

合肥工业大学

计算机与信息学院

Python 程序设计与应用大作业

题 目	:	场景文字图像标注小工具
专 业	:	电子信息工程
学 号	:	2018212085
姓 名	:	陈 银
授课老师	:	胡珍珍

2021 年 5 月 25 日

目录

1 任务描述.....	2
2 实验实施步骤.....	2
2.1 设计方案.....	2
2.2 开发环境	3
2.3 开发流程	3
2.3.1 设计 GUI 界面	3
2.3.2 代码编写	4
2.4 软件主要使用步骤	7
3 遇到的问题和解决办法.....	10
4 系统性能评估.....	10
4.1 测试打开单个文件	11
4.2 标记功能测试	11
5 实验总结.....	13
6 课程学习感想.....	13

场景文字图像标注小工具

1 任务描述

要求：具有 GUI 界面的图像标注工具

功能：从本地文件夹中加载依次加载多张包含场景文字的图片，用户用鼠标标出图片中场景文字所在位置，并可输入包含的文字字符。返回边界框坐标值和字符内容并保存，结果保存为 JSON 格式文件，包含图片名称、文字坐标、文字字符。一张图片可标记多个文字区域

2 实验实施步骤

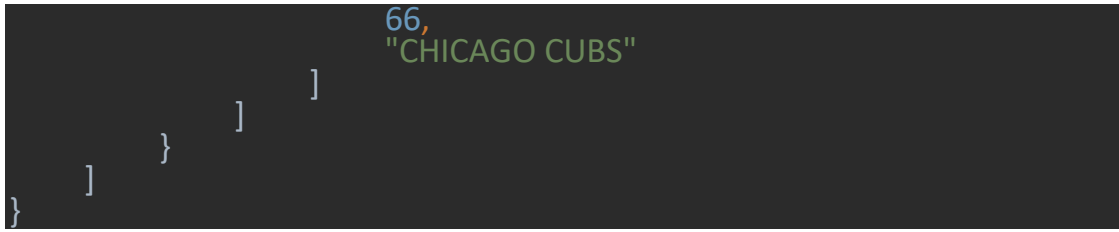
2.1 设计方案

1、采用 PyQt5 这个开源库作为 GUI 编写库。

2、数据格式

数据采用 JSON 格式保存，数据保存格式如下：

```
{
  "name": "json-data",
  "version": "1.0",
  "data": [
    {
      "image_name": "4e1db83fffdc5adf.jpg",
      "image_width": 768,
      "image_height": 1024,
      "object": [
        [
          44,
          383,
          309,
          176,
          "Cuvee des Fleurs"
        ]
      ]
    },
    {
      "image_name": "16dc9c57eeb1b8f8.jpg",
      "image_width": 1024,
      "image_height": 686,
      "object": [
        [
          213,
          386,
          553,
          101,
          "WELCOMETO WAIGLEY FIELD"
        ],
        [
          195,
          314,
          579,
          579,
          "WELCOMETO WAIGLEY FIELD"
        ]
      ]
    }
  ]
}
```



image_name 保存图片名称

image_width , image_height 保存图片大小

每张图片的一个文字区域称为一个 object, 每一个 object 包含文字的坐标和内容。坐标用 (x, y, width, height) 表示, 紧接着的字符串表示文字的内容。

2.2 开发环境

OS: Ubuntu 20.04.2 LTS x86_64

PyCharm Professional Edition

Python 3.8.5

QtDesigner

2.3 开发流程

2.3.1 设计 GUI 界面

1) 打开 QtDesigner 新建一个 MainWindow 文件

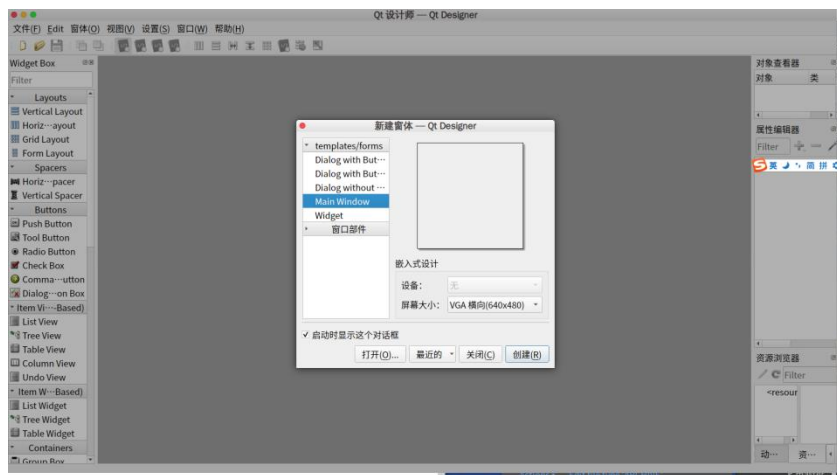


图 2-1 新建文件

2) GUI 初步设计草图

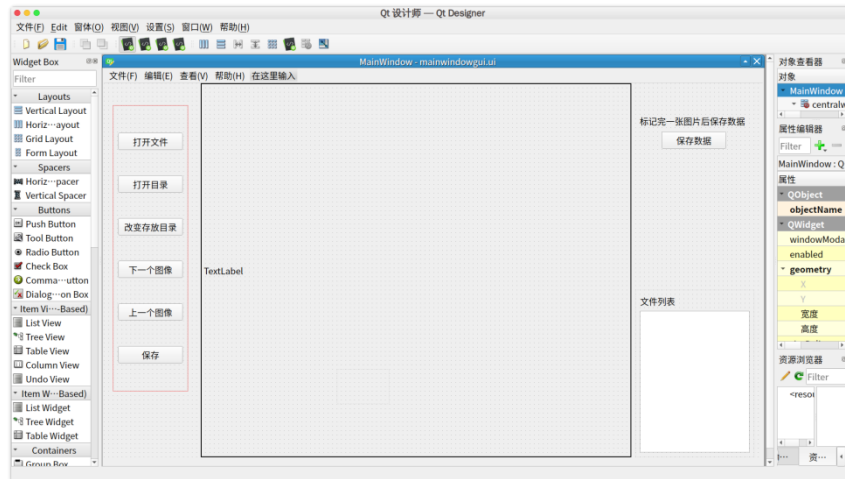


图 2-2 窗口样式和布局

- 3) PyUIC 将 ui 文件转为 Python 文件
`pyuic5 -o ui.py mainwindowgui.ui`
- 4) python 调用显示界面



图 2-3 运行 GUI 界面

2.3.2 代码编写

因为 PyQt5 的事件触发是使用信号与槽函数绑定的, 所以需要先为触发事件编写槽函数然后对槽函数进行绑定。

- 1) 编写槽函数, 实现触发事件

```
def openFile(self):
    dig = QFileDialog()
    filenames = dig.getOpenFileName(self, '选择图片', './', 'Image files (*.jpg *.png *.jpeg)')
    if filenames[0] == '':
        msg_box = QMessageBox(QMessageBox.Warning, 'Warning', '没有正确选择文件!')
        msg_box.exec_()
        return
    else:
        self.showImg(filenames[0])
```

```

def openDirs(self):
    fileDlg = QFileDialog()
    fileDlg.setFileMode(QFileDialog.DirectoryOnly)
    fileDlg.setOption(QFileDialog.DontUseNativeDialog, True)
    fileDlg.setDirectory("./data")
    listView = fileDlg.findChild(QListView, "listView")
    if listView:
        listView.setSelectionMode(QAbstractItemView.ExtendedSelection)
    treeView = fileDlg.findChild(QTreeView, "treeView")
    if treeView:
        treeView.setSelectionMode(QAbstractItemView.ExtendedSelection)
    if fileDlg.exec_():
        folders = fileDlg.selectedFiles()
        if len(folders) < 1:
            return
        # self.savePath = folders[0]
        for file in os.listdir(folders[0]):
            file = folders[0] + "/" + file
            self.filelists.append(file)
            self.plainTextEdit_fileLists.appendPlainText(file)
    if len(self.filelists) > 0:
        self.showImg(self.filelists[0])

def showImg(self, filename):
    if filename.split(".")[1] not in self.imgs:
        msg_box = QMessageBox(QMessageBox.Warning, 'Warning', '图片格式只能是常见格式！')
    )
    msg_box.exec_()
    return
    img = Image.open(filename)
    self.imgSize = img.size
    size = self.label_display.size()
    size = (size.width(), size.height())
    img = img.resize(size)
    img = img.toqixmap()
    self.label_display2.setPixmap(img)

def nextImg(self):
    if len(self.filelists) < 1:
        return
    if self.index == len(self.filelists) - 1:
        self.pushButton_nextImg.setEnabled(False)
        return
    else:
        self.pushButton_lastImg.setEnabled(True)
        self.index += 1
        self.showImg(self.filelists[self.index])

def lastImg(self):
    if len(self.filelists) < 1:
        return
    if self.index == 0:
        self.pushButton_lastImg.setEnabled(False)
        return
    else:
        self.pushButton_nextImg.setEnabled(True)
        self.index -= 1
        self.showImg(self.filelists[self.index])

```

```
def chooseSavePath(self):
    fileDlg = QFileDialog()
    fileDlg.setFileMode(QFileDialog.DirectoryOnly)
    fileDlg.setOption(QFileDialog.DontUseNativeDialog, True)
    fileDlg.setDirectory("./data")
    listView = fileDlg.findChild(QListView, "listView")
    if listView:
        listView.setSelectionMode(QAbstractItemView.ExtendedSelection)
    treeView = fileDlg.findChild(QTreeView, "treeView")
    if treeView:
        treeView.setSelectionMode(QAbstractItemView.ExtendedSelection)
    if fileDlg.exec_():
        folders = fileDlg.selectedFiles()
        if len(folders) < 1:
            return
        # self.savePath = folders[0]
    print("当前存放路径", self.savePath)

def saveData(self):
    name = self.filelists[self.index].split("/")[-1]
    data = {"image_name": name, "image_width": self.imgSize[0], "image_height": self.imgSize[1],
           "object": self.object_data}
    self.data.append(data)
    print(self.data)
    self.nextImg()

def saveFile(self):
    with open(self.savePath + "/data.json", "w") as f:
        data = {"name": "json-data", "version": "1.0", "data": self.data}
        # json.dumps(data, f)
        f.write(json.dumps(data, indent=4))
        f.close()
    print(self.savePath + "/data.json")
    print("数据保存成功")

# 处理返回的数据
def processData(self, data):
    # print(self.filelists[self.index], self.imgSize)
    # print(*data[0], data[1])
    scale_x = self.imgSize[0] / 981
    scale_y = self.imgSize[1] / 851
    x, y, w, h = data[0][0:4]
    x = scale_x * x
    y = scale_y * y
    w = scale_x * w
    h = scale_y * h
    # print(*data[0], x, y, w, h, scale_x, scale_y)
    self.object_data.append([int(x), int(y), int(w), int(h), data[1]])
```

2) 为每个按钮绑定触发事件，PyQt5 使用信号与槽进行绑定。

```
self.pushButton_openFile.clicked.connect(self.openFile)
self.pushButton_openDir.clicked.connect(self.openDirs)
self.pushButton_nextImg.clicked.connect(self.nextImg)
self.pushButton_lastImg.clicked.connect(self.lastImg)
self.pushButton_changeSavePath.clicked.connect(self.chooseSavePath)
```



```
self.pushButton_Save.clicked.connect(self.saveFile)
self.label_display.sendDataSig.connect(self.processData)
self.pushButton_Save_data.clicked.connect(self.saveData)
```

2.4 软件主要使用步骤

1) 安装软件依赖

```
pip install -r requirements.txt
```

2) python labeltool.py 运行软件



图 2-4 运行窗口

从运行界面可以看到整个界面除了最上面的标题栏以外分为三个区域，最左侧的为按钮区域，包括各种操作按钮。中间为图片显示区域，实时显示正在操作的图片，在图片上进行标注。最右侧为保存按钮和文件显示列表，用于保存数据和显示文件名称。下面详细介绍使用步骤。

3) 从按钮栏名称可以看出，我们可以分别打开单独的图片 and 打开一个包含图片的目录。点击打开目录，选择图片所在目录。

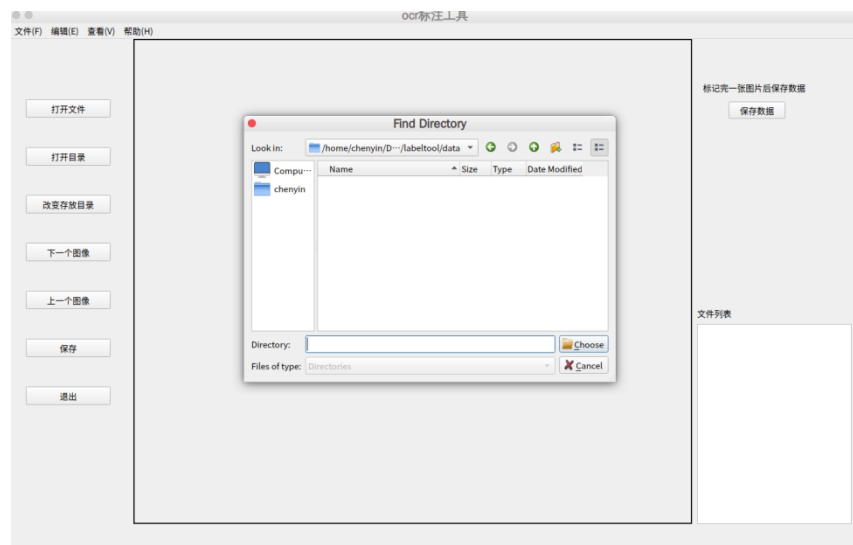


图 2-5 选择目录



界面会显示当前正在操作的图片和当前目录下的文件。

- 3) 使用鼠标拖拽，选中文字区域、在文字框中输入文字后按回车



图 2-6 标注

- 5) 标记完一张图片后单击保存数据按钮，保存数据



图 2-7 保存数据

5) 标记完所有图片以后点击保存按钮、JSON 数据文件会自动保存在当前目录下的 data/data.json 文件中。

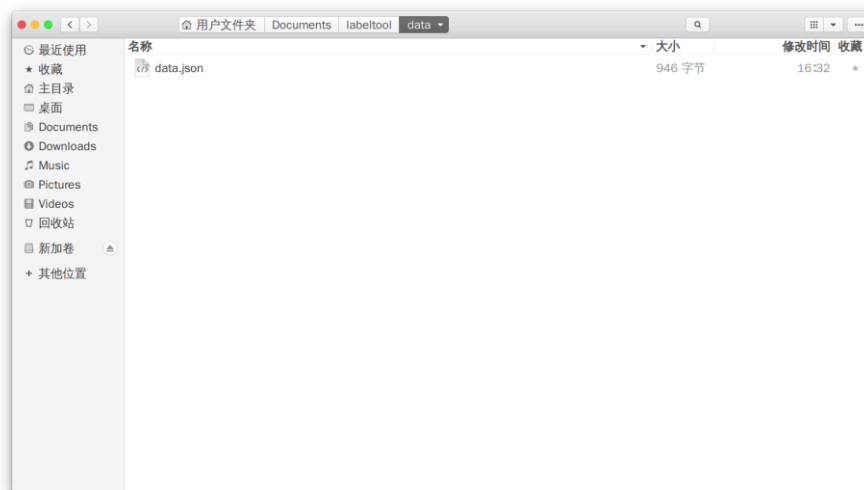




图 2-8 JSON 文件展示

3 遇到的问题和解决办法

问题 1: GUI 库的选择。市面上 python GUI 库有十几种、各有千秋，一时不知道作何选择。

解决方法: 经过查阅资料和多方比较，结合自己使用 qt c++做过项目的经历，选择了 pyqt5 这个开源库。在开发过程中发现，有过 c++使用的经验，pyqt5 上手很快。

问题 2: 界面的开发非常顺利，但是当要编写真正的逻辑操作，实现选中图片文字区域这个业务时，遇到了棘手的问题。由于 pyqt5 显示图片是直接在 label 标签上画出来的，但是我又想在 label 上框出来我选中的文字区域，者势必导致了会重新绘制图片，这样原来的图片就会被覆盖，只显示一个矩形框了。

解决办法: 经过查阅资料和多次尝试、自己重写了 QLabel 这个标签(详见 MyLabel.py 文件)。使用 PyQt5 自带的 QLabel 来显示图片，然后在我自己写的 MyLabel 上画矩形框。只需要将 MyLabel 的透明度改了，就可已既显示图片，画矩形框，又不会互相影响了。

问题 3: Segmentation fault 错误。今天做测试的时候莫名奇妙的出现了这个错误。百度都没法解决。

解决办法: 通过 git log 查看日志，发现是我使用的 pillow 库更新了，与之前的代码不兼容。重新安装了 pillow==7.2.0 尺，解决。

4 系统性能评估

由于本 label 实现的功能并没有非常复杂、并不吃性能。故只对系统的功能进行测试。

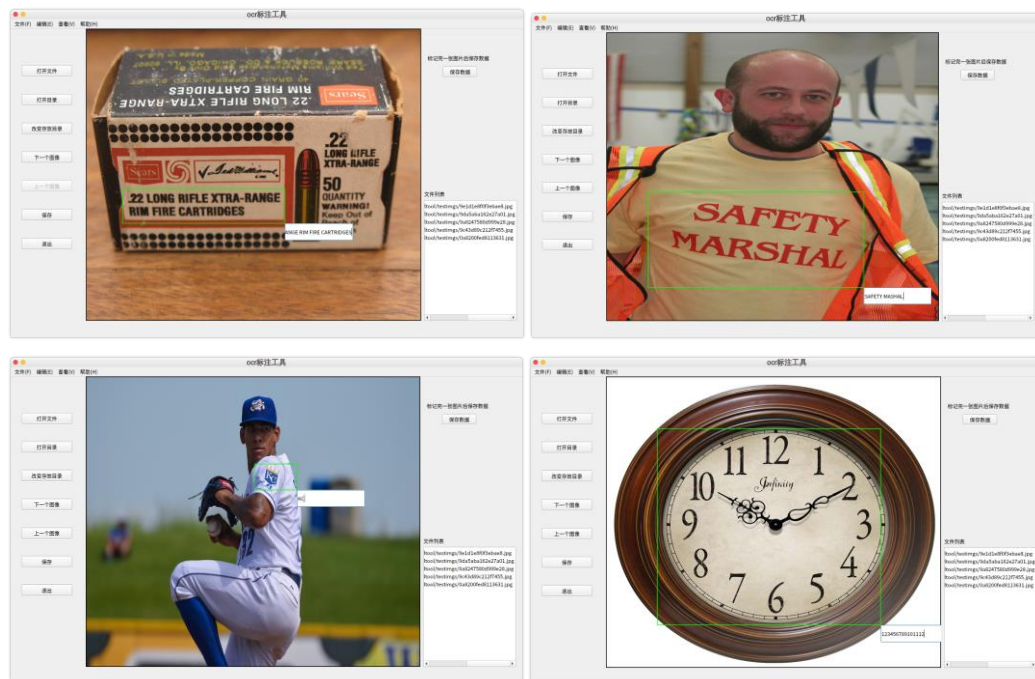
4.1 测试打开单个文件



没有问题

4.2 标记功能测试

选择 4 张图片作为数据集。分别对这五张图片进行标记



结果如下：

```
{
  "name": "json-data",
  "version": "1.0",
  "data": [
    {
      "image_name": "9e1d1e8f0f3ebae8.jpg",
      "image_width": 1024,
```

```

"image_height": 768,
"object": [
  [
    111,
    420,
    495,
    91,
    "22 LONG RIFLE XTRA-RANGE RIM FIRE CARTRIDGES"
  ]
],
},
{
  "image_name": "9da5aba162e27a01.jpg",
  "image_width": 683,
  "image_height": 1024,
  "object": [
    [
      115,
      536,
      406,
      351,
      "SAFETY MARSHAL"
    ]
  ]
},
{
  "image_name": "9c43d89c212f7455.jpg",
  "image_width": 1024,
  "image_height": 1024,
  "object": [
    [
      193,
      196,
      686,
      697,
      "12345678101112"
    ]
  ]
},
{
  "image_name": "9a8247580d999e28.jpg",
  "image_width": 1024,
  "image_height": 678,
  "object": [
    [
      486,
      203,
      141,
      74,
      "KC"
    ]
  ]
}
]
}

```

可以看到数据集中的图片名称、大小、文字区域的位置、文字内容都保存了下来。

经过以上的分析、本系统已经完成了所有的任务要求。

5 实验总结

经过一个多星期的实践和学习，终于把这个文字图像标注的工具的基本功能完善了。现在已经能够对一张图片文字所在的区域进行标注、也可以对一张图片的多个位置进行标注。

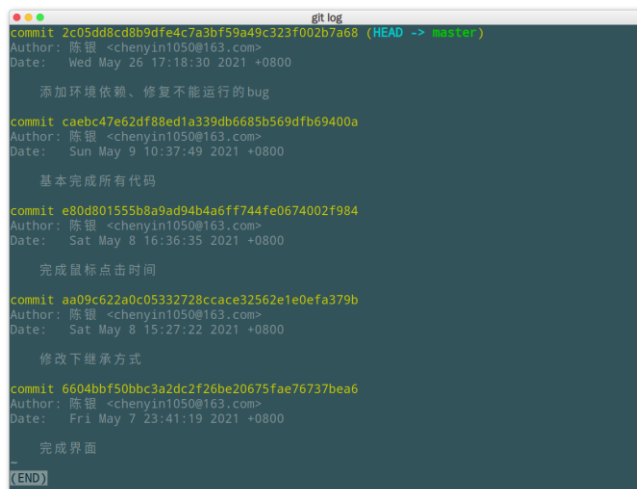
本次大作业、让我熟练的掌握了在 Linux 系统下使用 PyQt5 进行 GUI 界面的编写，也学会了如何在 Linux 快速的搭建一个实验环境。实验过程中也用到了 JSON、pillow 库，以及 git 工具，让我收获很多

6 课程学习感想

经过一个学期的 python 学习，系统的接触了 python 这个高级的语言。总的来说就是 python 非常高级，api 丰富，很多功能不需自己从零实现，着非常有益于我们做简单的开发。另一方面，本次课程我还接触了爬虫，感觉这是一个非常有趣的技术手段，可以在网上爬我们想要的信息，感觉很好玩。

本次也是第一次使用 python 完整的写一个 GUI 软件，也算得上一个小项目了。之前使用过 c++, java 等语言开发 GUI 界面、但是都没有使用 Python 方便。Python 下开源库众多，许多想要实现的功能都已经有了一个完整的库，这对我们快速的开发一个软件是非常方便的。

还有一个感想就是在写代码中的版本控制是非常重要的的一环。有时候代码改着改着就不知道改到哪里去了，但是在使用了 git 这个版本控制工具以后就可以方便的退回到自己想要的那个版本、这对我们开发是非常有帮助的。下图就是我本次代码的版本记录：



```
git log
commit 2c05dd8cd8b9dfe4c7a3bf59a49c323f002b7a68 (HEAD -> master)
Author: 陈银 <chenyin1050@163.com>
Date: Wed May 26 17:18:30 2021 +0800
    添加环境依赖、修复不能运行的bug

commit caebc47e62df88ed1a339db6685b569dfb69400a
Author: 陈银 <chenyin1050@163.com>
Date: Sun May 9 10:37:49 2021 +0800
    基本完成所有代码

commit e80d801555b8a9ad94b4a6ff744fe0674002f984
Author: 陈银 <chenyin1050@163.com>
Date: Sat May 8 16:36:35 2021 +0800
    完成鼠标点击时间

commit aa09c622a0c05332728ccace32562e1e0efa379b
Author: 陈银 <chenyin1050@163.com>
Date: Sat May 8 15:27:22 2021 +0800
    修改下继承方式

commit 6604bbf50bbc3a2dc2f26be20675fae76737bea6
Author: 陈银 <chenyin1050@163.com>
Date: Fri May 7 23:41:19 2021 +0800
    完成界面

(END)
```

图 4-1 git 记录