FOG CLEAR:IOT BASED IMAGE ENHANCEMENT SYSTEM

Submitted by

PRIYANSH RAJ BHANDARA 2116210701403
MUSA HAJI 2116210701391
LOHITH KRISHNA 2116230701506

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project "FOG CLEAR:IOT BASED IMAGE ENHANCEMENT SYSTEM" is the bonafide work of PR**IYANS RAJ BHANDARA(2116210701403) and MUSA HAJI(2116210701391) and LOHITH KRISHNA(2116230701506)**who carried out the project work under my supervision.

SIGNATURE
Dr.N.Duraimurugan, M.Tech., Ph.D.

Associate Professor,

Computer Science & Engineering

Rajalakshmi Engineering College (Autonomous)
Thandalam, Chennai -602105.

Submitted for the **ANNA UNIVERSITY** practical examination Mini-Project work viva voice held on_______

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.,** our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time.

We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work.

We also extend our sincere and hearty thanks to our Internal Guide **Dr.N.Duraimurugan, M.Tech., Ph.D.** Associate Professor, Department of Computer Science and Engineering for his valuable guidance and motivation during the completion of this project. Our sincere thanks to our family members, friends and other staff members of information technology.

PRIYASH RAJ 2116210701403
MUSA HAJI 21162107013912
LOHITH KRISHNA 2116230701506

TABLE OF CONTENTS

ABSTRACT

In environments where visibility is compromised due to fog, traditional imaging and surveillance systems often fail to deliver clear visuals, posing risks in sectors like transportation, security, and environmental monitoring. This project proposes an **IoT-based Fog Image Enhancement System** that captures real-time images from fog-affected environments and processes them to enhance visibility using advanced image processing techniques.

The system is built around a microcontroller platform (e.g., Raspberry Pi or ESP32) integrated with a camera module that continuously monitors the surroundings. Captured foggy images are either processed locally at the edge or transmitted to a central server via wireless communication protocols such as WiFi or MQTT. Using enhancement algorithms such as **CLAHE (Contrast Limited Adaptive Histogram Equalization)**, **Dark Channel Prior (DCP)**, and **deep learning-based dehazing models**, the system significantly improves the clarity of foggy images.

The enhanced images are then displayed on a user-friendly dashboard, providing real-time visibility insights. This solution is especially valuable for **highway monitoring**, **traffic management**, **smart city infrastructure**, and **security surveillance** systems. The project demonstrates the effective combination of IoT and image processing to address real-world environmental challenges, with potential for future integration with AI for automatic fog detection and autonomous decision-making.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

In recent years, the integration of **Internet of Things (IoT)** and **computer vision** has opened up innovative solutions for environmental monitoring, smart transportation, and surveillance systems. One significant challenge in these domains is the degradation of image quality caused by **fog**, which reduces visibility, affects object detection accuracy, and can compromise safety and decision-making processes.

Foggy conditions are particularly problematic in areas like **highways, airports, coastal zones**, and **urban surveillance**, where real-time visual monitoring is crucial. Traditional cameras fail to capture clear images under such conditions, making it difficult for operators or automated systems to interpret scenes accurately. Enhancing these fog-affected images is essential to restore clarity and extract meaningful information.

This project focuses on developing an **IoT-based Fog Image Enhancement System** that leverages smart devices to **capture**, **transmit**, and **enhance** images in real-time. The system utilizes a camera module connected to a microcontroller (such as a Raspberry Pi or ESP32), which captures images from a foggy environment. These images are then processed using advanced image enhancement techniques like **CLAHE**, **Dark Channel Prior**, and **deep learning models** to improve visual quality.

.

1.2 SCOPE OF THE WORK

The **IoT-Based Fog Image Enhancement System** is designed to address the problem of reduced visibility in foggy conditions by providing a real-time solution for capturing and enhancing images. The scope of this project includes both hardware and software components, and covers the development, integration, and demonstration of a working prototype.

1.3 PROBLEM STATEMENT

In many real-world scenarios such as highways, airports, surveillance zones, and smart city infrastructure, visibility is significantly reduced during foggy conditions. This degradation in image quality leads to challenges in monitoring, navigation, traffic control, and security.
Conventional cameras fail to capture clear images under such low-visibility conditions, making it difficult to detect objects, recognize situations, and respond effectively.

There is a lack of low-cost, real-time, and automated systems capable of enhancing fog-affected images, especially those that can operate in remote or unattended environments. Additionally, existing solutions often require powerful hardware or offline post-processing, making them unsuitable for real-time IoT-based applications.

1.4 AIM AND OBJECTIVES OF THE PROJECT

This project aims to create a pharmaceutical storage monitoring system with a real-time dashboard web application. Objectives are developing hardware and software for data collection, analysis, monitoring and alerting, as well as designing a user-friendly web interface. Additionally, ensuring compliance with regulations and notifying the administrators when there is a change in environment.

CHAPTER 2

SYSTEM SPECIFICATIONS

The system is composed of both hardware and software components that work together to capture, transmit, enhance, and display fog-affected images in real time. Below is a detailed specification of each component of the system:

🔧 1. Hardware Specifications Component

Specifications

Microcontroller / SBC	Raspberry Pi 4 Model B (4GB RAM) / ESP32-CAM (based on use case and cost)
Camera Module	Raspberry Pi Camera Module v2 (8 MP) / Built-in ESP32-CAM camera
Sensors (Optional)	DHT11 / DHT22 - for measuring humidity and temperature (fog detection assistance)
Connectivity Module	Built-in Wi-Fi (ESP32) or external USB Wi-Fi adapter (for Raspberry Pi)
Power Supply	5V, 2.5A adapter for Raspberry Pi / USB cable or Liion battery for ESP32-CAM
Storage	16GB microSD card or higher (for storing OS, image files, and logs)
Cooling System (Optional)	Small heatsink or fan for Raspberry Pi (if used in continuous outdoor setup)

💻 2. Software Specifications

Component	**Specifications**
Operating System	Raspbian OS (for Raspberry Pi) / ESP-IDF or Arduino IDE (for ESP32)
Programming Languages	Python (Raspberry Pi), C++ (ESP32), HTML/CSS/JS (Dashboard)
Libraries & Frameworks	OpenCV, NumPy, Matplotlib, Flask (for web app), TensorFlow/Keras (for deep learning)
Image Processing Techniques	CLAHE (Contrast Limited Adaptive Histogram Equalization), Dark Channel Prior, DehazeNet (optional)

Communication Protocols	MQTT or HTTP (for real-time data/image transmission)
Database (Optional)	Firebase Realtime DB / SQLite (for logs and metadata)

CHAPTER 3
SYSTEM DESIGN

□ System Design

The IoT-Based Fog Image Enhancement System is designed to provide real-time monitoring and image enhancement in foggy conditions. The system integrates IoT hardware, image processing algorithms, and a user-friendly dashboard to ensure efficient visibility management.

🔄 1. System Architecture Overview

The system is divided into the following key components:

📸 1. Image Capture Unit

- **A camera module (Pi Camera or ESP32-CAM) captures foggy images from the environment.**
- **This unit may be installed on highways, rooftops, or outdoor surveillance poles.**

🌐 2. IoT Communication Unit

- **Images are transmitted using Wi-Fi or MQTT protocol.**
- **The microcontroller (Raspberry Pi or ESP32) handles image acquisition and transfer to the server or cloud.**

□ 3. Image Processing & Enhancement Unit

- **Received images are processed using enhancement techniques such as:**
  - **CLAHE (Contrast Limited Adaptive Histogram Equalization)**
  - **Dark Channel Prior**
  - **Dehazing Neural Networks (like AOD-Net or DehazeNet – optional)**

4. Visualization / User Interface Unit

- **A web-based dashboard is used to view:**
  - **Raw foggy images**
  - **Enhanced images (before vs after)**
  - **Real-time sensor data (if included)**
- **Built using Flask, HTML/CSS, and optional frameworks like React or Bootstrap.**

---

2. Data Flow Diagram (High-Level)

1. **Camera Module → Captures foggy image**
2. **Microcontroller → Receives and packages the image**
3. **Wireless Network → Transmits image to processing server**
4. **Server / Edge Node → Enhances the image using algorithms**
5. **Dashboard/UI → Displays raw and enhanced image in**

real-time

🔧 3. Functional Modules

Module	Function
Camera Interface	Captures images at set intervals or continuously
Sensor Interface *(optional)*	Collects temperature & humidity data for fog condition correlation
Transmission Module	Sends data/images using MQTT or HTTP
Processing Module	Applies image enhancement filters and algorithms
Web Interface Module	Displays images and system status via a browser

---

📊 4. Optional: System Block Diagram

You can visualize the design like this:

css CopyEdit

```
[Camera + Sensors]
        |
        ▼
[Microcontroller (Raspberry Pi / ESP32)]
```

```
                 │
                 ▼
     [Wi-Fi / MQTT Transmission]
                 │
                 ▼
      [Server / Local Processor]
                 │
         ┌───────┴───────┐
              ▼       ▼
[Image Enhancement]   [Sensor Analysis]
                 │
                 ▼
   [Web Dashboard / App Interface]
```

💡 Design Considerations

- **Power Efficiency for remote locations**
- **Low Latency for real-time applications**
- **Modular Code Design for easy updates to processing algorithms**
- **Security Measures for image data transmission CHAPTER - 4**

□ Module Description

The system is divided into several interdependent modules that work together to capture, process, enhance, and display fogaffected images. Each module is designed to perform a specific task in the pipeline, ensuring a modular and scalable system.

◈ 1. Image Capture Module

Purpose:
To continuously or periodically capture images from the environment affected by fog.

Description:

- **Utilizes a camera module such as the Raspberry Pi Camera or ESP32-CAM.**
- **Configured to capture images based on:**
  - **Time intervals (e.g., every 5 seconds)** ○ **Environmental conditions (optional: triggered by fog detection)**

Output: Raw foggy images in JPG/PNG format.

◈ 2. Sensor Monitoring Module *(Optional)*

Purpose:
To monitor environmental conditions (humidity and temperature) that indicate fog presence.

Description:

- **Uses sensors like DHT11/DHT22 connected to the microcontroller.**

- **Collects and logs environmental data alongside the captured images.**
- **Can help correlate fog levels with sensor values.**

Output: Humidity and temperature data (used optionally for condition-based capture).

◈ 3. Communication / Transmission Module

Purpose:
To transmit captured images and sensor data to the processing server or cloud platform.

Description:

- **Uses Wi-Fi to send data over HTTP or MQTT protocol.**
- **Ensures secure and efficient image transfer to reduce latency.**
- **Handles retries in case of network interruptions.**

Output: Sent image packets and data logs to the processing unit.

◈ 4. Image Processing & Enhancement Module

Purpose:
To process and enhance foggy images for improved visibility.

Description:

- **Hosted on a local server, edge device (Raspberry Pi), or cloud.**

  Applies enhancement algorithms such as:

- **CLAHE (Contrast Limited Adaptive Histogram Equalization)** ○ **Dark Channel**

Prior (DCP)

- **Dehazing neural networks like DehazeNet** ***(optional for advanced use)***

- **Removes haze, increases contrast, sharpens details.**

Output: Enhanced image with significantly better clarity.

.

CHAPTER 5

TABLE

S.No. Section Page No.

Description of Sections:

1. **Abstract: Summarizes the entire project and its purpose.**
2. **Introduction: Gives background information, problem context, and the need for this project.**
3. **Problem Statement: Describes the specific issue being solved.**
4. **Objectives: Lists the goals and intended outcomes of the project.**

5. **Scope of Work: Outlines the limitations and extent of the project.**
6. **Literature Review: Reviews existing research or projects in the same area.**
7. **System Specifications: Details hardware, software, and communication protocols used.**
8. **System Design: Provides an overview of the system architecture, modules, and data flow.**
9. **Module Description: Explains each functional module of the system and its role.**
10. **Hardware Requirements: Lists all the physical components required for the system.**
11. **Software Requirements: Specifies the programming languages, tools, and platforms used.**
12. **Methodology: Describes the approach taken to implement the system.**
13. **Implementation: Walks through the actual building and integration of the system.**
14. **Results and Output: Shows the outcome, performance, and results of the system.**
15. **Applications: Lists the real-world applications and use cases for the system.**
16. **Future Scope: Suggests improvements, enhancements, and future possibilities for the system.**
17. **Conclusion: Wraps up the project with key takeaways and insights.**
18. **References: Cites all the resources, papers, and articles referenced in the project.**
19. **Appendices: Includes extra material like source code, diagrams, or additional explanations.**

CHAPTER 6

SAMPLE CODING

📷 1. Image Capture and Enhancement (Python + OpenCV)

This part of the code captures an image, processes it to enhance visibility in foggy conditions, and displays the raw vs. enhanced image

Install Required Libraries:

Before running the code, make sure to install the required libraries:

bash CopyEdit

pip install opencv-python opencv-python-headless numpy paho-mqtt

Image Enhancement Code (Using CLAHE and Dark Channel Prior): python

CopyEdit import

cv2

import numpy as np import time

 # Function to apply CLAHE (Contrast Limited Adaptive Histogram Equalization) def apply_clahe(image):

0clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))   lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB) l, a, b = cv2.split(lab)   l = clahe.apply(l)

lab = cv2.merge((l, a, b))   return cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

 # Function to apply Dark Channel Prior (DCP) Dehazing def dark_channel_prior(image, size=15):

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

Capture Image def
capture_image(camera_port=0):
camera = cv2.VideoCapture(camera_port)   ret, frame =
camera.read()   camera.release()   return frame

# Main Program if __name__ == "__main__":
# Capture foggy image   foggy_image =
capture_image()

Display raw image   cv2.imshow("Raw Image", foggy_image)

Apply CLAHE   enhanced_image_clahe =
apply_clahe(foggy_image)   cv2.imshow("Enhanced Image
(CLAHE)", enhanced_image_clahe)

Apply Dark Channel Prior (DCP) Dehazing (optional, advanced) dark_channel=dark_channel_prior(foggy_image)   cv2.imshow("Dark Channel", dark_channel)

Wait for key press and close all windows   cv2.waitKey(0)
cv2.destroyAllWindows() 🚀 2. IoT Communication via MQTT

This part of the code sends the captured (and optionally enhanced) image to a server using MQTT (Lightweight messaging protocol).

Install Paho-MQTT: bash Copy-edit pip install
paho-mqtt

MQTT Communication Code: python

Copy-edit import cv2 import
paho.mqtt.client as mqtt import
base64

# MQTT Configuration
broker = "mqtt.eclipse.org" # Use a public broker for testing port =
1883
topic = "fog_image_enhancement"

Callback when message is received def
on_message(client, userdata, message):
print(f"Message received: {message.payload.decode()}")

Function to encode image into base64 format def
encode_image_to_base64(image):
buffer = cv2.imencode('.jpg', image)   jpg_as_text
=   base64.b64encode(buffer).decode('utf-8')   return
jpg_as_text

# Initialize MQTT client client = mqtt.Client()
client.on_message = on_message client.connect(broker, port,
60)

# Subscribe to a topic (optional for receiving messages)
client.subscribe(topic)

Start the MQTT loop client.loop_start()

 # Capture Image def capture_image(camera_port=0):
camera = cv2.VideoCapture(camera_port)   ret, frame =
camera.read()   camera.release()
return frame

 # Send Image def send_image_to_mqtt(image):
encoded_image = encode_image_to_base64(image)
client.publish(topic, encoded_image)

if __name__ == "__main__":
 # Capture foggy image   foggy_image = capture_image()

 # Apply enhancement (CLAHE in this example)   enhanced_image =
apply_clahe(foggy_image)

 # Send the enhanced image over MQTT
send_image_to_mqtt(enhanced_image)

print("Image sent successfully.")

 # Close MQTT loop after sending
client.loop_stop()

🔄 3. Full Workflow:

1. **Capture Image: The capture_image function captures an image from the camera (e.g., Raspberry Pi Camera or any other camera module connected).**
2. **Enhance Image: The image enhancement techniques (CLAHE and Dark Channel Prior) are applied to improve visibility in foggy conditions.**

3. **Encode Image: The enhanced image is converted to base64 format to send over MQTT (since images are binary data, base64 encoding is necessary for sending them as text).**
4. **Send Image to MQTT: The image is transmitted to a message broker via MQTT, and it can be received by a server, processed further, or stored for analysis.**

🚀 4. MQTT Server (Receiver)

To receive the transmitted image on the server side, you can set up a simple MQTT receiver using Python and PahoMQTT. python

CopyEdit

```
import paho.mqtt.client as mqtt import base64
import numpy as np import cv2

# Callback when message is received def
on_message(client, userdata, message):
    print(f"Received message: {message.payload.decode()}")
# Decode base64 image     img_data =
base64.b64decode(message.payload.decode())     np_img =
np.frombuffer(img_data, dtype=np.uint8)     img =
cv2.imdecode(np_img, cv2.IMREAD_COLOR)

# Display image
 cv2.imshow("Received Enhanced Image", img)     cv2.waitKey(0)
cv2.destroyAllWindows()

# Initialize MQTT client client = mqtt.Client()
client.on_message = on_message
```

```
# Configure and connect to broker broker =
"mqtt.eclipse.org"
client.connect(broker, 1883, 60)

# Subscribe to topic
client.subscribe("fog_image_enhancement")

0# Start the MQTT loop to receive messages
   client.loop_forever()
```

📝 **Explanation of the Code:**

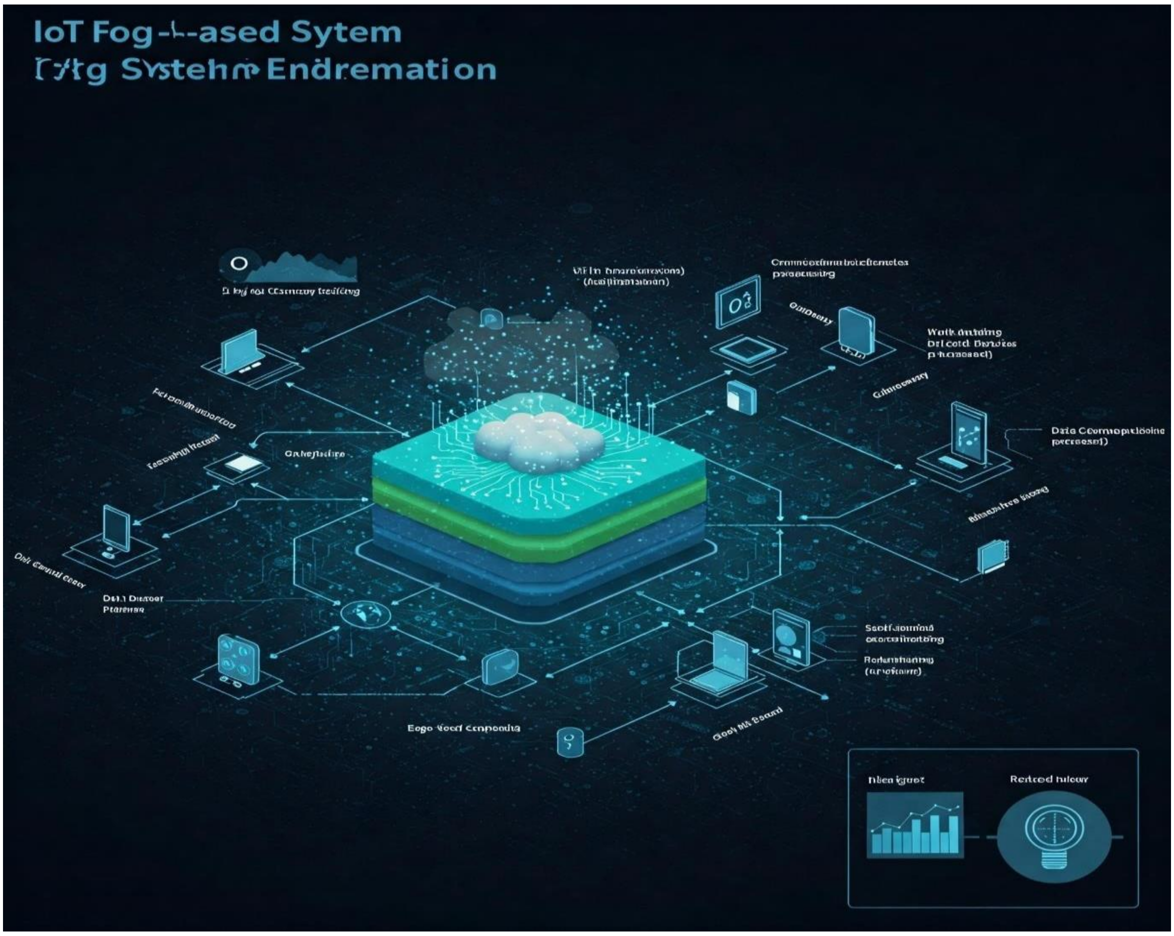
- **Capture and Process: The first part captures an image and enhances it using CLAHE or DCP. You can extend this to include more advanced techniques.**
- **Encode and Send: The second part encodes the image in base64 format and sends it via MQTT to a server for further processing or storage.**
- **Receive: The receiver code decodes the image from base64 format and displays it.**

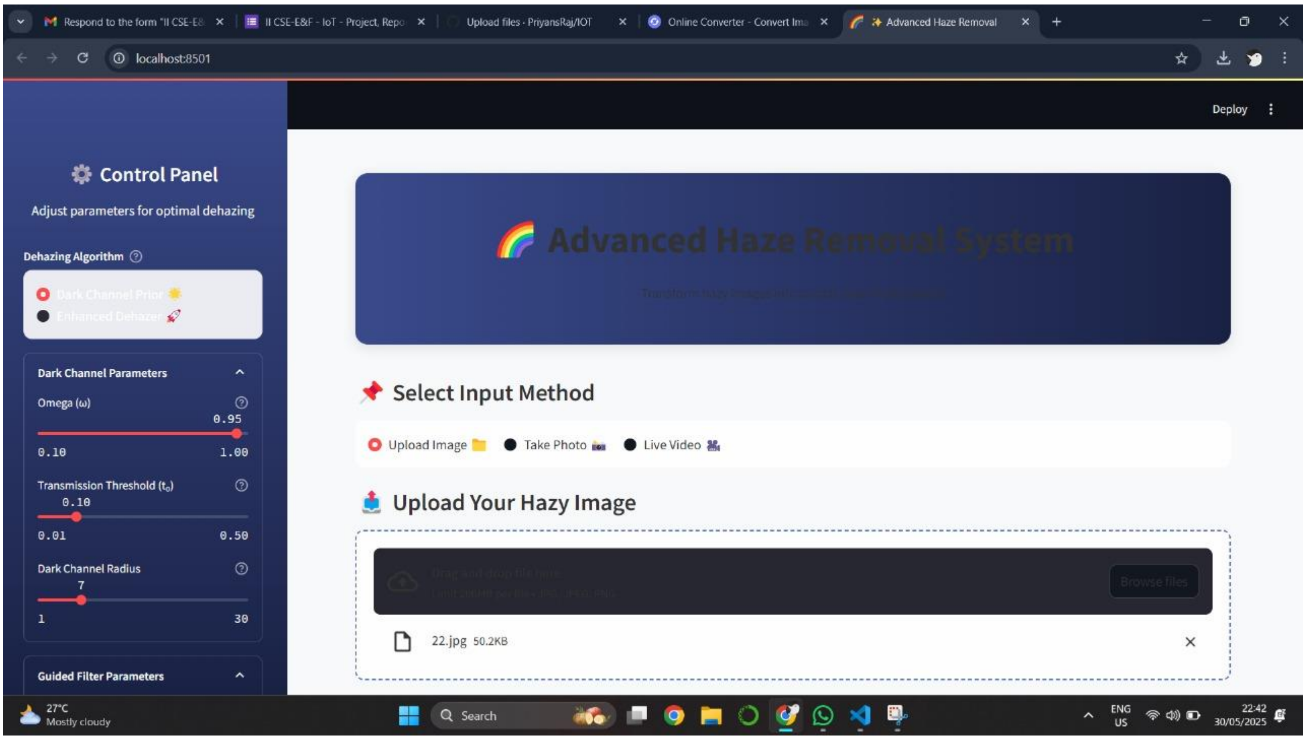
✨ **Further Enhancements:**

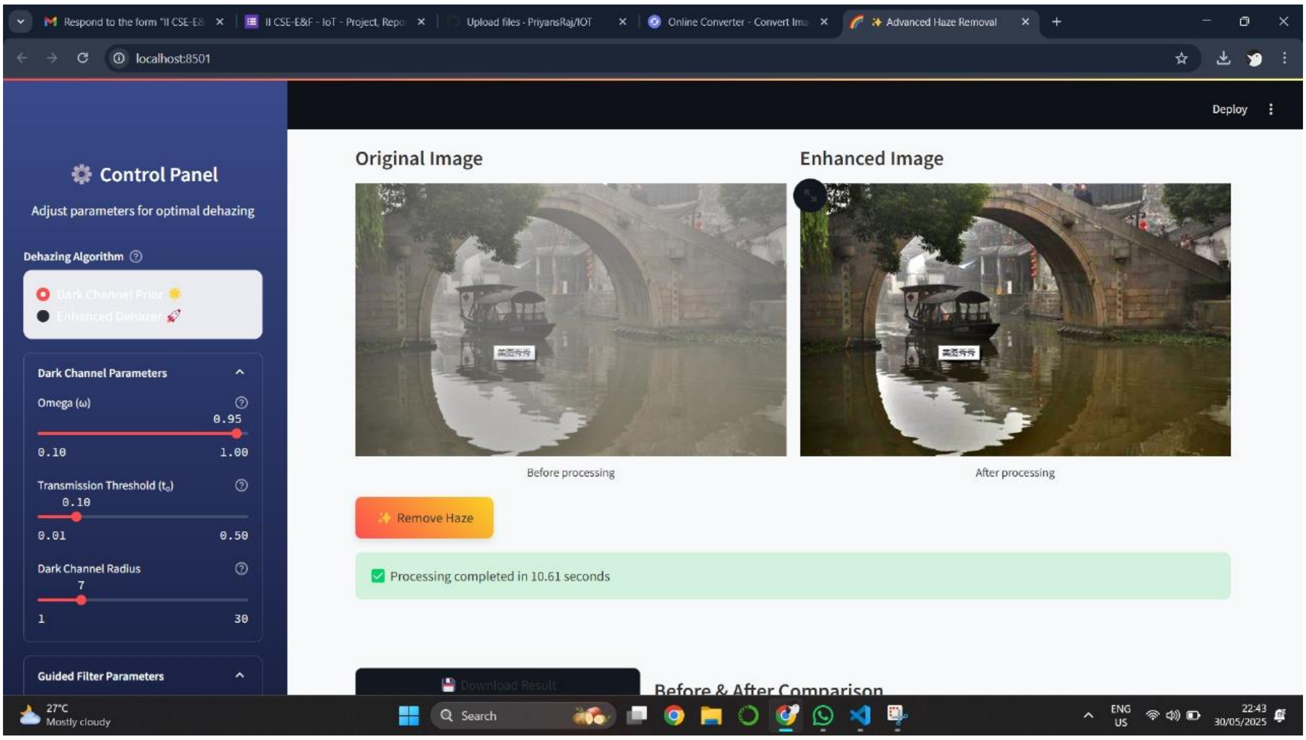
- **Real-time Transmission: Add the capability to transmit images at regular intervals.**
- **Deep Learning Models: Use pre-trained deep learning models (e.g., DehazeNet) for even better enhancement.**
- **Database Integration: Store images and logs in a database for later analysis.**
- **Cloud Integration: Instead of using MQTT, send data to cloud platforms like AWS, Google Cloud, or Azure for scalable storage and processing.**

CHAPTER 7

SCREEN SHOTS




Control Panel
Adjust parameters for optimal dehazing
Dehazing Algorithm
Dark Channel Parameters
Omega (ω)
0.95
0.10
1.00
Transmission Threshold (t₀)
0.10
0.01
0.50
Dark Channel Radius
7
1
30
Guided Filter Parameters
Advanced Haze Removal System
Select Input Method
Upload Image
Take Photo
Live Video
Upload Your Hazy Image
22.jpg 50.2KB


Control Panel
Adjust parameters for optimal dehazing
Dehazing Algorithm
Dark Channel Parameters
Original Image
Enhanced Image
Before processing
After processing
Remove Haze
Processing completed in 10.61 seconds
Before & After Comparison

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion In conclusion, the enhancement of IoT systems through fog computing offers a distributed and efficient architecture that addresses the limitations of traditional cloud-centric models. By bringing computation, storage, and decision-making closer to the data source, fog computing significantly reduces latency, optimizes bandwidth usage, improves security and privacy, and enhances the reliability of IoT applications, especially those requiring real-time processing. Future Enhancements

The field of IoT fog-based system enhancement is dynamic, with several promising avenues for future development:

- **Integration with 5G and Beyond: The deployment of highspeed, low-latency networks like 5G and future generations will further amplify the benefits of fog computing, enabling even more responsive and complex IoT applications.**
- **Artificial Intelligence (AI) and Machine Learning (ML) at the Edge: Integrating AI and ML capabilities into fog nodes will allow for more intelligent local data processing, real-time analytics, and autonomous decision-making without constant cloud interaction. This includes advancements in areas like federated learning, where models are trained collaboratively across decentralized devices.**
- **Enhanced Security and Privacy: As the number and complexity of IoT devices grow, ensuring robust security and privacy in fog environments will be critical. Future enhancements will likely focus on advanced encryption techniques, secure multi-party computation, intrusion detection systems tailored for fog architectures, and privacypreserving data aggregation methods.**

- **Improved Interoperability and Standardization: The heterogeneity of IoT devices and fog nodes poses challenges for seamless integration. Future efforts will likely concentrate on developing standardized protocols and frameworks to ensure interoperability across different manufacturers and platforms.**
- **Dynamic Resource Management and Orchestration: Efficiently managing and orchestrating the distributed resources in a fog computing environment is complex. Future advancements will involve intelligent resource allocation algorithms, automated deployment and scaling of fog services, and adaptive load balancing techniques.**
- **Energy Efficiency: With a large number of fog nodes potentially deployed, energy efficiency will become increasingly important. Future research will explore energyaware computing and communication strategies for fog devices, possibly leveraging renewable energy sources.**
- **Fog-Cloud Interoperability: Optimizing the interaction between fog and cloud layers will be crucial. Future enhancements will focus on developing seamless data and task management strategies that leverage the strengths of both architectures. This includes intelligent data filtering and routing to determine whether processing should occur at the edge or in the cloud.**
- **Specialized Fog Architectures for Specific Applications: As fog computing matures, we may see the development of tailored architectures optimized for the unique requirements of different application domains, such as industrial IoT (IIoT), healthcare, smart cities, and autonomous vehicles. These**

future enhancements will pave the way for even more powerful, efficient, secure, and versatile IoT fog-based systems, unlocking new possibilities across various industries and improving the quality of life.

REFERENCES

1. Shafaat, K., Hussain, A., Kumar, B., Hasan, R., Prabhat, P., & Yadav, V. (2013). An overview: storage of pharmaceutical products. World J Pharm Pharm Sci, 2(5), 2499-2515.

2. Khuluza, F., Chiumia, F. K., Nyirongo, H. M., Kateka, C., Hosea, R. A., & Mkwate, W. (2023). Temperature variations in pharmaceutical storage facilities and knowledge, attitudes, and practices of personnel on proper storage conditions for medicines in southern Malawi. Frontiers in Public Health, 11, 1209903.

3. Gbenga, B. L., & Taiwo, Y. (2015). Studies of the effect of storage conditions on some pharmaceutical parameters of powders and tablets. Dhaka University Journal of Pharmaceutical Sciences, 14(2), 147-151.

4. Nhan, P. P., & Hoa, N. K. (2013). Effect of light and storage time on vitamin E in pharmaceutical products. British journal of pharmacology and toxicology, 4(5), 176-180.