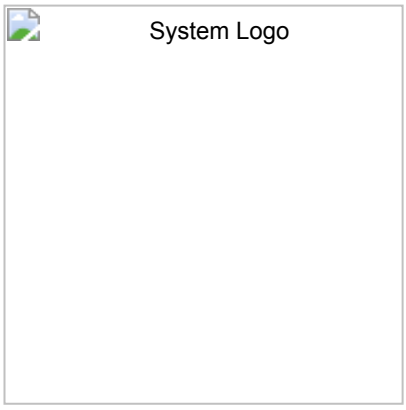
Smart Parking Management System

Technical Documentation

April 15, 2025



Repository: Friendlydevil03/park10 (https://github.com/Friendlydevil03/park10)

Version: 1.0

Last Updated: April 15, 2025

Table of Contents

1. System Overview
2. Architecture
3. Key Components
    1. User Interface (UI)
    2. Detection System
    3. Allocation Engine
    4. Parking Visualizer
    5. Vehicle Detector
4. Installation & Setup
5. Workflow
6. Features
7. System Requirements
8. Dependencies
9. Troubleshooting
10. Future Improvements

1. System Overview

The Smart Parking Management System is a comprehensive software solution designed to efficiently manage parking spaces through computer vision and AI technologies. The system provides real-time monitoring of parking spaces, automated vehicle detection, intelligent parking space allocation, and visual representations of parking status.

The system combines traditional computer vision techniques with machine learning models to offer high accuracy in detecting occupied and free parking spaces. Additionally, it includes an AI-driven allocation engine that optimizes parking space assignments based on various factors such as vehicle size, distance to entrance, and load balancing requirements.

---

2. Architecture

The project follows a modular architecture with clear separation of concerns:

```
park10/
│
├── main.py                 # Application entry point
├── ui/                     # User interface components
│   ├── app.py              # Main application window
│   ├── detection_tab.py    # Vehicle/parking space detection
│   ├── setup_tab.py        # Parking space configuration
│   ├── log_tab.py          # Event logging
│   ├── stats_tab.py        # Statistics display
│   ├── reference_tab.py    # Reference image management
│   └── parking_allocation_tab.py # Space allocation interface
│
├── models/                 # Core functionality models
│   ├── allocation_engine.py  # AI-based parking allocation
│   ├── parking_manager.py    # Parking management core
│   ├── parking_visualizer.py # Visualization components
│   └── vehicle_detector.py   # ML-based vehicle detection
│
├── utils/                  # Utility functions
│   ├── dialogs.py            # Custom dialog windows
│   ├── image_processor.py    # Image processing functions
│   ├── media_paths.py        # File path management
│   ├── resource_manager.py   # Resource management
│   ├── style_config.py       # UI styling configuration
│   ├── tracker_integration.py# YOLO and DeepSORT integration
│   ├── video_utils.py        # Video processing utilities
│   └── window_manager.py     # Window positioning
│
└── config/                 # Configuration storage
    ├── models/               # ML model storage
    └── window_config.json    # Window settings
```

The architecture implements a Model-View-Controller (MVC) pattern:

- **Model**: The `models/` directory contains core business logic
- **View**: The `ui/` directory contains user interface components
- **Controller**: The main application and tab controllers coordinate between models and views

---

3. Key Components

3.1 User Interface (UI)

The UI is built using Tkinter and provides a tabbed interface with the following main sections:

1. **Detection Tab**: Displays video feeds with parking space detection and vehicle counting capabilities.
2. **Setup Tab**: Allows users to configure parking spaces by drawing rectangles on a reference image.
3. **Allocation Tab**: Provides an interactive parking space allocation system with visualization.
4. **Log Tab**: Displays system events and logs.
5. **Stats Tab**: Shows parking statistics and analytics.
6. **Reference Tab**: Manages reference images for different parking layouts.

Key features of the UI include:

- Responsive design that adjusts to window resizing
- Dark and light theme support
- Real-time visualization updates
- Interactive parking space configuration

3.2 Detection System

The detection system utilizes multiple methods to identify vehicles and monitor parking spaces:

1. **Traditional Computer Vision**:
    - Uses image processing techniques like contour detection
    - Applies background subtraction for movement detection
    - Implements threshold-based occupancy determination
2. **Machine Learning Detection**:
    - Integrates with multiple ML models:
        - FasterRCNN with ResNet50 backbone
        - YOLOv8 object detection
        - OpenCV DNN as a fallback option
    - Provides configurable confidence thresholds
3. **Vehicle Tracking**:
    - Implements DeepSORT algorithm for vehicle tracking
    - Counts vehicles passing through a defined line

The system supports both image-based detection for static parking spaces and video-based detection for dynamic monitoring.

3.3 Allocation Engine

The `ParkingAllocationEngine` in `models/allocation_engine.py` is an AI-driven system that optimizes parking space allocation using machine learning:

1. **Technology**: Uses XGBoost classifier to determine optimal parking spaces

2. **Features considered**:
    - Distance to entrance
    - Time since last occupied
    - Vehicle size
    - Section occupancy rate
    - Load balancing requirements
3. **Adaptive Learning**:
    - Collects feedback on allocation effectiveness
    - Periodically retrains the model with new data
    - Improves allocation recommendations over time
4. **Load Balancing**:
    - Distributes vehicles evenly across parking sections
    - Prevents congestion in specific areas
    - Configurable weighting between convenience and load balancing

3.4 Parking Visualizer

The `ParkingVisualizer` in `models/parking_visualizer.py` provides visual representation of the parking lot:

1. **Features**:
    - Real-time visualization of parking space status
    - Color-coding (green: free, red: occupied, orange: partially occupied)
    - Graphical representation of individual and group spaces
    - Section-based visualization
    - Statistics overlay
2. **Technologies**:
    - Matplotlib for graphical rendering
    - Integration with Tkinter using FigureCanvasTkAgg
    - Responsive design adapting to different screen sizes

3.5 Vehicle Detector

The `VehicleDetector` in `models/vehicle_detector.py` implements machine learning-based vehicle detection:

1. **Features**:
    - Multi-model support (FasterRCNN, YOLOv8, OpenCV DNN)
    - Graceful degradation with fallback models
    - Performance optimization with caching
    - Configurable confidence thresholds

2. **Detection Pipeline**:
    - Image preprocessing
    - Model inference
    - Post-processing of detections
    - Filtering for vehicle classes

---

4. Installation & Setup

Prerequisites

- Python 3.8 or higher
- Tkinter
- OpenCV
- NumPy
- Matplotlib
- XGBoost
- PyTorch (for ML-based detection)
- Ultralytics (for YOLOv8)

Installation Steps

1. Clone the repository:

```
git clone https://github.com/Friendlydevil03/park10.git
cd park10
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Run the application:

```
python main.py
```

Initial Configuration

1. On first launch, the system will create necessary directories:
    - `config/` - For configuration data
    - `logs/` - For system logs

2. To set up parking spaces:
    - Go to the "Setup" tab
    - Load a reference image of your parking lot
    - Draw rectangles around parking spaces using left-click and drag
    - Save the configuration using the "Save" button

---

5. Workflow

The typical workflow for using the Smart Parking Management System is as follows:

1. **Initial Setup**:
    - Configure parking spaces in the Setup tab
    - Associate reference images with video sources
    - Adjust detection thresholds if needed
2. **Monitoring**:
    - Switch to Detection tab to monitor parking spaces in real-time
    - View occupancy statistics in the Stats tab
    - Check logs for events and alerts
3. **Parking Allocation**:
    - Use the Allocation tab to assign vehicles to parking spaces
    - Enable auto-allocation for automatic assignments
    - Adjust load balancing parameters as needed
4. **Analysis**:
    - Review statistics and trends in the Stats tab
    - Export logs for further analysis
    - Optimize space utilization based on data

---

6. Features

Parking Space Detection

- Draw and configure parking spaces visually
- Automatic detection of free and occupied spaces
- Support for individual spaces and group spaces
- Customizable detection thresholds

Vehicle Detection and Counting

- Real-time vehicle detection using computer vision
- Vehicle counting for entrance/exit monitoring
- Multi-model support for different environments
- Detection confidence adjustment

Intelligent Space Allocation

- AI-powered parking space allocation
- Load balancing across parking sections
- Preference-based allocation (section, proximity)
- Vehicle size consideration

Visualization

- Interactive visualization of parking status
- Color-coded occupancy indicators
- Group space visualization
- Real-time updates

Simulation and Testing

- Simulate vehicle arrivals and departures
- Test allocation strategies
- Visualize allocation results
- Reset simulation for different scenarios

---

7. System Requirements

Minimum Requirements

- **OS**: Windows 10, macOS 10.14, Ubuntu 18.04 or higher
- **CPU**: Intel Core i3 or equivalent
- **RAM**: 4GB
- **Storage**: 500MB free disk space
- **Display**: 1280x720 resolution

Recommended Requirements

- **OS**: Windows 11, macOS 12, Ubuntu 20.04 or higher
- **CPU**: Intel Core i5 or equivalent
- **RAM**: 8GB

- **GPU**: NVIDIA GPU with CUDA support (for ML acceleration)
- **Storage**: 2GB free disk space
- **Display**: 1920x1080 resolution

---

8. Dependencies

The system relies on the following key libraries:

- **UI and Visualization**:
  - Tkinter: UI framework
  - Matplotlib: Data visualization
  - PIL/Pillow: Image processing
- **Computer Vision**:
  - OpenCV: Core computer vision functionality
  - NumPy: Numerical operations
- **Machine Learning**:
  - PyTorch: Deep learning framework
  - XGBoost: Gradient boosting
  - Ultralytics: YOLOv8 implementation
- **Utilities**:
  - datetime: Time management
  - threading: Multi-threading support
  - queue: Thread-safe communication
  - json: Configuration storage

---

9. Troubleshooting

Common Issues and Solutions

Issue: Application fails to start

- **Solution**: Check Python version (3.8+ required)
- **Solution**: Verify all dependencies are installed
- **Solution**: Check for file permission issues

Issue: No video sources detected

- **Solution**: Check camera connections

- **Solution**: Verify video files exist in the correct directory
- **Solution**: Check video codec compatibility

Issue: Poor detection accuracy

- **Solution**: Adjust detection threshold
- **Solution**: Improve lighting conditions in the camera view
- **Solution**: Use ML detection for better accuracy

Issue: UI elements appear misaligned

- **Solution**: Use a supported screen resolution (1280x720 minimum)
- **Solution**: Restart the application
- **Solution**: Clear the window configuration file

Issue: Memory usage increases over time

- **Solution**: Limit the number of video sources
- **Solution**: Reduce video resolution
- **Solution**: Restart the application periodically

---

10. Future Improvements

The development roadmap includes the following planned enhancements:

1. **Enhanced Machine Learning**:
    - Integration of more advanced detection models
    - Custom model training for specific parking environments
    - Anomaly detection for unusual events
2. **Cloud Integration**:
    - Remote monitoring capabilities
    - Cloud storage for historical data
    - Mobile application for remote access
3. **Advanced Analytics**:
    - Predictive parking availability
    - Peak usage time identification
    - Usage pattern analysis
4. **Hardware Integration**:
    - Barrier control systems
    - Payment system integration
    - License plate recognition

5. **User Experience**:
    - More customization options
    - Additional visualization modes
    - Multi-language support

Contact and Support

For support or feature requests, please open an issue on GitHub: https://github.com/Friendlydevil03/park10/issues (https://github.com/Friendlydevil03/park10/issues)
