Jareerat Seebamrungsat, Suphachai Praising, and Panomkhawn Riyamongkol, " Fire Detection in the Buildings Using Image Processing " IEEE, 2014. ICT(International Student Project Conference).