Autonomous Parking System

YOUZHE DOU, Department of Electrical and Computer Engineering, Johns Hopkins University.

***Abstract* – This autonomous parking system project aims to develop an intelligent system that will be an essential part of autonomous vehicles. The system provides general algorithms and standard procedures for a car to find an empty parking slot and park itself. The system is implemented using a Finite State Machine (FSM) because of its step-by-step nature. An Arduino controlled robotic chassis is used to mimic real-life vehicles. Sensors such as ultrasonic range finders, magnetometer and photonic wheel encoder are used to extract information and parameters that are important for motor control and decision making at each state. This system is scalable and can be tuned to cope with vehicles of any size and parking facilities of any structures accordingly.**

I. Introduction

Driverless driving is becoming a trend in the motor vehicle industry. During the CES 2017, all the major car manufactures such as BMW, Volkswagen and Audi demonstrated their prototype for self-driving vehicles. As one of the essential part of self-driving, self-parking will be more frequently used since it is safer and the related technology is well developed. Different from traditional parking guidance tools such as rear camera and parking alarm, the self-parking technology aims to perform fully autonomous parking without any help needed from the driver. Self-parking significantly helps drivers who are not confident in their parking skills. Other than increasing the efficiency for the parking facilities, this hassle-free process also eliminates any safety concern for inexperienced drivers.

Such a complex system consists of many small tasks which will be carried out step by step. There are many researches focus on each task and solve the problems using different approaches. One way proposed to construct the environment is to use range finders arrays to build a 3D occupancy grids [1]. Kinematics models are created using both differential wheels [3] and Ackermann steering [4]. These models include parameters that can be modified according to the prototype. Daobin Wang proposed a systematic solution to perform self-parking using finish state machine (FSM) and fuzzy logic to make a robust system for different environment.

II. Approach

The approach consists of two steps, which is same as the real-life situation when a driver want to park a car. The first step is to search for an empty parking space in a car facility and the second step is to perform parking. There will be some additional features such as remote control and moving object detection to make the process safer and more user friendly.

A. Locate available parking slot

During this process, the car will be autonomously search around for available space. All the sensors are active and keep collecting information about the environment around the vehicle until the searching process is terminated. The approach used for this task is in door mapping using ultrasonic proximity sensors.

The entire space of the parking lot is represented by a 2D array of cells. A value ‘1’ means that there is something occupies the cell and therefore the car should never go to that point. A value ‘0’ means that the cell is empty and it could potentially be a parking slot. The size of this array determines the resolution of this mapping. Higher resolution yields better results with longer processing time which could be disastrous for such real-time system. Therefore, tuning of the size should be carried out during experiments.

Although indoor mapping without fixed reference points tends to produce poor results, it should be enough for parking slot detection since the parking space is relatively large compared to a single cell.

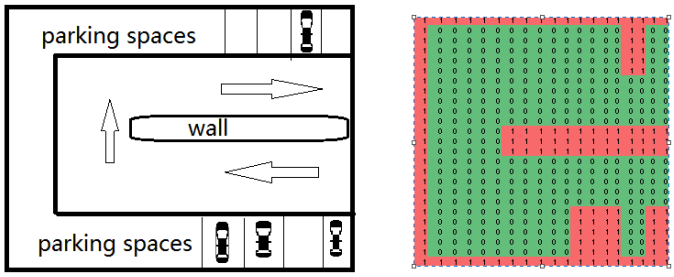

Figure 1 Indoor mapping using occupancy grids

B. Perform perpendicular parking

After finding the available parking slot, the vehicle will start to perform parking. The procedure is similar to what people usually do. The car starts to reverse at the position a little bit over the parking spot and follow a curved path into the spot.

The curve will be calculated using Bezier curve which generates a curve based on a few points depends on the order. The path following technique requires a transformation matrix relates joint space (steering angle) and Cartesian space (position and orientation of the car). This transformation matrix will be developed based on the parameters of the prototype.

C. *Additional features*

Other than the two steps system mentioned earlier, there are some additional features need to be implemented to build a safe and efficient self-parking system. For example, moving object detection and interrupt are needed if there is some people walking in the car park and the vehicle should stop immediately if the person is nearby. Another feature is that the system should not only be able to park the car, but also summon the car for the driver. Obstacles avoidance and path planning is another essential feature.

III. IMPLEMENTATION

Components selection

A. *Chassis*

In this project, a four-differential-wheeled chassis is used. There are four 6V DC motors with tachometer encoder attached. Only two motors are installed to mimic the real-life two-wheel drive vehicle. The dimension is 9.6 x 6 x 2.2 inches. Without motor drive controller, the speed of the car is fixed at 48m/min.

B. *Power*

Since Arduino board is not able to provide enough current for DC motors and speed and direction control is essential in this project, a motor driver controller is needed. L293D is initially tried. It provides 6V-9V with maximum current 600mA and speed control from 0-225. It works perfectly with higher speed (>150). However, this project requires the car to be able to move at low speed. Therefore, L298N which can provide current up to 2A is preferred.

C. *Microprocessor*

Arduino Mega is selected because there are not enough analog and digital pins on Arduino Uno. There are 54 digital and 16 analog pins on Arduino Mega and the clock speed is 16MHz.

D. *Ultrasonic sensors*

HC-SR04 distance sensors will be used to build the sensor array because of its low price, high resolution (1 cm) and wide range (2 cm – 500 cm). They can be operating at 40 kHz. They are also compatible with Xbee Zigbee which can be useful to implement wireless information transfer in the future.

E. *Triple axis magnetometer*

It is essential to know the orientation of the car during the parking process. HMC5883L is a triple axis magnetometer with I2C interface. It provides accuracy of 1-2 degree and 160 Hz data rate. There is also an integrated 12 bits ADC on the Adafruit breakout which makes easier to get readings.

F. *Photoelectric codec wheel speed encoder*

Speed encoder is a photoelectrical sensor can generate a pulse each time when something is blocking the transmitter and the receiver. HC-020K is selected because it can operate at high frequency and low current consumption.

Algorithm implementation

A. *Indoor mapping generation*

The distance sensors on the side are used to get readings and build the map. While the car is moving forward to search for available spot, the map is building inside the memory. As mentioned in the approach section, if the reading is x cm, the cell that is x cm away from the sensor will be marked as '1' which indicated that it is occupied.

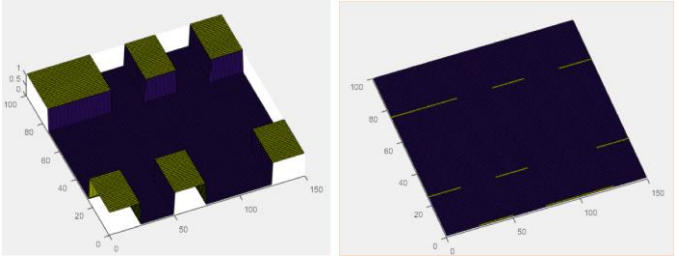

Figure 2 Indoor mapping algorithm simulation result

B. *Path generation*

The paths for both perpendicular and parallel parking are predefined and they are similar to real-life scenarios. Taking perpendicular parking for example, the starting point and the destination are predefined and a smooth curve is generated using cubic Bezier curve.

$$B(t) = (1-t)^3 Pt0 + (1-t)^2 tPt1 + (1-t)t^2 Pt2 + t^3 Pt3, 0 \le t \le 1$$

The smoothness of this curve is dependent on the amount of increment of t and there will be a heading value assigned with each point on the curve.

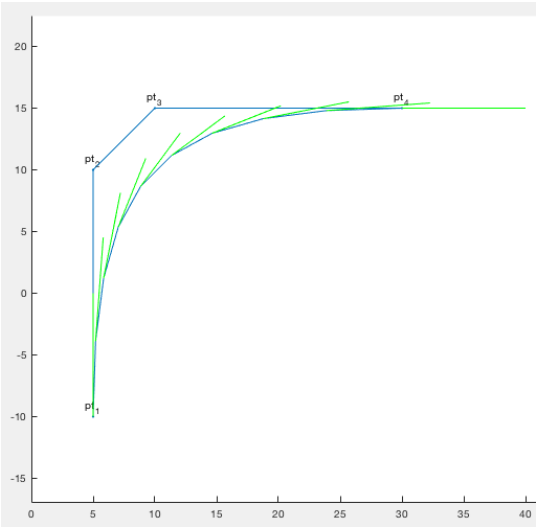

Figure 3 Ideal perpendicular parking trajectory

C. *Locomotion and kinematics*

The robotic chassis used in this project is equipped with differential drive wheels and the direction of motion is controlled by separately controlling speeds $v_l$and v_r of the left and right wheels respectively. Although differential-drive robot is able to turn "on the spot" by setting $v_l$and v_r same magnitude and different signs, this property is not used in this project because it will be irrelevant and impossible to apply

the motion algorithm to real-life cars with Ackermann steering.

Forward kinematic equations which describe the relationship between v_l ,v_r and the resulting path are derived based on the parameters of the specific model of chassis. These parameters include wheel size, the distance between left and right wheels and the Instantaneous Center of Curvature (ICC) or Instantaneous Center of Rotation (ICR).

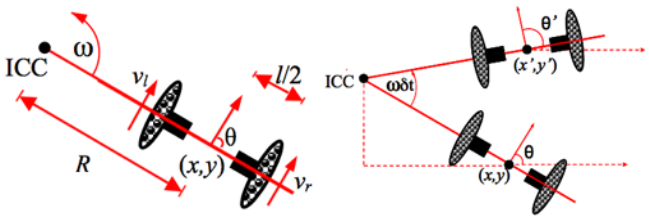

Figure 4 Locomotion for differential drive

The following equations can be easily derived based on the symbols used in the diagrams above.

$$\text{Angular velocity: } \omega * r = v$$

$$\omega * \left(R + \frac{l}{2}\right) = v_r$$

$$\omega * \left(R - \frac{l}{2}\right) = v_l$$

$$R = \frac{l}{2} * (v_r + v_l)/(v_r - v_l)$$

$$\omega = \frac{v_r - v_l}{l}$$

The new heading can be calculated using angular velocity and the time interval:

$$\theta' = \theta + \omega\Delta t$$

Adding the center of rotation ICC into the space:

$$ICC = [ICC_x, ICC_y] = [x - Rsin\theta, y + Rcos\theta]$$

$$\begin{bmatrix} x' \\ y' \\ \theta \end{bmatrix} = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\Delta t \end{bmatrix}$$

D. Precise control of the motor

The car is expected to be moving along either a straight line or a smooth curve in most cases. Ideally, the position of the car can be calculated based on the speed of each wheel. For example, if all wheels are turning at the same speed, we expect the car to be at some distances along the direction it is facing. However, in reality, due to the differences of the wheels, DC motors and the friction of the surface, the car is not moving along a straight line even the motors are programmed to run at the same speed.

There are mainly three methods to solve this problem. The first method is to use a gyroscope with a PID controller to adjust the wheel speed based on the current heading and the desired heading. However, low price electrical gyroscopes suffer from drift. The x,y,z-axsis rotation cannot be measured directly and they are calculated by integrating over the rotation acceleration. Taking MPU6050 for example, for z-axis rotation (Yaw), the heading is only accurate for around 3 seconds due to the lack of reference and the drift. The second method is using external reference such as a wall to make sure that the car is moving straight. This however does not tackle the curve path. A better approach is to use photoelectric wheel speed encoder with code wheels attached to motors. By counting the exact number of pulsed given by the sensors, the number of turns of the wheel can be calculated. Therefore, the distances each wheel traveled will be known.

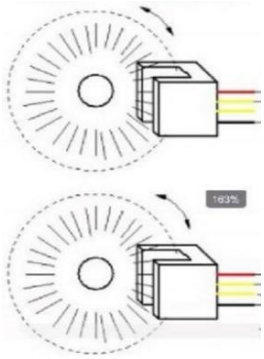

Figure 5 Photoelectric wheel speed encoder

IV. RESULT AND DISCUSSION

The working prototype is show below.

Size	9.6*6*3 inches
Weight	4.1 pounds
Power supply	9V
Time (searching for parking space)	Depends on parking facility
Time (perpendicular parking)	2.5 sec
Time (parallel parking)	4 sec
Space margin (perpendicular parking)	3 cm on left and right side
Space margin (parallel parking)	3 cm on left and right side 5 cm on front and back side

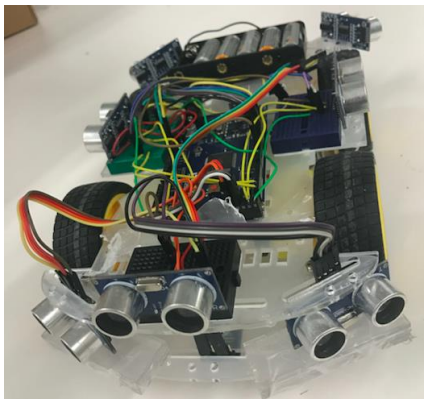

Figure 6 Working prototype

The searching for parking spot part and perpendicular parking part have 100% success rate. However, parallel

parking only works sometime. This proves that the algorithm itself is correct. The reason is that parallel parking requires a more complicated curve and the car must follow the curve more precisely. This can be achieved using more powerful motors with more accurate encoders.

V. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Ralph Etienne-Cummings, Prof. Pedro M. Julian , TAs: Johns Rattray, Tao Xiong and my fellow classmates for their guidance and suggestions. Without their support, it is not possible for me to learn and accomplish this mush.

REFERENCES

[1] Morris, William, Ivan Dryanovski, and Jizhong Xiao. "3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids." *In RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras*. 2010.

[2] Wang, Daobin, et al. "Research on self-parking path planning algorithms."*Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference on*. IEEE, 2011.

[3] DeSantis, Romano M. "Modeling and path-tracking control of a mobile wheeled robot with a differential drive." *Robotica* 13.04 (1995): 401-410.

[4] Simionescu, P. A., and D. Beale. "Optimum synthesis of the four-bar function generator in its symmetric embodiment: the Ackermann steering linkage." *Mechanism and Machine Theory* 37.12 (2002): 1487-1504.