DEPARTMENT OF COMPUTER SCIENCE ENGINEERING, SCHOOL OF ENGINEERING AND TECHNOLOGY, SHARDA UNIVERSITY, GREATER NOIDA

SMART CAR PARKING SYSTEM

A project submitted

In partial fulfillment of the requirements for

the degree of Bachelor of Technology in

Computer Science and Engineering

By

Aish Aggarwal (2018015448)
Soni Jain (2018016478)
Sukriti Sachan (2018006901)

Project Report

OF PROJECT-II (CSP 495)

CERTIFICATE

This is to certify that the report entitled **"Smart Car Parking System"** submitted by "Aish Aggarwal, Soni Jain, Sukriti Sachan"to Sharda University, towards the fulfillment of requirements of the degree of **"Bachelor of Technology"** is record of bonafide final year Project work carried out by him in the **"**Department of Computer Science and Engineering, School of Engineering and Technology, Sharda University**"**.

The results/findings contained in this Project have not been submitted in part or full to any other University/Institute for award of any other Degree/Diploma.

Signature of Supervisor

Name: Mrs. Rani Astya

Designation: Asst. Prof (CSE)
(CSE)

Signature of Co-supervisor

Name:

Designation: Asst. Professor

Signature of Head of Department

Name: Prof. (Dr.) Nitin Rakesh

Place: Greater Noida

Date:

Signature of External

Examiner Date:

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider ourselves very lucky and honored to have so many wonderful people lead us through in the completion of this project.

First and foremost we would like to thank Dr. Nitin Rakesh, HOD, CSE who gave us an opportunity to undertake this project.

My grateful thanks to **Mrs. Rani Astya** for their guidance in our project work.

Mrs. Rani Astya, who in spite of being extraordinarily busy with academics, took time out to hear, guide, and keep us on the correct path. We do not know where we would have been without her help.

The CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Name and signature of Students

Aish Aggarwal (2018015448)
Soni Jain (2018016478)
Sukriti Sachan (2018006901)

Abstract

The aim of the proposed approach is to provide a user-friendly,real-time and sustainable car parking system.Population in city is increasing multi-fold in present scenario,hence so are the number of car owners,which leads to need of an efficient car parking system which can be used by people at ease.There have been many good approaches earlier in order to contribute to the solution to this issue, but a considerable demerit is that many of them use a lot of hardware equipments which in turn leads to humongous expenditure and also when hardware is used considerably more, maintenance requirement also increases,also there is not an efficient monitoring system,which results in user wasting time to locate an empty slot amongst multiple parking lanes.In our approach we are minimizing use of hardware, along with we are using efficient and present age algorithms which can give maximum accuracy at a great speed,which will overcome the demerits of the current methodologies and can be used at a large scale so that the parking issues which are being faced can be curbed. The proposed system aims at providing a feasible and easy algorithm in order to provide functionality that can not only minimize the problem of conventional parking systems but will also be helpful as per the financial perspective. This algorithm can be implemented at a larger scale at a very less expense around the country.

Index Terms:Car Parking system,monitoring system,minimal hardware use

Contents

Chapter1: INTRODUCTION

In recent years, the world's population has continued to increase, and the complexity of transportation has increased significantly. Due to the rapid increase in the number of vehicles in recent years, searching for an empty parking space has become a serious problem for drivers in dense city traffic areas. Currently, in many public places such as stadiums, market squares, hospitals, shopping malls and airports, there is an obvious shortage of free spaces for cars, so governments are striving to improve their existing transportation systems and infrastructure. Parking becomes one of the major problems facing cities, and increasing costs. An ordinary parking lot has many problems, such as how to control the number of cars inside, how to Monitor internal/external movement of Parking spaces, and what are the safety measures available in the parking lot. Well, our project helps users find parking spaces in a given architecture. The article proposes an intelligent car parking system that will help users solve the problem of finding a parking space and minimize the time spent searching for the nearest free parking. The purpose of this project is to solve these problems by designing a system parking lot with a machine learning-based system that uses image processing and a camera to run the entire operation. The system is designed so that results are faster, more flexible and can meet market demands. It can also meet the aggregate needs of valuable parking that can keep the car very carefully, there is also a high level of security and also reduces the wastage of fuel consumption.Only one input device is used for the entire operation. All we have to do is place this camera in the ideal position from where we can see every parking space. Thus ,our approach uses minimal hardware so that our system is cost effective. Also ,High level of security is provided as the cameras would be installed for the image capture.

The conventional parking system simply means searching for vacant spaces around the parking lot, wasting time as well as fuel and if you don't get one then it becomes a cherry on the cake even. The smart car parking system aims to provide a relaxation from the problems of vacant spaces availability as well as saves our time and fuel also. Also, the proposed approach focuses on providing lesser expenditure as well if executed on a larger scale.

1.1 PROBLEM DEFINITION:

The traditional parking system lacks the facility of smart monitoring arrangement which leads to loss of time in locating a parking space for vehicle.Situation worsens in case of multiple parking lanes.In order to avoid these hassles, some proposed solution can be increasing car parking spaces and achieving at least minimum parking demands, increasing curbside parking provisions, underground car parking, etc, but these things don’t really prevent the problems.

Smart Parking includes the usage of affordable sensors, real-time data and applications that permits users to be able to monitor vacant and occupied parking slots. The purpose is to encourage automation and reduce manual intervention while searching for the optimal parking floor, spot and even lot. With the increasing traffic and development, the major problem after traffic jams is the parking. At public places like hospitals, malls etc it consumes a lot of time to go to the parking space and check for space, and if there is no space then the

time and fuel both get wasted. With the development of technologies, the researcher started working on the daily life problems of common people and thereby proposed different-different algorithms for the same.

The project's goal is to facilitate an efficient, cheaper and problem solving algorithm in order to achieve an efficient parking system algorithm focusing on day to day problems faced by people. This project aims to create an algorithm that will aid in the availability of vacant parking slots. The project has been divided into several modules starting from classifier training and followed by the algorithm in order to mark the parking regions by reading a yaml file following the coordinates and further using the same while detecting the vacant spaces availability later.The specified algorithm will be use to test different videos.The aim of the proposed system is to provide a faster and more flexible way and to meet the market needs by providing a smart car parking system with efficient features, which will also be financially feasible.

1.2 PROJECT OVERVIEW:

1.2.1 Functional Requirements

1.2.1.1 Introduction

Smart Car Parking System aims to solve these problems by designing a system to control the parking area using a machine learning-based system. This system is designed so that it results in faster, flexible, and can meet market needs. And it can also fulfil the total needs for valuable parking which can hold the vehicle very carefully and also with high security. The proposed approach is focussed on the field of image processing and also includes machine learning. The objective is to develop an efficient car parking system following certain major steps starting from research in the focused field, thereby collecting testing data for the implementation. OpenCV is one of the widely used libraries in the field of detection and so will be using it for the purpose of detection followed by the implementation of confusion matrix in order to detect the efficiency for the algorithm as well. Moving forward, it’ll be followed by marking of the slots in order to give access for the parking algorithm to work upon. This approach will move towards completion with testing and verification as per the requirements.

1.2.1.2 Input

Video: We have collected various videos from different real time sources in which we have applied an algorithm in order to identify or distinguish amongst the vacant and occupied parking slots/regions.

1.2.1.3 Processing

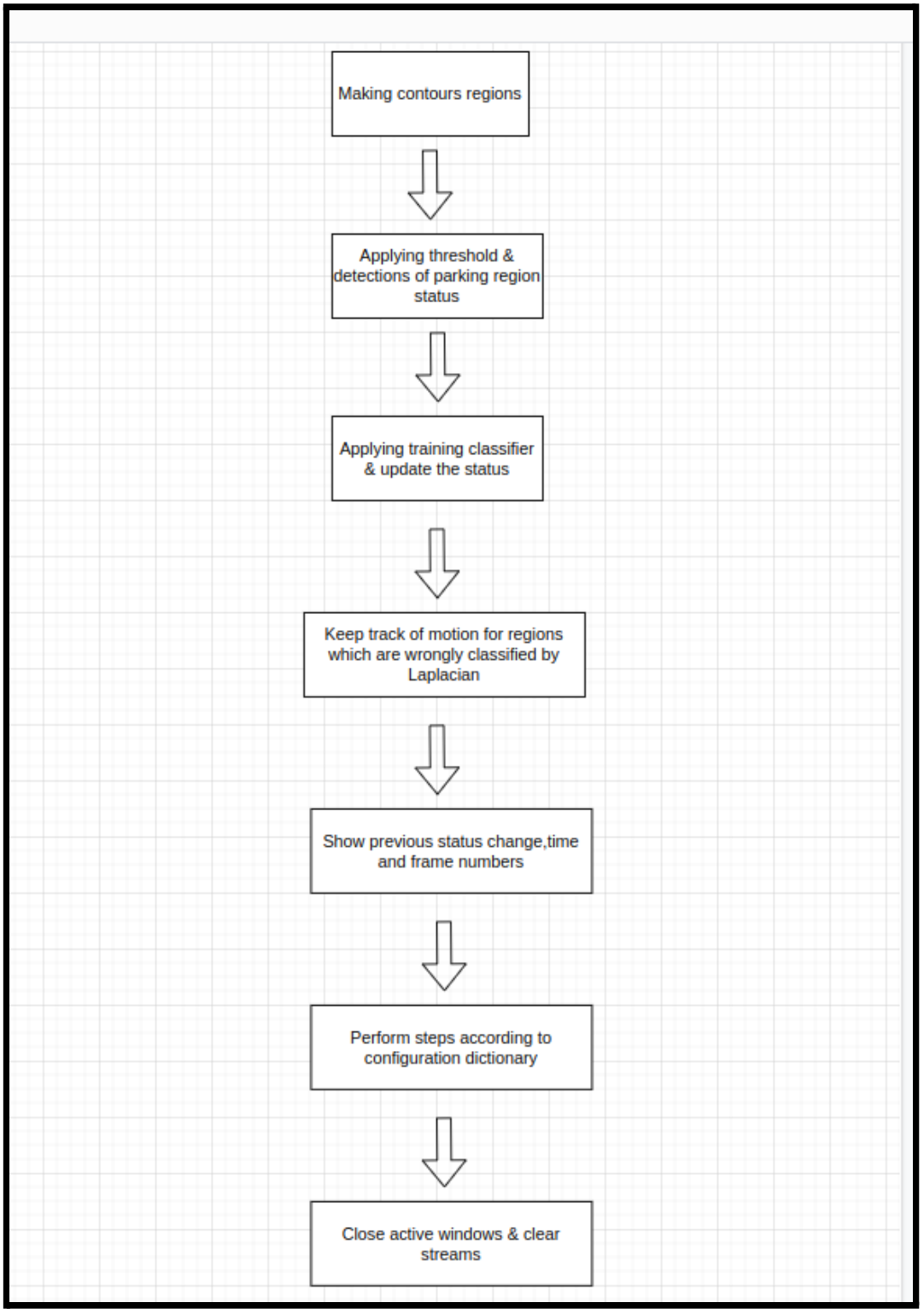

Fig i

1.2.2 Normal Requirements

N1: System must be able to figure out whether an object is a vehicle or not.

N2: System must be able to differentiate amongst vacant and engaged parking spaces.

N3: System must be able to provide a clean and properly visible image of the parking slot which is empty.

1.2.3 Non-Functional Requirements

These are the specifications, as the name implies, that are not specifically correlated with particular functions offered by the device.

1.2.3.1 Availability:

Taking availability into consideration the system is available almost 24*7. Service is available at any point of the day for customers.

1.2.3.2 Usability:

In terms of usability, certain systems may have more superior quality. It is permissible for customers to perform tasks through easy-to-use manual guides. However, this particular system is of excellent quality and user friendly.

1.2.3.3 Scalability:

Scalability is also taken into consideration as an essential and important non-functional requirement. This helps the system develop individual capabilities by addressing the needs and requirements of business operations processes. A proper and effective scalability management process helps ensure efficient customer service.

1.2.3.4 Performance:

The proposed system will be effective, efficient and versatile on the implementation part.

1.2.3.5 Security:

From the security point of view, the developed system will not support any peripheral devices as it creates a chance for the devices to be hacked which is not at all user friendly.

1.3 Hardware specifications:

- Pc/Laptop
- **Processor**: Minimum 1 GHz; Recommended 2GHz or more.

- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- **Hard Drive**: Minimum 32 GB; Recommended 64 GB or more.
- Memory (**RAM**): Minimum 1 GB; Recommended 4 GB or above.

1.4 Software Specifications:

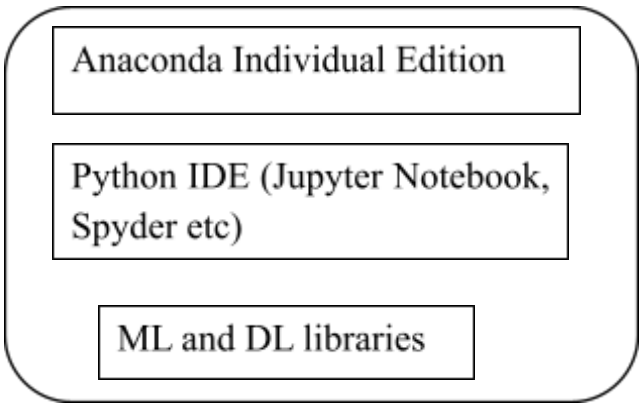

Fig ii

Chapter 2: Literature Survey

2.1 EXISTING SYSTEM:

- **Smart Car parking system using IOT**

    - This approach is basically focused on the multi-storied parking areas along with the concept of Energy Conservation and Management.
    - As per the approach, the vacant spaces will be depicted through lamps and the occupied ones will be virtually stored on the cloud which can further be accessed by the central system.
    - The approach is beneficial as this will be saving the customer's time in parking, along with providing a Real Estate Optimization along with planning. The energy saving parking lights fixtures upto 80% in car parking and this system also provides a better safety constraint as well.
    - This approach requires a scope of improvement, Has Low transmission, Can't be used in outdoor wi-fi systems, due to the limited coverage system.

- **Cloud Based Smart Car Parking System**

    - The approach focuses on equipping or letting the user know the availability status of parking slots
    - The system proposes, collects and filters out the raw data further extracting features by applying filtering and fusion techniques in order to prevent the transmission of extra data over the network. The transformed data will further be sent on cloud in order to process and evaluate using the ML algorithms.
    - The proposed system consists of sensor nodes, the indoor system is in the basement area and that of outdoor one is outside, other than that there are microcontroller devices.

Cloud will act as an intermediary factor amongst the car parking and the application.

- The proposed system minimises the time as well as energy of the user and also provides an effective approach for the cities with humongous traffic. The system also provides the shortest possible path in order to reach the available parking slot.
- The lone drawback in the system is that it becomes quite expensive and tedious to maintain the whole system.

- **NB-IOT based Smart Car parking system**

  - The approach is to provide a mobile application working with the help of NB-IoT technology. The user will need to get registered on the application with all the details, after logging in one can check the availability of the slot, and the user can book the slot. The available and unavailable slots will be displayed with different colours to the user and the timer will be started on parking, if the timer exceeds a notification will be sent to the user.
  - Since it'll be using a global network, therefore this turns out to be beneficial for the user. Also NB-IoT provides a stable span of four to five years.
  - There are few drawbacks of the system as well, including the higher hardware requirements making the system quite expensive. The system demands a lot of time for maintenance. Moreover, the devices on the network must be paid and the addition of telecommunication taxes increases the expenditure as well.

- **Smart Car Parking System Using Image Processing**

  - The proposed system is highly focused to replace the existing systems and has aimed not to use sensors and hence our system makes use of image processing algorithms with the goal to automate the parking with the image from security cameras in the parking spaces.
  - The algorithm tries to locate vacant parking spaces and communicate information to drivers who are entering the parking lots. The system has been implemented in MATLAB, by following some of the steps like that of image identification, acquisition, conversion thereby followed by enhancement and detection.
  - The system ensures reduced mechanical and electrical liability as there is no use of sensors, the system ensures a good level of security as the cameras would be installed for the image capture. Whereas there are things that make the

system less useful like the weather conditions might affect the visibility, also the system requires a power source ensuring the cameras and lighting for camera imaging.

- **Smart car parking system using WI-FI technology**

    - The approach focuses on working on Iot and wi-fi technology providing a real time system. This parking system aims to ensure the user a real time reservation before the time of arrival.
    - The proposed system has been worked out using a mobile application connected to Wi-Fi. The system is making use of infrared sensors in order to monitor the vacant parking slots. The user would need to check for the available spaces with the help of mobile application and further he/she can book the slot that they want to and the information would be shared with the users via notification.
    - The system facilitates a user-friendly environment and provides less time for the user in finding the slots, Also provides more security to the user.
    - The system consumes a lot of power and if the sensor gets affected due to weather it will need to be replaced too.

- **Sensor based smart car parking system**

    - The idea of smart car parking proposed in [10] aims for the single or multi floor parking spaces with the help of a partial automated system. The system also focuses on letting the user reach the nearest parking slot via a less traffic route. The system works with the help of several different equipment like during the entrance of the car. The IR sensors will be identifying its presence thereby opening the gates only if some vacant spaces are present.
    - The IR sensors will help in detecting the available spaces, and the LED's will be indicating the path for the specified paths. The system will be working on the objective of combinations, the several combinations of LED paths will be stored in the microcontroller as per the priority of distance and the turns as well.
    - The system reduces the user's time, helps in traffic management , Also provides a mode of communication with the drivers for the parking.
    - The system facilitates higher performance in order to track the car but is not fully an automated system, Also requires a high level of maintenance.

- **OBSERVATIONS OF LITERATURE SURVEY:**

Reference	Method Used	Advantages	Disadvantages
Denis ashok et. al[1],Jahnvi Nimble et. al[2],ElakyaR et. al[3],Bonde, et. al[6],Vinay Raj Tripathi [15]	IOT	1)Helps in reducing traffic congestion. 2)Controls pollution. 3)Saves time.	1)A lot of hardware will be required which will lead to more cost and will also need more maintenance. 2)It requires a lot of time to install
Wael Alsafery et. al[4],Chowdhury et. al[7],Abhirup Khanna et. al[12],	CLOUD BASED	1) Reducing time. 2) Reduces the energy of the user.	1) The system is quite expensive. 2) Tedious to maintain
Praveen, M., & Harini, V. [5]	NB-IOT	1) Provides deeper network coverage and signals penetrating through underground. 2) NB-IoT provides a stable span of four to five years.	1)Quite expensive. 2)Requires more hardware. 3)Requires a lot of time in maintaining the system. 4) The devices on the network must be paid and the addition of telecommunication taxes increases the expenditure as well.
Soundarya Rajesh et. al[9],Jiang Ruili et. al[11],Jaspreet Kaur et. al [14]	Image Processing	1)Acts a multi purpose parking system helping in various activities. 2)The system integrates several technologies providing every details of the vehicles to the system securely.	1)With time the maintenance of hardware will required. 2) Cost of the system is more.

Fig iii

2.2 PROPOSED APPROACH:

Smart Car Parking System aims to solve these problems by designing a system to control the parking area using a machine learning-based system. This system is designed so that it results in faster, flexible, and can meet market needs. And it can also fulfil the total needs for valuable parking which can hold the vehicle very carefully and also with high security.

The proposed approach is focussing towards the field of image processing as well as machine learning. The objective is to develop an efficient car parking system following certain major steps starting from research in the focused field, thereby collecting testing data for the implementation.

OpenCV is one of the widely used libraries in the field of detection and so will be using it for the purpose of detection followed by the implementation of confusion matrix in order to detect the efficiency for the algorithm as well. Moving forward, it'll be followed by marking of the slots in order to give access for the parking algorithm to work upon. This approach will move towards completion with testing and verification as per the requirements.

The proposed approach consists of various modules beginning with the video capture first, since the approach is based on an algorithm only the video will not be dynamic it will be in static version. The HOG descriptor will be used in order to perform the background subtraction and feature attraction with the help of morphological operations. The process further will be followed by the major logic building for marking the parking regions, for that a YAML file will be created which first be used in order to create the coordinates for the rectangles or the boxes being made in the final result. The yaml file will be having the numbers for the marked regions and the position of the video will be captured for the same which further be used to decode the index of the frame in use.

The algorithm moves forward with the motion detection phase for all the objects and then with the help of Laplacian operators the vacant and the occupied spaces will be detected and the color of the frames will be changed as per the category of the frames.

The intelligent parking system can detect a parking space in a given image and use the camera image to determine whether the given parking space is available or occupied, preferably in real time. It consists of two main steps:

I. **Find all the parking spaces-**
   In a given space that the camera covers, it is very important to know which of the spaces is intended for parking. Most car parking lots indicate the parking area by drawing white rectangles or white lines.

II. **Figure out whether the parking space is vacant or occupied-**

Once we know which ones are parking spaces, we can move on to the second step, which determines if a parking space is occupied or available.

2.3 FEASIBILITY STUDY:

1) TECHNICAL FEASIBILITY

The primary technologies and tools that will be used in the development of the proposed approach are:

- Python
- Python Machine Learning Libraries
- Image Processing Algorithm
- Python IDE (Jupyter Notebook)

All the mentioned technologies are easily available and the required skills will be manageable. Time limitations of the product development and the ease of implementation using these technologies are synchronized.

2) RESOURCE & TIME FEASIBILITY

Resources that are required are:

- Systems for programming
- Timeline Oriented Planning
- Programming Individuals

3) RISK FEASIBILITY

- **Risk Associated with Storage:**

There will not really be any storage related risks as the proposed approach doesn't really need much memory, and just needs a certain dataset in order to implement the required approach.

- **Development Environment Feasibility:**

The Approach will be developed in:

Python IDE (Jupyter Notebook, Spyder etc.)

Python will be used as the main programming language, along with the image processing algorithm and machine learning libraries.

4) **FINANCIAL FEASIBILITY**

The proposed approach will follow the freeware software standards and will earn as per the company policies in different ways in future.

2.4 Risk Management:

2.4.1 Risk Identification

2.4.1.1 Product Size Related

R1 Memory may be squandered as a result of additional lines of code or redundant algorithms.

2.4.1.2 Customer Related

R2 Since its consumer isn't a professional individual and it poses a challenge in interpreting the customer's additional specifications.

R3 If the consumer offers unnecessary details; it can result in an undisclosed danger.

2.4.1.3 Development Environment Related

R4 When a client requests a change or makes an unnecessary alteration later in the implementation process,

it is impossible to change the whole system configuration to accommodate the request.

R5 Inexperience and a lack of tool training can make it challenging to complete project modules.

2.4.1.5 Strategies used to manage Risks

S1 By reducing redundant coding, we can prevent Chance R1.

S2 Meeting with the customer regularly reduces the risk to some extent.

S3 R3 properly develops the system to incorporate modifications at a later stage and retains all necessary paperwork to minimize the risk, as previously stated.

Chapter 3: System Analysis and Design

3.1 Software Requirement Specification:

3.1.1 Product Perspective

Smart car parking systems help the user in order to find whether there's an empty parking slot or not in the particular area with much ease than the other existing system. The main aim of this system is to provide a safe and secure experience to the user .It also helps in management of the parking lot in an efficient manner.

3.1.2 Product Functions

The smart car parking system basically works on the principle of detecting the obstacles and segregates the parking slots into two different identifiers i.e vacant and occupied, where available slots are marked with green outline and the already engaged slots are marked with the red outline.

3.1.3 User Characteristics

3.1.3.1 Large Organizations

Once the system is developed to its highest capacity, it can be used anywhere irrespective of the size of the parking space, let it be a small mart or a multi storey parking system like a mall or an office.

3.1.3.2 Academic Organizations

In an academic organization, both students and staff members look for a parking space where they can park their vehicles with ease as well as they can be secure, our system provides the requirements which they want in an accurate and exact manner.

3.1.4 Design and implementation constraints

3.1.5 Assumption and dependencies

- There should be an Internet link that is secure and fast.
- The device's operating system should be compatible with the requirements that means it should not be outdated.

3.1.6 Requirement specification

3.1.6.1 Hardware Interfaces

A good quality camera is needed to detect the status of each parking space which is freely occupied correctly and accurately.

3.1.6.2 Software Interfaces

1. Anaconda Individual Edition
2. Python IDE (Jupyter Notebook, Spyder etc)
3. ML and DL libraries

3.1.7 System features or functional requirements

3.1.7.1 Introduction

This system aims to solve these problems by designing a system to control the parking area using a machine learning-based system. This system is designed so that it results in faster, flexible, and can meet market needs. And it can also fulfil the total needs for valuable parking which can hold the vehicle very carefully and also with high security.

3.1.7.2 Input

Video: We have collected various videos from different real-time sources in which we have applied an algorithm in order to identify or differentiate between the vacant and occupied parking slots/regions.

3.1.7.3 Processing

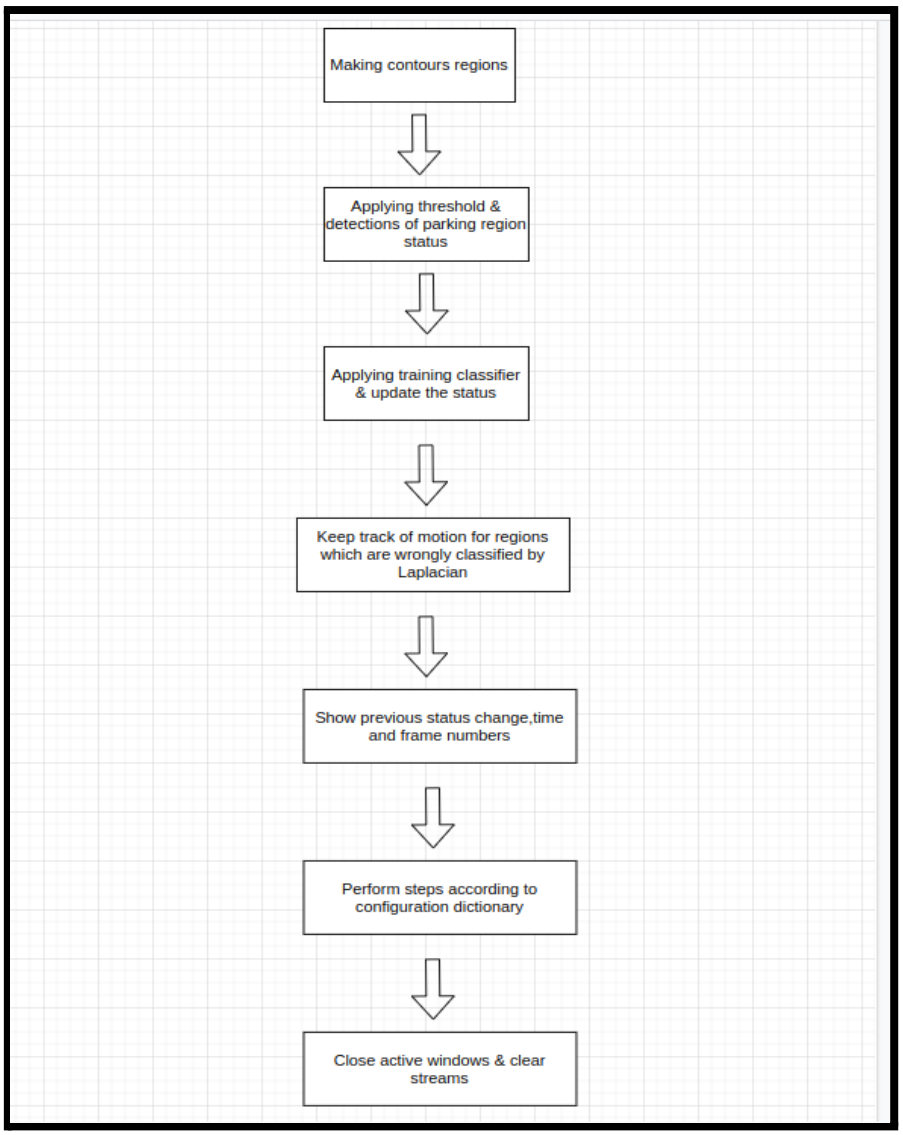

Fig iv

3.1.7.4 Normal Requirements

N1: System must be able to identify whether an object is a vehicle or not.

N2: System must be able to differentiate between empty and occupied space.

N3: System must be able to provide a clear image of an empty parking slot.

3.1.7.5 Non-Functional Requirements

These are the specifications, as the name implies, that are not specifically correlated with particular functions offered by the device.

3.1.7.5.1 Availability:

In terms of availability, the system runs almost 24*7. Customers can get the service they need at any time of the day.

3.1.7.5.2 Usability:

In terms of usability, certain systems will be of superior quality. It is permissible for customers to perform tasks through easy-to-use manual guides. However, this particular system is of excellent quality and user friendly.

3.1.7.5.3 Scalability:

Scalability is also considered as an most important non-functional requirement. This helps the system develop individual capabilities by addressing the needs and requirements of business operations processes. A proper and effective scalability management process helps ensure efficient customer service.

3.1.7.5.4 Performance:

The proposed system will be effective, efficient and versatile on the implementation part.

3.1.7.5.5 Security:

From the security point of view, the developed system will not support any peripheral devices as it creates a chance for the devices to be hacked which is not at all user friendly.

3.2 FLOWCHARTS:

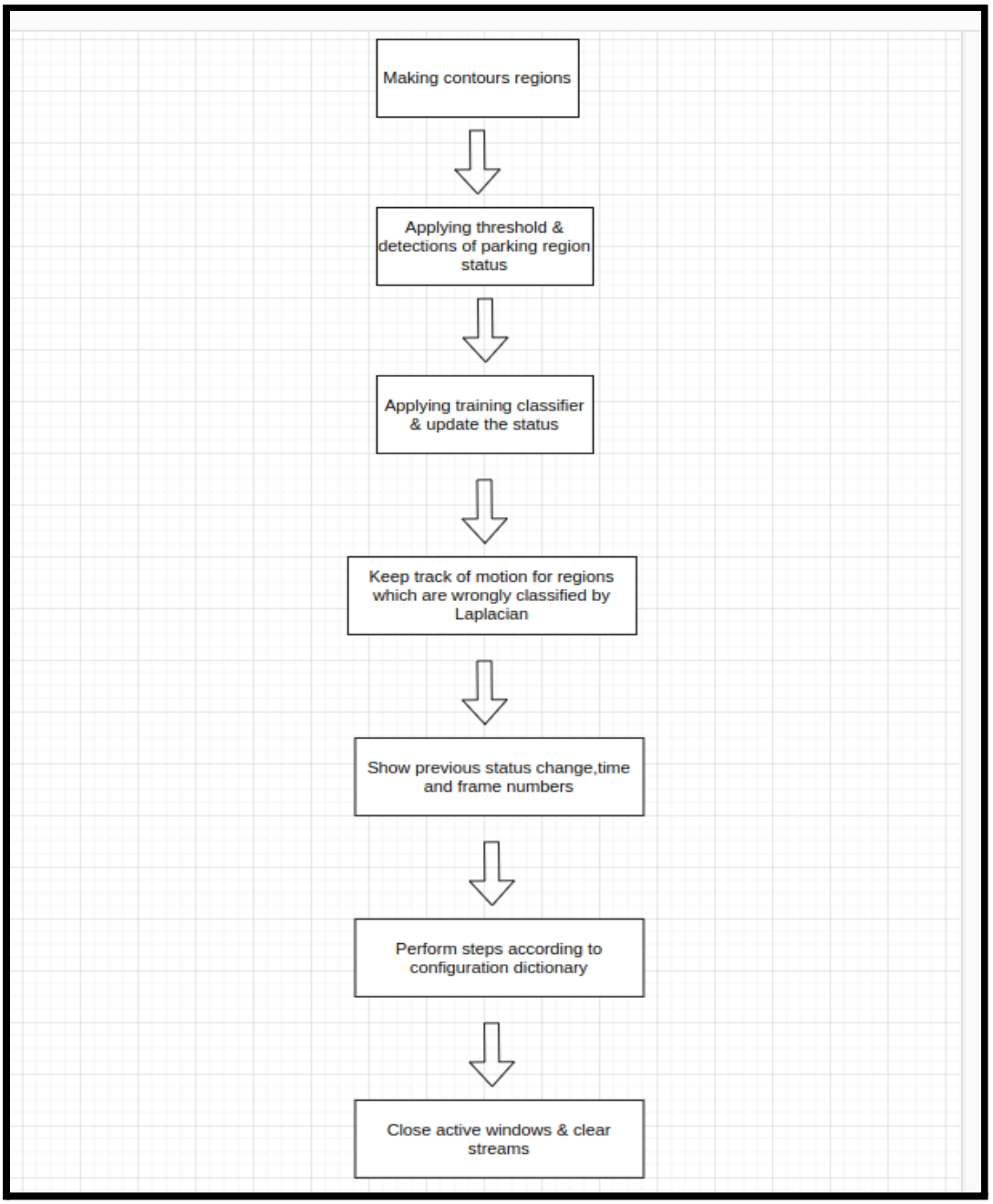

Fig v

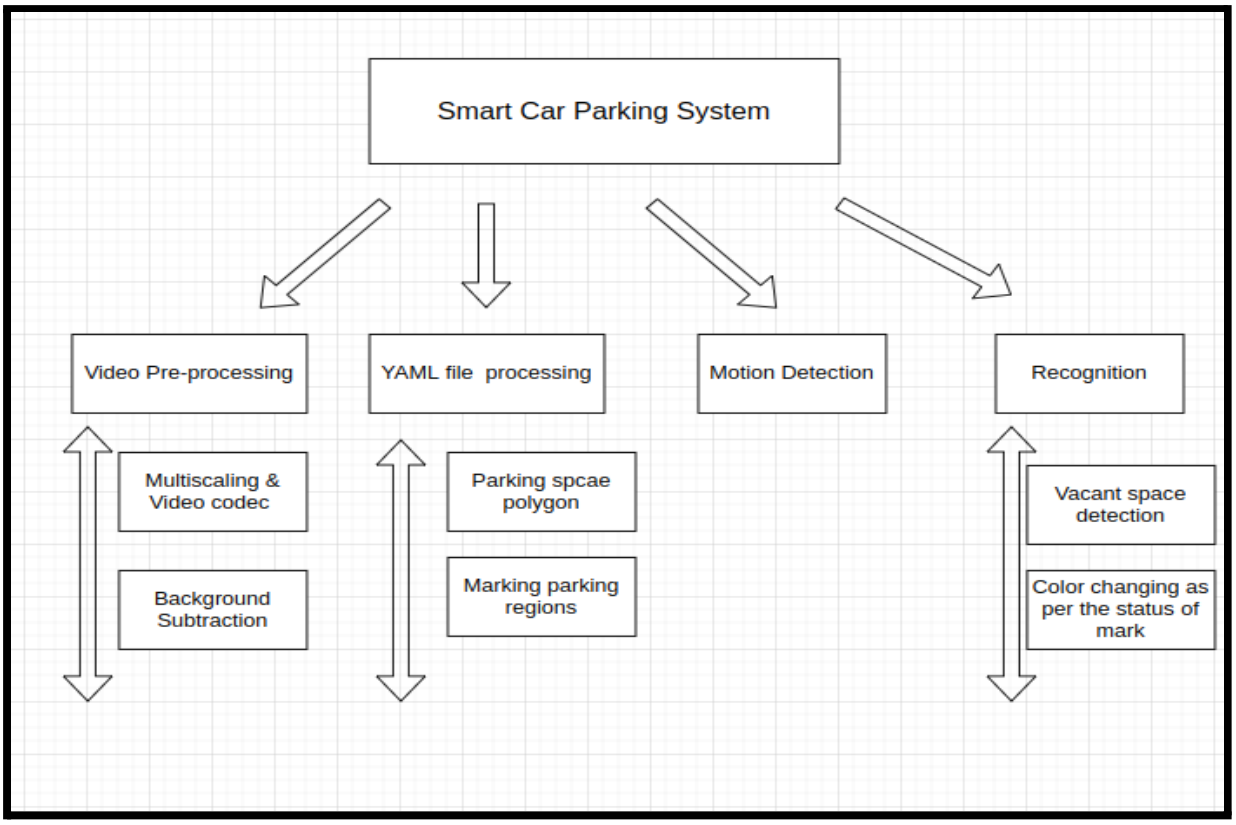

Fig vi

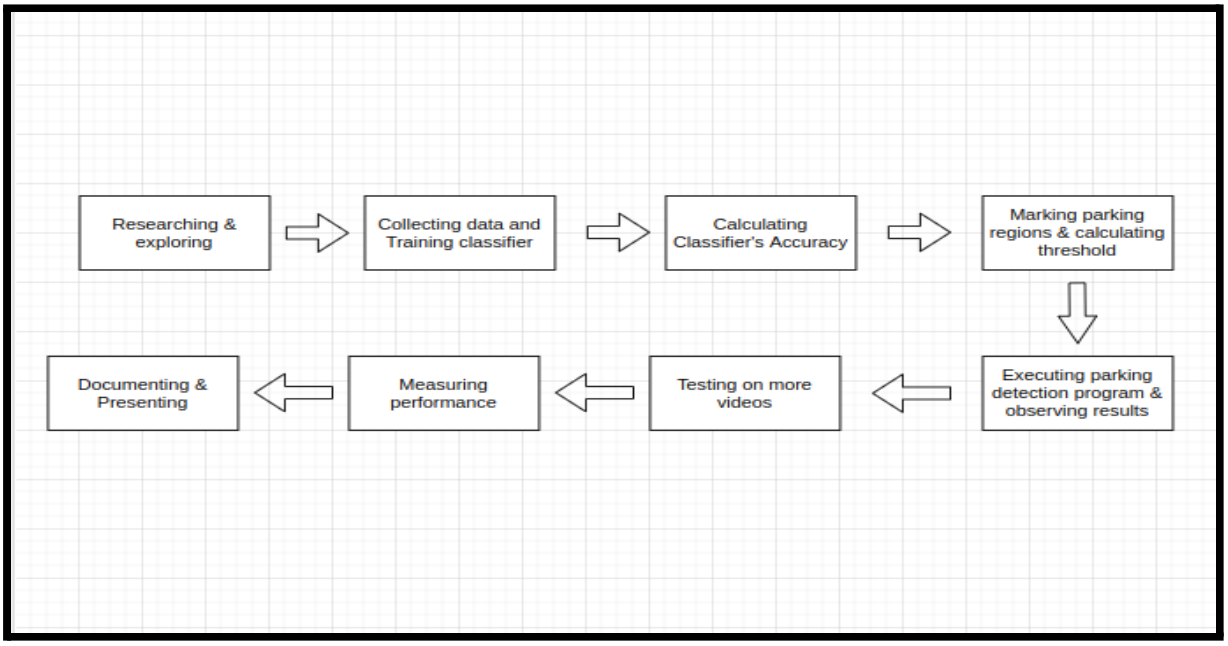

Fig vii

3.3 Design and Test Steps/Criteria

3.3.1 Process Model

The Process Model groups processes into models of the same type. As a result, a model is used to define a mechanism on a type-by-type basis. Even though the paradigm has progressed to the type stage, it is still in the process of being realised. The same method model is frequently used to construct several iterations and has a number of different instantiations. To define how tasks can be carried out in relation to the current situation, a system model should be employed.

The objective of a model is as follows:

- **Descriptive:**

  1. Keep track of everything that happens during the operation.

  2. Consider the viewpoint of an outside expert who assesses how an activity is carried out and determines whether changes are needed to improve its success or reliability.

- **Prescriptive:**

  1. A description of the procedures required, including how they will be carried out.

  2. Establish regulations, processes, and patterns of activity that, if followed, will contribute to the process's desired performance. It might range from stringent obedience to lose guidance.

- **Explanatory:**

  1. Describe the reasons for such procedures in detail.

  2. Analyze and analyse numerous prospective routes of action using logical reasoning.

  3. Establish a solid link between the methods and the standards that the model must achieve.

  4. Prior to the locations where tracking data can be obtained.

3.3.1.2 Waterfall Model

The traditional waterfall model divides the life cycle into several stages.This paradigm assumes that a new phase can be initiated after the previous one has been completed.That is, The output of one step is used as the input of the next step.
As a result, the development process can be considered as a sequential flow.

In the proposed approach since we can only move forward after completing the earlier step and since the project is a bit shorter, we have chosen a waterfall model for our project.

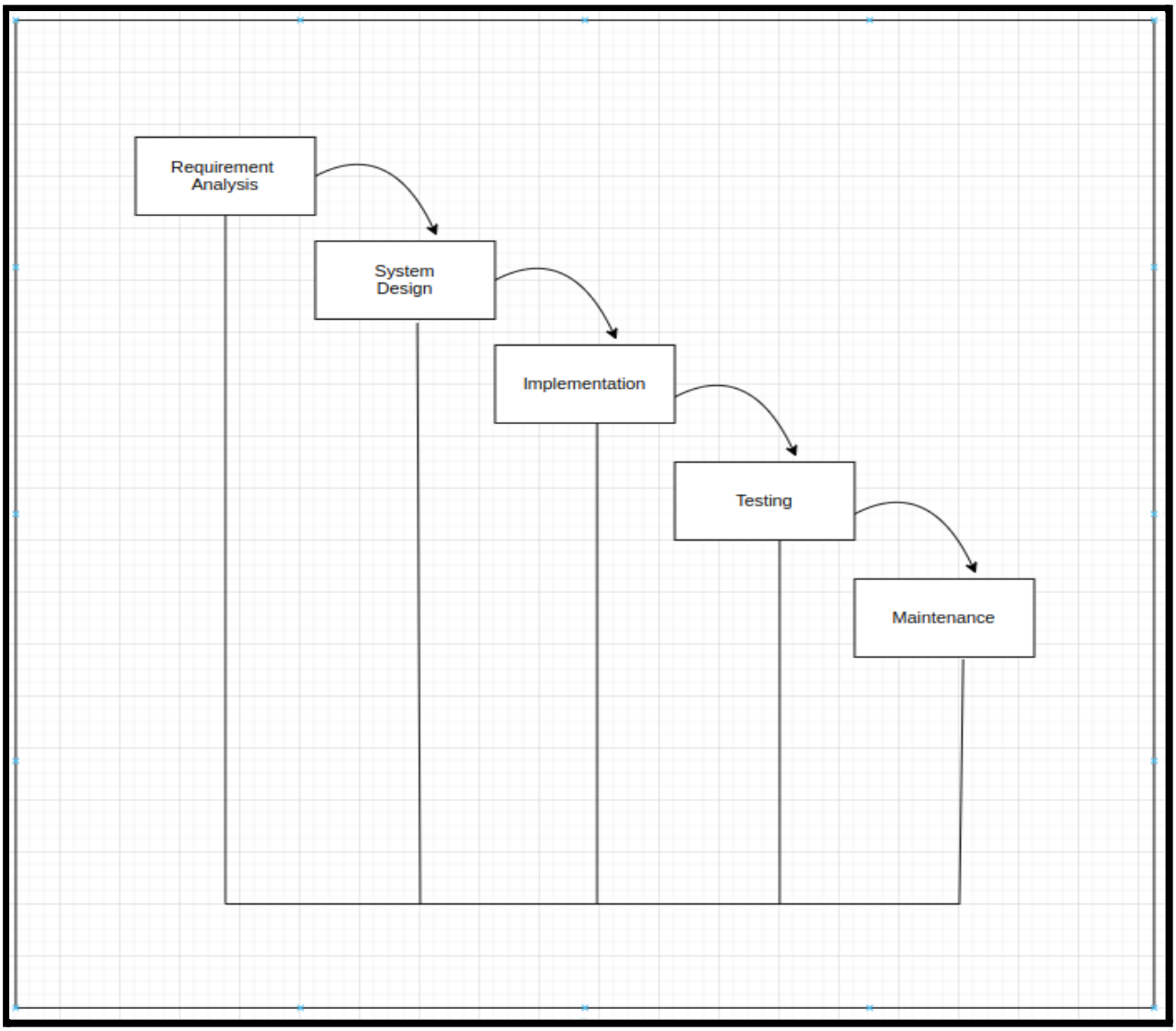

Fig viii

- **Requirement Gathering and analysis** −

  At this stage, all possible needs for the system being created are recorded and they are documented in a requirements specification document.

- **System Design** −

  At this stage, the requirement specifications of the previous stage are examined and the system design is prepared. This system blueprint helps you develop the overall system architecture and describe the hardware and system requirements.

- **Implementation** −

  The system is first built as discrete programmes called units, which are then merged in the next phase, using inputs from the system design. Unit testing is the process of developing and testing each unit for its functioning.

- **Integration and Testing** −

  After each module has been tested, all modules created during the implementation phase are combined into a system.The entire system is then tested for any flaws or failures after it has been integrated.

- **Maintenance** −

  There are some issues that appear in the client environment. Patches have been released to fix these issues. Also, some better versions will be released to improve the product. Maintenance is performed to deploy these changes to your environment.

Advantages of Waterfall Model:

- Ease of control due to model tension. Each section has specific deliverables and an evaluation process.
- Phases are processed and finished one at a time.
- Works properly for smaller initiatives wherein necessities are thoroughly understood. Clearly described stages.

3.3.2 Breakdown Structure (Modules_Analysis)

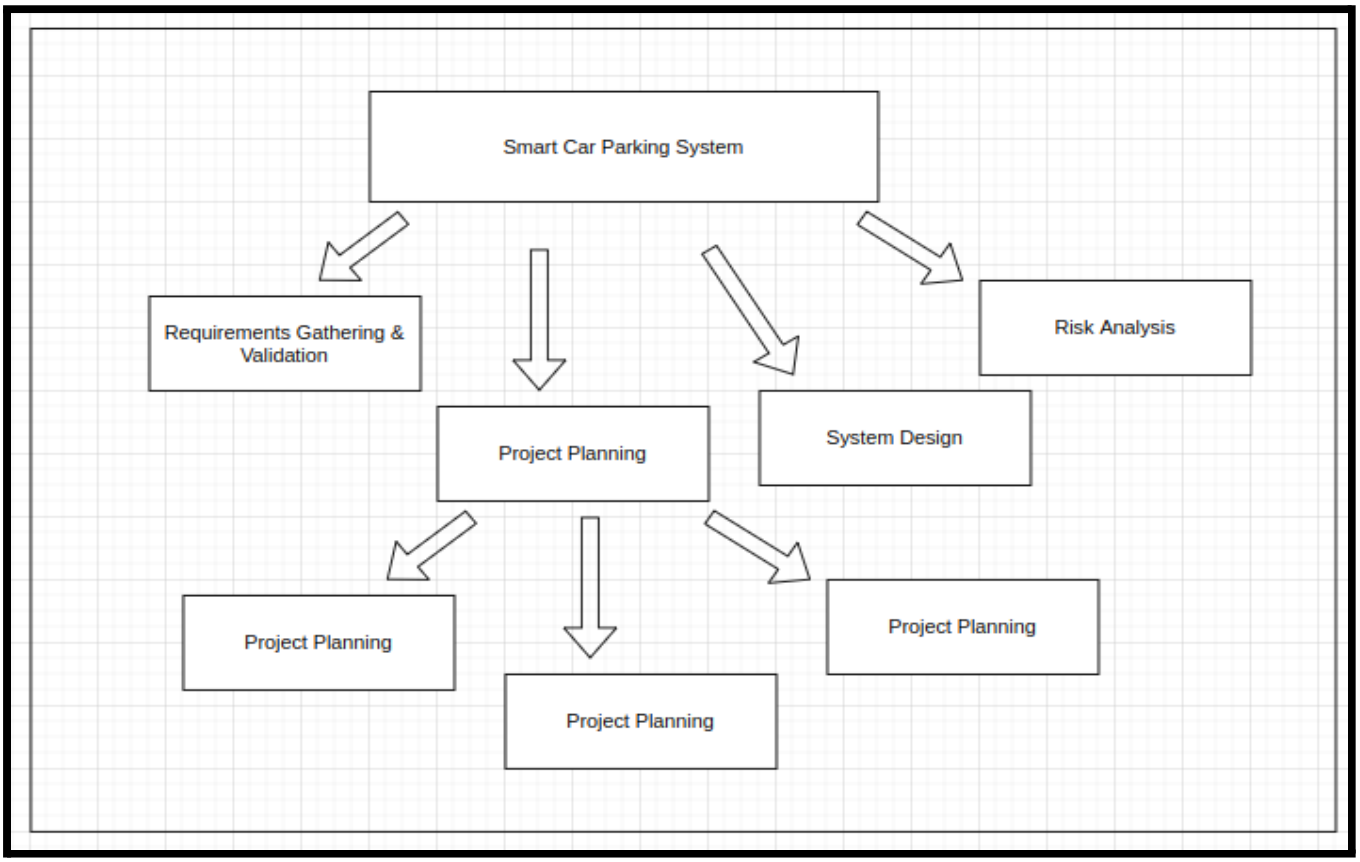

Fig ix

- **Communication:**

The product creation phase begins with a user-developer dialogue. We also collected project-related specifications according to work requirements.

- **System Design:**

The process model used to implement the system. This activity also determines the working exploded view (module). The breakdown tree shows the various components used by the framework.

- **Project Planning:**

Includes complete calculation and timing of the entire timeline diagram for project development and monitoring. Tasks are often expected to identify tools, timelines, and other details related to the project.

- **Modeling (Analysis & Design):**

Includes a thorough review of specifications and project planning. In needs analysis, the system is analyzed according to the customer's request, and the direction in which the system starts, the direction in which it moves, and the target are provided in the analysis process. In architecture, device design follows research.

3.3.2 Breakdown Structure (Modules_Implementation)

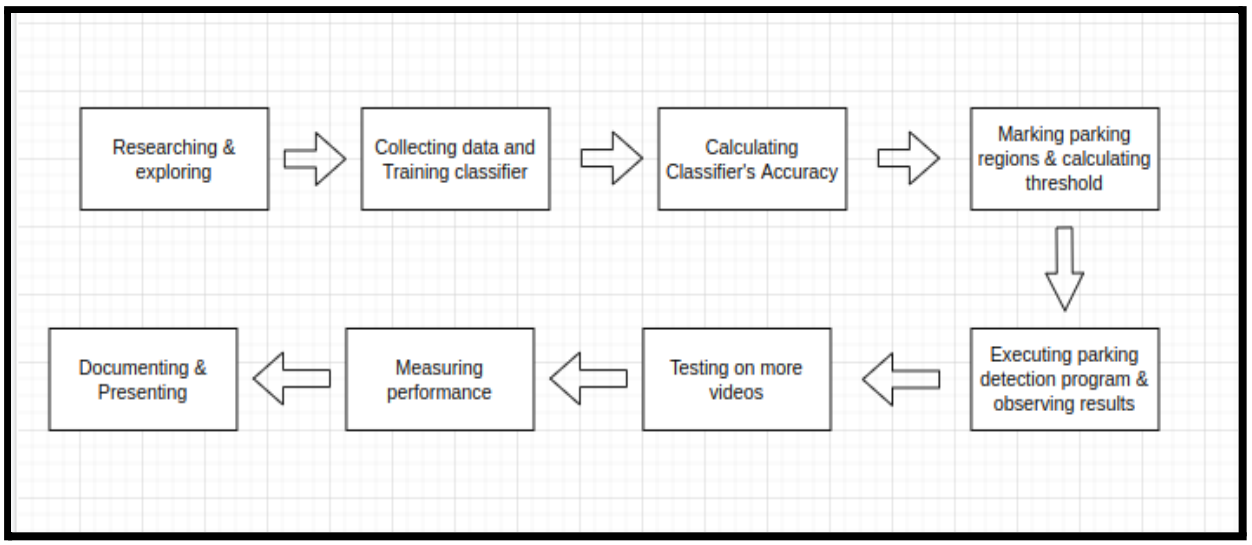

Fig x

- **Research and exploring**

  Firstly, we did research on the topic of smart car parking systems. Research and exploring are important as it focuses on the issues that help improve the project's effectiveness.

- **Collecting dataset and training classifier**

  For the dataset, we collect data on cars and parking spaces.

Neural networks and other artificial intelligence programs require a first dataset, the so-called training dataset.It serves as a basis for further application and use.
This dataset forms the basis of the program's growing information library. The training database must be accurately labeled before processing and training the model training set.To create an ML algorithm, you need both training and test information. When a model is taught in a training set, it is usually evaluated using a test set.Often these sets are taken from the same overall dataset, but training sets need to be marked or enhanced to increase the reliability and accuracy of the algorithm.

- **Marking Parking regions and calculating Threshold**

  For moving forward, we'll be needed with the calculation of threshold, which can only be possible by changing the images into the grayscale with the help of cvtcolor() function.The calculation of threshold will compare the pixel values with the threshold values.
  If the pixel value is less than the threshold, it is set to 0, otherwise it is set to the maximum value usually 255. Threshold is a very popular segmentation technique used to separate objects that are regarded as foreground from their background.

- **Executing parking detection program and observing results**

  The parking detection algorithm involves the steps starting from performing different thresholding techniques like binary and adaptive in order to perform the separation from the background that’ll further be followed by cascading the classifier that basically means training the system with positives and the negative images of the same size and as soon as the classifier gets trained, it can further be applied to a region in order to detect the object there.
  Further, the algorithm will be accompanied by the execution of the data augmentation process.
  Data augmentation is a strategy designed to increase the variety of data that can be used to train a model without actually collecting new data.This process modifies the training images to create a synthetic dataset that is larger than the original dataset.

- **Testing on more video**

  After that we are going to test our approach on more videos to get more appropriate/accurate results.

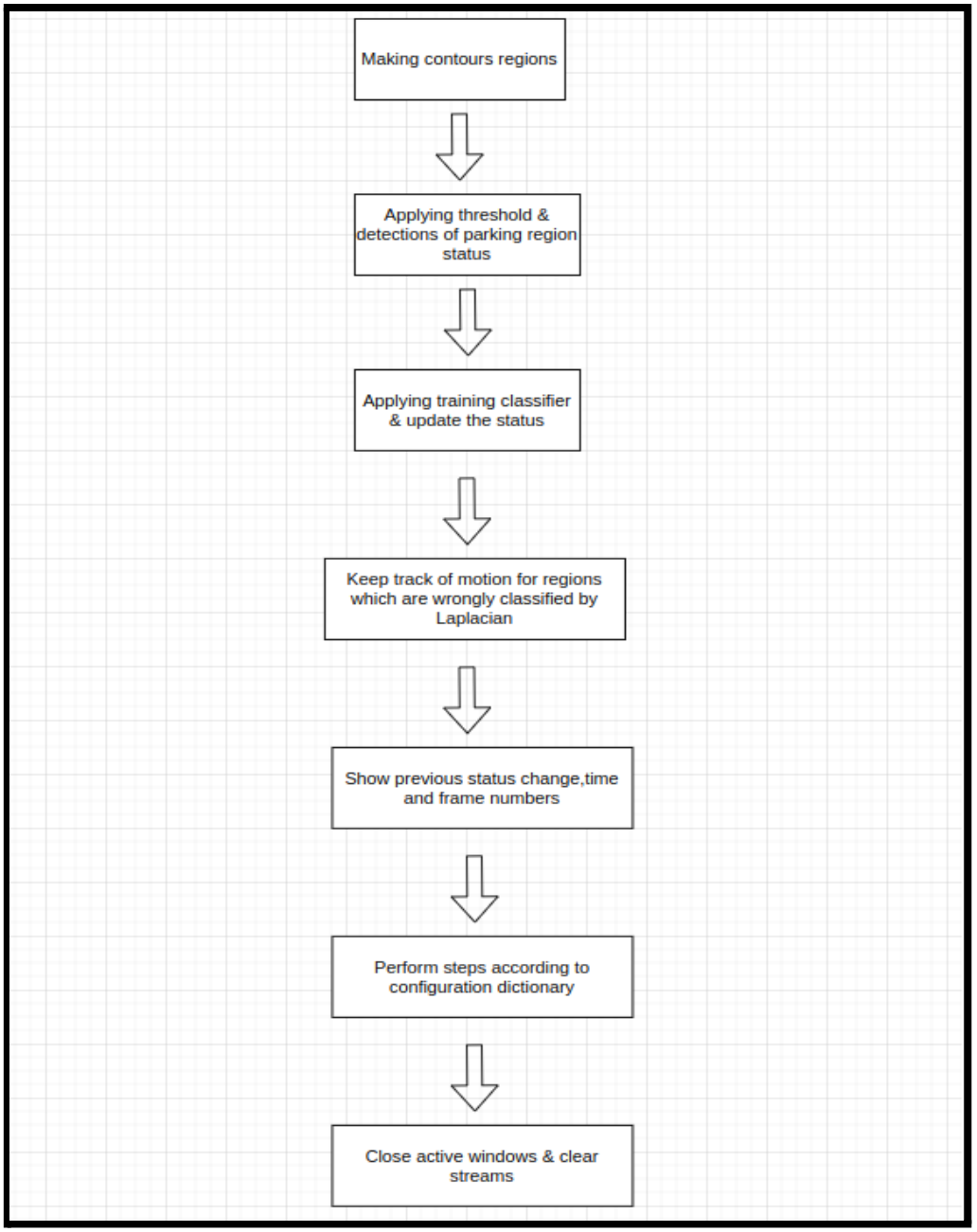
Making contours regions
Applying threshold & detections of parking region status
Applying training classifier & update the status
Keep track of motion for regions which are wrongly classified by Laplacian
Show previous status change,time and frame numbers
Perform steps according to configuration dictionary
Close active windows & clear streams

Fig xi

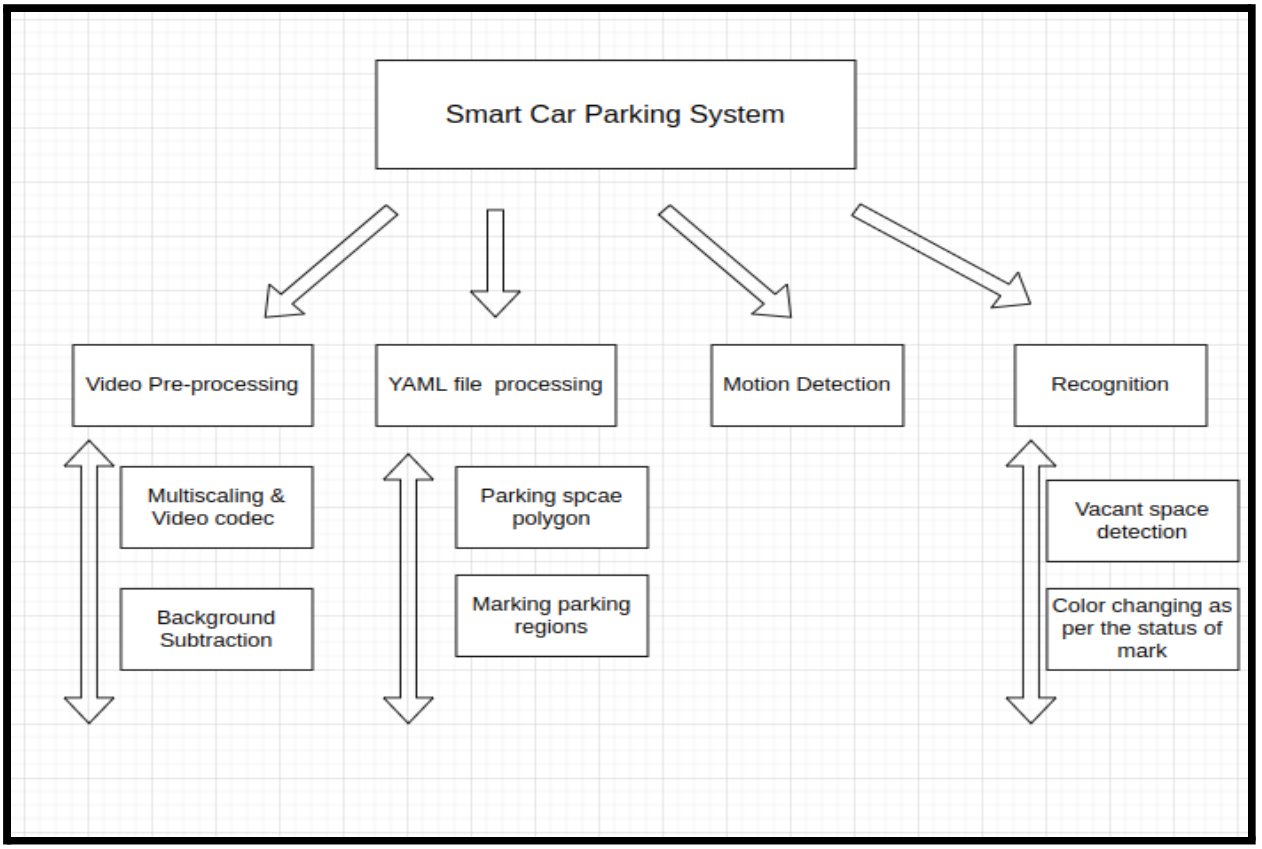

Fig xii

- **VIDEO PRE-PROCESSING**

☐ **MultiScaling & Video Codec**

The algorithm begins with the capturing of multiple videos, followed by starting the progress first of all by training the video for the process as well. The process begins with the path referencing for the video, assigning initial values to some of the variables that will be used in further process.

```
# path references
fn = "testvideo_01.mp4" #3
#fn = "datasets\parkinglot_1_720p.mp4"
#fn = "datasets\street_high_360p.mp4"
fn_yaml = "test_yml_01.yml"
fn_out =  "outputvideo_01.avi"
cascade_src = 'classifier_02.xml'
car_cascade = cv2.CascadeClassifier(cascade_src)
global_str = "Last change at: "
change_pos = 0.00
dict =  {
        'text_overlay': True,
        'parking_overlay': True,
        'parking_id_overlay': True,
        'parking_detection': True,
        'motion_detection': True,
        'pedestrian_detection': False, # takes a lot of processing power
        'min_area_motion_contour': 500, # area given to detect motion
        'park_laplacian_th': 2.8,
        'park_sec_to_wait': 1, # 4   wait time for changing the status of a region
        'start_frame': 0, # begin frame from specific frame number
        'show_ids': True, # shows id on each region
        'classifier_used': True,
        'save_video': False
        }
```

Fig xiii

After creating the dictionary for further use, we will simply set data from video in order to make things easier and feasible. Now since the classifier has been trained for the car the cascade will be used in the rest of the process. The multi scale method will be used in order to detect the different sized objects that will help in deciding the parking status as well. The process will further include the process of coding and decoding the video as well.

Basically, opencv helps us to get the video which we have operated upon in order to be used in the future as well and so it has provided a video writer object function for the same that consists of a code as well of four bytes in order to specify the video codec. There are many codecs as per our systems for the videos, the function also works upon the frames rate as well as the size of them.

Video codecs are basically the formats for the compressed files, also the isColor flag as the last parameter specifies if the encoder will further work with the colored frame or the grayscale will work.

```
        }

# Set from video
cap = cv2.VideoCapture(fn)
video_info = {  'fps':    cap.get(cv2.CAP_PROP_FPS),
                'width':  int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)*0.6),
                'height': int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)*0.6),
                'fourcc': cap.get(cv2.CAP_PROP_FOURCC),
                'num_of_frames': int(cap.get(cv2.CAP_PROP_FRAME_COUNT))}

cap.set(cv2.CAP_PROP_POS_FRAMES, dict['start_frame']) # jump to frame number specified

def run_classifier(img, id):
    # gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(img, 1.1, 1)
    if cars == ():
        return False
    else:
        # parking_status[id] = False
        return True

# Define the codec and create VideoWriter object
if dict['save_video']:
    fourcc = cv2.VideoWriter_fourcc('X','V','I','D') # options: ('P','I','M','1'), ('D','I','V','X'),
('M','J','P','G'), ('X','V','I','D')
    out = cv2.VideoWriter(fn_out, -1, 25.0,(video_info['width'], video_info['height']))
```

Fig xiv

☐ **Background Subtraction & Feature Extraction**

The process of background subtraction has been executed with the help of background segmentation method, the one used in this approach is the BackgroundSubtractorMOG2, which is a background segmentation algorithm based on Gaussian mixtures. This technique assigns a Gaussian Distribution to each of the backdrop pixels. The length of time the colours remain in the scene determines the distribution's weight. Essentially, the algorithm uses the Gaussian mixture information to identify the background. The theory is that the longer a colour stays in the backdrop, the more likely it is to be a part of it. This approach can adapt to variations in illumination thanks to the gaussian distribution. This class allows for parallel processing.

The parameters being used are:

history- Length of history,

varThreshold- It tells us if pixel is well describe by bg model or not,

detectShadows- It tells us if shadows are given importance in the scene.

It begins building a background model when applied to the first image submitted to this object. As frames are supplied into this object, the backdrop is updated.The varThreshold

parameter helps determine whether a pixel is considered a background pixel. This value does not affect background model updates. This number represents the distance between the pixel and the background model.The higher the threshold, the higher the level of assurance required.We use it to indicate that we only want pixels that the backdrop model accurately describes

Dilation: This operation consists in convolving the image A with the kernel (B). The core (B) can be of any shape and size (usually a square or a circle). Kernel B defines an anchor point, which is usually the center of the kernel.

When kernel B scans the entire image, B calculates the maximum value of the overlapping pixel and replaces the image pixel at the anchor point with that maximum value.As you can guess, this maximization operation "grows" the bright areas in the image (and thus the name grows).

Erosion: This operation is a sister of the extension. Calculates a local minimum within the specified kernel range. When kernel B scans the entire image, B calculates the minimum overlapping pixel value and replaces image pixels below the anchor point with that minimum value.

- ☐ **HAAR Classification**

For detecting the presence of a vehicle we will need to train our classifier for detection of absence or presence of vehicle in the parking slot. For this purpose, we use cascading Haar classifiers, which are an efficient and effective way to detect objects. It is a machine learning based algorithm that uses positive and negative images to train a classifier.

Positive Images:These images show that the car is in a parking space, which means that the parking space is already taken.

Negative Images:These images show that the car is not in the parking slot and is yet not occupied by any vehicle.

OpenCV library is being used for implementing HAAR classifier training.The HAAR features are represented by black and white boxes, the features are basically individual values that are obtained by calculating the delta, which we can find by subtracting the sum of the pixels under the white boxes from the sum of the pixels under the black boxes.

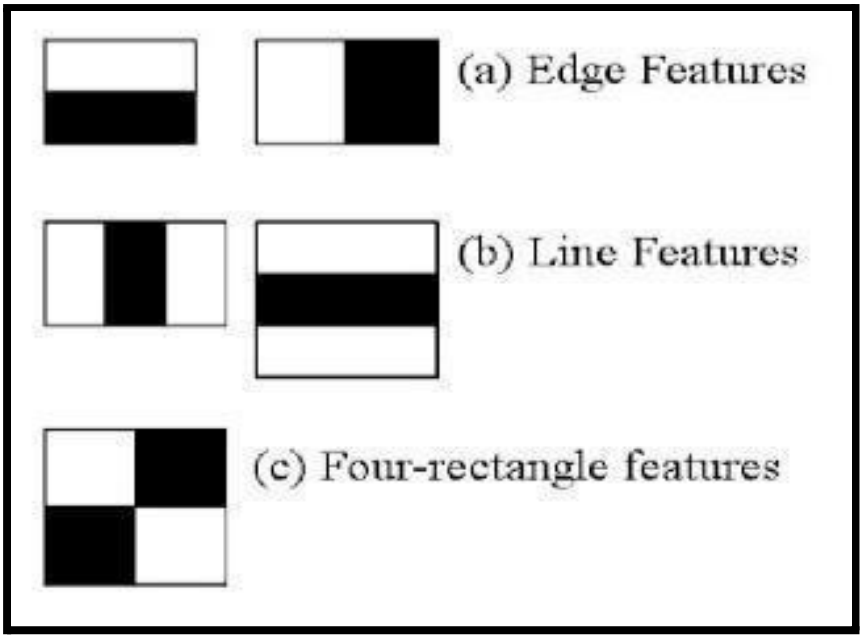

Fig xv

Advantages of HAAR classifier:

1) Classifiers do not need a large amount of dataset to be trained,the work can be done in less amount of images as compared to other classifiers which leads to less load on the software and less memory utilization.
2) It also has higher execution speed even if the number of pixels are relatively large and also gives more accurate results.

- **YAML File Processing**

  - ☐ **Parking space polygon**

In this phase of the parking space polygon, the process involves initialization of masks, motion,contours and other points which will further be used while creating the polygons for the parking. The process involves the morphological kernel in order to mark the regions whose coordinates will be updated in the yaml file which will further be used when we will be executing the recognition process later.

```
with open(fn_yaml, 'r') as stream:
    parking_data = yaml.safe_load(stream)
parking_contours = []
parking_bounding_rects = []
parking_mask = []
parking_data_motion = []
if parking_data != None:
    for park in parking_data:
        points = np.array(park['points'])
        rect = cv2.boundingRect(points)
        points_shifted = points.copy()
        points_shifted[:,0] = points[:,0] - rect[0] # shift contour to region of interest
        points_shifted[:,1] = points[:,1] - rect[1]
        parking_contours.append(points)
        parking_bounding_rects.append(rect)
        mask = cv2.drawContours(np.zeros((rect[3], rect[2]), dtype=np.uint8), [points_shifted], contourIdx=-1,
                                color=255, thickness=-1, lineType=cv2.LINE_8)
        mask = mask==255
        parking_mask.append(mask)

kernel_erode = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3)) # morphological kernel
kernel_dilate = cv2.getStructuringElement(cv2.MORPH_RECT,(5,19))
if parking_data != None:
    parking_status = [False]*len(parking_data)
    parking_buffer = [None]*len(parking_data)
# bw = ()
```

Fig xvi

The process involves the morphological ellipse and rectangle function respective to the kernel erosion and dilation. The morphological operation is basically a series of operations for transforming images into forms. Morphological operations generate an output image by applying a structuring element to an input image.

The following are the most fundamental morphological operations: Erosion and dilation are two terms that are often used interchangeably. They can be used for a variety of things, such as:

Noise reduction.

Individual elements are isolated and dissimilar elements in an image are joined.

Detection of image intensity bumps or holes.

- ☐ **Marking Parking Regions**

Marking parking regions is basically the next step to the parking space polygon procedure, it involves numbering the marked regions by putting text and further detecting the position of the video at the current time and updating the same. This process involves another concept of open cv that is of moments. Moments is basically the one that provides the distribution of something round an axis.

The first argument to the procedure cv2.threshold() is the image to which thresholding must be applied. The thresholdValue is the second argument, while the maxVal is the third. We'll use binary thresholding in this case.The last parameter thresholdingTechnique is set to 0 to achieve this.

```
# bw = ()
def print_parkIDs(park, coor_points, frame_rev):
    moments = cv2.moments(coor_points)
    centroid = (int(moments['m10']/moments['m00'])-3, int(moments['m01']/moments['m00'])+3)
    # putting numbers on marked regions
    cv2.putText(frame_rev, str(park['id']), (centroid[0]+1, centroid[1]+1), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,25
5,255), 1, cv2.LINE_AA)
    cv2.putText(frame_rev, str(park['id']), (centroid[0]-1, centroid[1]-1), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,25
5,255), 1, cv2.LINE_AA)
    cv2.putText(frame_rev, str(park['id']), (centroid[0]+1, centroid[1]-1), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,25
5,255), 1, cv2.LINE_AA)
    cv2.putText(frame_rev, str(park['id']), (centroid[0]-1, centroid[1]+1), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,25
5,255), 1, cv2.LINE_AA)
    cv2.putText(frame_rev, str(park['id']), centroid, cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

while(cap.isOpened()):
    video_cur_pos = cap.get(cv2.CAP_PROP_POS_MSEC) / 1000.0 # Current position of the video file in seconds
    video_cur_frame = cap.get(cv2.CAP_PROP_POS_FRAMES) # Index of the frame to be decoded/captured next
    ret, frame_initial = cap.read()
    if ret == True:
        frame = cv2.resize(frame_initial, None, fx=0.6, fy=0.6)
    if ret == False:
        print("Video ended")
        break
```

Fig xvii

The maximum value that can be assigned to a pixel in thresholding is equal to maxVal. Because we're using binary thresholding, we'll give a pixel the maximum value if it's higher than the threshold. If the pixel value is less than the threshold, the value is set to 0.It should return a tuple of two values. The first value is the threshold and the second tuple value returned is the thresholded image. Pass the cnt value as an argument to the cv2.moments () function to output the moment and print every moment.

- ☐ **Background Subtraction**

After capturing the parking regions, the frames created will be moved towards the process of background subtraction, the initial phase of which is the gaussian blur, which further be followed by the cvtColor() function in which the blurred frame and the color will also be included.

The blurring of an image simply means smoothening it and removing the extra noises . The blurring function involves following parameters:

src: source

ksize:Kernel is a matrix of degree (rows) * (columns). Its size is specified in the form of tuples (number of rows, number of columns). number. The number of rows and columns must be odd. If ksize is specified as (0 0), ksize is calculated from the specified sigma value. H. sigmaX and sigmaY.

sigmaX: standard deviation along horizontal one.

sigmaY: standard deviation along vertical one.

borderType: boundary of the image.

Fig xviii

Fig xix

- **MOTION DETECTION**

The motion detection phase includes majorly the erode and dilation kernels followed by finding the contours, also the contour area will be checked if lesser than the specified minimum contour area.The process of background subtraction has been executed with the help of background segmentation method, this approach uses BackgroundSubtractorMOG2, which is a background segmentation algorithm based on Gaussian mixtures. This method assigns a Gaussian distribution to each background pixel. The length of time the colours remain in the scene determines the distribution's weight.

Dilation: This operation consists in convolving the image A with the kernel (B). The core (B) can be of any shape and size (usually a square or a circle). Kernel B defines an anchor point, which is usually the center of the kernel.

When kernel B scans the entire image, B calculates the maximum value of the overlapping pixel and replaces the image pixel at the anchor point with that maximum value. As you can guess, this maximization operation "grows" the bright areas in the image (and thus the name grows).

Erosion: This operation is a sister of the extension. Calculates a local minimum within the specified kernel range. When kernel B scans the entire image, B calculates the minimum overlapping pixel value and replaces image pixels below the anchor point with that minimum value.

The motion detection algorithm is further followed by the contours phase, Contour lines connect all continuous points (along the boundaries) and can easily be described as curves of the same color or intensity. Contours are a very useful tool for object detection and recognition and shape analysis .

Use binary images for better accuracy. So apply thresholding or tricky edge detection before finding the edge. As of OpenCV 3.2, findContours() no longer modifies the original image, but returns the modified image as the first image with three return parameters. In OpenCV, finding a contour is like finding a white object on a black background. So keep in mind that the object you want to detect must be white and the background black.

- **VACANT SPACE DETECTION**

The detection of cars and the vacant spaces includes the major operation of the Laplacian operator. The Laplacian measures what is called the "curvature" or stress of a field. This shows how much the value of the field deviates from the average taken at the surrounding points. This is because it is a divergence of the gradient. This shows how the rate of change of the field differs from the smooth change expected in a divergent flow.

Look at one dimension. The Laplacian is simply ∂2∂x2, which is the curvature. If this is zero, the function is linear, so the value at the center of any interval is the average of the extrema. If the Laplace operator is zero in three dimensions, the function is a harmonic function and satisfies the principle of averaging.

- **CHANGING COLOR OF FRAME**

```
# changing the color on the basis on status change occured in the above section and putting numbers on areas
if dict['parking_overlay']:
    for ind, park in enumerate(parking_data):
        points = np.array(park['points'])
        if parking_status[ind]:
            color = (0,255,0)
            rect = parking_bounding_rects[ind]
            roi_gray_ov = frame_gray[rect[1]:(rect[1] + rect[3]),
                          rect[0]:(rect[0] + rect[2])]  # crop roi for faster calcluation
            res = run_classifier(roi_gray_ov, ind)
            if res:
                parking_data_motion.append(parking_data[ind])
                # del parking_data[ind]
                color = (0,0,255)
        else:
            color = (0,0,255)

        cv2.drawContours(frame_out, [points], contourIdx=-1,
                         color=color, thickness=2, lineType=cv2.LINE_8)
        if dict['show_ids']:
                print_parkIDs(park, points, frame_out)
```

Fig xx

After applying the above mentioned processes, the next step is to set the color of the frames as per the status of the vehicle being detected, for eg if the parking space is occupied the frame will be reflected as of red color, for the vacant one it will be green and further for the moving ones the frame will be of blue color.

```
if parking_data_motion != []:
    for index, park_coord in enumerate(parking_data_motion):
        points = np.array(park_coord['points'])
        color = (0, 0, 255)
        recta = parking_bounding_rects[ind]
        roi_gray1 = frame_gray[recta[1]:(recta[1] + recta[3]),
                               recta[0]:(recta[0] + recta[2])]  # crop roi for faster calcluation
        # laplacian = cv2.Laplacian(roi_gray, cv2.CV_64F)
        # delta2 = np.mean(np.abs(laplacian * parking_mask[ind]))
        # state = delta2<1
        # classifier_result = run_classifier(roi_gray1, index)
        # cv2.imshow('dsd', roi_gray1)
        fgbg1 = cv2.createBackgroundSubtractorMOG2(history=300, varThreshold=16, detectShadows=True)
        roi_gray1_blur = cv2.GaussianBlur(roi_gray1.copy(), (5, 5), 3)
        # cv2.imshow('sd', roi_gray1_blur)
        fgmask1 = fgbg1.apply(roi_gray1_blur)
        bw1 = np.uint8(fgmask1 == 255) * 255
        bw1 = cv2.erode(bw1, kernel_erode, iterations=1)
        bw1 = cv2.dilate(bw1, kernel_dilate, iterations=1)
        # cv2.imshow('sd', bw1)
        # cv2.imwrite("frame%d.jpg" % co, bw)
        cnts1, _ = cv2.findContours(bw1.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        # loop over the contours
        for c in cnts1:
            print(cv2.contourArea(c))
            # if the contour is too small, we ignore it
            if cv2.contourArea(c) < 4:
                continue
            (x, y, w, h) = cv2.boundingRect(c)
            classifier_result1 = run_classifier(roi_gray1, index)
            if classifier_result1:
            # print(classifier_result)
                color = (0, 0, 255)  # Red again if car found by classifier
            else:
                color = (0,255, 0)
        classifier_result1 = run_classifier(roi_gray1, index)
        if classifier_result1:
            # print(classifier_result)
            color = (0, 0, 255)  # Red again if car found by classifier
        else:
            color = (0, 255, 0)
        cv2.drawContours(frame_out, [points], contourIdx=-1,
```

Fig xxi

This part of code is to distinguish between the empty and occupied parking slots.If the area which is empty but not enough for the vehicle to be parked then we'll ignore that parking area. If the car is identified by the classifier to be there in the parking slot it will be marked by a red border or else if the parking slot is identified as empty that means no vehicle is parked in that particular region then it will be surrounded or shown to the user by the green border.

- **PEDESTRIAN DETECTION**

```
if dict['pedestrian_detection']:
        (rects, weights) = hog.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=1.05
            )
        # draw the  bounding boxes
        for (x, y, w, h) in rects:
            cv2.rectangle(frame_out, (x, y), (x + w, y + h), (255, 0, 0), 2)

    # write the output frames
    if dict['save_video']:
#         if video_cur_frame % 35 == 0: # take every 30 frames
            out.write(frame_out)
```

Fig xxii

After applying the above mentioned processes, the next step is to detect the people in the image. This part of the code detects whether any person is walking through the parking lane, if so it will slow down the process/program. This process will require high GP/Processor speed as it needs to be quick and is real time to detect people walking in the frame. At last it will write the output frame and save the video for future reference. Also it will take these snapshot every 30 frames so it will check if frame is divisible by 30 and take eerie 30 frames.This will be beneficial for user as well as for admin as user will be able to detect not only availability of parking slots(whether empty or occupied), but can also see if the parking lot is crowded from people or not and also admin can analyze the rush in parking lot.Hence pedestrian detection is an essential component of our system as it makes it even more efficient and user-friendly.

```
            out.write(frame_out)

    # Display video
    cv2.imshow('frame', frame_out)
    # cv2.imshow('background mask', bw)
    k = cv2.waitKey(1)
    if k == ord('q'):
        break
    elif k == ord('c'):
        cv2.imwrite('frame%d.jpg' % video_cur_frame, frame_out)
    elif k == ord('j'):
        cap.set(cv2.CAP_PROP_POS_FRAMES, video_cur_frame+1000) # jump 1000 frames
    elif k == ord('u'):
        cap.set(cv2.CAP_PROP_POS_FRAMES, video_cur_frame + 500)  # jump 500 frames
    if cv2.waitKey(33) == 27:
        break

cv2.waitKey(0)
cap.release()
if dict['save_video']: out.release()
cv2.destroyAllWindows()
```

Fig xxiii

This part of code is for the final outcome or the final video which will be visible to the users on their screens, through which they will be able to see the status of parking lot,which slots are available for parking ,which are vacant and the rush in the parking lot.Ord(c)represents the unicode character. On the basis of this unicode character the frames and the video are being displayed to the users.

3.3 Testing Process:

3.3.1 Software Testing

3.3.1.1 Introduction

Software testing is required in order to validate the efficiency and accuracy of our system. Software testing allows us to test our software under various scenarios and parameters, it gives us a pre-validation check of how accurately our software would behave when exposed to different scenarios. It allows us to improve our software and eradicate bugs if any and helps provide a better user experience.

3.3.2 Unit Testing

Unit testing is mainly implemented on each unit of the software as implemented in source code to verify whether our source code is giving expected results or not. If there is any discrepancy found in the source code it can be corrected before the integration testing.

3.3.3 Integration Testing

Integration testing is the process of building a program structure while running tests to eliminate the errors/bugs.

3.3.4 Validation Testing

Validation tests are carried out to make sure that the system or process we have created satisfies the needs of users or not. This is mainly done in order to test the system from the user's point of view.

Chapter 4: Results/ Outputs

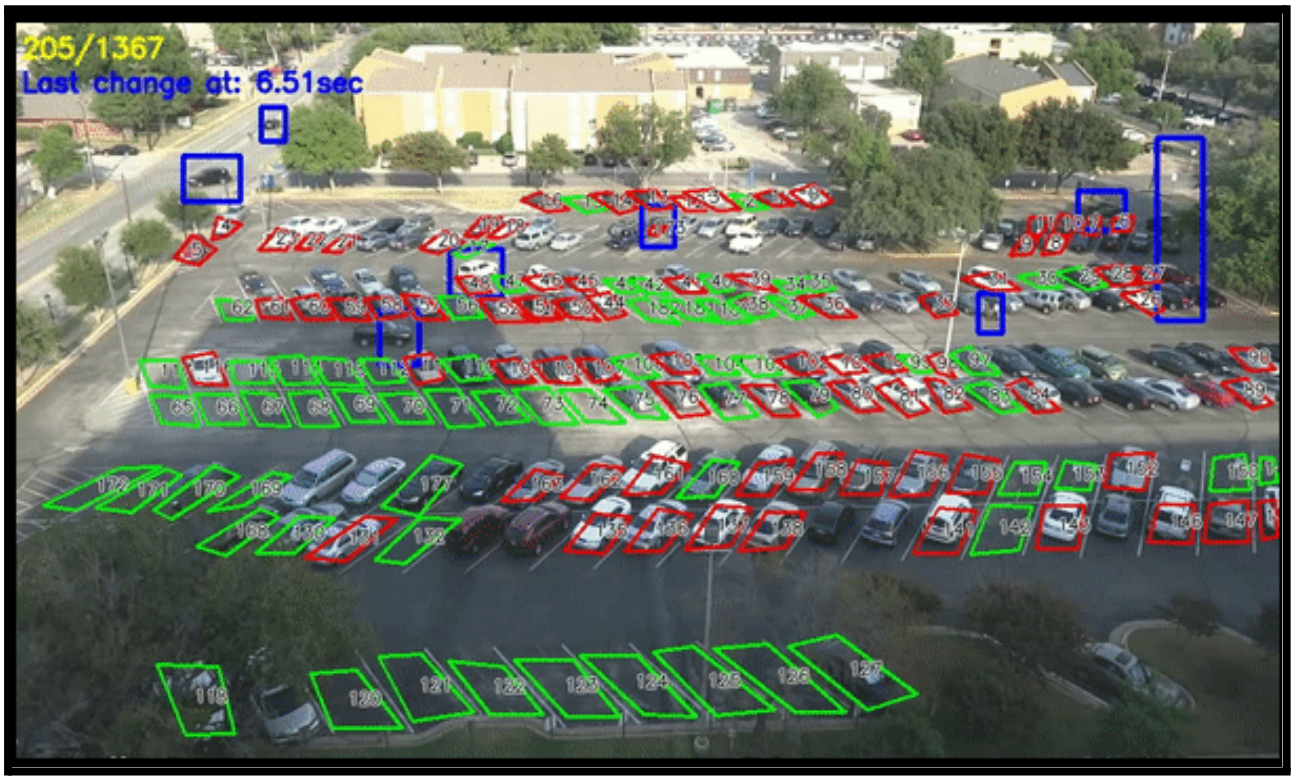

Fig xxiv

Chapter 5: Conclusion

As per the proposed approach, the conclusion for the Smart Car Parking System-

- Existing systems either don't really use any technology, that is simply users have to find the empty spot for parking thereby wasting time and fuel as well, or most of the smart systems being proposed in multiple literature surveys are either using sensors or IOT based approaches.
- Proposed approach focuses on developing an automatic parking system thereby fulfilling the requirements along with working on all the feasibility constraints.
- Proposed approach will start with research study followed by implementation, with the help of image processing and other algorithms.

Chapter 6: References

[1] Prof. Denis Ashok(Ph.D),Akshat Tiwari,Vipul Jirge(2020),Smart Parking System using IoT Technology,2020 International Conference on Emerging Trends in Information Technology and Engineering(ic-ETITE)

[2] Jahnvi Nimble,Priyanka Bhegade,Snehal Surve,Priya Chaugule,(2016) Automatic Smart Car Parking System, International Journal Of Advances in Electronics and Computer Science,Vol-3,Mar-2016.

[3] ElakyaR,Juhi Seth,Pola Ashritha,R Namith(2019),Smart Parking System using IoT,International Journal of Engineering and Advanced Technology(IJEAT),Vol-9,Octorber 2019.

[4] Wael Alsafery,Badraddin Alturki,Stephan Reiff-Marganiec & Kamal Jambi.Smart Car Parking System Solution for the Internet of Things in Smart Cities

[5] Praveen, M., & Harini, V. (2019). NB-IOT based smart car parking system. 2019 International Conference on Smart Structures and Systems (ICSSS).

[6] Bonde, D. J., Shende, R. S., Kedari, A. S., Gaikwad, K. S., & Bhokre, A. U. (2014). Automated car parking system commanded by Android application. 2014 International Conference on Computer Communication and Informatics.

[7] Chowdhury, L. H., Mahmud, Z. N. . Z., Islam, I.-U., Jahan, I., & Islam, S. (2019). Smart Car Parking Management System. 2019 IEEE International Conference on Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON).

[8]) Jayakshei Dadaji Bachhav1; Dept. Of Electronics and Telecommunication, SNJB's COE, Chandwad,Maharashtra, India ,Prof. Mechkul M. A.2; SDept. Of Electronics and Telecommunication, SNJB's COE, Chandwad,Maharashtra, India," Smart Car Parking System", (IRJET), Volume: 04 Issue: 06 | June -2017

[9] Soundarya Rajesh, Dr. M.G. Sumithra, "Smart Parking system using Image processing", International Research Journal of Engineering and Technology(IRJET), Vol 05, Issue 04, Apr42018.

[10] Balwant K. Patil, Avinash Deshpande, Sonal Suryavanshi ,Rudresh Magdum, Manjunath, "Smart Parking System for Cars", International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE).

[11] Jiang Ruili, Wang Haocong, Wang Han University of Electronic Science and Technology of China, Chengdu, China; Dr Eoin O'Connell Computer Science and Information Systems Department University Of Limerick; Dr Sean McGrath Electronic & Computer Engineering Department University Of Limerick, "Smart Parking System Using Image Processing and Artificial Intelligence", International Conference on Sensing Technology (ICST)-2018

[12] Abhirup Khanna; Rishi Anand , University of Petroleum and Energy Studies (UPES) Dehradun, Uttarakhand, “IoT based Smart Parking System”; 2016 International Conference on Internet of Things and Applications (IOTA) Maharashtra Institute of Technology, Pune, India 22 Jan - 24 Jan, 2016

[13] Bibi, N., Majid, M. N., Dawood, H., & Guo, P. (2017). Automatic Parking Space Detection System. 2017 2nd International Conference on Multimedia and Image Processing (ICMIP).

[14] Jaspreet Kaur,(2019).Implementation of Smart Parking using Artificial Intelligence; Computer Science and Engineering, Patiala Institute of Engineering and Technology for women, Nandpur kesho. August 2019 IJSDR, Volume 4,.

[15] Vinay Raj Tripathti, Dept. of Electrical & Electronics, KIET Group of Institutions,Ghaziabad;2020 International Conference on Electrical and Electronics Engineering (ICE3-2020).