Comprehensive Tutorial: Car Parking Space Detection System

1. Introduction

This tutorial guides you through building a simple yet effective car parking space detection system using OpenCV and cvzone. The system enables manual marking of parking spaces on a static image, followed by real-time monitoring of parking space occupancy from video feeds. It's ideal for smart parking management and surveillance applications.

---

2. System Overview

The system consists of two main components:

1. **Parking Space Picker:** Allows manual annotation of parking spaces on a static parking lot image. The coordinates are saved for later detection.
2. **Occupancy Detector:** Processes video frames, identifies parked cars in marked spaces, and displays the real-time occupancy count.

---

3. Prerequisites

3.1 Software & Libraries

- **Python 3.8+**
- **OpenCV (cv2)**
- **cvzone**
- **NumPy**
- **Pickle**

3.2 Hardware Requirements

- Static parking lot image (carParkImg.png)
- Parking lot video feed (carPark.mp4) or webcam
- Standard PC or laptop

---

4. Step-by-Step Explanation

4.1 Parking Space Picker (parkingspacepicker.py)

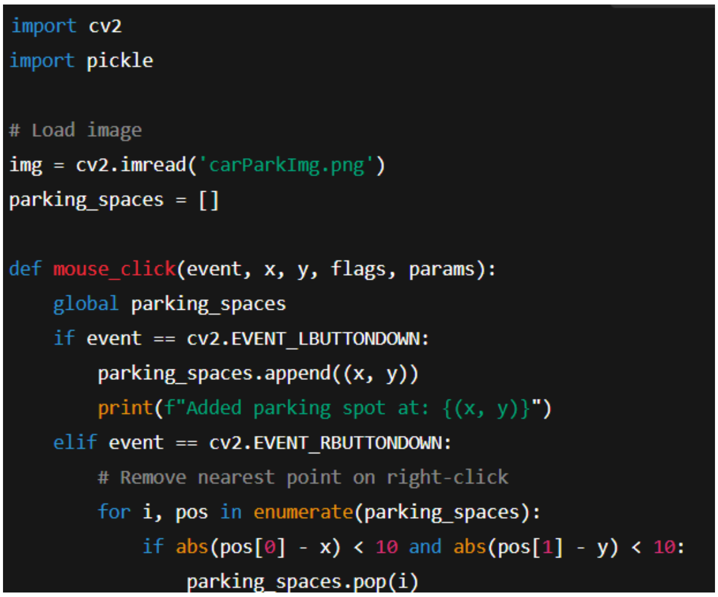
This script lets users manually mark parking spaces by clicking on the parking lot image.

Core Logic:

- Load the parking lot image

- Capture mouse clicks: left-click to add, right-click to remove points
- Save parking space coordinates using Pickle

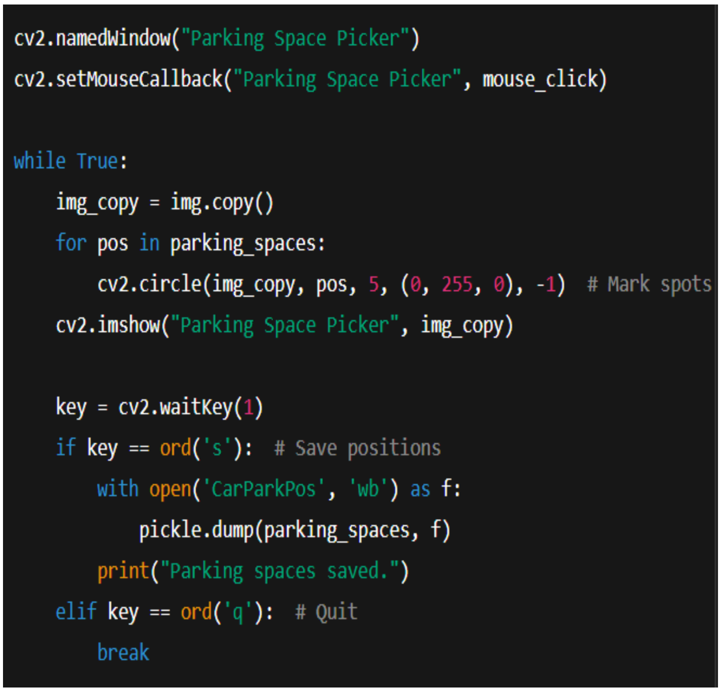
Code Snippet:



```
import cv2
import pickle

# Load image
img = cv2.imread('carParkImg.png')
parking_spaces = []

def mouse_click(event, x, y, flags, params):
    global parking_spaces
    if event == cv2.EVENT_LBUTTONDOWN:
        parking_spaces.append((x, y))
        print(f"Added parking spot at: {(x, y)}")
    elif event == cv2.EVENT_RBUTTONDOWN:
        # Remove nearest point on right-click
        for i, pos in enumerate(parking_spaces):
            if abs(pos[0] - x) < 10 and abs(pos[1] - y) < 10:
                parking_spaces.pop(i)
```



```
cv2.namedWindow("Parking Space Picker")
cv2.setMouseCallback("Parking Space Picker", mouse_click)

while True:
    img_copy = img.copy()
    for pos in parking_spaces:
        cv2.circle(img_copy, pos, 5, (0, 255, 0), -1)  # Mark spots
    cv2.imshow("Parking Space Picker", img_copy)

    key = cv2.waitKey(1)
    if key == ord('s'):  # Save positions
        with open('CarParkPos', 'wb') as f:
            pickle.dump(parking_spaces, f)
        print("Parking spaces saved.")
    elif key == ord('q'):  # Quit
        break
```

Explanation:

- The mouse callback captures left and right clicks for adding/removing parking spots.
- The image updates in real-time to show marked spots as green circles.
- Pressing **'s'** saves the coordinates; **'q'** quits the program.

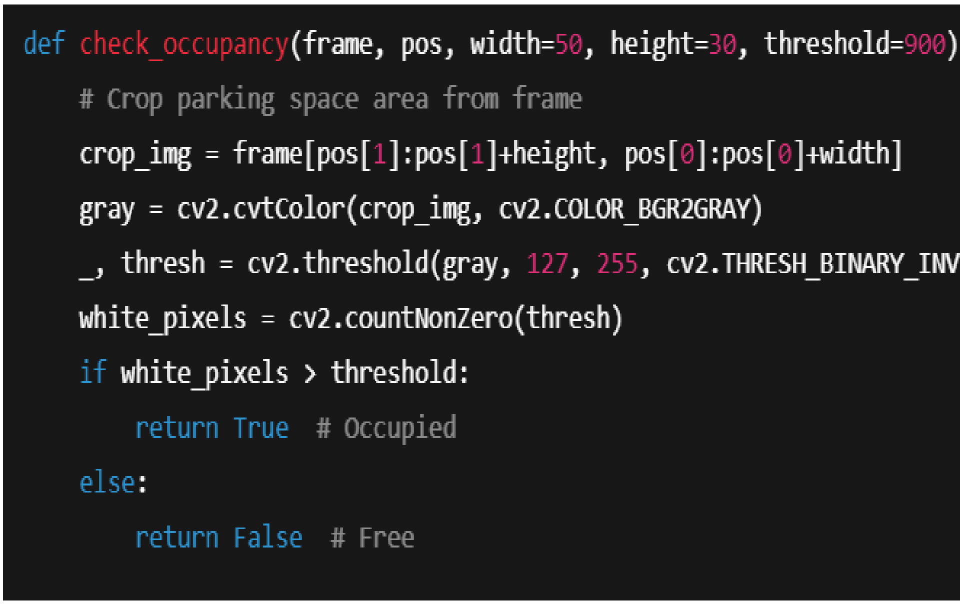
4.2 Occupancy Detection (main.py)

This script loads saved parking spot coordinates and processes a video to detect occupancy.

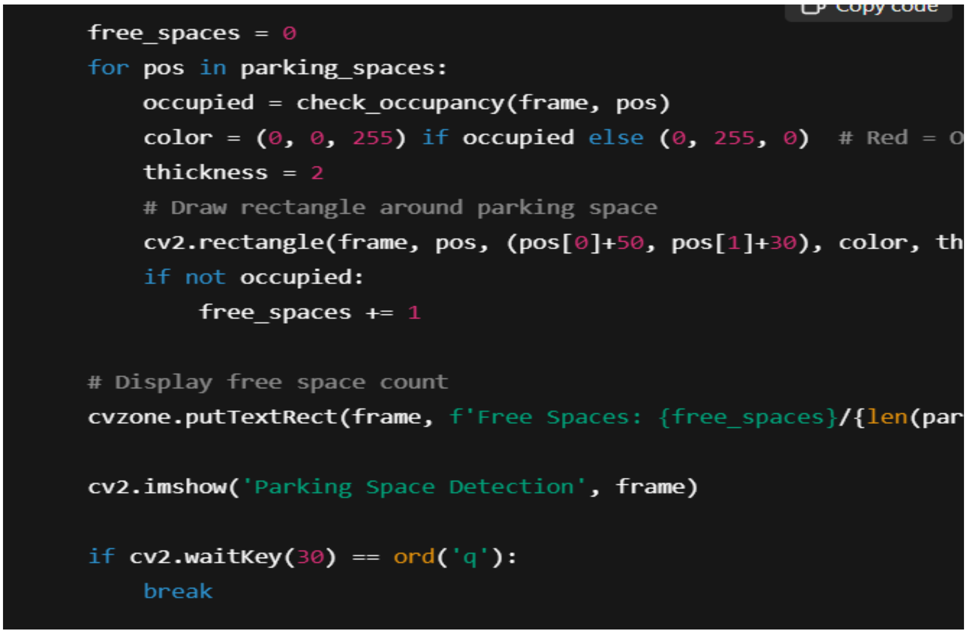
Core Logic:

- Load saved parking spaces
- Process each frame to check if space is occupied (based on pixel intensity or color difference)
- Mark parking spots as free or occupied on the frame
- Display real-time count of free spaces

Code Snippet:



```
def check_occupancy(frame, pos, width=50, height=30, threshold=900)
    # Crop parking space area from frame
    crop_img = frame[pos[1]:pos[1]+height, pos[0]:pos[0]+width]
    gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV
    white_pixels = cv2.countNonZero(thresh)
    if white_pixels > threshold:
        return True  # Occupied
    else:
        return False  # Free
```



```
    free_spaces = 0
    for pos in parking_spaces:
        occupied = check_occupancy(frame, pos)
        color = (0, 0, 255) if occupied else (0, 255, 0)  # Red = O
        thickness = 2
        # Draw rectangle around parking space
        cv2.rectangle(frame, pos, (pos[0]+50, pos[1]+30), color, th
        if not occupied:
            free_spaces += 1

    # Display free space count
    cvzone.putTextRect(frame, f'Free Spaces: {free_spaces}/{len(par

    cv2.imshow('Parking Space Detection', frame)

    if cv2.waitKey(30) == ord('q'):
        break
```

Explanation:

- The check_occupancy() function crops the area of each parking space and thresholds it to detect the presence of a car (white pixels represent occupied space).

- Spaces are outlined in green if free and red if occupied.
- The total count of free spaces updates dynamically and is shown on the video.
- Press **'q'** to quit the application.

---

5. Key Concepts Explained

5.1 Manual Parking Space Annotation

Enables customization for different parking lot layouts without complex automatic detection.

5.2 Occupancy Detection by Pixel Thresholding

Simple yet effective for static cameras; detects presence by analyzing pixel intensities.

5.3 Visualization and Real-Time Feedback

Provides intuitive on-screen information to easily verify the system's output.

---

6. Potential Improvements

1. **Machine Learning-based Occupancy Detection:** Replace thresholding with deep learning for robustness to lighting and shadows.
2. **Multi-angle Camera Support:** Handle varying perspectives and occlusions.
3. **Mobile and Web Dashboard:** For remote monitoring and alerts.
4. **Auto Calibration:** Detect parking spots automatically to eliminate manual annotation.
5. **Integration with Payment Systems:** For smart parking management.

---

7. Conclusion

This project demonstrates a practical car parking space detection system combining manual spot marking and real-time video analysis using OpenCV and cvzone. The system provides a foundation for scalable smart parking solutions, making parking management easier and more efficient.

Final Code Repository

https://github.com/thinkrobotics/Live-Smart-Parking-Detection-System