



**数字信号与图像处理**

**课程设计报告**

**学 院 计算机工程学院 班 级 计算2114**

**姓 名 庄佳强 学 号 202121331104 。**

**成 绩 指导老师 陈杰，汪志华**

**黄敏，洪玉玲**

**评 语**

**2024年 1月**

## 目的

通过对相关项目的设计与开发，并完成相应设计报告的撰写，使学生系统地、全面地掌握信号与图像处理的基本知识和方法，提高实际动手能力和主要知识技能的应用。项目的设计要求能够体现一定的实用价值，能够把所学的知识综合灵活地应用到项目中。

同时，也训练学生初步撰写文档的能力，培养学生在教师的指导下独立完成设计报告的能力，逐步提高学生检索资料的能力。

## 内容

必选模块

包含：构图（裁剪）、旋转、光感、亮度、曝光、对比度、曲线调整、直方图均衡化、饱和度调整，曲线调色、HSL，锐化，平滑、色温、色调、文字。具体内容请参考醒图APP中的相关功能。

三选一模块

去雾功能：完成图像去雾功能

本人实现内容：HSL，锐化，平滑、色调、文字、去雾功能、前端页面。

## 环境

PyCharm Community Edition 2023.2.1

Python 3.11

## 设计思想及系统结构

### 4.1问题的分析

HSL：即色调、饱和度、亮度（Hue, Saturation, Lightness）。

锐化：锐化主要是通过在边缘两侧，增加黑白相间的高对比线条“隔离带”，让边缘看起来更加突出锐利。

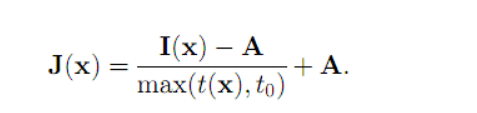
饱和度调整：调整HSL中的S。

曲线调色：通过一条曲线反映RGB三个通道输入输出的对应关系。

平滑：将噪声所在像素点的像素值处理为其周围临近像素点的值的近似值。

色调：色调是指图像的相对明暗程度，在彩色图像上表现为颜色。调高白色,相当于增加颜色的含量,调高黑色,相当于减少颜色的含量。

文字：在图像指定位置一个地方添加文字。

去雾：

I(x)为原图，J(x)为去雾的图，t(x)为透射率 ，t0=0.1，A是全球大气光成分

设计思想

饱和度调整：

见HSL。

曲线调色：

和曲线调整相似，只是只调节单个通道。

HSL：

将图像转换HSL格式，其中H对应色相，S对应饱和度，L对应亮度。然后对每一个进行调整，再转换成想要的格式会BGR格式。

锐化：

将原图转换成灰度图和拉普拉斯算子卷积计算得出模糊图像，原图减去得到锐化的图像。

平滑：

用高斯滤波器进行平滑处理。

色调：

见HSL。

文字：

先选定一个添加文字的区域，输入自己想要的文字（编码和OpenCV版本问题只能用英文）

去雾：

通过使用暗通道去雾法实现。

计算暗通道：对于彩色图像，首先计算每个像素的暗通道。暗通道的计算方式为对每个像素位置 (x, y)，在以该位置为中心的一个小窗口内找到最小的颜色值（R、G、B中的最小值）。这个过程使用快速滤波方法实现。

估计大气光：在暗通道的基础上，估计图像中的大气光值A。选择暗通道中最亮的像素点作为大气光。这个值表示在图像中最不透明的部分，即含有雾气最少的区域。

估计透射率： 利用大气光和原始图像计算透射率。透射率（Transmission）用来表示光在雾气中的传播损失程度。透射率的计算公式为：

descript

去雾处理:利用透射率对原始图像进行去雾处理。去雾的计算公式为：

descript

前端页面部分：

使用python的tkinter可以实现。

### 4.2系统结构

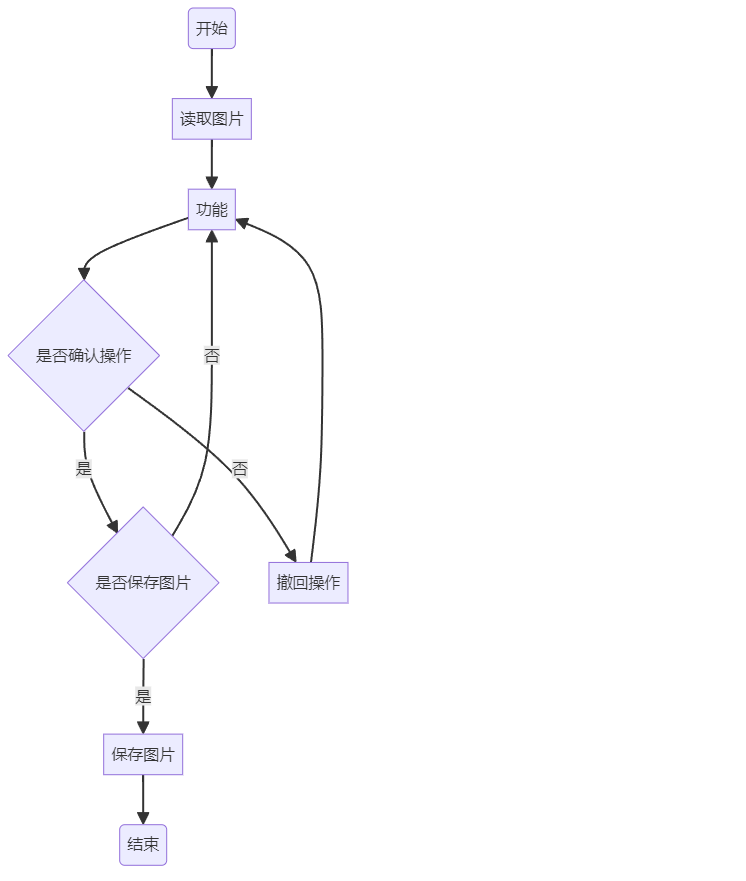


图4.1整体结构图

### 4.3代码分析

**HSL模块伪代码：**

HSL(图像, 色相调整因子 H, 饱和度调整因子 S, 亮度调整因子 L):

# 将图像从BGR颜色空间转换为HLS颜色空间

转换图像为 HLS 颜色空间

# 缩放调整因子以匹配HLS调整的期望范围

H = H / 5

S = S / 2

L = L / 2

# 调整HSL

对于每个像素 in 图像:

# 色相（H）调整

调整像素的 H 通道值为 (H + 原始 H) % 180

# 亮度（L）调整，将结果剪裁到[0, 255]范围内

调整像素的 L 通道值为 剪裁至 [0, 255] 的值 (L + 原始 L)

# 饱和度（S）调整，将结果剪裁到[0, 255]范围内

调整像素的 S 通道值为 剪裁至 [0, 255] 的值 (S + 原始 S)

# 将调整后的图像从HLS颜色空间转换回BGR颜色空间

转换图像为 BGR 颜色空间

# 返回调整后的图像

返回调整后的图像

**锐化模块伪代码:**

使用拉普拉斯算子实现。

LaplaceFilter(图像, 参数):

# 将参数从[0, 100]归一化为[0, 3]

归一化参数 = 参数 / 100 \* 3

# 应用Laplacian滤波器

滤波结果 = 应用Laplacian滤波器(图像, 数据类型=cv.CV\_64F)

# 将结果截断至非负值

滤波结果 = 对每个元素取最大值(0, 滤波结果)

# 将结果转换为8位无符号整数类型

滤波结果 = 转换为8位无符号整数(滤波结果)

# 返回滤波结果

返回 滤波结果

Crisp\_Enhancement(图像, 参数):

# 将参数归一化为浮点数，并除以70

归一化参数 = 参数 / 70.0

# 使用 LaplaceFilter 函数获取图像的 Laplace 变换

模糊图像 = LaplaceFilter(图像, 归一化参数)

# 计算锐化图像：原始图像 + (原始图像 - 模糊图像) \* 归一化参数

锐化图像 = 计算锐化图像(图像, 模糊图像, 归一化参数)

# 返回锐化图像

返回 锐化图像

**平滑模块：**

使用高斯模糊来实现平滑处理。

dst = cv.GaussianBlur(img, (7, 7), parameter)  
return dst

**色调模块:**

修改了色相，相当于修改了色调，因而在SHL中操作就行。

**添加文字模块:**

在前端把需要的点框选出来后，只需要使用cv.addText添加框即可。

def Add\_Text(img, text, x, y):  
 """  
 为图片指定位置添加文字  
 """  
 new\_img = img.copy()  
 cv.putText(new\_img, text, (x, y), cv.FONT\_HERSHEY\_SIMPLEX, 2,(0, 0, 0), 1, cv.LINE\_AA)  
  
 return new\_img

**去雾功能模块伪代码：**

原理使用使用暗通道去雾法,再其原理基础上就行修改。

函数 deHaze(m, r=81, eps=0.001, w=0.95, maxV1=0.80, bGamma=False):

m = m / 255.0

Y = 生成与 m 相同形状的零数组

Mask\_img, A = Defog(m, r, eps, w, maxV1) # 得到遮罩图像和大气光照

对于每个通道 k:

Y[:, :, k] = (m[:, :, k] - Mask\_img) / (1 - Mask\_img / A) # 颜色校正

Y = 限制在 [0, 1] 范围内

如果 bGamma 为真:

Y = Y \*\* (log(0.5) / log(Y.mean())) # gamma校正,默认不进行该操作

Y = Y \* 255

Return Y

Defog(m, r, eps, w, maxV1):

V1 = 取三通道最小值(m) # 得到暗通道图像

Dark\_Channel = zmMinFilterGray(V1, 7)

V1 = guidedfilter(V1, Dark\_Channel, r, eps) # 使用引导滤波优化

bins = 2000

ht = 直方图(V1, bins)

d = 累积直方图(ht[0]) / V1.size

对于 lmax 逆序遍历 range(bins - 1, 0, -1):

如果 d[lmax] <= 0.999:

中断循环

A = 在 V1 >= ht[1][lmax] 的地方取 m 的均值的最大值

V1 = np.minimum(V1 \* w, maxV1) # 对值范围进行限制

返回 V1, A

**前端页面模块：**

使用tkinter包实现前端页面。

例如:

实现读取图片功能。

def open\_file\_dialog(self, \*args):

file\_path = filedialog.askopenfilename(title="选择图片文件",

filetypes=[("Image files", "\*.png;\*.gif;\*.ppm;\*.pgm"

";\*.jpg")])

# print(file\_path)

if file\_path:

self.pil\_image = Image.open(file\_path)

numpy\_array = np.array(self.pil\_image)

self.cv\_image = cv.cvtColor(numpy\_array, cv.COLOR\_RGB2BGR)

self.show\_image()

1. **结果与分析**

程序的实现

HSL代码：

def HSL(img, H, S, L):

"""

HSL(色相，饱和度,亮度)调节

图像HSL调节(色调调节)

cf\_h 范围[0, 200] 对应[0, 2]

cf\_s, cf\_l 范围[0, 200] 对应 [0, 2]

整个调用过程：

对于一个img

1. hslImg = convertToHSL(img)

2. dst = HSL(hslImg, cf\_h, cf\_s, cf\_l)

3. dst = convertToBGR(dst)

"""

hls\_image = cv.cvtColor(img, cv.COLOR\_BGR2HLS)

H = np.float64(H / 5)

S = np.float64(S / 2)

L = np.float64(L / 2)

# 调整HSL

hls\_image[:, :, 0] = (hls\_image[:, :, 0] + H) % 180

hls\_image[:, :, 1] = np.clip(hls\_image[:, :, 1] + L, 0, 255)

hls\_image[:, :, 2] = np.clip(hls\_image[:, :, 2] + S, 0, 255)

result\_image = cv.cvtColor(hls\_image, cv.COLOR\_HLS2BGR)

return result\_image

锐化代码：

def LaplaceFilter(img, parameter):

# parameter 为[0,100]归一化为[0,3]

res = cv.Laplacian(img, ddepth=cv.CV\_64F)

res = np.maximum(0, res)

res = np.uint8(res)

return res

def Crisp\_Enhancement(img, parameter):

# laplace算子

parameter = np.float64(parameter / 70)

img\_laplace = LaplaceFilter(img, parameter)

# 锐化图像=原始图像+(原始图像-模糊图像)

Crisp\_enhancements = cv.convertScaleAbs(img - img\_laplace \* parameter)

return Crisp\_enhancements

平滑处理：

def Smooth\_processing(img, parameter):  
 dst = cv.GaussianBlur(img, (7, 7), parameter)  
 return dst

添加文字：

def Add\_Text(img, text, x, y):  
 """  
 为图片指定位置添加文字  
 """  
 new\_img = img.copy()  
 cv.putText(new\_img, text, (x, y), cv.FONT\_HERSHEY\_SIMPLEX, 2,  
 (0, 0, 0), 1, cv.LINE\_AA)  
 # cv.putText(new\_img, text, (x, y), (0, 0, 0), cv.FontFace('UTF-8'), 100)  
 return new\_img

去雾功能代码：

def zmMinFilterGray(src, r=7):  
 '''最小值滤波，r是滤波器半径'''  
 return cv.erode(src, np.ones((2 \* r + 1, 2 \* r + 1)))  
  
  
def guidedfilter(I, p, r, eps):  
 height, width = I.shape  
 m\_I = cv.boxFilter(I, -1, (r, r))  
 m\_p = cv.boxFilter(p, -1, (r, r))  
 m\_Ip = cv.boxFilter(I \* p, -1, (r, r))  
 cov\_Ip = m\_Ip - m\_I \* m\_p  
  
 m\_II = cv.boxFilter(I \* I, -1, (r, r))  
 var\_I = m\_II - m\_I \* m\_I  
  
 a = cov\_Ip / (var\_I + eps)  
 b = m\_p - a \* m\_I  
  
 m\_a = cv.boxFilter(a, -1, (r, r))  
 m\_b = cv.boxFilter(b, -1, (r, r))  
 return m\_a \* I + m\_b  
  
  
def Defog(m, r, eps, w, maxV1): # 输入rgb图像，值范围[0,1]  
 '''计算大气遮罩图像V1和光照值A, V1 = 1-t/A'''  
 V1 = np.min(m, 2) # 得到暗通道图像  
 Dark\_Channel = zmMinFilterGray(V1, 7)  
 # cv.imshow('20190708\_Dark',Dark\_Channel) # 查看暗通道  
 cv.waitKey(0)  
 cv.destroyAllWindows()  
  
 V1 = guidedfilter(V1, Dark\_Channel, r, eps) # 使用引导滤波优化  
 bins = 2000  
 ht = np.histogram(V1, bins) # 计算大气光照A  
 d = np.cumsum(ht[0]) / float(V1.size)  
 for lmax in range(bins - 1, 0, -1):  
 if d[lmax] <= 0.999:  
 break  
 A = np.mean(m, 2)[V1 >= ht[1][lmax]].max()  
 V1 = np.minimum(V1 \* w, maxV1) # 对值范围进行限制  
 return V1, A  
  
  
def deHaze(m, r=81, eps=0.001, w=0.95, maxV1=0.80, bGamma=False):  
 m = m / 255.0  
 Y = np.zeros(m.shape)  
 Mask\_img, A = Defog(m, r, eps, w, maxV1) # 得到遮罩图像和大气光照  
  
 for k in range(3):  
 Y[:, :, k] = (m[:, :, k] - Mask\_img) / (1 - Mask\_img / A) # 颜色校正  
 Y = np.clip(Y, 0, 1)  
 if bGamma:  
 Y = Y \*\* (np.log(0.5) / np.log(Y.mean())) # gamma校正,默认不进行该操作  
 Y = Y \* 255  
 cv.imwrite("temp.jpg", Y)  
 #return Y

前端界面：

使用点击事件来决定框的坐标:

def on\_canvas\_click(self, event):  
 # 获取点击的坐标  
 if self.crop\_mode:  
 x, y = event.x, event.y  
  
 if self.point1 and self.point2:  
 self.tk\_canvas\_draw.delete("rectangle")  
 self.point1 = None  
 self.point2 = None  
  
 # 记录两个点击点的坐标  
 if self.point1 is None:  
 self.point1 = (x, y)  
 else:  
 self.point2 = (x, y)  
 # 在两个点之间绘制矩形框  
 self.tk\_canvas\_draw.create\_rectangle(self.point1[0], self.point1[1], self.point2[0], self.point2[1],  
 outline="red",  
 width=1, tags="rectangle")

用滑块来实现随时调整：

def \_\_tk\_scale\_Light\_perception\_scale(self, parent):  
 scale = tk.Scale(parent, orient=tk.HORIZONTAL, from\_=-100, to=100, command=self.update\_Light\_perception\_image)  
 scale.place(x=43, y=32, width=511, height=36)  
 return scale

等等类似。

调试情况

主界面

****

图5.1主界面图

色调



图5.2色调模块图

分析:当提升色调/色相后整体的颜色会向绿色倾斜。而当降低色调/色相后整体的颜色会向红色倾斜。

饱和度

图5.3饱和度模块图

分析：饱和度降低会使图片变得冷色，提高会使图片变得暖色。

亮度



图5.4亮度模块图

分析：提高亮度会使图片像素点变得更高，从而变得更亮。反之。

锐化

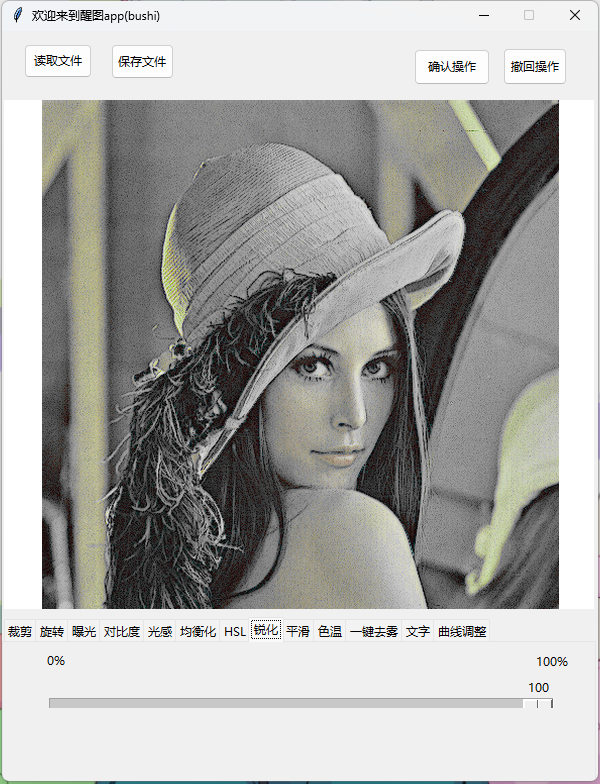


图5.5锐化模块图

分析:锐化度的提升会使图片的轮框更加的清晰，但过高会出现噪点。

平滑



图5.6平滑模块图

分析：调整平滑会使的图片更加的光滑，而过高会使的图片变得模糊。

一键去雾

去雾前：

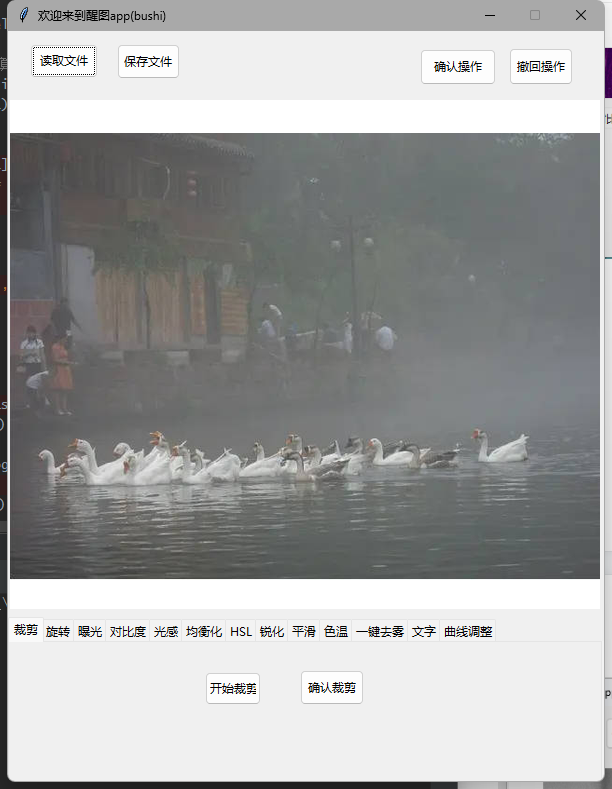


图5.7去雾前图

分析：由图可得图片中有大量的雾。

去雾后：

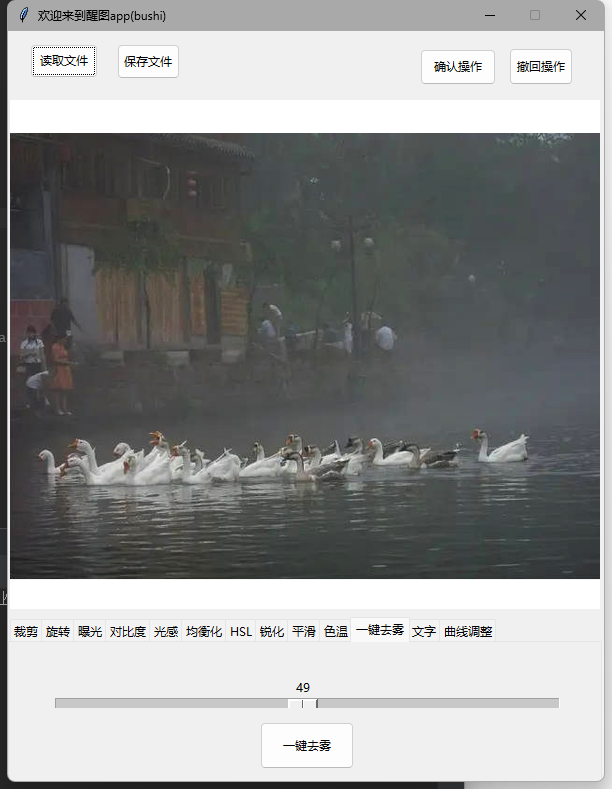


图5.8去雾后图

分析:通过暗通道法，可以得出去雾后图片整体变暗，但是雾确实去除。变暗是因为透射率比较高。

添加文字



图5.9添加文字模块图

分析:框选了位置之后，只需要选择文字就可以添加。

1. **小结：**

通过这一次课程设计学会了对照片进行较专业的处理。做了HSL功能，锐化功能，平滑功能、色调功能、文字功能、去雾功能、还设计了前端页面。

实现了去雾算法设计，使用了暗通道先验和大气光照估计的思想，结合了引导滤波进行优化。通过 Defog 函数计算了大气光照和遮罩图像。deHaze 函数通过颜色校正进行图像去雾处理。

学会了滤波器的应用，guidedfilter 函数被用于对图像进行引导滤波，用于平滑和增强图像。该函数在去雾算法中用于对遮罩图像和大气光照进行优化。

代码采用了函数的模块化设计，使得不同的功能块可以独立理解和修改。函数包含详细的注释，提高了代码的可读性和可维护性。

学会了图像处理库的使用，使用了 OpenCV 库进行图像处理，包括滤波、颜色空间转换等功能。通过库函数的调用，简化了一些复杂的图像处理操作。

掌握了参数调整和可调性来使用不同的应用，代码中的一些参数（如色相、亮度、饱和度的调整值）可以通过调整函数的参数进行灵活修改，以适应不同的应用场景。

总之，在这次的课程设计中，学到了数学信号的图像的处理，掌握了对图片处理的相关知识，这对以后的p图和摄影都有很大的帮助。