SMART PARKING

PHASE 2:

INNOVATION:

Consider integrating camera-based solutions for image processing to detect parking space availability.

Camera based parking space detection:

Image completion:

Camera Setup:

Set up one or more cameras in the parking area to capture images or video streams.

Image Capture:

Continuously capture images or frames from the camera feed.

Preprocessing

Image Enhancement:

Apply image enhancement techniques such as brightness and contrast adjustment, histogram equalization, and noise reduction to improve image quality.

Geometric Correction:

Correct for camera distortion and perspective distortion to ensure accurate object detection.

Object Detection

Vehicle Detection:

Use object detection techniques (e.g., Haar Cascades, YOLO, SSD, Faster R-CNN) to detect vehicles in each frame of the camera feed.

Object Tracking:

Implement object tracking algorithms to track the detected vehicles across multiple frames.

Parking Space Detection

Define Regions of Interest (ROIs):

Define the areas of the camera feed where parking spaces are located.

Occupancy Classification:

Use image segmentation or classification techniques to classify each parking space within the ROIs as "occupied" or "vacant."

Visualization and User Interface

Visualization:

Overlay parking space occupancy information on the camera feed. Highlight vacant and occupied spaces.

User Interface:

Create a user interface to display parking space availability information to users.

Data Storage and Analysis

Data Storage:

Store parking space occupancy data in a database or file for historical analysis.

Data Analysis:

Analyze historical data to predict parking availability trends and make informed decisions.

Alerting (Optional)

Implement an alerting system to notify users when parking spaces become available or when the parking lot is full.

Deployment

Deploy your parking space detection system in the parking area and ensure it runs continuously.

It's important to note that developing a robust parking space detection system can be a complex task that may require expertise in computer vision, machine learning, and software development. Additionally, you may need to consider factors like camera placement, lighting conditions, and environmental factors when designing your system.

PYTHON CV CODE:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt


# Load the pre-trained vehicle detection classifier (Haar Cascade)

car_cascade =
cv2.CascadeClassifier('/content/haarcascade_car.xml')


# Create a VideoCapture object to capture video from a file or
camera (you can specify the source)

video_source = '/content/pexels_videos_2406459 (2160p).mp4' #
Replace with your video path or use 0 for the default camera

cap = cv2.VideoCapture(video_source)


# Define parking space boundaries (x, y, width, height)

parking_spaces = [(60, 60, 50, 50),(60, 60, 50, 50),(60, 60, 50, 50),(60,
60, 50, 50),(60, 60, 50, 50)]  # Example: Two parking spaces


while True:

    # Read a frame from the video feed

    ret, frame = cap.read()
```

```
    if not ret:
        break

    # Convert the frame to grayscale for vehicle detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect cars in the frame using the car_cascade classifier
    cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(50, 50))

    # Initialize a list to track parking space occupancy
    occupancy_status = ['Vacant'] * len(parking_spaces)

    # Draw rectangles around detected cars and check parking
space occupancy
    for i, (x, y, w, h) in enumerate(cars):
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Check if the detected car is within any parking space
        for j, (px, py, pw, ph) in enumerate(parking_spaces):
```

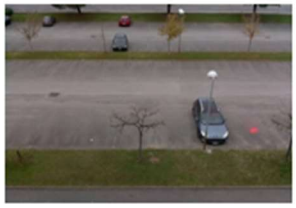
```
            if x >= px and y >= py and x + w <= px + pw and y + h <=
py + ph:
                occupancy_status[j] = 'Occupied'


    # Print parking space availability
    for j, status in enumerate(occupancy_status):
        print(f'Space {j + 1}: {status}')


    # Display the frame with detected cars and parking space
occupancy using Matplotlib
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    plt.imshow(frame_rgb)
    plt.axis('off')
    plt.show()


    # Exit the loop when the 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


# Release the VideoCapture and close any OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

PROTOTYPE:

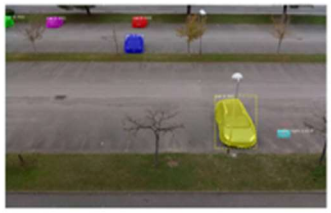
Input image

Instance identification of car objects in image

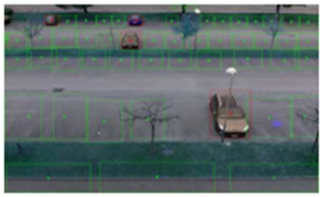
Obtain bounding boxes as locations of all parking lots



Input image at time t+1



Instance identification of car objects in image



Return output with list of parking spots, their centre locations & occupancy status

Notes: haarcascade_car Xml code is add in Github link

To use OpenCV (Open Source Computer Vision Library) in Python, you'll first need to install it if you haven't already. You can install OpenCV using pip:

```
pip install opencv-python
```

Once OpenCV is installed, you can use it in your Python code.

THANK YOU!