Journal of Electrical Engineering and Computer (JEECOM)
Vol. 7, No. 1 (2025), DOI: 10.33650/jeecom.v4i2
p-ISSN: 2715-0410; e-ISSN: 2715-6427

Water Level Detection and Flood Early Warning System Using Image Processing

Muhammad Akmal Ilmi [1], I Komang Somawirata[1], Michael Ardita[1]
[1] National Institute of Technology Malang, Indonesia

Article Info

Article history:

Received February 28, 2025
Revised March 10, 2025
Accepted April 22, 2025

Keywords:

Image Processing
YOLO
Water Level
Early Warning
Smart City

ABSTRACT

Image processing is a crucial method in modern technology, enabling computers to analyze and extract information from images or videos. This study focuses on the application of image processing technology to detect river water levels using CCTV cameras as part of a flood early warning system in a smart city. The YOLO (You Only Look Once) algorithm is utilized for real-time object detection, such as water levels, aiming to enhance prediction accuracy. This implementation is expected to provide richer visual data compared to traditional sensors. The study involves designing and testing a system that integrates hardware (CCTV cameras and high-spec computers) and software such as OpenCV and Python. Data in the form of river images is processed using image processing algorithms to analyze water levels in real-time. The system's performance is evaluated in terms of accuracy, precision, recall, and processing speed (FPS), as well as the environmental impact on detection results. The results indicate that the YOLO-based image processing system achieves high accuracy in detecting water levels. Additionally, the system is capable of sending early warnings via digital notifications, allowing more time for disaster mitigation. These findings suggest that image processing-based systems offer practical, efficient, and cost-effective solutions to support smart city technologies.

Corresponding Author:
I Komang Somawirata,
National Institute of Technology Malang, Jalan Raya Karanglo km 2 Malang, Malang and 65145, Indonesia
Email: kmgsomawirata@lecturer.itn.ac.id

1. INTRODUCTION

Natural disasters are events that can threaten human life, damage the environment, and cause economic and psychological losses [1]. One of the disasters that often occurs in Indonesia is flooding. Flooding is a natural phenomenon that occurs when the volume of water exceeds the capacity of the drainage system or river flow, causing harmful inundation. The main causative factors of flooding include high rainfall, topography of the area, and limited capacity of the drainage system [2]. Indonesia, as an archipelago with approximately 16,771 islands and 65,017 rivers, has a high risk of flooding, especially in urban areas with high population density and in the rainy season with large rainfall intensity [3]. Therefore, an early warning system that is able to detect potential flooding accurately and in real time is needed.

Various approaches have been developed to detect and predict floods, one of which is image processing technology. This technology enables water level monitoring by utilizing the relationship between the color of objects and certain parameters. Image processing technology offers a simpler, more practical and economical solution as it only requires a charge-coupled device (CCD) camera, a computer and software such as Visual Basic that is available in the market [4]. However, several challenges still need to be resolved in implementing this technology, including: (1) How can river water levels be detected using image processing technology using the YOLO algorithm? (2) How high is the accuracy, precision, and recall of the YOLO-based water level detection system? (3) How do frames per second (FPS) affect system performance in real time?

The utilization of image processing technology in flood early warning systems has been widely studied in previous research. Some studies show that the use of cameras and image processing algorithms can improve the accuracy of water level detection compared to conventional methods [5]. The YOLO (You Only Look Once) algorithm is known as one of the best approaches in object detection due to its ability to detect multiple objects in one process with high speed and good accuracy [6]. In addition, other studies have shown that the use of image processing methods in flood monitoring can provide more real-time data and assist in disaster mitigation by providing accurate information to authorities and the public [7].

This research aims to develop a water level detection system using YOLO-based image processing technology. The system will work by analyzing images or videos taken by CCTV cameras installed around the river to measure the water level automatically. The main parameters to be analyzed include accuracy, precision, recall, and processing speed (frames per second). In addition, this research also integrates an early warning system based on vehicle and other object detection using YOLOv8 to predict the impact of flooding on traffic and community activities. With this approach, the system is expected to function in real time and provide more accurate information for disaster mitigation.

This research makes a new contribution to the application of image processing technology for flood early warning systems, especially with the integration of the YOLO algorithm in river-level monitoring. The main innovation of this research is the combination of water level detection and impact analysis on traffic and community activities. The developed system is expected to be a cheaper, more practical and effective solution compared to traditional methods that rely on physical sensors or Internet of Things (IoT)-based approaches. Thus, this research not only improves the accuracy of flood detection but also provides broad benefits to the government and society in disaster risk mitigation efforts [8].

2. METHOD

This research uses an experimental method to design and test a flood early warning system based on image processing. The process includes hardware and software design, data collection, data analysis, and overall system testing. The research stages consist of:

a. System Design
   This stage involves preparing the required hardware and software specifications, such as CCTV cameras, computers, and image processing software, including the use of the YOLO algorithm for object detection.
b. System Implementation
   The designed system will first be implemented in a simulated environment. This test includes real-time monitoring of water levels and evaluation of system performance, including detection of objects that can be indicators of water levels. Then, install and configure the software in the test environment.
c. Data Collection
   The data collected are images or videos of river flow taken by CCTV cameras installed in strategic locations to record river flow. This data is then processed using image processing techniques to measure water levels.
d. System Testing
   System testing is carried out by simulating various water level conditions, both when normal and when there is an increase that indicates the potential for flooding. The goal is to ensure the accuracy of the system in detecting water levels and providing early warnings.
e. Data Analysis
   Data from image processing is analyzed to see the extent to which the system can predict flooding based on the increase in water levels.
f. System Evaluation
   The system will be evaluated based on the accuracy of detecting water levels, processing speed, and the effectiveness of early warnings using accuracy, precision, recall, and FPS (Frames Per Second) metrics.

2.1 Tool Block Diagram

The following block diagram of this tool design aims to get a scheme or series of tools made.

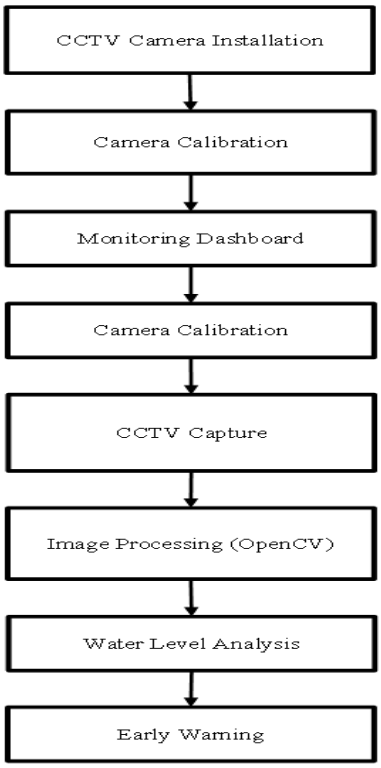

Figure 1. Tool Block Diagram

CCTV cameras are installed at flood-prone rivers to monitor water levels, with calibration ensuring accuracy. Captured images are processed using OpenCV, and if the water level exceeds the threshold, an alert is triggered. The system automatically sends early warnings to security personnel, while a monitoring dashboard provides real-time river condition updates.

2.2 Hardware Design

2.2.1 Computer Specifications

- **Device Name**: Axioo Pongo 725
- **Processor**: 12th Gen Intel(R) Core (TM) i7-12650H @ 2.30 GHz
- **Installed RAM**: 16.0 GB, sufficient to run the sign language recognition model without bottlenecks, capable of processing datasets of up to 50,000–100,000 letters in a single session.
- **System Type**: 64-bit operating system, x64-based processor
- **Pen and Touch**: No pen or touch input is available for this display
- **GPU**: NVIDIA GeForce RTX 2050 with 2,048 CUDA Cores and 4GB GDDR6 VRAM. In deep learning (e.g., using TensorFlow/PyTorch), the RTX 2050 can process up to 5,000–10,000 letters per batch in real-time inference (~60 FPS).
- **Benchmark**: Cinebench R23 (14,331 MBytes/Sec)
- **CPU Factor**: The CPU has 10 cores and 16 threads with a base clock of 2.30 GHz. This system can process approximately 500–1,000 letters per second in lightweight AI models.

2.2.2 CCTV (WebCam) Specifications

- **Camera Type**: Dahua IPC-HFW8232E-Z
- **Resolution**: 720p (2MP) Starlight
- **Infrared/Night Vision**: IR 100m
- **Lens**: Varifocal 2.8-12mm
- **Weather and Environmental Resistance**: IP67 Certification
- **Connectivity & Sensor Integration**: Supports PoE (Power over Ethernet)
- **Additional Features**: Motion Detection, AI-Based Flood Monitoring, and Time-Lapse Recording

2.3 Software Design
2.3.1 System Flowchart

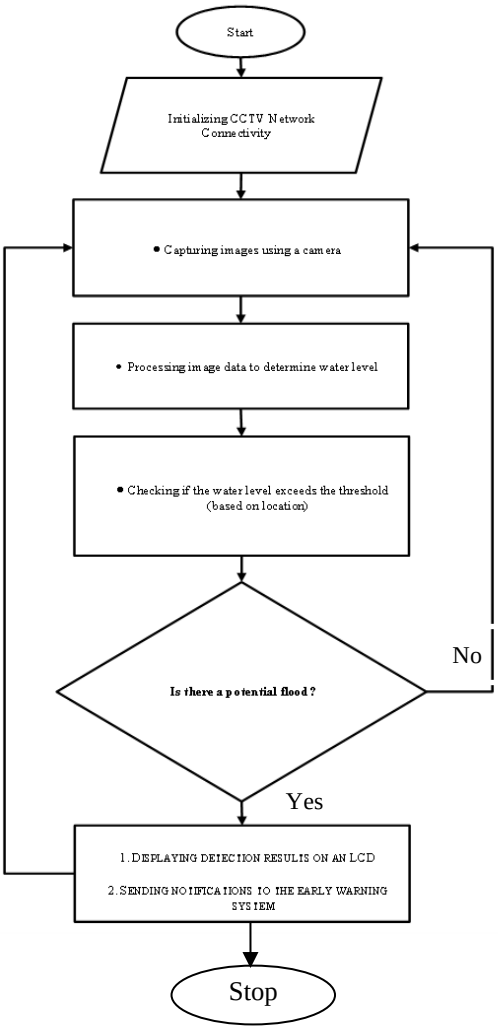

Figure 2. System Flowchart

The flowchart in Figure 2 illustrates the process of detecting potential floods using a CCTV system and image processing. The process begins with system initialization, followed by establishing CCTV network connectivity to ensure cameras are ready to capture images from monitored areas like rivers or flood-prone regions. The captured images undergo processing to determine the water level, which is then compared to a predefined threshold for that location. If the water level does not exceed the threshold, the system continues monitoring by periodically capturing new images. If a potential flood is detected, the system displays the detection results on an LCD, sends notifications to the early warning system, and activates alarms or warning signals to alert the public.

2.4 System Design
The system design covers the workflow from image capture to the delivery of early warning notifications as follows:

Figure 3. System Design

CCTV cameras are strategically installed along flood-prone river areas and calibrated for accurate water level measurements. Captured images are processed using OpenCV and YOLO for object detection and water level analysis. The system then analyzes the data, triggering an automatic early warning if the water level exceeds a predetermined threshold. Additionally, the processed image data is displayed on a monitoring dashboard for real-time observation.

Based on the block diagram, the entire system design can be illustrated in the image below.

Figure 4. Designing the Entire System

Example Code for Water Detection Using VSCode Application (Python)

This code is a GUI (Graphical User Interface) application built using Tkinter to detect water levels using video. The application utilizes the YOLO model for object detection and applies several image-processing techniques to enhance image visibility, particularly in low-light conditions. Additionally, the application calculates metrics such as accuracy, precision, and recall to evaluate detection performance.

1. Importing Libraries

```
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
import cv2
import numpy as np
import torch
from ultralytics import YOLO
import matplotlib.pyplot as plt
import winsound
import time
```

Figure 5. Importing Libraries

The necessary libraries are imported for GUI (Tkinter), image processing (OpenCV, NumPy), object detection model (YOLO), and visualization (Matplotlib).

2. Function to Enhance Image Brightness

```
def enhance_night_image(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    enhanced_gray = clahe.apply(gray)
    enhanced_bgr = cv2.cvtColor(enhanced_gray, cv2.COLOR_GRAY2BGR)
    gamma = 1.5
    look_up_table = np.array([((i / 255.0) ** (1.0 / gamma)) * 255
    return cv2.LUT(enhanced_bgr, look_up_table)
```

Figure 6. Function to Enhance Image Brightness

This function improves image brightness to enhance visibility at night. It uses CLAHE (Contrast Limited Adaptive Histogram Equalization) and gamma transformation. OpenCV and NumPy are used to enhance contrast and brightness. The steps are:

a. **Convert to Grayscale**
Converts the image from BGR (Blue-Green-Red) format to grayscale using cv2.cvtColor to simplify contrast enhancement without affecting color.

b. **CLAHE (Contrast Limited Adaptive Histogram Equalization)**
Creates a CLAHE object with clipLimit=3.0 to control contrast enhancement and tileGridSize=(8, 8) for localized adjustments. The apply function is used on the grayscale image to improve contrast.

c. **Convert Back to BGR Format**
Converts the CLAHE-enhanced grayscale image back to BGR format for further processing.

d. **Gamma Correction**
Sets gamma value to 1.5. Gamma correction adjusts image brightness. A lookup table is created using NumPy to map each input pixel intensity value to a new value based on the gamma equation:

$$Output\ Pixel = 255\ x\ \left(\frac{Input\ Pixel}{255}\right)^{1/\text{gamma}} \quad (1)$$

This adjustment makes the image appear brighter.

e. **Combining Results**
The cv2.LUT function applies the lookup table to the contrast-enhanced image, producing a brighter and clearer final image.
This function enhances night-time images with two main steps:
- Enhancing local contrast (handling dark or bright areas adaptively).
- Adjusting overall brightness using gamma correction.

3. Function to Detect Brown Water (Water Level Detection)

```
def detect_rising_water(frame):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_brown = np.array([5, 50, 50])
    upper_brown = np.array([35, 180, 180])
    mask = cv2.inRange(hsv, lower_brown, upper_brown)
    return cv2.countNonZero(mask), mask
```

Figure 7. Function to Detect Brown Water (Water Level Detection)

This function detects brown water areas in a frame using the HSV color space and returns the detected pixel count and a binary mask.

4. Function to Calculate Metrics

```
def calculate_metrics(true_positive, false_positive, false_negativ
    accuracy = (true_positive) / (true_positive + false_positive +
    precision = (true_positive) / (true_positive + false_positive)
    recall = (true_positive) / (true_positive + false_negative) if
    return accuracy, precision, recall
```

Figure 8. Function to Calculate Metrics

This function computes accuracy, precision, and recall metrics based on the number of correct and incorrect detections.

5. Entry Class with Placeholder

```
class EntryWithPlaceholder(tk.Entry):
    ...
```

Figure 9. Entry Class with Placeholder

This class creates an input field with a placeholder that disappears when the user starts typing.

6. Tkinter Main Window

```
window = tk.Tk()
window.title("🌊 Water Level Detection System")
window.geometry("600x600")
window.configure(bg="#f0f0f0")
```

Figure 10. Tkinter Main Window

Creates the main application window with a specific title and size:
- **Line 1**: window = tk.Tk() creates the main Tkinter window object.
- **Line 2**: window.title("Water Level Detection System") sets the window title.
- **Line 3**: window.geometry("600x600") sets the dimensions to 600x600 pixels.
- **Line 4**: window.configure(bg="#f0f0f0") sets the background color to light gray.

This code defines a basic Tkinter window with title, dimensions and background color.

7. Labels and Input Fields

```
inputs = [
    ("Pixels per Meter:", "E.g: 15"),
    ("Warning Level (m):", "E.g: 10"),
]
```

Figure 11. Labels and Input Fields

Adds labels and input fields for users to enter the required data.

8. **Treeview for Displaying Metrics**

```
metrics_tree = ttk.Treeview(window, columns=columns, show='headings'
```

Figure 12. Treeview for Displaying Metrics

Creates a table view to display computed metrics.

9. **Function to Process Video**

```
def process_video():
    ...
```

Figure 13. Function to Process Video

Handles video processing, including file selection, water detection, and metric computation. It also manages the display of processed video and updates graphs and metric tables in real-time.

10. **Video Processing and Water Detection**

```
while cap.isOpened():
    ...
```

Figure 14. Video Processing and Water Detection

A loop reads each video frame, enhances brightness if needed, detects brown water, and calculates water height. If the water level exceeds the warning threshold, the system flags a disaster alert.

11. **Displaying Detection Results**

```
cv2.putText(frame, f"Water Height: {real_water_height:.2f} m", (30,
```

Figure 15. Displaying Detection Results

Shows the detected water level in the processed video frame.

12. **Calculating FPS and Updating Metrics**

```
metrics_tree.insert("", "end", values=("Accuracy", f"{accuracy:.2f}"
```

Figure 16. Calculating FPS and Updating Metrics

Computes frames per second (FPS) and updates the metric display in the Treeview with accuracy, precision, recall, and FPS values.

13. **Disaster Alert**

```
if disaster_triggered:
    messagebox.showwarning("⚠ Disaster Alert!", "Flood detected! S
    winsound.Beep(1000, 1000)
```

Figure 17. Disaster Alert

If a disaster is detected, the application displays an alert and emits a beep sound to notify users.

14. **Displaying Water Level Graph**

```
plt.savefig("water_level_plot.png")
plt.show()
```

Figure 18. Displaying Water Level Graph

Saves and displays the water level graph during video processing.

15. **Button to Start the Process**

```
tk.Button(window, text="Start Processing", command=process_video, bg
```

Figure 19. Button to Start the Process

A button that initiates video processing when clicked.

16. **Displaying Metric Formula Table**

```
def show_metrics_table():
    ...
```

Figure 20. Displaying Metric Formula Table

Opens a new window displaying formulas for computed metrics, providing additional user information.

17. **Running the Tkinter Main Loop**

```
window.mainloop()
```

Figure 21. Running the Tkinter Main Loop

Starts the Tkinter application main loop, allowing user interaction with the interface.

2.5 System Workflow

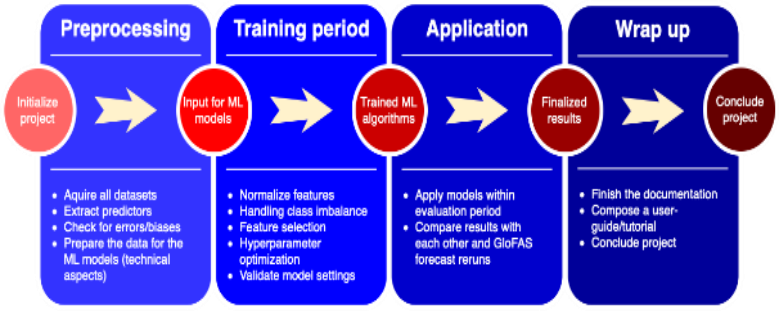

Figure 22. System Workflow

The ML model development process begins with preprocessing, where datasets are acquired, predictors are extracted, errors are checked, and data is prepared. In the training period, features are normalized, class imbalances are addressed, relevant features are selected, and model parameters are optimized. The Application phase involves applying the trained model to new data and comparing results with GloFAS forecasts. Finally, in the Wrap-Up, documentation is completed, a user guide is created, and the project is concluded.

3. RESULTS AND DISCUSSION

3.1 Data Collection

The data obtained in sign language results comes from finger measurements, classified into small, normal, and large sizes. For a predetermined distance of 30-50 cm, the sign language system will detect the input and display the accuracy results in terms of speed and graphic quality. These metrics are derived from dataset calculations using the following formula:

$$\frac{Letter\ Dataset}{2} \times 100\% \quad (2)$$

The data obtained in sign language results comes from finger measurements, classified into small, normal, and large sizes. For a predetermined distance of 30-50 cm, the sign language system will detect the input and display the accuracy results in terms of speed and graphic quality. These metrics are derived from dataset calculations using the following formula:

Table 1. Confusion Matrix

Prediction/Actual	Negatif	Positif
Negatif	TP	FP
Positif	FN	TN

Explanation:

- **True Positive (TP)** occurs when a positive prediction is made, and the actual result is also positive. For example, predicting that a woman is pregnant, and she is indeed pregnant.
- **True Negative (TN)** occurs when a negative prediction is made, and the actual result is also negative. For example, predicting that a man is not pregnant, and indeed, a man cannot be pregnant.
- **False Positive (FP)** occurs when a positive prediction is made, but the actual result is incorrect. For example, predicting that a man is pregnant, which is impossible.
- **False Negative (FN)** occurs when a negative prediction is made, but the actual result is incorrect. For example, predicting that a woman is not pregnant when she actually is.

Performance Evaluation Metrics

1. **Precision** (Measures the accuracy of the model's positive predictions)
   $$Precision = \frac{TP}{TP+FP} \quad (3)$$
2. **Recall** (Measures how many actual positive cases are correctly detected)
   $$Recall = \frac{TP}{TP+FN} \quad (4)$$
3. **Accuracy** (Indicates how well the model classifies all data)
   $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$
4. **F1-Score** (Balances between precision and recall)
   $$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall} \quad (6)$$
5. **Region of Interest (ROI)** (Used to focus on specific areas of an image, improving analysis accuracy)
   ROI Implementation in OpenCV:

```
x, y, w, h = 100, 200, 300, 400
roi = frame[y:y+h, x:x+w]
```

3.2 Training Water Level Detection

Table 2. Training Water Level

River	Recall	Precision Nilai 0/1	Accuracy	FPS
St. Lucie County India River	90 %	100 %	90 %	4.82 sec
Sri Gombak Puteri Malaysia River	85 %	100 %	85 %	4.54 sec
Batang Indonesia River	91 %	100 %	91 %	5.19 sec

The F1-score serves as a benchmark for the average precision and recall that have been determined, while accuracy can be used as a performance reference for the algorithm if the dataset in the system has a nearly equal number of false negatives and false positives.

- $F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall}$

  $F1 - Score = \frac{2\ x\ 1\ x\ 0.90}{1 + 0.90}$

  $F1 - Score = 0.95$

  $= 0.95 \times 100$

  $=$ **95 % (**St. Lucie County India River)
- $F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall}$

  $F1 - Score = \frac{2\ x\ 1\ x\ 0.85}{1 + 0.85}$

  $F1 - Score = 1.70$

  $= 1.70 \times 100$

  $=$ **170 % (**Sri Gombak Puteri Malaysia River)
- $F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall}$

  $F1 - Score = \frac{2\ x\ 1\ x\ 0.91}{1 + 0.91}$

  $F1 - Score = 0.95$

  $= 0.95 \times 100$

  $=$ 95 **% (**Batang Indonesia River)

The tests conducted using video input captured through CCTV cameras achieved a relatively high success rate, with the percentage obtained from calculations using the confusion matrix formula.

3.3 Data Analysis

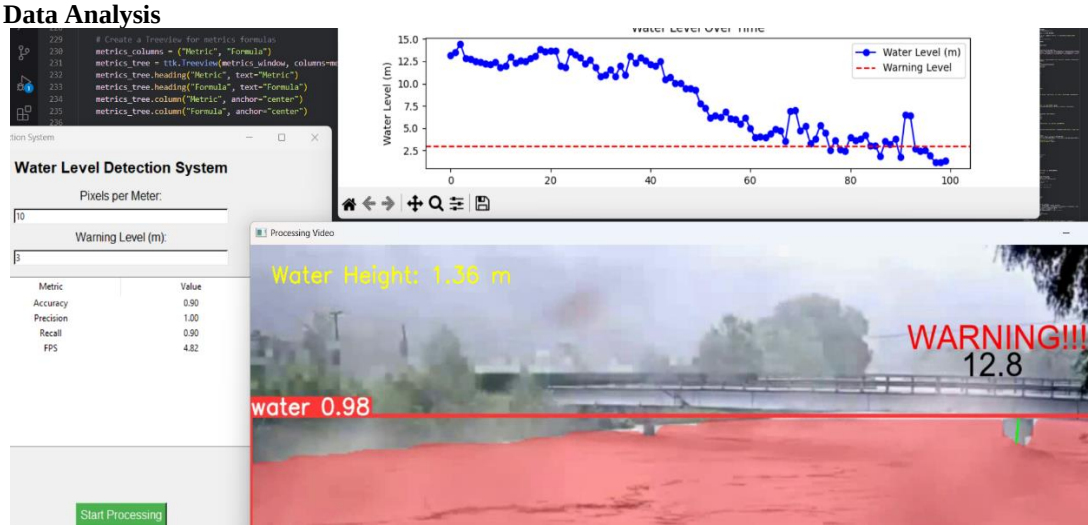

Figure 25. St. Lucie County India River

The image displays the interface of a water level detection system designed to monitor and alert potential flooding. This system likely utilizes computer vision and data analysis to measure water levels and compare them to predefined warning thresholds.

A. "Water Level Over Time" Graph

- Shows water level trends (in meters) over time.
- X-axis: Time (seconds or minutes).
- Y-axis: Water level (meters).
- Blue Line: Detected water level.
- Red Dashed Line: Warning threshold.

B. Video Processing with Overlays

- **Video Display:** CCTV footage with water areas highlighted in red using image processing.
- **"Water Height: 1.36 m" Indicator:** Current water level estimation based on pixel-to-meter scaling.
- **Red Horizontal Line:** Monitors water level.
- **"WARNING!!! 12.8" Text:** This is likely an alert when the water level exceeds a critical threshold.

C. Model Evaluation Parameters

- **Accuracy: 90%** – Correct water level classification.
- **Recall: 90%** – Flood detection capability.
- **Precision: 100%** – Accuracy in predicting dangerous water levels.
- **FPS: 4.82** – Video processing speed.

Technical Analysis

1. **Water Level Detection Method**
   - Image processing and machine learning can be used with techniques like thresholding, edge detection, or deep learning (CNN, YOLO, U-Net).
2. **Performance Evaluation**
   - 90% accuracy indicates reliable detection but room for improvement.
   - 90% recall favors early warnings over missed flood events.
   - 100% precision ensures all "flood" alerts are accurate.
3. **Hazard Warning System**
   - If the water level reaches 12.8 meters, the system issues a danger alert.
   - It can be integrated into an IoT-based early warning system that triggers alarms or SMS notifications to authorities.

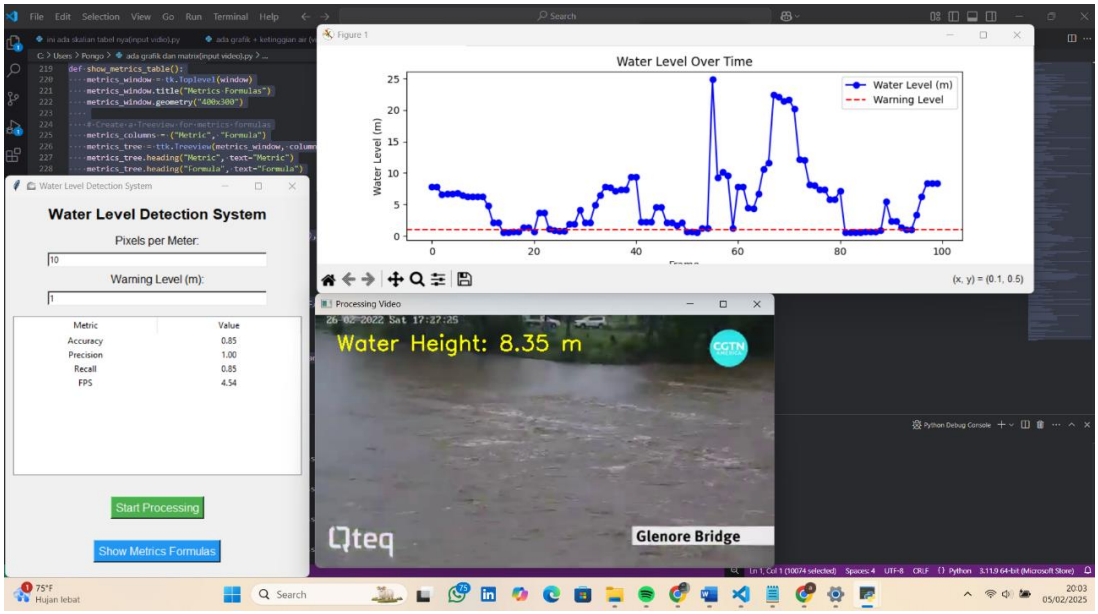

Figure 26. Sri Gombak Puteri Malaysia River

The image displays a water level monitoring system designed for early flood detection. It likely utilizes computer vision and data analysis to measure and compare water levels against predefined thresholds.

A. Water Level Detection System GUI

- Displays parameters for water level detection.
- "Pixels per Meter" and "Warning Level" are used for system configuration.
- "Start Processing" begins data analysis, while "Show Metrics Formulas" provides evaluation formulas.

B. Water Level vs. Time Graph

- Y-axis: Water level (meters).
- X-axis: Time (frames or seconds).
- The blue graph represents real-time water level fluctuations.
- The red dashed line indicates the "Warning Level" threshold.

C. Water Level Detection Video Output

- Real-time video annotation (e.g., "Water Height: 8.35 m").
- Measurement location: "Glenore Bridge."

Data Analysis

- The graph shows fluctuations with significant spikes.
- A sharp rise around point 60 suggests heavy rainfall or dam water release.
- Sudden drops may indicate flow changes or sensor errors.

System Accuracy Evaluation

Displayed Metrics:

- **Accuracy:** 85% (correct measurements vs. reference values).
- **Precision:** 100% (all detected valid levels are correct).
- **Recall:** 85% (system's ability to capture all valid data).
- **FPS:** 4 (processes 4 frames per second).

The system demonstrates high accuracy and precision, but the low FPS may limit performance in fast-flowing water conditions.

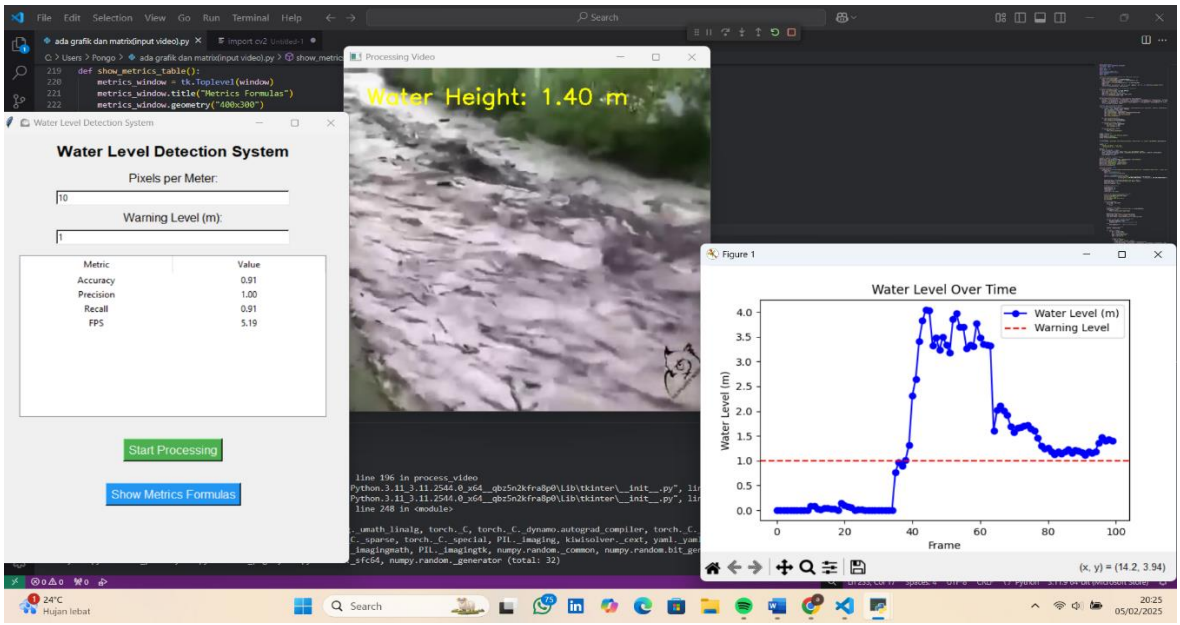

Figure 27. Batang Indonesia River

The graph represents water level variations over video frames:

- Initially low and stable.
- A significant spike around **frame 40** suggests increased water flow.
- Peaks at **~4 meters**, then fluctuates before stabilizing near the warning level.
- The **red horizontal line (e.g., 2.5m)** marks the warning threshold, with some sections exceeding it, indicating potential hazards.

System Performance Metrics

- **Accuracy (91%)** – Highly precise water level detection.
- **Precision (100%)** – No false positive detections.
- **Recall (91%)** – Few missed detections.
- **FPS (5.19)** – Processes ~5.19 frames per second, sufficient for near real-time monitoring.

This system effectively tracks water levels and provides early flood warnings, with potential improvements in processing speed.

4. CONCLUSION

This research confirms that using the YOLO model for water level detection is a promising approach, with its accuracy and reliability improving through continued testing. Key factors influencing performance include data quality, environmental conditions, real-time detection capabilities, and water level variability. While accuracy generally increases, occasional declines may occur due to calibration errors or environmental changes. The study also highlights the need for flood preparedness, as sudden water level rises were observed in some trials. To enhance detection, recommendations include regular instrument calibration, further research on improving efficiency, and integration of the system with an early flood warning system for better disaster preparedness.

ACKNOWLEDGEMENTS

With full gratitude to Allah SWT for all His blessings and grace, the author has completed this thesis as one of the requirements for obtaining a Bachelor of Electrical Engineering degree in the Electrical Engineering Undergraduate Program, Faculty of Industrial Technology, National Institute of Technology Malang. The author realizes that there are still many shortcomings in this thesis, so constructive criticism and suggestions are highly expected for future improvements. The author's deep gratitude goes to Allah SWT, family, supervisors Dr. Eng. I Komang Somawirata, ST, MT, and Dr. Michael Ardita, ST, MT, as well as all lecturers of Electrical Engineering ITN Malang for their guidance and support. Thank you also to my friends in arms, especially Nur Rizky Satrio and Aditya Djulqodri Rahman, as well as loyal friends who always provide motivation. Appreciation is also given to PT Detian Coking Indonesia in Indonesia Morowali Industrial Park

for the valuable experience provided, as well as all ITN Malang Electrical Engineering students class of 2021, alumni, and friends from the Abdul Toyyib Foundation, Unifier Generation, and other parties that cannot be mentioned one by one. Hopefully, this thesis will be useful to students and other readers and will be the basis for future scientific development.

REFERENCES

[1] Balahanti, R., Mononimbar, W., & Gosal, P. H. (2023). Analisis tingkat kerentanan banjir di Kecamatan Singkil Kota Manado. Jurnal Spasial, 11(1). ISSN 2442-3262.
[2] Bar, M.A., Sulistiyanto, S. and Basri, M.H., 2024. Perancangan Kontrol Sistem Fertigasi Pada Green House Berbasis IoT. *Akiratech*, *1*(1), pp.1-11.
[3] A. DLH, "Kerusakan Sungai dan Daerah Aliran Sungai di Indonesia," Dinas Lingkungan Hidup Kabupaten Grobogan, 2016. https://dlh.grobogan.go.id/index.php/info-lh/berita/36-kerusakansungai-dan-daerah-aliran-sungai-di-indonesia.
[4] Munir, R. 2004. Pengolahan Citra Digital dengan Pendekatan Algoritmik, Informatika. Bandung.
[5] Ginting, Segel dan M. Putuhena, Wiliam. (2014). SISTEM PERINGATAN DINI BANJIR JAKARTA. Jurnal Sumber Daya Air Vol. 10, No. 1. Pusat Penelitian dan Sumber Daya Air Jl. Ir. H. Juanda No. 193, Bandung.
[6] Amril Mutoi Siregar. "PENERAPAN ALGORITMA K-MEANS UNTUK PENGELOMPOKAN DAERAH RAWAN BENCANA DI INDONESIA", Universitas Buana Perjuangan Karawang.
[7] Hoeda, A.M.L., Tijaniyah, T. and Imaduddin, I., 2024. Perancangan Sistem Kontrol Aquarium Pintar Menggunakan Pompa Elektrik Berbasis IoT. *Akiratech*, *1*(1), pp.12-17.
[8] M. F. Noor, E. Utami, and E. Pramono, "Prediksi Curah Hujan Menggunakan Metode Anfis (Studi Kasus: Kabupaten Hulu Sungai Utara)," J. Inf. J. Penelit. dan Pengabdi. Masy., vol. 5, no. 2, pp. 24–29, 2019.
[9] K. S. Salamah and S. Anwar, "Rancang Bangun Sistem Pendeteksi Banjir Otomatis Berbasis Internet Of Things," Jurnal Teknologi Elektro, vol. 12, no. 1, p. 40, Jan. 2021, doi: 10.22441/jte.2021.v12i1.008.
[10] N. Pratama, U. Darusalam, and N. D. Nathasia, "Perancangan Sistem Monitoring Ketinggian Air Sebagai Pendeteksi Banjir Berbasis IoT Menggunakan Sensor Ultrasonik," JURNAL MEDIA INFORMATIKA BUDIDARMA, vol. 4, no. 1, p. 117, Jan. 2020, doi: 10.30865/mib.v4i1.1905.
[11] P. N. Rahardjo, "7 Penyebab Banjir di Wilayah Perkotaan yang Padat Penduduknya," J. Air Indones., vol. 7, no. 2, 2018.
[12] Hartono, Jogiyanto. 2011. Pengenalan Komputer. Yogyakarta: ANDI.
[13] Alfarisi, S. (n.d.). Program Studi Teknik Informatika Fakultas Teknik Matematika dan IPA Universitas Indraprasta PGRI. Retrieved from https://core.ac.uk/display/288089145?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1.
[14] Putri, Asti Riani. 2016. "Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya." JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika) 1 (01): 1–6. https://doi.org/10.29100/jipi.v1i01.18.
[15] Hutahaean, Harvei Desmon, Bakti Dwi Waluyo, and Muhammad Amin Rais. 2019. "Teknologi Identifikasi Objek Berbasis Drone Menggunakan Algoritma Sift Citra Digital" 04: 193–98.
[16] Effendi, Masud, Fitriyah Fitriyah, and Usman Effendi. 2017. "Identifikasi Jenis Dan Mutu Teh Menggunakan Pengolahan Citra Digital Dengan Metode Jaringan Syaraf Tiruan." Jurnal Teknotan 11 (2): 67.
[17] Widyaningsih, Maura. 2017. "Identifikasi Kematangan Buah Apel Dengan Gray Level Co-Occurrence Matrix (GLCM)." Jurnal SAINTEKOM 6 (1): 71. https://doi.org/10.33020/saintekom.v6i 1.7.
[18] Nugroho, A., 2025. Perancangan dan Pembuatan Dry Cabinet Menggunakan Konsep Internet of Things (IoT). *Akiratech*, *2*(1), pp.37-43.
[19] A. Kadir. 2019. Langkah Mudah Pemrograman OpenCV & Python. Elex Media Komputindo.
[20] Setyobudi, R., 2023. Utilization of tds sensors for water quality monitoring and water filtering of carp pools using IoT. *EUREKA: Physics and Engineering*, (6), pp.69-77.