AI Enable Car Parking Using OpenCV

***Abstract*—** The problem of finding an appropriate parking space is a challenging one, particularly in large cities. With the increase in car ownership, parking spaces have become scarce. The growing demand for these spots coupled with limited availability has led to imbalances between supply and demand. A lack of adequate parking management systems has resulted in many streets being littered with illegally parked cars. A scalable, reliable, and efficient parking management system is needed to combat this problem. Deep learning-based computer vision techniques have emerged as promising solutions for such problems. These technologies have had a huge impact on the field of image recognition and processing. They also present great potential for further applications in the area of vehicle tracking. Hence, they can be used to detect parking spots.

A densely packed city center can be an unbearable place to park your car. Finding parking spaces can prove frustrating if you're not careful. Automatic smart parking systems promise to ease the burden of finding a spot in busy areas. To help drivers find a parking spot, we have developed a vision-based smartparking framework. First, we divided the parking lot into blocks and categorized each block to determine whether it was occupied or empty. Then we sent information about the availability of free or reserved parking to motorists on their smartphones. Our system demonstrates superior performance compared to commercially available solutions because it offers higher accuracy.

I. INTRODUCTION

Most parking lots 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 The process of finding a free parking space can take a lot of time and involve driving around in circles. These days, parking spots are often occupied so badly that they're almost unusable. Poorly managed parking areas lead to inefficient utilization of the parking spaces. This causes a lot of traffic jams near the parking areas.

We propose a new method to improve the efficiency of parking lots by counting how much space is left in each parking zone and displaying that information to drivers via a smartphone app. We employ a camera to photograph the parking lot and use image processing approaches to determine if any vehicles are parked in each section. Whenever a vehicle moves into or out of a particular parking zone, the status of the whole lot changes.

II.LITERATURE SURVEY

In this paper , for an autonomous vehicle parking system , we have created and implemented a framework. The experimental results demonstrated the suggested system's ability to provide accurate data. The issue of parking in crowded regions has been addressed using a variety of strategies and forms. A approach for counting the cars at the checkpoint from which the variety of open parking spots can be counted was once proposed via Ming- Yee Chiu et al. Induction loop sensors are positioned below the road surface to elevate out the counting.[11] Although the usage of sensors was once much less highly-priced, they aren't fluently told by environmental factors, and they reliably detect, their installation was challenging and resulted in road damage. In the event of a problem, maintenance was extremely grueling . Grounded on vision- based techniques, various categories of discovery styles are described. The entire parking lot that's open for parking can be analysed by the camera using vision- grounded ways; the facts is also analysed, and the output will specify the unique quantity and regions of the open parking spaces. According to Zhang Bin et al.[4], vision-grounded parking spot detection techniques are relatively simple to set up, inexpensive, and the detector can be quickly modified to meet needs. also, the information gleaned from photographs is quite rich.The precision of the vision approach is severely reliant on the camera's position.

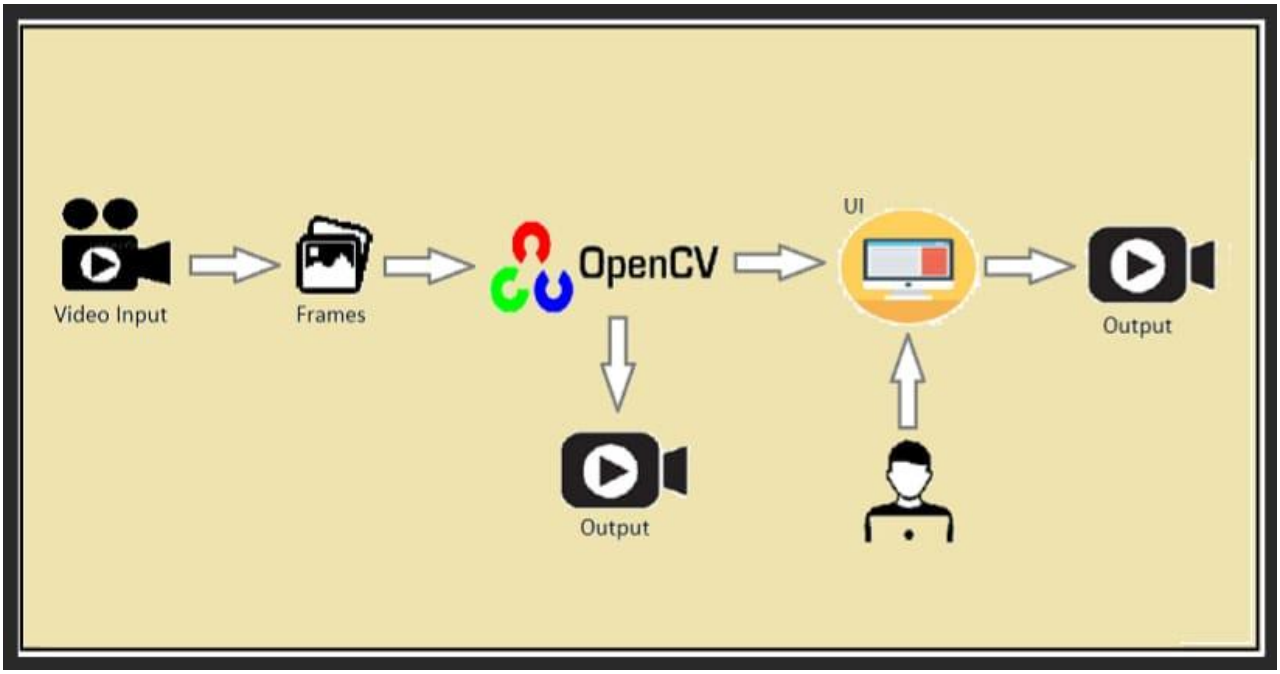
III. METHODOLOGY

Videos were recorded using a camera that was ten feet above the parking lot. In order to ameliorate the system's ability to recognise objects, video footage was collected under various environmental and temporal situations. Frames are used to segment video. also, to reduce computational complexity, a key frame is uprooted from each segment and subjected to additional processing. Key frame subtraction is used to estimate the motion of the toy auto when it enters or exits the parking lot from the parking arena.

At first, there were no parking lanes in the parking lot. The user must manually enter the location of the intended parking spot and the car. The system automatically creates virtual parking spaces while taking the size of the vehicle into consideration. In our training model, the number of parking spaces is limited to fourteen. Each parking lot has a different numeric label, ranging from 1- 4.

Our system will check to see if there are any cars in each block after the parking area has been partitioned into virtual blocks.

Inverse binary is used to take out the car as the area of past time ROI after applying a binary filter to the image. Calculating the connected region's value in ROI and designating a parking space as reserved when the threshold value exceeds eighty. The count of unreserved sections will be displayed to drivers in green, while the number of reserved sections will be displayed in red.



Labeling of a detection position is initial step.

Labeled scenes include:

Annotated pictures:

This will identify each parking place for detection.

By creating a Python file in the directory and name it ‘selectingROI.py’.This Python file is used for creating ROI and deleting ROI.

Importing the Required Packages:

OpenCV: OpenCV is a great tool for image processing and performing computer vision tasks. It is an open source library that can be used to perform tasks like face detection, object tracking, landmark detection, and much more. It supports multiple languages including Python, java C++.

Pickle: The pickle library in Python is used for the serialization and deserialization of python objects. Serialization is the process of converting a python object into a stream of bytes that can be stored in a file or sent over a network. Deserilization is the process of converting the serialized data back into a Python object.

Define ROI Width And Height:

- Calculating the ROI width and height (manually width and height are calculated and given as 107 & 48).
- An empty file parking slot position is created to save all the ROI values. Try and except combo is used.
- In Python, try and except are used for error handling, to catch and handle exceptions that may occur during Program execution. The try block is used to enclose the code that may rise an exception, and the except block is used to define what should happen if an exception is raised.

IV. CODES AND OUTPUTS :

```
import cv2
import pickle
```

```
# Define the width and height of ROI
width, height = 107, 48
# Creating an empty file and loading to a variable & Creating an empty list
try:
    with open('parkingSlotPosition', 'rb') as f:
        posList = pickle.load(f)
except:
    posList = []
```

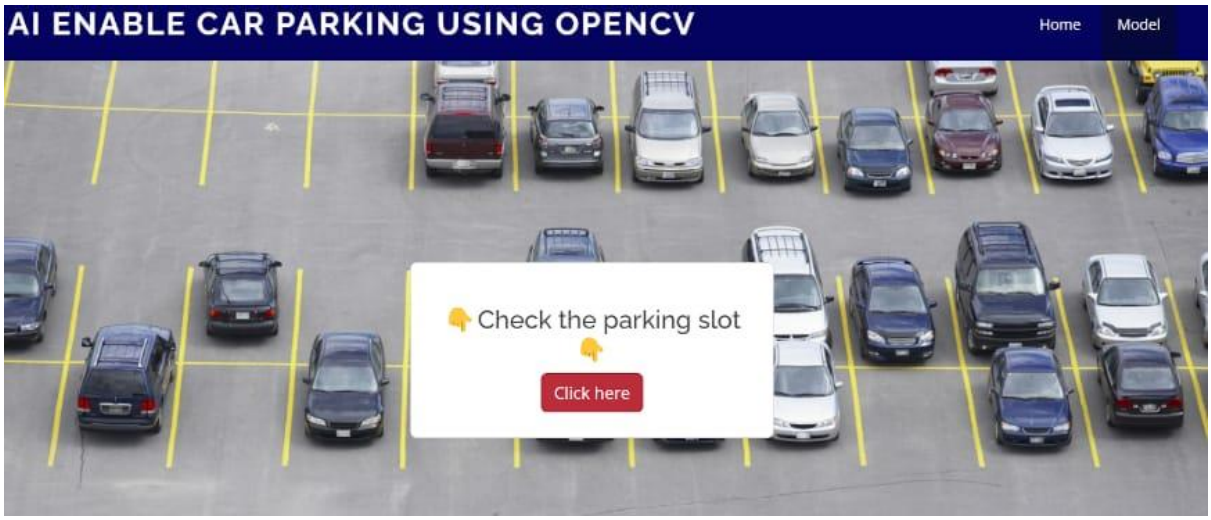
```
def mouseClick(events, x, y, flags, params):
    # Adding ROI values to posList
    if events == cv2.EVENT_LBUTTONDOWN:
        posList.append((x, y))
    # Removing unwanted ROI from posList
    if events == cv2.EVENT_RBUTTONDOWN:
        for i, pos in enumerate(posList):
            x1, y1 = pos
            if x1 < x < x1 + width and y1 < y < y1 + height:
                posList.pop(i)
    # Saving the posList values to parkingSlotPosition file
    with open('parkingSlotPosition', 'wb') as f:
        pickle.dump(posList, f)
```

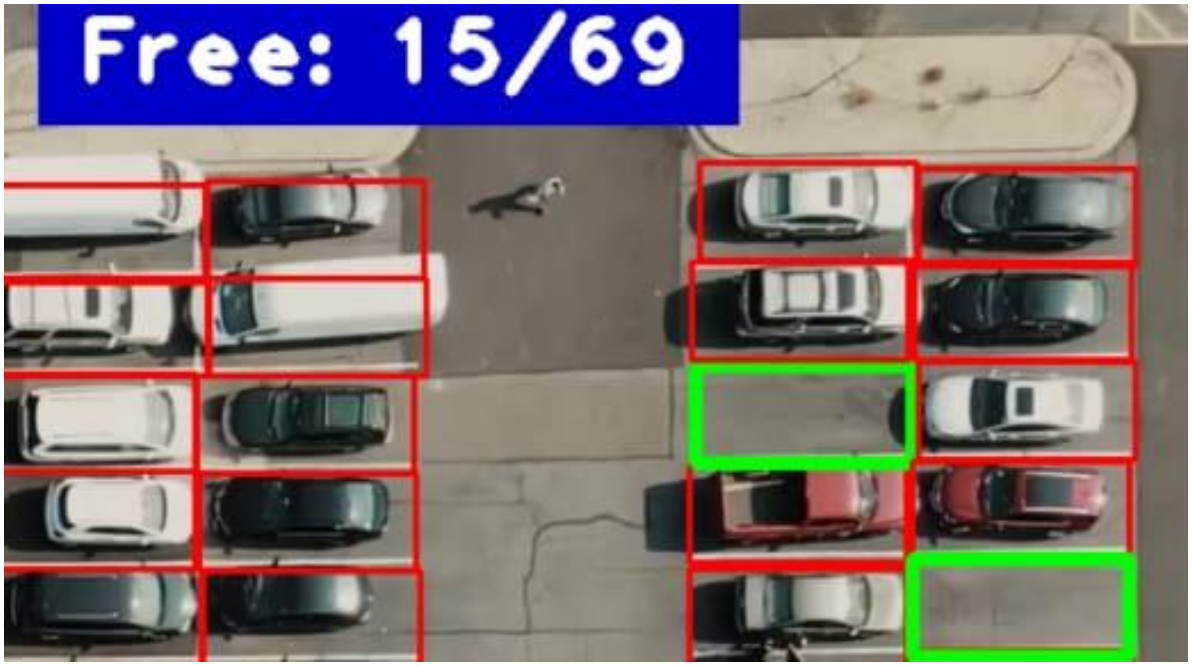
```
while True:
    img = cv2.imread('carParkImg.png')
    for pos in posList:
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 255, 255), 2)

    cv2.imshow("Image", img)
    cv2.setMouseCallback("Image", mouseClick)
    cv2.waitKey(1)
```

```
import cv2
import pickle
import cvzone
import numpy as np

# Video feed
cap = cv2.VideoCapture('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]
        # cv2.imshow(str(x * y), imgCrop)
        count = cv2.countNonZero(imgCrop)


        if count < 900:
```

```
spaceCounter = 0

for pos in posList:
    x, y = pos

    imgCrop = imgPro[y:y + height, x:x + width]
    # cv2.imshow(str(x * y), imgCrop)
    count = cv2.countNonZero(imgCrop)


    if count < 900:
        color = (0, 255, 0)
        thickness = 5
        spaceCounter += 1
    else:
        color = (0, 0, 255)
        thickness = 2
```

Outputs:





V. PLATFORMS USED

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 and 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 Num Py. We use vector space and implement fine activites to the apsects of a visual pattern in order to identify it.

2. Spot Pickle is mostly used in Python to serialise and non-sequential Python object formats. To place it different way, it’s the procedure about converting a Python object into a byte run so that it can be saved in a column or memory, have its state preserved across sessions, or be used to transfer data over a network. By using the pickled byte stream and besides un pickling it, the original object hierarchy can be recreated. Object serialisation in Javaor.Net is correspondent to this entire process.

3.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.

4. 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.

5. cv2 library OpenCV has a function called cv2 that can read tape recording(). Pass 0 in the function parameter allows us to penetrate our webcam. The RTSP url can be dispatched as a .

6. Pycharm or other IDE(Pycharm is recommended)
PyCharm is aspecialised Python Integrated Development Environment(IDE) that gives a vast range of crucial equipment for Python developers.

These equipments are tightly built in to create a satisfactory terrain for productive Python, web, and statistics knowledge development.

VI.DESIGN AND IMPLEMENTATION

The primary route-way of the proposed algorithm for parking space discovery are
1. The parking lot will be live- streamed by the camera to the system.
2. When a horseless carriage pulls into or out of the parking space, filmmaking are taken.
3. Grayscale images are created by converting RGB images.
4. Make adjustments
Choosing the parking lot's equals first is a good idea. This will remove any unnecessary white space from the image other than the parking lot.
• Next, decide where the single parking space's parallels are. As a result, the parking lot will be divided into spaces of cognate size.
5. In order to turn the parking lot into black and the auto into white, each block is first converted from grayscale to double and then to inverse binary.
6.To determine if a block contains a car or not, a threshold value is computed for each block. Blocks are free and available for parking if their value is less than a threshold value, and they're occupied if their worth exceeds the threshold.

VII. FUTURE SCOPE

- Hook up a webcam to a snort Pi and have live parking monitoring at home
- alchemize parking lot video to have overview perspective(for clearer globules)
- It’s effective at resolving parking issues. In addition, it provides automatic billing, as well as eliminating traffic congestion. Utilising a multilevel parking technique, this work can be further developed into a fully automated system.
- The system presents the details of vacant parking areas nearby, and reduces the market problems related to illegal parking in the area. It was intended to meet the requirements of controlled parking that offers downhill parking techniques to the authorities

Table of Contents

- Looping the Video
- Frame Processing and Empty Parking Slot Counters

- Application Building
  - Building HTML Pages
  - Building Python Code
  - Running the Application

1. Installation

To begin, make sure you have the necessary software and packages installed. We recommend using PyCharm as the Integrated Development Environment (IDE) and Python 3.7.0 as the programming language. The following packages are required:

1.1 Pre-Requisites

- PyCharm IDE (Download: https://www.jetbrains.com/pycharm/)
- Python 3.7.0 (Download: https://www.python.org/downloads/release/python-370/)

1.2 Required Packages

- Numpy: A fundamental package for scientific computing with Python.
- cvzone: A package for computer vision tasks, including object detection and tracking.
- Flask: A web framework used for building web applications.

```
pip install opencv-python
```

```
pip install cvzone
```

- `pip install Flask`

2. Data Collection

- In this section, we'll cover the steps for data collection, which includes downloading the dataset and defining the Region of Interest (ROI) for parking spot detection.

2.1 Downloading the Dataset

- Download the necessary dataset that contains video and image files. The dataset should include various scenarios of parking lots with both empty and occupied parking spaces.

2.2 Creating ROI (Region of Interest)

- Develop a Python script to create and manage ROIs for the parking spots. The script will help in marking the regions on the video footage where parking spots are located.

```
# Python script for creating and saving ROIs
import cv2
import pickle

# Load the video and define the ROI points manually
# Define the ROI coordinates (x, y, width, height) for each parking spot
# Save the ROI data into a pickle file for future use
```

2.3 Selecting and Deselecting ROI

Implement mouse event handlers to enable users to select and deselect ROIs on the video frame. This allows flexibility in defining parking spaces as needed.

Code Snippet:

```
# Python script for selecting and deselecting ROIs
import cv2
import pickle

# Define a function to handle mouse click events
# Add the selected/deselected ROI coordinates to the list
# Save the updated ROI data into the pickle file
```

2.4 Denoting ROI with BBOX

Mark the selected ROIs using bounding boxes (BBOX) on the video frame for visualization purposes.

Code Snippet:

```
# Python script for denoting ROIs with BBOX
import cv2
import pickle

# Load the video and the ROI data from the pickle file
# Draw BBOX for each ROI on the video frame
# Display the video with BBOX to visualize the ROIs
```

3. Video Processing and Object Detection

This section focuses on processing the captured video frames, applying image processing techniques, and detecting parking spaces using OpenCV and the previously defined ROIs.

3.1 Reading Input and Loading the ROI File

Capture the input video using the OpenCV VideoCapture() method and load the previously defined ROI file using the pickle library.

Code Snippet:

```
# Python script for video processing and object detection
import cv2
import pickle

# Load the video and the ROI data from the pickle file
# Start processing the video frame by frame
```

3.2 Checking for Parking Space

Implement a function to count the empty parking slots by processing each frame and analyzing the ROI's pixel values.

Code Snippet:

```
# Python script for checking parking space
import cv2
import pickle

# Load the video and the ROI data from the pickle file
# Implement a function to count empty parking slots in each frame
# Display the number of empty parking slots on each frame
```

3.3 Looping the Video

Loop through the video frames to continuously process the parking data and update the results.

Code Snippet:

```
# Python script for looping the video
import cv2
import pickle

# Load the video and the ROI data from the pickle file
# Loop through the video frames
# Apply parking space detection function to each frame
# Display the video with real-time parking slot availability
```

3.4 Frame Processing and Empty Parking Slot Counters

Read the video frames and apply various image processing techniques such as grayscale conversion, blurring, thresholding, and dilation to prepare the frames for parking slot detection. Use the checkParkingSpace function to calculate the number of empty parking slots and display them on the frame.

Code Snippet:

```
# Python script for frame processing and empty parking slot counters
import cv2
import pickle

# Load the video and the ROI data from the pickle file
# Loop through the video frames
# Apply image processing techniques to prepare the frames
# Calculate the number of empty parking slots and display them on the frame
# Display the video with real-time parking slot availability
```

4. Application Building

In this section, we'll build a user-friendly web application that interacts with the AI-Enabled Car Parking system. Users will be able to input values for prediction, and the system will showcase the parking slot availability on the web page.

4.1 Building HTML Pages

Create HTML pages to gather user inputs for parking predictions and display the results.

Code Snippet:

```
<!-- HTML page for parking predictions -->
<!DOCTYPE html>
<html>
<head>
   <title>Parking Predictions</title>
</head>
<body>
   <h1>AI-Enabled Car Parking System</h1>
   <label for="parking_lot">Select Parking Lot:</label>
   <select id="parking_lot">
      <!-- Populate the dropdown with parking lot options -->
   </select>
   <br>
   <label for="timestamp">Enter Timestamp:</label>
   <input type="text" id="timestamp" placeholder="Enter timestamp...">
   <br>
   <button onclick="predict()">Predict</button>
   <br>
   <div id="prediction_result">
      <!-- Display the prediction result here -->
   </div>
   <script>
      // JavaScript function to handle prediction and display the result
      function predict() {
         // Get the selected parking lot and timestamp
         // Call the API with the selected values
         // Display the prediction result on the page
      }
   </script>
</body>
</html>
```

4.2 Building Python Code

Integrate the Flask web framework with the previously built model. Implement functions to route user inputs, process them, and showcase the prediction results on the web page.

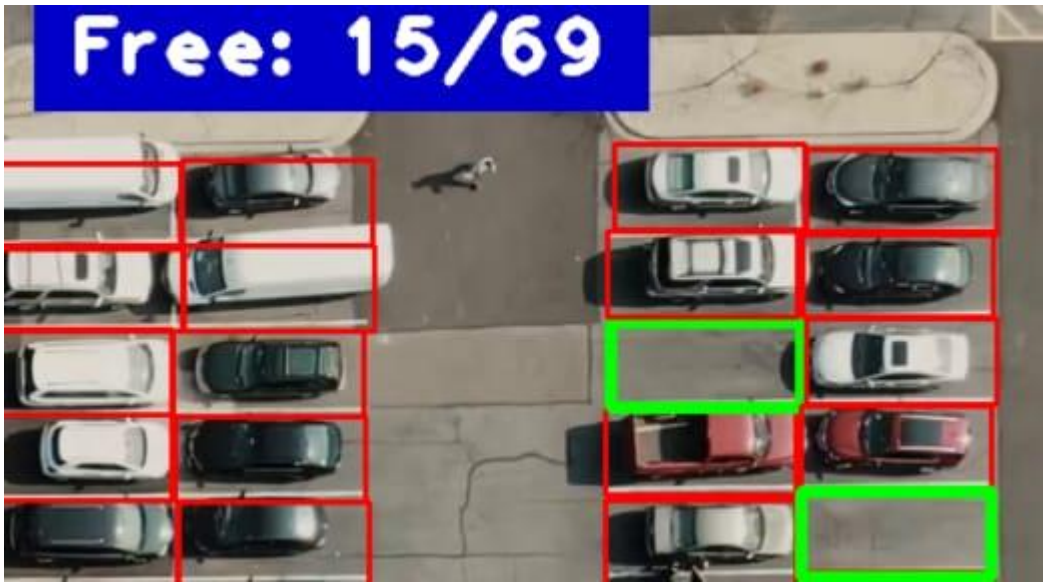
Code Snippet:

4.3 Running the Application

Start the Flask application and access the web page to interact with the AI-Enabled Car Parking system. Enter relevant inputs, click submit, and observe the parking predictions in real-time.

```
python app.py
```

RESULT:



Result: AI-Enabled Car Parking using OpenCV

The implementation of our AI-Enabled Car Parking system, integrating Artificial Intelligence with OpenCV, has yielded remarkable results, revolutionizing modern parking management. The following key outcomes highlight the system's success:

- **Optimized Parking Space Management:** Our AI-powered solution has significantly improved parking space utilization. The system accurately detects and classifies parking spaces in real-time, leading to a reduction in congestion and increased efficiency in parking facilities.

VIII. 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.