INTELLIGENT PARKING SPACE DETECTION SYSTEM BASED ON IMAGE PROCESSING

- R. Yusnita, Fariza Norbaya, and Norazwinawati Basharuddin

IMAGE PROCESSING

Introduction

With the rapid economic and social development, the number of motor vehicles has increased rapidly. In contrast, the construction of parking lots has been relatively slow, and the problem of parking difficulties has become increasingly prominent. Research on parking space detection methods can not only effectively increase the utilization rate of parking spaces but also alleviate the problem of limited parking space resources, and meet the requirements of the parking lot, including efficiency, safety, and management. Currently, there are many detection methods for parking spaces. According to the categories of the selected sensors, they can be divided into visual and non-visual detection methods.

Motivation

Drivers will be able to find the best spot available, saving time, resources, and effort. Overall traffic and fuel wastage will be reduced as the search for open parking space will be optimized. Parking lot employees and security guards contain real-time lot data that can help prevent parking violations and suspicious activity. License plate recognition cameras can gather pertinent footage to locate any suspicious drivers. Automation and less manual activity saves on labor cost and resource exhaustion.

Working

This Project is implemented by detection of circles in the parking lot using Hough Circle Transform which is a basic feature extraction technique used in digital image processing for detecting circles in imperfect images. The circle candidates are produced by "voting" in the Hough parameter space and then selecting local maxima in an accumulator matrix.

Hough Circle Transform

In a two-dimensional space, a circle can be described by: $(x - a)^2 + (y - b)^2 = r^2$ where (a,b) is the center of the circle, and r is the radius. If a 2D point (x,y) is fixed, then the parameters can be found according to (1). The parameter space would be three-dimensional, (a, b, r). And all the parameters that satisfy (x, y) would lie on the surface of an inverted right-angled cone whose apex is at (x, y, 0). In the 3D space, the circle parameters can be identified by the intersection of many conic surfaces that are defined by points on the 2D circle. This process can be divided into two stages. The first stage is fixing the radius then finding the optimal center of circles in a 2D parameter space. The second stage is to find the optimal radius in a one-dimensional parameter space.

Gaussian Blur

It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination.

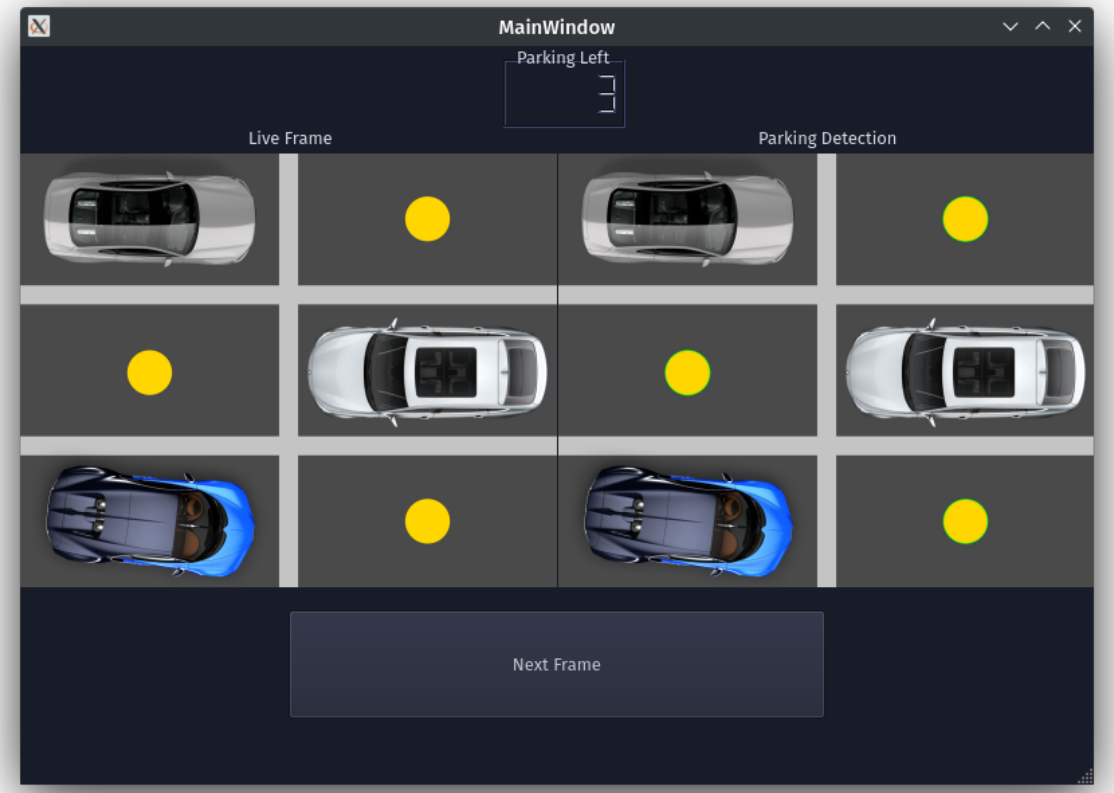
1. **Get the frame of the video.**
2. **Convert the RGB image to Greyscale.**
3. **Apply Gaussian filter to remove any noise.**
4. **Detect the circles using Hough Circle Transform.**

Implementation

```
def show_image(self):

        self.count+=1

        self.image_link =
['Images/Parking_Empty.jpg','Images/Parking_1.jpg','Images/Parking_3.jpg',
'Images/Parking_Full.jpg','Images/Parking_5.jpg']

        self.img = cv2.imread(self.image_link[self.count])
```

```
        self.output = self.img.copy()
        self.img1 = QtGui.QImage(self.img.data, self.img.shape[1],
self.img.shape[0],QtGui.QImage.Format_RGB888).rgbSwapped()
        self.imageinput.setPixmap(QtGui.QPixmap.fromImage(self.img1))
        self.detect_circles()

    def detect_circles(self):
        self.img = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
        self.img = cv2.GaussianBlur(self.img, (21,21), cv2.BORDER_DEFAULT)

        # Find circles
        circles = cv2.HoughCircles(self.img,
cv2.HOUGH_GRADIENT,0.9,120,param1 =50, param2 = 30 , minRadius= 60,
maxRadius=100)
        # If circle is found
        if circles is not None:
            # Get the (x, y, r) as integers
            circles = np.round(circles[0, :]).astype("int")
            self.lcdNumber.display(circles.shape[0])
            # loop over the circles
            for (x, y, r) in circles:
                cv2.circle(self.output, (x, y), r, (0, 255, 0), 2)
        else:
            self.lcdNumber.display(0)
        cv2.imwrite('circles_output.jpg',self.output)
        self.imageoutput.setPixmap(QtGui.QPixmap("circles_output.jpg"))
```

Screenshot



Resources

http://ijimt.org/papers/228-G0038.pdf

https://en.wikipedia.org/wiki/Gaussian_blur

https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

https://doc.qt.io/qtforpython/