AI ENABLED CAR PARKING USING OPENCV

ARTIFICIAL INTELLIGENCE

PROJECT REPORT

Submitted By

SAGUTURU KAVYA (20BCB0142)

VORUGUNTLA THAREESHA (20MID0013)

KUCHIBHATLAMOHAN DATTA(20MID0012)

ELISETTI MANOGNASREE (20MID0047)

Table of Contents

1. INTRODUCTION 2

1.1 Overview 2

1.2 Purpose 2

2. LITERATURE SURVEY 3

2.1 Existing Problem 3

2.2 Proposed Solution 3

3. THEORITICAL ANALYSIS 5

3.1 Block diagram 5

3.2 Hardware / Software Designing 6

4. EXPERIMENTAL INVESTIGATIONS 7

5. FLOWCHART 8

6. RESULT 9

7. ADVANTAGES & DISADVANTAGES 12

8. APPLICATIONS 14

9. CONCLUSION 15

10. FUTURE SCOPE 16

11. BIBILOGRAPHY 17

APPENDIX 17

1. INTRODUCTION

1.1 Overview

Most AI-enabled car parking systems parking a car today are still managed by hand. There is no automated monitoring system in place to keep track of how much capacity each parking place contains. In order to find an empty spot, drivers often have to make a circuitous trip through the parking lot. Where there are more people than parking spots, such problems are especially common near hospitals, malls, schools, and other large gathering places.

1.2 Purpose

The purpose of using AI-enabled car parking system is to automate and optimize the process of parking vehicles in a parking lot or garage. By employing computer vision techniques and AI algorithms, these systems can efficiently detect and track vehicles, analyze parking space availability, and guide drivers to vacant spots.

2. LITERATURE SURVEY

2.1 Existing Problem

- **Problem statement 1:**

Consequently, once a car enters a parking garage followed by a parking space, a ping ultrasonic sensor will then be able to determine if a car is parked in the space or not. This information would then be relayed to update the network.

- **Problem statement 2:**

Avoid excessive parking supply. Use Parking Management to encourage more efficient use of existing parking facilities and address any spill over problems that result from pricing. Develop Transportation Management Associations to provide parking management and brokerage services in a particular area.

- **Problem Statement 3:**

Traditional car parking systems are inefficient and time-consuming. Drivers often have to spend a lot of time searching for an empty parking space, leading to congestion in the parking lot. Additionally, traditional parking systems require manual intervention for ticketing and payment processing, which can lead to long queues and delays.

2.2 Proposed Solution

- Idea / Solution description The idea behind an AI-enabled car parking system using **OpenCV** is to leverage the power of computer vision to automate the parking process. By using cameras and image processing algorithms, the system can detect the presence of cars in the parking lot and guide drivers to empty parking spaces. The system can also automate the ticketing and payment processing, making the process faster and more convenient for drivers.

- **Novelty / Uniqueness** The use of computer vision technology for car parking systems is a novel approach that can significantly improve the efficiency and convenience of parking. By using OpenCV, the system can accurately detect and track cars, even in crowded parking lots. Additionally, the use of AI algorithms can optimize the parking process by analyzing parking patterns and occupancy rates.

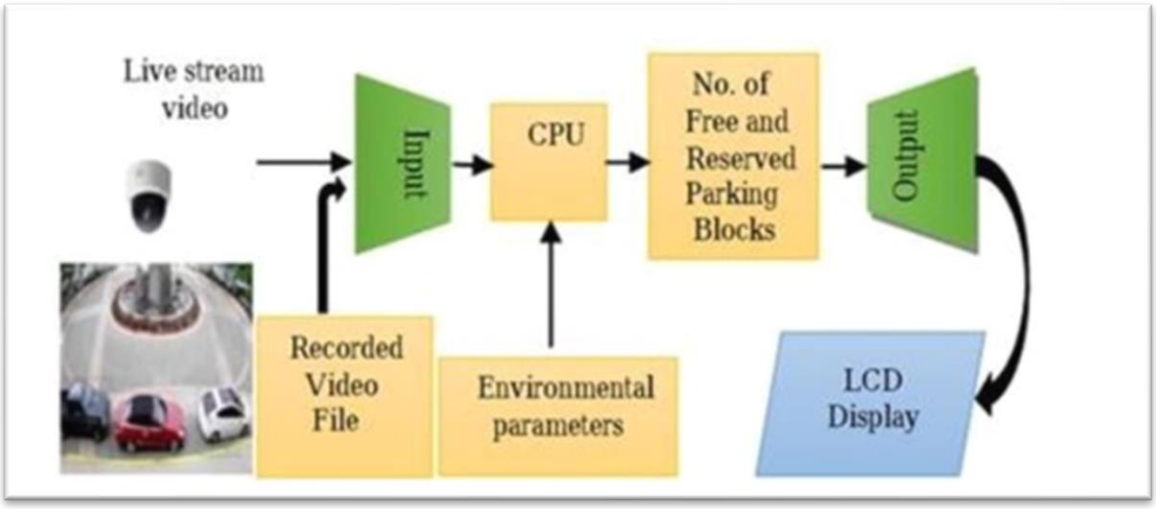
- **Social Impact / Customer Satisfaction** An AI-enabled car parking system using OpenCV can have a significant social impact by reducing traffic congestion and improving the overall driving experience. By guiding drivers to empty parking spaces, the system can reduce the time spent searching for parking and minimize traffic congestion in the parking lot. Additionally, by automating the ticketing and payment process, the system can improve the convenience and efficiency of the parking process.

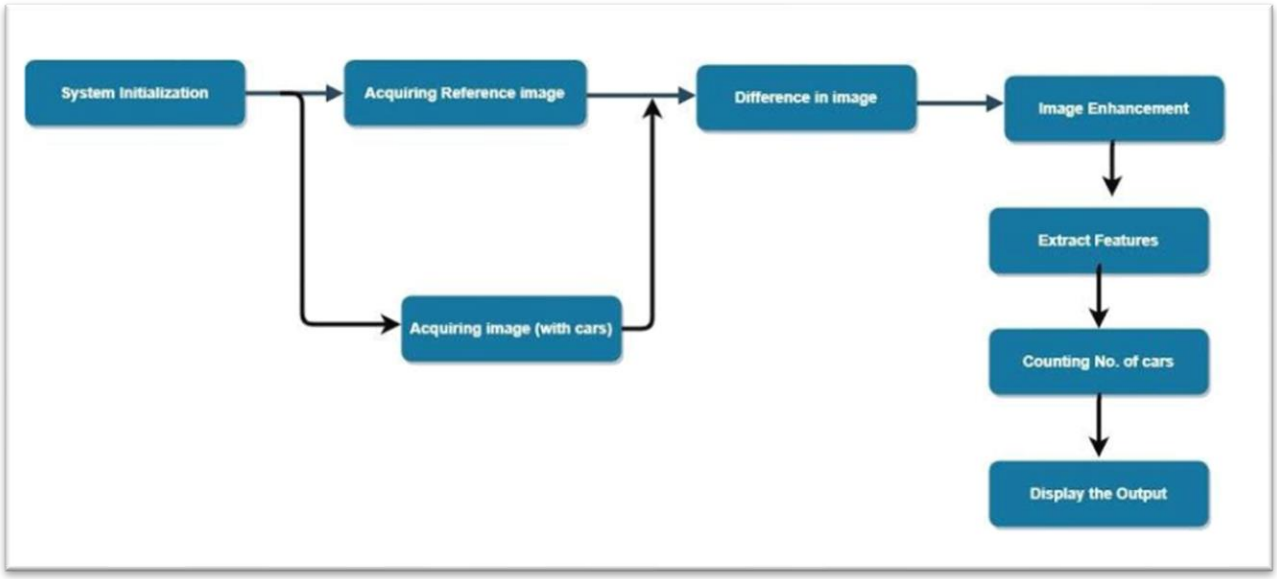
- **Business Model** (Revenue Model) The business model for an AI-enabled car parking system can be based on a pay-per-use model, where drivers pay a fee based on the duration of the parking. The system can be integrated with a payment gateway to enable online payments, making the process faster and more convenient for drivers. Additionally, the system can generate revenue by collecting data on parking patterns and occupancy rates, which can be used by parking lot operators to optimize the parking process.

- **Scalability of the Solution** An AI-enabled car parking system using OpenCV can be easily scaled to accommodate large parking lots and multiple parking locations. By using a centralized database and cloud-based architecture, the system can easily handle large volumes of data and support multiple users. Additionally, the system can be customized based on the specific requirements of the parking lot, making it adaptable to different environments and use cases.

3. THEORITICAL ANALYSIS

3.1 Block diagram





3.2 Hardware / Software Designing

Python is used for the project's execution. A compiler is used Pycharm. The project is carried out using the following libraries.

1. **Open CV**: python Open CV, a sizable open- source library for computer vision, machine erudition, and image processing currently plays a significant part in real- time operation, which is crucial in modern systems. It can be used to process pictures so that people can fete goods, people's faces, and even human handwriting. The Open CV array structure can be reclaimed by Python for figure out when it's coupled with a variety of libraries, cognate as NumPy. We use vector space and implement fine activities to the aspects of a visual pattern in order to identify it.

2. **CV zone**: This computer vision package facilitates the implementation of AI and image processing operations. It primarily makes use of the OpenCV and Media pipe libraries.

3. **NumPy** : For the Python programming language, NumPy is a library that adds support for largish , multi-layered arrays and matrices.it additionally affords a big count of high-stage mathematical features to work with these arrays.

4. **cv2 library:** OpenCV has a function called cv2 that can read tape recording().

5. **Pycharm**: PyCharm is a specialized Python Integrated Development Environment(IDE) that gives a vast range of crucial equipment for Python developers.

4. EXPERIMENTAL INVESTIGATIONS

Experimental investigations involve data analysis, algorithm development, simulation testing, performance evaluation, user studies, robustness testing, and real-world deployment trials.

- **Data Collection and Analysis:** Gathering real-world data from parking scenarios to create a representative dataset for training and evaluation. This involves collecting video footage, sensor data, and other relevant information from parking lots or environments. The data can be analyzed to identify patterns, challenges, and specific parking scenarios that require attention.

- **Algorithm Development and Optimization:** Experimenting with different AI algorithms and techniques for tasks like object detection, parking space detection, path planning, and collision avoidance. This investigation involves comparing the performance of various algorithms, optimizing hyperparameters, and fine-tuning the models to achieve better accuracy, efficiency, and robustness.

- **Simulation and Virtual Testing:** Creating simulated environments to test and evaluate the AI-enabled car parking system. This investigation allows for controlled experiments in diverse scenarios, including different parking lot layouts, traffic patterns, and environmental conditions. Simulations can help identify edge cases, validate algorithms, and assess system behavior before real-world deployment.

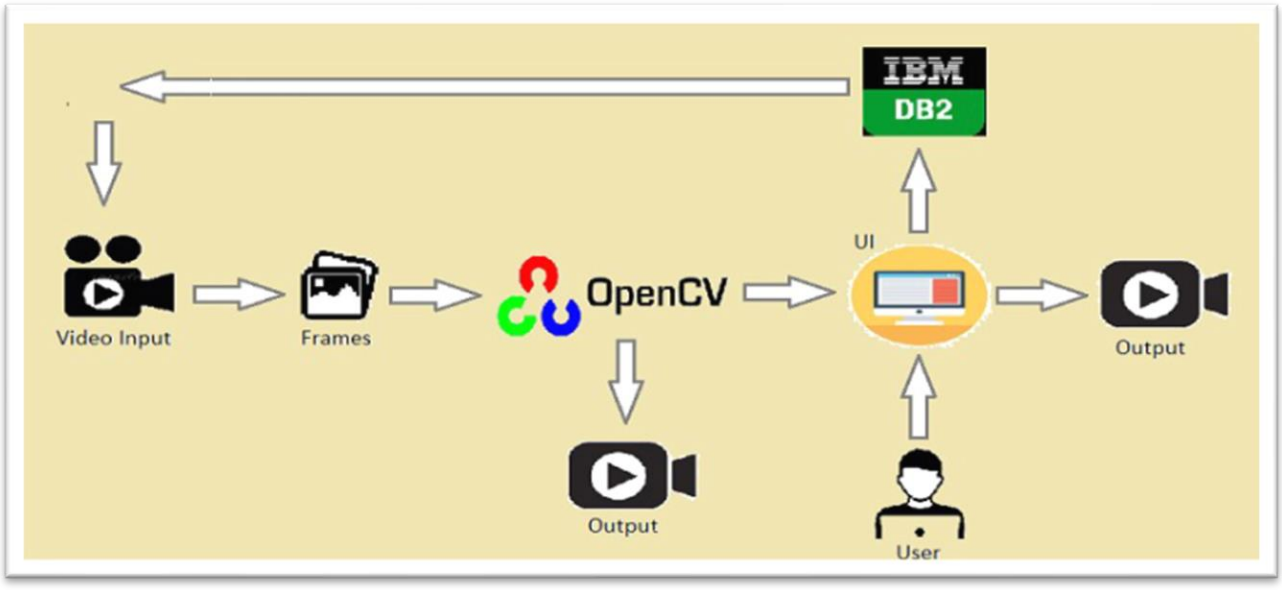
- **Performance Evaluation**: Conducting thorough performance evaluations to measure the system's accuracy, efficiency, and reliability. This investigation includes assessing metrics such as detection accuracy, false positives/negatives, parking success rates, time taken for parking maneuvers, and overall user satisfaction. Comparative evaluations with existing solutions or benchmarks can provide valuable insights.

- **Human Factors and User Studies**: Investigating user experience aspects through user studies and feedback collection. This investigation focuses on understanding the usability, intuitiveness, and acceptance of the AI-enabled car parking system by drivers. It involves conducting surveys, interviews, or usability tests to gather user opinions and suggestions for system improvements.

- **Robustness and Safety Testing**: Assessing the system's performance in challenging conditions and potential failure scenarios. This investigation involves deliberately introducing variations such as adverse weather conditions, occlusions, sensor failures, or unexpected pedestrian movements to evaluate the system's robustness and safety measures.

- **Real-world Deployment Trials**: Carrying out field trials and pilot deployments of the AI-enabled car parking system in real-world settings. This investigation allows for gathering feedback, identifying system limitations, and refining the solution based on practical challenges encountered during deployment.

5. FLOWCHART



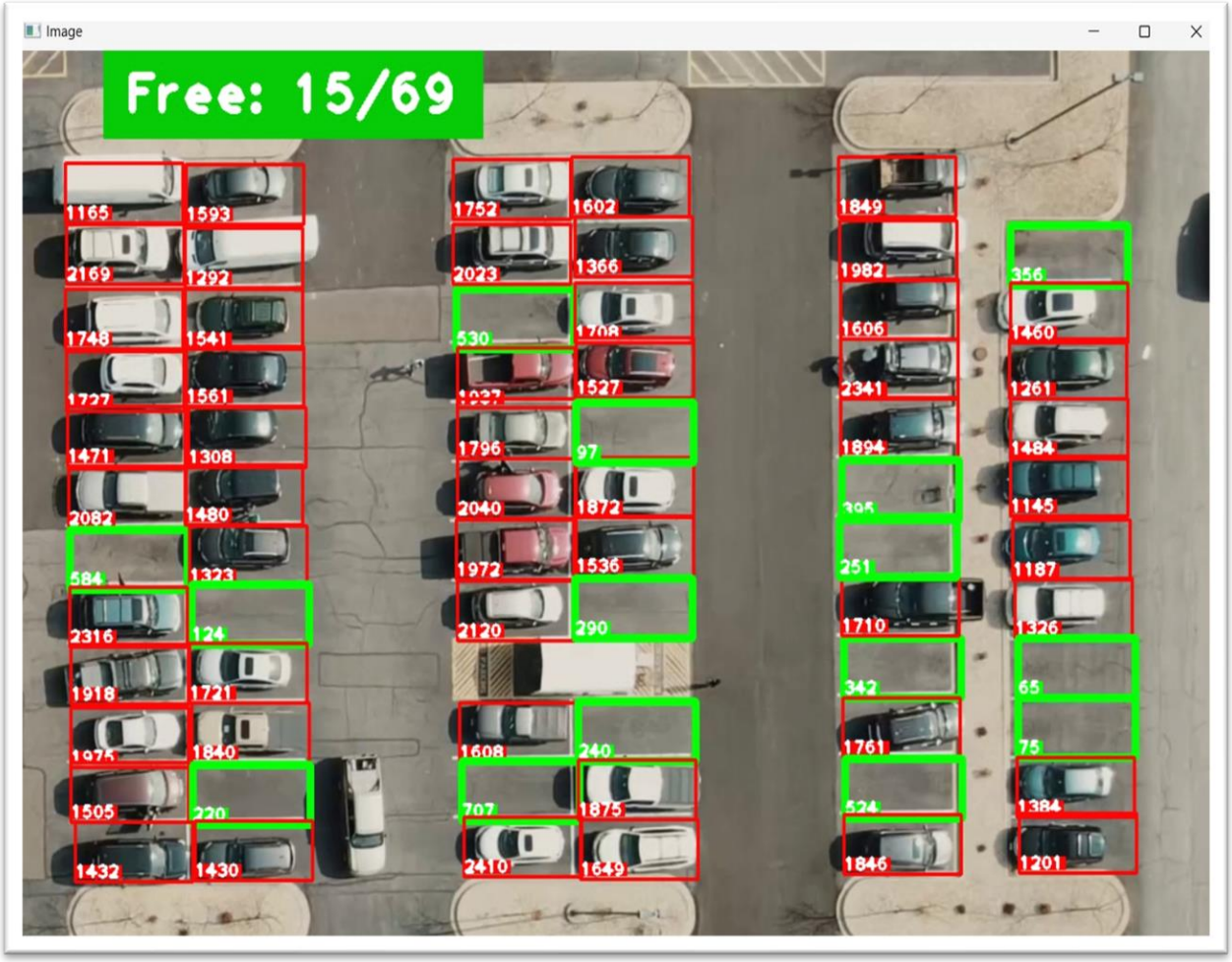
6. RESULT

The above image displays initial state of the plot where we will be selecting the parking plot manually which are represented with the purple rectangular box.

Now we are converting original colour photo to the black and white form an intermediary step before grey scale encoding.

Now as we can see image is in the grey scale visualization where occupied plots are represented with a white colour which depicts there is a car in plot, also there are some blank spaces where plot is vacant which means plot is empty.

With the help of those pixel values we can conclude how many are vacant to park next car or not.



The image/video is the final output which states wherever it is green rectangular box, it depicts plot is free. Whereas red plot depicts the plot is occupied with a car.

Also, we have displayed a counter which directly shows how many plot are vacant.

7. ADVANTAGES & DISADVANTAGES

Advantages:

- **Automatic Car Detection:** OpenCV, coupled with AI techniques, enables automatic car detection without the need for manual intervention. This reduces the reliance on human operators and streamlines the parking process.

- **Real-time Monitoring**: The system can provide real-time monitoring of parking spaces, allowing users to quickly identify available parking spots and reduce the time spent searching for parking.

- **Improved Efficiency:** By accurately tracking occupancy status and providing real-time updates, the system improves parking lot efficiency by optimizingspace utilization and reducing congestion.

- **Cost-effective:** Implementing an AI-enabled car parking system with OpenCV can be cost-effective compared to other advanced sensor-based solutions. OpenCV is an open-source library, and it can be combined with affordable cameras, making it a more accessible option.

- **Scalability:** The system can be scaled up to handle larger parking lots or multiple parking areas with ease. Adding more cameras and processing power allows for expansion without significant infrastructure changes.

Disadvantages:

- **Reliance on Camera Quality:** The accuracy of car detection and occupancy monitoring is heavily dependent on the quality of the camera feed. Poor lighting conditions, camera angles, or low-resolution images can impact the system's performance.

- **Sensitivity to Environmental Factors:** External factors such as weather conditions, shadows, and occlusions can affect the accuracy of car detection. Changes in lighting conditions or unexpected obstructions maylead to false positives or false negatives.

- **Training and Optimization:** Developing an effective car detection model and optimizing it for a specific parking lot or environment requires expertise in computer vision and machine learning. Training and fine-tuning the model can be time-consuming and resource-intensive.

- **Maintenance and System Upgrades:** Regular maintenance of cameras and system components is necessary to ensure consistent performance. Upgrading hardware or software may also be required to keep up with advancements in computer vision technology.

- **Privacy Concerns:** The use of cameras for car detection raises privacy concerns. It is crucial to implement appropriate measures to protect the privacy of individuals using the parking lot and comply with applicable data protection regulations.

8. APPLICATIONS

AI-enabled car parking systems have several potential applications that can improve efficiency, convenience, and safety in parking management. Here are some examples:

- **Automated Parking Space Detection:** AI algorithms can analyze video feeds from cameras installed in parking lots or garages to detect available parking spaces. This information can be relayed to drivers in real-time, helping them locate vacant spots quickly and reduce congestion.

- **Dynamic Pricing:** AI can optimize parking fees based on factors such as demand, time of day, location, and special events. By analyzing historical data and current conditions, AI algorithms can adjust pricing to incentivize efficient use of parking spaces and maximize revenue for parking operators.

- **Parking Guidance Systems:** AI can be used to develop smart parking guidance systems that direct drivers to available parking spots. Using sensors and real-time data, the system can guide drivers through the parking facility, reducing the time spent searching for a parking space and improving overall traffic flow.

- **Smart Payment Systems:** AI can simplify payment processes by automating ticketless or cashless payment systems. Using license plate recognition and mobile payment integration, drivers can enter and exit parking facilities seamlessly without the need for physical tickets or cash transactions.

9. CONCLUSION

This study's main beneficence is to perfect the unearthing of open parking spaces in an expenditure to ease parking arena slowdown. The development of machine learnedness and vision- grounded technology has made it possible for motorcars to find open spaces at parking lots using affordable automatic parking systems. unborn studies can concentrate on assigning specific emplacements to customers who have afore registered with an online parking management system. The precision about the proposal algorithm is inaugurated to be 92. The outcomes demonstrates that, when the captured photos of the parking lot aren`t clear due to low lighting or overlaps, the productivity drops and the exactitude for spotting decreases. It’s noticed that the average performance is 99.5 and is remarkably high as contrasted with other parking lot finding out procedures. The effectiveness of the proposed method in some cases drops down due to the strong darkness. The ultra precision of Get image frames RGB to Gray image Do Calibration Get equals of parking spot Get fellows of car Parking spot divided into Blocks Convert Block to inverse binary Get value of connected locality to determine autos number of free and Reserved Blocks Input Live stream recording 1313 the proposed task additionally relies on the kind of camera utilized for covering the parking lot.

AI-enabled car parking systems offer a range of benefits and applications that enhance efficiency, convenience, and safety in parking management. By leveraging advanced algorithms and data analysis, these systems can detect available parking spaces, guide drivers to vacant spots, optimize pricing, predict parking demand, and enhance security. Additionally, they can facilitate seamless payment processes, improve energy efficiency, and contribute to the development of smarter and more sustainable cities. AI-enabled car parking systems have the potential to revolutionize the way we park our vehicles, improving the overall parking experience for drivers and optimizing the utilization of parking facilities.

10. FUTURE SCOPE

- **Enhanced User Experience:** AI-enabled car parking systems can further improve the user experience by integrating with mobile apps and smart devices. Features such as real-time navigation, personalized parking recommendations, and seamless payment options can enhance convenience and provide a frictionless parking experience for drivers.

- **Sustainability and Green Initiatives:** Future AI-enabled car parking systems can prioritize sustainability by incorporating features such as electric vehicle charging infrastructure, energy-efficient operations, and optimization algorithms that minimize carbon emissions. These systems can play a significant role in promoting greener transportation options and reducing the environmental impact of parking.

- **Integration with Connected Vehicles:** As vehicles become increasingly connected, AI-enabled car parking systems can leverage vehicle-to-infrastructure communication. This integration can enable vehicles to communicate their parking needs and preferences to parking systems, allowing for more efficient allocation of parking spaces and personalized parking experiences.

- **Autonomous Vehicle Parking:** With the rise of autonomous vehicles, AI-enabled car parking systems can adapt to accommodate self-driving cars. These systems can provide dedicated parking areas equipped with charging stations and specialized infrastructure to support autonomous vehicle parking and servicing.

- **Advanced Parking Space Detection:** AI algorithms can continue to improve in their ability to detect available parking spaces accurately. Advanced computer vision techniques, including object recognition and 3D mapping, can enhance the precision and reliability of parking space detection, reducing the time spent searching for parking.

11. BIBILOGRAPHY

- https://towardsdatascience.com/find-where-to-park-in-real-time-using-opencv-and-tensorflow-4307a4c3da03
- https://www.irjmets.com/uploadedfiles/paper/volume_3/issue_12_december_2021/17668/final/fin_irjmets1639555391.pdf
- https://www.researchgate.net/publication/325979046_OpenCV_and_Matlab_based_car_parking_system_module_for_smart_city_using_circle_hough_transform

APPENDIX

Source Code:

```
from flask import Flask, render_template, request, session, redirect, url_for,Response
import cv2
import pickle
import cvzone
import numpy as np
import ibm_db
import base64
import re
import subprocess


app = Flask(_name_)
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb; HOSTNAME=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hdy10337;PWD=WP4Wy7keo6BGWpL0
;", "","")
print("connected")


@app.route('/')
def project():
   return render_template('index.html')

@app.route('/index.html')
```

```
def home():
    return render_template('index.html')

@app.route('/model')
def model():
    return render_template('model.html')



@app.route('/login.html')
def login():
    return render_template('login.html')

@app.route('/aboutus.html')
def aboutus():
    return render_template('aboutus.html')

@app.route('/signup.html')
def signup():
    return render_template('signup.html')

@app.route("/signup", methods=['POST', 'GET'])
def signup1():
    msg = ''
    if request.method == 'POST':
        name = request.form["name"]
        email = request.form["email"]
        password = request.form["password"]
        insert_sql = "INSERT INTO REGISTER VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.execute(prep_stmt)
        msg = "You have successfully registered!"
    return render_template('login.html', msg=msg)

@app.route("/login", methods=['POST', 'GET'])
def login1():

    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        sql = "SELECT * FROM REGISTER WHERE EMAIL=? AND PASSWORD=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
```

```
            ibm_db.bind_param(stmt, 2, password)
            ibm_db.execute(stmt)
            account = ibm_db.fetch_assoc(stmt)
            print(account)
            if account:
                session['Loggedin'] = True
                session['id'] = account['EMAIL']
                session['email'] = account['EMAIL']
                return render_template('model.html')
            else:
                msg = "Incorrect Email/password"
                return render_template('login.html', msg=msg)
        else:
            return render_template('login.html')


    @app.route('/modelq')


    def liv_pred():
        # Video feed
        cap = cv2.VideoCapture('Dataset/carPark.mp4')
        with open('CarParkPos', 'rb') as f:
            posList = pickle.load(f)
        width, height = 107, 48
        def checkParkingSpace(imgPro):
            spaceCounter = 0
            for pos in posList:
                x, y = pos
                imgCrop = imgPro[y:y + height, x:x + width]
                count = cv2.countNonZero(imgCrop)
                if count < 900:
                    color = (0, 255, 0)
                    thickness = 5
                    spaceCounter += 1
                else:
                    color = (0, 0, 255)
                    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))
```

```
            while 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()
                if not success:
                    break
                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)
                if cv2.waitKey(1) == ord('q'):
                    break

            cap.release()
            cv2.destroyAllWindows()

            return redirect(url_for('model'))  # Return a minimal response



        if _name_ == "_main_":
            app.run(debug=True)
```