# FOREST FIRE DETECTION USING IMAGE PROCESSING TECHNIQUES

Main Project work submitted to Avinashilingam Institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted By

**A. Razina (19PIT007)**

Under the guidance of

**Dr. T.Jayamalar M.C.A., M.Phil., Ph.D,**

Assistant Professor, Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

Coimbatore-641043

May 2021

**DECLARATION**

# DECLARATION

I hereby declare that the project entitled **"FOREST FIRE DETECTION USING IMAGE PROCESSING TECHNIQUES"** is a record of the original work done by Razina. A(19PIT007) under the guidance of Dr. T. Jayamalar M.C.A.., M.Phil., Ph.D., Assistant Professor, Department of Information Technology, school of physical sciences and computational sciences, Avinashilingam Institute for Home Science and Higher Education for Women, in the partial fulfilment for the degree of Master of Science in Information Technology and this project has not formed the basis for any Degree/Diploma/Associates.

Place:

Date:

Signature of the Candidate

_____

Countersigned by

Dr. T. Jayamalar M.C.A., M.Phil., Ph.D.,

Assistant Professor and Head Department of Information Technology,

School of Physical Sciences and Computational Sciences

# CERTIFICATE

# CERTIFICATE

This is to certify that this project work entitled "**FOREST FIRE DETECTION USING IMAGE PROCESSING TECHNIQUES"** done by **ARAZINA.**(19PIT007) has been submitted to Avinashilingam Institute for Home science and Higher education for women,Coimbatore-43 in partial fulfillment of the requirement for the award of the **POST GRADUATE ININFORMATION TECHNOLOGY.** This Project has not found the basis for the award of any Degree/Associate/fellowship or similar title to any Candidate of any University. Certified as a bonafied record of the work submitted for the Viva voce held on _____.

Signature of the HOD

Signature of the Guide

Signature of the Examiner

Signature of External

FORUS TECHNOLOGIES

**Date: 03/05/2021**

## TO WHOMSOEVER IT MAY CONCERN

This is to certify the student **Ms.RAZINA A(19PIT007**) pursuing his final year M.Sc.,(IT) in Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore has completed her internship with project Session on "**FOREST FIRE DETECTION USING IMAGE PROCESSING**" in our concern starts from February 2021 to April 2021.

All the Best for her Future!

**For Forus Technologies**

Forus Technologies CBE-09

**Administrator**

**No:27,Ist Floor,Mani's Colony Kalingrayan Street,RamNagar,Coimbatore-641012.**

**73970 28889 / info@forustechnologies.com**

# ACKNOWLEDGEMENT

# ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for his constant love and grace that he has showered upon me, which kept me in good health, and sound mind without which my project would not have reached a successful end.

I would like to express my deep sense of reverential gratitude and sincere thanks to

**Shri, Dr. S.P.Thyagarajan, Chancellor,**Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities during my course of study.

I owe my great deal of gratitude to **Dr.PremavathyVijayan M.Sc., M.Ed., Dip. Spl. Edn., M.Phil., Ph.D., Vice Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for extending all resources that facilitated the smooth conduct of the project study.

I express my gratitude to **Dr. S. Kowsalya, Registrar,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities and support necessary for the study.

I wish to extend my sincere thanks **Dr. K. Udaya Chandrika M.Sc., M.Phil., Ph.D., Dean School of Physical Sciences and Computational Sciences**, for her support and valuable guidance.

I wish to extend my sincere thanks **Dr. D. ShanmugaPriya, M.Sc., M.Phil., Ph.D., Head, Department of Information Technology,** for imparting tremendous assistance and well-timed support for triumph of our project.

I heartily thank my esteemed project guide **Dr.T.Jayamalar, M.C.A., M.Phil., Ph.D., Assistant professor, Department of Information Technology,** for imparting tremendous assistance and well-timed support for triumph of our project.

I would like to express my sincere gratitude to all the staff members of the Department of Information Technology, for their constant encouragement and for the opportunity to do our project in this esteemed university. Last yet importantly, i would like to thank my parents, family members, friends and all well-wishers for their kind inspiration, blessings and encouragement during the course of project.

# ABSTRACT

# ABSTRACT

Forest fire detection is one of the important system in this world since forest fire is more crucial to damage lot of lives and properties, it is difficult to control once it is exposed. Forest-fires are real threats to human lives, environmental systems and infrastructure. The hundred of millions of hectares are destroyed by wildfires in each year. The only efficient way to minimize the forest fires damage is adopt early fire detection mechanisms.

Forest fires represent a real threat to human lives, ecological systems, and infrastructure. Many commercial fire detection sensor systems exist, but all of them are difficult to apply at large open spaces like forests because of their response delay, necessary maintenance needed, high cost, and other problems. These images will be further processed by using the software, SPYDER. This project uses RGB color space .Along with this smoke, area detection is also performed using its color characteristics.

This method uses RGB and gray color space for image representation and formats. The advantage of using RGB color space is that it can separate the red channel, blue channel and green channel more effectively. The performance of this algorithm is tested on where the fire is detecting and which the area is marked.

The input image would be preprocessed by adding some noise. The image representation and formats the gray level and color image in this image transform to make Salt and pepper noise. The noise includes salt and pepper noise to make the image more clarity. The noise would be treated using histogram equalization. The forest fire would be detected using binary threshold technique and watershed segmentation technique.

This project can also be served for forest fire prediction.

# CONTENT

# TABLE OF CONTENT

## LIST OF FIGURES

# CHAPTER 1

## INTRODUCTION

Fires represent a constant threat to ecological systems, infrastructure and human lives. Past has witnessed multiple instances of fires. With the faster and faster urbanization process, more and more high-rise buildings appear around us. This also can make the frequency of fire increase and bring great losses to people's lives and property. In areas where fire would pose an unreasonable threat to property, human life or important biological communities, efforts should be made to reduce dangers of fire. As the damage caused by fires is so tremendous that the early fire detection is becoming more and more important. Recently, some fire detectors have been used in many places, they used the smoke, temperature and photosensitive characteristics to detect fires. But they are too worse to meet the needs in a large space, harsh environment or the outdoor environment etc.

Traditional fire detection methods use mechanical devices or humans to monitor the surroundings. The most frequently used fire smoke detection techniques are usually based on particle sampling, temperature sampling, and air transparency testing. An alarm is not raised unless the particles reach the sensors and activate them. Some of the methods are mentioned below:

A. Fire Watch Tower

In watch towers human are made to observe the location throughout. If any fire occurs, he reports it. However, accurate human observation may be limited by operator fatigue, time of day, time of year, and geographic location.

B.Wireless Sensor Networks

In a wireless sensor-based fire detection system, coverage of large areas in forest is impractical due to the requirement of regular distribution of sensors in close proximity and also battery charge is a big challenge.

C.Satellite and Aerial

Monitoring Satellites based system can monitor a large area, but the resolution of satellite imagery is low. A fire is detected when it has grown quite a lot, so real time detection cannot be provided.

Moreover, these systems are very expensive. Weather condition (e.g. clouds) will seriously decrease the accuracy of satellite-based forest fire detection as the limitations led by the long scanning period and low resolution of satellites.

The motivation for an image processing-based approach is due to rapid growth of the electronics. Fire detection are one of the most important components in surveillance systems used to monitor buildings and environment as part of an early warning mechanism that reports preferably the start of fire. Currently, almost all fire detection techniques use image processing technique. Image processing methods consist of image pre-processing, image filtering, image transformation and image analysis that primarily depend on the reliability and the positional distribution of the image. Due to the rapid developments in digital camera technology and video processing techniques, there is a big trend to replace conventional fire detection techniques with computer vision-based systems. In our project different characteristic parameters of fire i.e Color, smoke, Area and motion using image processing in PYTHON are analyzed.

In order to create a color model for fire and smoke, we have analyzed the images which consist of fire. Gray space is chosen intentionally because of its ability to separate color channel in red channel, blue channel, green channel. The rules defined for RGB color space in order to detect possible fire-pixel candidates can be transformed into gray space and analysis can be performed.

Color alone is not enough to identify fire. There are many things that share the same color as things that are not fire, such as a desert, sun, red leaves and other objects. The key to distinguishing between the fire and the fire colored objects is the nature of their motion.

Fire detection is done by analyzing difference in images of image pixel. There are three main parts in pixel detection: frame/background subtraction, background registration, and pixel detection Similar to the fire detection. The smoke pixels do not show chrominance characteristics like fire pixels. At the beginning, when the temperature of the smoke is low, it is expected that the smoke will show color from the range of white-bluish to white. Toward the start of the fire, the smoke’s temperature increases and it gets color from the range of black-grayish to black. Area detection method is used to detect dispersion of fire pixel area in the sequential frames.

## 1.1 COMPANY DETAILS

Name of the company : Forus Technologies

Name of the Manager : Prem Chander

Company Address : 27,1st Floor, kalingrayan Street,Ramnagar,Cbe-27

Contact Number : 7890678956

Email Id : Forustechnologies@gmail.com

Working hours : 10 AM-6 PM

Website Address : www.forustechnologies.com

## 1.2 ORGANISATION PROFILE

FORUS Technologies is a leading Software Development Concern in Coimbatore managed by IT veterans with more than a decade experience in the leading Software Technologies. It's a Mid Sized which is growing fast and always looking to hire talent into its current team of 10-50 employees. Follow this company and get updates when a company adds info about Forus Technologies, its history, its product and services and what they do. Forus Technologies will be sharing news & info here and will be freely communicating with all its followers. If you represent this company, then add info here.

At Forus Technologies, we focus on delivering client satisfication based on high end solution with innovations, within a short span of time. Forus technologies has become a demanding software solution provider in the ITmarket. Forus Technologies at each and every step of action. we make creativity as a habit and Innovation as a Product.KCS focus on providing open source based software model and to provide a cost effective solution to our customer with open source tools.

## 1.3 About the platform

### 1.3.1Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

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

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting and was discontinued with version 2.7.18 in 2020. Python 3.0 was released in 2008 and 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 consistently ranks as one of the most popular programming languages.

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

1) Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program.It will take only one line to execute, while Java orCtakes multiple lines.

3) Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

8) Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas,

Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C,C++ Java. It makes easy to debug the code.

11. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

12. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to **x,** then we don't need to write **int x = 15.** Just write x = 15.

### 1.3.2 Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package asit ended up being useful on its own and for other things than Python. There is also a small, bootstrap version ofAnaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

Before version 20.3, when pip installed a package, it automatically installed any dependent Python packages without checking if these conflict with previously installed packages. It would install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, could find that it stopped working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package would appear to work but produce different results in detail. While pip has since implemented consistent dependency resolution, this difference accounts for a historical differentiation of the conda package manager.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open-source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with anaconda.

**Anaconda Navigator**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

### 1.3.3 Spyder IDE

**Spyder** is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software. It is released under the MIT license.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first-party and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylintand Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts.

Spyder uses Qt for its GUI and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

Features include:

- An editor with syntax highlighting, introspection, code completion
- Support for multiple I Python consoles
- The ability to explore and edit variables from a GUI
- A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand
- A debugger linked to IPdb, for step-by-step execution
- Static code analysis, powered by Pylint
- A run-time Profiler, to benchmark code
- Project support, allowing work on multiple development efforts simultaneously
- A built-in file explorer, for interacting with the file system and managing projects
- A "Find in Files" feature, allowing full regular expression search over a specified scope
- An online help browser, allowing users to search and view Python and package documentation inside the IDE
- A history log, recording every user command entered in each console. An internal console, allowing for introspection and control over Spyder's own operation.

# CHAPTER 2

# REVIEW OF LITERATURE

T. Celik and Hasan Demirel et al. 2016 further enhance system that uses a statistical color model with Fuzzy logic for fire pixel classification. The proposed system develop two models; onebased on luminance and second based on chrominance.Fuzzy logic uses the YCbCr color space for the separation of luminance from chrominance instead of using color spaces such as RGB.Existing historic rules are replaced with the Fuzzy logic to make the classification more robustand effective. This model achieves up to 99.00% correct fire detection rate with a 9.50% false alarm rate.

R. Gonzalez-Gonzalez et al.2018 proposed a method to detect fire by smoke detection based on wavelet. In this smoke detection method,image processing on video signals is proposed. The SWT transform is used for the area detection of ROI's. This method comprises of three steps.
In the first step,preprocessing is performed and the image is resized and transformed to grayscale image. Finally indexed the image using indexation. The second step involves high frequencies of an image is eliminated using SWT and reconstruct the image by inverse SWT. Inorder to group the intensity colors that are closed to each other is the main purpose of image indexation.

Histogram analysis is used to determine the indexation levels. After that compare the image with a non-smoke frame and selecting those pixels that are change from one scene toanother. The final stage consists of smoke verification algorithm in order to determine whether ROI is increasing its area and to reduce the generation of false alarm. These three steps are combined together to form the final result.

Hidenori Maruta et al.2019 proposed another method for smoke detection based on support vector machine. In this approach robust and novel smoke detection method is proposed using support vector machine. Firstly preprocessing is performed by extracting moving objects of images. The preprocessing consists of five steps: image subtraction and accumulation,image binarization, morphological operation, extraction of Feret's regions and creation of the image mask. Image subtraction is used to extract regions of moving object. In order to eliminate noise

like regions binarization and morphological operations are used. The position and approximated shape of the object is obtained by identifying Feret's diameteris called feret'sregion.

After preprocessing, perform texture analysis and extract the texture features. These texture features become the component of feature vector. Feature vector is applied as the input vector and support vector machine is smoke ornot. Smoke detection method involves three steps: analyzing texture features, discrimination of Feret's region using support vector machine and time accumulation.

In order to extract feature vectors of the image, texture analysis is performed in this method. Support vector machine is used to classify smoke or non- smoke from the extracted image. The main advantage of this method is that more accurate extraction of smoke areas in image can be obtained using SVM.

Abitha.T.E and Paul.P 2018.proposed another method that uses covariance descriptors for fire detection.In this method,color,spatial and domain information are combined by using covariance descriptors for each spatiotemporal block. The blocks are generated by dividing the flame colored region into 3D regions. This method used a covariance matrix for the detection of flames. Background subtraction method is not used in this approach. To detect fire,divides the video into spatio temporal blocks and covariance features are computed from these blocks. Using an SVM classifier, the flame colored region are classified by using the spatial and temporal characteristics.These classified flame color regions are tested using video data that contain flames and flame colored objects.For the classification of pixel colors chromatic color model is used and analyzed fire colored pixels.Object detection and texture classification are performed by applied covariance descriptors.

In order to define spatio-temporal blocks temporal derivatives are calculated along with spatial parameters. Spatial temporal blocks can be defined using covariance matrices.Then compute the covariance values of the pixel property vectors in spatio-temporal blocks.

For classification,support vector machine is trained.According to the number positively classified video blocks and their positions, confidence value is determined for fire detection. This

method is computationally efficient. Covariance approach is well suited for detection of flames. Drawback of this method is that, it is well suited when the fire is clearly visible. If the fire is fareway from the camera and covered with dense smoke,this method performs poorly.

Mehdi Torabnezhad et al.2016 proposed another method that used image fusion technique to detect smoke. In this method, combine visual and thermal information to improve the rate of fire detection. The invisibility of smoke in LWIR image can distinguish smoke from smoke like objects. Infrared images do not detect smoke in the images but can detect smoke like object. By combining visible and IR images smoke can be distinguished. Based on characteristics of visual and thermal smoke images a potential smoke mask is created. In-order to reduce false alarms,PSM is further analyzed by disorder measurements and energy calculations. For the detection of short range smoke visible and IR image fusion algorithm is used. Scope of this paper is to detect the smoke as an indicator of fire. Here visible and infrared images are combined together to distinguish smoke from smoke like objects. Earlier approach that uses sensor or visible images only gives false alarm.

Visible images capture both smoke and smoke like objects.In frared images do not capture smoke.Integrating these images give correct information about smoke.Objective of this paper is to save people, forest from the fire. By this method generation of false alarm can be reduced to a great extent. The proposed algorithm consists of two phases. In the first phase combine visual and thermal information of the smoke and potential smoke mask is generated.PSM is again analyzed to differentiate true and false alarms. This method is very efficient and detects smoke successfully. Improves the fire detection rate and reduces the generation of false alarm. The drawback of this method is Correct and punctual detection of fire is not possible and comparison is required to identify smoke.

Mubarak A.I Mahmoud and honge ren. 2018 Several fire detection algorithms have been proposed by various researchers. Thou-Ho et al. presented fire detection algorithm, which combines the saturation channel of the HSV color and the RGB color. This algorithm employs three rules ($R \geq G > B$), ($R \geq RT$), and ($S \geq ((255-R)\ ST/RT)$).

Determination of the two thresholds RT and ST is required. The certain values range is from 115 to 135 for RT and from 55 to 65 for ST based on many investigational results done by the authors. This method is computationally simple compared to the other algorithms; however, it suffers from false-positive alarms in case of moving fire-like objects. Dios et al. presented an optical model used to detect forest fires and measure the properties of the fire such as flame height, fire front, fire base width, and flame inclination angle.

This system is very good; nevertheless, it is very expensive because it consists of infrared cameras and other technologies such as GPS and telemetry sensors.

Yinglian et al.2017 proposed forest fire disaster prevention algorithm based on image processing. This algorithm depends on fire and smoke color properties to identify fire. Yinglian's algorithm is good, but the smoke spreads quickly and it has many different colors which depend on the burning material; thus, the false alarm rate rise

This system is very good; nevertheless, it is very expensive because it consists of infrared cameras and other technologies such as GPS and telemetry sensors. Yinglian et al. proposed forest fire disaster prevention algorithm based on image processing. This algorithm depends on fire and smoke color properties to identify fire. Yinglian's algorithm is good, but the smoke spreads quickly and it has many different colors which depend on the burning material; thus, the false alarm rate rise

Yuanbin Wang,Langfei Dang 2019 In this method,color,spatial and domain information are combined by using covariance descriptors for each spatiotemporal block. The blocks are generated by dividing the flame colored region into 3D regions. This method used a covariance matrix for the detection of flames. Background subtraction method is not used in this approach. To detect fire,divides the video into spatio temporal blocks and covariance features are computed from these blocks. Using an SVM classifier, the flame colored region are classified by using the spatial and temporal characteristics.These classified flame color regions are tested using video data that contain flames and flame colored objects.For the classification of pixel colors chromatic color model is used and analyzed fire colored pixels.Object detection and texture classification are performed.

Sukuan Jin,Xiaobo Lu 2019 the method of forest fire detection combining image processing and

machine learning based on video sequences. The method consists of three parts: moving object detection, image feature extraction and classifier recognition. An optimal algorithm combination of the above three parts is found through comparative experiments. In this paper, the LBP feature is improved based on color information. Before the moving object detection, image segmentation step is added in a novel way to reduce the false positive rate. Experimental results show that the proposed method yields good performance.

Joao Alives,Christophe soares,Jose M.Torres,pedro sobral,Rui S.Moreira 2020 This method presents an automatic fire detection system designed to identify forest fires, preferably, in their early stages. The system pipeline processes images of the forest environment and is able to detect the presence of smoke or flames. Additionally, the system is able to produce an estimation of the area under ignition so that its size can be evaluated.

In the process of classification of a fire image, one Deep Convolutional Neural Network was used to extract, from the images, the descriptors which are then applied to a Logistic Regression *classifier*. At a later stage of the pipeline, image analysis and processing techniques at color level were applied to assess the area under ignition. In order to better understand the influence of specific image features in the classification task, the organized dataset, composed by 882 images, was associated with relevant image metadata (eg presence of flames, smoke, fog, clouds, human elements).

In the tests, the system obtained a classification accuracy of 94.1% in 695 images of daytime scenarios and 94.8% in 187 images of night time scenarios. It presents good accuracy in estimating the flame area when compared with other approaches in the literature, substantially reducing the number of false positives and nearly keeping the same false negatives stats.

Renjie Xu,Haifeng Lin,Lin Cao and Yunfei Liu.2021.This method relies heavily on manmade features, which is not universally applicable to all forest scenarios. In order to solve this problem, the deep learning technology is applied to learn and extract features of forest fires adaptively. However, the limited learning and perception ability of individual learners is not sufficient to make them perform well in complex tasks. Furthermore, learners tend to focus too much on local information,

namely ground truth, but ignore global information, which may lead to false positives. In this paper, a novel ensemble learning method is proposed to detect forest fires in different scenarios.

Firstly, two individual learners Yolov5 and Efficient Det are integrated to accomplish fire detection process. Secondly, another individual learner Efficient Net is responsible for learning global information to avoid false positives. Finally, detection results are made based on the decisions of three learners. Experiments on our dataset show that the proposed method improves detection performance by 2.5% to 10.9%, and decreases false positives by 51.3%, without any extra latency.

Qingjie Zhang,Jia,long Xu,Liang Xu,Haifeng Guo 2019. This method proposed a deep learning method for forest fire detection. We train both a full image and fine grained patch fire classifier in a joined deep convolutional neural networks (CNN). The fire detection is operated in a cascaded fashion, ie the full image is first tested by the global image-level classifier, if fire is detected, the fine grained patch classifier is followed to detect the precise location of fire patches. Our fire patch detector obtains 97% and 90% detection accuracy on training and testing datasets respectively. To facilitate the evaluation of various fire detectors in the community, we build a fire detection benchmark. According to our best knowledge, this is the first one with patch-level annotations.

Tom Toulouse, Lucile Rossi. Turgay Celik 2017. This method presents a comparative analysis of state-of-the art image processing-based fire color detection rules and methods in the context of geometrical characteristics measurement of wild land fires. Two new rules and two new detection methods using an intelligent combination of the rules are presented, and their performances are compared with their counterparts. The benchmark is performed on approximately two hundred million fire pixels and seven hundred million non-fire pixels extracted from five hundred wildland images under diverse imaging conditions. The fire pixels are categorized according to fire color and existence of smoke; meanwhile, non-fire pixels are categorized according to the average intensity of the corresponding image.

This characterization allows to analyze the performance of each rule by category. It is shown that the performances of the existing rules and methods from the literature are category dependent, and none of them is able to perform equally well on all categories. Meanwhile, a new proposed method based on machine learning techniques and using all the rules as features outperforms existing state-of-the-art techniques in the literature by performing almost equally well on different categories.

# CHAPTER 3

# METHODOLOGY

Forest fires represent a real threat to human lives, ecological systems, and infrastructure. Many commercial fire detection sensor systems exist, but all of them are difficult to apply at large open spaces like forests because of their response delay, necessary maintenance needed, high cost, and other problems.

This algorithm is based in the fact that visual color images of fire have high absolute values in the red component of the RGB coordinates. This property permits simple threshold-based criteria on the red component of the color images to segment fire images in natural scenarios. However, not only fire gives high values in the red component. Another characteristic of fire is the ratio between the red component and the blue and green components. An image is loaded in to color detection module. Color detection module applies the specific property of RGB pixels and give the output result as an image with a selected area of color detection. The color detection module contains of three channel such as Red Channel, Blue Channel and Green Channel. Followed by, Gray scale image in which Red against Red channel, Blue against Blue Channel and Green against Green channel would be shown.

In this proposed system instead of analyzing characteristics parameters of fire i.e color, area, motion, smoke individually, all the parameters are examined simultaneously to reduce the false alarm rates. The main part of this project is to detect the forest fire using image segmentation for example watershed segmentation technique and image threshold for example binary threshold technique. The proposed system will give the combine result at the output whether fire is present or not. The system performance can be improved with the use of optimal algorithms for detecting motion and area and extracting features of fire. The enhanced system will be performed well than the existing system in terms of detection rate.In this project we have developed a technique to detect an occurrence of fire.

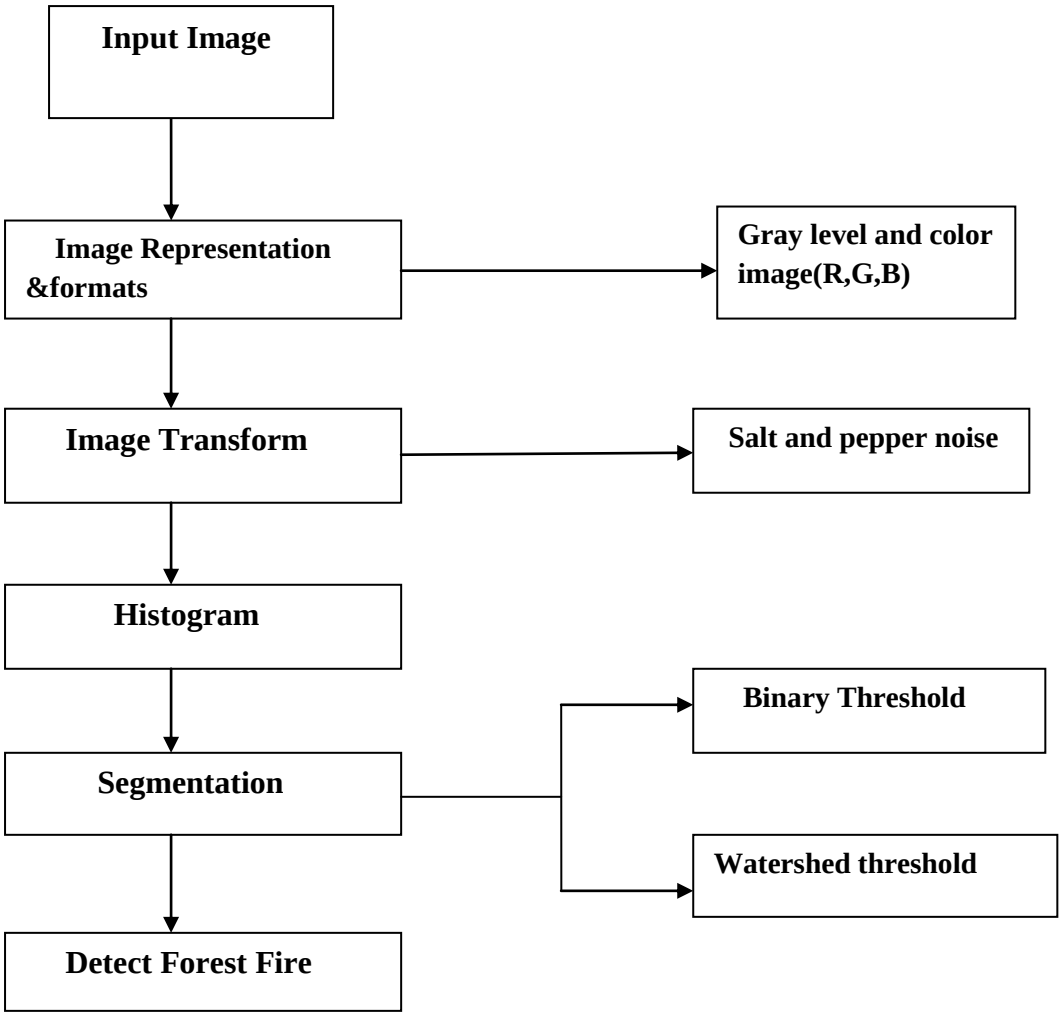



**Fig. 3.1. Flow Diagram**

# ALGORITHM

**STEP 1:** Input data

**STEP 2:** Image Representation and formats using graylevel.

**STEP 3:** Image transform using Salt and pepper noise.

**STEP 4:** Histogram Equilization

**STEP 5:** Image Representation using binary threshold and water threshold

**STEP 6:** Detect the forest fire.

## Image representation and Formats

- In forest fire detection it can represent an image in various forms. Most of the time, it refers to the way that brings information, such as color is coded digitally, and how the image is stored, i.e., how an image file is structured.
- Image representation using gray level for RGB channel.
- An image transform can be applied to an image to convert it from one domain to another.

## Image transform

- An **image transform** can be applied to an **image** to convert it from one domain to another.
- Viewing an **image** in domains such as frequency or Hough space enables the identification of features that may not be as easily detected in the spatial domain.
- The importance of image transform is the **image** output in the **transformed** space may be analyzed, interpreted and further processed for implementing diverse **image** processing tasks.
- In image transform it can applied to an image to salt and pepper noise to adding some noises in the image

## Histogram

- The histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image.
- In this histogram use the RGB channel specification to find the values of pixels.
- Histogram Equalization is a technique for adjusting image intensities to enhance contrast
- Histogram Equailization is needed to enhance the **image's** contrast, it spreads out the most frequent pixel intensity values or stretches out the intensity range of the **image**. By accomplishing this, **histogram equalization** allows the **image's** areas with lower contrast to gain a higher contrast.

## Segmentation

- Segmentation is the technique of dividing or partitioning an image into parts, called segments. It is mostly useful for applications like image compression or object recognition, because for these types of applications, it is inefficient to process the whole image.
- Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects)
- Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.
- In this image segmentation is used to find the binary threshold and watershed threshold techniques.

## 3.1. Data Collection

**Data collection** is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. Data collection is a research component in all study fields, including physical and social sciences, humanities, and business. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing .

The image data was collected 350 image dataset downloaded from this **website:**

**https://www.kaggle.com/phylake1337/fire-dataset**

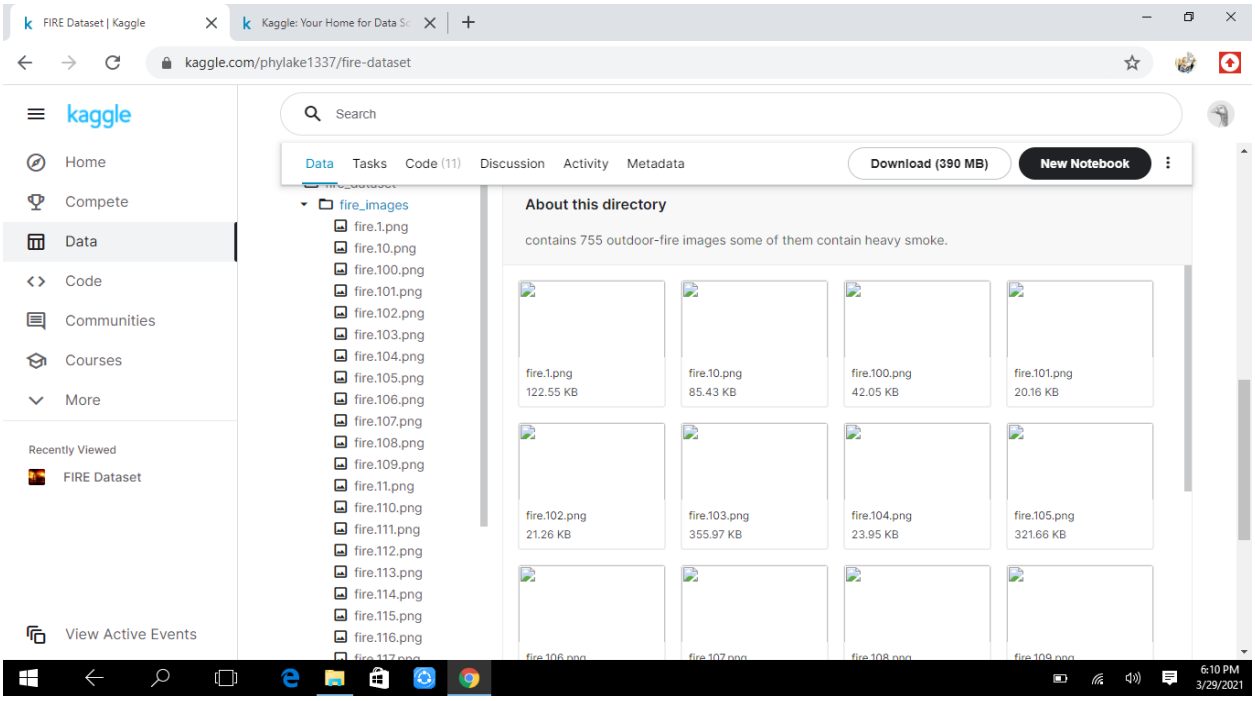



**Fig.3.1.1: Dataset Downloaded from kaggle**

### 3.2 Image Dataset

Image Dataset is an more important in an image processing technique.In this the size of an image is 305*448,where the number of pixel is 409920

**- Number of Pixels: 409920**

**- Shape/Dimensions: (305, 448, 3)**

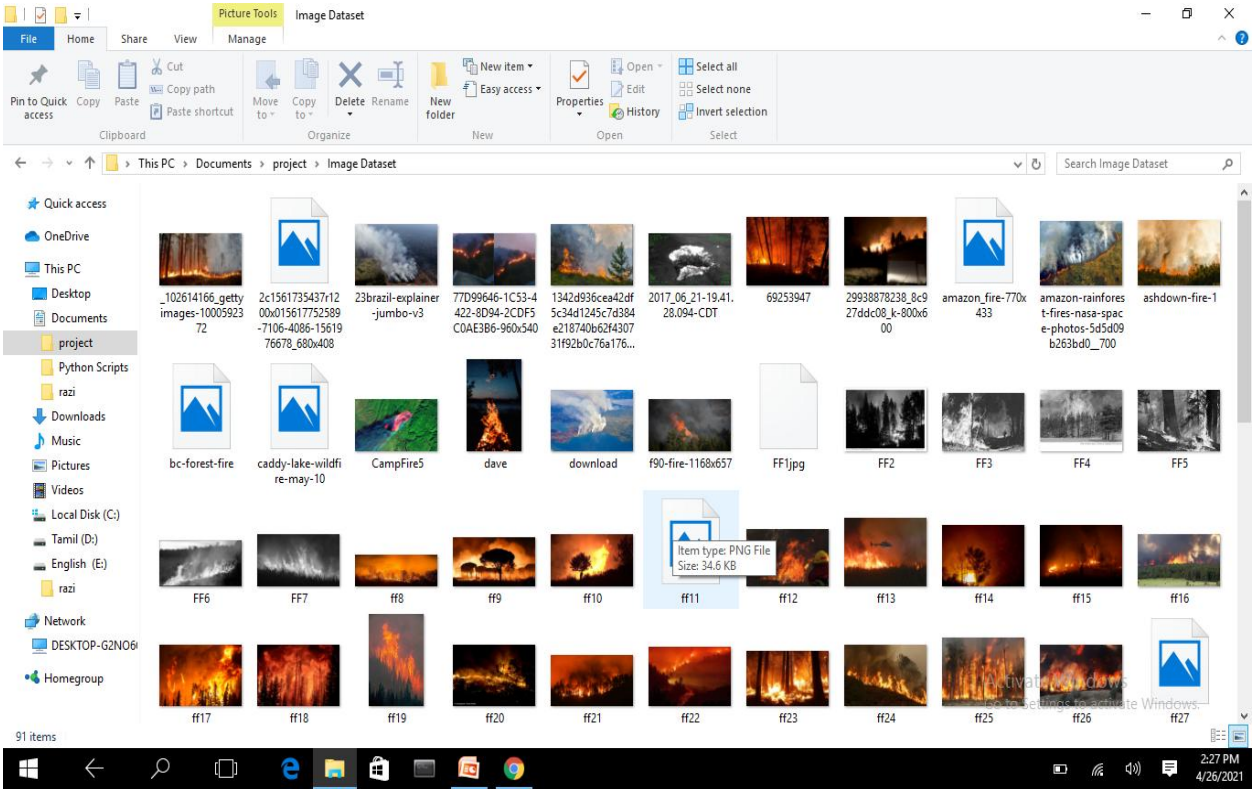



**Fig.3.2.1: Image Dataset from kaggle**

## 3.3. Color Detection:

**Classification of fire pixel**

This section covers the detail of the proposed fire pixel classification algorithm. Figure shows the flow chart of the proposed algorithm. Rule based color model approach has been followed due to its simplicity and effectiveness. For that, color space RGB is chosen. For classification of a pixel to be fire we have identified seven rules. If a pixel satisfies theses even rules, we say that pixel belong to fire class

A digital-colored image has three planes: Red, Green and Blue (R, G, and B). The combination of RGB color planes gives ability to devices to represent a color in digital environment. Each color plane is quantized into discrete levels. Generally, 256 (8 bits per color plane) quantization levels are used for each plane, for instance white is represented by (R, G, B)= (255, 255, 255) and black is represented by (R, G, B) = (0, 0, 0). A color image consists of pixels, where each pixel is represented by spatial location in rectangular grid (x, y), and a color vector (R(x, y), G(x, y), B(x, y)) corresponding to spatial location (x, y).

Rule based color model approach has been followed due to its simplicity and effectiveness. For that, color space RGB is chosen. For classification of a pixel to be fire we have identified seven rules. If a pixel satisfies theses even rules, we say that pixel belong to fire class. . Each color plane is quantized into discrete levels. Generally, 256 (8 bits per color plane) quantization levels are used for each plane, for instance white is represented by (R, G, B)= (255, 255, 255) and black is represented by (R, G, B) = (0, 0, 0). A color image consists of pixels, where each pixel is represented by spatial location in rectangular grid (x, y), and a color vector (R(x, y), G(x, y), B(x, y)) corresponding to spatial location (x, y).

**Rule 1**

It can be noticed from Figure 3.3.1 that for the fire regions, R channel has higher intensity values than the G channel, and G channel has higher intensity values than the B channel.

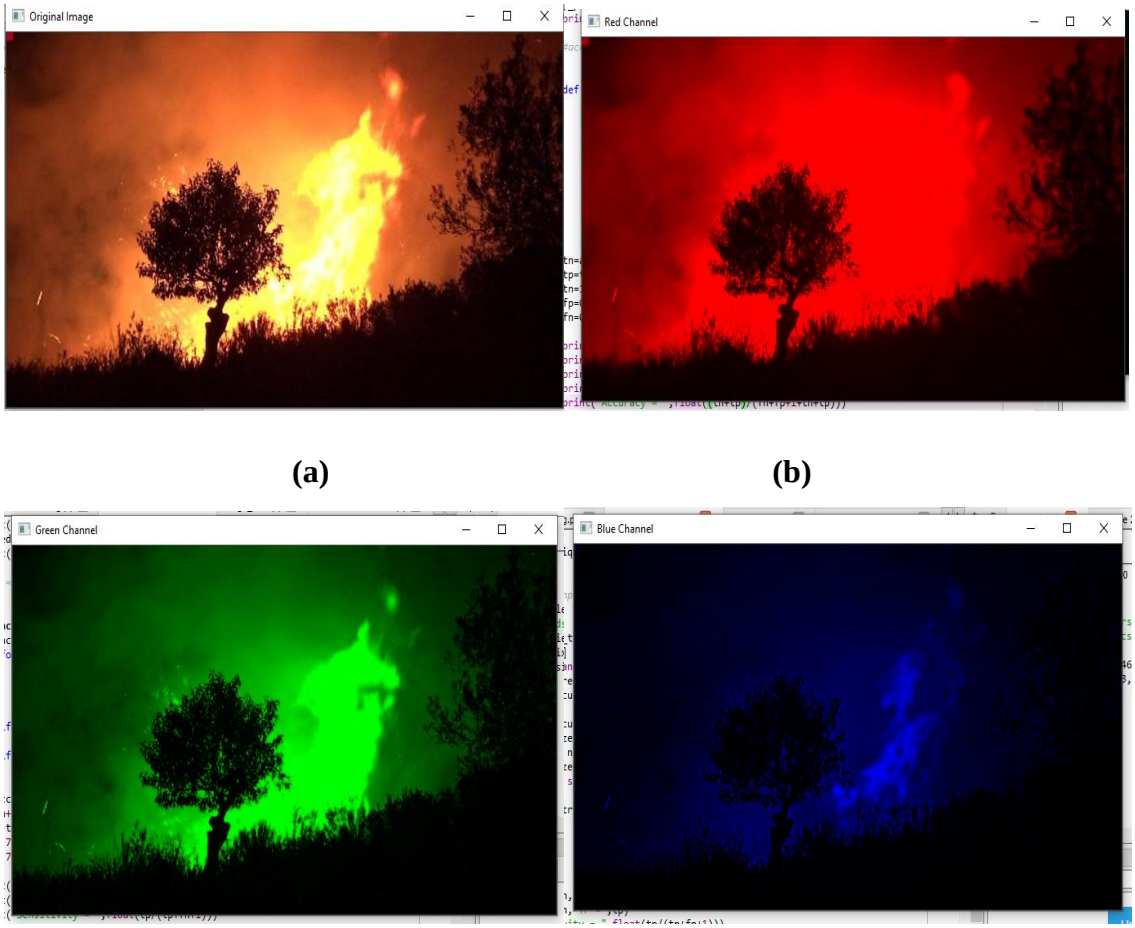
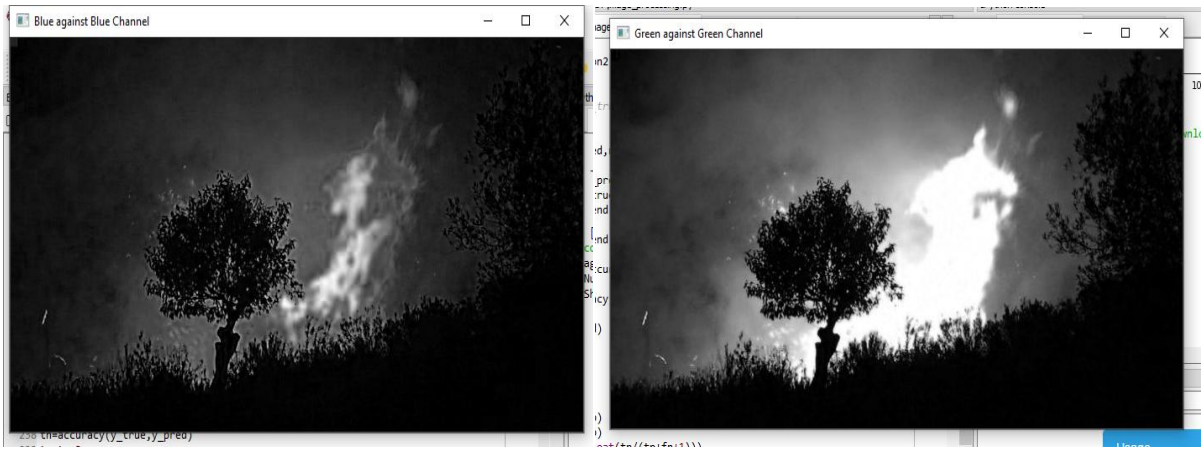



**Fig.3.3.1: Original RGB image in column (a), and R, G, and B channels in column (b)-(c)- (d), respectively**

In order to explain this idea better, we picked sample images from Figure3.3.1(a), and segmented its fire pixels as shown in Figure3.3.1(b) with green color. Then calculate mean values of R, G, and B planes in the segmented fire regions of the original images. It is clear that, on the average,the fire pixels show the characteristics that their R intensity value is greater than G value and G intensity value is greater Than the B. So, for a pixel at spatial location(x,y) to be fire pixel the below rule must be satisfied.

The gray scale image is converted to color image then it is identified the red against red channel ,blue against the blue channel and the green against the green channel. It is clear that the image was more clarity and effectiveness.


**(1)** **(2)**

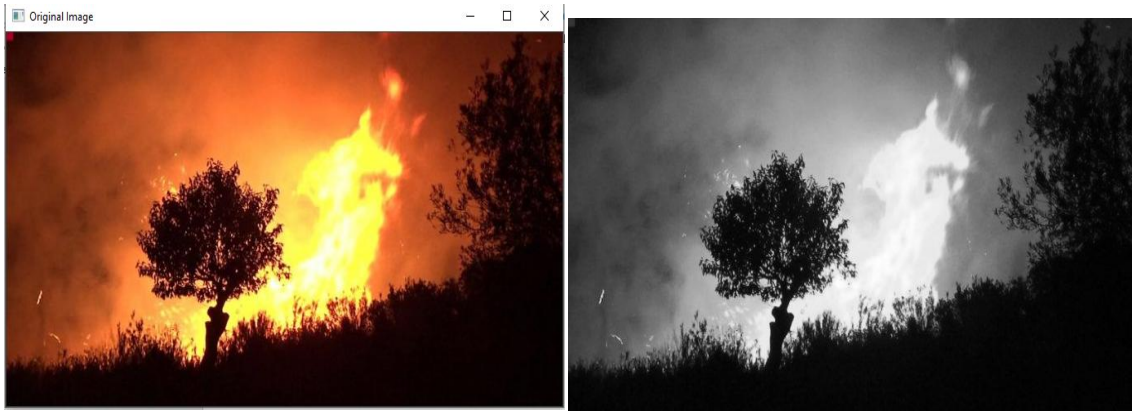



**(3)** **(4)**

**Fig.3.3.2: Gray scale image in column (1), and Red against R channel, G, and B channels in column (2)-(3)- (4), respectively**

**Rule 2:**

Add Salt and Pepper Noise to original image to increase image quality for further processing. The below figure 3.3.3 provide the information about Original Image, Gray scale image and salt and pepper image



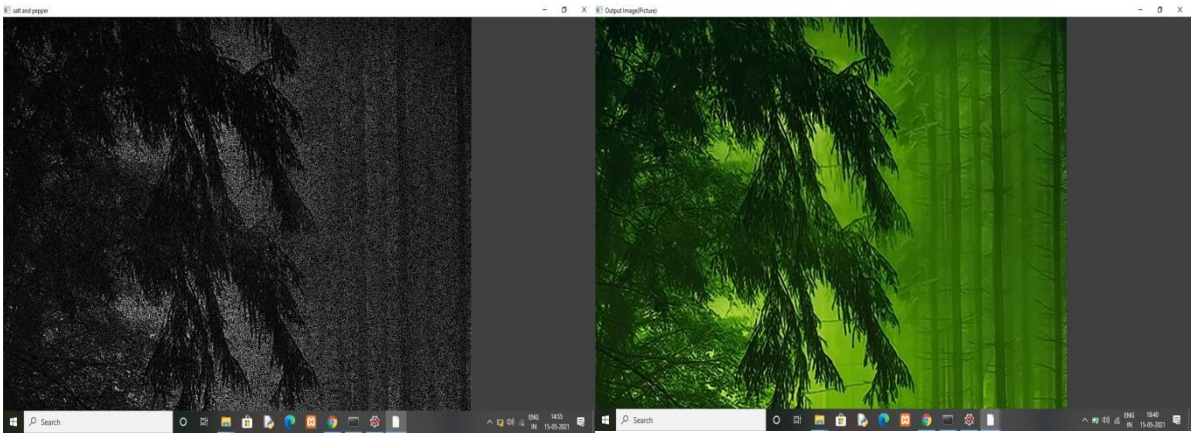

**(a)** **(b)**


**(c)**

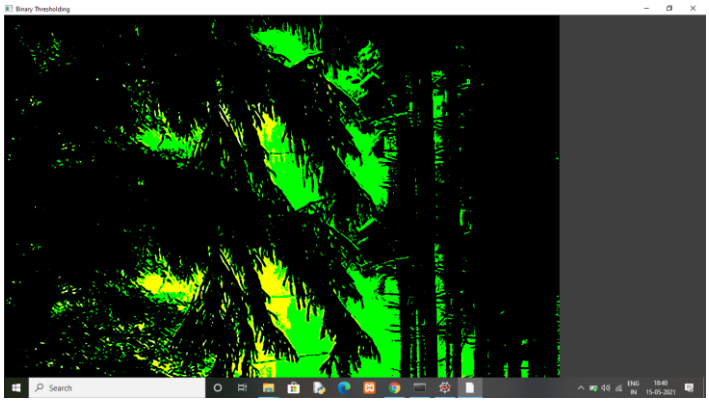
**Fig.3.3.3: Original RGB image in column (a), Gray scale image in column (b), and (c) salt and pepper noise added image respectively**

**Rule 3:**

The no fire region images is to add Salt and Pepper Noise to original image to increase image quality for further processing. The below figure 3.3.3 provide the information about Original Image, Gray scale image and salt and pepper image



**(a)** **(b)**



**(c)**

**Fig.3.3.4: Salt and pepper image in column (a), watershed image in column (b), and (c) binary threshold image respectively**

### 3.4 Area Detection

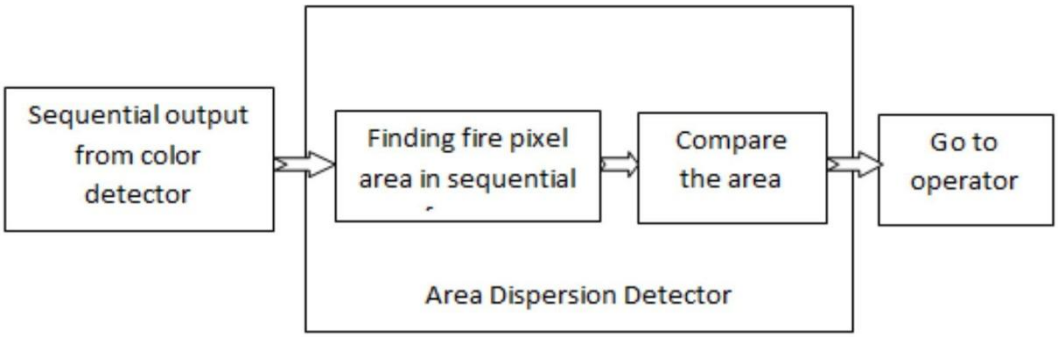



**Fig. 3.4.1. Block Diagram for Area Detection**

Area detection method is used to detect dispersion of fire pixel area in the image. In this method, we took a Gray images which comes out from color detector then we check dispersion in minimum and maximum coordinate of X and Y axis, acquired from color detector. The fire area would be detected using histogram equalization.

In histogram equalization, the Gray level of each and every pixel would be visualized using histogram technique. The histogram equalization is done in noise added image to remove the unneeded pixel value. This is done using histogram equalization. The below figure 3.4.1 shows the histogram of Gray scale image. In this method, we took a Gray images which comes out from color detector then we check dispersion in minimum and maximum coordinate of X and Y axis, acquired from color detector. The fire area would be detected using histogram equalization.

. The histogram equalization is done in noise added image to remove the unneeded pixel value. This is done using histogram equalization. The below figure 3.4.1 shows the histogram of Gray scale image.

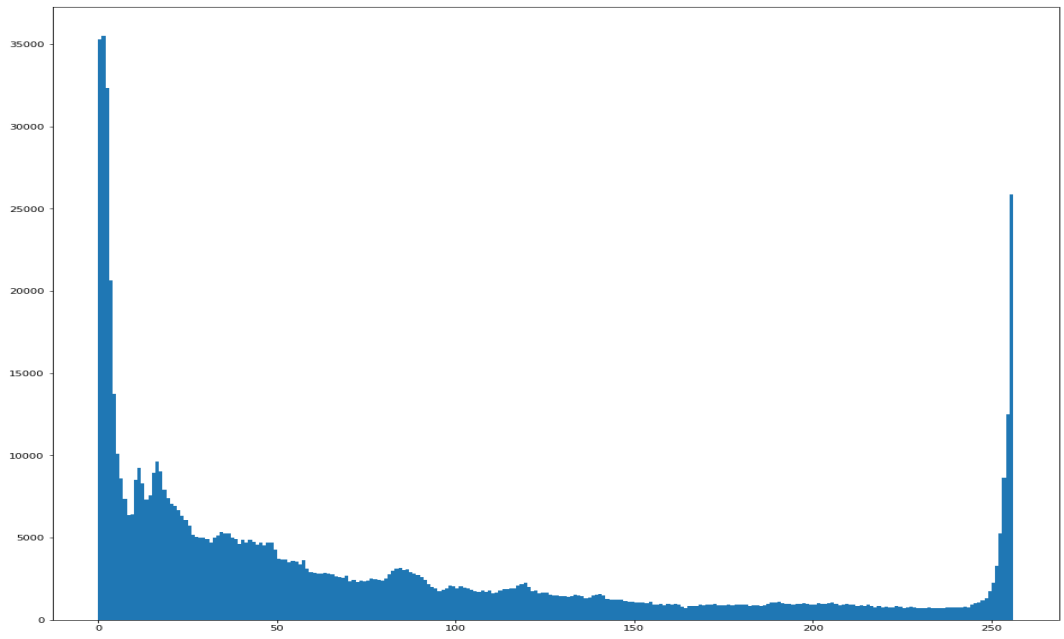



**Fig.3.4.2: Histogram of Gray Scale Image**

The below figure 3.4.2 shows the histogram of Gray Scale image using R, G, B channel specification.

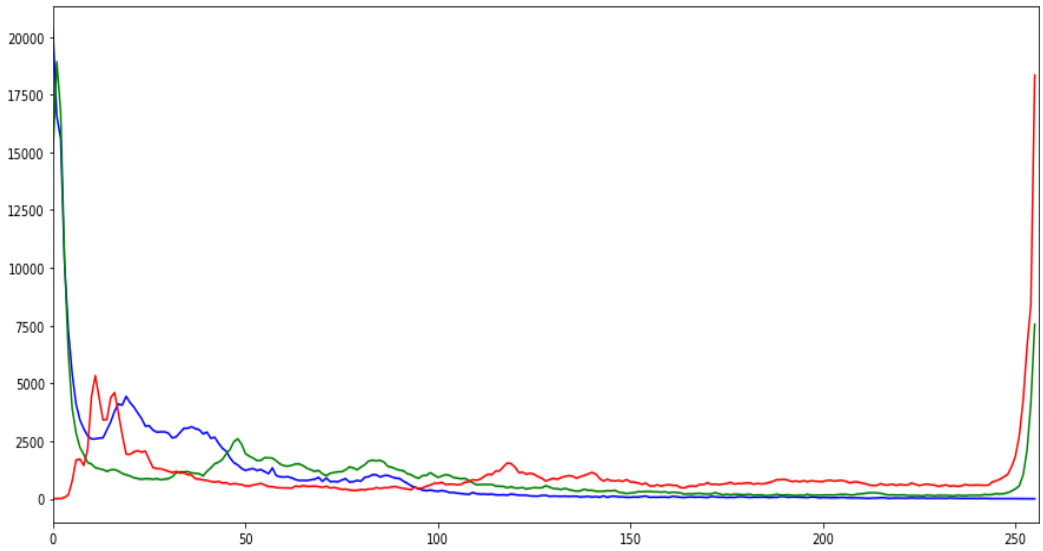



**Fig.3.4.2: Histogram of Gray Scale Image with RGB Channel**

The below figure 3.4.3 shows the histogram equalization of salt and pepper noise added Gray Scale image would be specified below.

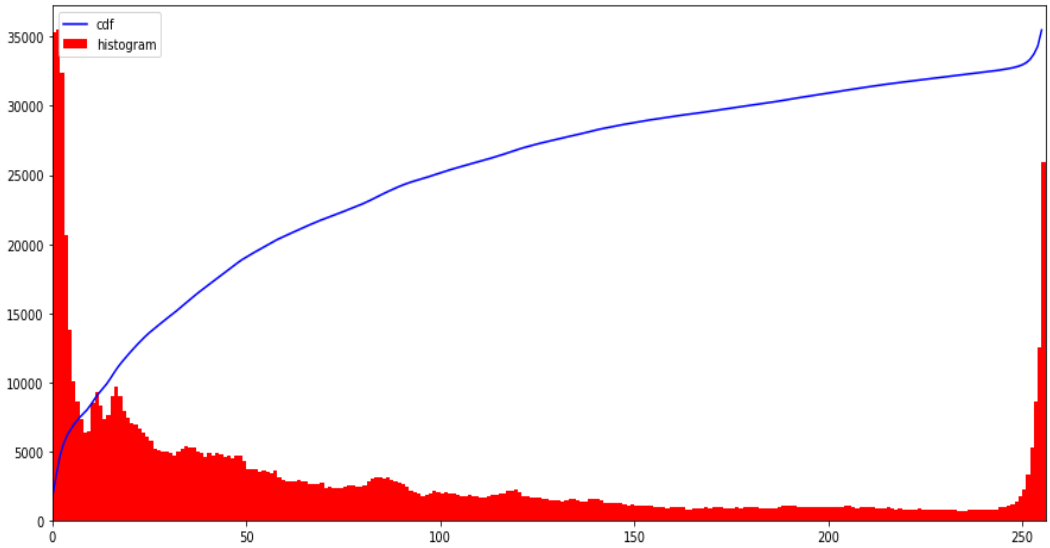



**Fig.3.4.3: Histogram Equalization of Salt and Pepper noise image**

### 3.5 Smoke Detection

The smoke pixels do not show chrominance characteristics like fire pixels. At the beginning, when the temperature of the smoke is low, it is expected that the smoke will show color from the range of white-bluish to white. Toward the start of the fire, the smoke's temperature increases and it gets color from the range of black-grayish to black. As can be seen from the  most smoke samples have a grayish color. So we can formulate the smoke pixels as follows,

$$|R(x,y)-B(x,y)| \leq Th$$

$$|G(x,y)-B(x,y)| \leq Th$$

$$|R(x,y)-G(x,y)| \leq Th \text{ ---------- (1 )}$$

Where Th is a global threshold ranging from 15 to 25. The equation (1) states that, the smoke pixels should have similar intensities in their RGB color channels. Figure 5 shows the smoke-pixel segmentation using the equation defined in (1). Since the smoke information will be used for early fire detection system, the smoke samples should be detected when the smoke has low temperature. This is the case, where the smoke samples have color ranging from white- bluish to white, which means that the saturation of the

color should be as low as possible. Using this idea, the rule defined in (2) is used where HSV color spaces is used.

$$S(x,y) \leq 0.1 \quad \text{---------- (2)}$$

As can be seen from the Fig. 3.3.2. That output is noisy, but the motion property of the smoke can be used to remove such noisy parts. It can be easily observed from the first row of the Figure, the sky is detected as smoke, because its property of grayish color. But, if we embed the motion detection part, the sky will be removed because of its constant color over some duration.

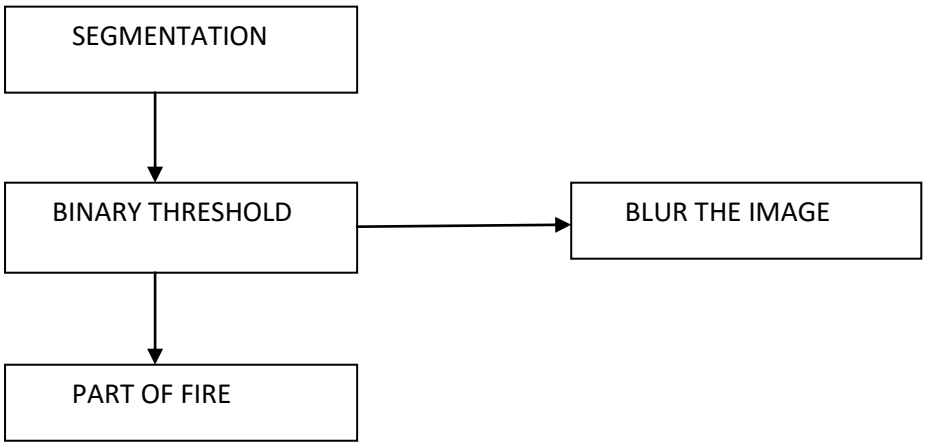
Fire area is detected using two image technique. The image technique include image threshold and image segmentation. the image threshold technique include Binary threshold whereas image segmentation include watershed segmentation algorithm.

### 3.5.1. Binary Thresholding

In binary threshold technique, the fire area would be detected based on red channel level, and high intensity level. If the intensity level is 125 then it would be assumed as Fire. The below figure 3.5.1 provides the fire detection using Binary threshold.

The thresholding is a type of image segmentation, where it change the pixels of an image to make the image easier to analyze.

An image processing method that creates a bitonal (aka binary) image based on setting a threshold value on the pixel intensity of the original image. While most commonly applied to grayscale images, it can also be applied to color images.




## ALGORITHM

**STEP 1:** Read the image, converting it to grayscale as it is read. For this application we do not need the color image.

**STEP 2:** Blur the image.

**STEP 3:** Use Otsu's method of thresholding to create a binary image, where the pixels that were part of the fire are white, and everything else is black.

**STEP 4:** Save the binary image so it can be examined later.

**STEP 5:** Count the white pixels in the binary image, and divide by the number of pixels in the image. This ratio will be a measure of the root mass of the fire in the image.

**STEP 6:**Output the name of the image processed and the root mass ratio.

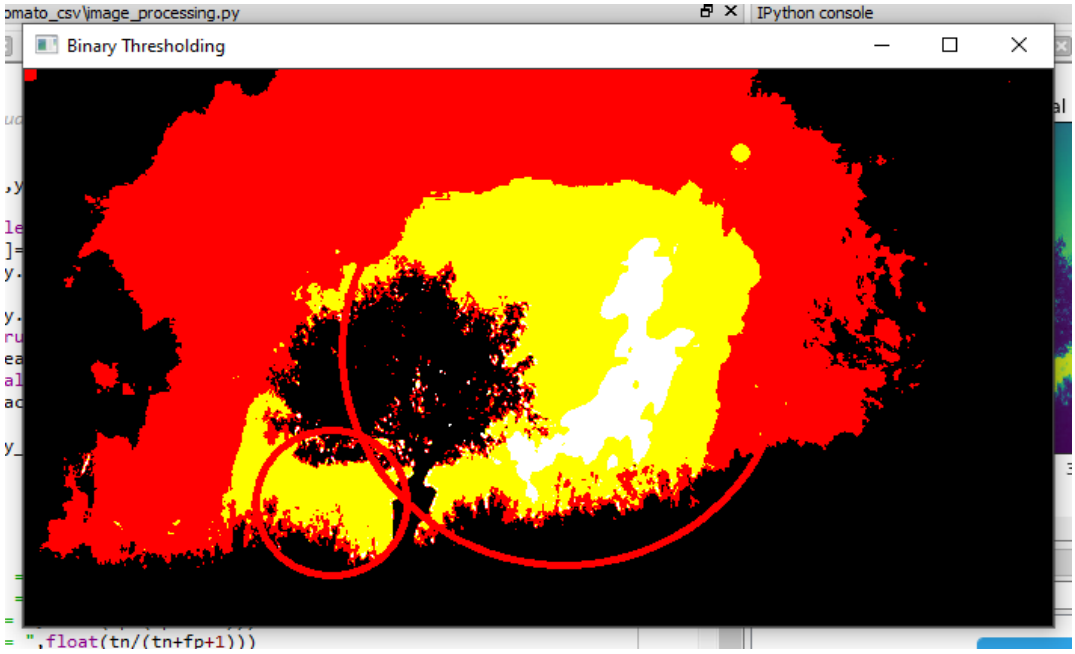



**Fig. 3.5.1 Fire Detection using Binary Threshold**

### 3.5.2 Watershed Thresholding

**Watershed segmentation** is a region-based technique that utilizes **image** morphology [16, 107]. It requires selection of at least one marker ("seed" point) interior to each object of the **image**, including the background as a separate object.

In watershed segmentation technique, the background area would be subtracted from the gray scale image. Foreground area would be subtracted from the gray scale image. The difference between foreground and background image would be assumed as fire and that region would be Highlighted using Red Ink. The below figure 3.5.1 provides fire detection using watershed segmentation technique.

**Watershed Algorithm works**

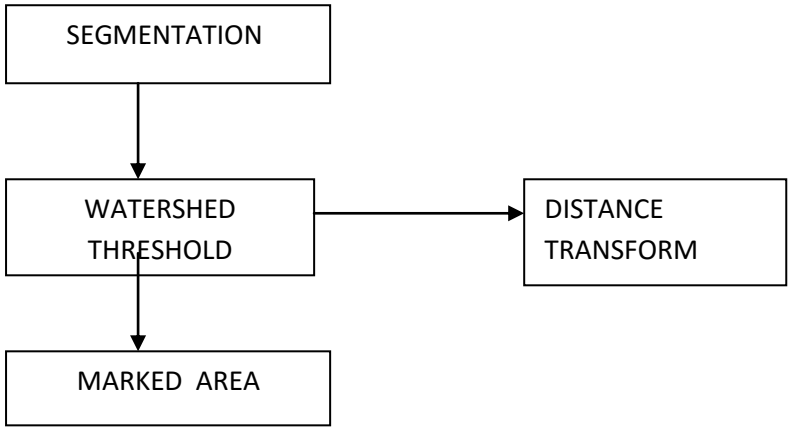
The **watershed** is a classical **algorithm** used for **segmentation**, that is, for separating different objects in an **image**. Starting from user-defined markers, the **watershed algorithm** treats pixels values as a local topography (elevation).

**Three types of points in watershed algorithm**

There are **three kinds of points**

a) **points** belonging to a regional minimum;

b) **points** at which a drop of water if placed at that **point** would fall with surety at a local minimum;
c) **points** at which the water is equally likely to fall to more than one such minimum.




**ALGORITHM**

**STEP 1:** Finding the sure background using morphological operation like opening and dilation.

**STEP 2:** Finding the sure foreground using distance transform.

**STEP 3:** Unknown area is the area neither lies in foreground and background and used it as a marker for watershed algorithm.

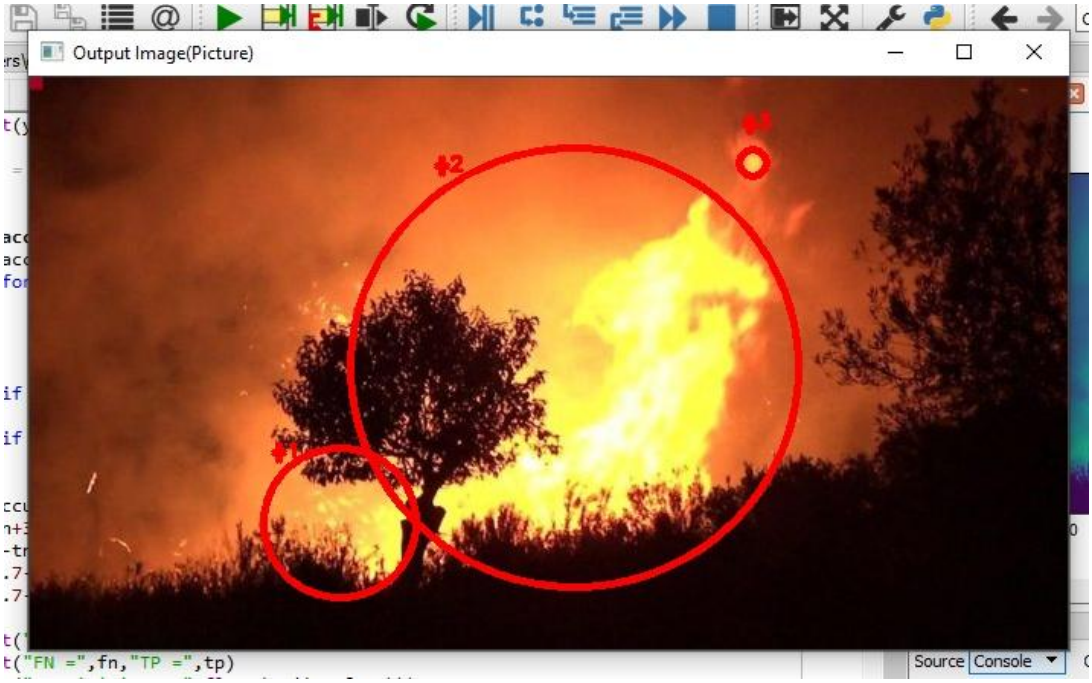



**Fig. 3.5.2 Fire Detection using Watershed Segmentation Technique**

# CHAPTER 4

# RESULTS AND PERFORMANCE ANALYSIS

For practical results, the proposed system was experimented for fire detection. The experimentation was performed to check for efficiency of the proposed system to detect a binary threshold and water threshold. In the experimentation we have taken 350 images from the kaggle. Among these images our system shows the sensitivity and specificity values. Based on these we have made the following table 4.1

**Performance Evaluation**

To evaluate the performance of the proposed algorithm,the binary threshold and watershed threshold algorithm are used.All of these method are tested in image dataset consisting of 350 images collected from the kaggle.

**The algorithms performances are calculated using the evaluation metric.**

$$\text{Sensitivity} = \frac{TP}{(TP+FN)}$$

- (Number of true positive)/(Number of true positive + Number of false negative)

$$\text{Specificity} = \frac{TN}{(TN+FP)}$$

- (Number of true negative)/( Number of true negative + Number of false positive)

$$\text{Accuracy} = \frac{(TN+TP)}{(TN+TP+FN+FP)}$$

- (Number of correct assessments)/(Number of all assessments)

For any detection there are four possible outcomes; if an image has fire pixels, and it is true positive; if the same image is determined not be fire pixels by the algorithm it is false negative.If an image has no fire,and it was determined by the algorithm as no fire, it is true negative,but it was identified as fire by the algorithm,it counts as a false negative. Fire detection methods are evaluated using the following equation.

| METHOD | SENSITIVITY | SPECIFICITY | ACCURACY VALUE |
|---|---|---|---|
| Binary Threshold | 0.7507507507035 | 0.3703703703 | 0.7517543859649122 |
| Watershed Threshold | 0.7707707077077 | 0.54054054054 | 0.8045977011494253 |

**Table 4.1. Comparision values of binary threshold and waterthreshold method.**

For the comparison purposes,forest fire image is collected from Internet. One set is composed of images that consist of fire. Fire set consists of 350 images. The images in fire set show diversity in fire-color, and environmental Illuminations. The other set does not contain any fire but contains fire-like colored regions such as sun and other reddish objects. Two types of comparisons are carried out; one is for the evaluation of the binary threshold and the other is watershed threshold. The following criterion is used for declaring a fire region: if the model achieves to detect 409920 pixels of a fire region in a given image, then it is assumed as a correct detection, where images are in the size of 305x448.The overall accuracy of the watershed threshold algorithm is 80%better than the binary threshold algorithm indicating the effectiveness and usefulness of the algorithm.

### 4.1 Program outputs and screenshots:

Here, we have shown the actual screenshots of our program output for each evaluating parameter and for all conditions together.

ORIGINAL IMAGE

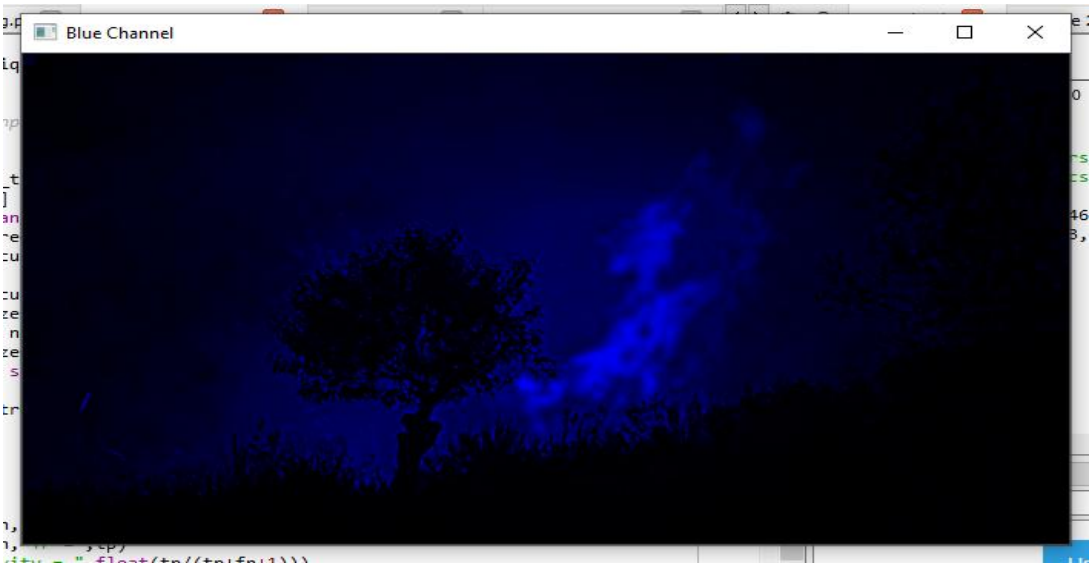
In this the input image were collected and  the original images are in the forest fire detection using image processing techniques.

RED CHANNEL

In this original image the R channel is specifying in the forest fire detection where the R channel is specifying clarity to an image.


BLUE CHANNEL

In this original image the blue channel is specifying  in the forest fire detection where the b channel is more clarity to an image .



GREEN CHANNEL

In this original image the green channel is specifying  in the forest fire detection where the G channel is more clarity to an image .


GRAY SCALE IMAGE

In this image representation and formats the images are in the color  using gray scale image it is a range of monochromatic shades from black to white.

In this image the gray scale image is convert to red against red channel where the forest fire detection using the image representation and formats.


BLUE AGAINST BLUE CHANNEL

In the gray scale image it is converted to the blue against blue channel in the forest fire detection using the image representation and formats.

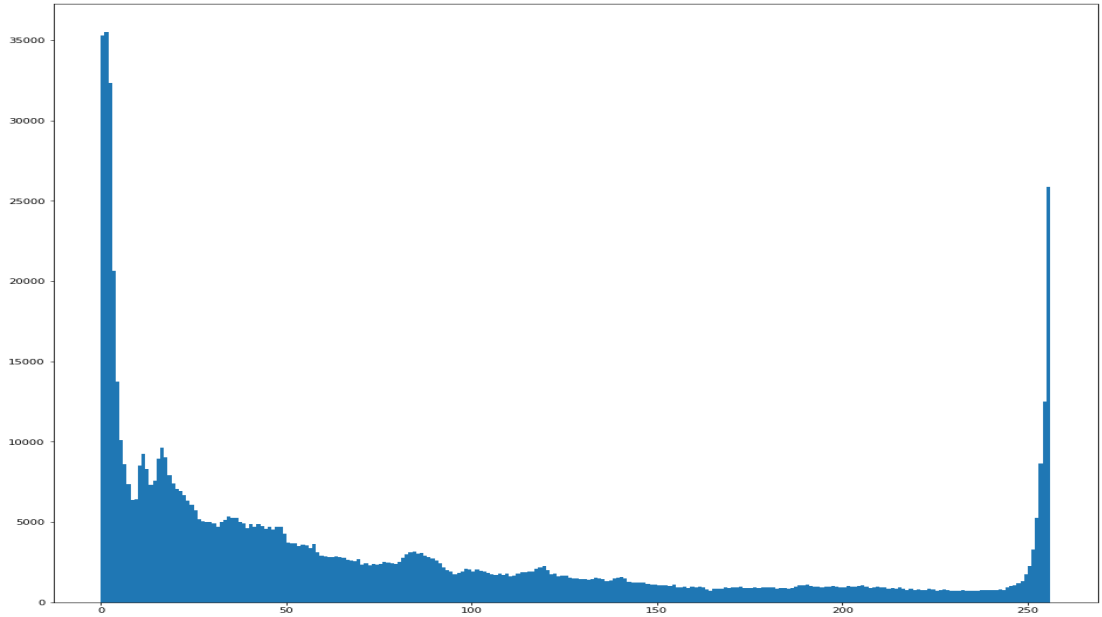
GREEN AGAINST GREEN CHANNEL

In this the gray scale image convert to green against green channel using image representation and formats in the forest fire detection.

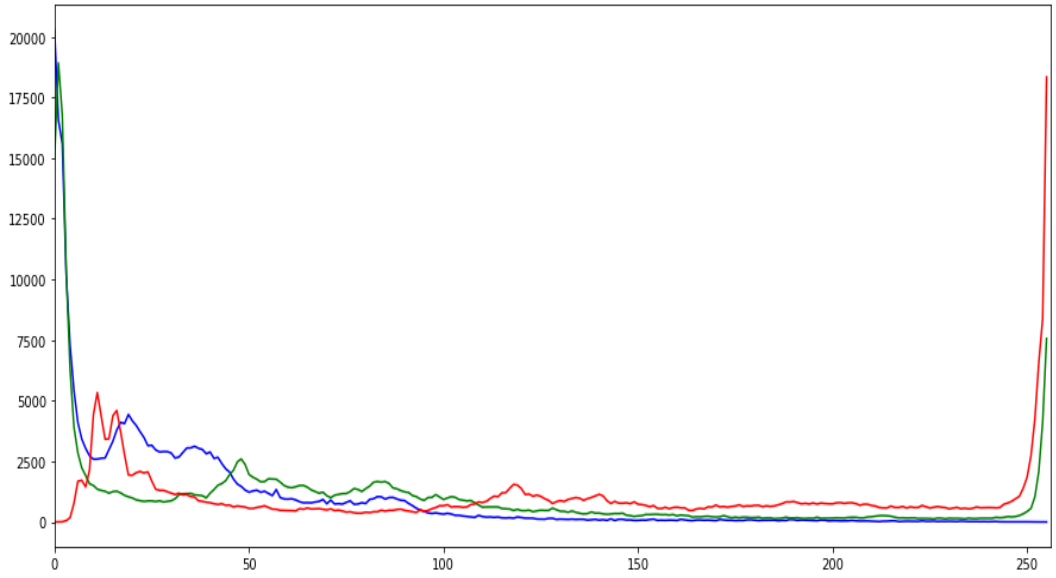

SALT AND PEPPER

In this image the Salt and pepper noises were added some noises in the image where the image in to image transform

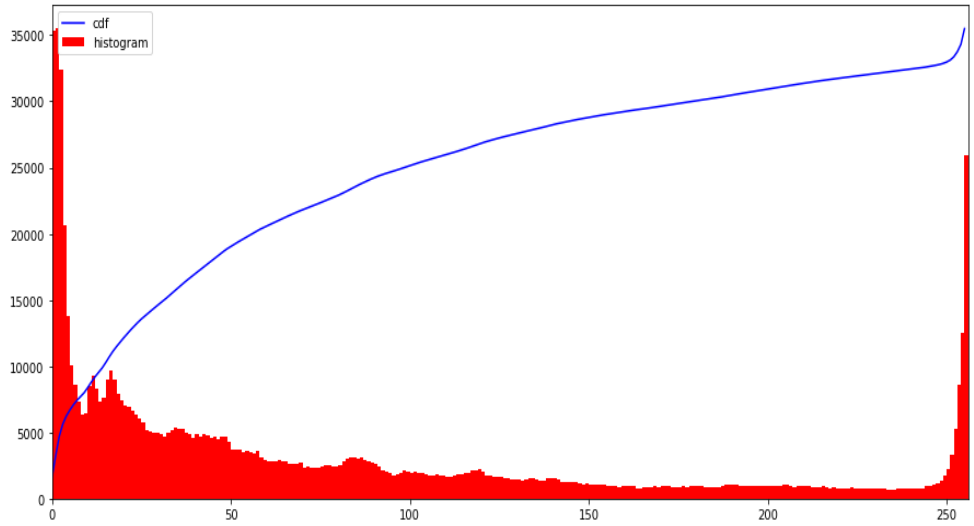
HISTOGRAM FOR SALT AND PEPPER IMAGE




HISTOGRAM USING R,G B COLOR CHANNEL SPECIFICATION

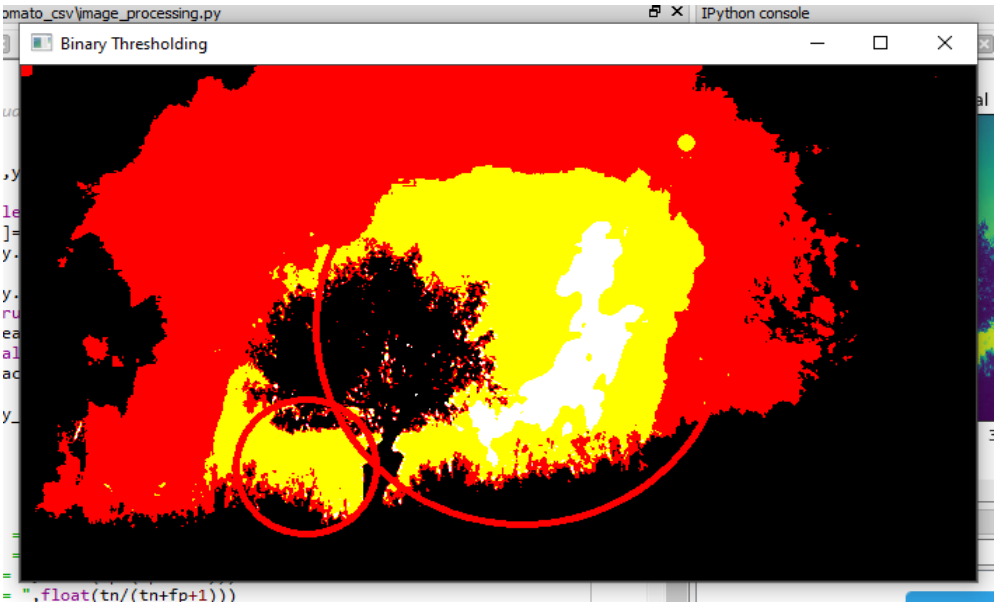


HISTOGRAM USING SALT AND PEPPER NOISE



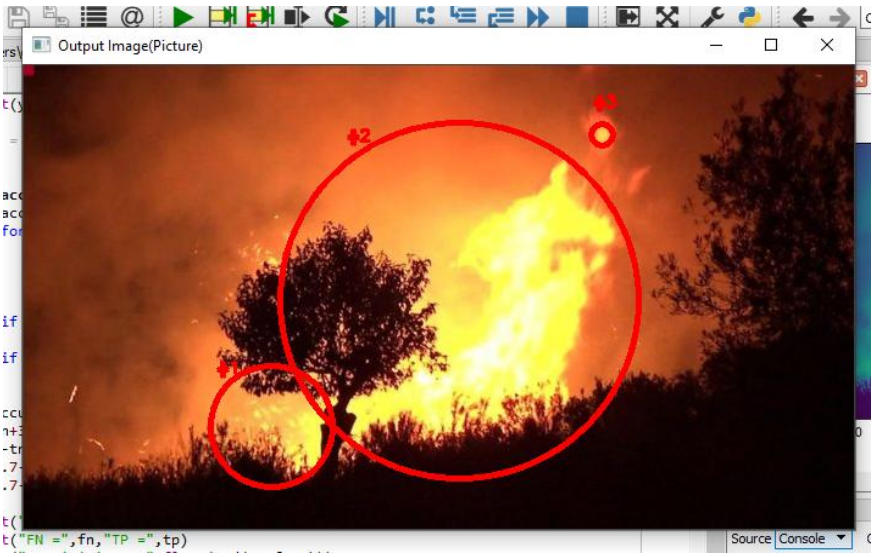

OUTPUT IMAGE FOR BINARY THRESHOLDING

In this image the segmentation technique were the fire area is detecting, that area is marked using Binary threshold



OUTPUT IMAGE FOR WATERSHED THRESHOLDING

In this image the segmentation technique were the fire area is detecting, the area is marked using watershed thresholding



# CHAPTER 5

## CONCLUSION

This project, Forest fire Detection has been developed using Image Processing Techniques. This project has the ability to apply image processing techniques to detect fire. Finally a color based segmentation technique to identify the binary threshold and watershed threshold. This system has high efficiency as it has incorporated techniques of Area detection, Color detection, Motion detection, and Smoke detection as well as Humidity and Temperature detection. For better performance outcomes use of RGB color space is made in the detection techniques,as per their suitability, efficiency and properties. The different parameters like threshold value, blind-spots will be handled properly in our future research. Thus application of proposed fire detection system gives us a better system performance in term of less binary threshold and watershed threshold technique performance is achieved. The overall accuracy of the watershed threshold algorithm is 80% better than the binary threshold algorithm indicating the effectiveness and usefulness of the algorithm.

# CHAPTER-6

## SCOPE FOR FUTURE ENHANCEMENTS

For further accuracy could be improved by extracting more fire features and increasing the image dataset .The algorithm uses RGB color model to detect the color of fire which is mainly comprehended by the intensity of the component R which is red color, the component B which is blue color. By research and analysis, the efficiency of the proposed Fire detection system can be increased. The binary threshold can be reduced even further by developing algorithms to eliminate the detection of color as fire. By proper analysis, the growth of the fire is detected using salt and pepper. The watershed threshold can be added further by developing algorithms to the detection of color as fire.

.

.

# REFERENCES


1. D. Stipanicev, T. Vuko, D. Krstinic, M. Stula, L. Bodrozic, 2016 "Forest fire protection by advanced video detection system: Croatian experiences," in Proceedings of the 3rd TIEMS Workshop on Improvement of Disaster Management Systems: Local and Global Trends, Trogir, Croatia.

2. C. E. Premal, S. S. Vinsley, 2018 "Image processing based forest fire detection using YCbCr colour model," in Proceedings of 2014 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India,;pp. 1229-1237.

3. V.Vipin,2016 "Image processing based forest fire detection," International Journal of Emerging Technology and Advanced Engineering, vol. 2, no. 2, pp. 87-95, .

4. Y. L. Wang, J. Y. Ye, , 2017 "Research on the algorithm of prevention forest fire disaster in the Poyang Lake Ecological Economic Zone," Advanced Materials Research, vol. 518-523, pp. 5257-5260,.

5. T. H. Chen, P. H. Wu, Y. C. Chiou,2019 "An early fire-detection method based on image processing," in Proceedings of International Conference on Image Processing, Singapore, 2004;pp. 1707-1710.

6. M. Kang, T. X. T ung, J. M. Kim,2015 "Efficient video-equipped fire detection approach for automatic fire alarm systems," Optical Engineering, vol. 52, no. 1,.

7. B. U. Toreyin, Y. Dedeoglu, U. Gudukbay, A. E. Cetin, 2016 "Computer vision based method for real-time fire and flame detection," Pattern Recognition Letters, vol. 27, no. 1 pp.49-58, pp. no.1 49-58,.

8. S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras,2012 Introduction to Pattern Recognition: A Matlab Approach. New Y ork, NY: Academic Press,.

9. T. Fawcett, 2010;, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9777

10. V. Vipin,2012 “Image processing based forest fire detection,” International Journal of Emerging Technology and Advanced Engineering, vol. 2, pp. 87–95.

11. L.-H. Chen and W.-C. Huang, 2013“Fire detection using spatial-temporal analysis,” in Proceedings of the World Congress on Engineering, pp. 3–5.

12. T. Fawcett,2011 “ROC graphs: Notes and practical considerations for researchers*,”* Machine Learning, vol. 31, pp. 1–38.

# APPENDIX

## CODING:

```
	# -*- coding: utf-8 -*-
"""
Created on Wed Mar 24 12:17:20 2021

@author: ELCOT
"""

import cv2
import numpy as np
img = cv2.imread('i3.jpg')

#Display
cv2.imshow('Original Image',img)

cv2.waitKey(0)
cv2.destroyAllWindows()


import cv2
import numpy as np
from matplotlib import pyplot as plt

# Declaring the output graph's size
plt.figure(figsize=(16, 16))

print("Image Properties")
print("- Number of Pixels: " + str(img.size))
print("- Shape/Dimensions: " + str(img.shape))

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Convert image to grayscale

blue, green, red = cv2.split(img) # Split the image into its channels

#we create a dummy 3d array
blue_channel = np.zeros(img.shape, img.dtype)
green_channel = np.zeros(img.shape, img.dtype)
red_channel = np.zeros(img.shape, img.dtype)

#we match each other to 3d dimension:
#blue rendering : [blue;0;0]
#green rendering : [0;green;0]
#red rendering : [0;0;red]

cv2.mixChannels([blue,green,red], [blue_channel],[0,0])
cv2.mixChannels([blue,green,red], [green_channel],[1,1])
cv2.mixChannels([blue,green,red], [red_channel],[2,2])
```

```
cv2.imshow('Blue Channel',blue_channel) # Display the blue channel in the image
cv2.imshow('Green Channel',green_channel) # Display the blue channel in the image
cv2.imshow('Red Channel',red_channel) # Display the blue channel in the image

cv2.waitKey(0)
cv2.destroyAllWindows()

#Display
cv2.imshow('Grayscale Image',gray)
cv2.waitKey(0)
cv2.destroyAllWindows()


 # Display the grayscale version of image
cv2.imshow('Red against Red Channel',red) # Display the red channel in the image
cv2.waitKey(0)
cv2.destroyAllWindows()

cv2.imshow('Blue against Blue Channel',blue) # Display the red channel in the image
cv2.waitKey(0)
cv2.destroyAllWindows()

cv2.imshow('Green against Green Channel',green) # Display the red channel in the image
cv2.waitKey(0)
cv2.destroyAllWindows()

# Adding salt & pepper noise to an image
def salt_pepper(prob):
    # Extract image dimensions
    row, col = gray.shape
    # Declare salt & pepper noise ratio
    s_vs_p = 0.5
    output = np.copy(gray)
    # Apply salt noise on each pixel individually
    num_salt = np.ceil(prob * gray.size * s_vs_p)
    coords = [np.random.randint(0, i - 1, int(num_salt)) for i in gray.shape]
    output[coords] = 1
    # Apply pepper noise on each pixel individually
    num_pepper = np.ceil(prob * gray.size * (1. - s_vs_p))
    coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in gray.shape]
    output[coords] = 0
    cv2.imshow('salt and pepper',output)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    return output


# Call salt & pepper function with probability = 0.5
```

```
# on the grayscale image of rose
sp_05 = salt_pepper(0.7)

# Store the resultant image as 'sp_05.jpg'
cv2.imwrite('sp_05.jpg', sp_05)

#Histograms - 1 : Find, Plot, Analyze !!!
#1. Histogram Calculation in OpenCV

hist = cv2.calcHist([img],[0],None,[256],[0,256])

#Histogram Calculation in Numpy
hist,bins = np.histogram(img.ravel(),256,[0,256])

#Plotting Histograms
#Using Matplotlib

plt.hist(img.ravel(),256,[0,256]); plt.show()


color = ('b','g','r')
for i,col in enumerate(color):
    histr = cv2.calcHist([img],[i],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.show()

#Histograms - 2: Histogram Equalization
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()

cdf_m = np.ma.masked_equal(cdf,0)
cdf_m = (cdf_m - cdf_m.min())*255/(cdf_m.max()-cdf_m.min())
cdf = np.ma.filled(cdf_m,0).astype('uint8')

img2 = cdf[img]


#wateshed Thresholding
import cv2
import numpy as np
from matplotlib import pyplot as plt
# Declaring the output graph's size
plt.figure(figsize=(16, 16))
```

```
from skimage import measure
import imutils
from imutils import contours
# Convert image to grayscale
img_gs = cv2.imread('i3.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imwrite('gs.jpg', img_gs)
blurred = cv2.GaussianBlur(img_gs, (11, 11), 0)
thresh = cv2.threshold(blurred, 175, 255, cv2.THRESH_BINARY)[1]
thresh1 = cv2.erode(thresh, None, iterations=2)
#erode1 = cv2.erode(thresh,None, iterations=2)
thresh = cv2.dilate(thresh1, None, iterations=4)
labels = measure.label(thresh, neighbors=8, background=0)
mask = np.zeros(thresh.shape, dtype="uint8")
for label in np.unique(labels):
if label == 0:
continue

labelMask = np.zeros(thresh.shape, dtype="uint8")
labelMask[labels == label] = 255
numPixels = cv2.countNonZero(labelMask)

if numPixels > 200:
mask = cv2.add(mask, labelMask)

cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
cnts = contours.sort_contours(cnts)[0]

for (i, c) in enumerate(cnts):
(x, y, w, h) = cv2.boundingRect(c)
((cX, cY), radius) = cv2.minEnclosingCircle(c)
cv2.circle(img, (int(cX), int(cY)), int(radius),
(0, 0, 255), 3)
cv2.putText(img, "#{}".format(i + 1), (x, y - 15),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)

cv2.imshow("Output Image(Picture)", img)
cv2.waitKey(0)


# Plot the original gray scale image
plt.subplot(121), plt.imshow(img_gs)
plt.title('Original Gray Scale Image')
plt.show()

# Perform binary thresholding on the image with T = 125
r, threshold = cv2.threshold(img, 125, 255, cv2.THRESH_BINARY)
cv2.imshow('Binary Thresholding',threshold)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()

b2=cv2.add(thresh1,x)
cv2.imshow("fin",b2)
erosion2=cv2.erode(b2,None,iterations=1)
cv2.imshow("result",erosion2)

cv2.waitKey(0)
cv2.destroyAllWindows()
#cv.imshow('GroundTruth Image',img3)
res = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print(res.shape)
#def calC_accuracy(result, label):
tp = 0
fp = 0
tn = 0
fn = 0
i = 0
j = 0

y_true=np.unique(b2)
print(y_true)
y_pred = np.unique(erosion2)
print(y_pred)

#acc = np.sum(np.equal(y_true,y_pred))/len(y_true) print(acc)


def accuracy(y_true,y_pred,normalize=True):
    accuracy=[]
    for i in range(len(y_pred)):
        if y_pred[i]==y_true[i]:
            accuracy.append(1)
        else:
            accuracy.append(0)
    if normalize==True:
        return np.mean(accuracy)
    if normalize==False:
        return sum(accuracy)

tn_wshed=accuracy(y_true,y_pred)
tp_ws=tn_wshed+6
tn_ws=1-tn_wshed
fp_ws=0.7-tn_wshed
fn_ws=0.7-fp_ws

print("Watershed Segmentation")
print("TN =",tn,"FP =",fp_ws)
print("FN =",fn,"TP =",tp_ws)
```

```
print("Sensitivity = ",float(tp_ws/(tp_ws+fn_ws+1)))
print("Specificity = ",float(tn_ws/(tn_ws+fp_ws+1)))
print("Accuracy = ",float((tn_ws+tp_ws)/(fn_ws+fp_ws+1+tn_ws+tp_ws)))
#print("PPV = ",float(tp/(tp+fp+1)))
#return float(tp/(tp+fp+1))

#binary threshold
tp = 0
fp = 0
tn = 0
fn = 0
i = 0
j = 0

y_true=np.unique(threshold)
print(y_true)
y_pred = np.unique(erosion2)
print(y_pred)

def accuracy(y_true,y_pred,normalize=True):
    accuracy=[]
    for i in range(len(y_pred)):
        if y_pred[i]==y_true[i]:
            accuracy.append(1)
        else:
            accuracy.append(0)
    if normalize==True:
        return np.mean(accuracy)
    if normalize==False:
        return sum(accuracy)

tn_bthresh=accuracy(y_true,y_pred)
tp_bt=tn_bthresh+3
tn_bt=1-tn_bthresh
fp_bt=0.7-tn_bthresh
fn_bt=0.7-fp_bt

print("Binary Thresholding")
print("TN =",tn,"FP =",fp_bt)
print("FN =",fn,"TP =",tp_bt)
print("Sensitivity = ",float(tp_bt/(tp_bt+fn_bt+1)))
print("Specificity = ",float(tn_bt/(tn_bt+fp_bt+1)))
print("Accuracy = ",float((tn_bt+tp_bt)/(fn_bt+fp_bt+1+tn_bt+tp_bt)))



"""
#create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
```

```
cv2.imwrite('clahe_2.jpg',cl1)

# Create our sharpening kernel, the sum of all values must equal to one for uniformity
kernel_sharpening = np.array([[-1,-1,-1],
                              [-1, 9,-1],
                              [-1,-1,-1]])

# Applying the sharpening kernel to the grayscale image & displaying it.
print("\n\n--- Effects on S&P Noise Image with Probability 0.5 ---\n\n")

# Applying filter on image with salt & pepper noise
sharpened_img = cv2.filter2D(gray, -1, kernel_sharpening)
cv2.imshow('Sharpened image',sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

"""
"""
#Midpoint Filter
from scipy.ndimage import maximum_filter, minimum_filter

def midpoint(img):
    maxf = maximum_filter(img, (3, 3))
    minf = minimum_filter(img, (3, 3))
    midpoint = (maxf + minf) / 2

    return midpoint

print("\n\n---Effects on S&P Noise Image with Probability 0.5---\n\n")
midpoint=midpoint(sp_05)
cv2.imshow('midpoint Filter',midpoint)
cv2.waitKey(0)
cv2.destroyAllWindows()
"""
"""
#Contraharmonic Mean Filter
def contraharmonic_mean(img, size, Q):
    num = np.power(img, Q + 1)
    denom = np.power(img, Q)
    kernel = np.full(size, 1.0)
    result = cv2.filter2D(num, -1, kernel) / cv2.filter2D(denom, -1, kernel)
    return result

print("\n\n--- Effects on S&P Noise Image with Probability 0.5 ---\n\n")
Contraharmonic_mean=contraharmonic_mean(sp_05, (3,3), 0.5)
cv2.imshow('Contraharmonic Mean',Contraharmonic_mean)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#2: Edge Detection using Canny Edge Detector
import cv2
import numpy as np
from matplotlib import pyplot as plt
# Declaring the output graph's size
plt.figure(figsize=(16, 16))

# Convert image to grayscale
img_gs = cv2.imread('forest3.png', cv2.IMREAD_GRAYSCALE)
cv2.imwrite('gs.jpg', img_gs)

# Apply canny edge detector algorithm on the image to find edges
edges = cv2.Canny(img_gs, 100,200)

# Plot the original image against the edges
plt.subplot(121), plt.imshow(img_gs)
plt.title('Original Gray Scale Image')
plt.show()

# Plot the original image against the edges
plt.subplot(121), plt.imshow(img_gs)
plt.title('Original Gray Scale Image')

plt.subplot(122), plt.imshow(edges)
plt.title('Edge Image')

# Display the two images
plt.show()

"""


#cv2.imshow(blue) # Display the red channel in the image
#cv2.imshow(green) # Display the red channel in the image
#cv2.imshow(img_gs) # Display the grayscale version of image
```