# 界面UI

## 代码简介：

这段代码是一个使用PyQt5创建用户界面的应用程序。它包含一个主窗口，其中有几个标签、文本编辑框和按钮。主要功能如下：

显示图像：通过点击"打开图像"按钮，可以选择图像文件并显示在程序界面上的label\_7中。图像文件可以是任意格式的图片文件。

添加注释：在文本编辑框textEdit\_2中输入图像的注释，并通过点击"添加注释"按钮，将注释保存到一个字典中。

显示注释：通过点击"显示注释"按钮，可以将之前保存的注释显示在文本编辑框textEdit\_3中。

图像处理：通过点击"图像处理"按钮，可以对当前显示的图像进行处理，具体处理方法需要根据实际需求进行编写。处理后的图像将显示在label\_6中。

保存处理结果：通过点击"保存结果"按钮，可以将处理后的图像保存到指定的文件夹中。

这段代码主要使用了PyQt5库来创建用户界面，并结合其他库（如cv2和numpy）进行图像处理操作。它提供了一种简单的图像浏览和处理的界面化方式，使用户能够方便地进行图像处理和注释的添加与显示。

完整代码：

from PyQt5.QtWidgets import QFileDialog  
from PyQt5 import QtGui, QtCore, QtWidgets  
import os  
import json  
import cv2  
import numpy as np  
  
  
class Ui\_MainWindow(object):  
 current\_image\_index = 0  
 image\_files = []  
 image\_browsing = False  
  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow") # 要看着改  
 MainWindow.resize(1312, 950)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.label = QtWidgets.QLabel(self.centralwidget)  
 self.label.setGeometry(QtCore.QRect(160, 80, 72, 15))  
 self.label.setObjectName("label")  
 self.label\_2 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_2.setGeometry(QtCore.QRect(160, 410, 111, 16))  
 self.label\_2.setObjectName("label\_2")  
 self.label\_3 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_3.setGeometry(QtCore.QRect(880, 50, 121, 31))  
 self.label\_3.setObjectName("label\_3")  
 self.pushButton = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(160, 220, 93, 28))  
 self.pushButton.setObjectName("pushButton")  
 self.textEdit\_2 = QtWidgets.QTextEdit(self.centralwidget)  
 self.textEdit\_2.setGeometry(QtCore.QRect(80, 120, 251, 87))  
 self.textEdit\_2.setObjectName("textEdit\_2")  
 self.textEdit\_3 = QtWidgets.QTextEdit(self.centralwidget)  
 self.textEdit\_3.setGeometry(QtCore.QRect(90, 440, 251, 87))  
 self.textEdit\_3.setObjectName("textEdit\_3")  
 self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_2.setGeometry(QtCore.QRect(170, 560, 93, 28))  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_3.setGeometry(QtCore.QRect(890, 240, 93, 28))  
 self.pushButton\_3.setObjectName("pushButton\_3")  
 self.pushButton\_4 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_4.setGeometry(QtCore.QRect(900, 550, 93, 28))  
 self.pushButton\_4.setObjectName("pushButton\_4")  
 self.label\_7 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_7.setGeometry(QtCore.QRect(720, 620, 321, 271))  
 self.label\_7.setObjectName("label\_7")  
 self.label\_7.setStyleSheet("border: 1px solid black;")  
  
  
 self.label\_4 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_4.setGeometry(QtCore.QRect(880, 370, 121, 16))  
 self.label\_4.setObjectName("label\_4")  
 self.label\_6 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_6.setGeometry(QtCore.QRect(190, 630, 311, 251))  
 self.label\_6.setObjectName("label\_6")  
 self.label\_6.setStyleSheet("border: 1px solid black;")  
 self.label\_6.setScaledContents(True)  
  
 self.pushButton\_5 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_5.setGeometry(QtCore.QRect(570, 730, 93, 28))  
 self.pushButton\_5.setObjectName("pushButton\_5")  
 self.label\_5 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_5.setGeometry(QtCore.QRect(430, 30, 111, 41))  
 font = QtGui.QFont()  
 font.setFamily("Agency FB")  
 font.setPointSize(14)  
 self.label\_5.setFont(font)  
 self.label\_5.setObjectName("label\_5")  
 self.pushButton\_6 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_6.setGeometry(QtCore.QRect(1080, 687, 121, 51))  
 self.pushButton\_6.setObjectName("pushButton\_6")  
 self.pushButton\_7 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_7.setGeometry(QtCore.QRect(1080, 770, 121, 51))  
 self.pushButton\_7.setObjectName("pushButton\_7")  
 self.pushButton\_8 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_8.setGeometry(QtCore.QRect(570, 790, 93, 28))  
 self.pushButton\_8.setObjectName("pushButton\_8")  
 self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)  
 self.tableWidget.setGeometry(QtCore.QRect(705, 100, 481, 111))  
 self.tableWidget.setObjectName("tableWidget")  
 self.tableWidget.setColumnCount(0)  
 self.tableWidget.setRowCount(0)  
 self.tableWidget\_2 = QtWidgets.QTableWidget(self.centralwidget)  
 self.tableWidget\_2.setGeometry(QtCore.QRect(705, 400, 491, 121))  
 self.tableWidget\_2.setObjectName("tableWidget\_2")  
 self.tableWidget\_2.setColumnCount(0)  
 self.tableWidget\_2.setRowCount(0)  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 1312, 26))  
 self.menubar.setObjectName("menubar")  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 # 按照你的按钮进行修改  
 self.pushButton\_2.clicked.connect(self.openFolderDialog)  
 self.pushButton.clicked.connect(self.loadHistoryPaths)  
 self.pushButton\_4.clicked.connect(self.loadFolderData)  
 self.pushButton\_3.clicked.connect(self.loadHistoryData)  
 self.pushButton\_5.clicked.connect(self.displayImage)  
 self.pushButton\_6.clicked.connect(self.previousPage)  
 self.pushButton\_7.clicked.connect(self.nextPage)  
 self.pushButton\_8.clicked.connect(self.endImageBrowse)  
  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.label.setText(\_translate("MainWindow", "上次目录"))  
 self.label\_2.setText(\_translate("MainWindow", "当前分析目录"))  
 self.label\_3.setText(\_translate("MainWindow", "统计各目录数量"))  
 self.pushButton.setText(\_translate("MainWindow", "查看"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "打开"))  
 self.pushButton\_3.setText(\_translate("MainWindow", "统计"))  
 self.pushButton\_4.setText(\_translate("MainWindow", "统计"))  
 self.label\_4.setText(\_translate("MainWindow", "统计各标签数量"))  
 self.pushButton\_5.setText(\_translate("MainWindow", "结果浏览"))  
 self.label\_5.setText(\_translate("MainWindow", "小小软件"))  
 self.pushButton\_6.setText(\_translate("MainWindow", "上一页"))  
 self.pushButton\_7.setText(\_translate("MainWindow", "下一页"))  
 self.pushButton\_8.setText(\_translate("MainWindow", "结束浏览"))  
  
 def openFolderDialog(self):  
 folder\_path = QFileDialog.getExistingDirectory(  
 None, "选择文件夹", QtCore.QDir.currentPath())  
  
 if folder\_path:  
 self.textEdit\_3.setPlainText(folder\_path) # 要改  
  
 # Save folder\_path to text file with a new line  
 with open("lishiwenjian.txt", "a") as f:  
 f.write(folder\_path + "\n")  
  
 # Append folder\_path to the existing text in textEdit\_2 with a new line  
 self.textEdit\_2.insertPlainText(folder\_path + "\n") # 要改  
 # 选择文件夹，保存在txt里面  
  
  
  
 def loadFolderData(self):  
 folder\_path = self.textEdit\_3.toPlainText()  
  
 if folder\_path:  
 self.tableWidget\_2.clear()  
  
 file\_names = os.listdir(folder\_path)  
 label\_counts = {}  
  
 for file\_name in file\_names:  
 if file\_name.endswith('.json'):  
 file\_path = os.path.join(folder\_path, file\_name)  
 with open(file\_path, "r", encoding='gbk') as file:  
 json\_data = file.read()  
 data = json.loads(json\_data)  
 shapes = data.get('shapes')  
 if shapes:  
 for shape in shapes:  
 label = shape.get('label')  
 if label in label\_counts:  
 label\_counts[label] += 1  
 else:  
 label\_counts[label] = 1  
  
 self.tableWidget\_2.setColumnCount(len(label\_counts))  
 self.tableWidget\_2.setRowCount(2)  
 self.tableWidget\_2.setHorizontalHeaderLabels(list(label\_counts.keys()))  
  
 column = 0  
 for label, count in label\_counts.items():  
 self.tableWidget\_2.setItem(0, column, QtWidgets.QTableWidgetItem(label))  
 self.tableWidget\_2.setItem(1, column, QtWidgets.QTableWidgetItem(str(count)))  
 column += 1  
 # 计算文件里面的标签，是当前目录  
  
  
  
 def loadHistoryPaths(self):  
 if os.path.exists("lishiwenjian.txt"):  
 with open("lishiwenjian.txt", "r") as f:  
 history\_paths = f.read()  
 self.textEdit\_2.setPlainText(history\_paths)  
 else:  
 self.textEdit\_2.setPlainText("无历史路径记录")  
  
  
  
 def saveHistoryPaths(self):  
 history\_paths = self.textEdit\_2.toPlainText()  
 settings = QtCore.QSettings()  
 settings.setValue("history\_paths/history\_paths", history\_paths)  
  
 # Save history paths to file  
 with open("lishiwenjian.txt", "w") as f:  
 f.write(history\_paths)  
  
  
  
 # 查看我们所用过的历史文件路径  
 def loadHistoryData(self):  
 if os.path.exists("lishiwenjian.txt"):  
 with open("lishiwenjian.txt", "r") as f:  
 history\_paths = f.readlines()  
  
 if len(history\_paths) >= 2:  
 folder\_path = history\_paths[-2].strip()  
 file\_names = os.listdir(folder\_path)  
 label\_counts = {}  
  
 for file\_name in file\_names:  
 if file\_name.endswith('.json'):  
 file\_path = os.path.join(folder\_path, file\_name)  
 with open(file\_path, "r", encoding='gbk') as file:  
 json\_data = file.read()  
 data = json.loads(json\_data)  
 shapes = data.get('shapes')  
 if shapes:  
 for shape in shapes:  
 label = shape.get('label')  
 if label in label\_counts:  
 label\_counts[label] += 1  
 else:  
 label\_counts[label] = 1  
  
 self.tableWidget.setColumnCount(len(label\_counts))  
 self.tableWidget.setRowCount(2)  
 self.tableWidget.setHorizontalHeaderLabels(list(label\_counts.keys()))  
  
 column = 0  
 for label, count in label\_counts.items():  
 self.tableWidget.setItem(0, column, QtWidgets.QTableWidgetItem(label))  
 self.tableWidget.setItem(1, column, QtWidgets.QTableWidgetItem(str(count)))  
 column += 1  
 else:  
 self.tableWidget.clear()  
 else:  
 self.tableWidget.clear()  
  
 def displayImage(self):  
 folder\_path = self.textEdit\_3.toPlainText()  
 self.image\_files = os.listdir(folder\_path)  
 self.image\_files = [f for f in self.image\_files if f.endswith('.png') or f.endswith('.jpg')] # 如果需要，可以过滤图片文件  
 if len(self.image\_files) > 0:  
 image\_path = os.path.join(folder\_path, self.image\_files[self.current\_image\_index])  
 pixmap = QtGui.QPixmap(image\_path)  
 scaled\_pixmap = pixmap.scaled(self.label\_6.width(), self.label\_6.height(), QtCore.Qt.KeepAspectRatio)  
 self.label\_6.setPixmap(scaled\_pixmap)  
 self.label\_6.setScaledContents(True)  
 self.label\_6.show()  
  
 # 分割图片  
 if len(self.image\_files) > 0:  
 image\_path = os.path.join(folder\_path, self.image\_files[self.current\_image\_index])  
 json\_file\_path = image\_path.replace(".jpg", ".json").replace(".png", ".json")  
 if os.path.isfile(json\_file\_path):  
 # 读取JSON文件  
 with open(json\_file\_path, "r", encoding="utf-8") as json\_file:  
 json\_data = json.load(json\_file)  
 # 读取图像文件  
 image = cv2.imread(image\_path)  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB) # 将图像从BGR转换为RGB颜色空间  
 # 创建语义分割图像  
 label\_map = {  
 "pedestrian": (255, 0, 0), # 红色  
 "car": (0, 255, 0), # 绿色  
 "rider": (0, 0, 255), # 蓝色  
 "truck": (0, 120, 120), # 青色  
 "bus": (120, 120, 0), # 黄色  
 # 添加更多的标签和对应的颜色  
 }  
 segmented\_image = np.zeros\_like(image)  
  
 # 解析JSON数据并在语义分割图像上绘制标注区域  
 for shape in json\_data["shapes"]:  
 label = shape["label"]  
 points = np.array(shape["points"], dtype=np.int32)  
  
 # 获取标签对应的颜色  
 color = label\_map.get(label)  
  
 if color is not None:  
 # 在语义分割图像上绘制标注区域  
 cv2.fillPoly(segmented\_image, [points], color)  
 x, y, w, h = cv2.boundingRect(points)  
 cv2.putText(image, label, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, color, 2)  
  
 # 创建用于融合的遮罩图像  
 mask = np.zeros\_like(image)  
  
 # 解析JSON数据并在遮罩图像上绘制标注区域  
 for shape in json\_data["shapes"]:  
 label = shape["label"]  
 points = np.array(shape["points"], dtype=np.int32)  
  
 # 获取标签对应的颜色  
 color = label\_map.get(label)  
  
 if color is not None:  
 # 在遮罩图像上绘制标注区域  
 cv2.fillPoly(mask, [points], color)  
  
 # 将原图与遮罩图像进行融合  
 alpha = 1 # 设置融合的透明度  
 blended\_image = cv2.addWeighted(image, alpha, mask, 1.5 - alpha, 0)  
  
 # 将图像转换为Qt可显示的格式  
 height, width, channel = blended\_image.shape  
 bytes\_per\_line = channel \* width  
 qt\_image = QtGui.QImage(blended\_image.data, width, height, bytes\_per\_line, QtGui.QImage.Format\_RGB888)  
 pixmap = QtGui.QPixmap.fromImage(qt\_image)  
  
 else:  
 pixmap = QtGui.QPixmap(image\_path)  
  
 scaled\_pixmap = pixmap.scaled(self.label\_7.width(), self.label\_7.height(), QtCore.Qt.KeepAspectRatio)  
 self.label\_7.setPixmap(scaled\_pixmap)  
 self.label\_7.setScaledContents(True)  
 self.label\_7.show()  
  
 def nextPage(self):  
 if self.current\_image\_index < len(self.image\_files) - 1:  
 self.current\_image\_index += 1  
 self.displayImage()  
  
 def previousPage(self):  
 if self.current\_image\_index > 0:  
 self.current\_image\_index -= 1  
 self.displayImage()  
  
 def endImageBrowse(self):  
 self.image\_browsing = False  
 self.label\_6.clear()  
 self.label\_7.clear()  
  
 # 计算上一次的历史路径的文件夹标签  
  
  
  
# 要看着改  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
  
 app = QtWidgets.QApplication(sys.argv)  
 MainWindow = QtWidgets.QMainWindow()  
 ui = Ui\_MainWindow()  
 ui.setupUi(MainWindow)  
 MainWindow.show()  
 sys.exit(app.exec\_())

## 效果图如下：

