第 1 章　智能应用开发项目实战

1.1 项目需求

如今，停车场的管理已经迈入了一个全新的智能化时代。在停车场的出入口，不再需要管理人员手动记录车辆的出入时间或计算停车费用。每个关键位置都配备了摄像头和先进的自动闸门系统。车辆入场时，系统能迅速自动识别车牌，并精确记录车辆的入场时间。而当车辆准备离开时，系统会再次识别车牌，获取当前时间，并即时计算出相应的停车费用。为方便支付，出口处的左侧还醒目地张贴了收款二维码。只需简单地扫描这个二维码，您就能迅速完成停车费用的支付，随后轻松驶离停车场。整个过程，实现了高度的智能化和自动化，为用户提供了便捷、高效的停车体验。

根据以上场景，实现《停车场车牌智能识别计费系统》，包括：

车辆视频显示模块：实时获取车辆监控视频并管理界面显示。

车辆图像抓拍模块：从车辆监控视频中，抓拍车辆图像并保存。

车辆车牌识别模块：利用保存的车辆图像，对车牌进行识别，获取车牌号，其中识别分为三种方式：（1）在线百度 API；（2）本地第三方库；（3）本地自训练模型。

车辆入场信息模块：车辆是否入场判断，车辆入场信息管理，停车场车位管理。

车辆出出场信息模块：车辆是否出场判断，车辆出场信息管理，停车场车位管理。

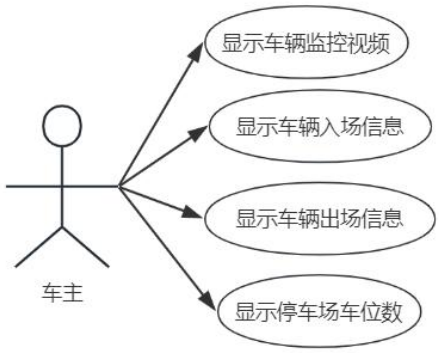
停车场车辆模块：停车场现有车辆信息显示。

停车场车位模块：停车场现有剩余停车位显示。

车辆图片选择模块：对于无法自动车牌识别，出入场车辆，可手动选择非监控拍摄的车辆图片进行车牌识别。

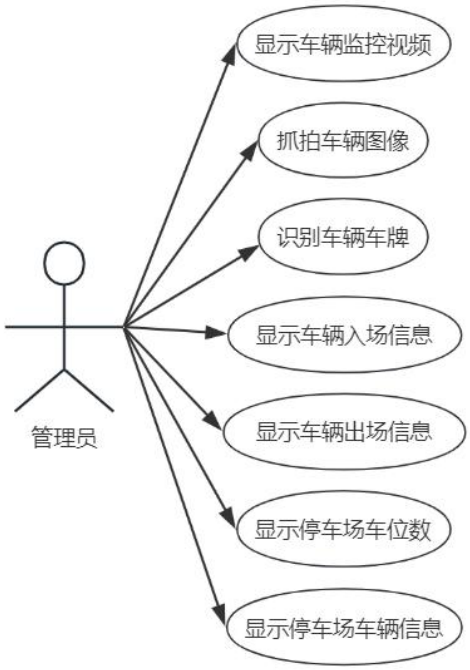
1.2 需求分析

1、车主用例图及功能描述



功能	详细说明
显示车辆监控视频	获取车主车辆入场前监控视频，实时显示在管理界面。
显示车辆入场信息	显示车主车辆入场信息，包括：车牌号、入场时间等。
显示车辆出场信息	显示车主车辆出场信息，包括：车牌号、出场时间、停车费等。
显示停车场车位数	显示停车场剩余车位数。

2、管理员用例图及功能描述



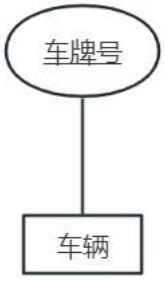
功能	详细说明
显示车辆监控视频	获取车主车辆入场前监控视频，实时显示在管理界面。

抓拍车辆图像	抓拍并保存车主车辆图像。
识别车辆车牌	识别车辆图像中车牌号，识别分为三种方式：（1）在线百度 API；（2）本地第三方库；（3）自训练模型。
显示车辆入场信息	显示车主车辆入场信息，包括：车牌号、时间等。
显示车辆出场信息	显示车主车辆出场信息，包括：车牌号、时间等。
显示停车场车位数	显示停车场剩余车位数。
显示停车场车辆信息	显示停车场车辆信息，包括：车辆总数、车牌号、时间等。

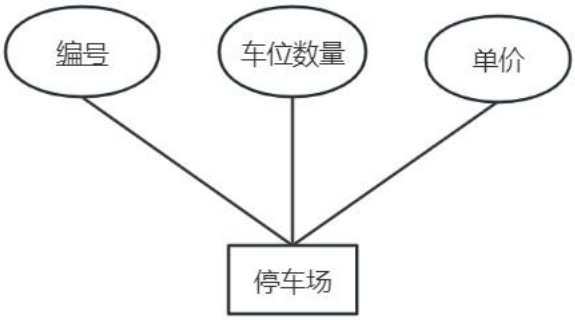
1.3 数据库设计

1.3.1 E-R 图

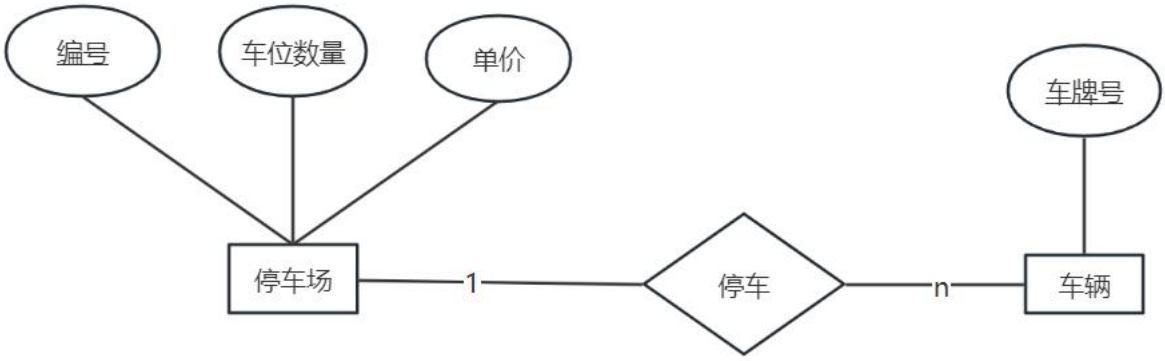
1. 车辆



2. 停车场



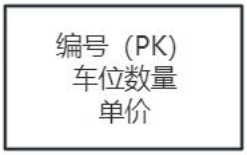
3. 停车场停车



1.3.2 模型图

1. 车辆

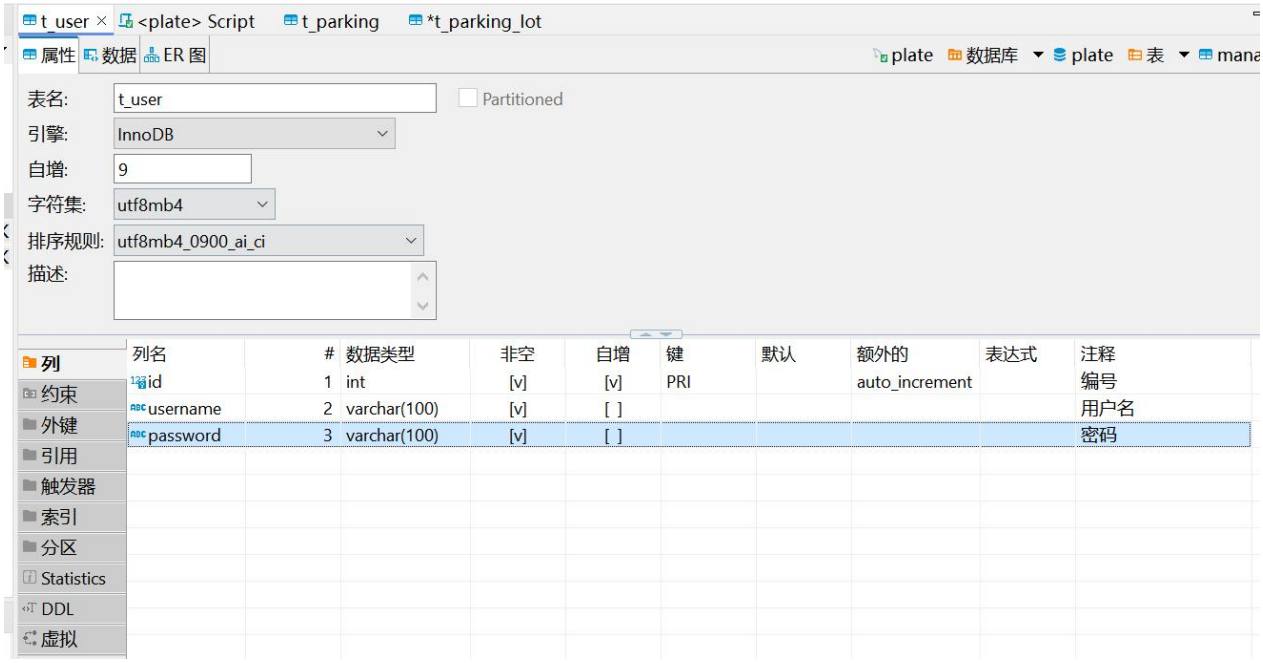
2. 停车场



3. 停车场停车

1.3.3 物理表

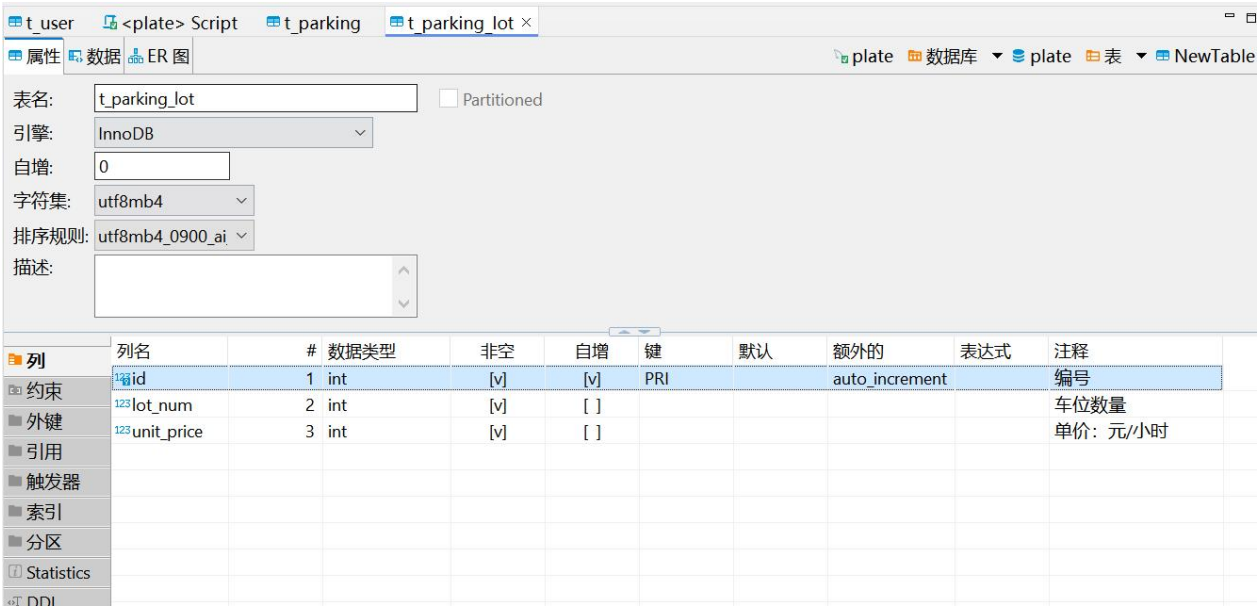
用户表：t_user



数据库脚本：

```
-- plate.t_user definition


CREATE TABLE `t_user` (
  `id` int NOT NULL AUTO_INCREMENT COMMENT '编号',
  `username` varchar(100) NOT NULL COMMENT '用户名',
  `password` varchar(100) NOT NULL COMMENT '密码',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


-- 样例数据
INSERT INTO plate.t_user (username,password) VALUES
    ('admin','123'),
    ('user','123');
```

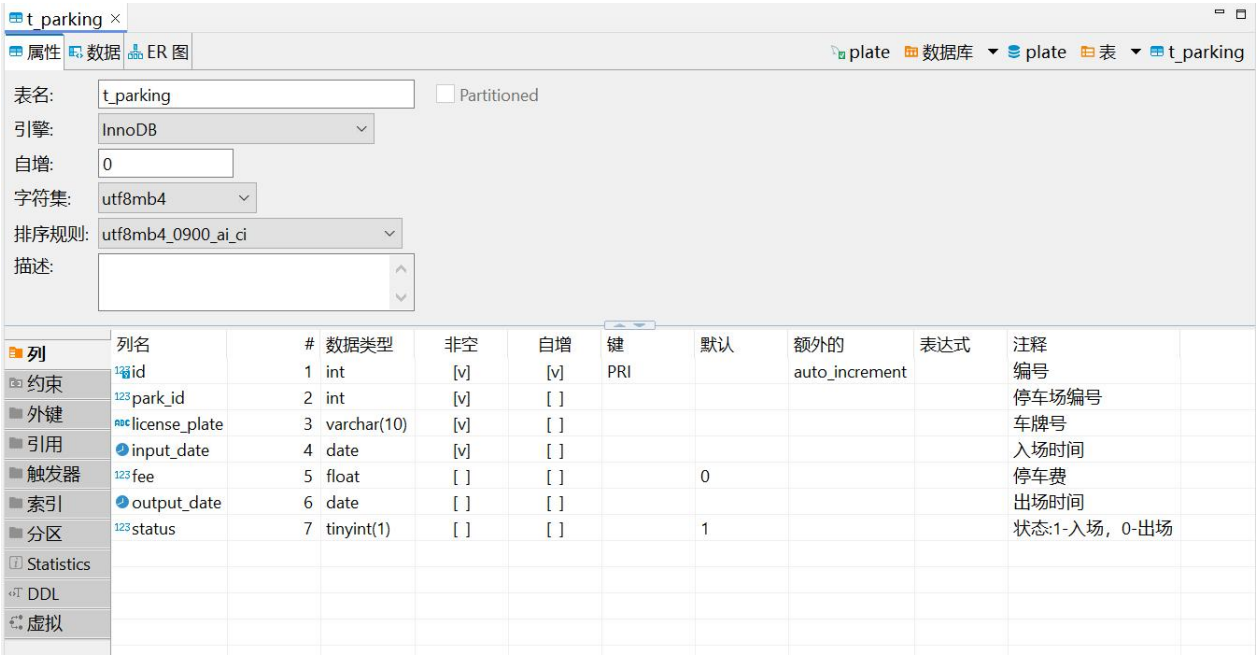
停车场表：t_parking_lot



数据库脚本：

```
-- plate.t_parking_lot definition


CREATE TABLE `t_parking_lot` (
  `id` int NOT NULL AUTO_INCREMENT COMMENT '编号',
  `lot_num` int NOT NULL COMMENT '车位数量',
  `unit_price` int NOT NULL COMMENT '单价：元/小时',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


-- 样例数据
INSERT INTO plate.t_parking_lot (lot_number,unit_price) VALUES
      (96,3);
```

停车场停车表：t_parking

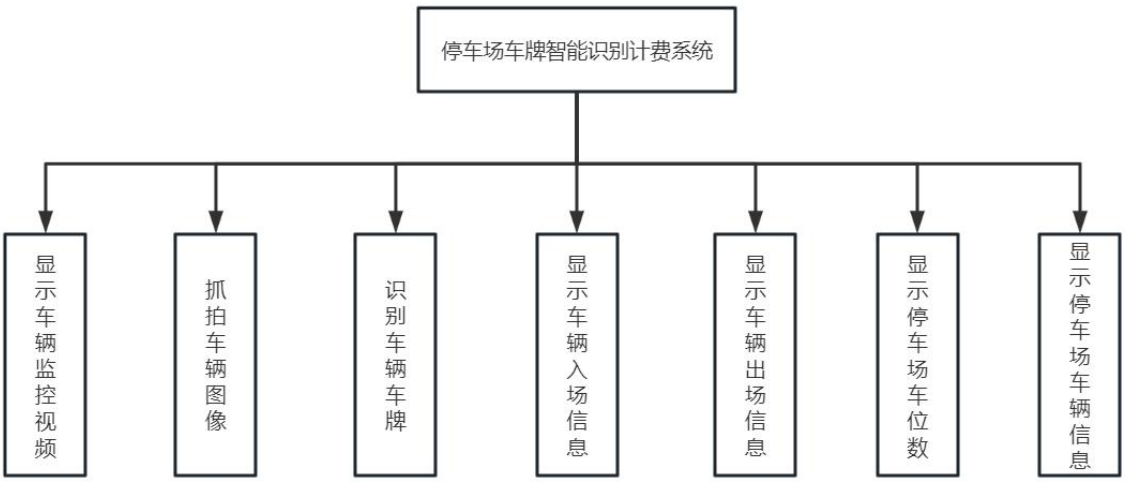


数据库脚本：

```
-- plate.t_parking definition


-- plate.t_parking definition


CREATE TABLE `t_parking` (
  `id` int NOT NULL AUTO_INCREMENT COMMENT '编号',
  `park_id` int NOT NULL COMMENT '停车场编号',
  `license_plate` varchar(10) NOT NULL COMMENT '车牌号',
  `input_date` datetime NOT NULL COMMENT '入场时间',
  `fee` float DEFAULT '0' COMMENT '停车费',
  `output_date` datetime DEFAULT NULL COMMENT '出场时间',
  `status` tinyint(1) DEFAULT '1' COMMENT '状态:1-入场，0-出场',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- 样例数据

INSERT INTO plate.t_parking (park_id,license_plate,input_date,fee,output_date,status) VALUES
```

```
(1,'鲁 V88888','2024-06-07 10:04:00',0.0,NULL,1),
(1,'鲁 G99999','2024-06-07 10:05:00',0.0,NULL,1),
(1,'皖 AD01846','2024-06-07 17:02:38',0.0,NULL,1),
(1,'冀 E87G70','2024-06-11 08:24:18',3.0,'2024-06-11 09:07:58',0),
(1,'京 A00461','2024-06-11 09:07:05',0.0,NULL,1),
(1,'冀 E87G70','2024-06-11 09:09:11',0.0,'2024-06-11 09:11:44',0),
(1,'冀 E87G70','2024-06-11 09:16:44',27.0,'2024-06-11 17:55:41',0);
```

1.4 系统结构设计



1.5 界面原型设计

1.6 详细设计与实现

1.6.1 系统依赖库（requirements.txt）

```
pyqt5
opencv-python
pymysql
hyperlpr3
pillow
chardet
ultralytics
paddlepaddle
paddleocr
shapely
numpy
```

1.6.2 系统配置、共享、入口

1.系统配置（config.py）

```
# 停车场编号
PARKING_ID = 1

# 主窗口设置信息
MAIN_WINDOW_TITLE = "智能停车场"
MAIN_WINDOW_WIDTH = 1080
MAIN_WINDOW_HEIGHT = 720

# 数据库连接信息
HOST = '172.23.218.93'
PORT = 3306
USER = 'plate'
PASSWORD = '@135qetQET'
DATABASE = 'plate'
CHARSET = 'utf8mb4'
```

2.系统共享（share.py）

```
import pymysql
import config


class ShareInfo:
    # 系统窗口面板
    main_panel = None
    login_panel = None
    register_panel = None
```

```
    # 数据库连接
    connection = pymysql.connect(
        host=config.HOST,
        port=config.PORT,
        user=config.USER,
        password=config.PASSWORD,
        database=config.DATABASE,
        charset=config.CHARSET
    )
```

3.系统入口（main.py）

```
import sys
from PyQt5 import QtWidgets
from login import LoginWindow
from share import ShareInfo

app = QtWidgets.QApplication(sys.argv)

ShareInfo.login_panel = LoginWindow()
ShareInfo.login_panel.show()

sys.exit(app.exec_())
```

1.6.3 系统登录与注册

1.系统登录（login.py）

```
import config
from window import WindowPanel
from PyQt5.QtWidgets import (QWidget, QLineEdit, QPushButton, QLabel,
```

```
QVBoxLayout, QHBoxLayout, QGridLayout, QMessageBox)
from register import RegisterWindow
from share import ShareInfo



class LoginWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        user_label = QLabel('用户名:')
        self.user_input = QLineEdit()
        pass_label = QLabel('密码:')
        self.pass_input = QLineEdit()
        self.pass_input.setEchoMode(QLineEdit.Password)

        login_button = QPushButton('登录')
        register_button = QPushButton('注册')

        grid_layout = QGridLayout()
        grid_layout.addWidget(user_label, 0, 0)
        grid_layout.addWidget(self.user_input, 0, 1)
        grid_layout.addWidget(pass_label, 1, 0)
        grid_layout.addWidget(self.pass_input, 1, 1)

        hbox_layout = QHBoxLayout()
        hbox_layout.addWidget(login_button)
        hbox_layout.addWidget(register_button)
```

```
        vbox_layout = QVBoxLayout()
        vbox_layout.addLayout(grid_layout)
        vbox_layout.addLayout(hbox_layout)

        self.setLayout(vbox_layout)

        self.setWindowTitle('登录')

        login_button.clicked.connect(self.login)
        register_button.clicked.connect(self.register)

    def login(self):
        username = self.user_input.text()
        password = self.pass_input.text()

        with ShareInfo.connection.cursor() as cursor:
            sql = "SELECT * FROM t_user WHERE username = %s AND password
= %s"
            cursor.execute(sql, (username, password))
            result = cursor.fetchone()
            if result:
                QMessageBox.warning(self, '登录成功', '用户名和密码正确，成功登
录')
                ShareInfo.main_panel = WindowPanel()
                ShareInfo.main_panel.setWindowTitle("智能停车场")

ShareInfo.main_panel.setMinimumSize(config.MAIN_WINDOW_WIDTH,
config.MAIN_WINDOW_HEIGHT)
                ShareInfo.main_panel.show()
                self.hide()
```

```
            else:
                QMessageBox.warning(self, '登录失败', '用户名或密码错误')
                self.user_input.setText('')
                self.pass_input.setText('')

    def register(self):
        self.hide()
        ShareInfo.register_panel = RegisterWindow()
        ShareInfo.register_panel.show()
```

2.系统注册（register.py）

```
from PyQt5.QtWidgets import QMessageBox, QWidget, QLineEdit, QPushButton, QVBoxLayout, QLabel, QGridLayout, QHBoxLayout
from share import ShareInfo
import pymysql



class RegisterWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle('注册')

        username_label = QLabel('用户名:')
        self.username_input = QLineEdit()
        password_label = QLabel('密码:')
        self.password_input = QLineEdit()
        self.password_input.setEchoMode(QLineEdit.Password)
```

```
        confirm_password_label = QLabel('确认密码:')
        self.confirm_password_input = QLineEdit()
        self.confirm_password_input.setEchoMode(QLineEdit.Password)

        register_button = QPushButton('注册')

        grid_layout = QGridLayout()
        grid_layout.addWidget(username_label, 0, 0)
        grid_layout.addWidget(self.username_input, 0, 1)
        grid_layout.addWidget(password_label, 1, 0)
        grid_layout.addWidget(self.password_input, 1, 1)
        grid_layout.addWidget(confirm_password_label, 2, 0)
        grid_layout.addWidget(self.confirm_password_input, 2, 1)

        hbox_layout = QHBoxLayout()
        hbox_layout.addWidget(register_button)

        vbox_layout = QVBoxLayout()
        vbox_layout.addLayout(grid_layout)
        vbox_layout.addLayout(hbox_layout)

        self.setLayout(vbox_layout)

        register_button.clicked.connect(self.register)

    def register(self):
        username = self.username_input.text()
        password = self.password_input.text()
        confirm_password = self.confirm_password_input.text()
```

```
            if password == confirm_password:
                try:
                    with ShareInfo.connection.cursor() as cursor:
                        sql = "INSERT INTO t_user(username, password) VALUES (%s, %s)"
                        cursor.execute(sql, (username, password))
                    ShareInfo.connection.commit()
                    QMessageBox.warning(self, '注册成功', f'用户名: {username}, 密码: {password} - 注册成功')
                    ShareInfo.login_panel.show()
                    self.close()
                except pymysql.Error as e:
                    QMessageBox.warning(self, '注册失败', f'注册失败: {e}')
            else:
                QMessageBox.warning(self, '注册失败', '密码和确认密码不一致')
                self.password_input.setText('')
                self.confirm_password_input.setText('')
```

1.6.4 车辆视频、抓拍、识别及相关信息显示

1.车辆识别-本地第三方库（lpr3.py）

```
import cv2
import warnings
import numpy as np
from PIL import ImageFont
from PIL import Image
from PIL import ImageDraw
import hyperlpr3 as lpr3

warnings.filterwarnings("ignore", message="Mean of empty slice")
```

```
warnings.filterwarnings("ignore", message="invalid value encountered in scalar divide")
catcher = lpr3.LicensePlateCatcher(detect_level=lpr3.DETECT_LEVEL_HIGH)
font_ch = ImageFont.truetype("platech.ttf", 20, 0)



def draw_plate_on_image(img, box1, text1, font):
    x1, y1, x2, y2 = box1
    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2, cv2.LINE_AA)
    data = Image.fromarray(img)
    draw = ImageDraw.Draw(data)
    draw.text((x1, y1 - 27), text1, (0, 0, 255), font=font)
    res = np.asarray(data)
    return res



def license_recognition_image(path):
    image = cv2.imread(path)
    results = catcher(image)
    code_image_list = []
    for code, confidence, type_idx, box in results:
        text = f"{code} - {confidence:.2f}"
        image = draw_plate_on_image(image, box, text, font=font_ch)
        code_image_list.append((code, image))
    return code_image_list



if __name__ == "__main__":
    file_pic = r"frame.jpg"
    code_image_list = license_recognition_image(file_pic)
```

```
    print(code_image_list[0][0])
    cv2.imshow("License Plate Recognition(Picture)", code_image_list[0][1])
    cv2.waitKey(0)
```

2.车辆识别-在线百度 API（baidu.py）

```
from aip.ocr import AipOcr


from PIL import Image, ImageDraw, ImageFont
import numpy as np
import cv2


APP_ID = '28222257'
API_KEY = 'YGMghwNb5VMfA5GAMC8NRY5E'
SECRET_KEY = 'LgzjKj75E3TlcjCwpXEEwrT1P2yboSc4'


client = AipOcr(APP_ID, API_KEY, SECRET_KEY)
client.setConnectionTimeoutInMillis(5000)
client.setSocketTimeoutInMillis(5000)



def cv2ImgAddText(img, text, left, top, textColor, textSize):
    if (isinstance(img, np.ndarray)):
        img = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    draw = ImageDraw.Draw(img)
    fontStyle = ImageFont.truetype("simsun.ttc", textSize, encoding="utf-8")
    draw.text((left, top), text, textColor, font=fontStyle)
    return cv2.cvtColor(np.asarray(img), cv2.COLOR_RGB2BGR)
```

```
def getFileContent(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()



def license_recognition_image(image_path):
    image_content = getFileContent(image_path)
    res = client.licensePlate(image_content)

    code_image_list = []
    if res is not None:
        car_number = res['words_result']['number']
        car_color = res['words_result']['color']
        print('车牌号码：' + car_number)
        print('车牌颜色：' + car_color)

        location = res['words_result']['vertexes_location']
        start_x = location[0]['x']
        start_y = location[0]['y']
        end_x = location[2]['x']
        end_y = location[2]['y']

        img = cv2.imread(image_path)
        cv2.rectangle(img, (start_x, start_y), (end_x, end_y), (0, 0, 255), 5)
        txt = car_number + ' ' + car_color
        img_txt = cv2ImgAddText(img, txt, start_x, start_y - 30, (0, 255, 0), 30)

        code_image_list.append((car_number, img_txt))
    else:
```

```
            print('车牌识别失败！')


        return code_image_list



if __name__ == '__main__':
    imgPath = 'frame.jpg'
    code_image_list = license_recognition_image(imgPath)
    print(code_image_list[0][0])
    cv2.imshow('img', code_image_list[0][1])

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

3.车辆识别-自训练模型（YOLOv8+PaddleOCR）

（1）数据配置（data.yaml）

```
train: images\train   # train images (relative to 'path') 128 images
val: images\val   # val images (relative to 'path') 128 images
test: images\test # val images (optional)

# number of classes
nc: 1

# Classes
names: ['LicensePlate']
```

（2）数据处理（datasets.py）

```
import cv2
import os
```

```
def txt_translate(path, txt_path):
    print(path)
    print(txt_path)
    for filename in os.listdir(path):
        list1 = filename.split("-", 3)
        subname = list1[2]
        list2 = filename.split(".", 1)
        subname1 = list2[1]
        if subname1 == 'txt':
            continue
        lt, rb = subname.split("_", 1)
        lx, ly = lt.split("&", 1)
        rx, ry = rb.split("&", 1)
        width = int(rx) - int(lx)
        height = int(ry) - int(ly)
        cx = float(lx) + width / 2
        cy = float(ly) + height / 2

        img = cv2.imread(path + filename)
        if img is None:
            print(path + filename)
            os.remove(path + filename)
            continue
        width = width / img.shape[1]
        height = height / img.shape[0]
        cx = cx / img.shape[1]
        cy = cy / img.shape[0]
```

```
            txtname = filename.split(".", 1)
            txtfile = txt_path + txtname[0] + ".txt"
            with open(txtfile, "w") as f:
                f.write(str(0) + " " + str(cx) + " " + str(cy) + " " + str(width) + " " +
str(height))



    if __name__ == '__main__':
        # det 图片存储地址
        trainDir = r"datasets/PlateData/images/train/"
        validDir = r"datasets/PlateData/images/val/"
        testDir = r"datasets/PlateData/images/test/"
        # det txt 存储地址
        train_txt_path = r"datasets/PlateData/labels/train/"
        val_txt_path = r"datasets/PlateData/labels/val/"
        test_txt_path = r"datasets/PlateData/labels/test/"
        txt_translate(trainDir, train_txt_path)
        txt_translate(validDir, val_txt_path)
        txt_translate(testDir, test_txt_path)
```

（3）模型训练（train.py）

```
    #coding:utf-8
    from ultralytics import YOLO
    # 加载预训练模型
    model = YOLO("yolov8n.pt")
    # Use the model
    if __name__ == '__main__':
        # Use the model
        results = model.train(data='datasets/PlateData/data.yaml', epochs=10,
```

```
batch=2)  #  训练模型
        #  将模型转为 onnx 格式
        success = model.export(format='onnx')
```

（4）车辆识别-自训练模型（yolo_paddleocr.py）

```
# coding:utf-8
import os
import numpy as np
import cv2
from ultralytics import YOLO
from paddleocr import PaddleOCR
from PIL import Image, ImageDraw, ImageFont

os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"



def img_cvread(path):
    img = cv2.imdecode(np.fromfile(path, dtype=np.uint8), cv2.IMREAD_COLOR)
    return img



def drawRectBox(image, rect, addText, fontC, color=(0, 0, 255)):
    cv2.rectangle(image, (rect[0], rect[1]),
                  (rect[2], rect[3]),
                  color, 2)

    font_size = int((rect[3] - rect[1]) / 1.5)
    img = Image.fromarray(image)
    draw = ImageDraw.Draw(img)
```

```
        draw.text((rect[0] + 2, rect[1] - font_size), addText, (0, 0, 255), font=fontC)
        imagex = np.array(img)
        return imagex



    def get_license_result(ocr, image):
        result = ocr.ocr(image, cls=True)[0]
        if result:
            license_name, conf = result[0][1]
            if ' • ' in license_name:
                license_name = license_name.replace(' • ', '')
            return license_name, conf
        else:
            return None, None



    def license_recognition_image(img_path):
        code_image_list = []
        now_img = img_cvread(img_path)
        fontC = ImageFont.truetype("platech.ttf", 50, 0)

        cls_model_dir = 'PaddleOCR/cls_infer'
        rec_model_dir = 'PaddleOCR/rec_infer'
        ocr = PaddleOCR(use_angle_cls=False, lang="ch", det=False,
cls_model_dir=cls_model_dir, rec_model_dir=rec_model_dir)

        path = 'runs/detect/train/weights/best.pt'
        model = YOLO(path, task='detect')

        results = model(img_path)[0]
```

```
        location_list = results.boxes.xyxy.tolist()
        if len(location_list) >= 1:
            location_list = [list(map(int, e)) for e in location_list]
            # 截取每个车牌区域的照片
            license_imgs = []
            for each in location_list:
                x1, y1, x2, y2 = each
                cropImg = now_img[y1:y2, x1:x2]
                license_imgs.append(cropImg)
            # 车牌识别结果
            lisence_res = []
            conf_list = []
            for each in license_imgs:
                license_num, conf = get_license_result(ocr, each)
                if license_num:
                    lisence_res.append(license_num)
                    conf_list.append(conf)
                else:
                    lisence_res.append('无法识别')
                    conf_list.append(0)

            for text, box in zip(lisence_res, location_list):
                now_img = drawRectBox(now_img, box, text, fontC)
                code_image_list.append((text, now_img))
        return code_image_list



if __name__ == '__main__':
    img_path = "frame.jpg"
```

```
        code_image_list = license_recognition_image(img_path)
        print(code_image_list[0][0])
        cv2.imshow("Recognition", code_image_list[0][1])
        cv2.waitKey(0)
```

4.车辆视频、抓拍、识别及相关信息显示（windows.py）

```
    from datetime import datetime
    import pymysql
    import cv2
    from PyQt5 import QtWidgets
    from PyQt5 import QtCore
    from PyQt5 import QtGui
    import config
    from share import ShareInfo
    import lpr3
    import baidu



    class WindowPanel(QtWidgets.QWidget):
        def __init__(self, parent=None):
            super().__init__(parent)

            self.timer = QtCore.QTimer(self)

            self.CAMERA_NUM = 0
            self.video_capture = cv2.VideoCapture()
            flag = self.video_capture.open(self.CAMERA_NUM)
            if flag is False:
                QtWidgets.QMessageBox.information(self, "警告", "摄像头【0】未正常
连接!", QtWidgets.QMessageBox.Ok)
```

```
        else:
            self.timer.start()
        self.frame = None
        self.capture_filename = "frame.jpg"
        self.now_lot_number = -1
        self.unit_price = -1

        self.video_image_label = QtWidgets.QLabel()
        self.capture_image_label = QtWidgets.QLabel()

        current_time_label = QtWidgets.QLabel('当前时间:')
        self.current_time_edit = QtWidgets.QLineEdit()

        remaining_parking_label = QtWidgets.QLabel('剩余车位数:')
        self.remaining_parking_edit = QtWidgets.QLineEdit()

        access_info_label = QtWidgets.QLabel('出入场信息:')
        self.access_info_edit = QtWidgets.QLineEdit()

        parking_cars_label = QtWidgets.QLabel('停车场车辆:')
        self.parking_cars_browser = QtWidgets.QTextBrowser()

        top_left_vertical_layout = QtWidgets.QVBoxLayout()
        top_left_vertical_layout.addWidget(self.video_image_label)
        top_left_vertical_layout.addWidget(self.capture_image_label)

        top_right_grid_layout = QtWidgets.QGridLayout()
        top_right_grid_layout.addWidget(current_time_label, 0, 0)
        top_right_grid_layout.addWidget(self.current_time_edit, 0, 1)
```

```
top_right_grid_layout.addWidget(remaining_parking_label, 1, 0)
top_right_grid_layout.addWidget(self.remaining_parking_edit, 1, 1)
top_right_grid_layout.addWidget(access_info_label, 2, 0)
top_right_grid_layout.addWidget(self.access_info_edit, 2, 1)
top_right_grid_layout.addWidget(parking_cars_label, 3, 0)
top_right_grid_layout.addWidget(self.parking_cars_browser, 3, 1)

top_left_panel = QtWidgets.QWidget()
top_left_panel.setLayout(top_left_vertical_layout)

top_right_panel = QtWidgets.QWidget()
top_right_panel.setLayout(top_right_grid_layout)

display_panel = QtWidgets.QWidget()
top_horizontal_layout = QtWidgets.QHBoxLayout()
top_horizontal_layout.addWidget(top_left_panel)
top_horizontal_layout.addWidget(top_right_panel)
display_panel.setLayout(top_horizontal_layout)

self.hyperlpr3_radio_button = QtWidgets.QRadioButton("HyperLPR3")
self.baidu_radio_button = QtWidgets.QRadioButton("百度云车牌识别 API")
self.train_radio_button = QtWidgets.QRadioButton("YOLOv8+PaddleOcr")

middle_horizontal_layout = QtWidgets.QHBoxLayout()
middle_horizontal_layout.addWidget(self.hyperlpr3_radio_button)
middle_horizontal_layout.addWidget(self.baidu_radio_button)
middle_horizontal_layout.addWidget(self.train_radio_button)

radio_button_panel = QtWidgets.QGroupBox("车牌识别方式")
```

```
radio_button_panel.setLayout(middle_horizontal_layout)

capture_button = QtWidgets.QPushButton("抓拍")
recognise_button = QtWidgets.QPushButton("识别")

bottom_horizontal_layout = QtWidgets.QHBoxLayout()
bottom_horizontal_layout.addWidget(capture_button)
bottom_horizontal_layout.addWidget(recognise_button)

button_panel = QtWidgets.QWidget()
button_panel.setLayout(bottom_horizontal_layout)

whole_vertical_layout = QtWidgets.QVBoxLayout()
whole_vertical_layout.addWidget(display_panel)
whole_vertical_layout.addWidget(radio_button_panel)
whole_vertical_layout.addWidget(button_panel)

self.setLayout(whole_vertical_layout)

self.hyperlpr3_radio_button.setChecked(True)

self.timer.timeout.connect(self.show_image)
self.timer.timeout.connect(self.show_datetime)
self.timer.timeout.connect(self.show_lot_number)
self.timer.timeout.connect(self.show_lot_cars)

capture_button.clicked.connect(self.on_capture_image)
recognise_button.clicked.connect(self.on_recognise_image)
```

```
        def __del__(self):
            self.video_capture.release()

        def on_recognise_image(self):
            license_image_list = None
            if self.hyperlpr3_radio_button.isChecked():
                license_image_list =
lpr3.license_recognition_image(self.capture_filename)
            if self.baidu_radio_button.isChecked():
                license_image_list =
baidu.license_recognition_image(self.capture_filename)
            if self.train_radio_button.isChecked():
                license_image_list =
lpr3.license_recognition_image(self.capture_filename)

            if len(license_image_list) == 0:
                return

            current_frame_pixmap_scaled =
self.frame_2_pixmap(license_image_list[0][1])

self.capture_image_label.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
            self.capture_image_label.setPixmap(current_frame_pixmap_scaled)

            plate_license = license_image_list[0][0]
            if plate_license == '无法识别':
                return
            now_datetime = self.get_now_datetime()
            now_lot_number = self.now_lot_number
            try:
                with ShareInfo.connection.cursor() as cursor:
```

```
                select_data = (config.PARKING_ID, plate_license)
                sql_select = '''SELECT license_plate, input_date
                                FROM plate.t_parking
                                WHERE park_id=%s and license_plate=%s and status=1;'''
                cursor.execute(sql_select, select_data)
                results = cursor.fetchall()
                if results:
                    input_datetime = results[0][1]
                    fee = self.calculate_parking_fee(input_datetime, datetime.now(), self.unit_price,
                                                     first_hour_free=False)

                    self.access_info_edit.setText(" ".join([plate_license, now_datetime, "出场", str(fee) + "元"]))
                    now_lot_number += 1

                    update_parking_data = (fee, now_datetime, 0, plate_license)
                    sql_update_parking = '''UPDATE plate.t_parking
                                SET fee=%s, output_date=%s, status=%s
                                WHERE status=1 and license_plate=%s;'''
                    cursor.execute(sql_update_parking, update_parking_data)
                else:
                    self.access_info_edit.setText(" ".join([plate_license, now_datetime, "入场"]))
                    now_lot_number -= 1

                    insert_parking_data = (config.PARKING_ID, plate_license, now_datetime, 0, None, 1)
                    sql_insert_parking = '''INSERT INTO
                                plate.t_parking(park_id, license_plate,
```

```
input_date, fee, output_date, status)
                                        VALUES(%s, %s, %s, %s, %s, %s);'''
                        cursor.execute(sql_insert_parking, insert_parking_data)


                    update_parking_lot_data = (now_lot_number,
config.PARKING_ID,)
                    sql_update_parking_lot = '''UPDATE plate.t_parking_lot
                                SET lot_number=%s
                                WHERE id=%s;'''
                    cursor.execute(sql_update_parking_lot, update_parking_lot_data)


                    self.access_info_edit.setReadOnly(True)


                    ShareInfo.connection.commit()
            except pymysql.MySQLError as e:
                ShareInfo.connection.rollback()
                print(f"Error: {e}")
                QtWidgets.QMessageBox.information(self, "警告", f"车辆
【{plate_license}】出入场失败!",
                                                QtWidgets.QMessageBox.Ok)


        def on_capture_image(self):
            cv2.imwrite(self.capture_filename, self.frame)
            frame = cv2.imread(self.capture_filename)
            current_frame_pixmap_scaled = self.frame_2_pixmap(frame)

self.capture_image_label.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
            self.capture_image_label.setPixmap(current_frame_pixmap_scaled)


        def show_lot_cars(self):
```

```
            try:
                with ShareInfo.connection.cursor() as cursor:
                    sql_data = (config.PARKING_ID,)
                    sql_select = '''SELECT license_plate, input_date
                                    FROM plate.t_parking
                                    WHERE park_id=%s and status=1;'''
                    cursor.execute(sql_select, sql_data)
                    results = cursor.fetchall()
                    self.parking_cars_browser.clear()
                    for index in range(len(results)):
                        self.parking_cars_browser.append(" | ".join(
                            [str(index + 1), results[index][0], results[index][1].strftime("%Y-%m-%d %H:%M:%S")]))
                        self.parking_cars_browser.setReadOnly(True)
            except pymysql.MySQLError as e:
                print(f"Error: {e}")
                QtWidgets.QMessageBox.information(self, "警告", "获取【停车场车辆】失败!", QtWidgets.QMessageBox.Ok)


        def show_lot_number(self):
            try:
                with ShareInfo.connection.cursor() as cursor:
                    sql_data = (config.PARKING_ID,)
                    sql_select = '''SELECT id, lot_number, unit_price
                                    FROM plate.t_parking_lot
                                    WHERE id=%s;'''
                    cursor.execute(sql_select, sql_data)
                    results = cursor.fetchall()
                    if len(results) == 1 and len(results[0]) == 3:
                        self.now_lot_number = results[0][1]
```

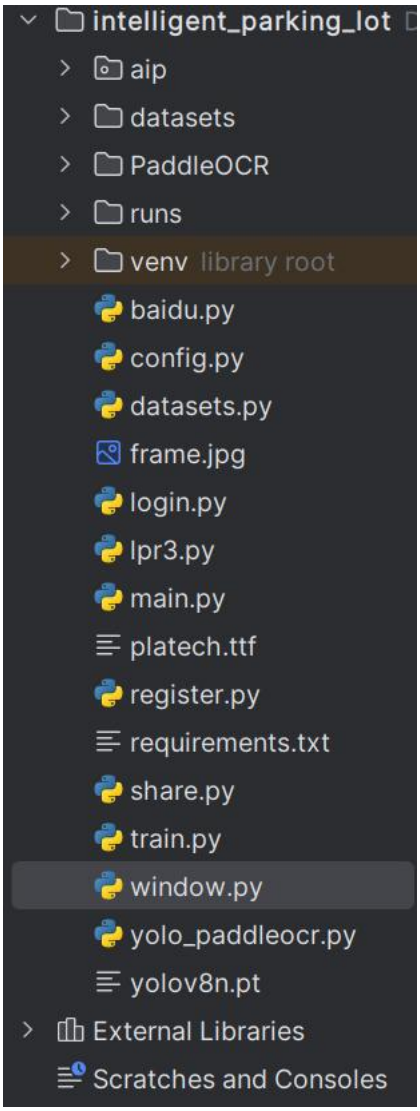
```
                self.unit_price = results[0][2]
                self.remaining_parking_edit.setText(str(self.now_lot_number) + "个")
                self.remaining_parking_edit.setReadOnly(True)
        except pymysql.MySQLError as e:
            print(f"Error: {e}")
            QtWidgets.QMessageBox.information(self, "警告", "获取【剩余车位数】失败!", QtWidgets.QMessageBox.Ok)

    def show_datetime(self):
        self.current_time_edit.setText(self.get_now_datetime())
        self.current_time_edit.setReadOnly(True)

    def show_image(self):
        success, frame = self.video_capture.read()
        if success:
            self.frame = frame
            current_frame_pixmap_scaled = self.frame_2_pixmap(frame)
            self.video_image_label.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
            self.video_image_label.setPixmap(current_frame_pixmap_scaled)

    @staticmethod
    def frame_2_pixmap(frame):
        current_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_height, frame_width, frame_channel = current_frame.shape
        current_frame_image = QtGui.QImage(current_frame, frame_width, frame_height, 3 * frame_width,
                                           QtGui.QImage.Format.Format_RGB888)
        current_frame_pixmap = QtGui.QPixmap(current_frame_image)
```

```
        current_frame_pixmap_scaled = current_frame_pixmap.scaled(QtCore.QSize(frame_height, frame_width),

QtCore.Qt.AspectRatioMode.KeepAspectRatio)
        return current_frame_pixmap_scaled


    @staticmethod
    def get_now_datetime():
        now_datetime = datetime.now()
        now_datetime_formatted = now_datetime.strftime('%Y-%m-%d %H:%M:%S')
        return now_datetime_formatted


    @staticmethod
    def calculate_parking_fee(start_time, end_time, base_rate, first_hour_free=False):
        """
        计算停车费用。
        参数:
        - start_time: 开始时间（datetime 对象）
        - end_time: 结束时间（datetime 对象）
        - base_rate: 每小时的基础费率（float）
        - first_hour_free: 是否第一个小时免费（bool），默认为 True
        返回:
        - 停车费用（float）
        """
        if first_hour_free:
            base_rate_first_hour = 0
            base_rate_additional_hours = base_rate
        else:
            base_rate_first_hour = base_rate
```

```
            base_rate_additional_hours = base_rate


        # 计算时间差，并转换为小时
        duration = (end_time - start_time).total_seconds() / 3600
        hours = duration // 1
        additional_minutes = (duration % 1) * 60


        # 计算费用
        fee = (base_rate_first_hour if hours > 0 else 0) + \
            (base_rate_additional_hours * (hours - 1) if hours > 1 else 0) + \
            (additional_minutes // 30 * base_rate_additional_hours if additional_minutes > 0 else 0)


        return fee
```

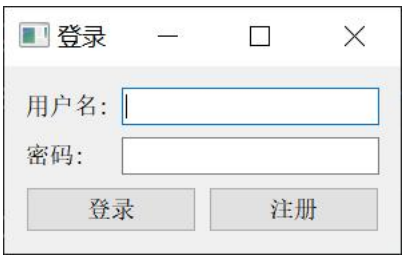
1.6.5 系统目录结构与实现效果

系统目录结构如下：


intelligent_parking_lot
aip
datasets
PaddleOCR
runs
venv library root
baidu.py
config.py
datasets.py
frame.jpg
login.py
lpr3.py
main.py
platech.ttf
register.py
requirements.txt
share.py
train.py
window.py
yolo_paddleocr.py
yolov8n.pt
External Libraries
Scratches and Consoles

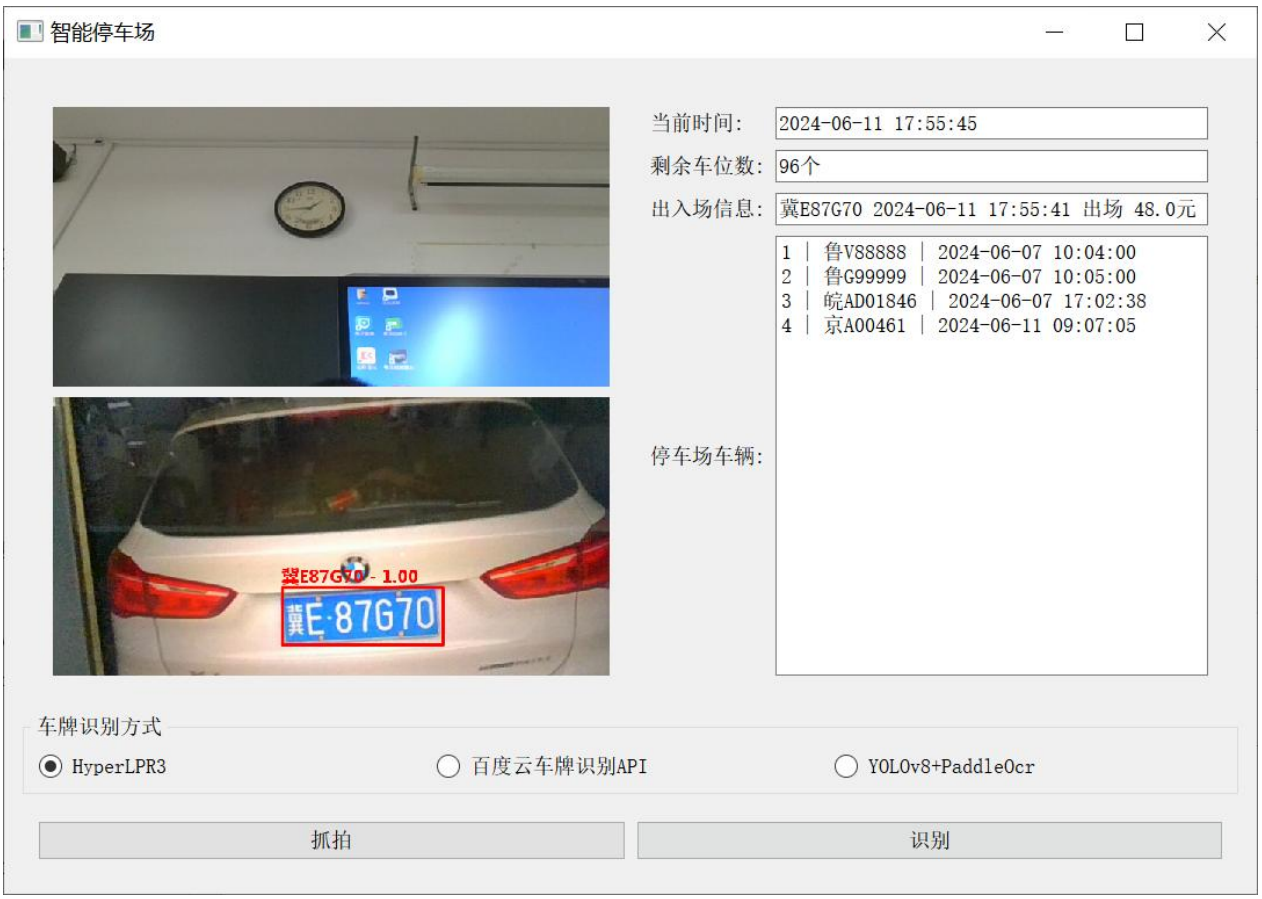
系统实现效果展示如下：

注册
用户名:
密码:
确认密码:
注册


登录
用户名:
密码:
登录
注册

智能停车场
当前时间:
2024-06-11 17:54:30
剩余车位数:
95个
出入场信息:
1 | 鲁V88888 | 2024-06-07 10:04:00
2 | 鲁G99999 | 2024-06-07 10:05:00
3 | 皖AD01846 | 2024-06-07 17:02:38
4 | 京A00461 | 2024-06-11 09:07:05
5 | 冀E87G70 | 2024-06-11 09:16:44
停车场车辆:
车牌识别方式
HyperLPR3
百度云车牌识别API
YOLOv8+PaddleOcr
抓拍
识别

智能停车场
当前时间:
2024-06-11 17:55:28
剩余车位数:
95个
出入场信息:
1 | 鲁V88888 | 2024-06-07 10:04:00
2 | 鲁G99999 | 2024-06-07 10:05:00
3 | 皖AD01846 | 2024-06-07 17:02:38
4 | 京A00461 | 2024-06-11 09:07:05
5 | 冀E87G70 | 2024-06-11 09:16:44
停车场车辆:
冀E·87G70
车牌识别方式
HyperLPR3
百度云车牌识别API
YOLOv8+PaddleOcr
抓拍
识别



1.7 扩展功能

可以参考如下资料，添加 PaddleOCR 训练：

https://blog.csdn.net/m0_70694811/article/details/138872422

1. 管理员可以对停车费进行按天、月、年等周期统计，并画柱状图进行展示。
2. 管理员可以对车辆的历史停车信息进行查询。
3. 管理员可以修改停车单价（每小时停车费用）。
4. 管理员可以修改停车场车位总数。