利用pyQt5搭建一个简易的图片编辑器（PhotoEdit）

姓名：胡成成 学号：41724260

北京科技大学计通学院 通信1701班

|  |
| --- |
| 摘要：在本报告中介绍了一套简易的图片编辑器软件系统的需求分析、设计、实现和测试。首先对于整个项目的基本模块，算法，框架，UI界面需求进行分析，根据需求引入相关技术。其次开始软件系统的整体设计，从布局，功能和代码层次进行分析设计。最后进行代码实现和功能测试，完成整个软件系统的正常使用。在这里主要利用Python的图形化界面搭建，运用图像处理技术对现有原图进行操作。 |
| 关键字：GUI图形化界面；图像处理；pyQt5；滤镜。 |

# 1. 项目背景和意义

## 1.1 项目背景简介

伴随着python课程的结束，最吸引我兴趣的便是python图形化界面的搭建与图像处理的部分。鉴于课上学习的pyQt4和图像处理的PIL库，通过类比学习了最新版pyQt5，通过对这方面的学习，能够综合到GUI界面的搭建，图像处理等方法，于是选择完成一个简单的图像编辑器，精简的修图小程序，来实践python的图形化理论学习。就这样，这个小项目就诞生了。

## 1.2 项目完成意义

通过搭建简易的图片编辑器主要是对学习的Qt的图形化操作熟悉，掌握基本的图像处理本领。同时，增强了自己的资料搜集能力，并判别资料中的自己需要的部分，以及工程开发的基本流程，测试的一些基本方法等等。

# 2. 需求分析

## 2.1 基本模块需求

针对一块基本的图形处理软件，包括了基本的文档打开存储，图形界面的搭建，图形处理，日志记录等。于是需要下载相关模块，包括pyQt5，pillow等。

## 2.2 基本算法需求

根据任意一个图像处理软件，必然会用到图像处理的相关函数，有些函数也就需要了解它的基本算法，比如滤镜的添加，便需要对图片的像素点进行处理，调用相关函数，已经一些调整函数，例如亮度，对比度，锐化这些图片调整的方式必然也有它的处理算法。于是需要查看相关文献，搜集资料来获取这些算法的原理和使用方式。

## 2.3 基本框架需求

类比正常的软件开发规范，工程文件创建的格式规范化，学习基本的开发要点，注释的规范，面向对象的开发方式，是否需要调用大框架等等。

## 2.4 基本UI需求

为了使UI界面更美观，需要一些图片挑选，字体选择规划，布局规划设计等。

# 3. 概要和详细设计

## 3.1 设计概要

围绕图像编辑处理这一大功能，在设计时主要考虑到一下几点工作方向：

1. 设计主界面窗口的布局：采用编辑主窗口+处理选项窗口设计。
2. 分析要加入的基本功能：主要包括滤镜，调整，尺寸，旋转四个大方面进行功能细化。
   * 滤镜：无滤镜原图，黑白滤波，负滤波，均值滤波。
   * 调整：对比度，亮度，锐化。
   * 尺寸：修改宽度，修改高度，比例修改。
   * 旋转：顺时针和逆时针90度旋转，垂直和水平翻转。
3. 分析功能并进行资料搜集处理：细化处理事项。
4. 分析需要写的代码的层次与功能：将代码按功能分类分块，是整个文件具有层次感。
5. 开始撰写代码：根据不同功能分别细化处理。
6. UI布局优化和细节调整。
7. 程序调试与修改。
8. 总结注意事项，并对比结果与预期的差距。

## 3.2 设计流程

## 3.3 技术引用

1. 滤镜相关函数算法：（下面我们将原图RGB三色值分别记为R，G，B，处理后图片RGB三色值记为r，g，b）。
   * 黑白滤波：
     + 将像素点R，G，B先加权平均并记为s
     + 取k值30
     + 新的RGB：***r=s+k\*2；g=s+k；b=s***
   * 负滤波：
     + 新的RGB：***r=255-R;g=255-G;b=255-B***
   * 均值滤波：
     + 将像素点R，G，B先加权平均并记为s
     + 新的RGB：***r=s；g=s；b=s***
2. 调整相关函数算法：（MAX参数限定最大值，MIN参数限定最小值，F\_MAX系数增强最大值，F\_MIN系数增强最小值，X给定参数值）
   * 计算比例系数r：***r=（X-MIN）/（MAX-MIN）***
   * 得到增强系数：***F\_MIN+r\*（MAX-MIN）***
   * 将增强系数赋值给相应的调整参数

# 4. 代码实现

## 4.1 Python版本与IDE说明

Python版本：Python 3.6

IDE：Pycharm

## 4.2 相关库调用

PyQt5库：PyQt5是基于图形程序框架Qt5的Python语言实现，由一组Python模块构成。需要借助该模块完成基本界面的搭建，主要包括QtWidgets，QtCore，QtGui组件的基本使用，完成部件布局和事件的基本处理。

PIL库：python的第三方图像处理库，支持python2版本，python3版本后移植到pillow库，这里的PIL引用就是最新的pillow库。借助该库完成基本的图像处理操作。

Sys库：提供对解释器使用或维护的一些变量的访问，以及与解释器强烈交互的函数。主要保持python主程序文件正常运行。

Ntpath库：处理基本的文件路径问题，扩展了Windows平台上提供os.path功能，可以在其他平台上处理Windows路径。

Functools库：主要是为函数式编程而设计，用于增强函数功能。这里主要用于高阶函数。

Logging库：日志模块，主要是用来记录我们想要的信息，以便于后期调试。

fileConfig库：用来读取配置文件报错解决方法。

Getopt库：专门用来处理命令行参数。

## 4.3 文件分块说明

1. 创建工程文件夹：命名为photoEdit。
2. 在工程文件夹下新建img\_modifier文件夹存放\_\_init\_\_.py，color\_filter.py，img\_helper.py文件。

* \_\_init\_\_.py：存放配置文件相关代码
* color\_filter.py：存放滤镜相关函数
* img\_helper.py：存放图片处理相关函数和参数

1. 工程文件下新建img\_modifier.py:存放初始化操作函数。
2. 工程文件下新建photo\_editor.py:存放布局代码，主程序代码，和基本事件函数和参数。
3. 工程文件夹下添加UI布局图片文件和日志文件logging\_config.ini。

## 4.4 关键代码说明

### 4.4.1 滤镜文件代码说明

1. 文件定位：img\_modifier文件夹下color\_filter.py文件。
2. 文件库导入：

import logging

1. 定义彩色滤镜类ColorFilters：存放滤镜键值对便于调用查询。

class ColorFilters:

filters = {"sepia": "Sepia", "negative": "Negative", "black\_white": "Black & White"}

SEPIA, NEGATIVE, BLACK\_WHITE = filters.keys()

1. 滤镜函数定义：这里我们主要引用三种简单的滤镜，主要原理是通过载入图片获取整个图片像素信息，对像素的RGB值进行重新计算赋值得到的新图片。对于这一过程，由于一张大图片的像素数量庞大，计算的时候耗时比较长，可能需要几秒钟时间等待计算。

#黑白滤镜

def sepia(img):

pix = img.load()

for i in range(img.width):

for j in range(img.height):

s = sum(pix[i, j]) // 3

k = 30

pix[i, j] = (s+k\*2, s+k, s)

#均值滤波

def black\_white(img):

pix = img.load()

for i in range(img.width):

for j in range(img.height):

s = sum(pix[i, j]) // 3

pix[i, j] = (s, s, s)

#负滤镜，底片滤镜

def negative(img):

pix = img.load()

for i in range(img.width):

for j in range(img.height):

pix[i, j] = (255 - pix[i, j][0], 255 - pix[i, j][1], 255 - pix[i, j][2])

1. 滤镜图片复制效果函数color\_filter(img, filter\_name)：用于多样式预览对比框展示，img为导入的图片，filter\_name为滤镜的键KEY（上面定义滤镜类中提到）。同时添加日志记录，记录报错信息，以便调试。

def color\_filter(img, filter\_name):

img\_copy = img.copy()

if filter\_name == ColorFilters.SEPIA:

sepia(img\_copy)

elif filter\_name == ColorFilters.NEGATIVE:

negative(img\_copy)

elif filter\_name == ColorFilters.BLACK\_WHITE:

black\_white(img\_copy)

else:

logger.error(f"can't find filter {filter\_name}")

raise ValueError(f"can't find filter {filter\_name}")

return img\_copy

### 4.4.2 图像处理文件代码说明

1. 文件定位：img\_modifier文件夹下img\_helper.py文件。
2. 相关库导入：

from PIL import Image, ImageEnhance

import logging

import img\_modifier.color\_filter as cf

1. 调整参数设定：针对对比度，亮度和锐化程度，设置参数进行限制，避免调整幅度过大产生不必要的变化。

# 对比度

CONTRAST\_FACTOR\_MAX = 1.5

CONTRAST\_FACTOR\_MIN = 0.5

# 锐化

SHARPNESS\_FACTOR\_MAX = 3

SHARPNESS\_FACTOR\_MIN = -1

# 亮度

BRIGHTNESS\_FACTOR\_MAX = 1.5

BRIGHTNESS\_FACTOR\_MIN = 0.5

1. 事件函数定义：
   * 上传获取图片get\_img(path)：path为图片路径
   * 尺寸改变函数resize(img, width, height)：img为待处理图片，width为图片宽度，height为图片高度
   * 旋转图片函数rotate(img, angle)：img为待处理图片，angle为旋转角度
   * 滤镜调用函数color\_filter(img, filter\_name)：img为待处理图片，filter\_name为滤镜键
   * 亮度调整函数brightness(img, factor)：img为待处理图片，factor为调整增强程度参数
   * 对比度调整函数contrast(img, factor)：img为待处理图片，factor为调整增强程度参数
   * 锐化调整函数sharpness(img, factor)：img为待处理图片，factor为调整增强程度参数
   * 翻转函数flip\_left(img)和flip\_top(img)：img为待处理图片
   * 保存图片函数save(img, path)：img为待处理图片，path为保存路径
   * 打开图片函数open\_img(img)：img为待处理图片

def get\_img(path):

if path == "":

logger.error("path is empty of has bad format")

raise ValueError("path is empty of has bad format")

try:

return Image.open(path)

except Exception:

logger.error(f"can't open the file {path}")

raise ValueError(f"can't open the file {path}")

def resize(img, width, height):

"""Resize image"""

return img.resize((width, height))

def rotate(img, angle):

"""Rotate image"""

return img.rotate(angle, expand=True)

def color\_filter(img, filter\_name):

"""Filter image"""

return cf.color\_filter(img, filter\_name)

def brightness(img, factor):

"""Adjust image brightness form 0.5-2 (1 - original)"""

if factor > BRIGHTNESS\_FACTOR\_MAX or factor < BRIGHTNESS\_FACTOR\_MIN:

raise ValueError("factor should be [0-2]")

enhancer = ImageEnhance.Brightness(img)

return enhancer.enhance(factor)

def contrast(img, factor):

"""Adjust image contrast form 0.5-1.5 (1 - original)"""

if factor > CONTRAST\_FACTOR\_MAX or factor < CONTRAST\_FACTOR\_MIN:

raise ValueError("factor should be [0.5-1.5]")

enhancer = ImageEnhance.Contrast(img)

return enhancer.enhance(factor)

def sharpness(img, factor):

"""Adjust image sharpness form 0-2 (1 - original)"""

if factor > SHARPNESS\_FACTOR\_MAX or factor < SHARPNESS\_FACTOR\_MIN:

raise ValueError("factor should be [0.5-1.5]")

enhancer = ImageEnhance.Sharpness(img)

return enhancer.enhance(factor)

def flip\_left(img):

"""Flip left to right"""

return img.transpose(Image.FLIP\_LEFT\_RIGHT)

def flip\_top(img):

"""Flip top to bottom"""

return img.transpose(Image.FLIP\_TOP\_BOTTOM)

def save(img, path):

"""Save image to hard drive"""

img.save(path)

def open\_img(img)：

img.open()

### 4.4.3 图片配置文件代码说明：

1. 文件定位：主文件下img\_modifier.py文件。
2. 文件库导入：

import getopt

import sys

import logging

from img\_modifier import img\_helper

1. 函数初始化函数init()：命令处理函数，将操作函数进行封装便于选择调用。

def init():

args = sys.argv[1:]

if len(args) == 0:

logger.error("-p can't be empty")

raise ValueError("-p can't be empty")

logger.debug(f"run with params: {args}")

# transform arguments from console

opts, rem = getopt.getopt(args, "p:", ["rotate=", "resize=", "color\_filter=", "flip\_top", "flip\_left"])

rotate\_angle = resize = color\_filter = flip\_top = flip\_left = None

path = None

for opt, arg in opts:

if opt == "-p":

path = arg

elif opt == "--rotate":

rotate\_angle = int(arg)

elif opt == "--resize":

resize = arg

elif opt == "--color\_filter":

color\_filter = arg

elif opt == "--flip\_top":

flip\_top = True

elif opt == "--flip\_left":

flip\_left = arg

if not path:

raise ValueError("No path")

img = img\_helper.get\_img(path)

if rotate\_angle:

img = img\_helper.rotate(img, rotate\_angle)

if resize:

w, h = map(int, resize.split(','))

img = img\_helper.resize(img, w, h)

if color\_filter:

img = img\_helper.color\_filter(img, color\_filter)

if flip\_left:

img = img\_helper.flip\_left(img)

if flip\_top:

img = img\_helper.flip\_top(img)

if \_\_debug\_\_:

img.show()

### 4.4.4 布局和主程序文件代码说明：

1. 文件定位：主文件下photo\_editor.py文件。
2. 文件库导入：

import sys

import ntpath

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import Qt

from PyQt5.QtGui import \*

from functools import partial

from img\_modifier import img\_helper

from img\_modifier import color\_filter

from PIL import ImageQt

from logging.config import fileConfig

import logging

1. 基础参数设定：基本调整参数设定，包括颜色，基本尺寸限制。

\_img\_original = None

\_img\_preview = None

THUMB\_BORDER\_COLOR\_ACTIVE = "#3893F4"

THUMB\_BORDER\_COLOR = "#ccc"

BTN\_MIN\_WIDTH = 120

ROTATION\_BTN\_SIZE = (70, 30)

THUMB\_SIZE = 120

SLIDER\_MIN\_VAL = -100

SLIDER\_MAX\_VAL = 100

SLIDER\_DEF\_VAL = 0

1. 操作类Operations定义：其中包括初始化，重置属性，变化检测，字符处理函数。

* 初始化函数\_\_init\_\_(self)：

def \_\_init\_\_(self):

self.color\_filter = None

self.flip\_left = False

self.flip\_top = False

self.rotation\_angle = 0

self.size = None

self.brightness = 0

self.sharpness = 0

self.contrast = 0

* 重置函数reset(self)：

def reset(self):

self.color\_filter = None

self.brightness = 0

self.sharpness = 0

self.contrast = 0

self.size = None

self.flip\_left = False

self.flip\_top = False

self.rotation\_angle = 0

* 变化检测函数has\_changes(self)：

def has\_changes(self):

return self.color\_filter or self.flip\_left\

or self.flip\_top or self.rotation\_angle\

or self.contrast or self.brightness\

or self.sharpness or self.size

* 字符处理函数\_\_str\_\_(self)：

def \_\_str\_\_(self):

return f"size: {self.size}, filter: {self.color\_filter}, " \

f"b: {self.brightness} c: {self.contrast} s: {self.sharpness}, " \

f"flip-left: {self.flip\_left} flip-top: {self.flip\_top} rotation: {self.rotation\_angle}"

1. 基础调整函数定义：包括尺寸变换限定函数，调整主原理函数，操作主函数。

* 尺寸变换限定函数\_get\_ratio\_height(width, height, r\_width)和\_get\_ratio\_width(width, height, r\_height)：

def \_get\_ratio\_height(width, height, r\_width):

return int(r\_width/width\*height)

def \_get\_ratio\_width(width, height, r\_height):

return int(r\_height/height\*width)

* 调整主原理函数\_get\_converted\_point(user\_p1, user\_p2, p1, p2, x)：user\_p2参数限定最大值，user\_p1参数限定最小值，p2系数增强最大值，p1系数增强最小值，x给定参数值。

def \_get\_converted\_point(user\_p1, user\_p2, p1, p2, x):

r = (x - user\_p1) / (user\_p2 - user\_p1)

return p1 + r \* (p2 - p1)

* 操作主函数get\_img\_with\_all\_operations()：

def \_get\_img\_with\_all\_operations():

logger.debug(operations)

b = operations.brightness

c = operations.contrast

s = operations.sharpness

img = \_img\_preview

if b != 0:

img = img\_helper.brightness(img, b)

if c != 0:

img = img\_helper.contrast(img, c)

if s != 0:

img = img\_helper.sharpness(img, s)

if operations.rotation\_angle:

img = img\_helper.rotate(img, operations.rotation\_angle)

if operations.flip\_left:

img = img\_helper.flip\_left(img)

if operations.flip\_top:

img = img\_helper.flip\_top(img)

if operations.size:

img = img\_helper.resize(img, \*operations.size)

return img

1. 布局类ActionTabs(QTabWidget)定义：主要存放四大编辑器布局。
2. 四大编辑区卡片类定义：
   * RotationTab(QWidget)：旋转翻转页布局类
     + \_\_init\_\_(self, parent)：初始化函数
     + on\_rotate\_left(self)和on\_rotate\_right(self)：顺时针逆时针每次转90度函数：

def on\_rotate\_left(self):

logger.debug("rotate left")

operations.rotation\_angle = 0 if operations.rotation\_angle == 270 else operations.rotation\_angle + 90

self.parent.parent.place\_preview\_img()

def on\_rotate\_right(self):

logger.debug("rotate left")

operations.rotation\_angle = 0 if operations.rotation\_angle == -270 else operations.rotation\_angle - 90

self.parent.parent.place\_preview\_img()

* + - on\_flip\_left(self)和on\_flip\_top(self)：左右翻转和上下翻转函数：

def on\_flip\_left(self):

logger.debug("flip left-right")

operations.flip\_left = not operations.flip\_left

self.parent.parent.place\_preview\_img()

def on\_flip\_top(self):

logger.debug("flip top-bottom")

operations.flip\_top = not operations.flip\_top

self.parent.parent.place\_preview\_img()

* + ModificationTab(QWidget)：尺寸改变页布局类
    - \_\_init\_\_(self, parent)：初始化函数
    - set\_boxes(self):尺寸展示函数：

def set\_boxes(self):

self.width\_box.setText(str(\_img\_original.width))

self.height\_box.setText(str(\_img\_original.height))

* + - on\_width\_change(self, e)和on\_height\_change(self, e)：高度和宽度改变函数：

def on\_width\_change(self, e):

logger.debug(f"type width {self.width\_box.text()}")

if self.ratio\_check.isChecked():

r\_height = \_get\_ratio\_height(\_img\_original.width, \_img\_original.height, int(self.width\_box.text()))

self.height\_box.setText(str(r\_height))

def on\_height\_change(self, e):

logger.debug(f"type height {self.height\_box.text()}")

if self.ratio\_check.isChecked():

r\_width = \_get\_ratio\_width(\_img\_original.width, \_img\_original.height, int(self.height\_box.text()))

self.width\_box.setText(str(r\_width))

* + - on\_ratio\_change(self, e)：比例改变函数
    - on\_apply(self, e)：尺寸改变生效函数：

def on\_apply(self, e):

logger.debug("apply")

operations.size = int(self.width\_box.text()), int(self.height\_box.text())

self.parent.parent.update\_img\_size\_lbl()

self.parent.parent.place\_preview\_img()

* + AdjustingTab(QWidget)：像素调整页布局类
    - \_\_init\_\_(self, parent)：初始化函数
    - reset\_sliders(self)：重置函数：

def reset\_sliders(self):

self.brightness\_slider.setValue(SLIDER\_DEF\_VAL)

self.sharpness\_slider.setValue(SLIDER\_DEF\_VAL)

self.contrast\_slider.setValue(SLIDER\_DEF\_VAL)

* + - on\_brightness\_slider\_released(self)：亮度调节函数
    - on\_sharpness\_slider\_released(self)：锐化调节函数
    - on\_contrast\_slider\_released(self)：对比度调节函数：

def on\_contrast\_slider\_released(self):

logger.debug(self.contrast\_slider.value())

self.contrast\_slider.setToolTip(str(self.contrast\_slider.value()))

factor = \_get\_converted\_point(SLIDER\_MIN\_VAL, SLIDER\_MAX\_VAL, img\_helper.CONTRAST\_FACTOR\_MIN,

img\_helper.CONTRAST\_FACTOR\_MAX, self.contrast\_slider.value())

logger.debug(f"contrast factor: {factor}")

operations.contrast = factor

self.parent.parent.place\_preview\_img()

* + FiltersTab(QWidget)：滤镜选择页布局库
    - \_\_init\_\_(self, parent)：初始化函数
    - add\_filter\_thumb(self, name, title="")：添加滤镜函数
    - on\_filter\_select(self, filter\_name, e)：滤镜选择函数：

def on\_filter\_select(self, filter\_name, e):

logger.debug(f"apply color filter: {filter\_name}")

global \_img\_preview

if filter\_name != "none":

\_img\_preview = img\_helper.color\_filter(\_img\_original, filter\_name)

else:

\_img\_preview = \_img\_original.copy()

operations.color\_filter = filter\_name

self.toggle\_thumbs()

self.parent.parent.place\_preview\_img()

* + - toggle\_thumbs(self):切换滤镜函数：

def toggle\_thumbs(self):

for thumb in self.findChildren(QLabel):

color = THUMB\_BORDER\_COLOR\_ACTIVE if thumb.name == operations.color\_filter else THUMB\_BORDER\_COLOR

thumb.setStyleSheet(f"border:2px solid {color};")

1. 主布局类MainLayout(QVBoxLayout)定义：
   * \_\_init\_\_(self, parent)：初始化函数
   * place\_preview\_img(self)：背景图片设置函数
   * on\_save(self)：保存图片函数
   * on\_upload(self)：上传图片函数
   * update\_img\_size\_lbl(self)：图片大小加载函数
   * on\_reset(self)：重置函数
2. 主窗口类EasyPzUI(QWidget)定义：
   * \_\_init\_\_(self, parent)：初始化函数
   * center(self)：居中函数
   * closeEvent(self, event)：程序退出函数

def closeEvent(self, event):

logger.debug("close")

if operations.has\_changes():

reply = QMessageBox.question(self, "",

"您还没有保存<br>确定退出?", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

if reply == QMessageBox.Yes:

event.accept()

else:

event.ignore()

1. 主程序：程序入口

if \_\_name\_\_ == '\_\_main\_\_':

fileConfig('logging\_config.ini')

app = QApplication(sys.argv)

win = EasyPzUI()

sys.exit(app.exec\_())

# 5. 代码测试

## 5.1 运行进入主界面

运行主程序photo\_editor.py，进入如下主界面：包括提示信息“点击上传图片编辑”，个人证件照和创作说明，三个按钮：“上传”，“重置”，“保存”。其中后两个按钮要上传照片后开始编辑才能启用。

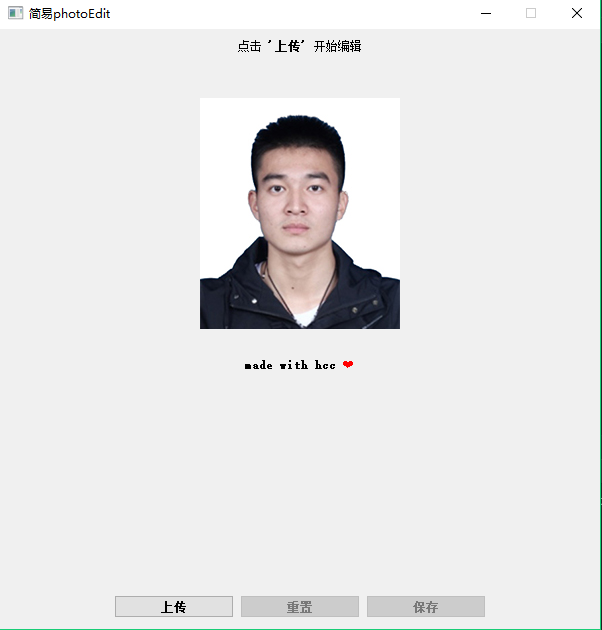


Figure :程序运行入口

## 5.2 上传图片功能测试

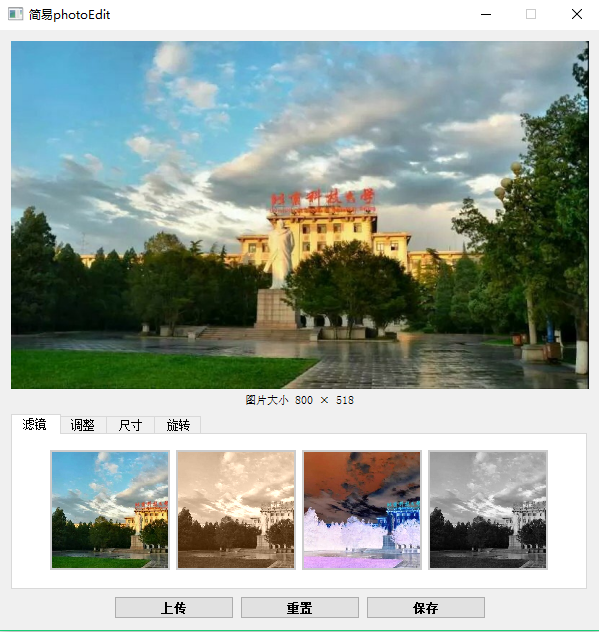
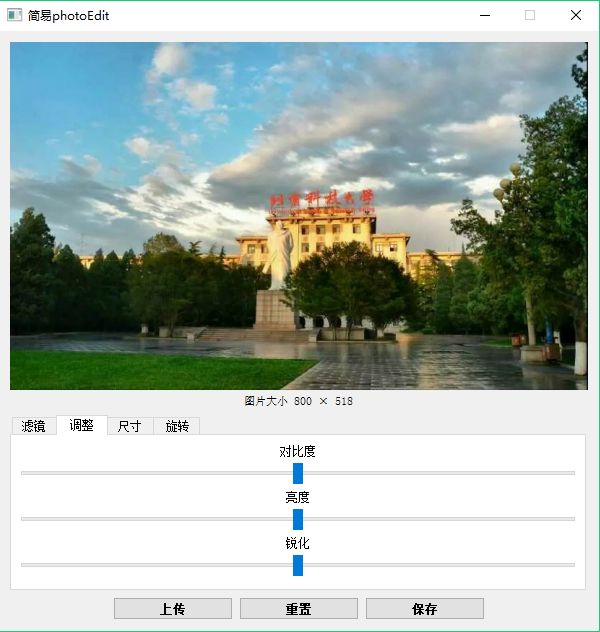
1. 预先选择好一张北科大西门图片，上传后进入编辑页面：首先展示的是滤镜添加区，下方是预览，点击任意一个我们设置好的滤镜即可添加滤镜更改照片，下方预览同时还提供与原图对比效果。

Figure ：上传图片展示

1. 点击调整进入调整相关操作：并对比调节后的效果（右图）



**Figure 3：调整对比图**

1. 点击尺寸进入尺寸编辑区：左图为原始尺寸，右图将尺寸改为400\*270后的结果。

**Figure 4：尺寸对比图**

1. 点击旋转进入旋转翻转编辑区：下方四图对比四类旋转翻转操作。

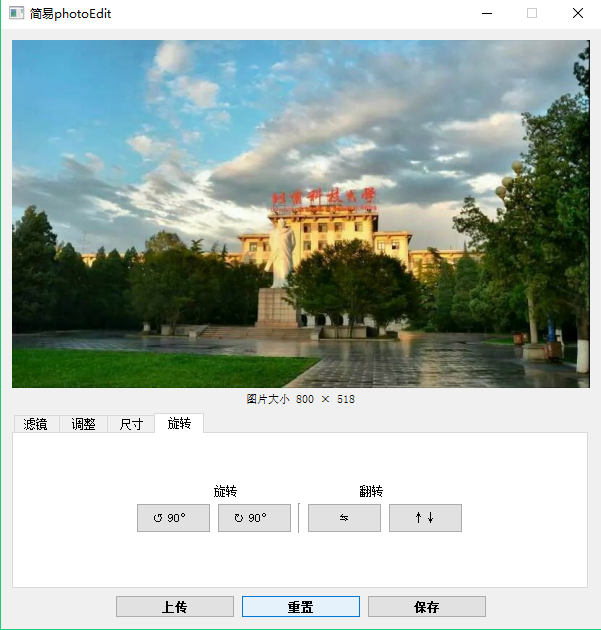


Figure 5：旋转编辑区

**Figure 6：功能对比图**

1. 点击重置和退出展示：退出时如果图片进行了改变操作，将会提示保存退出。

**Figure 7：保存与退出**

# 6. 结论与未来方向

## 6.1 结论

通过该项目的编程实践学习，基本完成了pyQt5基本界面布局和部件的应用，以及图像处理中的加滤镜方式；修改图片对比度，亮度和锐化；图片尺寸的修改与缩放；图片的旋转和翻转等操作。文件导入以及保存等方式。完成简单的图像处理的几个方面。

通过实践得出项目实践的基本流程：首先选好一个开发方向，分析开发的基本需求，根据需求查找资料部署。然后开始设计整个项目的基本框架，包括界面，功能，调整。其次对工程文件的层次分析，将各类型代码分类分块。紧接着就可以开始进行代码构建。基本代码形成后进行优化调整，并进行bug调试，测试基本功能的正常，最终完成整个项目。

## 6.2 未来方向

整个项目工程完成的功能不多，可以在以后扩展滤镜的样式，同时增加模糊功能，尝试使用高斯模糊，表面模糊等模糊方式，增加鼠标的功能性，通过鼠标自由控制图片旋转，缩放，移动等功能，增加添加文字功能等等，将整个工程扩展。

# 7. 致谢

感谢皇甫伟老师八周python课程的悉心教导。

# 8. 参考文献与链接

[1]李晓军.计算机图像处理的相关技术[J/OL].电子技术与软件工程,2019(08):73[2019-05-03].

[2]王昕瑞,林忠.图像特效滤镜算法研究与实现[J].电脑知识与技术,2018,14(33):200-202.

[3]耿颖.使用Python语言的GUI可视化编程设计[J].单片机与嵌入式系统应用,2019,19(02):20-22+44.

[4]李艳梅. 图像增强的相关技术及应用研究[D].电子科技大学,2013.

[5]李光鑫. 基于亮度-对比度传递技术的彩色图像融合算法[A]. 中国光学学会.中国光学学会2010年光学大会论文集[C].中国光学学会:中国光学学会,2010:7.

[6] <https://www.jianshu.com/p/e8d058767dfa>

[7] <http://www.xdbcb8.com/pyqt5/>

[8] <https://blog.csdn.net/ajaxhe/article/details/7541705>

[9] <https://www.cnblogs.com/Security-Darren/p/4168310.html>

[10] <https://www.cnblogs.com/liujiacai/p/7804848.html>

[11] <https://www.cnblogs.com/zz22--/p/7719285.html>

[12] <https://blog.csdn.net/lz0499/article/details/80963696>