KLE Society's

KLE Technological University

An Industry Project Report

On

Real-Time Object Detection and Recognition for Industrial Pallet Car Maintenance and Electrical Panel Analysis

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering in

Computer Science and Engineering

Submitted By

CHIRAG SANDEEP METGUD
01FE20BCS018

Under the guidance of

Mrs. Pratibha R Malagatti

SCHOOL OF COMPUTER SCIENCE &ENGINEERING,

HUBBLLI–580 031 (India).

Academic year 2023-24

B. V. Bhoomaraddi College Campus, Vidyanagar, Hubballi - 580031.
Karnataka (India)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Industry Project entitled "**Real-Time Object Detection and Recognition for Industrial Pallet Car Maintenance and Electrical Panel Analysis**" is a bonafide work carried out by the student **Mr. Chirag Sandeep Metgud** bearing USN **01FE20BCS018** in partial fulfillment of the completion of 8th semester B.E. course during the year 2023 – 24 at **Docketrun Tech Private Limited**. The Industry Project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said course.

Name of the Guide(s)
Mrs. Pratibha R Malagatti

Head of SoCSE
Dr. Vijayalakshmi M

Name of the examiners	**Signature with date**
1.....	1.....
2.....	2.....

DOCKETRUN TECH PRIVATE LIMITED

1st Floor, R.H. Kulkarni Building, KLE Tech University, Vidyanagar Hubballi Karnataka – 580031

CIN: U72900KA2019PTC128107 GST: 29AAHCD4356L1Z6 PAN: AAHCD4356L

Mob: +91-9449034387 Email: info@docketrun.com Website: www.docketrun.com

INTERNSHIP COMPLETION CERTIFICATE

We are delighted to certify that **Mr. Chirag Sandeep Metgud,** a student of Computer Science Engineering, **BVB College, KLE Tech University, Vidyanagar Hubballi**, has successfully completed the internship program (08/01/2024 – 31/05/2023) at **DocketRun Tech Pvt Ltd.**

During the internship, he played a pivotal role in multiple projects, focusing on AI development & completed numerous assignments, showcasing adaptability and excellence. Notably, he was a key contributor to the **"Real-Time Object Detection and Recognition for Industrial Pallet Car Maintenance and Electrical Panel Analysis"** project.

We would like to express our sincere gratitude for your exemplary performance and wish you continued success in all your future endeavors.

Warm Regard's

Ajay S Kabadi

Ajay S Kabadi
Founder, C.E.O
DocketRun Tech Pvt Ltd.

DECLARATION

I hereby declare that the Industrial Project Report entitled "**Real-Time Object Detection and Recognition for Industrial Pallet Car Maintenance and Electrical Panel Analysis**" is an authentic record of my own work as requirements of Industry Project Report during the period from 8th January,2024 to 31st May,2024 for the award of the degree of B.E. Under the guidance of Mrs. Pratibha R Malagatti.

(Signature of student)

Chirag Sandeep Metgud

Date:

01fe20bcs018

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of a number of individuals whose professional guidance and encouragement helped me in the successful completion of this report work.

I also take this opportunity to thank Dr. Vijaylakshmi M, Professor and Head, School of Computer Science and Engineering for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

I sincerely thank our guide Mrs. Pratibha R Malagatti, School of Computer Science and Engineering for her guidance and whole hearted co-operation during the course of completion.

I sincerely thank Mr. Huchcha Reddi, Docketrun Tech Private Limited for his support, inspiration and whole hearted co-operation during the course of completion.

My gratitude will not be complete without thanking our beloved parents, our seniors and our friends who have been a constant source of aspirations.

Chirag Sandeep Metgud

ABSTRACT

In this project, we developed a comprehensive system for real-time video analysis using advanced object detection models to enhance the maintenance and monitoring processes for Sinter Machine Pallet cars and electrical panels. The project encompassed four main objectives: nut missing detection, wheel counting, optical character recognition (OCR), and electrical panel detection.

For the nut missing detection, we utilized a YOLOv9 object detection model to identify missing nuts on the wheels of Pallet cars. By manually reviewing videos to pinpoint occurrences of missing nuts, we cropped and converted these segments to images for annotation. The model was trained and tested, and timestamps of detections were stored in PostgreSQL. A function was implemented to reduce false positives by recording detections every ninth consecutive frame, achieving high accuracy in real-time video display with OpenCV integration.

The wheel count detection involved drawing a specific line in the video frame to count wheels as they crossed it. Using a YOLOv8 model, we labeled and annotated images of Pallet car wheels, trained and tested the model, and developed a function to increment the wheel count and store timestamps in PostgreSQL. The solution was displayed in real-time with OpenCV, ensuring accurate counting and timestamp logging.

For the OCR task, we focused on detecting and enhancing the visibility of numbers on electrical panels. After annotating the target areas on images using LabelImg and training a YOLOv9 model, we implemented OCR to recognize and display numbers above the detected regions, addressing visibility issues on electrical panels.

ABSTRACT

Lastly, the electrical panel detection aimed to identify panels in video datasets, even when tilted. Utilizing the YOLOv8 segmentation model and LabelMe for precise annotation with polygons, we trained and tested the model to segment panels accurately. Real-time detection was achieved through OpenCV, enabling effective monitoring of electrical panel conditions.

Overall, the integration of YOLO models with real-time video processing and database management significantly improved the detection accuracy and operational efficiency in industrial maintenance tasks.

***Keywords**: Optical Character Recognition (OCR), Electrical panel detection, PostgreSQL, YOLOv9.*

TABLE OF CONTENTS

Chapter No.	Table of Contents		Page No.
1.	**Introduction**		1-2
	1.1	Literature Survey	3-5
	1.2	Motivation	6
	1.3	Objectives	7
	1.4	Problem Definition	8
2.	**Requirement Analysis**		9
	2.1	System Model	9-11
	2.2	Functional Requirements	12
	2.3	Non-Functional Requirements	13
	2.4	Database Requirements	13
	2.5	Software Requirements	14
3.	**System Design**		15
	3.1	Architecture Design	15-17
	3.2	Data Flow Design	18-23
4.	**Implementation**		24-29
5.	**Testing**		30-33
6.	**Results and Discussion**		34-35
7.	**Conclusion and Future Scope**		36
8.	**References**		37-39

LIST OF FIGURES

Figure No.	Page No.
Figure 1 System Model	11
Figure 2 Architecture Design	17
Figure 3 Data Flow Diagram	23
Figure 4: Realtime detection of nut missing on wheels of pallet car	25
Figure 5: Storing timestamp in pgAdmin 4 when nut missing detection occurs	25
Figure 6: Realtime simulation of wheel count when it crosses a certain line	27
Figure 7: Storing Timestamp and wheel count of wheels when they cross a certain line	27

Chapter 1

Introduction

Advancements in computer vision have significantly enhanced industrial maintenance and monitoring processes. This project leverages these advancements to develop a robust system for real-time video analysis, focusing on Sinter Machine Pallet cars and electrical panels. Utilizing state-of-the-art object detection models, specifically YOLOv8 and YOLOv9, the system automates the detection of missing nuts on pallet car wheels, counts wheels, recognizes characters on electrical panels, and identifies electrical panels in various orientations.

For detecting missing nuts on Pallet car wheels, videos were manually reviewed to identify instances of missing nuts. Relevant segments were cropped and converted to images for annotation, followed by training and testing with the YOLOv9 model. A function was implemented to record detections at every ninth consecutive frame, reducing false positives. The system stored detection timestamps in a PostgreSQL database and saved relevant images locally. Real-time video processing and display were achieved using OpenCV.

In the wheel counting process, a line was drawn on the video frame to count wheels as they crossed it. The YOLOv8 model was employed to label and annotate images of Pallet car wheels. The trained model incremented the wheel count and stored timestamps in a PostgreSQL database each time a wheel crossed the line. This data, along with real-time display using OpenCV, ensured accurate and efficient wheel counting.

For optical character recognition (OCR) on electrical panels, areas containing numbers were annotated, and the YOLOv9 model was used for detection. Post-detection, OCR was applied to these regions to recognize and display the numbers above the detected areas, enhancing visibility and addressing the issue of poorly visible numbers on electrical panels.

Lastly, the detection of electrical panels involved segmenting the panels, even when tilted. Using the YOLOv8 segmentation model and LabelMe for precise annotation, the system was trained to accurately segment and detect electrical panels. Real-time detection capabilities were implemented using OpenCV.

Overall, the integration of YOLO models with real-time video processing and database management significantly improved the accuracy and efficiency of industrial maintenance operations. This project highlights the potential of advanced computer vision techniques in automating and enhancing industrial inspection processes.

1.1 Literature Survey

Learning-based approaches for classifying text regions typically rely on features like Histogram of Gradients (HOG), edge density, or contrast, which are computationally expensive and language-dependent. In contrast, heuristic-based methods exploit text properties such as connected components or stroke width transform but are sensitive to changing conditions. Previous studies have highlighted the saliency of scene text, leading to the development of a new visual attention model to enhance text detection. Text in natural scenes not only adds semantic information but also aids in object recognition when visual cues are insufficient. The proposed method in this paper introduces a novel algorithm for text detection in natural images based on saliency cues and context fusion, ultimately improving object recognition with the assistance of recognized text [1][2].

The paper delves into the advancements of YOLOv8, the latest iteration of the YOLO model, emphasizing its enhanced real-time object detection capabilities compared to other popular models like Faster R-CNN, SSD, and RetinaNet. YOLOv8 has demonstrated superior performance over YOLOv5 in domain-specific tasks, achieving a mean average precision (mAP) score of 80.2 on Roboflow 100, showcasing its efficacy for specialized applications. By integrating cutting-edge techniques such as attention modules, self- attention mechanisms, spatial pyramid pooling, and deformable convolutions, YOLOv8 has significantly improved object detection accuracy, speed, and efficiency. The evolution of deep learning and deep convolutional neural networks has played a pivotal role in revolutionizing object detection, enabling the development of more precise and robust methods in computer vision. This review underscores the significance of YOLOv8 in advancing real-time object detection frameworks and its versatility across diverse domains [5][6][7].

The research paper delves into the evolution of YOLO, from its inception with YOLOv1 to the latest iteration, YOLOv8, and beyond. It meticulously examines the advancements made in each version, focusing on changes in network architecture, training methodologies, and performance enhancements. YOLOv4, for instance, introduced notable improvements like pre-training with ImageNet and transitioning to a fully convolutional design. The progression of YOLO models reveals a shift from anchor-based approaches in earlier versions to anchor-less strategies in recent models like YOLOX. Additionally, the migration to PyTorch from Darknet and exploration of frameworks like PaddlePaddle have influenced the trajectory of YOLO models. The paper not only highlights the technical enhancements in each iteration but also underscores the crucial balance between speed and accuracy, emphasizing the need to select the most appropriate model based on specific application requirements. Furthermore, it outlines potential research avenues for advancing real-time object detection systems in the future [9][10][11].

The research paper "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information" by Wang et al. introduces the concept of programmable gradient information (PGI) to address information loss in deep networks, focusing on data transmission challenges like information bottleneck and reversible functions. The study presents the Generalized Efficient Layer Aggregation Network (GELAN) architecture, showcasing superior results on lightweight models and improved parameter utilization compared to state-of-the-art methods. Through ablation studies, the authors demonstrate the effectiveness of PGI in enhancing object detection performance across different model sizes, from small to extended models. The paper also emphasizes the computational efficiency of YOLOv9 compared to existing methods, highlighting its competitiveness in terms of computational complexity and model performance. This work contributes to advancing lightweight network architectures and improving gradient information utilization for various deep learning tasks [13][14][15][16].

YOLOv9, an advanced object detection model, builds upon the success of YOLOv8 by enhancing accuracy and efficiency. It introduces additional complexity compared to its predecessor, resulting in superior performance. YOLOv8 and YOLOv9 both faced challenges in misclassifying certain classes, such as 'Background' and 'Healthy,' with YOLOv9 showing a slightly higher misclassification rate in the 'Healthy' class. YOLOv9 also exhibited a higher percentage of false negatives in the 'SunScaled' category compared to YOLOv8. The experimental results analysis of training both YOLO algorithms on a tomato disease detection dataset evaluated their performance based on precision, recall, and mean Average Precision (mAP). This literature survey highlights the advancements in object detection models like YOLOv9, their challenges in misclassification, and the evaluation metrics used to assess their performance [17][18][19][20].

1.2 Motivation

My journey in computer vision and machine learning has been marked by a relentless pursuit of excellence and innovation, driving me to solve complex real-world problems through advanced technological solutions. In the "Nut Missing Detection" project, I utilized the YOLOv9 object detection model to identify missing nuts on pallet car wheels, meticulously reviewing and annotating videos to train the model with high accuracy. Implementing a strategic function to record timestamps of missing nuts at every ninth consecutive frame significantly reduced false positives, showcasing my ability to enhance detection reliability. Integrating OpenCV for real-time display and PostgreSQL for data storage demonstrated my proficiency in handling complex datasets and real-time processing. In the "Wheel Count Detection" project, I applied the YOLOv8 model to detect and count pallet car wheels as they crossed a predefined line, developing a system that displayed the wheel count and recorded timestamps and images efficiently. This project highlighted my skills in creating robust real- time detection systems and effective data management. The "Optical Character Recognition (OCR) on Electrical Panels" project involved detecting and reading numbers on electrical panels using the YOLOv9 model and LabelImg for precise annotation, enhancing visibility by overlaying detected numbers. This solution addressed the issue of unclear visibility and demonstrated my ability to integrate detection and recognition technologies seamlessly. The "Electrical Panel Detection" project required handling datasets with panels at various angles, utilizing the YOLOv8 segmentation model and labelme tool for accurate annotation to develop a system for real-time detection and segmentation of electrical panels. These projects underscore my dedication to advancing in the field of computer vision, with a strong focus on solving diverse challenges, optimizing detection accuracy, and managing data efficiently. My commitment to pushing the boundaries of technology and making meaningful contributions to the field drives my passion for continuous learning and innovation.

1.3 Objectives

1) Integrate OpenCV with YOLO models for real-time video processing and display of detection results.
2) Utilize PostgreSQL for efficient storage and retrieval of detection timestamps and image data.
3) Use LabelImg and labelme for precise annotation of video frames, addressing complex detection tasks.
4) Apply OCR technology to enhance the visibility of numbers on electrical panels by overlaying detected numbers.
5) Develop and implement YOLOv9 and YOLOv8 models to accurately detect missing nuts and count wheels in real-time.

1.4 Problem Statement:

Manual inspection of pallet cars for missing nuts and accurate wheel counts is inefficient and prone to errors, while poor visibility of numbers on electrical panels complicates maintenance. This project aims to develop automated, real-time detection systems using YOLO models and OpenCV, with efficient data management via PostgreSQL, to enhance operational efficiency and reliability.

Chapter 2

Requirement Analysis

2.1 System Model

The system model consists of the following components:

1. Video Input:

The process begins with the video input of the Sinter Machine Pallet cars and electrical panels. These videos serve as the raw data for the various detection tasks to be performed.

2. Video Processing (OpenCV):

The input videos are processed using OpenCV to extract frames. This pre-processing step is crucial for preparing the data for subsequent detection tasks, making it easier to apply object detection algorithms on individual frames.

3. Nut Detection (YOLOv9):

This module detects missing nuts on the wheels of the pallet car. The YOLOv9 object detection model is trained on annotated images to identify instances of missing nuts. To improve accuracy and reduce false positives, nut missing detections are refined by checking every 9th frame.

4. Wheel Count Detection (YOLOv8):

This module counts the wheels as they cross a specified line. Annotated images with the class name "wheel" are used to train the YOLOv8 model. The system increments the wheel count each time a wheel crosses the predefined line and displays the counts in real-time.

5. Electrical Panel Detection (YOLOv8):

This module detects and segments electrical panels from the video frames. The task involves labelling and annotating images with polygons to accurately detect electrical panels, even when they are tilted. The YOLOv8 segmentation model uses these annotations to segment the panels using different colored masks.

6. OCR on Panels (YOLOv9 + OCR):

OCR is performed on detected areas of electrical panels to enhance the visibility of numbers. After detecting these regions using YOLOv9, OCR is applied to read the numbers and print them above the detected areas, making them more visible.

7. Timestamps and Data Storage:

The timestamps of detections and counts are stored in a PostgreSQL database. This database maintains a comprehensive log of all detection events. Additionally, all detected images are stored locally for further review and validation.

8. Real-time Display (OpenCV):

All detections and counts are displayed in real-time using OpenCV. This feature provides immediate visual feedback, allowing for quick decision-making and validation of detection results.

This structured approach ensures accurate detection and efficient data handling for the monitoring and inspection tasks related to Sinter Machine Pallet cars and electrical panels. By integrating advanced object detection models, OCR technology, and robust data storage solutions, the system achieves high accuracy and reliability in real-time detection and analysis.

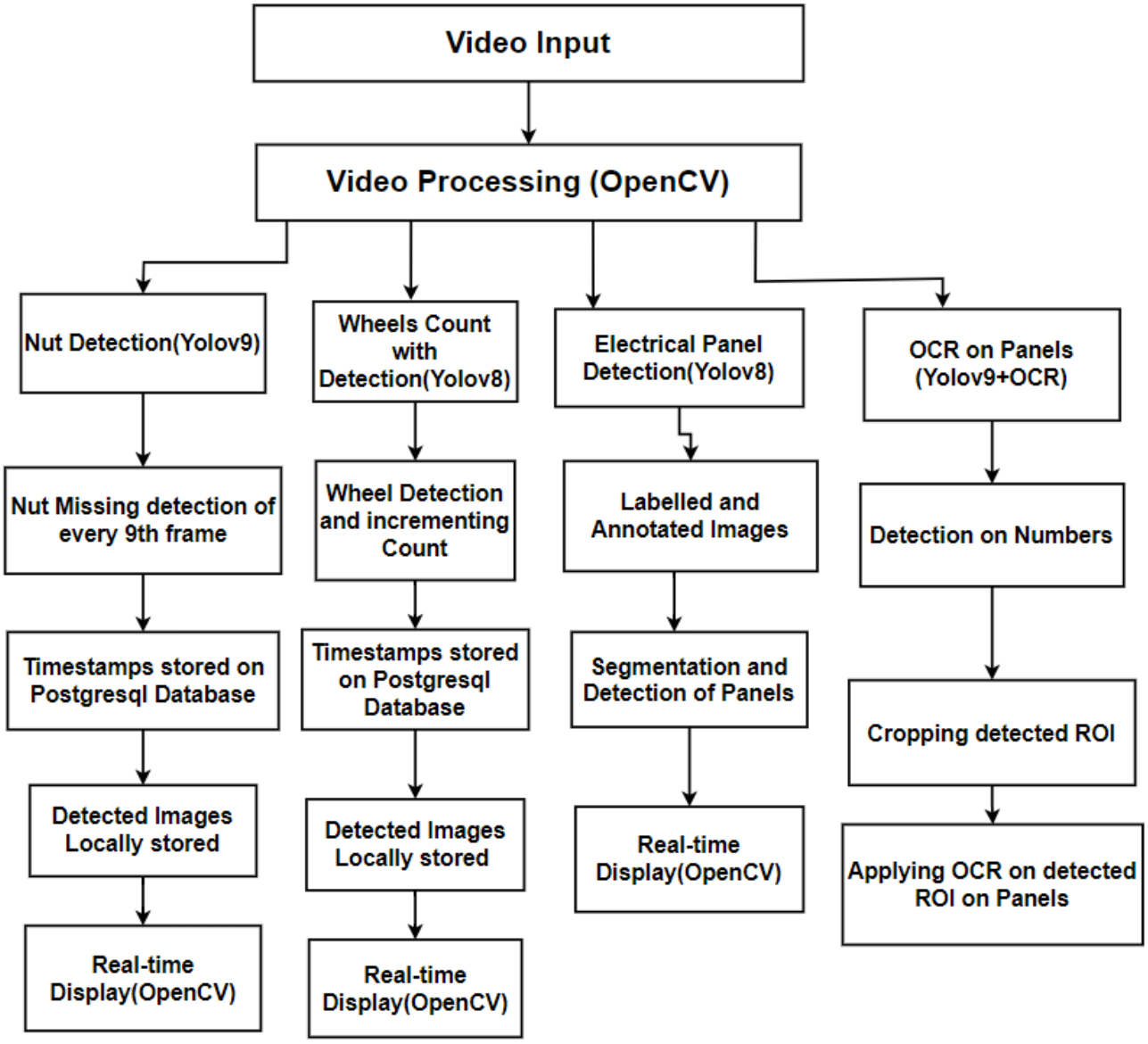


Figure 1 System Model

2.2 Functional Requirements

1) Detect missing nuts on pallet car wheels using YOLOv9 and save timestamps and images of detections.
2) Count and display the number of wheels crossing a specified line in real -time using YOLOv8.
3) Detect and enhance visibility of numbers on electrical panels using YOLOv9 and OCR.
4) Accurately detect and segment electrical panels in various angles using YOLOv8.
5) Implement real-time video processing and display for all detection tasks using OpenCV.

2.3 Non-Functional Requirements

1) The system must process video frames and perform detections in real-time with minimal latency.
2) The detection models must achieve a high level of accuracy, reducing false positives and negatives.
3) The system should be able to handle large datasets and multiple video inputs simultaneously.
4) The system should function consistently without crashes or significant downtime.
5) The interface for real-time display and data annotation should be intuitive and user-friendly.

2.4 Database Requirements

1) PostgreSQL or pgAdmin 4 database for storing nut detection timestamps and count of wheels and wheels detection timestamp.
2) Local storage for saving images of nut missing detections and also local storage for saving images when a wheel passes the line and count is incremented.

2.5 Software Requirements

1) PostgreSQL or pgAdmin 4
2) Visual Studio Code
3) labelImg and labelme
4) Anaconda Prompt (anaconda3)
5) CUDA, works only with inbuilt GPU support in laptop for model training and testing purpose.

Chapter 3

System Design

In designing the system for the specified projects, a modular and scalable approach is paramount. Each task, be it nut missing detection, wheel counting, optical character recognition (OCR), or electrical panel detection, necessitates seamless integration of multiple components. This integration includes video processing, employing object detection models such as YOLOv9 and YOLOv8, database management utilizing PostgreSQL or pgAdmin 4, and real-time display facilitated by OpenCV. A well-structured system should emphasize real- time processing efficiency, ensuring accurate detection, and robust database storage for timestamps, counts, and OCR results. Furthermore, it should be flexible enough to accommodate future enhancements or modifications while upholding scalability and performance standards.

3.1 Architecture Design

The architecture design for the project starts with the initial step of receiving a video dataset related to the Sinter Machine. In the case of detecting missing nuts on the pallet car wheels, the process begins with manually reviewing the videos to identify segments where nuts are missing. These specific video portions are cropped and converted into images, which are then annotated and labelled with the class name "nut missing." These annotated images serve as the training data for the YOLOv9 object detection model. The trained model is tested on the original video dataset to ensure its efficacy. For real-time detection, OpenCV is integrated with the YOLOv9 model, allowing the video to display with real-time detection overlays. Timestamps of detected events are recorded and stored in a PostgreSQL database via pgAdmin4, and images of detections are saved locally.

To minimize false positives, a function is implemented to save timestamps only if the detection is confirmed across every 9th consecutive frame, enhancing the overall accuracy. Similarly, the architecture for counting wheels involves converting the received video dataset into images and annotating them with the class name "wheel." These labelled images are used to train a YOLOv8 object detection model, which is then tested to ensure accurate detection of wheels. OpenCV is utilized to facilitate real-time video display where a specified line is drawn on the output video. Each time a wheel crosses this line, the count is incremented and displayed in real-time. Timestamps of these crossing events are recorded and stored in PostgreSQL or pgAdmin 4, along with the corresponding images. This setup ensures precise counting and documentation of each wheel crossing event, contributing to high accuracy in the final results.

For the OCR task on electrical panels, the received image dataset is annotated with areas containing numbers, labelled as "number." These annotated images train a YOLOv9 object detection model, which is then tested to validate its performance. Once detections are accurately made, a logic is implemented to perform OCR on the detected areas. The detected sections are cropped, and OCR is applied to recognize and print the numbers above the detected regions in the images. This approach addresses the challenge of improving the visibility and readability of numbers on electrical panels, ensuring clear and accurate identification.

Finally, the architecture for detecting electrical panels involves converting video datasets into images, which are annotated using the LabelMe tool. This tool allows for precise labelling, even for panels at tilted angles, by drawing polygons. The labelled images train a YOLOv8 segmentation model, which segments the panels using colored masks. The model is tested to confirm its accuracy before being applied in real-time video processing with OpenCV. This real-time application ensures that electrical panels are accurately detected and segmented regardless of their orientation, enhancing the robustness and reliability of the detection process.

Throughout all these processes, real-time processing is achieved using OpenCV integrated with the respective YOLO models. Detection results, including timestamps and images, are meticulously stored both locally and in a PostgreSQL database using pgAdmin 4. This comprehensive data management ensures that all detection events are thoroughly documented and accessible for future analysis, contributing to the overall reliability and accuracy of the system. By implementing strategies to reduce false positives and maintain high detection accuracy, the architecture design effectively handles complex detection tasks, providing precise and efficient solutions.

Figure 2 Architecture Diagram

3.2 Data Flow Design:

The data flow diagram illustrates a comprehensive process for several machine learning tasks involving object detection and optical character recognition (OCR). Let's break down the flow and connect it with the provided project details:

1. Start

- The process begins here and branches into different tasks, each corresponding to the distinct project requirements.

2. Process Video Dataset (First Occurrence)

- Nut Missing Task: The video dataset from the Sinter Machine's Pallet car is processed to detect missing nuts on each wheel.
- Manually inspect the videos to identify sections with missing nuts.

3. Manual Inspection

- After manual inspection, determine if there are any missing nuts.
- Decision Point: "Are there any missing nuts?"

Yes Path (If Missing Nuts are Found):

a. Crop and Convert to Images:

- Convert the relevant video parts to images where nuts are missing.

b. Annotate and Label Images:

- Annotate these images with the class name "nut missing".

c. Train YOLOv9 Model:

- Train the YOLOv9 object detection model with the annotated images.

d. Test YOLOv9 Model:

- Test the YOLOv9 model to ensure it can accurately detect missing nuts.

e. Store Timestamp of Missing Nut Detection:

- Use PostgreSQL or pgAdmin 4 to store timestamps of detections.

f. Store Images Locally:

- Save images locally when a missing nut is detected.

g. Display Real-Time Output:

- Use OpenCV integrated with YOLOv9 for real-time video display of detections.

h. Reduce False Detections:

- Implement a function to save detection timestamps every 9th consecutive frame to reduce false positives.

i. Achieve High Accuracy:

- Ensure high accuracy in detecting missing nuts.

j. End:

- Completion of the nut detection task

No Path (If No Missing Nuts are Found):

Proceed to Next Step:

- Move on to the next task or step in the process.

4. **Process Video Dataset (Second Occurrence)**

• Wheel Count Task: Detect and count wheels of the pallet car when they pass a specified line.

• Process the video dataset for wheel detection.

5. **Label and Annotate Images:**

- Annotate images with the class name "wheel".

6. **Train YOLOv8 Model:**

- Train the YOLOv8 model for wheel detection.

7. **Test YOLOv8 Model:**

- Test the model to verify its accuracy in detecting wheels.

8. **Draw Line on Output Video:**

- Draw a line on the video at a specific coordinate to count wheels crossing it.

9. **Increment Wheel Count:**

- Increment the wheel count each time a wheel crosses the line.

10. **Store Timestamp of Wheel Crossing Line:**

- Store timestamps of each wheel crossing in PostgreSQL/pgAdmin 4.

11. **Store Images Locally:**

- Save images locally when a wheel crosses the line.

12. **Store Wheel Count:**
    - Save the wheel count in the database.

13. **Display Real-Time Output:**
    - Use OpenCV integrated with YOLOv8 for real-time display.

14. **Achieve High Accuracy:**
    - Ensure the wheel count task achieves high accuracy.

15. **End:**
    - Completion of the wheel counting task.

16. **Process Image Dataset**
    - OCR Task: Detect specific areas on electrical panels for OCR.
    - Convert video frames or images of electrical panels to individual images.

17. **Label and Annotate Images:**
    - Annotate images with class name "number" on areas containing numbers.

18. **Train YOLOv9 Model:**
    - Train YOLOv9 for detecting numbers on panels.

19. **Test YOLOv9 Model:**
    - Test the model to ensure accurate detection of numbers.

20. **Perform OCR on Detected Area:**
    - Crop the detected number areas and perform OCR to read and print numbers.

21. Print OCR Results:

- Print detected numbers above the ROI for clarity.

22. End:

- Completion of the OCR task.

23. Process Video Dataset (Third Occurrence)

- Electrical Panel Detection Task: Detect electrical panels in video datasets.
- Convert video frames to images for panel detection.

24. Label and Annotate Images:

- Annotate images with polygons using labelme for tilted angles.

25. Train YOLOv8 Segmentation Model:

- Train YOLOv8 segmentation model for detecting and segmenting panels.

26. Test YOLOv8 Model:

- Test the model for accurate panel detection.

27. Store Images Locally:

- Save segmented images locally.

28. Display Real-Time Output:

- Use OpenCV integrated with YOLOv8 for real-time panel detection.

29. Achieve High Accuracy:

- Ensure high accuracy in detecting electrical panels.

30. End:

- Completion of the electrical panel detection task.

The below Data Flow diagram visualizes the workflow for each task, from data preprocessing to real-time implementation and accuracy enhancement. Each project involves processing datasets, training/testing models, annotating images, real-time display, and storing relevant information in a database, ensuring high accuracy and reducing false detections. The integration of OpenCV with YOLO models (YOLOv8 and YOLOv9) plays a crucial role in achieving the desired outcomes.

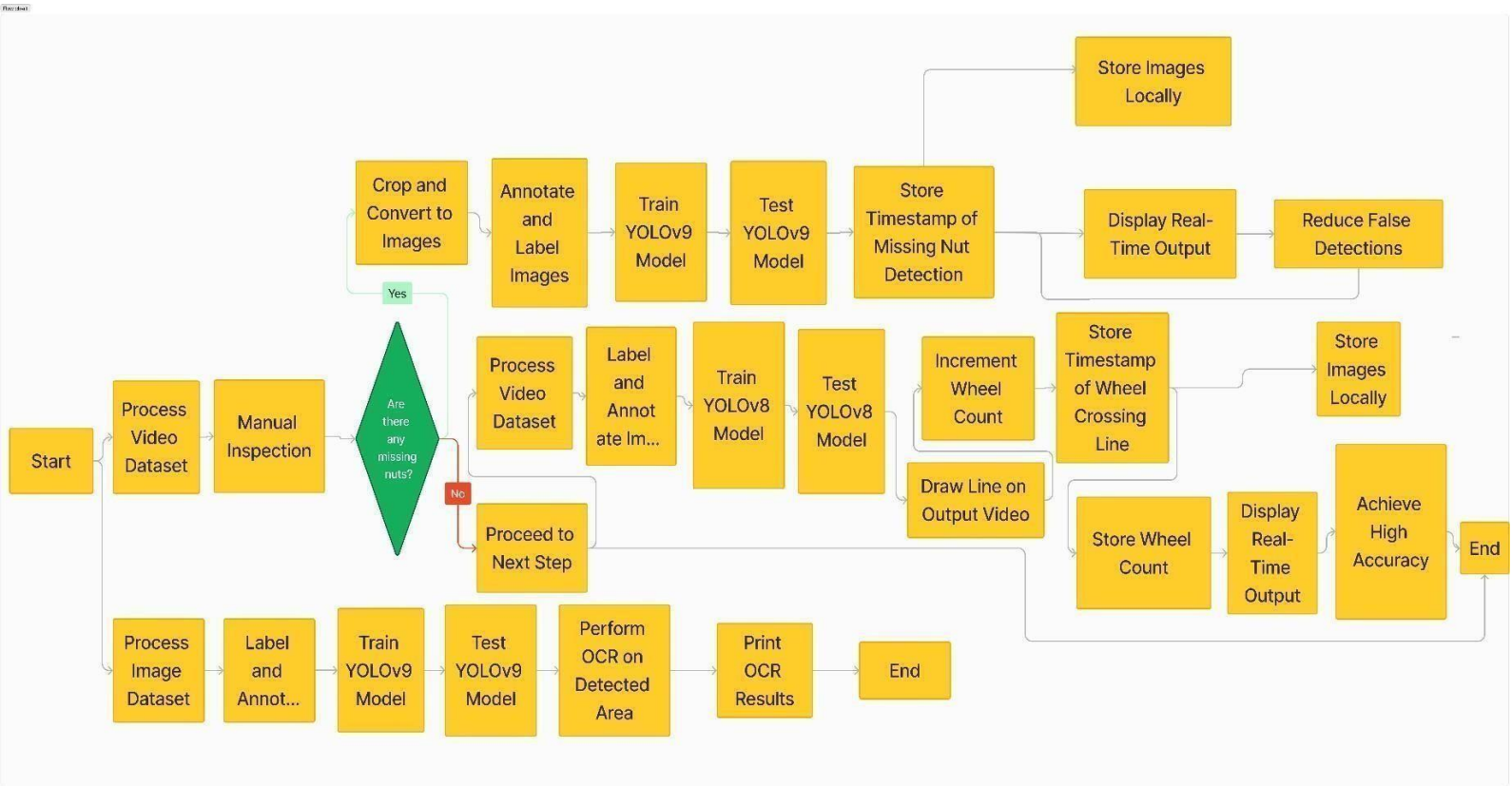


Figure 3 Data Flow Diagram

Chapter 4

Implementation

1) Nut Missing Detection

The nut missing detection involved analyzing video datasets related to a Sinter Machine to identify missing nuts on the wheels of a pallet car. Initially, the videos were manually reviewed to locate frames where nuts were missing. These segments were then cropped and converted into individual images. Using an annotation tool, these images were labeled with the class name "nut missing." The YOLOv9 object detection model was employed to train and test the dataset, ensuring the model could accurately detect missing nuts in various frames.

To manage the detection timestamps, a function was implemented to store these timestamps in PostgreSQL (pgAdmin 4). This function recorded the exact time when a missing nut was detected. Additionally, images were saved locally in a specified folder whenever a detection occurred. For real-time video display, OpenCV was integrated with the YOLOv9 model. Each time the testing code was executed, the video displayed the detections in real time, providing immediate visual feedback. To reduce false detections, a function was implemented to record nut missing detections at every 9th consecutive frame. This strategy significantly minimized false positives and enhanced the accuracy of the detection results.

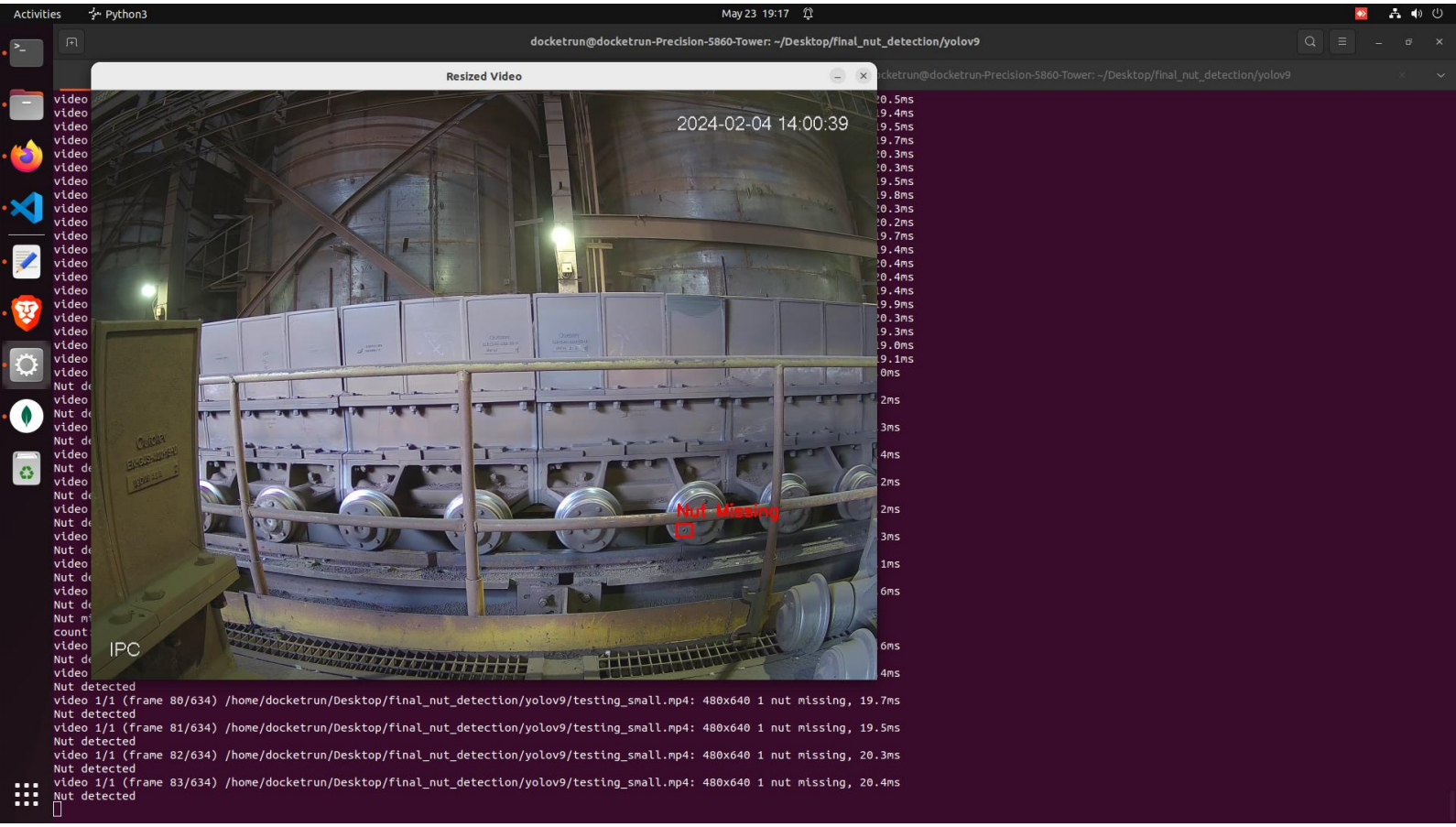
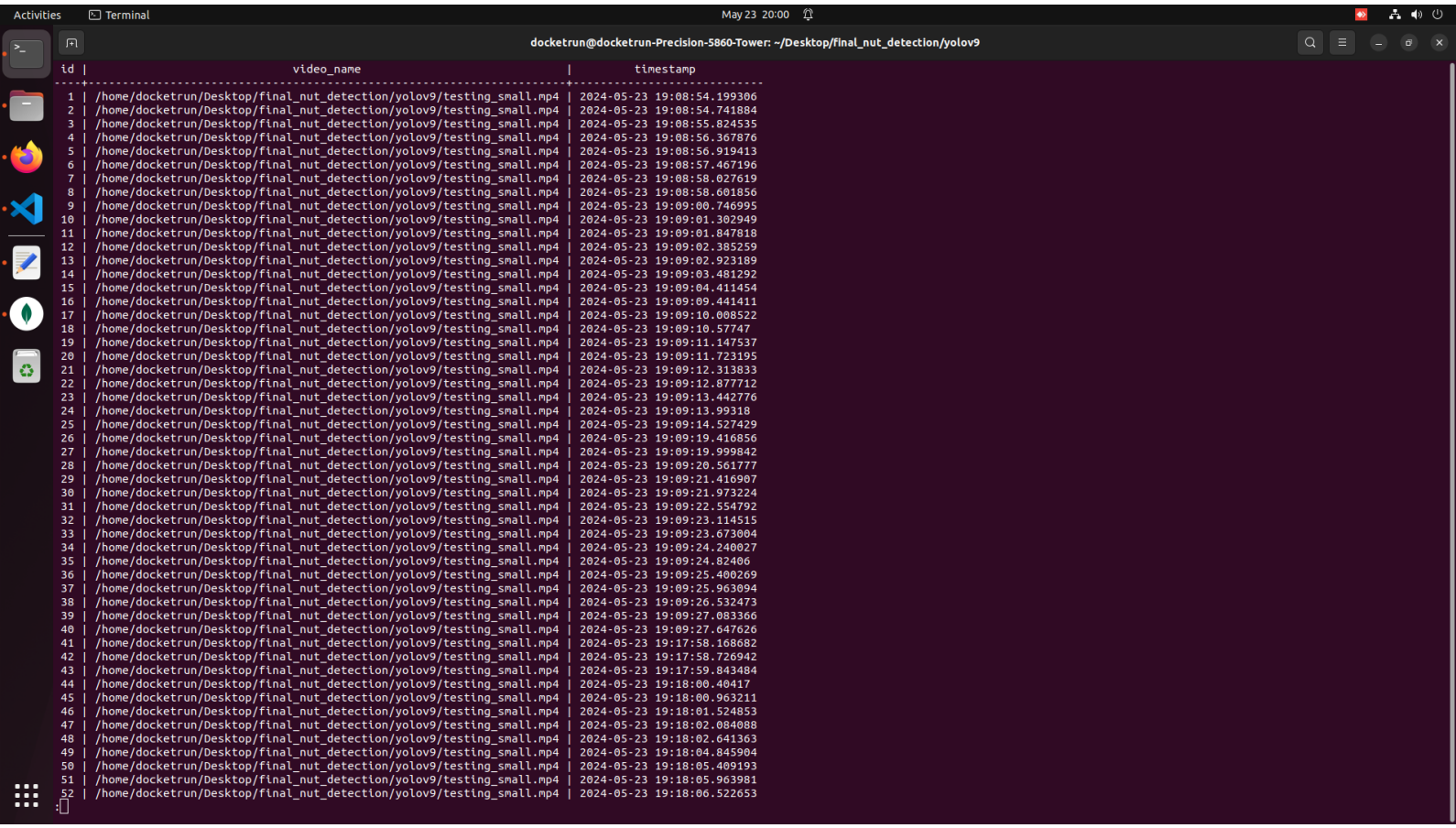


Figure 4: Realtime detection of nut missing on wheels of pallet car



id	video_name	timestamp
1	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:54.199306
2	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:54.741884
3	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:55.824535
4	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:56.367876
5	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:56.919413
6	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:57.467196
7	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:58.027619
8	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:08:58.601856
9	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:00.746995
10	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:01.302949
11	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:01.847818
12	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:02.385259
13	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:02.923189
14	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:03.481292
15	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:04.411454
16	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:09.441411
17	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:10.008522
18	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:10.57747
19	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:11.147537
20	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:11.723195
21	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:12.313833
22	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:12.877712
23	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:13.442776
24	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:13.99318
25	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:14.527429
26	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:19.416856
27	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:19.999842
28	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:20.561777
29	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:21.416907
30	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:21.973224
31	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:22.554792
32	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:23.114515
33	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:23.673004
34	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:24.240027
35	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:24.82406
36	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:25.400269
37	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:25.963094
38	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:26.532473
39	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:27.083366
40	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:09:27.647626
41	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:17:58.168682
42	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:17:58.726942
43	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:17:59.843484
44	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:00.40417
45	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:00.963211
46	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:01.524853
47	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:02.084088
48	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:02.641363
49	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:04.845904
50	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:05.409193
51	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:05.963981
52	/home/docketrun/Desktop/final_nut_detection/yolov9/testing_small.mp4	2024-05-23 19:18:06.522653

Figure 5: Storing timestamp in pgAdmin 4 when nut missing detection occurs

2) Wheel Count Detection

The wheel count detection task aimed to count the number of wheels on a pallet car as it passed a specific line in the video dataset related to a Sinter Machine. The videos were converted into images and each wheel was annotated with the class name "wheel." The YOLOv8 object detection model was then utilized to train and test this dataset, ensuring precise wheel detection.

To count the wheels, code was implemented to draw a line on the output video by specifying coordinate values. A function was developed that incremented the wheel count each time a wheel crossed this line. To store the timestamps of these events, PostgreSQL (pgAdmin 4) was used. Additionally, images were saved locally each time a wheel crossed the line and the count was incremented. For real-time display, OpenCV was integrated with the YOLOv8 model. The real-time output video displayed the wheel count, providing an accurate and immediate visual representation of the counting process.

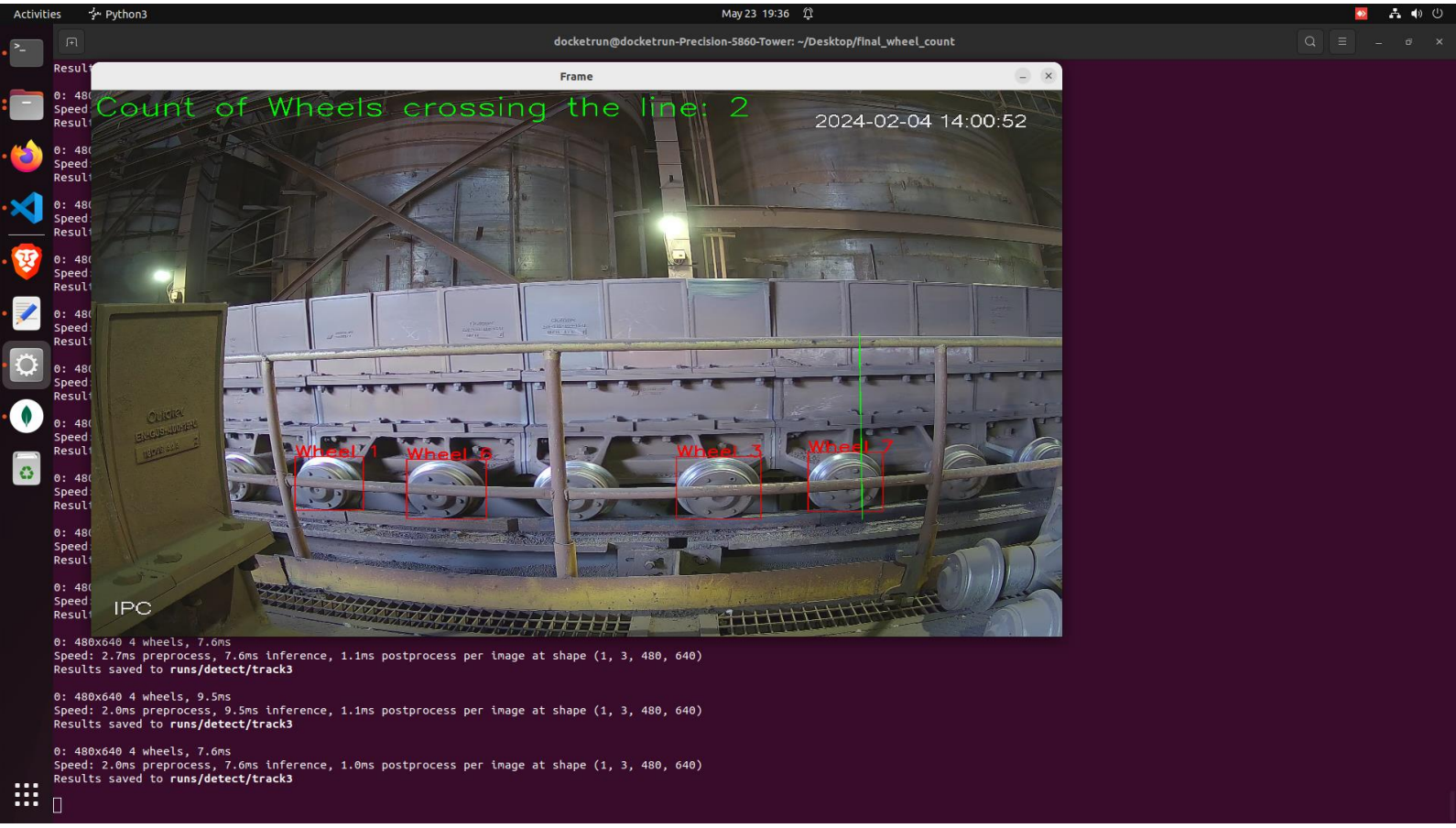


Figure 6: Realtime simulation of wheel count when it crosses a certain line

```
docketrundb=# select * from wheel_crossing2;
 id |    video_name     |         timestamp          | wheels_count
----+-------------------+----------------------------+--------------
  1 | testing_small.mp4 | 2024-05-23 20:08:16.35715  |            1
  2 | testing_small.mp4 | 2024-05-23 20:08:30.922557 |            2
  3 | testing_small.mp4 | 2024-05-23 20:08:45.91368  |            3
(3 rows)
```

Figure 7: Storing Timestamp and wheel count of wheels when they cross a certain line

3) Optical Character Recognition (OCR)

The OCR task involved detecting and recognizing numbers on electrical panels in an image dataset. First, I annotated the images by labelling the areas containing numbers with the class name "number." The YOLOv9 object detection model was then used to train and test the dataset, ensuring accurate detection of numbered areas on the electrical panels.

After achieving correct detection, I integrated OCR into the process. The logic involved cropping the detected regions and applying OCR to recognize the numbers within these areas. The recognized numbers were then printed above the detected regions to enhance their visibility. This approach was particularly useful for making poorly visible numbers on electrical panels more legible, thereby improving the overall readability of the numbers on the panels.

NOTE: Due to confidentiality constraints, the snapshots of the OCR output have been omitted.

4) Electrical Panel Detection

For the electrical panel detection, the goal was to detect various electrical panels in a video dataset, including panels at tilted angles. The videos were converted into images and annotated and labelled using the LabelMe tool. LabelMe was particularly useful because it allowed polygons to be drawn around the panels, accommodating their tilted angles accurately.

The YOLOv8 segmentation model was used for this task. This model was trained to detect and segment the electrical panels, applying colored masks to differentiate between different panels. After training, the model was tested on the video dataset to ensure accurate detection. For real-time detection, the trained model was integrated with OpenCV. When the code was executed, the input video was processed in real time, displaying the segmented electrical panels accurately. This real-time capability ensured immediate visual feedback and validated the model's performance in detecting electrical panels in various orientations.

NOTE: Due to confidentiality constraints, the snapshots of the Electrical Panel Detection task output have been omitted.

Chapter 5

Testing

5.1 Black Box Testing

Black Box Testing, also known as behavioral testing, is a software testing method in which the internal structure, design, and implementation of the item being tested are not known to the tester. This type of testing focuses on examining the functionality of the application without peering into its internal structures or workings. The primary goal is to check whether the software meets the specified requirements and performs as expected. Testers provide input to the software and examine the output, without considering how the software processes the input. This method is highly effective in uncovering discrepancies between the actual output and the expected output.

Techniques and Approaches

Several techniques are utilized in Black Box Testing to ensure comprehensive test coverage. Equivalence Partitioning involves dividing input data into equivalent partitions that are expected to produce similar results, thereby reducing the number of test cases to a manageable level. Boundary Value Analysis focuses on the values at the boundaries of these partitions, as errors frequently occur at these points. Decision Table Testing is another technique that uses tables to represent combinations of inputs and their corresponding outputs, helping to systematically examine all possible scenarios. These techniques help in identifying critical areas that need thorough testing.

Advantages and Applications

Black Box Testing offers numerous advantages, including the ability to test large code segments and the fact that it is not influenced by the developer's perspective, leading to unbiased testing.

It is particularly useful in validating that the software meets user requirements and specifications. This testing method can be applied to various levels of software testing, such as unit, integration, system, and acceptance testing. Additionally, Black Box Testing can be performed by testers without programming knowledge, making it an accessible and versatile testing method for ensuring software quality.

5.2 White Box Testing

White box testing, also known as clear box testing, glass box testing, or structural testing, is a method of software testing that involves the internal workings of an application. Unlike black box testing, which focuses solely on the inputs and outputs of software systems, white box testing delves into the code structure, internal logic, and system architecture. This approach requires testers to have a deep understanding of the code and often involves examining the control flow, data flow, and other internal mechanisms. The primary goal is to verify that the code is functioning as intended, identify any logical errors, and ensure that all paths are tested.

Techniques and Methods

Several techniques are employed in white box testing to thoroughly assess the code. These include:

1. Unit Testing: Testing individual units or components of the software to ensure each part functions correctly.
2. Integration Testing: Ensuring that different modules or components of the software interact correctly.
3. Code Coverage: Measuring the extent to which the source code is tested. Common types of code coverage include statement coverage, branch coverage, and path coverage.
4. Static and Dynamic Analysis: Static analysis involves examining the code without executing it, whereas dynamic analysis involves running the code and observing its behavior.
5. Control Flow Testing: Analyzing the order in which individual statements, instructions, or function calls are executed.

These techniques help in identifying hidden errors, optimizing code performance, and ensuring the reliability and security of the software.

Advantages and Challenges

White box testing offers several advantages. It allows for thorough validation of the code, enabling the detection of hidden errors and vulnerabilities that might not be apparent through black box testing. It also facilitates optimization by identifying inefficient code segments and potential performance bottlenecks. Moreover, white box testing provides detailed coverage reports, which help in ensuring that all parts of the code are tested and verified.

However, white box testing also presents some challenges. It requires a high level of expertise and a deep understanding of the software's internal workings, making it resource- intensive. Additionally, it can be time-consuming, especially for large and complex systems. There is also the risk of missing high-level functionality issues since the focus is primarily on the internal code structure. Despite these challenges, white box testing remains a crucial aspect of software development, providing a comprehensive approach to ensuring code quality and robustness.

5.3 Agile Testing

Agile Testing is a software testing practice that follows the principles of Agile software development, emphasizing iterative progress, collaboration, and flexibility. Unlike traditional testing methodologies, Agile Testing is integrated into the development process from the start. This integration ensures continuous feedback and immediate identification of defects, facilitating a more responsive and adaptive approach to software development. Agile Testing is not a phase that comes after development but is an ongoing activity, where testing and development go hand in hand.

Key Principles and Practices

The key principles of Agile Testing include continuous testing, early and frequent feedback, and a focus on delivering a working product at the end of each iteration. This approach employs various practices such as Test-Driven Development (TDD), where tests are written before the code, and Behavior-Driven Development (BDD), which involves collaboration among developers, testers, and business stakeholders to define the desired behavior of the system. Automation plays a crucial role in Agile Testing, enabling frequent regression testing and ensuring that new changes do not break existing functionality. Tools like Selenium, JUnit, and Cucumber are commonly used to automate tests and support the Agile workflow.

Collaboration and Communication

Agile Testing emphasizes strong collaboration and communication within the team. Testers work closely with developers and business analysts to understand requirements, provide immediate feedback, and adjust testing strategies as the project evolves. Daily stand-up meetings, sprint reviews, and retrospectives are integral practices in Agile that foster an environment of continuous improvement and collective ownership of quality. This close-knit collaboration ensures that issues are detected and resolved quickly, enhancing the overall quality and speed of software delivery. The ultimate goal of Agile Testing is to deliver high-quality software that meets customer needs and adapts swiftly to changing requirements.

Chapter 6

Results and Discussion

The implementation of the nut missing detection yielded significant results. By manually reviewing and annotating the video datasets related to the Sinter Machine, it was possible to accurately identify instances of missing nuts on the wheels of the pallet car. The YOLOv9 object detection model demonstrated high accuracy in detecting missing nuts, especially after reducing false detections by saving timestamps at every 9th consecutive frame. Storing the detection timestamps in PostgreSQL and saving images locally facilitated effective tracking and management of the detections. The real-time video display using OpenCV provided immediate visual feedback, showcasing the model's efficiency in live scenarios. Overall, the approach resulted in a reliable detection system with minimized false positives and enhanced accuracy.

In the wheel count detection project, the YOLOv8 object detection model was employed to count the wheels of the pallet car as it passed a specified line. By converting videos to images and meticulously annotating each wheel, the model achieved high precision in detecting wheels. Implementing a function to increment the wheel count each time a wheel crossed the designated line, and storing these events' timestamps in PostgreSQL, ensured accurate tracking. The real-time display integrated with OpenCV effectively showcased the wheel count as the pallet car moved, demonstrating the model's robustness. The approach successfully delivered a high-accuracy wheel counting system, proving the model's capability in handling real-time data with precise results.

The Optical Character Recognition (OCR) task focused on detecting and recognizing numbers on electrical panels. Using the YOLOv9 object detection model, the detection of numbered areas on the panels was highly accurate. Integrating OCR to crop the detected regions and print the recognized numbers above these areas significantly improved the visibility of the numbers. This enhancement was particularly beneficial in scenarios where the numbers on the panels were not clearly visible. The approach successfully achieved its goal of making the numbers legible, ensuring that the OCR system effectively enhanced the readability of the electrical panels.

In the electrical panel detection project, the YOLOv8 segmentation model was used to detect panels, even those at tilted angles. By converting videos to images and using LabelMe to annotate the panels with precision, the model accurately segmented the panels. The application of colored masks differentiated various panels effectively. Testing the model on video datasets and integrating it with OpenCV for real-time detection demonstrated its efficiency. The system provided immediate visual feedback and validated the model's performance in detecting electrical panels in different orientations. The approach successfully delivered a robust detection system that accurately identified electrical panels, proving the model's capability to handle real-time video data with high precision.

Chapter 7

Conclusion and Future Scope

The implementation of various detection models for the Sinter Machine video datasets resulted in high-accuracy detection systems tailored to specific tasks. The nut missing detection system, leveraging the YOLOv9 model, effectively identified missing nuts on the pallet car wheels with minimized false positives, facilitated by strategic timestamp logging and real-time display using OpenCV. The wheel count detection system, utilizing the YOLOv8 model, accurately counted wheels crossing a specified line, demonstrating robust performance in real-time applications. The OCR system successfully enhanced the readability of numbers on electrical panels, making previously unclear numbers visible by integrating YOLOv9 for detection and OCR for recognition. The electrical panel detection system, powered by the YOLOv8 segmentation model, accurately identified and segmented panels, even those at tilted angles, providing reliable real-time detection. These implementations collectively showcased the effectiveness of YOLO models and OpenCV integration in delivering precise and responsive detection solutions.

The future scope of these detection systems holds significant potential for further enhancements and broader applications. For the nut missing detection system, incorporating advanced machine learning techniques to further reduce false positives and improve detection accuracy across diverse environmental conditions would be beneficial. The wheel count detection system could be expanded to include additional features such as real-time alert mechanisms for maintenance purposes when certain thresholds are met. For the OCR system, integrating more sophisticated OCR algorithms and expanding the dataset to include a wider variety of electrical panels could enhance its robustness and accuracy in diverse scenarios. The electrical panel detection system could benefit from incorporating 3D modeling techniques to improve detection accuracy for panels at extreme angles and in complex environments. Additionally, extending these systems to other industrial applications and integrating them with IoT devices for real-time monitoring and maintenance can significantly enhance operational efficiency and predictive maintenance capabilities.

Chapter 8

References

[1] Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: ICCV. (2011)

[2] Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: CVPR. (2010) 2963–2970.

[3] Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: ACCV'10. (2011) 770–783

[4] Shahab, A., Shafait, F., Dengel, A., Uchida, S.: How salient is scene text? In: IAPR International Workshop on Document Analysis Systems. (2012).

[5] Deng, J., Xuan, X., Wang, W., et al. "A Review of Research on Object Detection Based on Deep Learning." Journal of Physics: Conference Series 1684 (2020): 012028. doi: 10.1088/1742-6596/1684/1/012028.

[6] Bianchini, M., Simic, M., Ghosh, A., Shaw, R. N. In Machine Learning for Robotics Applications. Springer Verlag, Singapore, S.l., 2022.

[7] Agrawal, T., Kirkpatrick, C., Imran, K., Figus, M. "Automatically Detecting Personal Protective Equipment on Persons in Images Using Amazon Rekognition." Amazon, 2020. https://aws.amazon.com/blogs/machine-learning/automatically-detecting-personal- protective-equipment-on-persons-in-images-using-amazon-rekognition/. Accessed April 27, 2023.

[8] Rasouli, A., Tsotsos, J. K. "Autonomous Vehicles That Interact with Pedestrians: A Survey of Theory and Practice." IEEE Transactions on Intelligent Transportation Systems 21 (2019): 900–918. doi: 10.1109/TITS.2019.2901817.

[9] W. Lan, J. Dang, Y. Wang, and S. Wang, "Pedestrian detection based on yolo network model," in 2018 IEEE international conference on mechatronics and automation (ICMA), pp. 1547–1551, IEEE, 2018.

[10] W.-Y. Hsu and W.-Y. Lin, "Adaptive fusion of multi-scale yolo for pedestrian detection," IEEE Access, vol. 9, pp. 110063–110073, 2021.

[11] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, "Yolo-z: Improving small object detection in yolov5 for autonomous vehicles," arXiv preprint arXiv:2112.11798, 2021.

[12] N. M. A. A. Dazlee, S. A. Khalil, S. Abdul-Rahman, and S. Mutalib, "Object detection for autonomous vehicles with sensor-based technology using yolo," International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 1, pp. 129–134, 2022.

[13] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In International Conference on Learning Representations (ICLR), 2022.

[14] Santosh Adhikari, Bikesh Shrestha, Bibek Baiju, Er. Saban Kumar K.C, "Tomato Plant Diseases Detection System using Image Processing", in 1st Kantipur Engineering College, Dhapakhel, Lalitpur Conference Proceedings, 2018.

[15] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[16] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiang-wen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. In International Conference on Learning Representations (ICLR), 2023.

[17] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision (ECCV), pages 213–229, 2020.

[18] J. Redmon & Ali Farhadi. "YOLOv3: An Incremental Improvement. Computer Vision and Pattern Recognition", in Computer Vision and Pattern Recognition, 2018.

[19] Lu Tan, Tianran Huangfu, Liyao Wu & Wenying Chen. "Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification", in BMC Medical Informatics and Decision Making, 2021, vol. 21, p. 324.

[20] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection", in Computer Vision and Pattern Recognition, 2020.