王临寒_19377404_Python图像处理

🧍‍♀️19377404 王临寒

STEP 1 实现基类 Filter

```python
class Filter:
    def __init__(self, image):
        self.image = image

    def filter(self):
        pass
```

基类中的数据属性为 image；

参数列表可以放到 ImageShop 中实现；

定义空的 filter 方法让多态子类继承。

STEP 2 Filter 类的多个子类

以我的头像为例做处理。

原图：

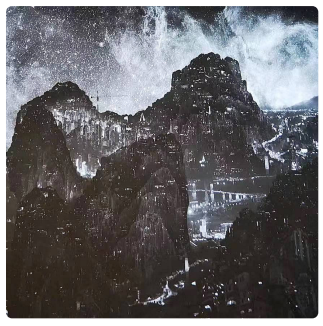
STEP 2.1 边缘提取

```python
class EdgeExtraction(Filter):
    def filter(self):
        self.image = self.image.filter(PIL.ImageFilter.FIND_EDGES)
        return self.image
```

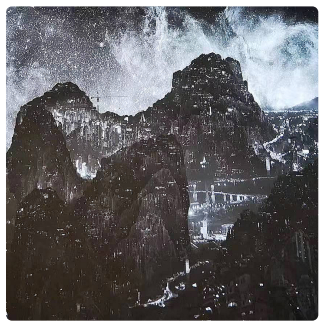
STEP 2.2 锐化

```python
class Sharpen(Filter):
    def filter(self):
        self.image = self.image.filter(PIL.ImageFilter.SHARPEN)
        return self.image
```

锐化一次的图像：



锐化两次的图像：



STEP 2.3 模糊

```python
class Blur(Filter):
    def filter(self):
        self.image = self.image.filter(PIL.ImageFilter.BLUR)
        return self.image
```

Python

图片模糊之后的效果：

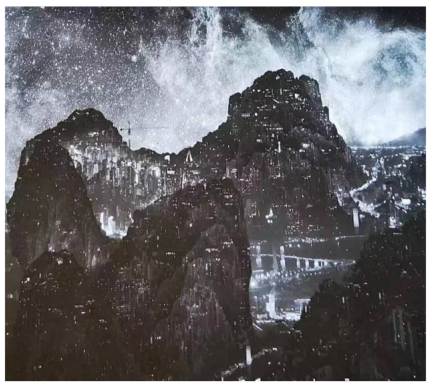
STEP 2.4 大小调整

```python
class Resize(Filter):

    def filter(self, ratio):
        '''
        :param ratio: 调整大小的比例
        :return:
        '''
        width = self.image.size[0]  # 获取宽度
        height = self.image.size[1]  # 获取高度
        self.image = self.image.resize((int(width * ratio), int(height * ratio)), Image.ANTIALIAS)
        return self.image
```

Python

原图尺寸：



原图大小为 866*867

调整后大小：

调整后大小为 259*260

STEP 2.5 亮度调整

```python
class Bright(Filter):

    def filter(self, factor_num):
        '''
        :param factor_num: 增强亮度的参数
        :return:
        '''
        enh_bri = PIL.ImageEnhance.Brightness(self.image)
        self.image = enh_bri.enhance(factor=factor_num)
        return self.image
```

Python

Ref.

Python 图像处理 Pillow 库 基础篇

图像处理是常用的技术，python 拥有丰富的第三方扩展库，Pillow 是 Python3 最常用的图像处理库，目前最高版本5.2.0。Python2 使用Pil库，两者是使用方法差不多，区别在于类的引用不同。 注意：Pil 库与 Pillow 不...

zhuanlan.zhihu.com

Python图像处理库Pillow入门

page.description

ivanzz1001.github.io

STEP 3 ImageShop 类

首先加载类的数据属性：

```python
class ImageShop:
    def __init__(self, path, file_format):
        '''
        :param path: 要处理的图片路径
        :param file_format: 图片格式
        '''
        # 图片路径
        self.path = path
        # 图片格式
        self.file_format = file_format
        # 图片实例
        self.image = []
        # 处理之后的图片
        self.adjusted = []
        # 存储操作
        self.operation = []
```

Python

再加载图片实例：

```python
    def load_images(self):
        '''
        :return: 加载指定路径下指定格式的所有图片到类中
        '''
        if self.path[-1] == "/":
            for f_name in os.listdir(self.path):
                if f_name.endswith(self.file_format):
```

```
                now_directory = self.path + f_name
                self.image.append(Image.open(now_directory))
        else:
            self.image.append(Image.open(self.path))
```

Python

Ref.

Python文件操作，看这篇就足够

本文为译文，原文链接 working-with-files-in-python 本人博客: 编程禅师Python中有几个内置模块和方法来处理文件。这些方法被分割到例如 os, os.path , shutil 和 pathlib 等等几个模块中。文章将列...

zhuanlan.zhihu.com

再实现__batch_ps 函数：

```
    def __batch_ps(self, img_order):
        '''
        对指定的图片进行处理
        :param img_order: 要处理的图片在self.image中的索引
        :return:
        '''
        for j in self.operation:
            if j[0] == "EdgeExtraction":
                self.image[img_order] = EdgeExtraction(self.image[img_order]).filter()
                print("EdgeExtraction!")
            elif j[0] == "Sharpen":
                self.image[img_order] = Sharpen(self.image[img_order]).filter()
                print("Sharpen!")
            elif j[0] == "Blur":
                self.image[img_order] = Blur(self.image[img_order]).filter()
                print("Blur!")
            elif j[0] == "Resize":
                self.image[img_order] = Resize(self.image[img_order]).filter(j[1])
                print("Resize!")
            else:
                self.image[img_order] = Bright(self.image[img_order]).filter(j[1])
                print("Bright!")
```

Python

再实现 batch_ps 函数：

```
    def batch_ps(self, operation):
        '''
        输入每一张图片到__batch_ps()
        :param operation: 存储操作参数的列表，元素是元组，元组的第一个元素是操作名，后面的元素是参数
        :return:
        '''
        self.operation = operation
        for origin_img_order in range(len(self.image)):
            self.__batch_ps(origin_img_order)
        self.adjusted = self.image
```

Python

再实现 display 函数：

```
    def display(self, row_num, col_num, sum_num):
        '''
        :param row_num: 显示行数
        :param col_num: 显示列数
        :param sum_num: 显示的最大图片数量
        :return:
        '''
        for img_num in range(1, sum_num + 1):
            plt.subplot(row_num, col_num, img_num)
            plt.imshow(self.adjusted[img_num - 1])
            plt.axis('off')
        plt.show()
```

Python

最后实现 save 函数：

```
    def save(self, save_path, output_format):
        '''
```

```
        :param save_path: 保存路径
        :param output_format: 输出图片格式
        :return:
        '''
        for i in range(len(self.adjusted)):
            self.adjusted[i].save(save_path + str(i) + output_format)
```

Python

STEP 4 实现测试类 TestImageShop

继承 ImageShop 类别，并对属性进行初始化：

```
class TestImageShop(ImageShop):
    def __init__(self, path, file_format, operation, output_path, row_num, col_num, sum_num):
        '''
        :param path: 输入图片路径
        :param file_format: 图片格式
        :param operation: 操作参数
        :param output_path: 输出路径
        :param row_num: 展示的行数
        :param col_num: 展示的列数
        :param sum_num: 展示的最大图片数量
        '''
        super().__init__(path, file_format)
        self.operation = operation
        self.output_path = output_path
        self.row_num = row_num
        self.col_num = col_num
        self.sum_num = sum_num
```

Python

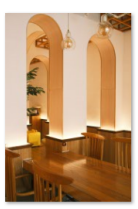
测试的 main 函数：

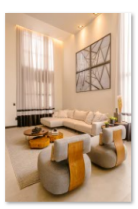
```
if __name__ == "__main__":
    photoshop = TestImageShop("./待处理图片/", ".jpg", [("Bright", 1.5), ("Resize", 0.3), ("Blur", 0)], "./批量输出图片/",
    photoshop.load_images()
    photoshop.batch_ps(photoshop.operation)
    photoshop.display(photoshop.row_num, photoshop.col_num, photoshop.sum_num)
    photoshop.save(photoshop.output_path, photoshop.file_format)
```

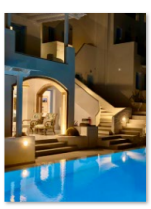
Python

选取的测试集为


2y-kang-d2p-gjzPWh4-unsplash

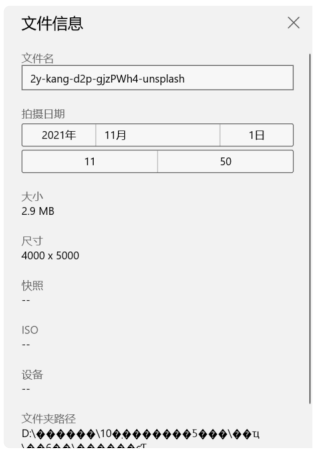

jason-sung-orn59FnvySA-unsplash


jonathan-borba-UisC7KLAWjs-unsplash

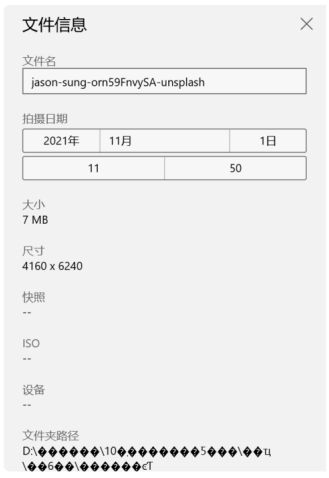

szabolcs-varnai-qYs1QQHPO0A-unsplash


tom-podmore-n4z9z2EJlvw-unsplash

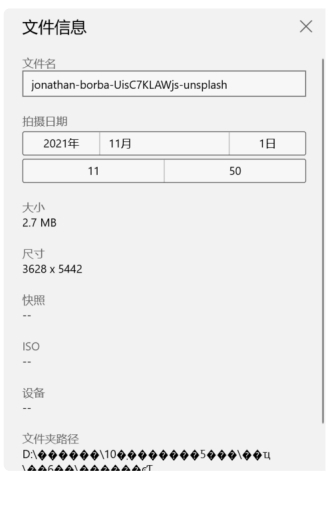
五张高清图片，其大小分别为



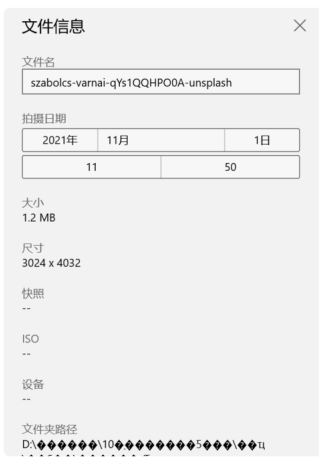
2.9MB（4000*5000）

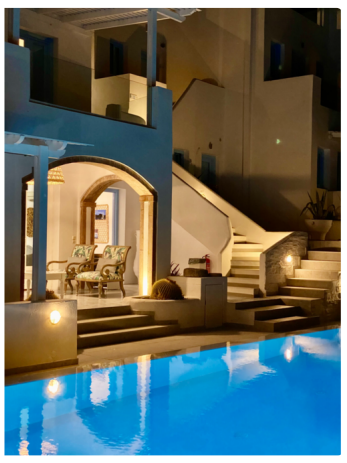


7MB（4160*6240）



2.7MB（3628*5442）





1.2MB（3024*4032）

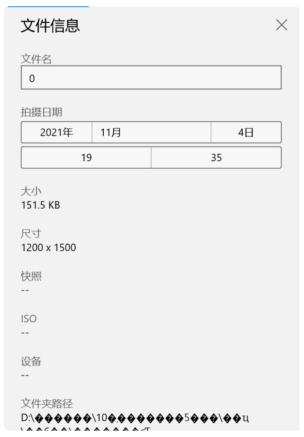
4.1MB（3947*5936）

初始化的参数对图片进行了变亮 1.5 倍、变小为原来的 0.3 倍、模糊（参数为 0 代表无参数）。

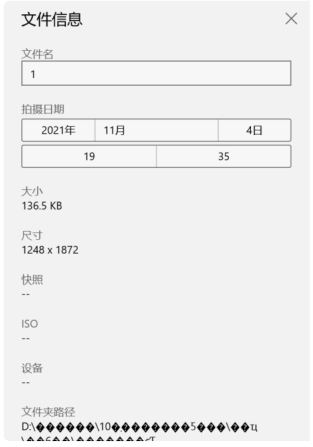
运行之后的效果经 display 函数（初始化参数为 1 行 5 列，最多显示 5 张图片）处理之后得到：

可以看到图片明显变亮了，输出到文件夹中如下：

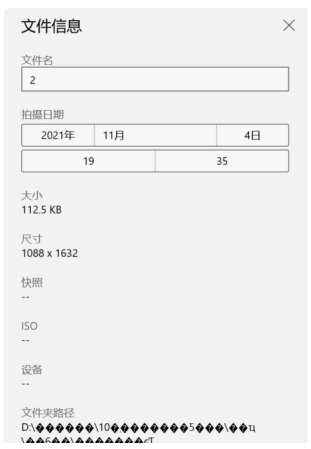
对每张图片的具体信息进行验证：



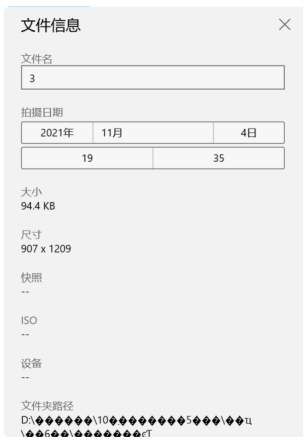
151.5KB(1200*1200)

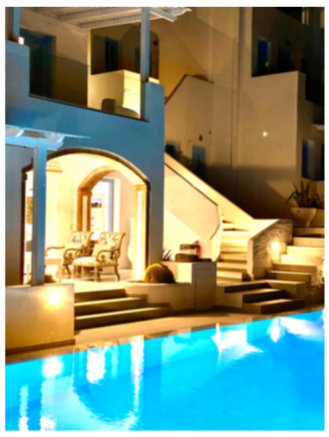


136.5KB(1248*1872)

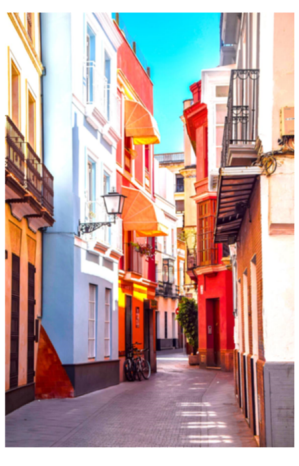


112.5KB(1088*1632)





94.4KB(907*1209)



233.4KB(1187*1780)

综上所述，图像处理功能基本实现。