**Simple Smart Parking System Via Raspberry Pi**

Aashik Rasool, Mushtariy Abdurakhmonova, Juraev Hasan

Department of Computer Engineering, Gachon University, Seongnam, Republic of Korea

Abstract - An efficient and intelligent method to automate the management of parking systems utilizing Internet of Things (IoT) technology is proposed to allocate parking space efficiently. IoT allows for wireless access to computers and enables users to monitor parking spaces remotely. Thus, we propose a system employing a Raspberry Pi camera, ultrasonic HC-SR04 sensors, and a 9g servo motor. This system detects available spaces and processes image detection, handling data within a Python text file efficiently.

**Introduction**

The parking system faces many challenges within the parking environment. To address these issues, a smarter parking system has been developed. Different approaches and ongoing research aim to overcome the difficulties encountered in parking lots, which are often congested due to existing systems and technologies. Various systems and methods are described in subsequent sections. The technology in the parking system uses a Wireless Sensor Network (WSN) for identification and a communication network process. It has the capability to transmit data over an IoT network without human interaction. IoT also provides users with affordable access to wireless technology, allowing for data transfer to the cloud and supporting transparency. The concept of IoT began with the identification of objects to connect various devices, controlling or monitoring these devices via a computer through the Internet. There are two key terms in IoT: "Internet" and "things," where the Internet is an extensive network that connects servers and devices, enabling the sending, receiving, and exchange of information. The parking problem contributes to air pollution and traffic congestion. In today's scenario, finding parking is a daily challenge for many people. According to recent forecasts, by 2035, the number of parking spaces will need to accommodate over 1.6 billion cars. The world consumes approximately a million barrels of oil daily. Therefore, an intelligent parking system is a crucial solution to reduce fuel wastage. This solution addresses the issue by using sensors to collect data, which is then analyzed and processed to produce a result.

**SMART PARKING SYSTEM**

The car parking system has now evolved into an intelligent system with various technologies and advanced research. Smart parking systems are implemented in numerous environments to address the problems encountered in daily activities, benefiting users significantly. The system's structure is based on the principle of multiple functions. There are three tiered functions: the basic detection function at the lower level, data sharing at the middle level, and data storage, processing, and client interfaces at the higher level. The system architecture, as depicted in Figure 1.

Diagram

Description automatically generated

Fig 1. (It’s connected to our 1st Raspberry Pi)

Diagram

Description automatically generated

Fig.2(2nd Raspberry Pi)

Diagram

Description automatically generated

Fig 3. (3rd Raspberry Pi)

At the entrance, the Raspberry Pi detects an object within 10cm using the ultrasonic sensor and returns a parameter of 1. If the parameter is 1, it triggers the servo motor to open to 90 degrees. After 5 seconds, the toll gate will close automatically. In the parking lot, if a vehicle approaches, the Raspberry Pi detects the object within 10cm with the ultrasonic sensor, returns a parameter of 1, and the camera begins to detect the number plate and stores it in a text file. Should another vehicle arrive, it appends this new information to the same file. At the exit, the camera detects the number plate and removes the corresponding entry from the text file.

A picture containing athletic game, sport

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

**Method and Theory**

1. **Object Detection (Canny Edge Detection)**

Canny Edge Detection is one of the most popular algorithms for finding edges. This algorithm was developed in 1986 by John F. Canny. Canny Edge Detection consists of a multi-step algorithm, which is detailed below.

Finding the edges properly can be difficult when there is noise present in the image. Therefore, the first step in the algorithm is to reduce the noise by applying a 5x5 Gaussian filter to the image. After noise reduction, gradients in each direction are obtained by applying the Sobel kernel—previously learned—to the smoothed image in both horizontal and vertical directions. The gradient in the horizontal direction is denoted as Gx, and in the vertical direction as Gy. The edge gradient at a pixel located at (x, y) can be calculated using the following equation:



The entire image is then scanned to remove pixels that do not contribute to the edges. While scanning, the algorithm identifies the pixel with the maximum gradient value in the scan area along the direction of the gradient. Refer to the image below for a visual aid in understanding this process.

Chart

Description automatically generated

1. **Distance measurement using waves:**

Diagram

Description automatically generated

To calculate distance, use the formula D = S × T, where D represents distance, S is speed, and T is time.

**Distance = Speed × Time**

Rate and speed are similar concepts because they both describe a certain distance covered within a unit of time, such as miles per hour or kilometers per hour. If the rate (R) is equivalent to speed (S), then we have R = S = D/T. This leads to the formula D = R × T, which indicates that distance is equal to rate times time.

**Distance = Rate × Time**

To calculate speed or rate, use the formula S = D/T, which states that speed equals distance divided by time.

**Speed = Distance ÷ Time**

To determine the time, apply the formula T = D/S, signifying that time is equal to distance divided by speed.

**Time = Distance ÷ Speed**

**References**

1. Author Last Name, First Name. Year. "Title of Article." Journal Name volume number (issue number): page range. DOI.
2. Joshi, Aashish, Arni Tharakaram Hariram, K M Vishal Somaiya, and Mubashir Hussain. Year. Title of the Work. Bangalore: Publisher.
3. Samsjang. Year. "Title of Blog Post." Blog Name. URL.