Project Report

PARKING SPKACE MONITERING SYSTEM

(Using Python)

Presented to:

Dr. Adnan

Presented by:

Muhammad Hasnain (21-CS-96)

Haroon-Ur-Rasheed (21-CS-144)

Department of Computer Science
University of Engineering and Technology, Taxila

Fall 2023

Contents

-: Introduction: -

The Parking Space Monitoring System represents a comprehensive solution to address the growing need for efficient parking management. With urban areas experiencing increased vehicular traffic, optimizing parking space utilization becomes paramount. This project harnesses the power of computer vision to develop a real-time monitoring system that tracks and analyses parking space occupancy in a given parking lot.

In contemporary urban landscapes, finding available parking spaces is a common challenge, contributing to traffic congestion and wasted time for drivers. This system aims to alleviate this issue by providing an intelligent and automated approach to parking space monitoring. Leveraging the capabilities of computer vision, the system processes a video feed from a parking lot, identifies individual parking spaces, and determines their occupancy status.

The implementation utilizes OpenCV for video capture and image processing, CVZone for enhanced text rendering, and Pickle for efficient storage and retrieval of parking space positions. The system not only monitors parking space occupancy but also allows users to manually select and adjust parking space positions through a dedicated interface.

This project serves as a practical demonstration of how technology, specifically computer vision, can contribute to the development of smart solutions for everyday challenges. As cities continue to expand, the demand for intelligent parking management systems becomes increasingly vital. The Parking Space Monitoring System represents a step towards creating more efficient and user-friendly parking solutions, with the potential for further enhancements and integration into smart city initiatives.

-: System Elements: -

Language used:

- Python

Libraries Used:

1. **OpenCV (cv2):** A computer vision library providing tools for image and video processing, essential for tasks such as video capture, image conversion, and thresholding in the parking space monitoring system.

2. **CVZone:** A computer vision library extending OpenCV functionalities, used for improved text rendering on images, enhancing the visual representation of parking space occupancy information in the project.

3. **Pickle**: A Python module for serializing and deserializing objects, employed in this project for efficient storage and retrieval of parking space positions, enabling seamless management of parking space data.

-: Working: -

Certainly! Here's the modified detailed working of the Parking Space Monitoring System according to your provided code:

1. ### Video Feed Capture

   a. The system initializes by capturing a video feed from the file 'carPark.mp4' using OpenCV's `VideoCapture` module.

2. ### Loading Parking Space Positions

   a. The pre-defined positions of parking spaces are loaded from the 'CarParkPos' file using Pickle. This list of positions (`posList`) will be utilized for monitoring the occupancy of individual parking spaces.

3. Image Preprocessing

a. Each video frame undergoes a series of image preprocessing steps to enhance parking space detection:

- Grayscale Conversion: Convert the frame to grayscale.
- Gaussian Blur: Apply a Gaussian blur to reduce noise.
- Adaptive Thresholding: Convert the blurred image to a binary image using adaptive thresholding.
- Median Blur: Further reduce noise using median blur.
- Dilation: Dilate the image to enhance the visibility of parking space boundaries.

4. Parking Space Checking

a. For each parking space in `posList`, a region of interest (ROI) is extracted from the preprocessed frame. The count of non-zero pixels within this ROI is calculated. If the count is below 900, the space is considered free; otherwise, it is marked as occupied.

5. Visualization

a. The identified parking spaces are visually represented on the original frame. Green rectangles indicate vacant spaces, while red rectangles denote occupied spaces. The count of free spaces is displayed on the image using CVZone's `putTextRect` method, providing real-time information.

6. User Interaction

a. The application runs indefinitely, continuously processing video frames and updating the display. The user can stop the application by pressing the 'q' key.

Parking Picker Code (Run Once)

b. A separate script is provided for selecting and adjusting parking space positions. This script allows users to interactively add or remove parking spaces by left or right-clicking, respectively. The selected positions are stored in the 'CarParkPos' file using Pickle for future reference.

c. This project effectively combines computer vision and user interaction to create a practical solution for monitoring parking space occupancy in real-time.

-: Features: -

Feature Description

1. **Real-time Video Feed Analysis**
    a. The system captures a video feed from a specified file ('carPark.mp4') in real-time, utilizing OpenCV's `VideoCapture` module. This feature allows for continuous monitoring and analysis of the parking lot.

2. **Parking Space Position Management**
    a. The positions of parking spaces are stored and loaded from the 'CarParkPos' file using Pickle. This feature facilitates easy configuration and modification of parking space positions, allowing for adaptability to different parking lot layouts.

3. **Intelligent Image Preprocessing**
    a. The system applies a series of image preprocessing techniques to enhance parking space detection accuracy:
        - Grayscale Conversion: Converts each frame to grayscale for simplicity in subsequent processing.
        - Gaussian Blur: Reduces noise in the image to improve edge detection.
        - Adaptive Thresholding: Creates a binary image using adaptive thresholding, highlighting potential parking spaces.
        - Median Blur: Further reduces noise in the binary image.
        - Dilation: Enhances the visibility of parking space boundaries through dilation.

4. **Parking Space Occupancy Detection**
    a. The system intelligently checks the occupancy status of each parking space based on pixel count within designated regions of interest (ROIs). If the count falls below a predefined threshold (900 in this case), the space is considered free; otherwise, it is marked as occupied.

5. **Visual Representation**
    a. The identified parking spaces are visually represented on the original frame. Vacant spaces are highlighted with green rectangles, while occupied spaces are marked with red rectangles. The count of free spaces is displayed on the image using CVZone's `putTextRect` method, providing users with instant visual feedback.

6. User Interaction and Control

a. The application runs indefinitely, continuously processing video frames. Users can interact with the system by pressing the 'q' key to halt the application. Additionally, a separate script allows users to dynamically select and adjust parking space positions through mouse clicks, enhancing user control and customization.

7. Enhanced Text Rendering

a. CVZone's `putTextRect` method is employed to enhance text rendering on images. This feature improves the clarity of displayed information, making it more readable and visually appealing.

8. Modular Design and Extensibility

a. The project is designed with a modular structure, facilitating easy modifications and future enhancements. The code is organized into functions, allowing for straightforward integration of additional features or improvements.

9. Efficient Data Storage

a. Pickle is utilized for efficient storage and retrieval of parking space positions. This ensures seamless management of parking space data across multiple runs of the application.

10. Practical Parking Space Picker

a. A separate script is provided for the initial configuration of parking space positions. Users can interactively add or remove parking spaces through mouse clicks, providing a practical and intuitive method for setting up the system.

The Parking Space Monitoring System combines these features to create a versatile and user-friendly solution for real-time parking space management, with potential for further customization and integration into smart city initiatives.

-: Benefits: -

The Parking Space Monitoring System offers several benefits, contributing to improved parking management and user experience:

- **Optimized Parking Space Utilization:**
    - The real-time monitoring system enables efficient use of parking spaces by providing instant information about occupancy status.
    - Users can quickly identify and utilize available parking spaces, reducing congestion and optimizing parking lot capacity.
- **Time and Fuel Efficiency:**
    - Drivers can save time and fuel by avoiding unnecessary searches for parking spaces. The system's real-time updates allow users to navigate directly to available spots.
- **User-Friendly Interface:**
    - The system provides a user-friendly interface for setting up and adjusting parking space positions interactively.
    - Visual representations of parking space occupancy enhance user understanding and decision-making.
- **Adaptability to Different Parking Lot Layouts:**
    - The ability to configure and modify parking space positions allows the system to adapt to various parking lot layouts and configurations.
- **Automated Monitoring:**
    - The system automates the monitoring process, eliminating the need for manual surveillance.
    - Continuous monitoring ensures that parking space occupancy information is always upto-date.
- **Enhanced User Control:**
    - Users have control over the system, with the ability to start and stop monitoring as needed.
    - The separate parking space picker script allows for dynamic adjustment of parking space positions.
- **Data-Driven Decision Making:**
    - The system provides valuable data on parking space occupancy, supporting data-driven decision-making for parking lot management and optimization.
- **Potential for Integration into Smart City Initiatives:**
    - The modular design and extensibility of the project make it a suitable candidate for integration into broader smart city initiatives.
    - Real-time parking space data can contribute to city planning and traffic management strategies.
- **Efficient Data Storage:**
    - The use of Pickle for data storage ensures efficient handling of parking space positions, allowing for seamless data management across multiple system runs.

- **Customization and Extension:**
    - The modular code structure enables easy customization and extension of the system's functionality.
    - Future enhancements and additional features can be integrated into the existing framework.
- **Improved Traffic Flow:**
    - By providing real-time information on parking space availability, the system contributes to improved traffic flow around parking lots and nearby areas.
- **Cost Savings:**
    - Drivers can save costs associated with fuel consumption and potential parking fines by efficiently navigating to available parking spaces.

The Parking Space Monitoring System addresses common challenges in parking management, offering a practical and adaptable solution with the potential for positive impacts on both individual drivers and urban infrastructure.

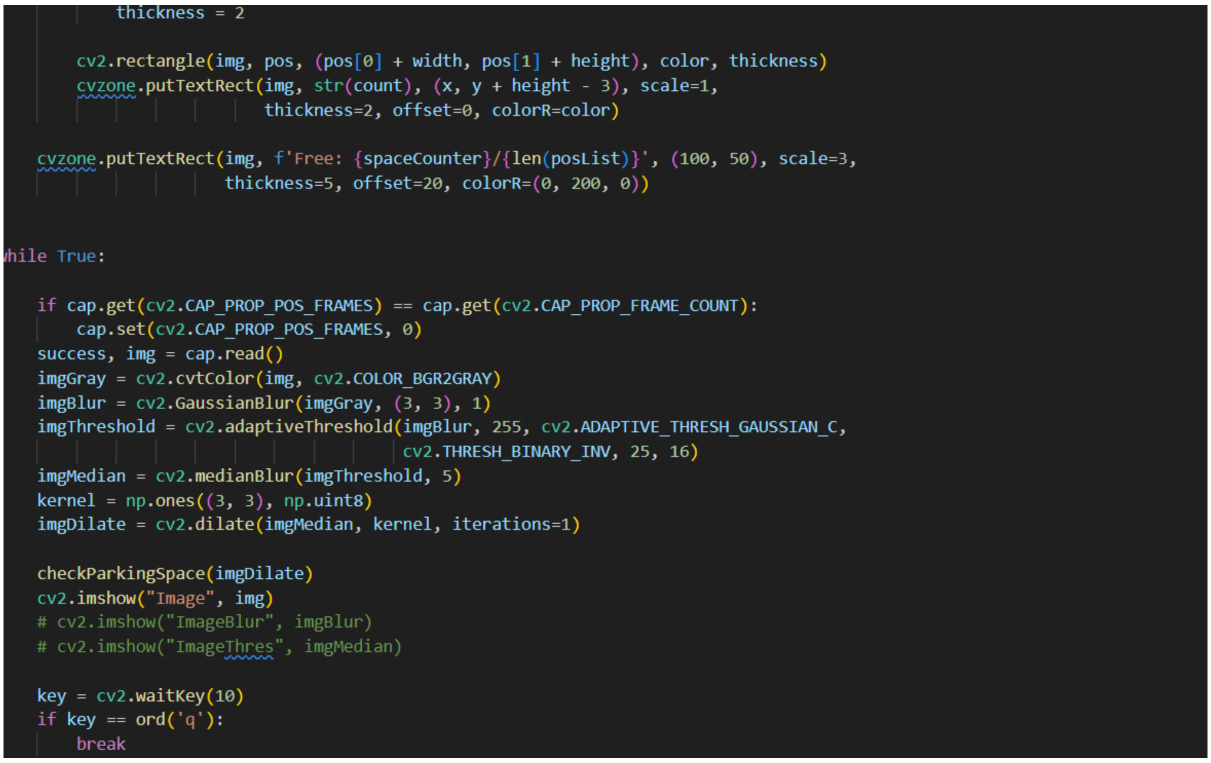
-: Future improvements: -

- **Automated Position Detection:**
    - Implement advanced computer vision algorithms for dynamic and automated detection of parking space positions, reducing the need for manual configuration.
- **Mobile Application Development:**
    - Create a mobile application for on-the-go access, navigation to available spaces, personalized notifications, and enhanced user experience.
- **Machine Learning for Predictive Analytics:**
    - Explore machine learning models for anomaly detection and predictive analytics to forecast peak parking times, optimizing resource allocation.
- **Web Integration for Remote Monitoring:**
    - Develop a web interface to enable remote monitoring, real-time updates, historical data analysis, and user notifications for efficient parking space management.

-: Conclusion: -

In conclusion, the Parking Space Monitoring System represents a significant step towards addressing the challenges of parking space management through the application of computer vision and user-friendly interfaces. The system's real-time monitoring, dynamic configurability, and potential for future enhancements make it a valuable tool for optimizing parking utilization and contributing to more efficient urban mobility. As technology continues to evolve, the project lays a foundation for smart city initiatives and user-centric solutions in the realm of parking management.

-: Code Snippets: -



```
            thickness = 2

        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
        cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1,
                           thickness=2, offset=0, colorR=color)

    cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50), scale=3,
                       thickness=5, offset=20, colorR=(0, 200, 0))


hile True:

    if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    success, img = cap.read()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                         cv2.THRESH_BINARY_INV, 25, 16)
    imgMedian = cv2.medianBlur(imgThreshold, 5)
    kernel = np.ones((3, 3), np.uint8)
    imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)

    checkParkingSpace(imgDilate)
    cv2.imshow("Image", img)
    # cv2.imshow("ImageBlur", imgBlur)
    # cv2.imshow("ImageThres", imgMedian)

    key = cv2.waitKey(10)
    if key == ord('q'):
        break
```

Output: