Empty Parking slot detection using OpenCV,CNN and Keras

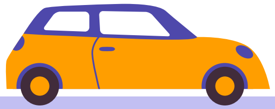
A Project By:-

THANDA SRI DIKSHITHA(100521729087)
THATIPAMULA SOUMYA(100521729084)
NAMDHARI ANUSHA(100521729082)

Under the guidance of:-

Dr. B.SUJATHA
ASSISTANT PROFESSOR
DEPT.OF CSE,UCEOU.

Table of Contents



Abstract

This project presents an automated system for detecting empty parking slots using a Convolutional Neural Network (CNN) model implemented with Keras and OpenCV for image processing. The system leverages a pre-trained CNN model to classify parking slots as empty or occupied based on grayscale image inputs.

The system is evaluated on real-world parking slot images, demonstrating its ability to accurately identify empty parking slots. The results are visualized by drawing bounding boxes around detected slots and displaying the count of available parking spaces on the image. The system's performance is enhanced by careful selection of model parameters, appropriate image preprocessing techniques, and the implementation of an efficient prediction .

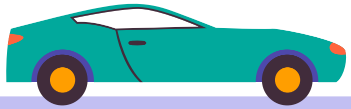
Introduction

Efficient parking management reduces congestion and enhances urban mobility. Manual detection of empty spots is inefficient and error-prone. This project, "Automated Parking Spot Detection Using Computer Vision and Deep Learning," aims to automate this process.

Key components include:

Image Processing: *Preprocesses images to identify potential parking spots.*
Manual Annotation: *Tool for defining parking spot coordinates.*
Model Training: *CNN trained to classify spots as empty or occupied.*
Real-Time Detection: *Detects and annotates parking spots in real-time.*

This system optimizes parking resource use and improves the user experience by reducing search time for parking.



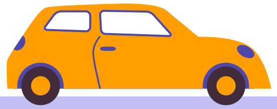
Literature Survey

Existing Systems

-High installation costs due to the need for multiple sensors.
- Maintenance can be challenging and costly, especially in large parking areas.
- Sensors can be affected by environmental factors like weather and dirt.
- Requires substantial computational resources to process and analyze images in real-time.
- Regular maintenance is needed to keep cameras clean and functional.
- Performance can be affected by lighting conditions, such as shadows, glare, and darkness.

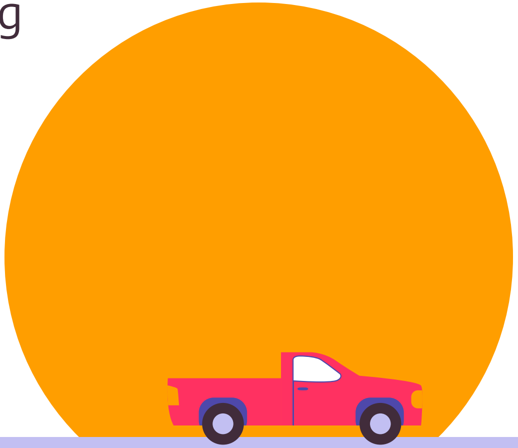
Proposed Methodologies

-Adaptable to various parking slot layouts without the need for physical sensor adjustments, making it more scalable for different environments.
-By using computer vision techniques,system provides real-time analysis of parking slot availability without the latency associated with sensor data transmission.
-Deep learning models, such as CNNs, excel in feature extraction and pattern recognition, leading to higher accuracy in detecting empty and occupied parking slots. They can handle complex scenarios such as varying lighting conditions.



Problem Statement

This project aims to address this challenge by developing an automated system using deep learning and computer vision techniques to detect and classify empty parking slots from images of parking Slots. The system will provide real-time feedback on parking availability, helping to streamline parking management, reduce traffic congestion, and enhance the overall urban mobility experience for drivers.



System Design

1. Image Processing

- **Purpose**: Preprocess images to identify potential parking spots.
- **Key Techniques**: Grayscale conversion, Gaussian blurring, edge detection.
- **Script**: `ImageProcessing.py`

2. Manual Annotation

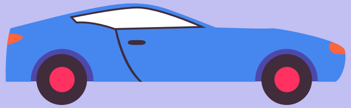
- **Purpose**: Define parking spot coordinates manually.
- **Key Techniques**: Mouse event handling to draw rectangles.
- **Script**: `parkingspotcoordinate.py`

3. Model Training

- **Purpose**: Train a CNN to classify parking spots as empty or occupied.
- **Key Techniques**: Data loading, preprocessing, CNN architecture, model training.
- **Script**: `emptyparkingspotdetectionmodel.py`

4. Real-Time Detection

- **Purpose**: Detect and annotate parking spots in real-time images.
- **Key Techniques**: Model inference, real-time image processing, annotation.
- **Script**: `emptyparkingspotdetection.py`



Implementation

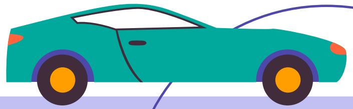
1. Image processing

Purpose

- Preprocess images to identify potential parking spots.

Key Techniques

1. **Grayscale Conversion:** Simplifies the image by removing color information.
   - **Code**: `gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`
2. **Gaussian Blurring:** Reduces noise and detail, enhancing edge detection.
   - **Code**: `blurred_gray = cv2.GaussianBlur(gray, (7,7), 0)`
3. **Edge Detection:** Identifies edges in the blurred image.
   - **Code**: `edges = cv2.Canny(blurred_gray, 30, 150)`
4. **Contour Detection:** Finds the boundaries of objects in the image.
   - **Code**: `contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)`
5. **Bounding Box Drawing:** Draws rectangles around detected contours that meet criteria.



Implementation

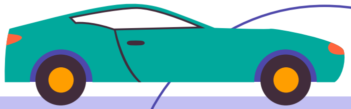
2. Manual annotation

Purpose:

- Define parking spot coordinates manually for training and validation purposes.

Key Techniques:

1. **Mouse Event Handling:**
    - Captures mouse events to draw rectangles.
    - `cv2.setMouseCallback('image', draw_rectangle)`
2. **Drawing Rectangles:**
    - Allows users to click and drag to define parking spot boundaries.
3. **Display and Update Image:**
    - Shows real-time updates as rectangles are drawn.
    - `cv2.imshow('image', img)`



Implementation

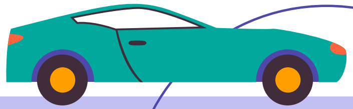
3. Model-Training

Purpose:

- Train a Convolutional Neural Network (CNN) to classify parking spots as empty or occupied.

Key Techniques:

1. **Data Loading and Preprocessing:**
   - Load images and labels from directories.
   - Convert images to grayscale and resize them.
   - Split data into training and testing sets.
   - Normalize image data and convert labels to categorical format.
2. **CNN Architecture:**
   - Define a Sequential model with Conv2D, MaxPooling2D, Flatten, Dense, and Dropout layers.
3. **Model Training:**
   - Compile the model with categorical cross-entropy loss and the Adam optimizer.
   - Train the model with the training data and validate it with the test data.
4. **Model Saving:**
   - Save the trained model for later use in real-time detection.



Implementation

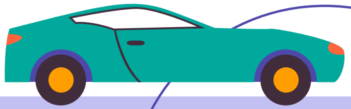
4. Real-Time detection

Purpose:

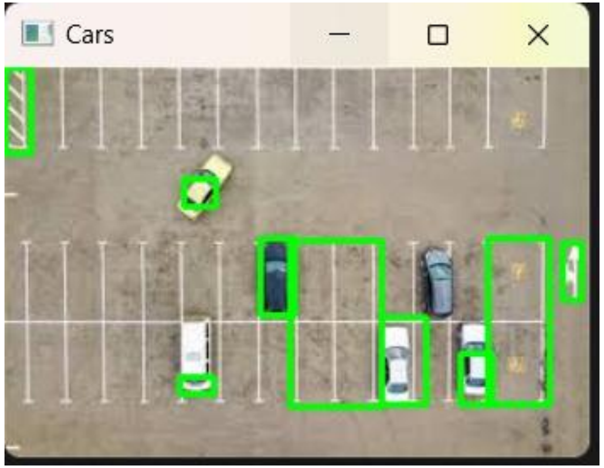
- Detect and annotate parking spots in real-time images using the trained model.

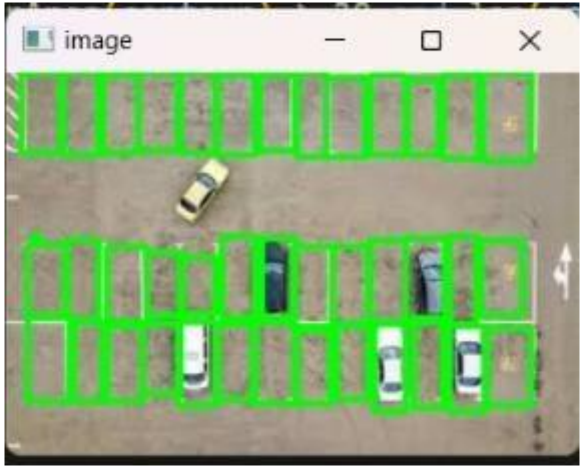
Key Techniques:

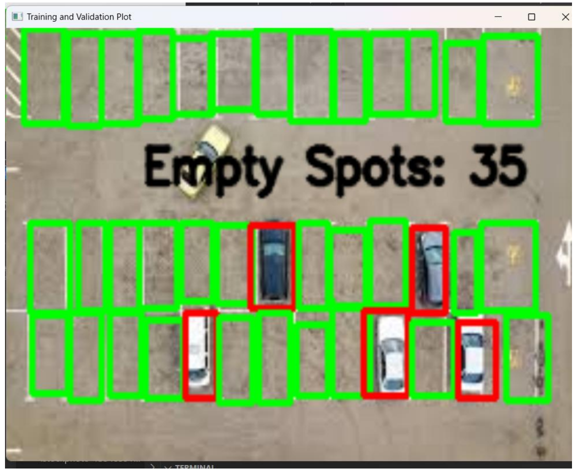
1. **Loading the Trained Model:**
   - Load the pre-trained model for inference.
2. **Define Parking Spot Coordinates:**
   - List of predefined coordinates for parking spots.
3. **Detect Empty Parking Spots:**
   - Extract region of interest (ROI) for each spot, preprocess it, and use the model to predict if the spot is empty.
4. **Annotate Image:**
   - Draw rectangles around detected spots and annotate the image with the count of empty spots.
5. **Display the Result:**
   - Show the annotated image with detected empty and occupied spots.
   - `cv2.imshow("Parking Lot", current_image)`



Results







Manual annotation & Real-Time detection

Conclusions

The project aims to address the challenge of efficient parking space management by leveraging computer vision and machine learning techniques to detect empty parking spots in real-time. Through the development of a convolutional neural network (CNN) model trained on a dataset of parking slot images, the system can accurately identify and monitor vacant parking spaces, providing valuable insights for optimizing parking allocation and improving overall parking efficiency.

The implementation process involved various stages, including data collection, preprocessing, model training, and real-time detection. we have created a robust and scalable solution capable of detecting empty parking spots with high accuracy and reliability.

Future Scope

The automated parking slot detection system can significantly contribute to the development of smart cities by integrating with urban management systems to reduce traffic congestion and enhance parking efficiency. The scalability of the system allows it to be deployed in multi-level parking structures and across entire cities, making it a versatile solution. Future applications could include the development of mobile apps and in-car systems that provide real-time parking availability to drivers, improving user experience.

Thank You