IBM NAAN MUDHALVAN INTERNET OF THINGS

Phase 2:

Innovation

Topic:

Smart parking

Team members:

M. Sabitha jones(922121106075)

A.Santhi (922121106079)

B.Sathyadevi(922121106084)

K. Varnigadevi (922121106102)

M. Varsha (922121106103)

College name:SSM Institute Of Engineering and Technology.

College code:9221

INNOVATIVE SOLUTION THAT SOLVE PARKING PROBLEMS:

COMMON PROBLEMS :DIFFICULTIES FINDING PARKING

As touched upon, many cities have trouble with creating adequate parking and are seeking innovative parking solutions. If you live in or have visited a major urban area, you are probably already familiar with the common need to drive around in circles after reaching a destination just to find an available parking space. When you do locate one, it could be far from your destination building, meaning that you must walk several blocks from car to building and back.

Smart parking innovations – look beyond parking

In spotting the real city innovation opportunities with smart parking, consider what data is collected by the sensors. Then analyze how it can be used along one or more innovation paths (technology, product, services, customer experience and business model).

INNOVATIONS IN SMART PARKING SOLUTIONS

1.Tracking cars with sensor systems

IoT is the technology at the core of vehicle tracking platforms. Tools like GPS or OBD sensors help collect location data on a car or a fleet and monitor the occupancy of parking spaces. This information is transferred to the cloud gateway, processed and sent to the network server. The data will be presented to drivers and car company managers in the form of understandable, clear insights. As for now, IoT-based vehicle trackers are mainly used large-scale corporate organizations for fleet management. In the future, when the release of 5G makes the Internet of Things more accessible, parking lot technology will spread among car drivers and will be used to manage daily commutes and mitigate parking challenges.

2. Smart counter systems

A connected counting system can detect when a vehicle enters and leaves the parking facility. This way, IoT-based platforms will be able to offer drivers a real-time counter of available spots.Facility managers can use counter systems to increase the efficiency of the parking facility, determine trends and patterns regarding the customer flow, and be able to predict future vehicle surges.

3. Automated parking systems

Automatic parking systems help reduce the land use for parking and maximize the efficiency of space usage. An automated system is used to move cars up and down to the upper levels of the facility.Since APS facilities are fully automated and have restricted access, parking a vehicle there is more secure. Automatic parking systems help reduce the parking search time, along with engine emissions that accumulate due to the increase in driving time.The resource usage within such a facility is minimized as there is little light and ventilation needed to maintain an automated parking system.

4. Control systems

IoT contributes to urban safety and order by acting as a powerful traffic compliance reinforcer.Thanks to a network of sensors and fast data processing algorithms, parking control systems can detect parking rules violations, register them, collect and store the neededevidence, issue a ticket, and notify the violating party in a matter of seconds.

THE ROLE OF IOT IN PARKING SOFTWARE TECHNOLOGIES

Smart parking will have an outstanding impact on all stakeholders involved in the process. Drivers will be able to book parking spaces beforehand, plan trips and commutes with the lot occupancy in mind. Reinforcement agencies will be able to detect and evaluate the gravity of parking rules violations in split seconds.Parking facility managers will be able to optimize the use of space and resources within their parking lots, efficiently strategize and plan future development. Community leaders will increase the comfort of city residents by implementing IoT parking solutions.

Here are a few additional benefits IoT offers in the parking sector:

- Sponsored meter time extension

Connected platforms will notify drivers when the parking meter is about to expire. Such tools will help extend the parking time duration in one click as soon as the driver paid for an extension. Automated parking meter extension systems will reduce the number of traffic law violation cases and increase the revenue of the facility.

- Innovative parking solutions that identify the safety of parking spots.

The ‘red’ zones like bus stops, passenger loading-unloading areas, and parking spots for handicapped people will be identified by the platform and alerted to the driver. This way, the number of parking violations by negligence will be reduced. If a driver still parked a vehicle in a no-parking zone, a connected platform will instantly notify a reinforcement department about the violator, increasing the odds of penalizing the violation successfully.

- Efficient citywide parking space utilization

automated parking planners can collect occupancy data for all parking venues thanks to a network of connected sensors. To disperse the number of parked cars equally throughout the city, municipal communities can adjust meter rates and allowed parking times using the data provided by IoT platforms in the decision-making process.

Program for innovation ideas in smart parking

```
Import cv2

# Load a pre-recorded video or connect to a camera feed

Cap = cv2.VideoCapture('your_camera_feed.mp4')  # Use 0 for
webcam

# Load a pre-trained car detection classifier (you might need a
custom model)

Car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')

While True:

    Ret, frame = cap.read()

   If not ret:

        Break

  # Convert the frame to grayscale for faster processing

    Gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

   # Detect cars in the frame
```

```
    Cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30)
    For (x, y, w, h) in cars:
        # Draw rectangles around detected cars
        Cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
        # Display the frame with car detections
    Cv2.imshow('Parking Space Availability Detection', frame)
       If cv2.waitKey(1) & 0xFF == ord('q'):
        Break
Cap.release()
Cv2.destroyAllWindows()
```

Explanation for above coding:

Certainly! Let's break down the Python code provided for car detection using OpenCV:

1.Import necessary libraries:

~~~
   ```python
   Import cv2
   ```
~~~

This code imports the OpenCV library, which is used for computer vision tasks.

1. Load the video source:

~~~
   ```python
   Cap = cv2.VideoCapture('your_camera_feed.mp4')  # Use 0 for
webcam
~~~

``` Here, a video capture object (`cap`) is created. You can specify the video source by providing the filename of a video file (e.g., `'your_camera_feed.mp4'`) or use the webcam by setting it to `0`.

2. Load a pre-trained car detection classifier:

```python
Car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
```

This line loads a pre-trained Haar Cascade classifier for car detection. In practice, you might need a custom model or dataset for more accurate results.

3. Start a loop to process frames:

```python
While True:
```

This `while` loop continuously captures and processes frames from the video source until you press the 'q' key.

4. Read a frame from the video source:

```python
Ret, frame = cap.read()
```

It reads a frame from the video source and stores it in the `frame` variable. `ret` is a flag indicating whether the frame was successfully read.

5. Convert the frame to grayscale:

```python

```
Gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Converting the frame to grayscale simplifies the processing and speeds it up.

6. Detect cars in the frame:

```python
Cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
```

The Haar Cascade classifier is applied to the grayscale frame to detect cars. Parameters like `scaleFactor`, `minNeighbors`, and `minSize` control the detection process.

7. Draw rectangles around detected cars:

```python
For (x, y, w, h) in cars:
    Cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

For each detected car, a red rectangle is drawn around it. This helps visualize the detected cars in the frame.

8. Display the frame with car detections:

```python
Cv2.imshow('Parking Space Availability Detection', frame)
```

This line displays the frame with car detections in a window titled 'Parking Space Availability Detection'.

9. <u>Check for the ‘q’ key press to exit:</u>

```python
If cv2.waitKey(1) & 0xFF == ord(‘q’):
    Break
```

This code waits for a key press, and if the ‘q’ key is pressed, it breaks out of the loop and ends the program.

10. <u>Release video capture and close windows:</u>

```python
Cap.release()
Cv2.destroyAllWindows()
```

Finally, it releases the video capture object and closes any OpenCV windows. This code provides a basic example of car detection in a video feed using OpenCV’s Haar Cascade classifier. For a more advanced and accurate parking space availability detection system, more sophisticated techniques and models may be needed.