AUTONOMOUS ROBOTICS AND COMPUTER VISION FOR INDUSTRIAL AUTOMATION

End Semester Report

Student Name
Student ID
Discipline

BITS Pilani
Practice School - II

December 2024

Contents

1 Acknowledgments

I would like to express my sincere gratitude to all the individuals and teams who contributed to the success of this project. Special thanks to the T4 team for their collaboration on dataset processing and model training, the T2 team for their assistance with Raspberry Pi integration, and Anirudh for help with hardware installation and protective measures.

I am also grateful to my project supervisors and mentors for their guidance throughout this journey, and to BITS Pilani for providing this valuable learning opportunity through the Practice School program.

2 Executive Summary

This report presents the comprehensive work undertaken during the post-midsem period of the Practice School project focused on autonomous robotics and computer vision for industrial automation. The project involved significant advancements in dataset preparation, hardware setup, software development, and exploration of new automation technologies.

Key achievements include the collection and preprocessing of 82 video samples resulting in 800-1200 high-quality images, successful integration of camera systems with protective measures, development of CNN-based image classification systems, and implementation of robotic navigation algorithms. The work demonstrates practical applications of machine vision in industrial settings and explores emerging areas such as indoor farming automation.

3 Introduction

3.1 Project Overview

The project focuses on developing autonomous robotics systems integrated with computer vision capabilities for industrial automation applications. This interdisciplinary approach combines hardware engineering, software development, and machine learning to create robust automation solutions.

3.2 Objectives

The primary objectives of this project include:

- Development of comprehensive datasets for machine learning applications
- Integration of camera systems with industrial hardware
- Implementation of computer vision algorithms for real-time processing
- Creation of autonomous navigation systems for mobile robots
- Exploration of robotic arm control for precision manipulation
- Investigation of emerging automation applications

3.3 Scope and Applications

The scope encompasses multiple domains including industrial quality control, autonomous navigation, robotic manipulation, and emerging applications in agricultural automation. The work addresses real-world challenges in lighting control, hardware integration, and system reliability.

4 Post-Midsem Activities and Progress

4.1 Dataset Preparation

4.1.1 Video Data Collection

A comprehensive video dataset was collected consisting of 82 video samples, each ranging from 3 to 20 seconds in duration. The collection strategy focused on capturing diverse scenarios including:

- Multiple camera angles (top-view and side-view perspectives)
- Various lighting conditions to ensure robustness
- Different object orientations and positions
- Dynamic scenarios with conveyor belt movement

4.1.2 Image Extraction and Preprocessing

From the collected video dataset, approximately 800-1200 high-quality images were extracted using automated frame extraction techniques. The preprocessing pipeline included:

- Automated cropping to focus on regions of interest
- Resizing to standardized dimensions for model training
- Format conversion for compatibility with machine learning frameworks
- Quality filtering to remove blurred or poorly lit samples

4.1.3 Dataset Collaboration

The preprocessed dataset was shared with the T4 team for advanced processing and model training. This collaboration enabled:

- Specialized expertise in deep learning model development
- Access to computational resources for training
- Iterative feedback for dataset improvement
- Knowledge transfer between teams

4.2 Hardware Setup and Integration

4.2.1 Camera Installation

Successful installation of dual camera systems was completed:

- **Top-view camera**: Mounted for overhead perspective of conveyor operations
- **Side-view camera**: Positioned for profile analysis and dimensional measurements
- **Mounting hardware**: Custom brackets designed for stability and adjustability
- **Cable management**: Organized routing to prevent interference with moving parts

4.2.2 Protective Measures

Comprehensive protective systems were implemented with assistance from Anirudh and the hardware team:

- Protective covers for lighting systems to prevent damage
- Camera enclosures to shield from industrial environment
- Lighting control enclosures to minimize ambient interference
- Safety barriers to protect personnel during operation

4.2.3 Lighting Control Systems

Advanced lighting control was implemented to ensure consistent image quality:

- Enclosed lighting systems to reduce ambient light interference
- Programmable LED arrays for consistent illumination
- Diffusion systems to eliminate harsh shadows
- Automatic brightness adjustment based on ambient conditions

4.2.4 Raspberry Pi Integration

Initial integration of Raspberry Pi systems was completed:

- GPIO interface setup for sensor and actuator control
- Communication protocols established with main control systems
- Real-time image processing capabilities configured
- Collaboration with T2 team for module interfacing expertise

4.3 Software Development

4.3.1 Image Classification System

A comprehensive CNN-based image classification system was developed:

```python
import tensorflow as tf
from tensorflow.keras import layers, models

class IndustrialCNN:
    def __init__(self, input_shape, num_classes):
        self.model = models.Sequential([
            layers.Conv2D(32, (3, 3), activation='relu',
                          input_shape=input_shape),
            layers.MaxPooling2D((2, 2)),
            layers.Conv2D(64, (3, 3), activation='relu'),
            layers.MaxPooling2D((2, 2)),
            layers.Conv2D(64, (3, 3), activation='relu'),
            layers.Flatten(),
            layers.Dense(64, activation='relu'),
            layers.Dense(num_classes, activation='softmax')
        ])

    def compile_model(self):
        self.model.compile(
            optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy']
        )
```

Listing 1: CNN Architecture Implementation

The system incorporates:

- Iterative development based on T4 team feedback
- Real-time inference capabilities
- Robust error handling and logging
- Performance optimization for industrial deployment

4.3.2 YOLOv5 Integration

Comprehensive study and implementation of YOLOv5 for object detection:

- Literature review of relevant research papers
- Implementation of YOLOv5 architecture
- Custom dataset adaptation for industrial objects
- Performance benchmarking against existing solutions

4.3.3 Camera Control Systems

Advanced camera control using the pypylon SDK:

```python
from pypylon import pylon

class CameraController:
    def __init__(self):
        self.camera = pylon.InstantCamera(
            pylon.TlFactory.GetInstance().CreateFirstDevice()
        )

    def configure_camera(self, exposure_time, gain, brightness):
        self.camera.Open()
        self.camera.ExposureTime.SetValue(exposure_time)
        self.camera.Gain.SetValue(gain)
        self.camera.Gamma.SetValue(brightness)

    def capture_image(self):
        self.camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)
        grabResult = self.camera.RetrieveResult(5000)
        if grabResult.GrabSucceeded():
            image = grabResult.Array
            return image
        return None
```

Listing 2: Camera Control Implementation

Features implemented:

- Real-time exposure control for varying lighting conditions
- Programmable zoom and focus adjustment
- Brightness and contrast optimization
- Integration with VSCode development environment

4.3.4 Autonomous Navigation System

Development of sophisticated waypoint navigation with obstacle avoidance:

```python
from enum import Enum, auto

class RobotState(Enum):
    IDLE = auto()
    NAVIGATING = auto()
    AVOIDING_OBSTACLE = auto()
    GOAL_REACHED = auto()

class WaypointNavigator:
    def __init__(self):
        self.state = RobotState.IDLE
        self.current_x = 0.0
        self.current_y = 0.0
        self.current_heading_deg = 0.0
        self.target_waypoints = []

    def navigate_to_waypoint(self, target_x, target_y):
        distance = self.calculate_distance(target_x, target_y)
        angle = self.calculate_bearing(target_x, target_y)

        if distance < self.GOAL_TOLERANCE_M:
            self.state = RobotState.GOAL_REACHED
            return True

        if self.detect_obstacle():
            self.state = RobotState.AVOIDING_OBSTACLE
            self.handle_obstacle_avoidance()
        else:
            self.state = RobotState.NAVIGATING
            self.move_toward_target(angle, distance)

        return False
```

Listing 3: Navigation State Machine

4.3.5 SLAM Algorithm Implementation

Participation in SLAM (Simultaneous Localization and Mapping) algorithm development:

- Integration with Jackal robot platform
- ROS (Robot Operating System) exploration and implementation
- Linux-based development environment setup
- Real-time mapping and localization capabilities

4.4 Robotic Arm Control System

4.4.1 Servo Control Architecture

Implementation of comprehensive servo control for robotic manipulation:

```python
from gpiozero import Servo
import time

class RoboticArmController:
    def __init__(self):
        self.servo_pins = {
            "Base": 18, "Shoulder": 19, "Elbow": 20,
            "WristPitch": 21, "WristRoll": 26, "Gripper": 16
        }
        self.servos = {}
        self.current_angles = {}
        self.initialize_servos()

    def initialize_servos(self):
        for name, pin in self.servo_pins.items():
            try:
                self.servos[name] = Servo(
                    pin,
                    min_pulse_width=0.5/1000,
                    max_pulse_width=2.5/1000
                )
                self.current_angles[name] = 90  # Default position
            except Exception as e:
                print(f"Failed to initialize {name}: {e}")

    def move_to_position(self, target_angles, delay=1.0):
        for servo_name, angle in target_angles.items():
            if servo_name in self.servos:
                self.set_angle(servo_name, angle)
                time.sleep(delay)
```

Listing 4: Servo Control System

4.5 New Prototyping Initiative

4.5.1 Indoor Farming Automation

Initiated development of machine vision-based indoor farming automation:

- Conceptual design for automated plant monitoring systems
- Computer vision algorithms for plant health assessment
- Environmental control integration
- Scalability analysis for commercial applications

5 Technical Challenges and Solutions

5.1 Lighting and Image Quality Issues

5.1.1 Challenge

Significant issues encountered with lighting reflections and image blur during conveyor movement, affecting the quality of captured data for machine learning applications.

5.1.2 Solution Approach

- Implementation of controlled lighting environments with diffusion systems
- Calibration of camera settings for optimal exposure and shutter speed
- Development of image stabilization algorithms
- Creation of protective enclosures to minimize ambient light interference

5.1.3 Results

Achieved substantial improvement in image quality with reduced reflections and motion blur, enabling more reliable dataset collection and processing.

5.2 Robotic Arm Optimization

5.2.1 Challenge

The robotic arm system was operational but not performing at optimal levels, requiring refinement for robust industrial functionality.

5.2.2 Solution Approach

- Comprehensive calibration of servo control parameters
- Implementation of error handling and recovery mechanisms
- Development of smooth trajectory planning algorithms
- Integration of feedback systems for position verification

5.2.3 Current Status

Ongoing optimization with improved precision and reliability, though further refinement is required for full industrial deployment.

5.3 Navigation System Challenges

5.3.1 Challenge

The Jackal robot successfully avoids obstacles but requires enhancement in navigation precision and goal-reaching capabilities.

5.3.2 Solution Approach

- Implementation of advanced odometry tracking
- Development of sophisticated obstacle avoidance algorithms
- Integration of multiple sensor inputs for improved situational awareness
- Fine-tuning of control parameters for various terrain conditions

6 Key Learnings and Skills Developed

6.1 Technical Skills

6.1.1 Hardware Integration

- Hands-on experience with industrial cameras and mounting systems
- Understanding of lighting calibration and environmental control
- Expertise in Raspberry Pi GPIO programming and interfacing
- Knowledge of protective systems design for industrial environments

6.1.2 Software Development

- Proficiency in computer vision algorithms and implementation
- Experience with machine learning frameworks (TensorFlow, PyTorch)
- Understanding of real-time image processing techniques
- Expertise in robotic control systems and state machines

6.1.3 System Integration

- Understanding of hardware-software integration challenges
- Experience with multi-team collaboration and knowledge sharing
- Appreciation for real-world constraints in industrial automation
- Knowledge of system reliability and error handling requirements

6.2 Project Management

- Coordination with multiple teams (T2, T4) for specialized expertise
- Management of hardware installation with technical support staff
- Documentation and knowledge transfer practices
- Timeline management and milestone achievement

6.3 Problem-Solving Approaches

- Systematic debugging of hardware and software issues
- Iterative development and testing methodologies
- Root cause analysis for performance optimization
- Creative solutions for environmental and technical constraints

7 Future Work and Recommendations

7.1 Immediate Improvements

- **Robotic Arm Optimization**: Complete the calibration and fine-tuning process for industrial-grade precision
- **Navigation Enhancement**: Implement advanced path planning algorithms for improved goal-reaching accuracy
- **Lighting System**: Deploy automated adaptive lighting controls for varying environmental conditions
- **Integration Testing**: Conduct comprehensive system-level testing with all components integrated

7.2 Medium-term Developments

- **Machine Learning Enhancement**: Deploy advanced deep learning models for improved object recognition and classification
- **Predictive Maintenance**: Implement condition monitoring systems for proactive maintenance scheduling
- **Human-Robot Interaction**: Develop safe and intuitive interfaces for human operators
- **Performance Optimization**: Implement real-time performance monitoring and optimization systems

7.3 Long-term Vision

- **Full Automation**: Achieve complete autonomous operation with minimal human intervention
- **Scalability**: Design systems for easy replication and deployment across multiple facilities
- **AI Integration**: Incorporate advanced AI decision-making capabilities for adaptive behavior
- **Industry 4.0**: Integrate with broader Industry 4.0 initiatives including IoT and cloud connectivity

8 Conclusion

The post-midsem period has been highly productive, resulting in significant advancements across multiple domains of the autonomous robotics and computer vision project. The comprehensive dataset preparation, successful hardware integration, and sophisticated software development have laid a strong foundation for future industrial automation applications.

Key achievements include the creation of a robust dataset with over 800 high-quality images, successful integration of dual camera systems with protective measures, development of CNN-based classification systems, and implementation of autonomous navigation capabilities. The collaborative approach with specialized teams (T2, T4) has proven effective in leveraging diverse expertise and accelerating development.

The challenges encountered, particularly in lighting control and system optimization, have provided valuable learning experiences and driven innovative solutions. The ongoing work on robotic arm optimization and navigation enhancement demonstrates the iterative nature of complex system development.

The exploration of new applications, such as indoor farming automation, showcases the versatility and potential impact of the developed technologies beyond traditional industrial settings. This forward-looking approach positions the project for continued relevance and impact.

The technical skills developed, ranging from hardware integration to advanced software development, provide a strong foundation for future careers in robotics and automation. The experience of working with real-world constraints and industrial requirements has been particularly valuable in understanding the practical aspects of deploying automation solutions.

Looking forward, the project is well-positioned for continued development and eventual deployment in industrial settings. The systematic approach to problem-solving, emphasis on collaboration, and focus on practical applications ensure that the work will contribute meaningfully to the advancement of autonomous robotics and computer vision in industrial automation.

9 References

References

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[2] Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT Press.

[3] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[4] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[5] Craig, J. J. (2017). Introduction to robotics: mechanics and control. Pearson.

[6] Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.

[7] Groover, M. P. (2014). Automation, production systems, and computer-integrated manufacturing. Pearson.

[8] Upton, E., & Halfacree, G. (2016). Raspberry Pi user guide. John Wiley & Sons.