

基于 PyQt 与 PIL 制作的简易 PS

姓名：吴义豪 学号：41824334
北京科技大学计通学院 通信 1804 班

摘要：本报告介绍了一套简易 PS 照片处理软件的需求分析、设计、实现和测试。文章第一部分介绍了项目的设计背景与意义，文章第二部分针对项目目标进行需求分析；文章第三部分根据需求选择合适模块、框架、进行 UI 界面布局，并介绍主要算法的细节；第四部分对软件实现过程进行了简单说明，介绍代码实现过程，本软件的 GUI 界面采用 pyqt5 进行设计；第五部分进行了相关功能测试，实现对一张特定图片的缩放、旋转、添加滤镜、调整亮度、裁剪。，并选取一张代表图片进行功能说明；文章第六部分指出了现阶段软件的不足，并对未来的改进提供了简单思路。

关键字：照片处理；GUI 设计；PyQt5。

1. 项目背景和意义

1.1 项目背景

当前流行的计算机桌面应用程序大多数为图形化用户界面（Graphic User Interface, GUI），即通过鼠标对菜单、按钮等图形化元素触发指令，并从标签、对话框等图型化显示容器中获取人机对话信息。而在本次专选课中，从一开始的海龟画图，到之后学习 GUI 图形界面设计，一直希望通过 python 的 PyQt 和 PIL 设计一个简单应用，由于本人平常热爱摄影，故设计的应用为简单图形界面处理。

1.2 项目意义

通过本次小项目设计，将 python 课程中所学知识加以应用，主要对 PyQt 和 PIL 库加以再学习和应用，搭建简单的图片处理器，同时此次学习中搜索了大量相关文献与资料，拓展了课外知识。

2.需求分析

2.1 项目功能目标

- (1) 完成 GUI 界面，包含显示图片
- (2) 调整图片亮度
- (3) 缩放旋转图片
- (4) 加注水印

2.2 主要模块调用

PyQt5>=5.8.2

Pillow>=4.1.1

模块说明：PyQt5 用来创建 Python GUI 应用程序的工具包。Pillow 提供了基本的图像处理功能，如：改变图像大小，旋转图像，图像格式转换，色场空间转换，图像增强，直方图处理，插值和滤波等等。

2.3 功能流程图



3. 概要和详细设计

3.1 设计概要

- (1) 搭建主界面基本框架，设计布局
- (2) 基本功能设计，包括滤镜，调整，尺寸，旋转，添加水印五大功能实现
- (3) 添加按钮，联系功能与布局设置
- (4) 基本功能测试
- (5) 界面 UI 美化，代码优化与结构调整

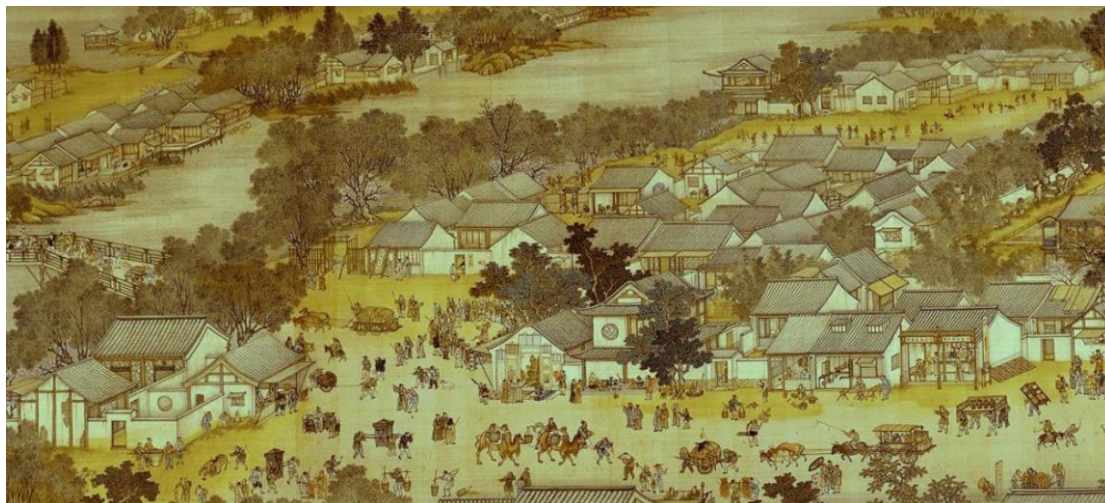
3.2 功能实现及引用

(1) 滤镜函数

(a) 滤镜的原理，常见的是针对数字图像的像素矩阵，使用一个 $n \times n$ 的方形矩阵做滤波器（即 kernel，常见的如 3×3 ， 5×5 等），对该像素矩阵进行遍历，遍历后的图像就是输出图像，ImageFilter 是 Python PIL 的滤镜模块，当前版本支持 10 种加强滤镜，通过这些预定义的滤镜，可以方便的对图片进行一些过滤操作，从而去掉图片中的噪音(部分的消除)，这样可以降低图像处理算法的复杂度(如模式识别等)，更方便的实现和预览一些算法的效果。

```
1. #例如，ImageFilter.EMBOSS 浮雕滤镜
2. class EMBOSS(BuiltinFilter):
3.     name = "Emboss"
4.     filterargs = (3, 3), 1, 128, (
5.         -1, 0, 0,
6.         0, 1, 0,
7.         0, 0, 0
8.     )
```

效果图：



原图



浮雕效果展示图

(b)基本思想:通过对 RGB 三个通道的值进行一定的运算得到一个新的值,然后再显示出来。鉴于这种算法思路,本项目滤镜实现方案:

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

(c) 实现方法:

例如黑白滤波:

- ☛ 将像素点 R, G, B 先加权平均并记为 s
- ☛ 取 k 值 30
- ☛ 新的 RGB: $r=s+k*2; g=s+k; b=s$

```
1. def sepia(img):  
2.     pix = img.load()  
3.     for i in range(img.width):
```

```

4.     for j in range(img.height):
5.         s = sum(pix[i, j]) // 3
6.         k = 30
7.         pix[i, j] = (s+k*2, s+k, s)

```

(2) 对比度、亮度、锐化调整函数

(a) 亮度：图像亮度通俗理解便是图像的明暗程度，如果灰度值在 $[0, 255]$ 之间，则 f 值越接近 0 亮度越低， f 值越接近 255 亮度越高。可给定一定范围对图片进行亮度调节，例如设置图像的灰度级是 $[Lmin, Lmax]$ 。

(b) 饱和度指的是图像颜色种类的多少，上面提到图像的灰度级是 $[Lmin, Lmax]$ ，则在 $Lmin$ 、 $Lmax$ 的中间值越多，便代表图像的颜色种类多，饱和度也就更高，外观上看起来图像会更鲜艳，调整饱和度可以修正过度曝光或者未充分曝光的图片。使图像看上去更加自然。

(c) 对比度指的是图像暗和亮的落差值，即图像最大灰度级和最小灰度级之间的差值。

(d) 以 RGB 空间图像亮度、对比度调节为例展示实现过程：

对于数字图像变换，设原像素灰度为 $f(i, j)$ ，转化后的像素灰度为 $g(i, j)$ ，则常用的线性变换是 $g(i, j) = af(i, j) + b$ ，其中系数 a 影响图像的对比度，系数 b 影响图像的亮度，具体如下：

- $a=1$ 时是原图；
- $a>1$ 时对比度增强，图像看起来更加清晰；
- $a<1$ 时对比度减弱，图像看起来变暗；
- b 影响图像的亮度，随着增加 b ($b>0$) 和减小 b ($b<0$)，图像整体的灰度值上移或者下移，也就是图像整体变亮或者变暗，不会改变图像的对比度

```

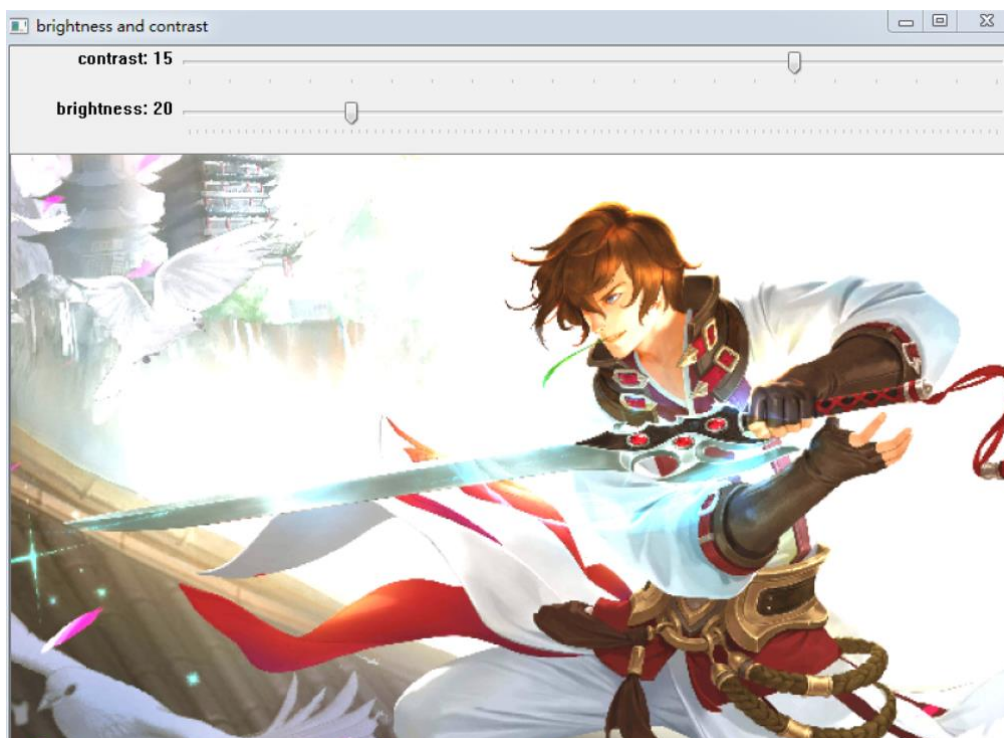
1. #学习参考代码:
2. import cv2
3. import imutils
4. import numpy as np
5.
6. def c_and_b(arg):
7.     cnum = cv2.getTrackbarPos(trackbar_name1, wname)

```



```
8. bnum = cv2.getTrackbarPos(trackbar_name2, wname)
9. cimg = np.ones((img.shape[0], img.shape[1], 3), dtype=np.uint8)
10. for i in range(img.shape[0]):
11.     for j in range(img.shape[1]):
12.         lst = 0.1*cnum*img[i, j] + bnum
13.         cimg[i, j] = [int(ele) if ele < 255 else 255 for ele in lst]
14. cv2.imshow(wname, imutils.resize(cimg, 800))
15.
16. wname = 'brightness and contrast'
17. trackbar_name1 = 'contrast'
18. trackbar_name2 = 'brightness'
19. img = cv2.imread("E:/img/libai.jpg")
20. height, width = img.shape[:2]
21. img = cv2.resize(img, (int(width/height*400), 400), interpolation=cv2.INTER_CUBIC)
22.
23. cv2.namedWindow(wname)
24. cv2.createTrackbar(trackbar_name1, wname, 10, 20, c_and_b)
25. cv2.createTrackbar(trackbar_name2, wname, 0, 100, c_and_b)
26.
27. c_and_b(0)
28. if cv2.waitKey(0) == 27:
29.     cv2.destroyAllWindows()
```

效果图：



李白调整后

4. 代码实现

4.1 Python 版本与 IDE 说明

Python 版本: Python 3.7

IDE: Pycharm

4.2 相关库调用

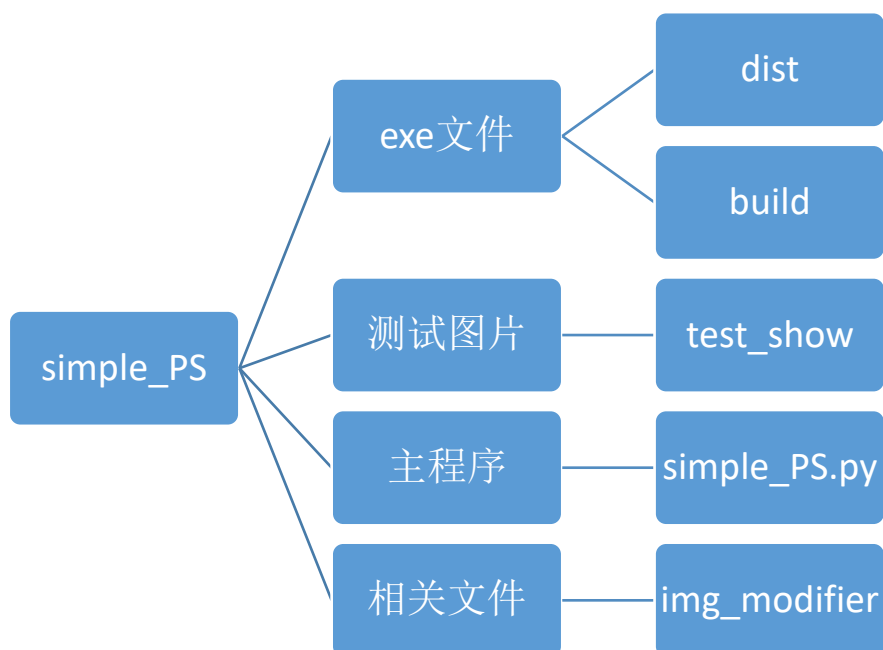
PyQt5、PIL、ntpath、sys、Functools

4.3 文件分区说明

目录结构如下:

名称	修改日期	类型	大小
.idea	2020/4/27 0:13	文件夹	
__pycache__	2020/4/26 16:30	文件夹	
build	2020/4/25 23:50	文件夹	
dist	2020/4/25 23:51	文件夹	
img_modifier	2020/4/26 15:41	文件夹	
test_show	2020/4/25 23:31	文件夹	
logo.ico	2020/4/25 0:37	图标	36 KB
logo.png	2020/4/25 0:37	PNG 文件	36 KB
requirements.txt	2020/4/27 0:16	文本文档	1 KB
simple_PS.exe	2020/4/25 23:51	应用程序	5,244 KB
simple_PS.py	2020/4/27 0:12	JetBrains PyChar...	22 KB
simple_PS.spec	2020/4/25 23:50	SPEC 文件	1 KB

文件目录



4.4 关键代码说明

4.4.1 simple_PS.py 主要代码说明

(1) 库文件导入

```
1. import sys
2. import ntpath
3. from PyQt5.QtWidgets import *
4. from PyQt5.QtCore import Qt
5. from PyQt5.QtGui import *
6. from PyQt5 import QtCore
7. from PIL import ImageDraw, ImageFont
8. from functools import partial
9. from img_modifier import img_helper
10. from img_modifier import color_filter
11. from PIL import ImageQt
```

(2) 基础参数设置

```
1. THUMB_BORDER_COLOR_ACTIVE = "#3893F4"
2. THUMB_BORDER_COLOR = "#ccc"
3. BTN_MIN_WIDTH = 120
4. ROTATION_BTN_SIZE = (50, 30) #旋转模块按钮大小
5. THUMB_SIZE = 150 #滤镜预览模块大小
6.
7. SLIDER_MIN_VAL = -100
8. SLIDER_MAX_VAL = 100
9. SLIDER_DEF_VAL = 0
```

(3) 操作类 Operations 定义：其中包括初始化，重置，变化检测，字符处理函数。

● 初始化

```
1. def __init__(self):
2.     self.color_filter = None
3.     self.flip_left = False
4.     self.flip_top = False
5.     self.rotation_angle = 0
6.     self.size = None
7.     self.brightness = 0
8.     self.sharpness = 0
9.     self.contrast = 0
```

● 重置


```
1.     def reset(self):
2.         self.color_filter = None
3.         self.brightness = 0
4.         self.sharpness = 0
5.         self.contrast = 0
6.         self.size = None
7.         self.flip_left = False
8.         self.flip_top = False
9.         self.rotation_angle = 0
```

● 变化检测

```
1.     def has_changes(self):
2.         return self.color_filter or self.flip_left\
3.             or self.flip_top or self.rotation_angle\
4.             or self.contrast or self.brightness\
5.             or self.sharpness or self.size
```

● 字符处理

```
1.     def __str__(self):
2.         return f"size: {self.size}, filter: {self.color_filter}, " \
3.             f"b: {self.brightness} c: {self.contrast} s: {self.sharpness}, " \
4.             f"flip-left: {self.flip_left} flip-top: {self.flip_top} rotation: {self.rotation_angle}"
```

(4) 功能性操作按钮类，五个功能按钮

```
1.     class ActionTabs(QTabWidget):
2.         """Action tabs widget"""
3.         def __init__(self, parent):
4.             super().__init__()
5.             self.parent = parent
6.
7.             self.filters_tab = FiltersTab(self)
8.             self.adjustment_tab = AdjustingTab(self)
9.             self.modification_tab = ModificationTab(self)
10.            self.rotation_tab = RotationTab(self)
11.            self.text_tab = TextTab(self)
12.
13.            self.addTab(self.filters_tab, "滤镜")
14.            self.addTab(self.adjustment_tab, "调整")
15.            self.addTab(self.modification_tab, "尺寸")
16.            self.addTab(self.rotation_tab, "旋转")
```

```
17.     self.addTab(self.text_tab, "水印")
18.
19.     self.setMaximumHeight(190)
```

(5) 五个功能编辑按钮布局配置类似，以添加水印按钮介绍主要布局，并介绍五个功能编辑按钮类的关键函数：

- 添加水印（TextTab）按钮类：

```
1.  class TextTab(QWidget):
2.      """Text tab widget"""
3.      def __init__(self, parent):
4.          super().__init__()
5.          self.parent = parent
6.
7.          self.width_lbl = QLabel('点击“添加水印”输入文字', self)
8.
9.          self.apply_btn = QPushButton("添加水印")
10.         self.apply_btn.setFixedWidth(90)
11.         self.apply_btn.clicked.connect(self.text_apply)
12.
13.         width_layout = QHBoxLayout()
14.
15.         apply_layout = QHBoxLayout()
16.         apply_layout.addWidget(self.apply_btn)
17.         apply_layout.setAlignment(Qt.AlignRight)
18.
19.         lbl_layout = QHBoxLayout()
20.         lbl_layout.setAlignment(Qt.AlignLeft)
21.         lbl_layout.addWidget(self.width_lbl)
22.
23.         main_layout = QVBoxLayout()
24.         main_layout.setAlignment(Qt.AlignCenter)
25.
26.         main_layout.addLayout(lbl_layout)
27.         main_layout.addLayout(width_layout)
28.         main_layout.addLayout(apply_layout)
29.
30.         self.setLayout(main_layout)
31.
32.     def set_boxes(self):
33.         self.width_box.setText(str(_img_original.width))
34.         self.height_box.setText(str(_img_original.height))
35.
```

```

36. def on_width_change(self, e):
37.     if self.ratio_check.isChecked():
38.         r_height = _get_ratio_height(_img_original.width, _img_original.height, int(self.width_box.text(
           )))
39.         self.height_box.setText(str(r_height))
40.
41. def text_apply(self, e):
42.     global _img_preview
43.     text_tuple = QDialog.getText(self, "添加水印", "请输入文本", text="吴义豪
       ")#gettext return tuple
44. #此处有一个坑调了好久才发现，getText 返回值是一个元组
45.     text = text_tuple[0]
46.     draw = ImageDraw.Draw(_img_preview)
47.     setFont = ImageFont.truetype('C:/windows/fonts/Dengl.ttf', 100)
48.     draw.text((40, 40), text, font=setFont, fill="#0000ff", direction=None)
49.     self.parent.parent.place_preview_img()

```

● 滤镜按钮类关键函数

■ 滤镜选择函数：on_filter_select

```

1. def on_filter_select(self, filter_name, e):
2.     global _img_preview
3.     if filter_name != "none":
4.         _img_preview = img_helper.color_filter(_img_original, filter_name)
5.     else:
6.         _img_preview = _img_original.copy()
7.     operations.color_filter = filter_name
8.     self.toggle_thumbs()
9.     self.parent.parent.place_preview_img()

```

■ 切换滤镜函数：def toggle_thumbs

```

1. def toggle_thumbs(self):
2.     for thumb in self.findChildren(QLabel):
3.         color = THUMB_BORDER_COLOR_ACTIVE if thumb.name == operations.color_filter else THU
           MB_BORDER_COLOR
4.         thumb.setStyleSheet(f"border:2px solid {color};")

```

● 调整按钮类（AdjustingTab）关键函数

■ 亮度调节：on_brightness_slider_released

```

1. def on_brightness_slider_released(self):
2.     self.brightness_slider.setToolTip(str(self.brightness_slider.value()))
3.     factor = _get_converted_point(SLIDER_MIN_VAL, SLIDER_MAX_VAL, img_helper.
4.         BRIGHTNESS_FACTOR_MIN,
5.         img_helper.BRIGHTNESS_FACTOR_MAX, self.brightness_slider.value())
6.     operations.brightness = factor
7.     self.parent.parent.place_preview_img()

```

■ 锐化调节: on_sharpness_slider_released

```

1. def on_sharpness_slider_released(self):
2.     self.sharpness_slider.setToolTip(str(self.sharpness_slider.value()))
3.     factor = _get_converted_point(SLIDER_MIN_VAL, SLIDER_MAX_VAL,
4.         img_helper.SHARPNESS_FACTOR_MIN,
5.         img_helper.SHARPNESS_FACTOR_MAX, self.sharpness_slider.value())
6.     operations.sharpness = factor
7.     self.parent.parent.place_preview_img()

```

■ 对比度调节: on_contrast_slider_released

```

1. def on_contrast_slider_released(self):
2.     self.contrast_slider.setToolTip(str(self.contrast_slider.value()))
3.     factor = _get_converted_point(SLIDER_MIN_VAL, SLIDER_MAX_VAL,
4.         img_helper.CONTRAST_FACTOR_MIN,
5.         img_helper.CONTRAST_FACTOR_MAX, self.contrast_slider.value())
6.     operations.contrast = factor
7.     self.parent.parent.place_preview_img()

```

● 尺寸按钮类 (ModificationTab) 关键函数

■ 尺寸配置: set_boxes

```

1. def set_boxes(self):
2.     self.width_box.setText(str(_img_original.width))
3.     self.height_box.setText(str(_img_original.height))

```

■ 尺度改变: on_width_change(self, e) 和 on_height_change(self, e)

```

1. def on_width_change(self, e):
2.     if self.ratio_check.isChecked():
3.         r_height = _get_ratio_height(_img_original.width, _img_original.height, int(self.width_box.text(
4.             )))

```

```

4.         self.height_box.setText(str(r_height))
5.     def on_height_change(self, e):
6.         if self.ratio_check.isChecked():
7.             r_width = _get_ratio_width(_img_original.width, _img_original.height, int(self.height_box.text(
            )))
8.             self.width_box.setText(str(r_width))

```

● 旋转按钮类（RotationTab）关键函数：

■ 旋转 90°：on_rotate_left（左旋）和 on_rotate_right（右旋）

```

1.     def on_rotate_left(self):
2.         operations.rotation_angle = 0 if operations.rotation_angle == 270 else operations.rotation_angle
           + 90
3.         self.parent.parent.place_preview_img()
4.
5.     def on_rotate_right(self):
6.         operations.rotation_angle = 0 if operations.rotation_angle == -270 else operations.rotation_angle
           - 90
7.         self.parent.parent.place_preview_img()

```

■ 翻转函数：on_flip_left（左右）和 on_flip_top（上下）

```

1.     def on_flip_left(self):
2.         operations.flip_left = not operations.flip_left
3.         self.parent.parent.place_preview_img()
4.
5.     def on_flip_top(self):
6.         operations.flip_top = not operations.flip_top
7.         self.parent.parent.place_preview_img()

```

（6）主布局类：MainLayout

```

1.     class MainLayout(QVBoxLayout):
2.         """Main layout"""
3.         def __init__(self, parent):
4.             super().__init__()
5.             self.parent = parent
6.
7.             self.img_lbl = QLabel("点击 <b>'上传'</b> 开始编辑<br>"
8.                                   "<div style='margin: 50px 0'><img src='logo.png' /></div>"
9.                                   "<span style='color:red'>♥</span><span style='color:white;font-size:16px;'> Wh
               en technology enters life, you will find many casual beauty in life.</span> <span style='color:red'>
               ♥</span>")

```

```
10.     self.img_lbl.setAlignment(Qt.AlignCenter)
11.
12.     self.file_name = None
13.
14.     self.img_size_lbl = None
15.     self.img_size_lbl = QLabel()
16.     self.img_size_lbl.setAlignment(Qt.AlignCenter)
17.
18.     upload_btn = QPushButton("上传")
19.     upload_btn.setMinimumWidth(BTN_MIN_WIDTH)
20.     upload_btn.clicked.connect(self.on_upload)
21.     upload_btn.setStyleSheet("font-weight:bold;")
22.
23.     self.reset_btn = QPushButton("重置")
24.     self.reset_btn.setMinimumWidth(BTN_MIN_WIDTH)
25.     self.reset_btn.clicked.connect(self.on_reset)
26.     self.reset_btn.setEnabled(False)
27.     self.reset_btn.setStyleSheet("font-weight:bold;")
28.
29.     self.save_btn = QPushButton("保存")
30.     self.save_btn.setMinimumWidth(BTN_MIN_WIDTH)
31.     self.save_btn.clicked.connect(self.on_save)
32.     self.save_btn.setEnabled(False)
33.     self.save_btn.setStyleSheet("font-weight:bold;")
34.
35.     self.addWidget(self.img_lbl)
36.     self.addWidget(self.img_size_lbl)
37.     self.addStretch()
38.
39.     self.action_tabs = ActionTabs(self)
40.     self.addWidget(self.action_tabs)
41.     self.action_tabs.setVisible(False)
42.
43.     btn_layout = QHBoxLayout()
44.     btn_layout.setAlignment(Qt.AlignCenter)
45.     btn_layout.addWidget(upload_btn)
46.     btn_layout.addWidget(self.reset_btn)
47.     btn_layout.addWidget(self.save_btn)
48.
49.     self.addLayout(btn_layout)
```

(7) 主窗口类: Simple_PS

```
1.  class Simple_PS(QWidget):
```



```
2.     """Main widget"""
3.     def __init__(self):
4.         super().__init__()
5.         self.main_layout = MainLayout(self)
6.         self.setLayout(self.main_layout)
7.         self.setMinimumSize(640,600)
8.         self.setMaximumSize(1280, 900)
9.         self.setGeometry(600, 600, 600, 600)
10.        self.setWindowTitle('Simple-PS')
11.        self.setWindowOpacity(0.9) # 设置窗口透明度
12.        self.setAttribute(QtCore.Qt.WA_TranslucentBackground) # 设置窗口背景透明
13.        # self.setWindowFlag(QtCore.Qt.FramelessWindowHint) # 隐藏边框
14.        self.center()
15.        self.show()
16.
17.    def center(self):
18.        """align window center"""
19.        qr = self.frameGeometry()
20.        cp = QDesktopWidget().availableGeometry().center()
21.        qr.moveCenter(cp)
22.        self.move(qr.topLeft())
23.
24.    def closeEvent(self, event):
25.        if operations.has_changes():
26.            reply = QMessageBox.question(self, "",
27.                                         "您还没有保存<br>确定退出?", QMessageBox.Yes |
28.                                         QMessageBox.No, QMessageBox.No)
29.            if reply == QMessageBox.Yes:
30.                event.accept()
31.            else:
32.                event.ignore()
33.
34.    def resizeEvent(self, e):
35.        pass
```

4.4.2 color_filter.py 主要代码说明

(1) 黑白滤镜

```
1.     def sepia(img):
2.         pix = img.load()
3.         for i in range(img.width):
4.             for j in range(img.height):
```

```
5.         s = sum(pix[i, j]) // 3
6.         k = 30
7.         pix[i, j] = (s+k*2, s+k, s)
```

(2) 均值滤镜

```
1.  def black_white(img):
2.     pix = img.load()
3.     for i in range(img.width):
4.         for j in range(img.height):
5.             s = sum(pix[i, j]) // 3
6.             pix[i, j] = (s, s, s)
```

(3) 负滤镜

```
1.  def negative(img):
2.     pix = img.load()
3.     for i in range(img.width):
4.         for j in range(img.height):
5.             pix[i, j] = (255 - pix[i, j][0], 255 - pix[i, j][1], 255 - pix[i, j][2])
```

4.4.3 img_helper.py 主要代码说明

(1) 确定相关参数

```
1.  # constants
2.  # contrast ratio
3.  CONTRAST_FACTOR_MAX = 1.5
4.  CONTRAST_FACTOR_MIN = 0.5
5.  # sharpening
6.  SHARPNESS_FACTOR_MAX = 3
7.  SHARPNESS_FACTOR_MIN = -1
8.  # brightness
9.  BRIGHTNESS_FACTOR_MAX = 1.5
10. BRIGHTNESS_FACTOR_MIN = 0.5
```

(2) 事件函数:

```
1.  def get_img(path):
2.     """上传获取图片"""
3.     if path == "":
4.         raise ValueError("path is empty of has bad format")
5.     try:
6.         return Image.open(path)
```

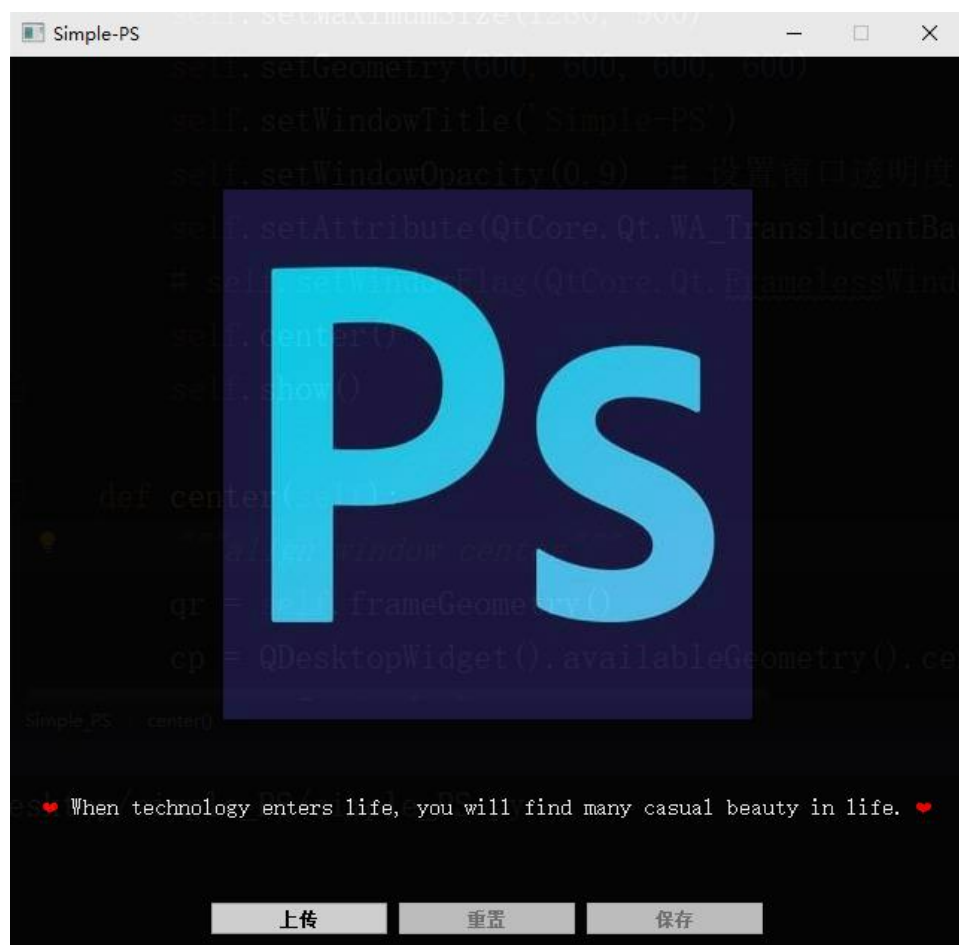
```
7.     except Exception:
8.         raise ValueError(f"can't open the file {path}")
9.
10.    def resize(img, width, height):
11.        """尺寸改变"""
12.        return img.resize((width, height))
13.
14.    def rotate(img, angle):
15.        """旋转图片"""
16.        return img.rotate(angle, expand=True)
17.
18.    def color_filter(img, filter_name):
19.        """滤镜调用"""
20.        return cf.color_filter(img, filter_name)
21.
22.    def brightness(img, factor):
23.        """亮度调整, factor 为调整增强程度参数: 0.5-2 (1 - original)"""
24.        if factor > BRIGHTNESS_FACTOR_MAX or factor < BRIGHTNESS_FACTOR_MIN:
25.            raise ValueError("factor should be [0-2]")
26.
27.        enhancer = ImageEnhance.Brightness(img)
28.        return enhancer.enhance(factor)
29.
30.    def contrast(img, factor):
31.        """对比度调整, factor 为调整增强程度参数: 0.5-1.5 (1 - original)"""
32.        if factor > CONTRAST_FACTOR_MAX or factor < CONTRAST_FACTOR_MIN:
33.            raise ValueError("factor should be [0.5-1.5]")
34.
35.        enhancer = ImageEnhance.Contrast(img)
36.        return enhancer.enhance(factor)
37.
38.    def sharpness(img, factor):
39.        """锐化调整, factor 为调整增强程度参数: 0-2 (1 - original)"""
40.        if factor > SHARPNESS_FACTOR_MAX or factor < SHARPNESS_FACTOR_MIN:
41.            raise ValueError("factor should be [0.5-1.5]")
42.
43.        enhancer = ImageEnhance.Sharpness(img)
44.        return enhancer.enhance(factor)
45.
46.    def flip_left(img):
47.        """左右翻转"""
48.        return img.transpose(Image.FLIP_LEFT_RIGHT)
49.
50.    def flip_top(img):
```

```
51. """上下翻转"""
52. return img.transpose(Image.FLIP_TOP_BOTTOM)
53.
54. def save(img, path):
55.     """保存图片"""
56.     img.save(path)
57.
58. def open_img(img):
59.     """打开图片"""
60.     img.open()
```

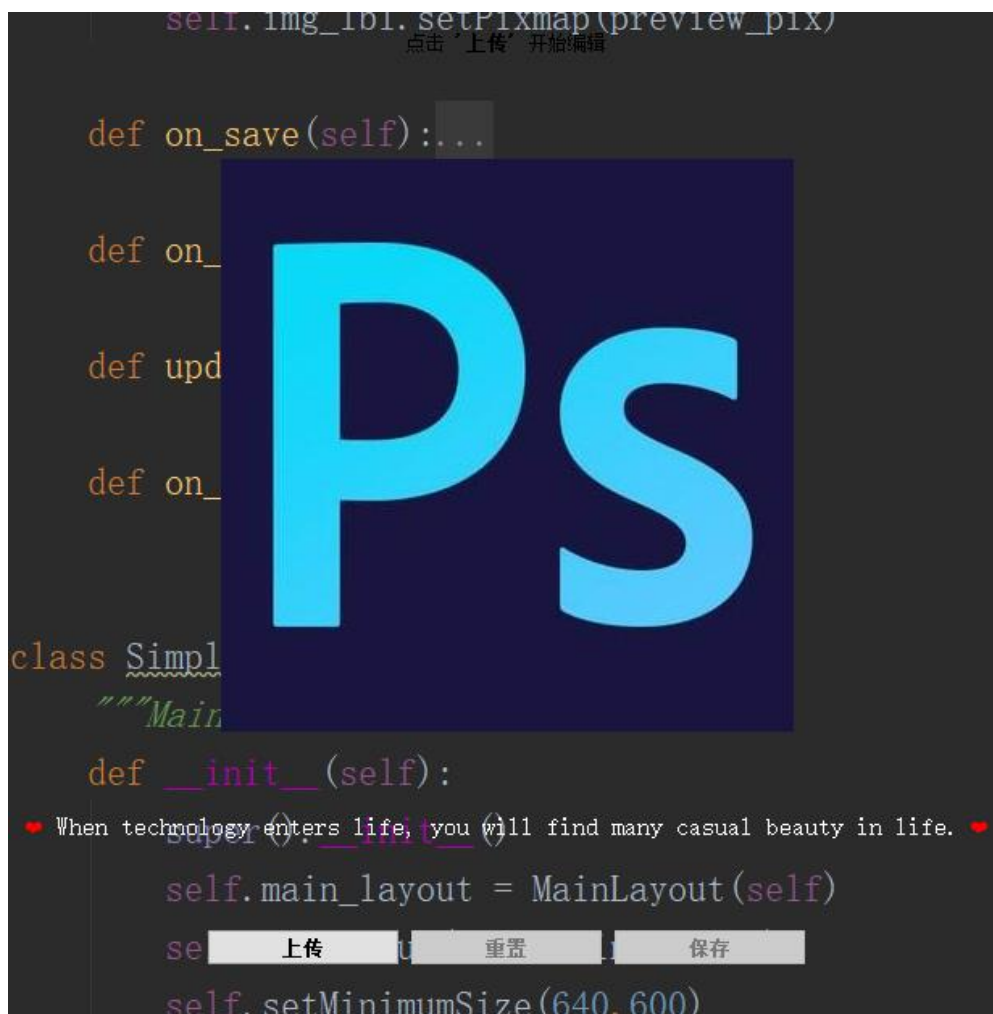
5. 代码测试

5.1 运行测试

5.1.1 运行进入主界面

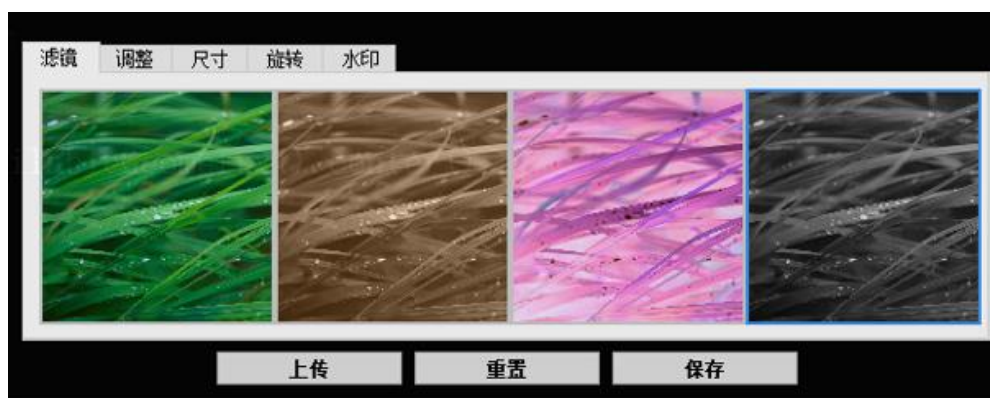


透明度参数可调，外边框可去，以求更惊艳的科技效果，如下：



- 上传：点击可上传图片
- 重置：初始化成照片原始状态，需上传照片后可用此功能。
- 保存：保存更改后照片，需上传照片后可用此功能。

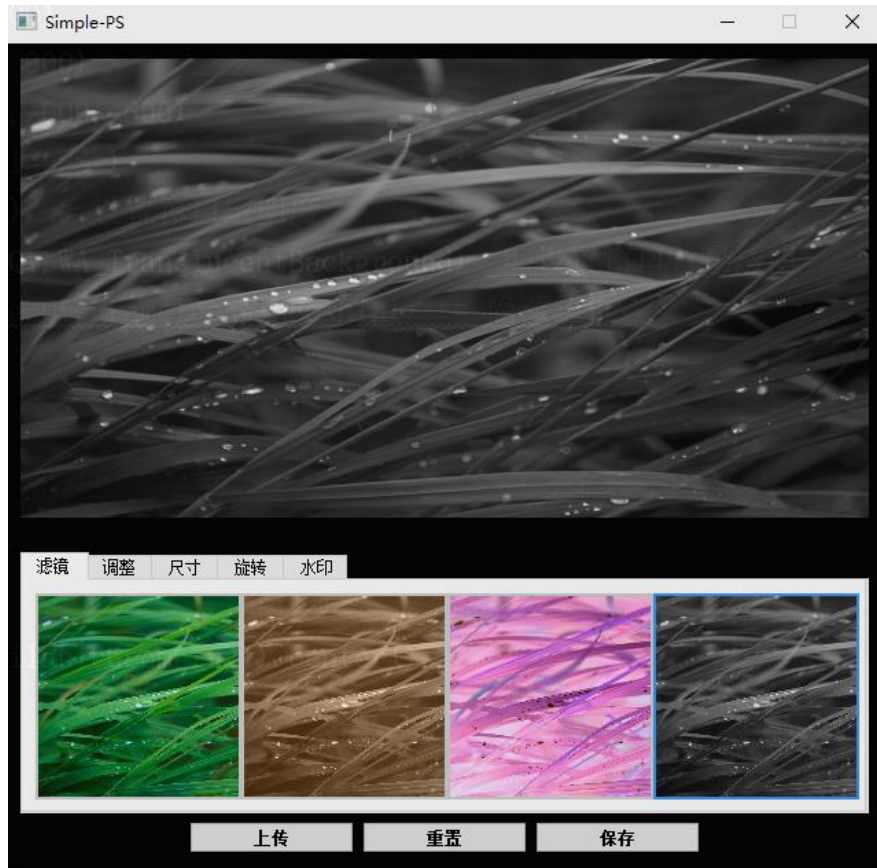
5.1.2 上传一张照片后进入编辑界面



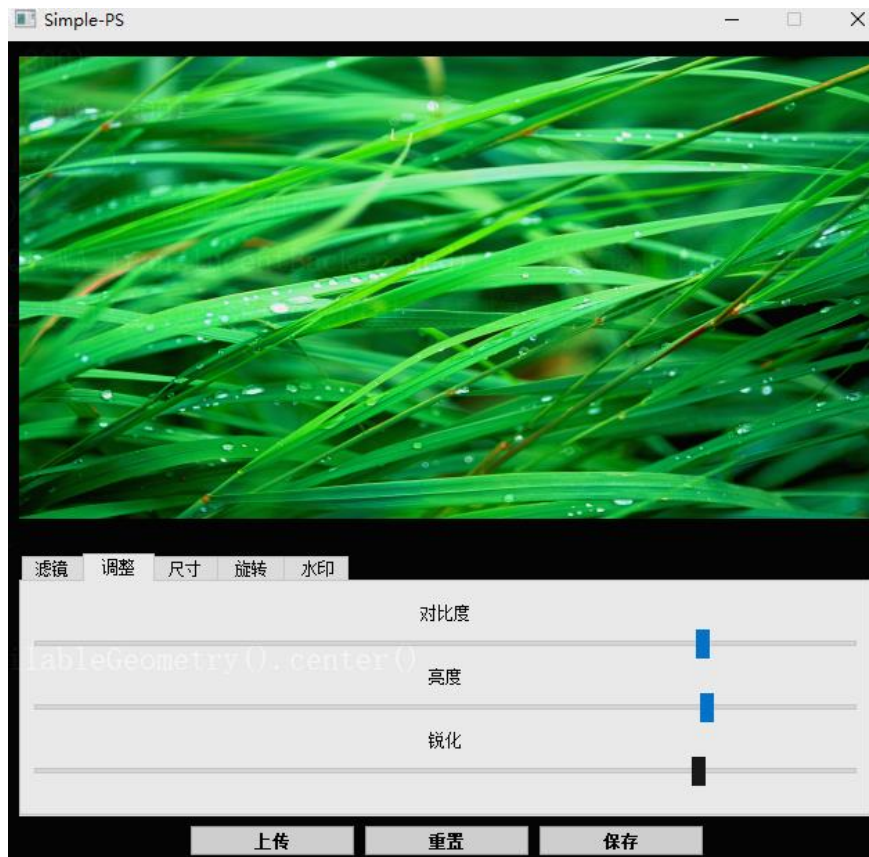
功能区

5.2 功能测试

5.2.1 滤镜功能:

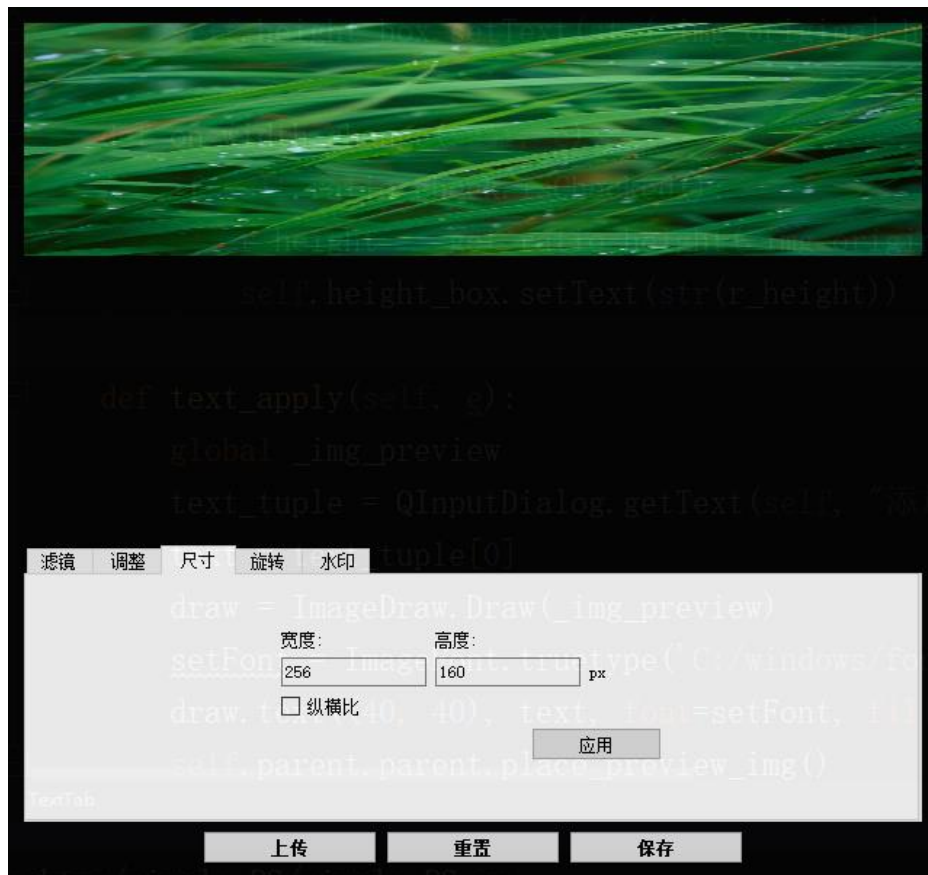


5.2.2 调整功能:



5.2.3 尺寸功能:

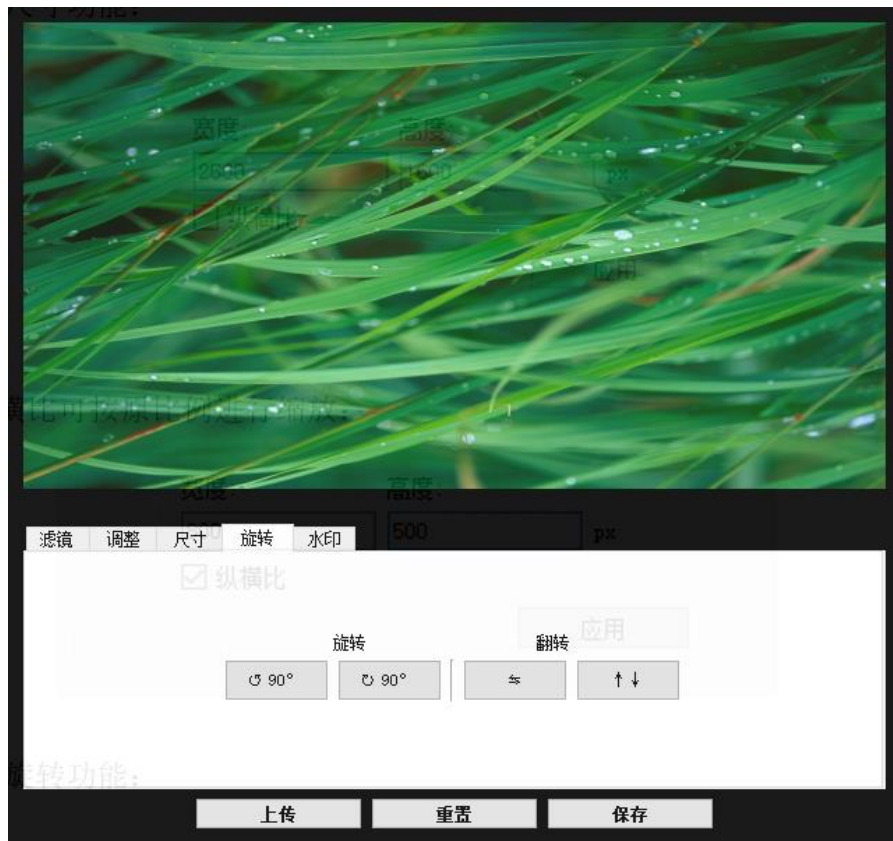




锁定纵横比可按原比例进行缩放：

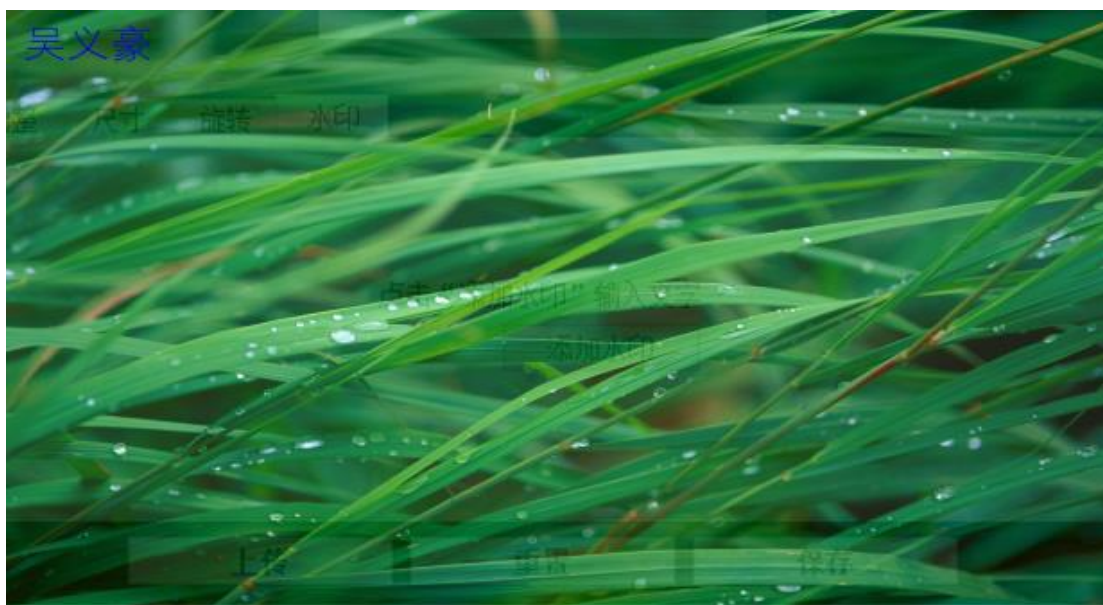


5.2.4 旋转功能：



5.2.5 水印功能：文本默认值是 “吴义豪”





效果图

6. 结论与未来方向

6.1 结论

通过制作此次简易 PS 照片处理器，利用 PyQt 完成了页面基本布局，基本上实现了预期功能，具有添加滤镜，调整对比度、亮度、锐化，调整尺寸大小，旋转与添加水印的功能。

在项目实践过程中，自己的项目开发能力也得到了提升，也掌握了开发一个基本软件的流程步骤。项目中也出现了很多 bug 和闪退现象，但最终经过耐心调试，逐渐解决，最终完成了整个项目。

6.2 未来方向

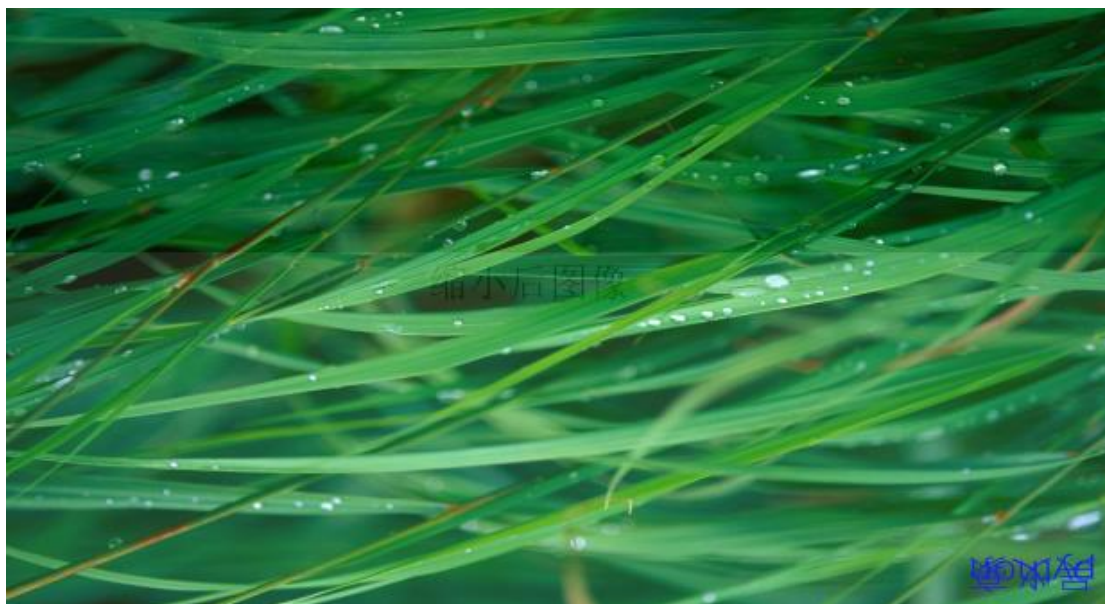
6.2.1 项目缺点

(1) 尺寸改变时，是对图片进行整体缩放，未添加裁剪功能，锁定纵横比时，输入一条边长度，若另一条边非整数，常出现闪退现象。



缩小后图像

(2) 水印文字与图片耦合程度相对较高，水印功能固定添加在图像相对位置左上角，不论进行缩放还是旋转，水印位置相对照片都是在左上角不变，且重复添加会出现重叠现象。



水印重叠图片

(3) UI 界面美观程度有待提升。

6.2.2 未来改进方向

- (1) 尺寸功能处添加裁剪功能，方便照片处理。
- (2) 水印提供字体、大小选择，鼠标可改动水印相对位置，使显示不是那么单一。
- (3) 改进界面设计，增加美观程度。

7. 致谢

感谢皇甫老师上半学期的教学指导，受益匪浅。

8. 参考文献与链接

- [1] https://blog.csdn.net/sinat_38682860/article/details/86510324
- [2] <https://zmister.com/archives/477.html>
- [3] <https://zmister.com/archives/477.html>
- [4] <https://blog.csdn.net/Eastmount/article/details/83548652>
- [5] <https://blog.csdn.net/lj6052317/article/details/78289953>
- [6] https://blog.csdn.net/weixin_39540045/article/details/80542651