COLLEGE CODE : 3108

COLLEGE NAME : JEPPIAAR ENGINEERING COLLEGE

DEPARTMENT : B. TECH INFORMATION TECHNOLOGY

STUDENT NM-ID : B75C8D82CA9BODE82A0409CD1A422BEF

ROLL NO : 23JEIT206

DATE : 14-5-2005

TECHNOLOGY-PROJECT NAME : QUALITY CONTROL IN MANUFACTURING

SUBMITTED BY,
Your Name and team member names.

MY Nama :
THENMOZHI T

Team Members Nams :
POOVIZHI R
SAKTHI SHREE IYYAPPAN
PRIYADHARSHINI SK
SIDRA FARHEEN S

PHASE 5 : PROJECT DEMONSTRATION AND DOCUMENTATION

Title: AI-Driven Quality Control for Automotive Surface Defects

Abstract:

This project focuses on enhancing quality control in the manufacturing of metal car body panels at AutoForm Industries. The primary challenge is the high rate of surface defects-scratches, dents, and paint inconsistencies-that currently exceed a 2.5% threshold. By integrating computer vision, Statistical Process Control (SPC), and Six Sigma methodologies, this solution aims to reduce the defect rate to below 1%. The final phase integrates AI-based image recognition for real-time defect detection, data analytics for process optimization, and a secure dashboard for monitoring quality metrics. This document outlines the system's architecture, performance metrics, demonstration results, and future development potential.

1. Project Demonstration

Overview: The system will be demonstrated to stakeholders, showing the detection of defects on production lines in real time using AI-powered cameras and how insights guide corrective action.

Demonstration Details:

System Walkthrough: Live demonstration of defect detection from image capture to classification and alert generation.

AI Detection Accuracy: Showcase precision and recall metrics of defect classification models.

Data Integration: Real-time monitoring of production parameters (e.g., temperature, pressure, tooling wear) from PLCs and IoT sensors.

Performance Metrics: Display model performance, system latency, and data processing throughput.

Security & Privacy: Explain data encryption and compliance measures for industrial standards (e.g., ISO/TS 16949).

Outcome:

Stakeholders will observe how AI and automation can enhance quality control, reduce waste, and support predictive maintenance.

2. Project Documentation

Documentation Sections:

System Architecture: Diagrams showing image capture setup, AI model pipeline, and integration with manufacturing execution systems (MES).

Code Documentation: Includes image preprocessing scripts, model training code , and system integration APIs.

User Guide: How to operate the dashboard, respond to alerts, and adjust model sensitivity.

Admin Guide: Maintenance of the AI model, retraining workflow, and monitoring system logs.

Testing Reports: Includes model evaluation, accuracy reports, and Six Sigma performance charts.

Outcomes :
Comprehensive documentation ensures clarity for future upgrades and deployment across multiple plants.

3. Feedback and Final Adjustments

Steps:

Collect feedback from QA engineers and plant managers during pilot testing.

Refine the model to reduce false positives/negatives.

Perform stress testing under various production scenarios.

Outcome:

The system is optimized for real-world factory conditions, ensuring reliable performance.

4. Final Project Report Submission

Report Sections:

Executive Summary: Overview of objectives, solutions, and key outcomes.

Phase Breakdown: From problem identification and data collection to model deployment.

Challenges & Solutions: Examples include misclassification due to lighting variations or surface reflections, solved using data augmentation and improved lighting setups.

Outcomes: Document reduction in defect rate, improved inspection speed, and user adoption.

Outcome: A final report showcasing the system's effectiveness and readiness for scale-up.

5. Project Handover and Future Works

Handover Details:

Future plans include integration with ERP systems, support for additional defect types, and expansion to other manufacturing lines.

Include backup datasets, model training checkpoints, and retraining protocols.

Outcome:

The project is ready for production deployment with clear guidance for future development.

SOURCE CODE AND WORKING OF PROJECT :

main.py

```python
def calculate_process_capability(mean, std_dev, usl=0.86, lsl=0.74):
    """
    Calculate Cp and Cpk for a process given the mean and standard deviation.

    Parameters:
    - mean (float): Process mean
    - std_dev (float): Process standard deviation
    - usl (float): Upper Specification Limit (default: 0.86)
    - lsl (float): Lower Specification Limit (default: 0.74)

    Returns:
    - (Cp, Cpk): Tuple of process capability and process capability index
    """
    cp = (usl - lsl) / (6 * std_dev)
    cpk = min((usl - mean) / (3 * std_dev), (mean - lsl) / (3 * std_dev))

    return cp, cpk

# Example usage
mean_value = 0.80
std_deviation = 0.015

cp, cpk = calculate_process_capability(mean_value, std_deviation)
print(f"Process Capability (Cp): {cp:.2f}")
print(f"Process Capability Index (Cpk): {cpk:.2f}")
```

Output

```
Process Capability (Cp): 1.33
Process Capability Index (Cpk): 1.33

=== Code Execution Successful ===
```

main.py

Share Run

```python
import numpy as np

def generate_samples(mean=0.8, std_dev=0.03, n=100, seed=0):
    np.random.seed(seed)
    return np.random.normal(loc=mean, scale=std_dev, size=n)

def calculate_control_limits(samples):
    mean = np.mean(samples)
    std_dev = np.std(samples)
    ucl = mean + 3 * std_dev
    lcl = mean - 3 * std_dev
    return mean, std_dev, ucl, lcl

def check_control_violations(samples, ucl, lcl):
    violations = [(i, val) for i, val in enumerate(samples) if val > ucl or val < lcl]
    return violations

# Run the process
samples = generate_samples()
mean, std_dev, ucl, lcl = calculate_control_limits(samples)
violations = check_control_violations(samples, ucl, lcl)

# Print results
print(f"Mean: {mean:.4f}")
print(f"Standard Deviation: {std_dev:.4f}")
print(f"UCL: {ucl:.4f}")
print(f"LCL: {lcl:.4f}")
print("\nControl Violations (Index, Value):")
```

Output

```
Mean: 0.8018
Standard Deviation: 0.0302
UCL: 0.8925
LCL: 0.7111

Control Violations (Index, Value):

=== Code Execution Successful ===
```

main.py

Share Run

```python
import numpy as np
np.random.seed(0)

non_defective = np.random.normal(loc=[0.8, 0.6], scale=0.05, size=(50, 2))

defective = np.random.normal(loc=[0.6, 0.4], scale=0.07, size=(50, 2))

X = np.vstack((non_defective, defective))
y = np.array([0]*50 + [1]*50)  # 0 = non-defective, 1 = defective
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
def simple_classifier(sample, threshold=(0.7, 0.5)):
    return int(sample[0] < threshold[0] or sample[1] < threshold[1])


correct = 0
for i in range(len(X_test)):
    prediction = simple_classifier(X_test[i])
    actual = y_test[i]
    correct += (prediction == actual)

accuracy = correct / len(X_test) * 100
print(f"Simple defect detection accuracy: {accuracy:.2f}%")
```

Output

Clear

```
Simple defect detection accuracy: 100.00%

=== Code Execution Successful ===
```