**Integrating camera-based solutions for image processing to detect parking space availability.**

TEAM MEMBER:R.Ragavan

Phase\_2 document submission

**Introduction**

* The rapid advancements of the third industrial revolution and computer technology have ushered in a new era of “smart life”, where integration with smart devices, cities, airports, and more is becoming a reality.
* In this context, many urban designers and planners did not anticipate the pace of technological progress and its continued development.
* However, the city of Macao is actively promoting education, training, and awareness related to smart city development, intending to significantly enhance the overall quality of life.
* One area where smart technology can have a significant impact is the airport, which is a crucial infrastructure that can benefit from innovations such as smart parking and automated baggage handling.
* While there have been successful implementations of smart technology in airports around the world, such as London Heathrow [and Singapore Changi Airport most research studies focus on improving the “in-system” experience, with less attention being paid to the “out-system”, such as the parking lot. This is where our contribution comes in.
* In this paper, we propose an innovative, economical, and plug-and-play smart parking system for Macao International Airport that integrates various methods such as IPM (inverse perspective mapping), object detection with deep learning, and a guidance system with a rig of multiple cameras.
* Our system is easy to install, making it an attractive solution for other airports looking to adopt smart parking technology. The possibilities of smart technology are endless, and our contribution is just the beginning of what can be achieved with the integration of smart technology in airport management.

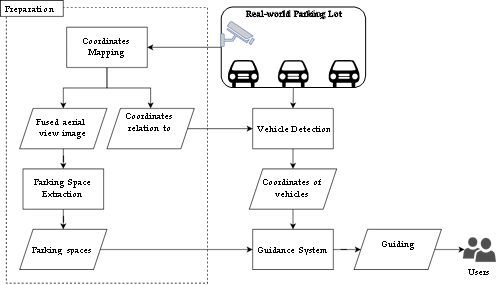
# 1.System Architecture:

# In this section, we will describe the architecture of our proposed smart parking system based on surveillance videos.

# Figure [1](#_bookmark2) illustrates the various major components of the system.

# The operations in the preparation stage are only required to be performed once before the system starts running.

# During preparation, the system takes images from its cameras to generate coordinate relations and extract parking spaces. When the system is running, it uses images from its



**Parking Lot Coordinates Mapping:**

* Perspective transformation is widely used in image restoration, which is the projection of an image onto a new view plane.
* For example, when two cameras take pictures of a plane at different angles, the coordinates of the same point in the plane are different in the two pictures.
* Perspective transformation achieves the function of mapping the same point in the two pictures.
* By the same token, when multiple cameras are taking pictures of a parking lot, the picture of each camera contains information about a part of the parking lot
* . Using this series of transformations can make it possible to obtain aerial view images without the need for special equipment, such as a UAV
* .Many methods can achieve coordinate mapping, such as stereo-based , adaptive- based , and 4-point-based methods.
* The 4-point-based method is recommended as the preferred option in our system among these methods because it has fewer factors (parameters of algorithms, position or pose of cameras changed by installation mistakes, or extreme weather) involved.
* In addition, the 4-point-based method completely satisfies the requirements of this system with regard to robustness; therefore, we apply this method to our system.
* There are two steps to achieve parking lot coordinate mapping using a 4-point-based method:

**The workflow of parking space extraction:**

**Target Line Extraction:**

* In this stage, the lines of the parking spaces are extracted for constructing bounding boxes.
* Firstly, the preprocessing stage contains a blur function. When transforming surveillance images to aerial view images, the pixels are mapped onto the fused image
* . The pixels far from the camera are expanded and distorted. As a result, the edges of transformed lines may be jagged, which affects the performance of the following steps.
* The blur function can alleviate this situation. Secondly, we use the Canny edge detector [[15](#_bookmark41)] to detect the edges in the fused image.
* The result of the edge detector is an image containing edges, where the image has the same size as the fused image. To further ensure the edges of the parking space can be extracted as lines, we perform a dilation operation on the edge image in case the edges are shattered.
* The third step is to extract lines from the edges
* . A line detection method based on Hough Transformation is provided in [We use this method to extract lines.
* The majority of the extracted lines are not from the edges of parking spaces; therefore, we need to filter the lines. Since the fused aerial view image is generated with respect to the real coordinates, the length of parking spaces in pixels can be calculated.
* Extracted lines with large length differences from the actual length are filtered. In addition, the direction of parking spaces can also be used to filter the lines.
* For example, if the parking spaces are perpendicular to the edge of the parking lot, the lines that are not perpendicular to the edge of the parking lot are filtered.
* If there are multiple alignment directions of parking spaces, the lines perpendicular to each direction can be reserved. After this step, the lines reserved should be those extracted from the edges of parking spaces and the length should be similar to the edges.

**Bounding Box Extraction:**

* To extract the boxes of parking spaces, we consider,
* repeated boxes with the same size as the characteristic. Similar to filtering lines, the distance of lines, i.e., the width of the parking space, can be calculated
* . Figure [4](#_bookmark8) illustrates extracting the bounding box with a verification line.
* Firstly, when there are two lines with centers in one of the alignment directions and their distances meet the expected length, the box they form is considered as a candidate bounding box.
* Then, we follow the alignment direction and the opposing direction to check if there exists a verification line.
* The verification line should have the same direction and its distance to the candidate bounding box should meet the expected distance.
* If a verification line is found, we consider the candidate bounding box to be a box of a parking space and store it for the next step. If the parking spaces have multiple alignment directions, the lines can be divided into groups depending on their direction, and then we perform the extraction. Finally, sometimes more than one line is extracted from the edge of a parking space, which leads to redundant boxes being extracted. Therefore, in the last step of box extraction, we calculate the IOUs

**Route Modeling:**

* Route information is another important type of information in a parking lot.
* However, it is difficult to extract because there is no obvious and common characteristic of routes in different parking lots.
* Furthermore, some routes are designed to be one-way and the direction signs are variable, making it more difficult to extract information accurately.
* Compared with the parking spaces, routes are easier to label manually as the number of routes in a parking lot is much smaller than the number of parking spaces.
* Therefore, in our work, we label the routes and their directions manually from the fused aerial view image.
* A graph is a classical data structure using vertices (also known as nodes) and edges to express the relationships among vertices.
* An edge usually contains a weight, which refers to the consumption between two nodes. Because the routes are one-way in some parking lots, a directed graph can be used to model a parking lot.
* The turns in the parking lot are modeled by nodes in the graph.
* An edge in the graph represents the road between two turns and its direction follows the direction of the road.
* Meanwhile, the entrance and exit of the parking lot are modeled as nodes as well. A node in the graph records its coordinates
* and the weight of an edge is the distance between two vertices.

**SIMPLE PROGRAM:**

**Requirements:**

1. Camera: You'll need a camera (e.g., USB webcam or Raspberry Pi Camera Module).
2. Python with OpenCV: Ensure you have Python installed along with the OpenCV library for image processing.
3. GUI Framework: You can use libraries like Tkinter or PyQt for the user interface.

Here's an outline of the program:

**1. Camera Setup:**

First, initialize and configure your camera for capturing video frames.

**import cv2**

**# Initialize camera (0 for default camera, change if necessary)**

**cap = cv2.VideoCapture(0)**

**# Set camera resolution and frame rate (adjust as needed)**

**cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 640)**

**cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480)**

**cap.set(cv2.CAP\_PROP\_FPS, 30)**

**2. Image Processing:**

**Implement image processing algorithms to detect parking space availability. In this example, we'll use basic techniques to detect empty parking spaces based on color thresholding.**

**def detect\_parking\_spaces(frame):**

**# Convert the frame to grayscale (if necessary)**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**# Apply color thresholding to detect empty parking spaces**

**lower\_color = (0, 0, 200) # Adjust the color range**

**upper\_color = (50, 50, 255)**

**mask = cv2.inRange(frame, lower\_color, upper\_color)**

**# Find contours of parking spaces**

**contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)**

**# Draw rectangles around empty parking spaces**

**for contour in contours:**

**x, y, w, h = cv2.boundingRect(contour)**

**cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)**

**return frame**

3. User Interface:

Create a simple GUI to display the camera feed and parking space availability.

python

**import tkinter as tk**

**from tkinter import PhotoImage**

**from PIL import Image, ImageTk**

**# Create a tkinter window**

**window = tk.Tk()**

**window.title("Parking Space Detection")**

**# Create a label for displaying the camera feed**

**label = tk.Label(window)**

**label.pack()**

**def update():**

**ret, frame = cap.read()**

**if ret:**

**processed\_frame = detect\_parking\_spaces(frame)**

**processed\_frame = cv2.cvtColor(processed\_frame, cv2.COLOR\_BGR2RGB)**

**img = Image.fromarray(processed\_frame)**

**img = ImageTk.PhotoImage(image=img)**

**label.config(image=img)**

**label.image = img**

**window.after(10, update)**

**update() # Start updating the camera feed**

**window.mainloop()**

**CONCLUSION:**

A smart parking system offers numerous benefits and improvements over traditional parking management approaches. In conclusion, here are some key points to consider:

1. **Efficiency:** Smart parking systems significantly improve the efficiency of parking space utilization. They reduce the time spent searching for parking spots, resulting in less congestion and traffic in parking lots and urban areas.
2. **Convenience:** Users can easily find available parking spaces through mobile apps or real-time signage, making the parking experience more convenient and user-friendly.
3. **Cost Savings:** Businesses and municipalities can optimize parking resources and reduce operational costs by efficiently managing parking spaces. This can include dynamic pricing to encourage off-peak usage.
4. **Environmental Benefits:** Reduced circling for parking spots leads to fewer emissions and contributes to environmental sustainability. It also reduces fuel consumption and associated air pollution.