CHAPTER 1
INTRODUCTION

There are many definitions for a smart city. A simple definition is that a smart city incorporates numerous technologies to provide a better life for citizens and improve the quality of services that are provided by the government. One of the goals of a smart city is to optimize all available resources for better civic life. It has been argued that simply providing more technologies creates a smart city, but the real value of a smart city lies in how these technologies are used. The readiness of infrastructures, environmental initiatives, and the ability of people to be able to thrive are crucial characteristics of a smart city. These initiatives, including the development of waste management and smart parking systems, can be extremely beneficial in reaching the aforementioned objectives.

The government seeks reliable solutions for the complex issues and challenges associated with the development and execution of its plan. Transportation management has become a particularly important issue in the present. The number of automobiles has increased rapidly in recent years, which has raised concerns about fuel emission. The issue requires urgent solutions that provide environmental sustainability and reduce the impact of climate change. Searching for parking spaces leads to unnecessary fuel consumption, which harms the environment.The wasted time spent parking can also cause frustration for drivers, especially in big and busy cities.

Cities need feasible and reliable smart parking systems. Several solutions for smart parking have been developed to utilize available parking spaces in a city more effectively. These systems are aimed at providing cost-effective solutions that will significantly reduce and minimize human labour. This contributes to the problem by developing and presenting a novel approach for a smart car parking system that can be efficiently managed. This method is very cost-effective since there is no need to install sensors; it just requires one camera to provide real-time information about the parking area, either in images or video streams. These images or video streams are processed to identify the total number of vehicles in the parking area and to verify whether there are vacant spaces. Then, instructions are provided to drivers about the vacant spaces and where to park their vehicles.

CHAPTER 2
PROPOSED SYSTEM

It contained two stages, one for scanning the parking area and another for securing data. The method showed the status of the parking area to drivers and motorists before entering and provided directions to available parking spaces. The hashing method was utilized to secure the system itself. A Raspberry Pi microcontroller was utilized to detect vehicles. An SD card and a board module were also employed to increase the rate of reliability of the implemented method. This utilized a 32- byte hashing key for authenticating purposes. In addition, no sensors were used to reduce the cost. It performed only two scenarios while four scenarios were conducted using the proposed system. The proposed system starts by taking data from the installed camera which provides real-time information. These images or video streams are fed into the algorithm, which starts the processing.

If no vacant spaces are available, the system displays a message saying that the parking lot is fully occupied, and drivers must search for alternatives. The gate at the entrance remains closed. If there are empty spaces, the system opens the gate and provides directions to vacant spots where drivers may park their vehicles. The system additionally displays the total number of vehicles currently parked and the number of vacant spaces. The proposed algorithm is demonstrated as Very cost-effective, environmentally friendly since it decreases fuel consumption, power, and energy dissipated, Easy to maintain and operate. It can be integrated with other systems easily.

Additional features such as checking the parking area ahead of arrival, booking a vacant space, and adding billing components can be added with minor changes in the system design. The proposed system requires fewer operations to run and manage its features; thus, its execution time is less when compared to other methods in the literature. Further more, it is cost-effective as it utilizes only a single camera that sends its data in real-time to the connected machine. This system is intelligent since it employs the convolutional neural network tool to train itself and adapt to any type of conditions and circumstances that may exist such as a traffic accident or car failure inside the parking lot or a vehicle that parks in the driveway which could lead to a deadlock.

2.1 FLOW CHART

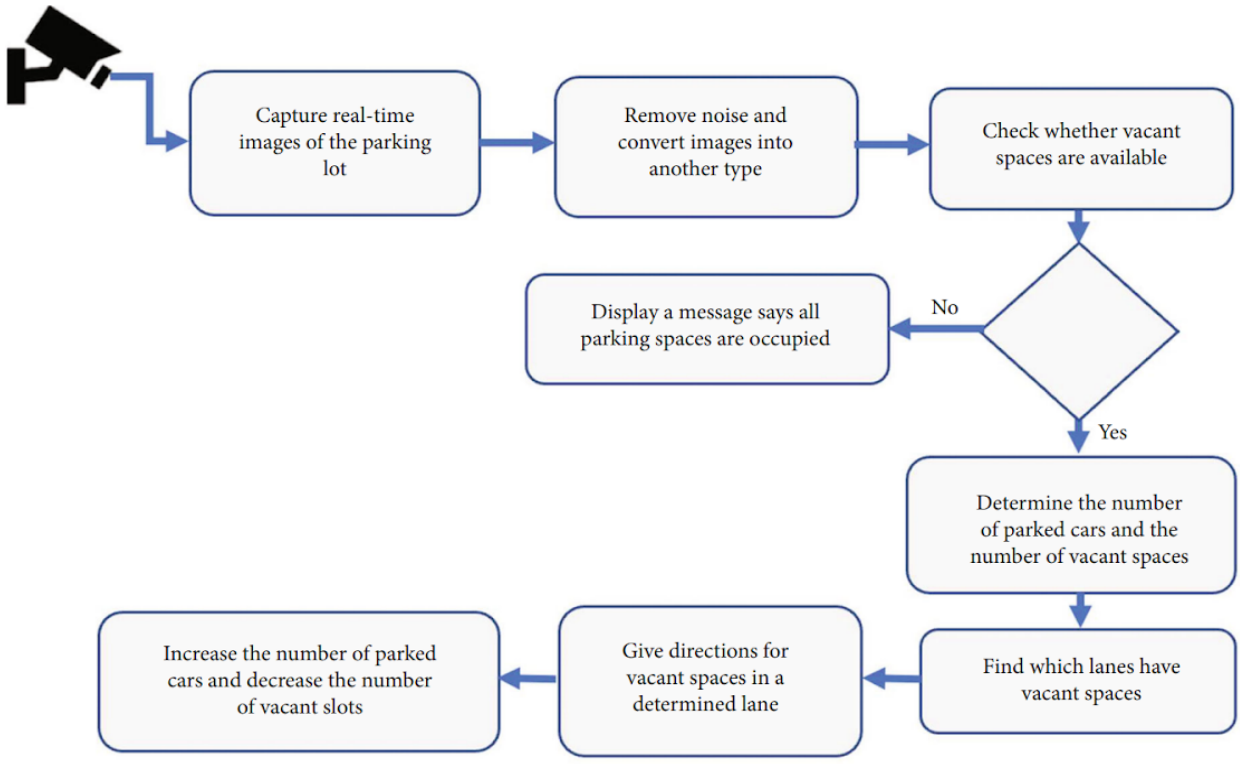

Fig 2.1 flow chart of the proposed system

Numerous simulation experiments were carried out in MATLAB to verify the workflow of the developed algorithm and test its accuracy and efficiency. In these two figures, vehicles are represented as X during the simulation runs, where X can refer to a vehicle of any size or a motorcycle. Four different scenarios are presented in this article to demonstrate the effectiveness of the proposed approach. Scenario 1 represents a full parking lot in which there is no vacant space while Scenario 2 demonstrates a scenario where there are vacant spaces in one of the lanes and a driver is directed there.

Since the proposed algorithm in this article only requires one camera to operate and provide its outputs, it is incredibly cost-effective. Other developed works in the literature reviewed above use IoT sensors, a far more expensive technology, to detect vacant spaces. In contrast, because CAMs are so inexpensive, the method that is being provided here costs practically nothing. relies on Internet or Wi-Fi services which is a disadvantage.

2.2 BLOCK DIAGRAM

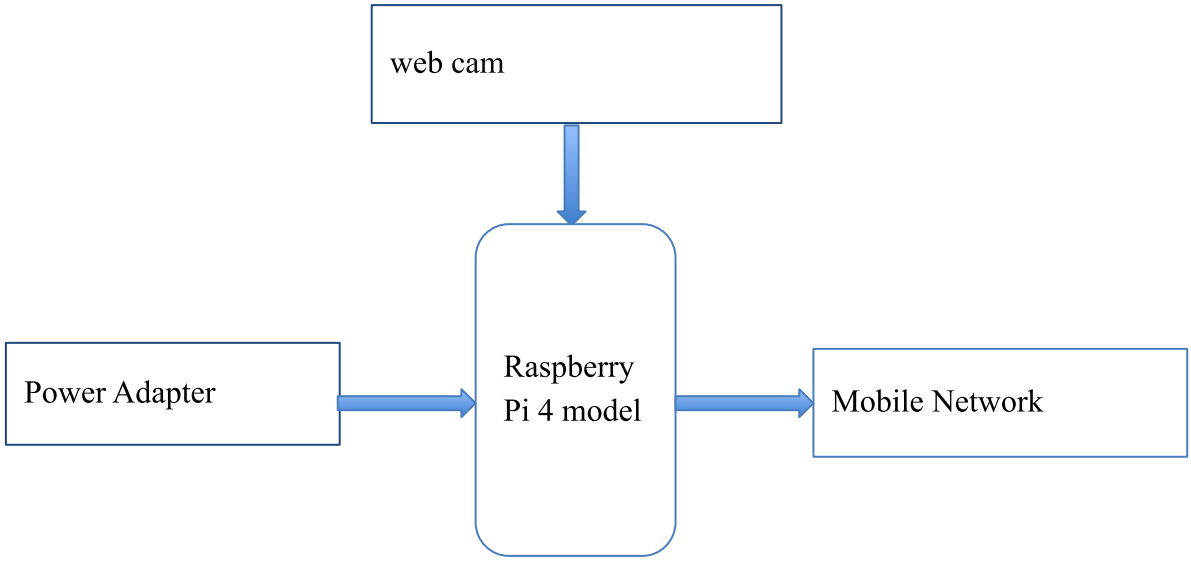

Fig 2.2 block diagram of the proposed system

The proposed system starts by taking data from the installed camera which provides real-time information. These images or video streams are fed into the algorithm, which starts the processing. If no vacant spaces are available, the system displays a message saying that the parking lot is fully occupied, and drivers must search for alternatives. The gate at the entrance remains closed. If there are empty spaces, the system opens the gate and provides directions to vacant spots where drivers may park their vehicles. The system additionally displays the total number of vehicles currently parked and the number of vacant spaces.

This system is intelligent since it employs the convolutional neural network tool to train itself and adapt to any type of conditions and circumstances that may exist such as a traffic accident or car failure inside the parking lot or a vehicle that parks in the driveway which could lead to a deadlock. Once an abnormal activity inside the parking area is detected, then the considered team such as the administration team is alerted and notified to take proper action. Since the implemented system depends and relies on a single camera, thus its fault tolerance can be considered low. This drawback is resolved by having a backup camera that comes up immediately once the system detects no inputs from the main camera for a long time. An alternative way is to rely on a human intervention to take control manually if the installed camera goes down for any reason.

2.3 HARDWARE USED

Raspberry pi 4 model

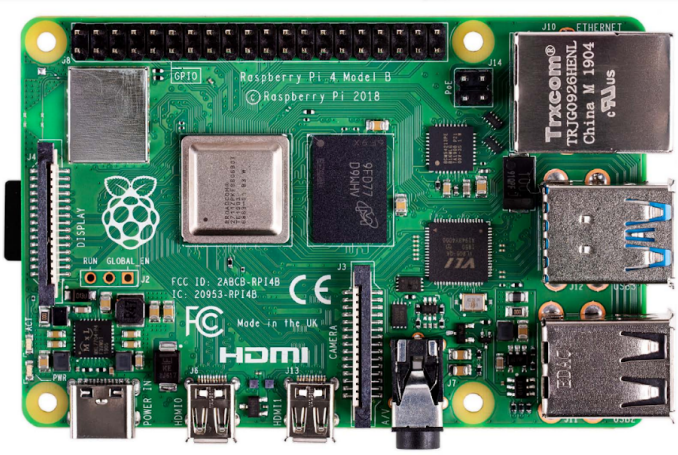

Fig 2.3 Raspberry Pi 4 model

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market. Raspberry Pi can be used as a Wi-Fi router. This offers the system the opportunity to interface with the peripherals in the system, which need higher data rates and don't have a power concern, securely and easily. It is then proposed that Wi-Fi be used for full-duplex communications with the ANPR camera, and between the barrier/signage to the Central Processing Hub.

Raspberry Pi is known as a single-board computer, which means exactly what it sounds like: it's a computer, just like a desktop, laptop, or smartphone, but built on a single printed circuit board. Like most single-board computers, Raspberry Pi is small roughly the same footprint as a credit card but that doesn't mean it's not powerful: a Raspberry Pi can do anything a bigger and more power hungry computer can do, though not necessarily as quickly. The Raspberry Pi family was born from a desire to encourage more hands-on computer education around the world. Its creators, who joined together to form the non-profit Raspberry Pi Foundation, had little idea that it would prove so popular: the few thousand built in 2012 to test the waters were immediately sold out, and millions have been shipped all over the world in the years since. These boards have found their ways into homes, classrooms, offices, data centres, factories, and even self-piloting boats and spacefaring balloons.

Various models of Raspberry Pi have been released since the original Model B, each bringing either improved specifications or features specific to a particular use case. The Raspberry Pi Zero family, for example, is a tiny version of the full-size Raspberry Pi which drops a few features in particular the multiple USB ports and wired network port in favour of a significantly smaller layout and reduced power requirements. All Raspberry Pi models have one thing in common, though: they're compatible, meaning that software written for one model will run on any other model. It's even possible to take the very latest version of Raspberry Pi's operating system and run it on an original prelaunch Model B prototype. It will run more slowly, it's true, but it will still run.

Raspberry pi's components

Like any computer, Raspberry Pi is made up of various components, each of which has a role to play in making it work. The first, and arguably most important, of these can be found just above the centre point on the top side of the board (Figure 2.4), covered in a metal cap: the system-on-chip (SoC). The name system-on-chip is a great indicator of what you would find if you prised the metal cover off: a silicon chip, known as an integrated circuit, which contains the bulk of Raspberry Pi's system. This includes the central processing unit (CPU), commonly thought of as the 'brain' of a computer, and the graphics processing unit (GPU), which handles the visual side of things.

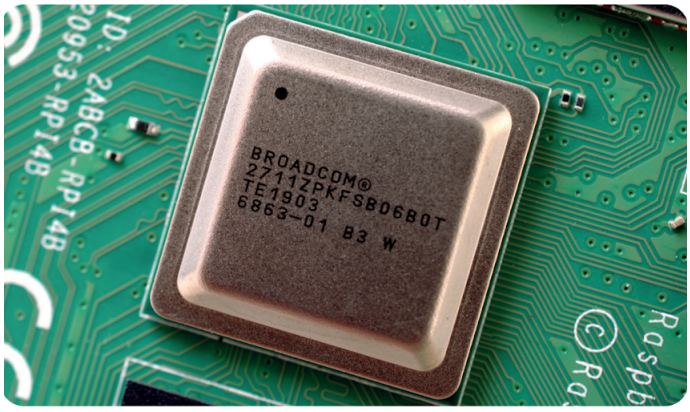

Fig 2.4 Raspberry Pi's system- on- Chip (SoC)

A brain is no good without memory, however, and just to side of the SoC you'll find exactly that: another chip, which looks like a small, black, plastic square (Figure 2.5, overleaf). This is Raspberry Pi's random access memory (RAM). When you're working on Raspberry Pi, it's the RAM that holds what you're doing; only when you save your work will it be written to the microSD card. Together, these components form Raspberry Pi's volatile and non-volatile memories: the volatile RAM loses its contents whenever Raspberry Pi is powered off, while the non-volatile microSD card keeps its contents.

Fig 2.5 Raspberry Pi's random access memory (RAM)

A top right of the board you'll find another metal lid (Figure 2.6) covering the radio, the component which gives Raspberry Pi the ability to communicate with devices wirelessly. The radio itself acts as two main components, in fact: a WiFi radio, for connecting to computer networks; and a Bluetooth radio, for connecting to peripherals like mice and for sending data to or receiving data from nearby smart devices like sensors or smartphones.

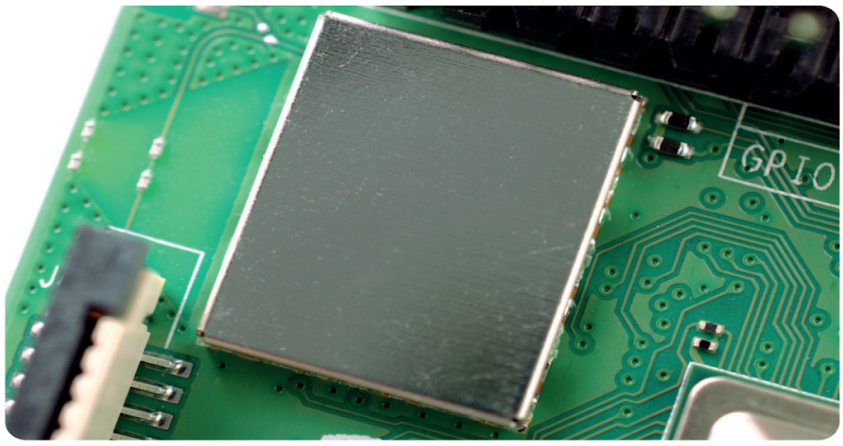

Fig 2.6 Raspberry Pi's radio module

Another black, plastic-covered chip can be seen to the bottom edge of the board, just behind the middle set of USB ports. This is the USB controller, and is responsible for running the four USB ports. Next to this is an even smaller chip, the network controller, which handles Raspberry Pi's Ethernet network port. A final black chip, smaller than the rest, can be found a little bit above the USB Type-C power connector to the upper-left of the board (Figure 2.7); this is known as a power management integrated circuit (PMIC), and handles turning the power that comes in from the micro USB port into the power Raspberry Pi needs to run.

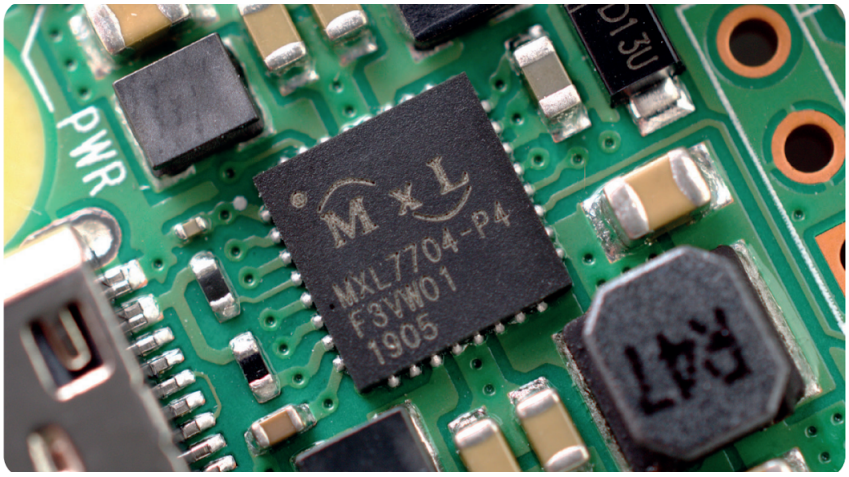

Fig 2.7 Raspberry Pi's Power management integrated circuit (PMIC)

Raspberry Pi's Ports

Raspberry Pi has a range of ports, starting with four Universal Serial Bus (USB) ports (Figure 2.8) to the middle and right-hand side of the bottom edge. These ports let you connect any USB-compatible peripheral, from keyboards and mice to digital cameras and flash drives, to Raspberry Pi. Speaking technically, there are two types of USB ports: the ones with black parts inside are USB 2.0 ports, based on version two of the Universal Serial Bus standard; the ones with blue parts are faster USB 3.0 ports, based on the newer version three.

Fig 2.8 Raspberry Pi's USB ports

To the right of the USB ports is an Ethernet port, also known as a network port (Figure 2.9). You can use this port to connect Raspberry Pi to a wired computer network using a cable with what is known as an RJ45 connector on its end. If you look closely at the Ethernet port, you'll see two light-emitting diodes (LEDs) at the bottom; these are status LEDs, and let you know that the connection is working.

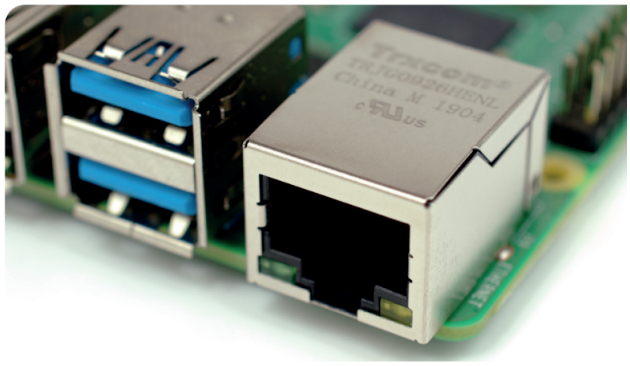

Fig 2.9 Raspberry Pi's Ethernet port

Just above the USB ports, on the left-hand edge of Raspberry Pi, is a 3.5 mm audio-visual (AV) jack (Figure 2.10). This is also known as the headphone jack, and it can be used for that exact purpose – though you'll get better sound connecting it to amplified speakers rather than headphones. It has a hidden, extra feature, though: as well as audio, the 3.5 mm AV jack carries a video signal which can be connected to TVs, projectors, and other displays that support a composite video signal using a special cable known as a tip-ring-ring-sleeve (TRRS) adapter.

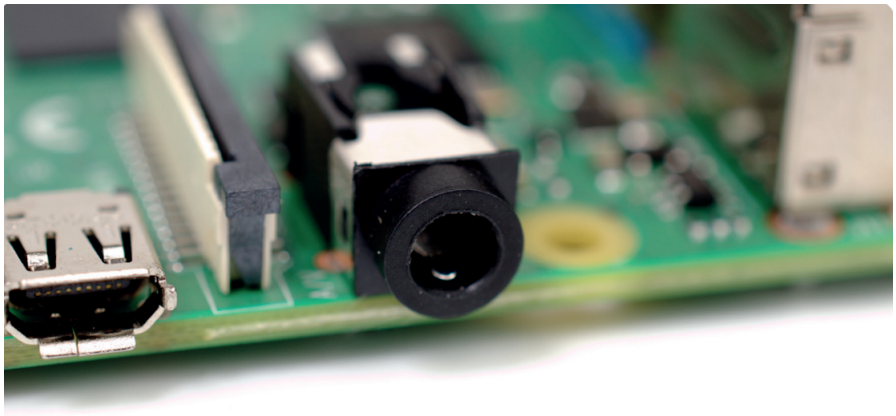

Fig 2.10 Raspberry Pi's 3.5mm AV jack

Directly above the 3.5 mm AV jack is a strange-looking connector with a plastic flap which can be pulled up; this is the camera connector, also known as the Camera Serial Interface (CSI) (Figure 2.11). This allows you to use the specially designed Raspberry Pi Camera Module (about which you'll learn more in Chapter 8, Raspberry Pi Camera Module.).

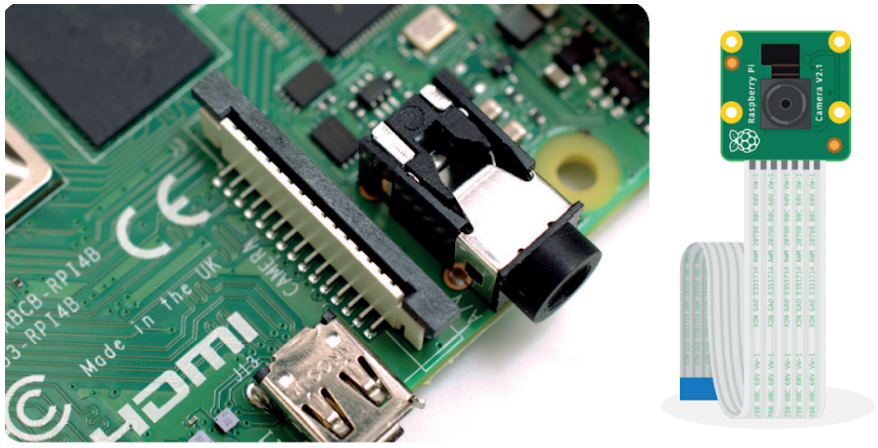

Fig 2.11 Raspberry Pi's camera connector

Above that, still on the left-hand edge of the board, are the micro High Definition Multimedia Interface (micro-HDMI) ports, which are a smaller version of the connectors you'll find on a games console, set-top box, or TV (Figure 2.12). The multimedia part of its name tells you that it carries both audio and video signals, while high-definition tells you that you can expect excellent quality. You'll use these to connect Raspberry Pi to one or two display devices: a computer monitor, TV, or projector.

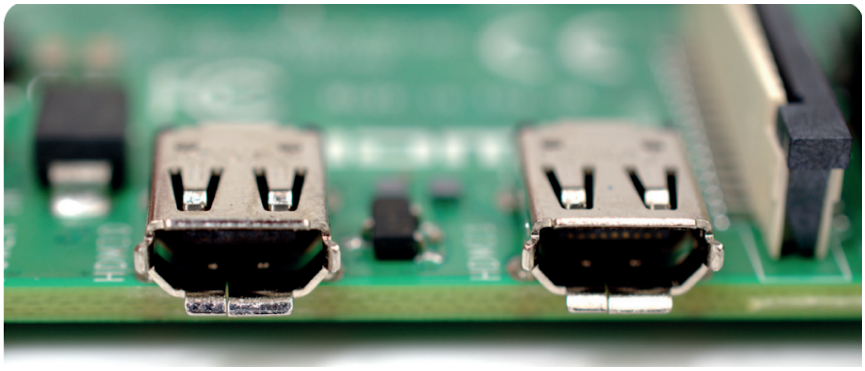

Fig 2.12 Raspberry Pi's Micro HDMI Ports

Above the HDMI ports is a USB Type-C power port (Figure 2.13), which you'll use to connect Raspberry Pi to a power source. The USB Type-C port is a common sight on smartphones, tablets, and other portable devices. While you could use a standard mobile charger to power Raspberry Pi, for best results you should use the official Raspberry Pi USB Type-C Power Supply.

Fig 2.13 Raspberry Pi's USB Type – C power port

At the top edge of the board is another strange-looking connector (Figure 2.14), which at first glance appears to be identical to the camera connector. This, though, is the exact opposite: a display connector, or Display Serial Interface (DSI), designed for use with a Raspberry Pi Touch Display (Figure 2.15, overleaf).

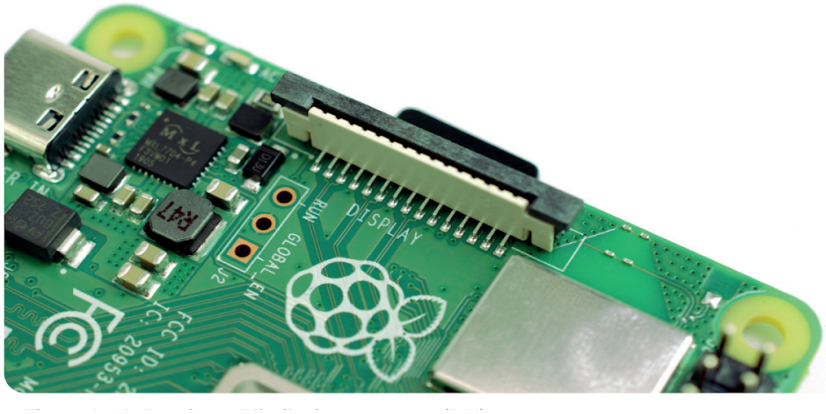

Fig 2.14 Raspberry Pi's display connector (DSI)

At the right-hand edge of the board you'll find 40 metal pins, split into two rows of 20 pins (Figure 2.15). This is the GPIO (general-purpose input/output) header, a feature of Raspberry Pi used to talk to additional hardware from LEDs and buttons all the way to temperature sensors, joysticks, and pulse-rate monitors. You'll learn more about the GPIO header in Chapter 6, Physical computing with Scratch and Python. Just below and to the left of this header is another, smaller header with four pins: this is used to connect the Power over Ethernet (PoE) HAT, an optional add-on which lets Raspberry Pi receive power from a network connection rather than the USB Type-C port.

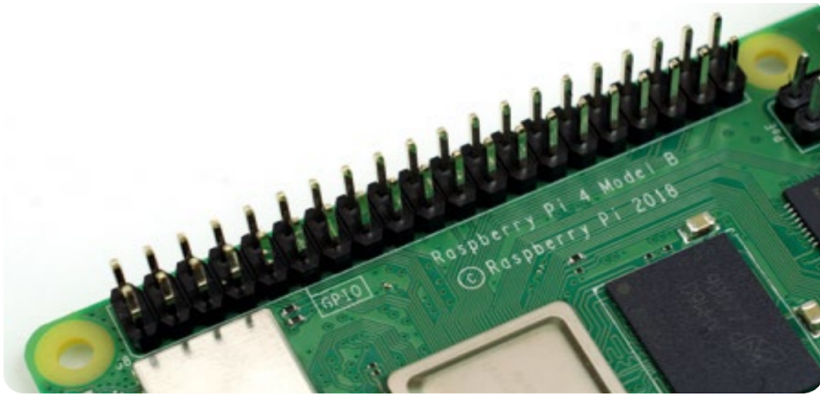

Fig 2.15 Raspberry Pi's GPIO header

There's one final port on Raspberry Pi, but you won't see it on the top. Turn the board over and you'll find a microSD card connector on the opposite side of the board to the display connector (Figure 1-15). This is Raspberry Pi's storage: the microSD card inserted in here contains all the files you save, all the software you install, and the operating system that makes Raspberry Pi run.

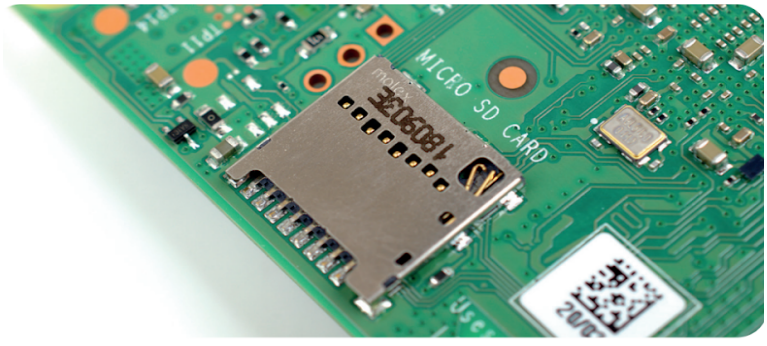

Fig 2.16 Raspberry Pi's micro SD card Connector

Raspberry Pi has been designed to be as quick and easy to set up and use as possible, but like any computer it relies on various external components, called peripherals. While it's easy to take a look at the bare circuit board of Raspberry Pi which looks significantly different to the encased, closed-off computers you may be used to and worry things are about to get complicated, that's not the case. Raspberry Pi is safe to use without a case, providing you don't place it on a metal surface which could conduct electricity and cause a short-circuit. An optional case, however, can provide additional protection; the Desktop Kit includes the Official Raspberry Pi Case, while third-party cases are available from all good stockists. If you want to use Raspberry Pi on a wired network, rather than a wireless (WiFi) network, you'll also need a network cable. This should be connected at one end to your network's switch or router. If you're planning to use Raspberry Pi's built-in wireless radio, you won't need a cable; you will, however, need to know the name and key or passphrase for your wireless network.

USB power supply – A 5 V power supply rated at 3 amps (3 A) and with a USB Type-C connector. The Official Raspberry Pi Power Supply is the recommended choice, as it can cope with the quickly switching power demands of Raspberry Pi.

microSD card with NOOBS – The microSD card acts as Raspberry Pi's permanent storage; all the files you create and software you install, along with the operating system itself, are stored on the card. An 8GB card will get you started, though a 16GB one offers more room to grow. Using a card with NOOBS (New Out-Of-Box Software) pre-installed will save you time; otherwise see Appendix A for instructions on installing an operating system (OS) onto a blank card.

Micro-HDMI cable – This carries sound and images from Raspberry Pi to your TV or monitor. One end of the cable has a microHDMI connector for Raspberry Pi; the other, a full-size HDMI connector for your display. Or, you can use a micro-HDMI to HDMI adapter and a standard, full-size HDMI cable. If using a monitor without an HDMI socket, you can buy microHDMI to DVI-D, DisplayPort, or VGA adapters. To connect to an older TV which uses composite video or has a SCART socket, use a 3.5 mm tip-ring-ring-sleeve (TRRS) audio/video cable.

Raspberry Pi Specifications

Raspberry Pi 4 Model B and Raspberry Pi 400's system-on-chip is a Broadcom BCM2711B0, which will see written on its metal lid if you look closely enough (on Raspberry Pi 4). This features four 64-bit ARM Cortex-A72 central processing unit (CPU) cores, each running at 1.5GHz or 1.8GHz (1.5 or 1.8 thousand million cycles per second), and a Broadcom Video Core VI (Six) graphics processing unit (GPU) running at 500MHz (500 million cycles per second) for video tasks and for 3D rendering tasks such as games. The system-on-chip is connected to 2GB, 4GB, or 8GB (two, four, or eight thousand million bytes) – 4GB on Raspberry Pi 400 – of LPDDR4 (Low-Power Double-Data-Rate 4) RAM (random-access memory) which runs at 3200MHz (three thousand two hundred million cycles per second). This memory is shared between the central processor and graphics processor. The microSD card slot supports up to 512GB (512 thousand million bytes) of storage. The Ethernet port supports up to gigabit (1000Mbps, 1000-Base-T) connections, while the radio supports 802.11ac WiFi networks running on the 2.4GHz and 5GHz frequency bands Bluetooth 5.0, and Bluetooth Low Energy (BLE) connections.

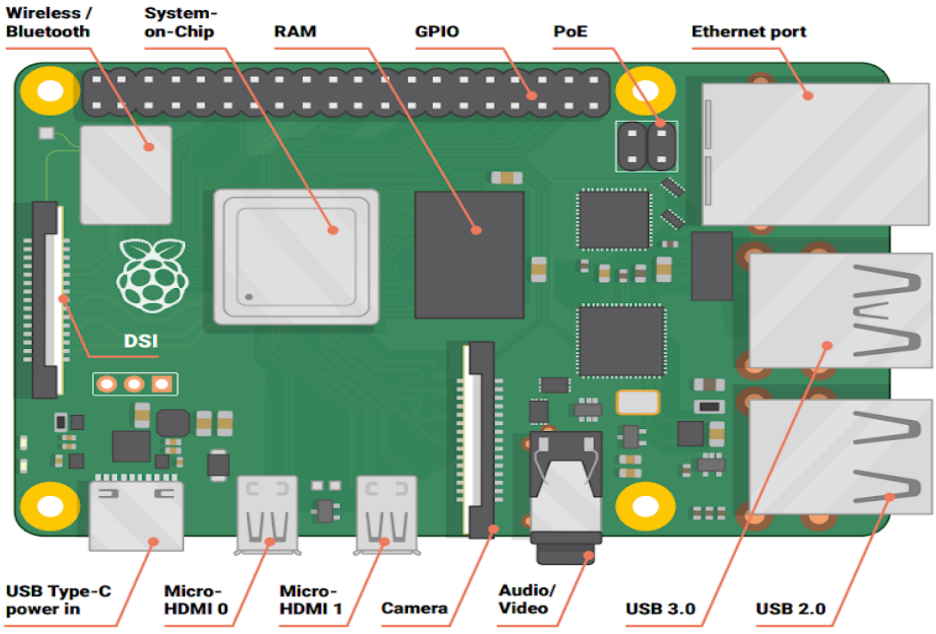

Fig 2.17 Specifications of Raspberry Pi

- **CPU:** 64-bit quad-core ARM Cortex-A72 at 1.5GHz
- **GPU:** Video Core VI at 500MHz
- **RAM:** 1GB, 2GB, or 4GB of LPDDR4
- **Networking:** Gigabit Ethernet, dual-band 802.11ac, Bluetooth 5.0, Bluetooth Low Energy
- **Audio/Video Outputs:** 3.5 mm analogue AV jack, 2 × micro-HDMI 2.0 n
- **Peripheral Connectivity:** 2 × USB 2.0 ports, 2 × USB 3.0 ports, Camera Serial Interface, Display Serial Interface (DSI)
- **Storage:** microSD, up to 512GB n Power: 5 volts at 3 amps via USB Type-C
- **Extras:** 40-pin GPIO header, Power over Ethernet compatibility (with additional hardware.

Web cam

Like any hardware add-on, the Camera Module or HQ Camera should only be connected to or disconnected from Raspberry Pi when the power is off and the power cable unplugged. If your Raspberry Pi is turned on, choose Shutdown from the raspberry menu, wait for it to power off, and unplug it. In most cases, the supplied ribbon cable will already be connected to the Camera Module or HQ Camera; if it isn't, turn your camera board upside-down so the sensor is on the bottom and look for a flat plastic connector. Carefully hook your fingernails around the sticking-out edges and pull outwards until the connector pulls part-way out. Slide the ribbon cable, with the silver edges downwards and the blue plastic facing upwards, under the flap you just pulled out, then push the flap gently back into place with a click; it doesn't matter which end of the cable you use. If the cable is installed properly, it will be straight and won't come out if you give it a gentle tug; if not, pull the flap out and try again.

Fig 2.18 Web Cam

Algorithm 1: Smart parking and management system

Input: Real-time images. Output: Number of parked cars, number of vacant spaces, and directions to these spaces.

1. Scan the parking lot every time a vehicle enters or leaves.
2. If no vacant spaces are available, then
3. Display a message to inform drivers and keep gate closed.
4. Else
5. Open the gate
6. Remove noise from captured image.
7. Transform the resultant image into a gray image.
8. Determine the dimensions of the parking area.
9. End of preprocessing phase.
10. Assign a number of vacant spaces to a threshold variable.
11. Subtract captured image of current parking area from the parking map structure.
12. Apply AlexNet for deep learning objective.
13. For i=1: length of parking area
14. For)-1: width of parking area
15. Do the following:
16. Find resultant image from subtraction process and compare it with the threshold. .
17. If result> threshold, them:
18. Place 1 in the current index.
19. Else
20. Place
21. End

24. Check which lanes have vacant spaces to direct drivers to the nearest one. 25. Display instructions for drivers or motorists to park their vehicles in the detected lane.

26. End of algorithm.

2.4 SOFTWARE USED

In computer science, artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans. AI textbooks define the field as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving". The traditional problems of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI.

Object detection is a computer vision technique whose aim is to detect objects such as cars, buildings, and human beings, just to mention a few. The objects can generally be identified from either pictures or video feeds. Object detection has been applied widely in video surveillance, self-driving cars, and object/people tracking. Object detection is widely used in computer vision tasks such as face detection, face recognition and video object co-segmentation.

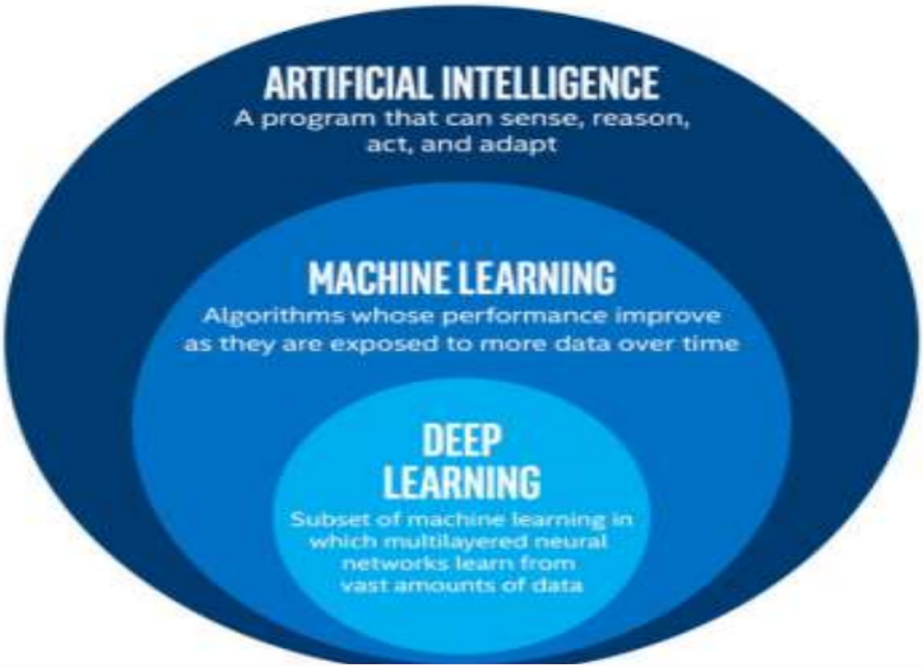

Fig 2.19 AI is a superset within which deep learning and machine learning

AI is human intelligence demonstrated by machines to perform simple to complex tasks where in ML it provides machines the ability to learn and understand without being explicitly programmed. The idea behind AI is to program machines to carry out tasks in more human ways or smart ways in ML the key to teach computers to think and understand like we do is ML. Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

When we view an image the object in it are recognized by our brain instantaneously, but the machines take a lot of time for training and testing to identify the objects. Machines cannot do this task easily. People are trying hard to solve this problem, but they are able to achieve 65% of accuracy only. It is so hard for the machines to categorize and recognize objects like humans. This is actually the difficult task of computer vision. Identifying each object in a picture or scene with the help of computer/software called as object detection. Face detection, driver less cars, vehicle detection and few other technologies uses object detection. For object detection, artificial neurons are used in deep neural networks they are similar to humans composed of neurons and process forwarding. The three major methods widely adopted in this field are: You Only Look Once (YOLO), Single Shot Detector (SSD) and Faster Region CNN (F-RCNN) [1]. In the first section, the introduction of object detection and deep learning process are shown. Second section demonstrates related work. In third section, we have reviewed the existing techniques of object detection. Section four, put a light on the proposed model. Fifth section represents the experimental results. Finally, section six concludes the proposed research work.

The proposed approach for detecting the objects in real-time from images by using convolutional neural network deep learning process for that we have used OpenCV libraries. The proposed scheme uses single shot multi-box detector (SSMBD) algorithm for higher detection precision with real-time speed. However, the single shot multi-box detector (SSMBD) algorithm is not appropriate to detect tiny objects, since it overlooks the context from out of the boxes. Our proposed approach uses a new architecture as a combination of Faster R-CNN with convolutional features and SSMBD with multi-scale contexts in additional layers. The algorithm comprises of two phases: first is feature maps extraction and the other one is concerned with application of small convolutional filters for detecting objects.

The major objective during the training is to get a high-class confidence score by matching the default boxes with the ground truth boxes. Our scheme uses separate filters with different default boxes to tackle the difference in aspect ratio and also used multi-scale feature maps for object detection.

2.4.1 YOLO

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region-based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes. In YOLO we take an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster (45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes.

Object Detection Flowchart:

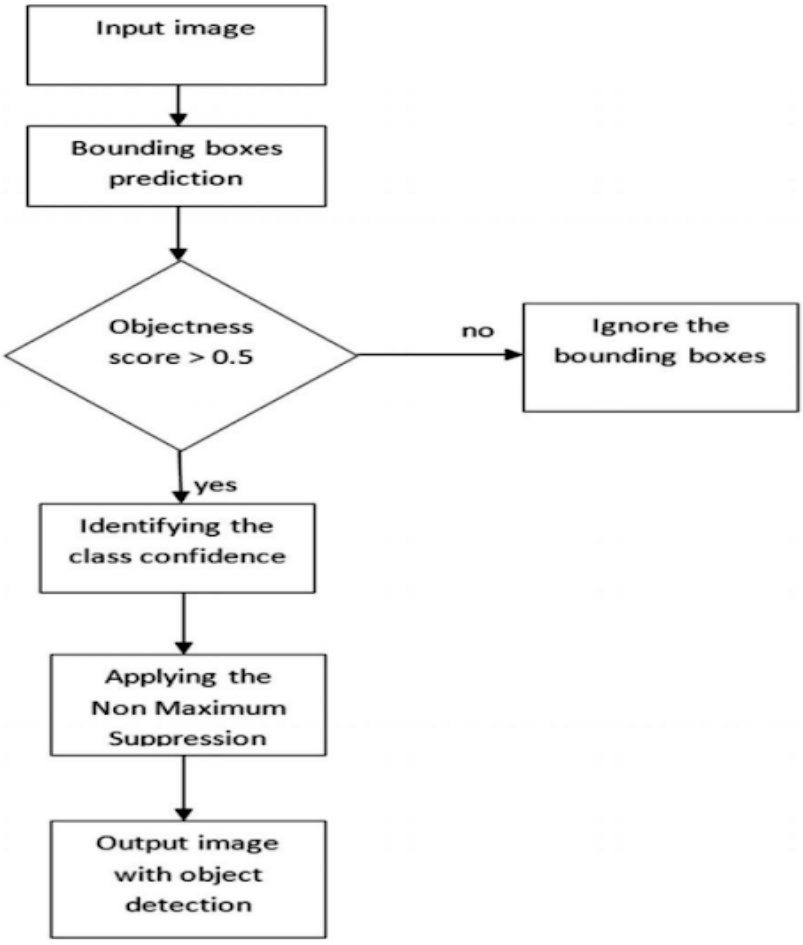

Fig 2.20 flow chart of object detection

The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation. This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms.

This algorithm starts with extraction of a single image from a video stream, the next step the extracts image and is resized to 416*416 which represents input to the Yolo network. YOLO v3 neural network has 106 layers. Apart from using convolutional layers, its architecture also contains residual layers, up sampling layers, and skip (shortcut) connections. An image which is extracted from video is taken as an input and returns tensor as output which represents Coordinates and positions of predicted bounding boxes which should contain objects, A probability that every bounding box contains object from set of images, Probabilities and confidence ratio that every object in the bounding box belongs to a fixed class. The detection of objects is done on the three different layers, whose input dimensions can be given as 13x13, 26x26 and 52x52. Object detection is done at 3 specific scales which depicts the issue of prior YOLO neural network architectures, the detection of smaller objects.

Output sensors have same widths and heights like their inputs, but depth is different and can be defined by the following formula: depth = (4 + 1 + class probabilities) * 3 This network is capable of detecting multiple objects on the single input image at the same time. During the training process of the network, it also learns to analyze the complete input image and does the predictions. In this way, the network gets to know about the complete scenario and environment of the object which helps the network to give better performance and also precise results as compared to the sliding windows approach methods. The breakdown of image into the grid cells is unique in the YOLO algorithm, unlike the other object detection solutions. The input is split up into S*S grid of cells where every grid cell can predict 3 bounding boxes.

Fig 2.21 Input image

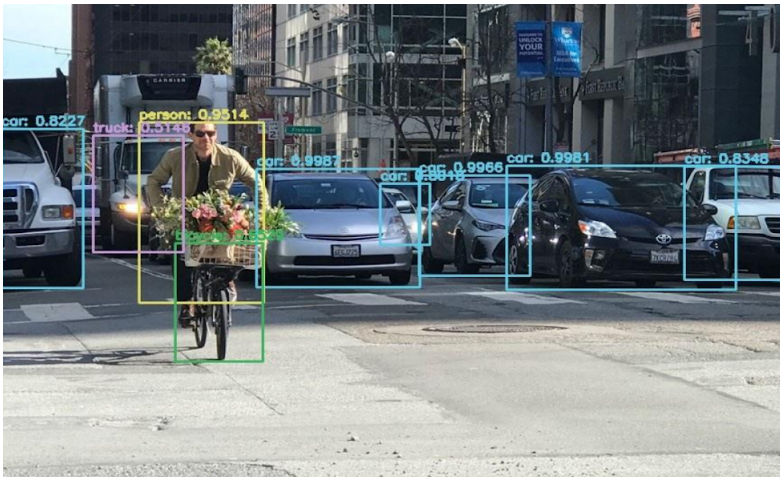

Fig 2.22 output image

The neural network was trained for 120 epochs and that took two weeks on the previously specified hardware. Key assets of the training model were checked regularly which are: the loss function value, precision, recall, mAP, and average IoU. All of these values were improving until the 120th epoch when the model started overfitting, then the training was stopped. Because of the too many small and close, but labelled objects in the dataset, we were not reaching high precision and recall values as expected. Common false detections found in the output were further investigated on the smaller custom dataset. Most of the undetected objects were in heavy traffic situations where one cell in the detection grid would detect more than three objects. In a few cases, some of the objects were undetected because they were covered by other detected objects. As the accuracy is good, and we use this algorithm for the detection of traffic participants the safety of vehicles and passengers will not be put in danger at any moment. The YOLO algorithm processed an input video stream from the dashboard camera with the frame resolution 1920 x 1080 px at an average of 23 FPS using previously described hardware. Which confirmed the statement that the YOLO algorithm can be used for real-time video processing.

The main benefit of the solution proposed in this paper is using YOLO neural network for car detection using yolo. The solution provides a real-time response, which is required for the development of the ADAS components. The application of the YOLO algorithm can provide a solid base for the object detection module as a part of the ADAS. In future work, the accuracy of this algorithm can be increased with training on the bigger and more diverse datasets that cover different weather and lighting conditions and if we use it in fusion with other sensor's readings.

The neural network was trained for 120 epochs and that took two weeks on the previously specified hardware. Key assets of the training model were checked regularly which are: the loss function value, precision, recall, mAP, and average IoU. All of these values were improving until the 120th epoch when the model started overfitting, then the training was stopped refer Fig.3.

Epoch	Precision	Recall	F_1 score	mAP value	Average IoU
40	0.37	0.35	0.36	18.98%	24.19%
47	0.39	0.37	0.38	21.44%	26.12%
56	0.37	0.39	0.38	23.49%	25.44%
75	0.40	0.48	0.44	30.98%	28.12%
90	0.58	0.53	0.56	44.06%	44.06%
109	0.60	0.54	0.57	44.53%	43.65%
120	0.63	0.55	0.59	46.60%	45.98%

Fig 2.23 Training YOLOv3 Neural Network Results

Because of the too many small and close, but labelled objects in the dataset, we were not reaching high precision and recall values as expected. Common false detections found in the output were further investigated on the smaller custom dataset. Most of the undetected objects were in heavy traffic situations where one cell in the detection grid would detect more than three objects. In a few cases, some of the objects were undetected because they were covered by other detected objects. However, in both previous cases, all of the closest objects to the camera's position (vehicle) were successfully detected and classified as shown in Fig 3. As the accuracy is good, and we use this algorithm for the detection of traffic participants the safety of vehicles and passengers will not be put in danger at any moment. The YOLO algorithm processed an input video stream from the dashboard camera with the frame resolution 1920 x 1080 px at an average of 23 FPS using previously described hardware. Which confirmed the statement that the YOLO algorithm can be used for real-time video processing.

2.4.2 OPEN CV

Object detection is a well-known computer technology connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images and videos. There are various applications of object detection that have been well researched including face detection, character recognition, and vehicle calculator. Object detection can be used for various purposes including retrieval and surveillance. In this study, various basic concepts used in object detection while making use of OpenCV library of python 2.7, improving the efficiency and accuracy of object detection are presented. Object detection and location in digital images has become one of the most important applications for industries to ease user, save time and to achieve parallelism. This is not a new technique but improvement in object detection is still required in order to achieve the targeted objective more efficiently and accurately.

The main aim of studying and researching computer vision is to simulate the behaviour and manner of human eyes directly by using a computer and later on develop a system that reduces human efforts. Computer vision is such kind of research field which tries to perceive and represents the 3D information for world objects. Its main purpose is reconstructing the visual aspects of 3D objects after analysing the 2D information extracted. Real life 3D objects are represented by 2D images. The process of object detection analysis is to determine the number, location, size, position of the objects in the input image. Object detection is the basic concept for tracking and recognition of objects, which affects the efficiency and accuracy of object recognition. The common object detection method is the colour-based approach, detecting objects based on their colour values. The method is used because of its strong adaptability and robustness, however, the detection speed needs to be improved, because it requires testing all possible windows by exhaustive search and has high computational complexity.

Object detection from a complex background is a challenging application in image processing. The goal of this project is to identify objects placed over a surface from a complex background image using various techniques. The detection of the objects can be extended using automation and robotics for plucking of the objects like apples, bananas from the corresponding tree using the image processing techniques and it will be easier, faster and convenient to pluck the apples and bananas rather than the manual plucking.

OpenCV (Open Source Computer Vision) is an open source computer vision and machine learning software library. OpenCV was initially built to provide a common infrastructure for applications related to computer vision and to increase the use of machine perception in the commercial products. As it is a BSD-licensed product so it becomes easy for businesses to utilize and modify the existing code in OpenCV. Around 3000 algorithms are currently embedded inside OpenCV library, all these algorithms being efficiently optimized. It supports real-time vision applications. These algorithms are categorized under classic algorithms, state of art computer vision algorithms and machine learning algorithms. These algorithms are easily implemented in Java, MATLAB, Python, C, C++ etc. and are well supported by operating system like Window, Mac OS, Linux and Android.

A full-featured CUDA and OpenCL interfaces are being actively developed for the betterment of technology. There are more than 500 different algorithms and even more such functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. For OpenCV to work efficiently with python 2.7 we need to install NumPy package first.

Object Classification In Moving Object Detection

Object classification approach is based on shape, motion, colour and texture. The classification can be done under various classes such as trees, animals, humans, objects etc. Tracking objects and analysing their features is a key concept of object classification.

a) **Shape Based**

A mixture of image-based and scene-based object parameters such as image blob (binary large object) area, the aspect ratio of blob bounding box and camera zoom is given as input to this detection system. Classification is performed on the basis of the blob at each and every frame. The results are kept in the histogram.

b) **Motion Based**

When a simple image is given as an input with no objects in motion, this classification is not needed. In general, non-rigid articulated human motion shows a periodic property, hence this has been used as a strong clue for classification of moving objects. Based on this useful clue, human motion can be distinguished from other objects motion.

c) Colour Based

Though colour is not an appropriate measure alone for detecting and tracking objects, but the low computational cost of the colour-based algorithms makes the colour a very good feature to be exploited. For example, the colour-histogram based technique is used for detection of vehicles in real-time. Colour histogram describes the colour distribution in a given region, which is robust against partial occlusions.

d) Texture Based

The texture-based approaches with the help of texture pattern recognition work similar to motion-based approaches. It provides better accuracy, by using overlapping local contrast normalization but may require more time, which can be improved using some fast techniques.

2.4.2 STEPS INVOLVED IN OBJECT DETECTION IN PYTHON 2.7

I. Install OpenCV-Python

Below Python packages are to be downloaded and installed to their default location - Python-2.7.x, NumPy and Matplotlib. Install all packages into their default locations. Python will be installed to C/Python27/. Open Python IDLE. Enter import NumPy and make sure NumPy is working fine. Download OpenCV from Source forge. Go to OpenCV/build/python/2.7 folder. Copy cv2.pyd to C:/Python27/lib/site-packages.

II. Read an Image

Use the function CV2.imread() to read an image. The image should be in the current working directory otherwise, we need to specify the full path of the image as the first argument. The second argument is a flag which specifies the way image should be read.

1. CV2.IMREAD_COLOR: This function is used to load a colour image. Transparency of image, if present will be neglected. It is the default flag.
2. CV2.IMREAD_GRAYSCALE: Loads image in grayscale mode.
3. CV2.IMREAD_UNCHANGED: Loads image as such including alpha channel.

III Feature detection and description

- Understanding features (What are the main features in an image? How can finding those features be useful to us?).
- Corner detection (Okay, Corners are good features? But how do we find them).
- Feature matching (We know a great deal about feature detectors and descriptors. So let us now learn to match different descriptors. OpenCV provides two techniques, Brute-Force matcher, and FLANN based matcher.).
- Homography (As we are aware of feature matching, so let us now blend it with Camera calibration and 3D reconstruction (calib3d module) to find objects for description in a complex image.).

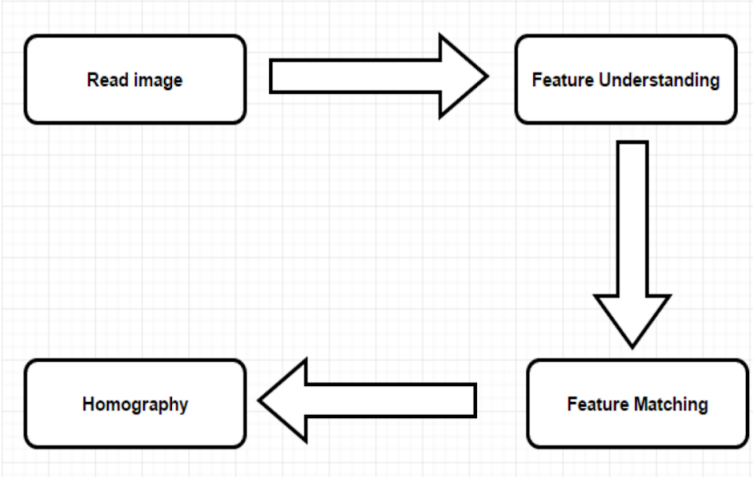

Fig 2. 24 Flow Chart for Object Detection

2.4.3 PYTHON VS OTHER LANGUAGES FOR OBJECT DETECTION

Object detection is a domain-specific variation of the machine learning prediction problem. Intel's OpenCV library that is implemented in C/C++ has its interfaces available in a number of programming environment such as C#, MATLAB, Octave, R, Python etc. Some of the benefits of using Python codes over other language codes for object detection are

- More compact and readable code.
- Python uses zero-based indexing.
- Dictionary (hashes) support is offered.
- Simple and elegant Object-oriented programming.

Computer vision is still a developing discipline, it has not been matured to that level where it can be applied directly to real life problems. After few years‟ computer vision and particularly the object detection would not be any more futuristic and will be ubiquitous. For now, we can consider object detection as a sub-branch of machine learning.

2.5 VARIOUS OBJECT DETECTION ALGORITHMS IMPLEMENTED IN PYTHON

Haar-like features

It is an effective object detection technique which is proposed by Paul Viola and Michael Jones in 2001. It is a machine learning based method for object detection where we train a classifier from a lot of images. This classifier is then used in detecting objects in an image. Initially, the algorithm needs images with faces (positive images) and images without faces (negative images) to train a classifier and then extract features from this classifier.This method introduces a concept cascade of the classifier. Instead of applying all the features at once we group the features into different stages of the classifier and apply one by one. Discard the window if it fails in the first stage. If it passes the stage then continue the process. The window which passes through all the stages will be our desired region.

Circular Hough Transformation

Hough transformation was invented by Richard Duda and Peter Hart in 1992, this transformation was initially meant to detect arbitrary shapes from an image. It was later modified to detect circular objects in low-contrast noisy images and referred as Circular Hough Transformation.

CHT relies on equations for circle

r 2 = (x-a)2 +(x-b)2

where a and b are the coordinate of the center, and r is the radius of the circle. CHT relies on three parameters, which require larger computation time and memory and it increases the complexity to extract information from the image. For simplicity, CHT programs are provided with a constant value of radius or provided with a range of radius prior to running the application.

Template matching

Template matching is a high-level machine vision technique to detect objects from an image that matches a given image pattern. This technique matches the source image with the template image or patch. If the template image has strong features, the feature-based approach may be used otherwise template based approach is used.

Blob detection

This method is used to detect regions in an image that differs in properties. A blob is a region in the image in which all the points can be considered to be similar to each other. There are two classes of blob detection method: differential method and local extrema method.

The Gradient-based method

The gradient-based method uses spatial and temporal partial derivatives to estimate image flow at every position in the image. If the motion is not known in advance to be restricted to a small range of possible values then a multi- scale analysis must be applied so that the scale of the smoothing prior to derivatives estimation is appropriate to the scale of the motion. This can make this method computationally expensive.

Bag-of-words method

This method can be applied to image classification. Images are treated like visual words. These visual words are important points in the images. These points are called features. This method can be used for image classification by creating a large vocabulary of many visual words and representing each histogram of the frequency words that are in the image.

Deep Face method

Facebook AI research group has developed Deep Face Software in Menlo Park, California by the support of an advanced deep learning neural network. A piece of software that simulates an approximation of how real neurons works is called a Neural Network. Deep Face Learning can be performed by Machine Learning. It can be defined as a huge body of data that develops a high-level abstraction by looking for recurring faces.

2.6 APPLICATIONS AND FUTURE SCOPE OF PYTHON

Computer vision is still a developing discipline, it has not been matured to that level where it can be applied directly to real life problems. After few years‟ computer vision and particularly the object detection would not be any more futuristic and will be ubiquitous. For now, we can consider object detection as a sub-branch of machine learning.

Have you ever wondered how Facebook detects your face when you upload a photo? Not only it detects, it remembers the face too. This is a simple application of object detection that we see in our daily life. Object detection can be also used for counting purpose, it is used for keeping a count of particular or all objects in an image or a frame. For e.g. from a group photograph it can count the number of persons and if implemented smartly you may also find out different people with different dresses. Similarly, when the object is a vehicle, object detection along with tracking can be used for finding the type of vehicle, this application may be extended to even make a traffic calculator. Identification of unwanted or suspicious objects in any particular area or more specifically object detection techniques are used for detecting bombs/explosives. It is also even used for personal security purpose.

Objects can be recognized and tracked in videos for security purpose. Object recognition is required so that the suspected person or vehicle can be tracked. Human gestures can be stored in the system and can be used for recognition in a dynamic environment by computers to interact with humans. Object detection‟s scope is not yet limited here. You can use it for any purpose you can think of. For e.g. for solving number puzzles by just giving their images as input and applying some proper algorithms after detecting different numbers and their places from the input image.

2.5 ADVANTAGES AND DISADVANTAGES

Efficiency - Manually handling parking is rarely as precise and efficient as some might think. The work requires a lot of focus, as it deals with managing a large volume of people simultaneously in the same place. And if you miss or ignore one of them during that time, it could result in fines or even an accident for them! Then you'll have to deal with issues which would be much worse. You can avoid this by working with software instead; an automated ticketing solution is much more convenient for drivers and easy to use because they just need to get on their smartphone, enter their vehicle registration information, and drive into the car park. If a user wants to pay digitally, they can do so using their pre-registered credit card details; above all else, there are no mistakes allowed while parking.

Faster processing - Sometimes we get very bored and want to go home as quickly as possible. We are in a rush, and standing at the gates is a waste of our precious time. Are there any parking spaces available? Are my car keys still working? How many more minutes will I still have to wait? A lot of our thoughts race through our heads. Employees can now take advantage of the fantastic parking management systems to avoid the long line and wait for entry. These parking solutions can completely automate the parking process and make car parking time- efficient.

Improved Security - One of the perks of a parking management system is that it provides security. A barrier and reservation feature controls the vehicles allowed to enter and exit a space, for instance, at a business or event. This way, one doesn't have to worry about guests leaving valuables in their car, vandalism, theft, or illegal dumping. It can make your car park utterly secure. A CCTV (security camera) will monitor cars as well as licence plate numbers, so owners can keep track of their vehicles' whereabouts even if they've left them for someone else to watch over. Hence, having parking management systems in a car park is indispensable.

Disadvantages:

Expensive Construction and Installation - A parking management system can cost a lot of money. For example, the statistical feature, ticketing technology, and reporting tools are just some things that increase the price. In addition, the other things you might need to pay for include high usage or peak access fees, software maintenance fees, and fee waivers, to name a few. Your budget may not allow you to purchase everything at once, so make sure you prioritise your needs based on your organisations requirements.

System Breakdown - Utilising technology to manage a car park is unquestionably an excellent decision. Still, we cannot ignore that machines can start malfunctioning anytime, no matter how meticulously they are manufactured or what software they are integrated with. In these cases, chaos may occur. Imagine if cars couldn't access buildings and parked inside vehicles couldn't move. If the system malfunctions, this could lead cars to park in the wrong places. This is another considerable downside of using a parking management system.

CHAPTER 3
RESULT AND DISCUSSION

The proposed system starts by taking data from the installed camera which provides real-time information. These images or video streams are fed into the algorithm, which starts the processing. If no vacant spaces are available, the system displays a message saying that the parking lot is fully occupied, and drivers must search for alternatives. The gate at the entrance remains closed. If there are empty spaces, the system opens the gate and provides directions to vacant spots where drivers may park their vehicles. The system additionally displays the total number of vehicles currently parked and the number of vacant spaces.

This system is intelligent since it employs the convolutional neural network tool to train itself and adapt to any type of conditions and circumstances that may exist such as a traffic accident or car failure inside the parking lot or a vehicle that parks in the driveway which could lead to a deadlock. Once an abnormal activity inside the parking area is detected, then the considered team such as the administration team is alerted and notified to take proper action. Since the implemented system depends and relies on a single camera, thus its fault tolerance can be considered low. This drawback is resolved by having a backup camera that comes up immediately once the system detects no inputs from the main camera for a long time.

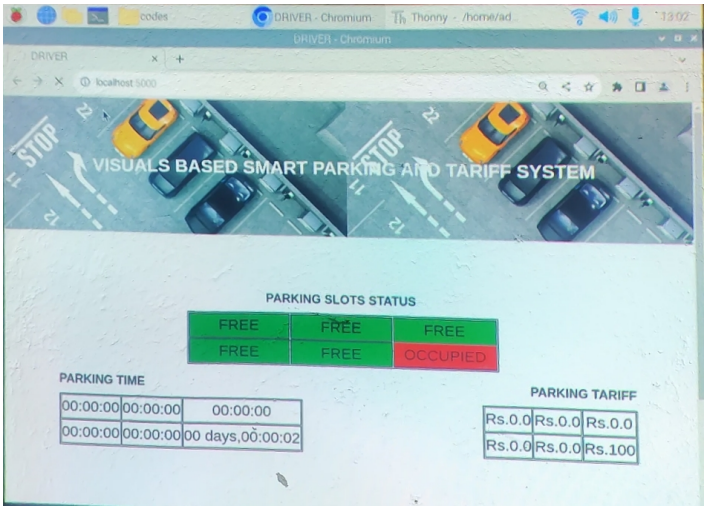

Fig 3.1 Web Page of Smart Parking System

3.1 DISCUSSION

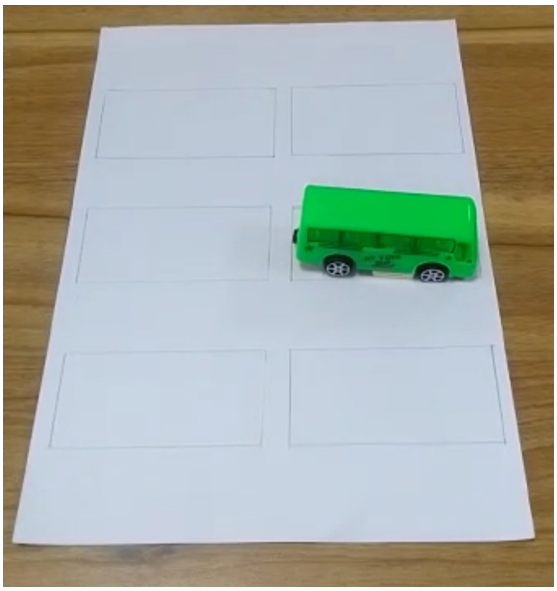

Fig 3.2 Smart Parking Slots

It contained two stages, one for scanning the parking area and another for securing data. The method showed the status of the parking area to drivers and motorists before entering and provided directions to available parking spaces. The hashing method was utilized to secure the system itself. A Raspberry Pi microcontroller was utilized to detect vehicles. An SD card and a board module were also employed to increase the rate of reliability of the implemented method. This utilized a 32- byte hashing key for authenticating purposes. In addition, no sensors were used to reduce the cost. It performed only two scenarios while four scenarios were conducted using the proposed system. The proposed system starts by taking data from the installed camera which provides real-time information. These images or video streams are fed into the algorithm, which starts the processing.

If no vacant spaces are available, the system displays a message saying that the parking lot is fully occupied, and drivers must search for alternatives. The gate at the entrance remains closed. If there are empty spaces, the system opens the gate and provides directions to vacant spots where drivers may park their vehicles. The system additionally displays the total number of vehicles currently parked and the number of vacant spaces. The proposed algorithm is demonstrated as Very cost-effective, environmentally friendly since it decreases fuel consumption, power, and energy dissipated, Easy to maintain and operate. It can be integrated with other systems easily.

Additional features such as checking the parking area ahead of arrival, booking a vacant space, and adding billing components can be added with minor changes in the system design. The proposed system requires fewer operations to run and manage its features; thus, its execution time is less when compared to other methods in the literature. Further more, it is cost-effective as it utilizes only a single camera that sends its data in real-time to the connected machine. This system is intelligent since it employs the convolutional neural network tool to train itself and adapt to any type of conditions and circumstances that may exist such as a traffic accident or car failure inside the parking lot or a vehicle that parks in the driveway which could lead to a deadlock.

The system also utilises a dashboard for the management of each parking lot, which is aimed at helping the manager monitor the usage of the parking lot. Each dashboard reflects information of its parking lot to the manager: the current empty spaces, cars parked in the lot per day, user credentials of those cars whose drivers break the rules and so on.There can be no doubt this proposed solution could add huge value to any organisation, yet it has its limitations. Multiple technologies are needed to provide a full solution. There are also simpler solutions, yet these require the user to possess an object, thus denying them flexibility. This paper proves that there is a viable solution to this challenge, yet debunks the myth that IoT can simply solve all challenges.

This can be upgraded to be able to be deployed with radio frequency identification (RFID) to authenticate at the gate management service (GMS) to assign a definitive slot. The system provides an additional feature to monitor parking lot over the internet. Analysis shows that in almost every parking lot there will be some drivers disobeying the rules. For instance, some drivers may occupy more than one place and some may park on the road instead of the correct places. Our system will also set up some punishment for those drivers who break the rules. The system will warn those drivers at the first instance, which will be recorded in the system. If these drivers do not change their behaviours, they will be fined. The fine will be added to their parking fee. However, if they keep doing the same things, they will be forbidden to park in this parking lot.

This report on artificial intelligence (AI) in object detection shows different approaches in the modern word used for object detection as can be seen from above sections there are different techniques with their upgraded versions are there which help in detection of object as well as real time object detection of objects. Real time object detection in today's world has been made easy using the various object detection techniques more advancements can be done in this research field. for example: in YOLO V3 Anchor box offset prediction, focal loss and liner prediction instead of logistic didn't work so in future it can further be extended by finding solutions to their problems. There is always a possibility of improvement in this world of researchers Nothing Is Perfect.

CHAPTER 4
CONCLUSION

It is apparent that the request for the smart car parking system will continue to increase in the forthcoming years. Though the smart parking system already exists, this paper adds a new idea by detected the wrong parking and is aimed at making the system more cost effective and user-friendly thus increasing its adoption in the market. The paper was successful and cost-effective, user-friendly and had 95% accuracy, we did a 20 case, and there was one case form the 20 is wrong cases. We found that if the model and the objectives were bigger, then the reads will be more accurate.

The proposed solution integrates multiple technologies to provide the most useful long-term solution. In the future the use of the parking space allocation algorithm, could be used to data model user parking requirements in any parking lot, potentially maximising occupancy by allocating the correct ratio of parking spaces per user group. Pre-process: Even though we are facing an allocation problem in a small district, the problem scale is still very large. Buildings in the same block are treated as one destination, and we have totally 12 destinations.

Implementation: To check the implement ability of the “smart parking” approach, we have built an urban-like testbed. This contains the basic elements of an urban traffic environment, such as roads, vehicles, traffic lights, urban blocks, etc. To meet the three main requirements of “smart parking,” we include the following features. First, vehicle locations and parking spot status are detected by overhead cameras, which serve as a GPS. Second, vehicles communicate with the central allocation system (a computer) by Wifi. Third, we use wireless controllable parking barriers to guarantee a reservation. The computer periodically executes the optimal allocation algorithm and makes assignments, sends the assignment results to vehicles, and controls the opening and closing of each parking barrier. Due to space limitations, we only have 5 parking spots and 4 vehicles. However, the “smart parking” concept is clearly illustrated in the implementation, including allocations dynamically updated when random events occur, e.g., a vehicle joining the system or delays due to traffic lights. One can also see how vehicles access reserved parking spots that meet their requirements without blindly competing.

4.1 Future Scope

This can be upgraded to be able to be deployed with radio frequency identification (RFID) to authenticate at the gate management service (GMS) to assign a definitive slot. The system provides an additional feature to monitor parking lot over the internet. Analysis shows that in almost every parking lot there will be some drivers disobeying the rules. For instance, some drivers may occupy more than one place and some may park on the road instead of the correct places. Our system will also set up some punishment for those drivers who break the rules. The system will warn those drivers at the first instance, which will be recorded in the system. If these drivers do not change their behaviours, they will be fined. The fine will be added to their parking fee. However, if they keep doing the same things, they will be forbidden to park in this parking lot.

The system also utilises a dashboard for the management of each parking lot, which is aimed at helping the manager monitor the usage of the parking lot. Each dashboard reflects information of its parking lot to the manager: the current empty spaces, cars parked in the lot per day, user credentials of those cars whose drivers break the rules and so on.There can be no doubt this proposed solution could add huge value to any organisation, yet it has its limitations. Multiple technologies are needed to provide a full solution. There are also simpler solutions, yet these require the user to possess an object, thus denying them flexibility. This paper proves that there is a viable solution to this challenge, yet debunks the myth that IoT can simply solve all challenges.

The possibilities of using computer vision to solve real world problems are immense. The basics of object detection along with various ways of achieving it and its scope has been discussed. Python has been preferred over MATLAB for integrating with OpenCV because when a Matlab program is run on a computer, it gets busy trying to interpret all that MATLAB code as Matlab code is built on Java. OpenCV is basically a library of functions written in C\C++. Additionally, OpenCV is easier to use for someone with little programming background. So, it is better to start researching on any concept of object detection using OpenCV-Python.

REFERENCES

[1] Nihal Deochand Chaudhari and Ayushi Singh, “Automatic car parking system using Arduino” International research journal of engineering and technology May 2020.

[2] E. Saranya, V Gokula Prasath, G. Gokulraj and Kowshik’ “IoT based smart car parking system" International journal of Research in Engineering, science and management, Feb 2019.

[3] Suvarna S. Nandyal and Sabiya Sultana, “Smart car parking system using Arduino UNO”, International journal of Computer application July2017.

[4] Omkar Walvekar, Amar Kulkarni and Rohith mane, “Automatic car parking “, International Research journal of Engineering and Technology March 2017.

[5] Khushboo Khurana and Reetu Awasthi,“Techniques for Object Recognition in Images and Multi-Object Detection”,(IJARCET), ISSN:2278-1323,4th, April 2017.

[6] Latharani T.R., M.Z. Kurian, Chidananda Murthy M.V,“Various Object Recognition Techniques for Computer Vision”, Journal of Analysis and Computation, ISSN: 0973-2861.

[7] Md Atiqur Rahman and Yang Wang, “Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation,” in Object detection, Department of Computer Science, University of Manitoba, Canada, 2017.

[8] R. Hussin, M. Rizon Juhari, Ng Wei Kang, R..Ismail, A.Kamarudin, “Digital Image Processing Techniques for Object Detection from Complex Background Image,”Perlis, Malaysia: School of Microelectronic Engineering, University Malaysia Perlis, 2018.

ANNEXURE

PYTHON MODULES

timer_func.py

```
import time

days = 0
hours = 0
minutes = 0
seconds = 0
n = 1
m = 1
p = 1
rate = 100
#start_time = time.time()
#new_time = time.time()
#timer = new_time-start_time
while True:
    time.sleep(1)
    seconds = seconds + 1

    if seconds % (n*61) == 0:
        #n = n+1
        minutes = minutes + 1
        seconds = 1

        if minutes % (m*61) == 0:
            #m = m+1
            hours = hours + 1
            minutes = 1

            if hours % (p*25) == 0:
                #p = p+1
                days = days + 1
                hours = 1

    fees = (rate*hours)+rate
    print('days=', days)
    print('hours=', hours)
    print('minutes=', minutes)
    print('seconds=', seconds)
    print('parking fee', fees)
    print('**********************')
```

roipoly.py

```
"""Draw polygon regions of interest (ROIs) in matplotlib images,
similar to Matlab's roipoly function.
"""

import sys
import logging
import warnings

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.path import Path as MplPath
from matplotlib.widgets import Button

logger = logging.getLogger(__name__)

warnings.simplefilter('always', DeprecationWarning)


def deprecation(message):
    warnings.warn(message, DeprecationWarning)


class RoiPoly:

    def __init__(self, fig=None, ax=None, color='b',
                 roicolor=None, show_fig=True, close_fig=True):
        """

        Parameters
        ----------
        fig: matplotlib figure
            Figure on which to create the ROI
        ax: matplotlib axes
            Axes on which to draw the ROI
        color: str
           Color of the ROI
        roicolor: str
            deprecated, use `color` instead
        show_fig: bool
            Display the figure upon initializing a RoiPoly object
        close_fig: bool
            Close the figure after finishing ROI drawing
        """
```

```
    if roicolor is not None:
        deprecation("Use 'color' instead of 'roicolor'!")
        color = roicolor

    if fig is None:
        fig = plt.gcf()
    if ax is None:
        ax = plt.gca()

    self.start_point = []
    self.end_point = []
    self.previous_point = []
    self.x = []
    self.y = []
    self.line = None
    self.completed = False  # Has ROI drawing completed?
    self.color = color
    self.fig = fig
    self.ax = ax
    self.close_figure = close_fig

    self.__cid1 = self.fig.canvas.mpl_connect(
        'motion_notify_event', self.__motion_notify_callback)
    self.__cid2 = self.fig.canvas.mpl_connect(
        'button_press_event', self.__button_press_callback)

    if show_fig:
        self.show_figure()

@staticmethod
def show_figure():
    if sys.flags.interactive:
        plt.show(block=False)
    else:
        plt.show(block=True)

def get_mask(self, image):
    """Get binary mask of the ROI polygon.

    Parameters
    ----------
    image: numpy array (2D)
        Image that the mask should be based on. Only used for determining
        the shape of the binary mask (which is made equal to the shape of  the image)
```

```
    Returns
    -------
    numpy array (2D)

    """
    ny, nx = np.shape(image)
    poly_verts = ([(self.x[0], self.y[0])]
                  + list(zip(reversed(self.x), reversed(self.y))))
    # Create vertex coordinates for each grid cell...
    # (<0,0> is at the top left of the grid in this system)
    x, y = np.meshgrid(np.arange(nx), np.arange(ny))
    x, y = x.flatten(), y.flatten()
    points = np.vstack((x, y)).T

    roi_path = MplPath(poly_verts)
    mask = roi_path.contains_points(points).reshape((ny, nx))
    return mask

def display_roi(self, **linekwargs):
    line = plt.Line2D(self.x + [self.x[0]], self.y + [self.y[0]],
                      color=self.color, **linekwargs)
    ax = plt.gca()
    ax.add_line(line)
    plt.draw()

def get_mean_and_std(self, image):
    """Get statistics about pixel values of an image inside the ROI.

    Parameters
    ----------
    image: numpy array (2D)
       Image on which the statistics should be calculated

    Returns
    -------
    list of float:
       mean and standard deviation of the pixel values inside the ROI

    """
    mask = self.get_mask(image)
    mean = np.mean(np.extract(mask, image))
    std = np.std(np.extract(mask, image))
    return mean, std
```

```
def display_mean(self, image, **textkwargs):
    """Display statistics about pixel values of an image inside the ROI.

    Parameters
    ----------
    image: numpy array (2D)
        Image on which the statistics should be calculated

    Returns
    -------
    None

    """
    mean, std = self.get_mean_and_std(image)
    string = "%.3f +- %.3f" % (mean, std)
    plt.text(self.x[0], self.y[0],
             string, color=self.color,
             bbox=dict(facecolor='w', alpha=0.6), **textkwargs)

def get_roi_coordinates(self):
    """Get co-ordinates of the ROI polygon.

    Returns
    -------
    numpy array (2D)
    """
    roi_coordinates = list(zip(self.x, self.y))
    return roi_coordinates

def __motion_notify_callback(self, event):
    if event.inaxes == self.ax:
        x, y = event.xdata, event.ydata
        if ((event.button is None or event.button == 1) and
                self.line is not None):
            # Move line around
            x_data = [self.previous_point[0], x]
            y_data = [self.previous_point[1], y]
            logger.debug("draw line x: {} y: {}".format(x_data, y_data))
            self.line.set_data(x_data, y_data)
            self.fig.canvas.draw()

def __button_press_callback(self, event):
    if event.inaxes == self.ax:
        x, y = event.xdata, event.ydata
        ax = event.inaxes
```

```
        if event.button == 1 and not event.dblclick:
            logger.debug("Received single left mouse button click")
            if self.line is None:  # If there is no line, create a line
                self.line = plt.Line2D([x, x], [y, y],
                                       marker='o', color=self.color)
                self.start_point = [x, y]
                self.previous_point = self.start_point
                self.x = [x]
                self.y = [y]

                ax.add_line(self.line)
                self.fig.canvas.draw()
                # Add a segment
            else:
                # If there is a line, create a segment
                x_data = [self.previous_point[0], x]
                y_data = [self.previous_point[1], y]
                logger.debug(
                    "draw line x: {} y: {}".format(x_data, y_data))
                self.line = plt.Line2D(x_data, y_data,
                                       marker='o', color=self.color)
                self.previous_point = [x, y]
                self.x.append(x)
                self.y.append(y)

                event.inaxes.add_line(self.line)
                self.fig.canvas.draw()

        elif (((event.button == 1 and event.dblclick) or
               (event.button == 3 and not event.dblclick)) and
              self.line is not None):
            # Close the loop and disconnect
            logger.debug("Received single right mouse button click or "
                         "double left click")
            self.fig.canvas.mpl_disconnect(self.__cid1)
            self.fig.canvas.mpl_disconnect(self.__cid2)

            self.line.set_data([self.previous_point[0],
                                self.start_point[0]],
                               [self.previous_point[1],
                                self.start_point[1]])
            ax.add_line(self.line)
            self.fig.canvas.draw()
            self.line = None
            self.completed = True
```

```
            if not sys.flags.interactive and self.close_figure:
                #  Figure has to be closed so that code can continue
                plt.close(self.fig)

    # For compatibility with old version
    def displayMean(self, *args, **kwargs):
        deprecation("Use 'display_mean' instead of 'displayMean'!")
        return self.display_mean(*args, **kwargs)

    def getMask(self, *args, **kwargs):
        deprecation("Use 'get_mask()' instead of 'getMask'!")
        return self.get_mask(*args, **kwargs)

    def displayROI(self, *args, **kwargs):
        deprecation("Use 'display_roi' instead of 'displayROI'!")
        return self.display_roi(*args, **kwargs)


class MultiRoi:
    def __init__(self,
                 fig=None, ax=None,
                 roi_names=None,
                 color_cycle=('b', 'g', 'r', 'c', 'm', 'y', 'k')
                 ):
        """

        Parameters
        ----------
        fig: matplotlib figure
            Figure on which to draw the ROIs
        ax: matplotlib axes
           Axes on which to draw the ROIs
        roi_names: list of str
            Optional names for the ROIs to draw.
            The ROIs can later be retrieved by using these names as keys for
            the `self.rois` dictionary. If None, consecutive numbers are used
            as ROI names
        color_cycle: list of str
            List of matplotlib colors for the ROIs
        """

        if fig is None:
            fig = plt.gcf()
        if ax is None:
            ax = fig.gca()
```

```
        self.color_cycle = color_cycle
        self.roi_names = roi_names
        self.fig = fig
        self.ax = ax
        self.rois = {}

        self.make_buttons()

    def make_buttons(self):
        ax_add_btn = plt.axes([0.7, 0.02, 0.1, 0.04])
        ax_finish_btn = plt.axes([0.81, 0.02, 0.1, 0.04])
        btn_finish = Button(ax_finish_btn, 'Finish')
        btn_finish.on_clicked(self.finish)
        btn_add = Button(ax_add_btn, 'New ROI')
        btn_add.on_clicked(self.add)
        plt.show(block=True)

    def add(self, event):
        """Add a new ROI"""

        # Only draw a new ROI if the previous one is completed
        if self.rois:
            if not all(r.completed for r in self.rois.values()):
                return

        count = len(self.rois)
        idx = count % len(self.color_cycle)
        logger.debug("Creating new ROI {}".format(count))
        if self.roi_names is not None and idx < len(self.roi_names):
            roi_name = self.roi_names[idx]
        else:
            roi_name = str(count + 1)

        self.ax.set_title("Draw ROI '{}'".format(roi_name))
        plt.draw()
        roi = RoiPoly(color=self.color_cycle[idx],
                      fig=self.fig,
                      ax=self.ax,
                      close_fig=False,
                      show_fig=False)
        self.rois[roi_name] = roi

    def finish(self, event):
        logger.debug("Stop ROI drawing")
        plt.close(self.fig)
```

```
# For compatibility with old version
def roipoly(*args, **kwargs):
    deprecation("Import 'RoiPoly' instead of 'roipoly'!")
    return RoiPoly(*args, **kwargs)
```

PROGRAMS ON RASPBERRY PI

park_excel.py

```
import xlsxwriter

workbook = xlsxwriter.Workbook('park1.xlsx')
worksheet = workbook.add_worksheet("My sheet")

cell_format = workbook.add_format({'font_color': 'black'})
cell_format.set_align('center')

worksheet.write('A1', '1', cell_format)
worksheet.write('B1', '2', cell_format)
worksheet.write('C1', '3', cell_format)

worksheet.write('A2', '4', cell_format)
worksheet.write('B2', '5', cell_format)
worksheet.write('C2', '6', cell_format)



workbook.close()
```

loop_dray.py

```
from matplotlib import pyplot as plt
import cv2
from roipoly import RoiPoly
import numpy as np

cap = cv2.VideoCapture(0)
_, img = cap.read()
fh, fw, _ = img.shape
img = cv2.resize(img, (fw, fh))
area = img

while True:
    area = cv2.resize(area, (fw, fh))
    fig = plt.figure()
    plt.imshow(area, interpolation='nearest')
    plt.title('draw zone 1')
    plt.show(block=False)
    roi1 = RoiPoly(color='b', fig=fig)
    roi11 = roi1.get_roi_coordinates()
```

```
    # Show the image with the first ROI
    fig = plt.figure()
    plt.imshow(img, interpolation='nearest')
    roi1.display_roi()
    plt.title('Zone 1')
    plt.show()
    break

rois11 = np.array(roi11)
roiss11 = rois11.astype(int)
fig = plt.figure()
plt.imshow(area, interpolation='nearest')
roi1.display_roi()
file = open("loop6.txt", "w+")
rois11 = np.array(roi11)
np.savetxt('loop6.txt', rois11, fmt="%i")
file.close()
```

loops.py

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
_, img = cap.read()
fh, fw, _ = img.shape
img = cv2.resize(img, (fw, fh))

file = open("loop1.txt", "r")
roiss11 = np.loadtxt("loop1.txt", dtype=int)
file.close()
mask1 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask1, [roiss11], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask1 = cv2.resize(mask1, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask1', mask1)

file = open("loop2.txt", "r")
roiss22 = np.loadtxt("loop2.txt", dtype=int)
file.close()
mask2 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask2, [roiss22], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask2 = cv2.resize(mask2, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask2', mask2)
```

```
file = open("loop3.txt", "r")
roiss33 = np.loadtxt("loop3.txt", dtype=int)
file.close()
mask3 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask3, [roiss33], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask3 = cv2.resize(mask3, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask3', mask3)

file = open("loop4.txt", "r")
roiss44 = np.loadtxt("loop4.txt", dtype=int)
file.close()
mask4 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask4, [roiss44], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask4 = cv2.resize(mask4, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask3', mask3)

file = open("loop5.txt", "r")
roiss55 = np.loadtxt("loop5.txt", dtype=int)
file.close()
mask5 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask5, [roiss55], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask5 = cv2.resize(mask5, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask3', mask3)

file = open("loop6.txt", "r")
roiss66 = np.loadtxt("loop6.txt", dtype=int)
file.close()
mask6 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask6, [roiss66], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask6 = cv2.resize(mask6, (fw, fh), cv2.INTER_NEAREST)

################################################################################################################################
prototxt = 'MobileNetSSD_deploy.prototxt.txt'
model = 'MobileNetSSD_deploy.caffemodel'

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
net = cv2.dnn.readNetFromCaffe(prototxt, model)
```

```
while True:
    _, frame = cap.read()
    (fH, fW) = frame.shape[:2]
    frame = cv2.resize(frame, (fW, fH))
    res1 = cv2.bitwise_and(frame, frame, mask=mask1)
    res2 = cv2.bitwise_and(frame, frame, mask=mask2)
    res3 = cv2.bitwise_and(frame, frame, mask=mask3)
    res4 = cv2.bitwise_and(frame, frame, mask=mask4)
    res5 = cv2.bitwise_and(frame, frame, mask=mask5)
    res6 = cv2.bitwise_and(frame, frame, mask=mask6)

    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)
    blob1 = cv2.dnn.blobFromImage(cv2.resize(res1, (300, 300)), 0.007843, (300, 300), 127.5)
    blob2 = cv2.dnn.blobFromImage(cv2.resize(res2, (300, 300)), 0.007843, (300, 300), 127.5)
    blob3 = cv2.dnn.blobFromImage(cv2.resize(res3, (300, 300)), 0.007843, (300, 300), 127.5)
    blob4 = cv2.dnn.blobFromImage(cv2.resize(res4, (300, 300)), 0.007843, (300, 300), 127.5)
    blob5 = cv2.dnn.blobFromImage(cv2.resize(res5, (300, 300)), 0.007843, (300, 300), 127.5)
    blob6 = cv2.dnn.blobFromImage(cv2.resize(res6, (300, 300)), 0.007843, (300, 300), 127.5)


    net.setInput(blob1)
    detections1 = net.forward()
    if detections1 is not None:
        for i in np.arange(0, detections1.shape[2]):
            confidence = detections1[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx1 = int(detections1[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections1[0, 0, i, 3:7] * dims
            (startX1, startY1, endX1, endY1) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx1], confidence * 100)
            y = startY1 - 15 if startY1 - 15 > 15 else startY1 + 15
            if CLASSES[idx1] == "car" or CLASSES[idx1] == "bus" or CLASSES[idx1] ==
"motorbike" or CLASSES[idx1] == "bottle":
                cv2.putText(frame, label, (startX1, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx1], 2)
                cv2.rectangle(frame, (startX1, startY1), (endX1, endY1), COLORS[idx1], 2)
                cv2.rectangle(res1, (startX1, startY1), (endX1, endY1), COLORS[idx1], 2)

    net.setInput(blob2)
    detections2 = net.forward()
    if detections2 is not None:
        for i in np.arange(0, detections2.shape[2]):
```

```
            confidence = detections2[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx2 = int(detections2[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections2[0, 0, i, 3:7] * dims
            (startX2, startY2, endX2, endY2) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx2], confidence * 100)
            y = startY2 - 15 if startY2 - 15 > 15 else startY2 + 15
            if CLASSES[idx2] == "car" or CLASSES[idx2] == "bus" or CLASSES[idx2] ==
"motorbike" or CLASSES[
                idx2] == "bottle":
                cv2.putText(frame, label, (startX2, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx2], 2)
                cv2.rectangle(frame, (startX2, startY2), (endX2, endY2), COLORS[idx2], 2)
                cv2.rectangle(res2, (startX2, startY2), (endX2, endY2), COLORS[idx2], 2)

    net.setInput(blob3)
    detections3 = net.forward()
    if detections3 is not None:
        for i in np.arange(0, detections3.shape[2]):
            confidence = detections3[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx3 = int(detections3[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections3[0, 0, i, 3:7] * dims
            (startX3, startY3, endX3, endY3) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx3], confidence * 100)
            y = startY3 - 15 if startY3 - 15 > 15 else startY3 + 15
            if CLASSES[idx3] == "car" or CLASSES[idx3] == "bus" or CLASSES[idx3] ==
"motorbike" or CLASSES[
                idx3] == "bottle":
                cv2.putText(frame, label, (startX3, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx3], 2)
                cv2.rectangle(frame, (startX3, startY3), (endX3, endY3), COLORS[idx3], 2)
                cv2.rectangle(res3, (startX3, startY3), (endX3, endY3), COLORS[idx3], 2)

    net.setInput(blob4)
    detections4 = net.forward()
    if detections4 is not None:
        for i in np.arange(0, detections4.shape[2]):
            confidence = detections4[0, 0, i, 2]
            if confidence < 0.2:
                continue
```

```
            idx4 = int(detections4[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections4[0, 0, i, 3:7] * dims
            (startX4, startY4, endX4, endY4) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx4], confidence * 100)
            y = startY4 - 15 if startY4 - 15 > 15 else startY4 + 15
            if CLASSES[idx4] == "car" or CLASSES[idx4] == "bus" or CLASSES[idx4] ==
"motorbike" or CLASSES[
                idx4] == "bottle":
                cv2.putText(frame, label, (startX4, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx4], 2)
                cv2.rectangle(frame, (startX4, startY4), (endX4, endY4), COLORS[idx4], 2)
                cv2.rectangle(res3, (startX4, startY4), (endX4, endY4), COLORS[idx4], 2)

    net.setInput(blob5)
    detections5 = net.forward()
    if detections5 is not None:
        for i in np.arange(0, detections5.shape[2]):
            confidence = detections5[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx5 = int(detections5[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections5[0, 0, i, 3:7] * dims
            (startX5, startY5, endX5, endY5) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx5], confidence * 100)
            y = startY5 - 15 if startY5 - 15 > 15 else startY5 + 15
            if CLASSES[idx5] == "car" or CLASSES[idx5] == "bus" or CLASSES[idx5] ==
"motorbike" or CLASSES[
                idx5] == "bottle":
                cv2.putText(frame, label, (startX5, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx5], 2)
                cv2.rectangle(frame, (startX5, startY5), (endX5, endY5), COLORS[idx5], 2)
                cv2.rectangle(res5, (startX5, startY5), (endX5, endY5), COLORS[idx5], 2)

    net.setInput(blob6)
    detections6 = net.forward()
    if detections6 is not None:
        for i in np.arange(0, detections6.shape[2]):
            confidence = detections6[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx6 = int(detections6[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
```

```
        box = detections6[0, 0, i, 3:7] * dims
        (startX6, startY6, endX6, endY6) = box.astype("int")
        label = "{}: {:.2f}%".format(CLASSES[idx6], confidence * 100)
        y = startY6 - 15 if startY6 - 15 > 15 else startY6 + 15
        if CLASSES[idx6] == "car" or CLASSES[idx6] == "bus" or CLASSES[idx6] ==
"motorbike" or CLASSES[
            idx6] == "bottle":
            cv2.putText(frame, label, (startX6, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx6], 2)
            cv2.rectangle(frame, (startX6, startY6), (endX6, endY6), COLORS[idx6], 2)
            cv2.rectangle(res6, (startX6, startY6), (endX6, endY6), COLORS[idx6], 2)

    cv2.imshow("Frame", frame)
    cv2.imshow("Slot1", res1)
    cv2.imshow("Slot2", res2)
    cv2.imshow("Slot3", res3)
    cv2.imshow("Slot4", res4)
    cv2.imshow("Slot5", res5)
    cv2.imshow("Slot6", res6)


    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

cv2.destroyAllWindows()
cap.stop()
```

park_app.py

```
from flask import Flask, render_template
from openpyxl import load_workbook
import cv2
import numpy as np
import xlsxwriter

cap = cv2.VideoCapture(0)
_, img = cap.read()
fh, fw, _ = img.shape
img = cv2.resize(img, (fw, fh))
```

```
file = open("loop1.txt", "r")
roiss11 = np.loadtxt("loop1.txt", dtype=int)
file.close()
mask1 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask1, [roiss11], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask1 = cv2.resize(mask1, (fw, fh), cv2.INTER_NEAREST)

file = open("loop2.txt", "r")
roiss22 = np.loadtxt("loop2.txt", dtype=int)
file.close()
mask2 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask2, [roiss22], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask2 = cv2.resize(mask2, (fw, fh), cv2.INTER_NEAREST)
file = open("loop3.txt", "r")
roiss33 = np.loadtxt("loop3.txt", dtype=int)
file.close()
mask3 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask3, [roiss33], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask3 = cv2.resize(mask3, (fw, fh), cv2.INTER_NEAREST)

file = open("loop4.txt", "r")
roiss44 = np.loadtxt("loop4.txt", dtype=int)
file.close()
mask4 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask4, [roiss44], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask4 = cv2.resize(mask4, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask3', mask3)

file = open("loop5.txt", "r")
roiss55 = np.loadtxt("loop5.txt", dtype=int)
file.close()
mask5 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask5, [roiss55], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask5 = cv2.resize(mask5, (fw, fh), cv2.INTER_NEAREST)
#cv2.imshow('mask3', mask3)

file = open("loop6.txt", "r")
roiss66 = np.loadtxt("loop6.txt", dtype=int)
file.close()
mask6 = np.zeros((fh, fw), dtype=np.uint8)
cv2.drawContours(mask6, [roiss66], -1, (255, 255, 255), -1, cv2.LINE_AA)
mask6 = cv2.resize(mask6, (fw, fh), cv2.INTER_NEAREST)
```

```
prototxt = 'MobileNetSSD_deploy.prototxt.txt'
model = 'MobileNetSSD_deploy.caffemodel'

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
net = cv2.dnn.readNetFromCaffe(prototxt, model)

app = Flask(__name__)

seconds1 = 0
minutes1 = 0
hours1 = 0
days1 = 0
rate = 100
seconds2 = 0
minutes2 = 0
hours2 = 0
days2 = 0
seconds3 = 0
minutes3 = 0
hours3 = 0
days3 = 0
seconds4 = 0
minutes4 = 0
hours4 = 0
days4 = 0
seconds5 = 0
minutes5 = 0
hours5 = 0
days5 = 0
seconds6 = 0
minutes6 = 0
hours6 = 0
days6 = 0


@app.route('/')
def index():
    book1 = load_workbook("park1.xlsx")
    data1 = book1.active
    data1["A1"] = 'FREE'
```

```
    data1["B1"] = 'FREE'
    data1["C1"] = 'FREE'
    data1["A2"] = 'FREE'
    data1["B2"] = 'FREE'
    data1["C2"] = 'FREE'
    book1.save(filename="park1.xlsx")
    book2 = load_workbook("park2.xlsx")
    data2 = book2.active
    data2["A1"] = '00:00:00'
    data2["B1"] = '00:00:00'
    data2["C1"] = '00:00:00'
    data2["A2"] = '00:00:00'
    data2["B2"] = '00:00:00'
    data2["C2"] = '00:00:00'
    book2.save(filename="park2.xlsx")

    book3 = load_workbook("park3.xlsx")
    data3 = book3.active
    data3["A1"] = 'Rs.0.0'
    data3["B1"] = 'Rs.0.0'
    data3["C1"] = 'Rs.0.0'
    data3["A2"] = 'Rs.0.0'
    data3["B2"] = 'Rs.0.0'
    data3["C2"] = 'Rs.0.0'
    book3.save(filename="park3.xlsx")

    global cap, mask1, mask2, mask3, seconds1, minutes1, hours1, days1, seconds2, minutes2,
hours2, days2
    global seconds3, minutes3, hours3, days3, seconds4, minutes4, hours4, days4, seconds5,
minutes5, hours5, days5
    global seconds6, minutes6, hours6, days6
    _, frame = cap.read()
    (fH, fW) = frame.shape[:2]
    frame = cv2.resize(frame, (fW, fH))
    res1 = cv2.bitwise_and(frame, frame, mask=mask1)
    res2 = cv2.bitwise_and(frame, frame, mask=mask2)
    res3 = cv2.bitwise_and(frame, frame, mask=mask3)
    res4 = cv2.bitwise_and(frame, frame, mask=mask4)
    res5 = cv2.bitwise_and(frame, frame, mask=mask5)
    res6 = cv2.bitwise_and(frame, frame, mask=mask6)

    blob1 = cv2.dnn.blobFromImage(cv2.resize(res1, (300, 300)), 0.007843, (300, 300), 127.5)
    blob2 = cv2.dnn.blobFromImage(cv2.resize(res2, (300, 300)), 0.007843, (300, 300), 127.5)
    blob3 = cv2.dnn.blobFromImage(cv2.resize(res3, (300, 300)), 0.007843, (300, 300), 127.5)
    blob4 = cv2.dnn.blobFromImage(cv2.resize(res4, (300, 300)), 0.007843, (300, 300), 127.5)
```

```
    blob5 = cv2.dnn.blobFromImage(cv2.resize(res5, (300, 300)), 0.007843, (300, 300), 127.5)
    blob6 = cv2.dnn.blobFromImage(cv2.resize(res6, (300, 300)), 0.007843, (300, 300), 127.5)

    net.setInput(blob1)
    rate = 100
    detections1 = net.forward()
    if detections1 is not None:
        for i in np.arange(0, detections1.shape[2]):
            confidence = detections1[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx1 = int(detections1[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections1[0, 0, i, 3:7] * dims
            (startX1, startY1, endX1, endY1) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx1], confidence * 100)
            y = startY1 - 15 if startY1 - 15 > 15 else startY1 + 15
            if CLASSES[idx1] == "car" or CLASSES[idx1] == "bus" or CLASSES[idx1] ==
"motorbike" or CLASSES[
                idx1] == "bottle":
                data1["A1"] = 'OCCUPIED'
                book1.save(filename="park1.xlsx")

                workbook = load_workbook(filename="park2.xlsx")
                sheet = workbook.active
                seconds1 = seconds1 + 1
                if seconds1 % 61 == 0:
                    minutes1 = minutes1 + 1
                    seconds1 = 1
                    if minutes1 % 61 == 0:
                        hours1 = hours1 + 1
                        minutes1 = 1
                        if hours1 % 25 == 0:
                            days1 = days1 + 1
                            hours1 = 1
                sheet['A1'] = str(days1).rjust(2, '0') + ' days,' + str(hours1).rjust(2, '0') + ':' + str(
                    minutes1).rjust(2, '0') + ':' + str(seconds1).rjust(2, '0')
                workbook.save(filename="park2.xlsx")
                data2 = sheet

                fees = (rate * hours1) + rate
                workbook = load_workbook(filename="park3.xlsx")
                sheet = workbook.active
                sheet['A1'] = 'Rs.' + str(fees)
                workbook.save(filename="park3.xlsx")
```

```
            data3 = sheet

        else:
            print('no vehicle in A1...')
            seconds1 = 0
            minutes1 = 0
            hours1 = 0
            days1 = 0


    net.setInput(blob2)
    detections2 = net.forward()
    if detections2 is not None:
        for i in np.arange(0, detections2.shape[2]):
            confidence = detections2[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx2 = int(detections2[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections2[0, 0, i, 3:7] * dims
            (startX2, startY2, endX2, endY2) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx2], confidence * 100)
            y = startY2 - 15 if startY2 - 15 > 15 else startY2 + 15
            if CLASSES[idx2] == "car" or CLASSES[idx2] == "bus" or CLASSES[idx2] ==
"motorbike":
                data1["B1"] = 'OCCUPIED'
                book1.save(filename="park1.xlsx")

                workbook = load_workbook(filename="park2.xlsx")
                sheet = workbook.active
                seconds2 = seconds2 + 1
                if seconds2 % 61 == 0:
                    minutes2 = minutes2 + 1
                    seconds2 = 1
                    if minutes2 % 61 == 0:
                        hours2 = hours2 + 1
                        minutes2 = 1
                        if hours2 % 25 == 0:
                            days2 = days2 + 1
                            hours2 = 1
                sheet['B1'] = str(days2).rjust(2, '0') + ' days,' + str(hours2).rjust(2, '0') + ':' + str(
                    minutes2).rjust(2, '0') + ':' + str(seconds2).rjust(2, '0')
                workbook.save(filename="park2.xlsx")
                data2 = sheet
```

```
            fees = (rate * hours2) + rate
            workbook = load_workbook(filename="park3.xlsx")
            sheet = workbook.active
            sheet['B1'] = 'Rs.' + str(fees)
            workbook.save(filename="park3.xlsx")
            data3 = sheet

         else:
            print('no vehicle in B1...')
            seconds2 = 0
            minutes2 = 0
            hours2 = 0
            days2 = 0


   net.setInput(blob3)
   detections3 = net.forward()
   if detections3 is not None:
      for i in np.arange(0, detections3.shape[2]):
         confidence = detections3[0, 0, i, 2]
         if confidence < 0.2:
            continue
         idx3 = int(detections3[0, 0, i, 1])
         dims = np.array([fW, fH, fW, fH])
         box = detections3[0, 0, i, 3:7] * dims
         (startX3, startY3, endX3, endY3) = box.astype("int")
         label = "{}: {:.2f}%".format(CLASSES[idx3], confidence * 100)
         y = startY3 - 15 if startY3 - 15 > 15 else startY3 + 15
         if CLASSES[idx3] == "car" or CLASSES[idx3] == "bus" or CLASSES[idx3] ==
"motorbike" or CLASSES[
            idx3] == "bottle":
            data1["C1"] = 'OCCUPIED'
            book1.save(filename="park1.xlsx")

            workbook = load_workbook(filename="park2.xlsx")
            sheet = workbook.active
            seconds3 = seconds3 + 1
            if seconds3 % 61 == 0:
               minutes3 = minutes3 + 1
               seconds3 = 1
               if minutes3 % 61 == 0:
                  hours3 = hours3 + 1
                  minutes3 = 1
                  if hours3 % 25 == 0:
                     days3 = days3 + 1
```

```
                    hours3 = 1
            sheet['C1'] = str(days3).rjust(2, '0') + ' days,' + str(hours3).rjust(2, '0') + ':' + str(
                minutes3).rjust(2, '0') + ':' + str(seconds3).rjust(2, '0')
            workbook.save(filename="park2.xlsx")
            data2 = sheet

            fees = (rate * hours3) + rate
            workbook = load_workbook(filename="park3.xlsx")
            sheet = workbook.active
            sheet['C1'] = 'Rs.' + str(fees)
            workbook.save(filename="park3.xlsx")
            data3 = sheet

        else:
            print('no vehicle in C1...')
            seconds3 = 0
            minutes3 = 0
            hours3 = 0
            days3 = 0


    net.setInput(blob4)
    detections4 = net.forward()
    if detections4 is not None:
        for i in np.arange(0, detections4.shape[2]):
            confidence = detections4[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx4 = int(detections4[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections4[0, 0, i, 3:7] * dims
            (startX4, startY4, endX4, endY4) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx4], confidence * 100)
            y = startY4 - 15 if startY4 - 15 > 15 else startY4 + 15
            if CLASSES[idx4] == "car" or CLASSES[idx4] == "bus" or CLASSES[idx4] ==
"motorbike" or CLASSES[
                idx4] == "bottle":
                data1["A2"] = 'OCCUPIED'
                book1.save(filename="park1.xlsx")

                workbook = load_workbook(filename="park2.xlsx")
                sheet = workbook.active
                seconds4 = seconds4 + 1
                if seconds4 % 61 == 0:
                    minutes4 = minutes4 + 1
```

```
                seconds4 = 1
                if minutes4 % 61 == 0:
                    hours4 = hours4 + 1
                    minutes4 = 1
                    if hours4 % 25 == 0:
                        days4 = days4 + 1
                        hours4 = 1
            sheet['A2'] = str(days4).rjust(2, '0') + ' days,' + str(hours4).rjust(2, '0') + ':' + str(
                minutes4).rjust(2, '0') + ':' + str(seconds4).rjust(2, '0')
            workbook.save(filename="park2.xlsx")
            data2 = sheet

            fees = (rate * hours4) + rate
            workbook = load_workbook(filename="park3.xlsx")
            sheet = workbook.active
            sheet['A2'] = 'Rs.' + str(fees)
            workbook.save(filename="park3.xlsx")
            data3 = sheet

        else:
            print('no vehicle in A2...')
            seconds4 = 0
            minutes4 = 0
            hours4 = 0
            days4 = 0


    net.setInput(blob5)
    detections5 = net.forward()
    if detections5 is not None:
        for i in np.arange(0, detections5.shape[2]):
            confidence = detections5[0, 0, i, 2]
            if confidence < 0.2:
                continue
            idx5 = int(detections5[0, 0, i, 1])
            dims = np.array([fW, fH, fW, fH])
            box = detections5[0, 0, i, 3:7] * dims
            (startX5, startY5, endX5, endY5) = box.astype("int")
            label = "{}: {:.2f}%".format(CLASSES[idx5], confidence * 100)
            y = startY5 - 15 if startY5 - 15 > 15 else startY5 + 15
            if CLASSES[idx5] == "car" or CLASSES[idx5] == "bus" or CLASSES[idx5] ==
"motorbike" or CLASSES[
                idx5] == "bottle":
                data1["B2"] = 'OCCUPIED'
                book1.save(filename="park1.xlsx")
```

```
            workbook = load_workbook(filename="park2.xlsx")
            sheet = workbook.active
            seconds5 = seconds5 + 1
            if seconds5 % 61 == 0:
               minutes5 = minutes5 + 1
               seconds5 = 1
               if minutes5 % 61 == 0:
                  hours5 = hours5 + 1
                  minutes5 = 1
                  if hours5 % 25 == 0:
                     days5 = days5 + 1
                     hours5 = 1
            sheet['B2'] = str(days5).rjust(2, '0') + ' days,' + str(hours5).rjust(2, '0') + ':' + str(
               minutes5).rjust(2, '0') + ':' + str(seconds5).rjust(2, '0')
            workbook.save(filename="park2.xlsx")
            data2 = sheet

            fees = (rate * hours5) + rate
            workbook = load_workbook(filename="park3.xlsx")
            sheet = workbook.active
            sheet['B2'] = 'Rs.' + str(fees)
            workbook.save(filename="park3.xlsx")
            data3 = sheet

         else:
            print('no vehicle in B2...')
            seconds5 = 0
            minutes5 = 0
            hours5 = 0
            days5 = 0


net.setInput(blob6)
detections6 = net.forward()
if detections6 is not None:
   for i in np.arange(0, detections6.shape[2]):
      confidence = detections6[0, 0, i, 2]
      if confidence < 0.2:
         continue
      idx6 = int(detections6[0, 0, i, 1])
      dims = np.array([fW, fH, fW, fH])
      box = detections6[0, 0, i, 3:7] * dims
      (startX6, startY6, endX6, endY6) = box.astype("int")
```

```
        label = "{}: {:.2f}%".format(CLASSES[idx6], confidence * 100)
        y = startY6 - 15 if startY6 - 15 > 15 else startY6 + 15
        if CLASSES[idx6] == "car" or CLASSES[idx6] == "bus" or CLASSES[idx6] ==
"motorbike" or CLASSES[
            idx6] == "bottle":
            data1["C2"] = 'OCCUPIED'
            book1.save(filename="park1.xlsx")

            workbook = load_workbook(filename="park2.xlsx")
            sheet = workbook.active
            seconds6 = seconds6 + 1
            if seconds6 % 61 == 0:
               minutes6 = minutes6 + 1
               seconds6 = 1
               if minutes6 % 61 == 0:
                  hours6 = hours6 + 1
                  minutes6 = 1
                  if hours6 % 25 == 0:
                     days6 = days6 + 1
                     hours6 = 1
            sheet['C2'] = str(days6).rjust(2, '0') + ' days,' + str(hours6).rjust(2, '0') + ':' + str(
               minutes6).rjust(2, '0') + ':' + str(seconds6).rjust(2, '0')
            workbook.save(filename="park2.xlsx")
            data2 = sheet

            fees = (rate * hours6) + rate
            workbook = load_workbook(filename="park3.xlsx")
            sheet = workbook.active
            sheet['C2'] = 'Rs.' + str(fees)
            workbook.save(filename="park3.xlsx")
            data3 = sheet

        else:
            print('no vehicle in C2...')
            seconds6 = 0
            minutes6 = 0
            hours6 = 0
            days6 = 0

    return render_template('park_table.html', sheet1=data1, sheet2=data2, sheet3=data3)


if __name__ == "__main__":
    app.run(host="localhost", port=int("5000"))
```

WEBPAGE

index.html

```
<!DOCTYPE html>
<html>
<title>DRIVER</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="refresh" content="1">
<link rel="stylesheet" href="w3.css">
<style>
* {
  box-sizing: border-box;
}

/* Create two equal columns that floats next to each other */
.column {
  float: left;
  width: 50%;
  padding: 10px;
  height: 550px; /* Should be removed. Only for demonstration */
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Header/Logo Title */
.header {
  padding: 100px;
  text-align: center;
  color: white;
  font-size: 20px;
  background-image: url('/static/park1.jpg');
}

.content_t {
  padding: 100px;
  text-align: center;
  color: black;
  font-size: 30px;
}
```

```
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}

table, th, td {
  border: 1px solid black;
  text-align: center;
  font-size: 30px;
}
</style>
<body>
<div class="header">
<h1><b>VISUALS BASED SMART PARKING AND TARIFF SYSTEM</b></h1>
</div>

<div class="content_t">
<body>
                <div align="center">
                                <h3><b>PARKING SLOTS STATUS</b></h3>
                                <table >
                                {% for row in range(1, sheet1.max_row + 1) %}
                                <tr>
                                        {% for col in range(1, sheet1.max_column + 1): %}

                                        {% if sheet1.cell(row, col).value=='OCCUPIED' %}
                                        <td bgcolor="red" style="width:200px">{{ sheet1.cell(row,
col).value }}</td>
                                        {% else %}
```

```
                                        <td bgcolor="green" style="width:200px">{{
sheet1.cell(row, col).value }}</td>
                                        {% endif %}

                                        {% endfor %}
                                </tr>
                                {% endfor %}
                                </table>
                                </h1>
                        </div>
                        <div class="row">
                        <div class="column" align="left">
                                <h3 ><b>PARKING TIME</b></h3>
                                <table>
                                {% for row in range(1, sheet2.max_row + 1) %}
                                <tr>
                                        {% for col in range(1, sheet2.max_column + 1): %}

                                        <td>{{ sheet2.cell(row, col).value }}</td>
                                        {% endfor %}
                                </tr>
                                {% endfor %}
                                </table>
                        </div>

                        <div class="column" align="right">
                                <h3 ><b>PARKING TARIFF</b></h3>
                                <table>
                                {% for row in range(1, sheet3.max_row + 1) %}
                                <tr>
                                        {% for col in range(1, sheet3.max_column + 1): %}

                                        <td>{{ sheet3.cell(row, col).value }}</td>
                                        {% endfor %}
                                </tr>
                                {% endfor %}
                                </table>
                        </div>
                        </div>

                </body>
</div>

</body>
</html>
```

style.css

```
/* * Reset all elements */
* {
  margin: 0;
  padding: 0;
}


/* * HTML elements */
body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
  font-size: 18px;
  font-weight: normal;
  line-height: 1.5em;
  width: 800px;
  margin: auto;
}


/* * Local selectors */
#container {
  margin: 0px auto;
  margin-top: 40px;
  width: 100%;
  height: 450px;
  border: 10px #333 solid;
  background-color: black;
}

#container #videoElement {
  width: auto;
  height: 100%;
  margin-left: auto;
  margin-right: auto;
  display: block;
  background-color: black;
}

#control {
  margin-top: 40px;
}
```