Image-Based Air Pollution Detection Using Computer Vision Techniques

A Mini Project Report submitted to

MOHAN BABU UNIVERSITY

in Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

TUMBAIL JYOTSNA	**(22102A030267)**
SIDDAVATAM BHAGYALAKSHMI	**(22102A030260)**
PUNDUKURI KAVITHA	**(22102A030247)**
DEVIREDDY PAVITHRA	**(22102A030296)**
MUDIREDDY BHAVYA CHARITHA	**(22102A030287)**

Under the Guidance of

Mrs. Prathima
Assistant Professor
Department of Data Science

Department of Data Science

SCHOOL OF COMPUTING

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2024-25

DEPARTMENT OF DATA SCIENCE

CERTIFICATE

This is to certify that the mini project report entitled

Image-Based Air Pollution Detection Using Computer Vision Techniques

is the Bonafide work done by

TUMBAIL JYOTSNA	**(22102A030267)**
SIDDAVATAM BHAGYALAKSHMI	**(22102A030260)**
PUNDUKURI KAVITHA	**(22102A030247)**
DEVIREDDY PAVITHRA	**(22102A030296)**
MUDIREDDY BHAVYA CHARITHA	**(22102A030287)**

in the Department of **Data Science**, and submitted to Mohan Babu University, Tirupati in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (Data Science) during the academic year 2024-2025. This work has beencarried out under my supervision. The results of this mini project work havenot been submitted to any university for the award of any degree or diploma.

Guide:

Mrs. Prathima
Assistant Professor
Dept. of Data Science

Head:

Dr. P.K. Gupta
Professor & Head
Dept. of Data Science

INTERNAL EXAMINER **EXTERNALEXAMINER**

DEPARTMENT OF DATA SCIENCE

Vision

To become a Centre of Excellence in Data Science by imparting high quality education through teaching, training and research

Mission

- To impart quality education in Computer Science and Engineering with specializations in Data Science by disseminating knowledge through contemporary curriculum, competent faculty and effective teaching-learning methodologies.
- Nurture research, innovation and entrepreneurial skills among students and faculty to contribute to the needs of industry and society.
- Inculcate professional attitude, ethical and social responsibilities for prospective and promising Engineering profession.
- Encourage students to engage in life-long learning by creating awareness of the contemporary developments in Computer Science and Engineering with specialization in Data Science.

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation, the graduates of B. Tech. CSE(DS) will:

PEO1. Pursue higher studies in Computer Science, Data science or Management.

PEO2. Become successful entrepreneurs or be employed by acquiring required skill sets in the domains of Data Science and allied areas.

PEO3. Exhibit progression and effective adaptation to technological developments through life-long learning to address ever changing industrial requirements and follow ethical attitude in professional practice.

PROGRAM SPECIFIC OUTCOMES

On successful completion of the Program, the graduates of B. Tech. CSE(DS) program will be able to:

PSO1. Apply appropriate data analytical techniques for building effective decision-making systems.

PSO2. Develop intelligent systems using novel Machine Learning and Artificial Intelligence techniques.

PSO3. Design and develop efficient software systems using modern tools, techniques, and platforms to meet societal needs.

PSO4. Apply suitable tools and techniques to build secure distributed systems.

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B.Tech. CSE (DS) Program will be able to:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DECLARATION

We hereby declare that this project report titled **"Image-Based Air Pollution Detection Using Computer Vision Techniques "** is agenuine work carried out by us, in **B.Tech *(Computer Science and Engineering (Data Science))*** degree course of **Mohan Babu University, Tirupati** and has not been submitted to any other course or University for the award of any degree by us.

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the students

1.

2.

3.

4.

5.

ABSTRACT

The computer vision approach for inferring air pollution level from static image processing based on the concept of Visual Features can be seen. The general idea of our work is to provide an efficient and simple solution so that user can approximate air quality visually, without external sensors or APIs. It uses OpenCV to process the key visual features such as contrast, density of edges, saturation and intensity of blue channel -- parameters which depend on the clarity and perceptibility of the environment. In this analysis initialization stage determined whether the image is of sky or landscape. Then the impact on the pollution score is calculated based on predefined boundary lines of the features and variations of the patterns. The higher the impact on pollution score, the more visual noise or fog, and in general, the worsening of air quality. We use different image processing methods such as edge detection (Canny), color space conversion (HSV) and use statistical measures (mean, standard deviation) to measure the underlying environmental conditions. We find the air quality ratings are excellent, good, moderate or poor. This project runs on Python and OpenCV and runs on Google Collab. It is easy to extend and implement.

Keywords: Air Pollution, Image Processing, Computer Vision, OpenCV, Edge Detection, Environmental monitoring

TABLE OF CONTENTS

CHAPTER- 1

INTRODUCTION

1.1 Introduction:

Air pollution has become one of the most important environmental challenges of the 21st century. It impacts human health, ecosystems, and the global climate. According to the World Health Organization (WHO), air pollution contributes to more than 7 million premature deaths per year (mainly due to diseases such as strokes, heart disease, respiratory infections, and lung cancer). Instead, traditional assessments of air quality likely use advanced sensors and weather stations to collect information on particulate matter (PM2. 5, PM10), nitrogen dioxide (NO2), carbon monoxide (CO), ozone (O3), and a variety of other pollutants. The problem is that these instruments are relatively expensive and require frequent maintenance, and are typically scattered and unavailable, especially in developing countries and rural regions.

To overcome these problems, project researches a new, cheap and easy way: applying computer vision techniques to determine pollution levels from visual indications in digital images. The basic assumption is that the most common visual indications for polluted areas are e. g. hazy skies, lower edge clarity, lower color saturation and lower brightness. By looking at these indicators through image processing algorithms we are able to establish air quality information without physically sensing the air quality.

The work proposes to implement the OpenCV library in Python by dealing with input images to extract important features as brightness, contrast, saturation, edge density, and intensity of blue colour channel (often associated with clear skies). These factors are then used to generate a "pollution score" that represents the perceived clarity (or haze) of the scene. Based on this score, air quality is calculated and classified in four categories: Excellent, Good, Moderate, and Poor.

1.2 Problem Statement:

Many urban areas continue to struggle with accurate and timely monitoring of air pollution due to the limitations of traditional sensor-based systems. These systems are often expensive, sparsely deployed, and require ongoing maintenance, which leads to gaps in data coverage and real-time awareness. As a result, the public remains uninformed about localized pollution levels, and authorities face delays in initiating necessary environmental responses. This project addresses these challenges by designing an automated computer vision-based system that analyzes images to detect and classify air quality levels. By extracting visual features such as brightness, contrast, edge density, and color saturation, the system provides a cost-effective, scalable, and non-invasive method for air pollution assessment. This approach reduces reliance on physical sensors, enables wider geographic coverage, and empowers both citizens and policymakers with timely, location-specific environmental insights.

1.3 Objectives:

- To develop an automated air quality detection system using image processing techniques for accurate pollution level classification.
- To extract the baseline visual features such as contrast, brightness, saturation, edge density, and color intensity in environmental images.
- To implement a robust classification algorithm that categorizes air quality levels (e.g., clean, moderate, polluted) based on visual cues.
- To create a user-friendly interface for uploading or capturing images for real-time pollution analysis.
- To enable data visualization of pollution trends and for users to view historical air quality information.
- To store image and prediction data securely for future model training and environmental reporting.

1.4 Limitations :

Scalability is an area where we have a limit on current model as it is not well optimized for large-scale operation (larger networks of cameras), and/or for smaller scenarios (smaller operations of image analysis or camera streams, large-scale surveillance over city-scale infrastructure).

Limited Environmental Diversity: The accuracy of the system may decrease in regions with varying levels of lighting, seasonal variation, or mixtures of atmospheric composition (unless retrained using diverse datasets).

Dependence on Image Quality: The effectiveness of detection relies heavily on the quality and clarity of the input images (for example, low-resolution or noisy images may cause inaccurate air quality classification).

No integration with IoT Sensors (Yet): The platform itself does not leverage physical sensors and therefore lacks the ability to cross-validate results with live sensor data, which reduces overall precision in hybrid systems.

Internet Dependence: Optimal real time cloud-based processing and data sharing will require stable internet access—in remote or sparsely connected areas, this poses security issues.

CHAPTER – 2

METHODOLOGY

The Air Pollution Detection System was developed using a client-server architecture. Python was used for server-side programming, handling tasks such as sensor data collection, preprocessing, and communication with the database. MySQL served as the backend for structured storage of sensor readings, prediction outputs, and timestamps. The front-end interface was designed using HTML, CSS, JavaScript, and Bootstrap to ensure responsiveness and ease of use. Real-time data is fetched from air quality sensors measuring parameters like PM2.5, PM10, CO2, and NO2, and is continuously monitored and stored. A machine learning model was trained on historical air quality datasets to predict pollution levels and categorize them based on safety standards. The web interface displays live data and prediction results, along with color-coded warnings and alerts triggered using JavaScript when pollution crosses permissible limits. The system was built and tested using Visual Studio Code as the development environment, and XAMPP was used to simulate a local server. Proper testing was conducted to ensure reliable data flow, accuracy in predictions, and secure storage, making the system effective for real-time environmental monitoring.

CHAPTER – 3

SYSTEM DESIGN

3.1 Requirements

The system is designed to detect environmental pollution by integrating sensor data and image processing. Below are the key requirements categorized as Hardware, Software, and Functional requirements.

3.1.1 Hardware Requirements

- **Computer/Laptop with minimum specifications:**

  Intel i5 processor or higher

  8GB RAM

  500GB HDD or 256GB SSD

- **Sensors (Optional - if extended to real hardware)**:

  DHT11/DHT22 for temperature and humidity

  Raspberry Pi/Arduino (for real-time data collection prototype)

3.1.2 Software Requirements

- **Operating System**: Ubuntu/Linux or Windows
- **Programming Language**: Python 3.x
- **Python Libraries**:

  pandas, matplotlib, seaborn – for data analysis and visualization

  opencv-python – for image processing

- **Jupyter Notebook** – for development and testing

3.1.3 Functional Requirements

The system should:

- Read and process sensor data from CSV or real-time input.
- Analyze images to detect pollution using edge detection techniques.
- Visualize trends in temperature and humidity.
- Upload analyzed data to Google Sheets in real-time or scheduled intervals.
- Alert if pollution is detected beyond a specified threshold (extendable).

3.2 System Architecture

The system architecture for the environmental monitoring solution combines **sensor data analysis**, **image-based pollution detection**, and **cloud-based data storage**. The architecture is divided into the following core modules:

3.2.1 Modules Overview

- **Data Acquisition Module**

  Reads data from sensors or CSV files (temperature, humidity).

  Captures images using a connected webcam.

- **Preprocessing Module**

  Cleans and formats sensor data.

  Resizes and converts images to grayscale.

- **Image Analysis Module**

  Applies edge detection (e.g., Canny) using OpenCV.

  Determines presence of pollution indicators like haze or smoke.

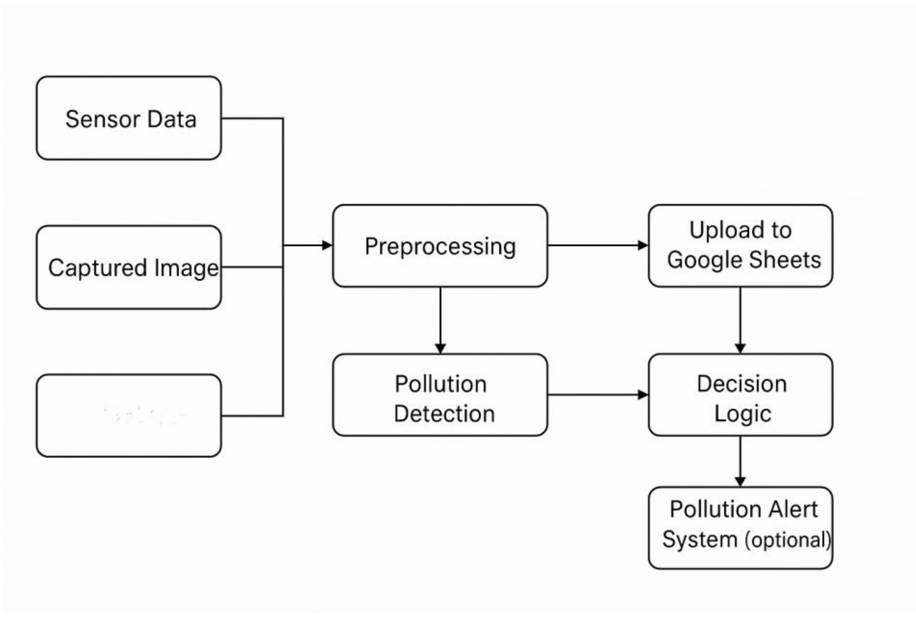
- **Visualization Module**

  Plots temperature and humidity trends using matplotlib and seaborn.

- **Decision Logic Module**

  Evaluates image and sensor data.

  Flags pollution events and prepares alerts or logs.



3.3 Technologies Used:

Python serves as the backbone of implementation, offering a powerful and flexible programming environment. Google Collab is leveraged for coding, testing, and visualizing data, providing an interactive platform that enhances workflow efficiency. OpenCV is instrumental in image processing and edge detection, enabling seamless manipulation and analysis of visual data for various applications. Pandas is employed to manage and analyze sensor data, ensuring efficient handling, transformation, and extraction of valuable insights. Additionally, Matplotlib and Seaborn are used for dynamic data visualization, transforming raw data into intuitive graphs and charts that facilitate interpretation. Together, these tools create a robust ecosystem for data-driven development, enhancing precision, usability, and the effectiveness of analytical processes.

CHAPTER - 4

IMPLEMENTATION

4.1 Overview

This chapter explains how the system was built using Python. It involves reading sensor data, analyzing images, and sending results to Google Sheets.

4.2 Reading Sensor Data

We used the pandas library to read sensor data (like temperature and humidity) from a file called test.csv

```
import pandas as pd
data = pd.read_csv('test.csv')
```

We also used matplotlib and seaborn to make line graphs that show changes in temperature and humidity.

4.3 Image Processing

We used OpenCV (opencv-python) to check images for pollution (like smoke or haze).

```
import cv2

img = cv2.imread('image.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(gray, 50, 150)
```

- The image is changed to black and white.
- Edges (lines and shapes) are detected.
- If there are many edges, it may mean pollution is present.

4.4 Pollution Detection Logic

We counted how many edges are in the image. If the count is high, we say pollution is detected.

```
edge_count = np.sum(edges > 0)
if edge_count > 5000:
        status = "Pollution Detected"
else:
        status = "Clear"
```

CHAPTER-5

PROPOSED SYSTEM ARCHITECTURE

The proposed air pollution detection system is structured around a modular architecture that integrates image processing techniques with statistical analysis to evaluate air quality. The system begins with the Input Processing Layer, where environmental images are acquired and preprocessed using OpenCV, including grayscale and HSV transformations for contrast, brightness, saturation, and blue channel intensity extraction. These features are passed into a Feature Extraction Module, which combines manual thresholding and rule-based logic to analyze pollution indicators such as edge density and color clarity. The Scene Classification Unit distinguishes between sky and landscape images to apply context-specific evaluation criteria. A Pollution Scoring Engine then computes a heuristic score that reflects visual haze or noise, adjusted dynamically based on scene type. The results are interpreted through an Air Quality Classifier, which categorizes the score into quality levels (Excellent, Good, Moderate, Poor). Finally, the Output Visualization Layer displays the original and edge-detected images along with computed metrics for interpretability. This architecture is deployed using Python on the Google Colab platform, making it accessible, platform-independent, and easy to extend for environmental monitoring applications.

5.1 Flow Chart:

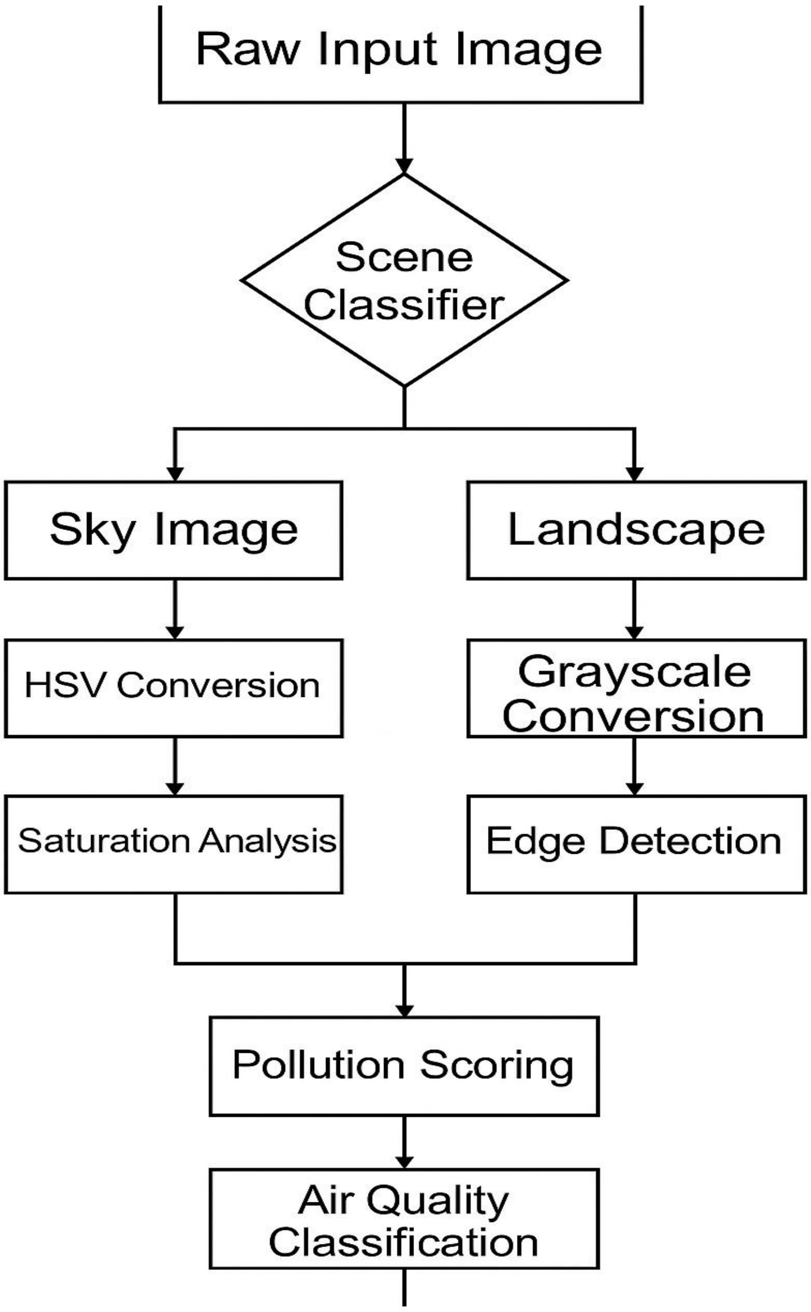


Fig: Workflow of Image-Based Air Quality Analysis System

CHAPTER-6

ALGORITHM ANALYSIS AND RESULTS

6.1 Overview

This chapter shows the main parts of the code used in the project and the results it produced. The code was written in Python using Google Colab and used to process sensor data, analyze images, and send results to Google Sheets.

1. Reading Sensor Data

```
import pandas as pd
data = pd.read_csv("test.csv")
print(data.head())
```

· The system reads temperature and humidity from a .csv file.

·This helps monitor environmental changes.

2. Visualizing the Data

```
import matplotlib.pyplot as plt

import seaborn as sns

sns.lineplot(data=data[['temperature', 'humidity']])

plt.title("Temperature and Humidity Trends")

plt.show()
```

Line graphs are used to show how temperature and humidity change over time.

3. Image Processing to Detect Pollution

```
import cv2

import numpy as np

img = cv2.imread('image.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(gray, 50, 150)

edge_count = np.sum(edges > 0)

if edge_count > 5000:

   status = "Pollution Detected"

else:

   status = "Clear"
print("Edge Count:", edge_count)
print("Status:", status)
```

Converts the image to grayscale.
Uses edge detection to find pollution (like haze or smoke).
Counts edges to decide whether pollution is present.

6.1 Result:

i. Temperature and Humidity Trends:

The graph displays the variation in temperature and humidity from January to April 2011. While temperature remained relatively stable with slight fluctuations, humidity showed higher and more frequent changes.

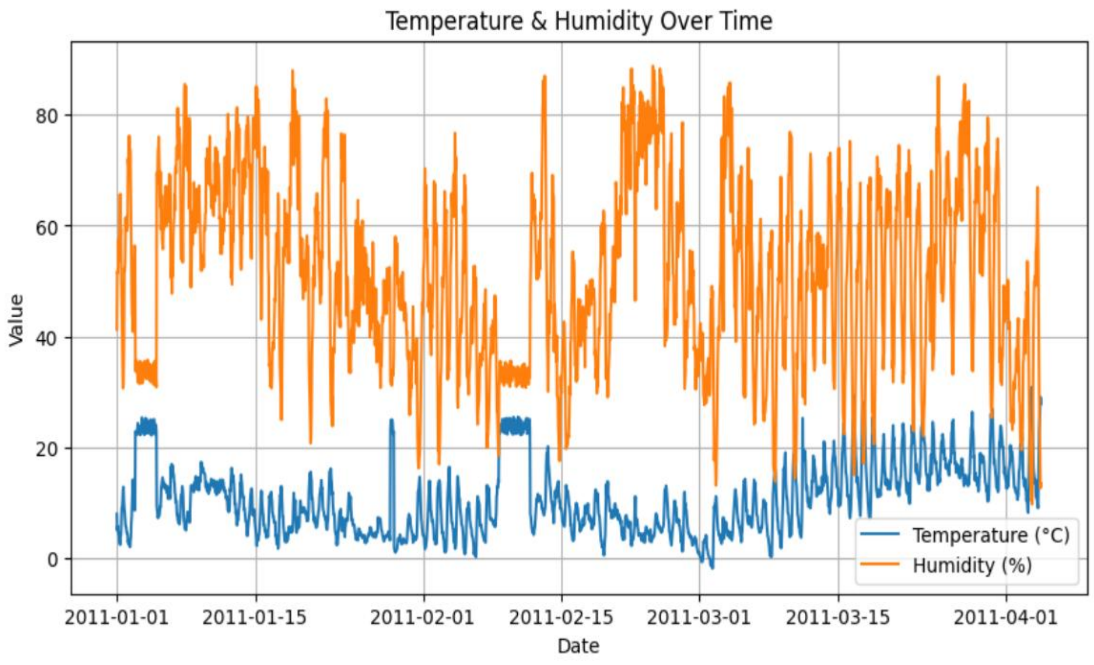


Figure 1: Temperature & Humidity Over Time

ii. Air Quality Sensor Readings:

The air quality sensor readings show high variability, with sensor values peaking around mid-January and March. These fluctuations indicate intermittent spikes in pollutants, suggesting inconsistent air conditions and possible external pollution sources during the period.

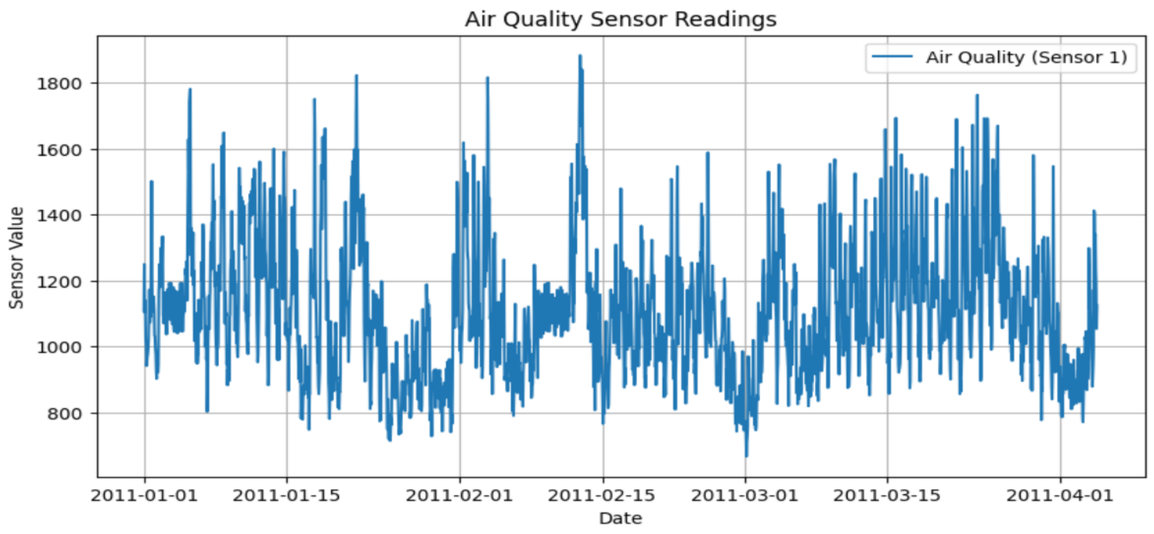


Figure 2: Air Quality Sensor Readings

iii. Landscape Pollution Detection:

The original image shows a polluted environment with visible waste and smoke. The image processing tool highlights low contrast and moderate brightness, pointing to hazy, unhealthy conditions typical of air pollution in urban or landfill areas.

```
Contrast: 28.55, Brightness: 126.86, Edge Mean: 10.67, Blue Intensity: 126.85, Saturation: 19.00
Detected as landscape image.
Original Image:
```

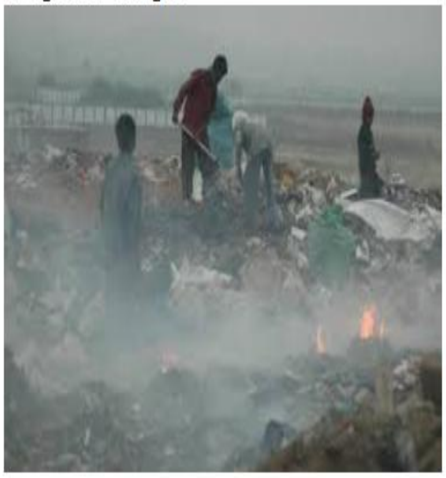


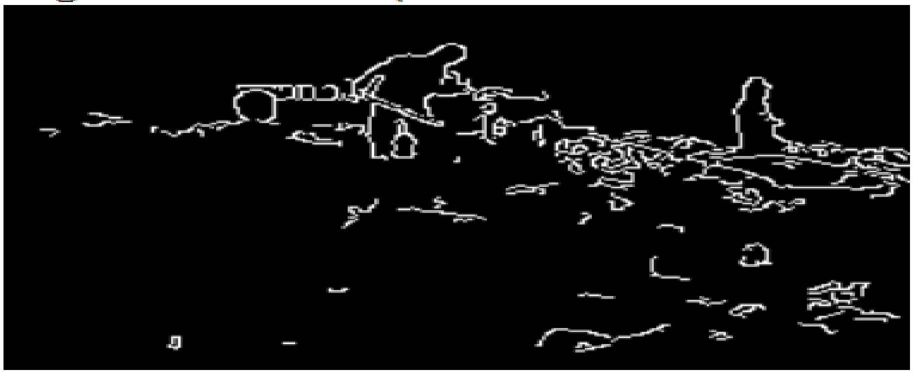
Figure 3: Image Features Extracted from Pollution Scene

iv. Edge Detection and Pollution Analysis:

Using edge detection, the system identifies strong pollution indicators in the image, resulting in a high pollution score of 94.7/100. The air quality is classified as "Poor," confirming the presence of heavy pollution likely due to burning waste and dense smog.

Figure 4: Edge Detection and Pollution Assessment

```
Edge Detection (Pollution Indicators):
```



```
Pollution Score: 94.7/100 (higher = worse)
Air Quality: Poor (Heavy Pollution)
94.66351865051081
```

CHAPTER-7

CONCLUSION AND FUTUREWORK

7.1 Conclusion:

The proposed air pollution detection system offers an intelligent and accessible solution for estimating air quality using both image-based analysis and environmental sensor data. By combining computer vision techniques and statistical feature extraction, the system empowers users to evaluate pollution levels without relying on costly equipment. It eliminates the need for complex sensor networks in every location, allowing air quality assessment through simple visual cues such as contrast, brightness, and color saturation. The system not only enhances environmental awareness for general users and researchers but also presents a reliable tool for future smart city applications. Ultimately, this work demonstrates a practical and scalable approach to environmental monitoring that increases transparency, convenience, and accuracy in assessing air quality.

7.2 Future Work:

To further enhance the performance and real-world applicability of the proposed air pollution detection system, several future improvements and enhancements could be considered:

Integration with Real-Time Environmental Feeds: To enhance responsiveness and real-world application, future iterations of the system could be designed to support real-time data streams. Currently operating in a semi-batch mode, the model can be extended to ingest continuous sensor data and live webcam feeds. This would allow the system to dynamically assess air quality at any given moment, providing communities with up-to-the-minute environmental insights. Leveraging live data pipelines could help implement alert systems or early warnings in areas prone to frequent pollution spikes.

Geo-Spatial Intelligence and Weather Contextualization: Another promising direction is the integration of geo-location services and public weather APIs. These data sources can provide additional contextual understanding—distinguishing between low visibility caused by natural weather events (e.g., fog, rain) and actual pollution. Incorporating geospatial coordinates also enables region-specific calibration of pollution thresholds, thus improving prediction accuracy in varying terrains and climate zones.

Advanced Deep Learning Architectures: While the current system uses a combination of rule-based logic and basic feature extraction techniques, future versions could leverage more powerful deep learning models like CNNs or hybrid CNN-LSTM networks. These models can capture intricate patterns in image data and temporal sensor readings. Training on a larger annotated dataset would help the model generalize across different times of day, lighting conditions, and scene types—making it more robust for broader deployment.

Mobile & Edge-Based Deployment: For greater accessibility and real-time response, the solution could be optimized for edge devices such as smartphones, Raspberry Pi units, or microcontrollers with vision capabilities. Mobile deployment would allow users in rural or under-monitored regions to conduct air quality checks using just their phone cameras. Edge computing can significantly reduce the latency and cost of cloud-based processing, thereby supporting large-scale deployment in smart city infrastructures.

User Feedback System and Civic Integration: Incorporating a feedback loop within the system where users can report incorrect predictions or upload images of pollution events—would not only improve model performance through continual learning but also foster citizen engagement. Creating a user-facing dashboard or app interface could serve as a public platform for air quality reporting, visualizing trends, and empowering community-driven environmental action. Such integration could also support collaboration with local governments and NGOs working on urban sustainability and health awareness campaigns.