**《数字图像处理技术》大作业**

题目： 基于pyqt的综合图像美化系统

班级： 数媒2002

姓名： 李子睿

学号： 1191200226

**一、实验的总体设计方案**

本课设选题为图像美化系统，编写语言为python,任务实现使用了python-opencv,PIL，numpy库等。GUI(Graphical User Interface)图形用户界面搭建基于pyqt5的QtWidgets库来生成主界面窗口，QtCore库来完成按钮映射与事件触发，PyQt5.QtGui库来美化界面等。

本图片美化系统将支持用户读入较广类型的灰度或彩色图（bmp，jpg，png等），提供图像预处理，图像增强，目标识别与描边，图像截取与拼接，特殊美化处理，照片滤镜添加，人脸美化等一系列功能。此外，本系统还内置了学习监督模块，可通过调用电脑摄像头对使用者进行瞌睡提醒功能。

本系统有较强的用户交互性，许多功能的调用参数可由用户通过多种方式指定，尽可能满足用户在不同场景下对图像处理的需求。此外，本系统具有较强的健壮性，广泛地使用try语句捕捉错误，消息框交互提醒等手段，保证用户在误操作时，系统不会崩溃退出，尽可能提高系统的使用体验。

**二、各功能实现的算法原理**

1. **图像存取与显示**

读取文件时，利用QFileDialog下的getOpenFileName函数模块弹出对话框引导用户动态选择目标图像的路径。之后利用imread根据用户所提供的图像路径将图像读取到系统中。显示时，创建QPixmap对象与显示容器QGraphicsScene对象，将图像读入QPixmap对象后，装入显示容器中。最后调用setScene场景设置函数，即可将QGraphicsScene对象设置到QGraphicsView区域实现图片显示，将图片以可视化形式呈现给用户。显示模块会在用户每次对图像进行一次操作后被显式调用，实时更新当前图像的结果。

保存图像文件提供了两种方法，保存按钮默认用户以jpg形式导出操作后的结果图，保存路径为当前工作默认路径。另存为按钮将弹出对话框引导用户选择存储路径，存储格式支持bmp，jpg，png，jpeg等多种。

1. **图像预处理**

灰度化利用cv2.cvtColor内置函数完成与颜色翻转同样利用cv2内置函数按位取反，将每个像素点进行灰度值取反。二值化会先将图片进行灰度化操作，之后进行像素点阈值化处理，最小阈值设置为127，灰度小于127的像素点将直接被置0，另一半像素点用类似的方法置为255。

1. **图像增强**

在对进行亮度，对比度，锐度，饱和度，色调等调节时，系统会默认跳出子调节窗口滑动条slider，并通过signal.connect，实时接收用户通过滑动对以上参数的调节情况。并在用户暂时完成一次滑动后，在显示窗口显示图片当前结果，方便用户查看图像变换后的情况。

调节亮度与对比度共用一个模块，公式均为g(i,j)=αf(i,j)+(1-α)black+β，α用来调节对比度,β用来调节亮度。调节亮度时，系统将用户通过滑动条选取的参数值读入后，通过新建一张全黑基底图片，根据参数调节这张纯黑的透明度修改其变亮的程度，之后用将其与原图直接相加融合，提高原图像素点的灰度值改变亮度。对比度则不对基底图进行变换，而将参数作为加权融合的占比，改变融合后图像的对比度效果。

修改锐度时用到了二维拉普拉斯卷积核算子，该算子根据图像某个像素的周围像素到此像素的突变程度（图像像素的变化程度）进行某点像素值的改变。当邻域中心像素灰度低于它所在的领域内其它像素的平均灰度时，此中心像素的灰度应被进一步降低，当邻域中心像素灰度高于它所在的邻域内其它像素的平均灰度时，此中心像素的灰度应被进一步提高，以此实现图像的锐化处理。模板算子矩阵为八邻域，将算子卷积处理算得的值替换原(x,y)处像素点的值。传入的参数用于改变卷积核算子的矩阵中心系数值，从而改变锐化程度的大小。

修改饱和度与色相时，先将图片转换到HLS空间，直接根据传入参数对S与H分量进行定向加减，且注意各分量保持阈值0-255之间，即可完成修改。

1. **图像运算**

这部分内容包含两张图像的简单加减乘等，通过cv2库下的函数的add, subtract,addweighted函数等实现。由于调用了cv2库的函数，在融合转换前会进行从RGB色彩空间到BGR色彩空间的转换。这是由于cv2默认打开BGR空间，Qt界面显示需要RGB空间，因此用到cv2库函数的部分都需要进行空间转换的预操作，变换后的结果再进行从BGR空间向RGB空间的转换。这个颜色的转换部分是系统设计时最容易出错的部分，带来的结果就是色彩完全失真。

加减乘等操作需要严格要求图像尺寸相同，而图片融合操作考虑到了两张图片大小不一致的情况，若不相同，模块将用.shape图像操作读出长宽等参数，重新构造一张新图承接融合结果，利用白底补足空出像素。

此外，本部分内容还包括了图像的旋转，缩放功能，这两个功能都将实时都依赖changeZoom这个自定义函数，将变换结果实时呈现，接收到确定信号后才会保存当前结果于qt后台数据中。对于旋转功能，利用opencv自定义的旋转逻辑将保存图片本身的尺寸，因此旋转过程中会出现图片信息丢失与黑边的情况。因此旋转功能未利用自带模块，而是通过计算旋转角度得出旋转后图片的尺寸，调整旋转后图片的显示大小，并利用warpAffine函数补足之前存在但旋转后丢失的像素点，尽可能呈现较为合理的旋转结果。

对于对称操作，本系统提供了多种对称方式，用户可以以子菜单旋转的形式选取合适的变换方式，均用cv2.filp这个内置函数完成，变换时，同样需要进行颜色空间的两次转换（原始图读入opencv空间与结果图输出至qt空间）。

1. **直方图均衡化**

此部分包含了直方图均衡化与绘制频率直方图两个功能，可以对单通道灰度图与多通道“真彩色图像”进行操作。在进行直方图均衡化时，选用的思路是将灰度图与彩色图的实现逻辑分开进行处理，如果是“真彩色图像”，会先利用cv2.split分离BGR三个颜色通道，再分别调用均衡化函数，最后利用merge语句合并三通道至一张图中，合成的图像是用多波段图像合成的所谓“假彩色图像”。如果是灰度图，则直接调用均衡化函数。

绘制频率直方图时，同样将两类图的实现逻辑分开。对于彩色图，就初始化color = {'r', 'g', 'b'}颜色数组，分别捕获每一个通道的频谱情况，最后利用matplotlib 同时绘制多通道的直方图。灰度图则直接获取单通道的频谱情况后绘制。绘制直方图与均衡化操作分离，方便用户查看均衡化操作前后，图像的频谱变化情况。

1. **目标特征描述**

此部分包含了对图像中某一特定目标的一系列参数，包括面积，周长，矩形度，细长度，重心，球状度，圆形度等一系列参数的测量。

对于图像中目标任何参数的提取，都需要先将其轮廓进行识别，彩色图在进行轮廓识别时，需要先进行灰度转化。之后利用cv2下的findContours，识别图中轮廓最明显的物体。轮廓的检索模式为CV\_RETR\_LIS，检测到的所有的轮廓，包括内围、外围轮廓，但是检测到的轮廓不建立等级关系，彼此之间独立，没有等级关系。定义轮廓的近似方法选用CV\_CHAIN\_APPROX\_SIMPLE：仅保存轮廓的拐点信息，把所有轮廓拐点处的点保存入contours向量内，拐点与拐点之间直线段上的信息点不予保留。Point偏移量参数是所有的轮廓信息相对于原始图像对应点的偏移量，相当于在每一个检测出的轮廓点上加上该偏移量，在本模块中，偏移量参数置为None。本系统设计的目标检测被设计为只输出一个目标的参数，因此，只读取存储在向量集contours中的第一个元素（这个元素是一个含点集合）。

计算面积时，调用cv2.contourArea函数，传入参数为一系列点，即上文中所提到的contours[0]，函数将返回识别到的所有点所围成区域的面积，也正因为这个识别逻辑，当传入的目标轮廓存在较明显的噪点时，可能导致所围成的区域比真实值偏大或偏小。面积的计算单位为1个像素点的长度，此外，下文的所有参数，单位均为1个像素点的长度（含比例项无单位）。计算周长时，同样传入这个点集，调用arcLength，计算点围成区域的周长。最小外接矩形面积调用minAreaRect。

在计算重心时，我选用了OpenCV 3中更新的函数模块，connectedComponentsWithStats，这个函数具有和contourArea类似的功能，也是起到求得连通域的作用，返回的第四个参数centroids 中的第一个元素存储的就是重心的坐标。

之后便是一些比例度参数的测量，矩形度是区域面积与区域最小外接矩形面积之比，反映了目标矩形化的情况，与矩形越相似，矩形度值越往1逼近。细长度为外接矩形长与宽之比，数值可取在任何大于0的范围内，越大说明目标图形在垂直角度上看越细长。圆形度则通过公式(4 \* math.pi \* area) / (length \* length)计算，最大同样不超过1。球形度为区域最大内切圆与最小外接圆的半径之比。

Harris角模块主要选用cornerHarris函数，判断出某一点是不是图像的角点，在轮廓分割时，Harris函数选用的是Sobel 核算子，调用结束会标记出所有可能的角点。而有时用户需要最大精度的角点检测，因此我除了常规的Harris模块，还用了函数 cv2.cornerSubPix()，它可以提供亚像素级别的角点检测，减少标记点的数量。在使用这个函数时我们要定义一个迭代停止条件。当迭代次数达到或者精度条件满足后迭代就会停止。默认迭代次数为100次。我们同样需要定义进行角点搜索的邻域大小，默认在每个角点11\*11邻域内进行。传入初步Harris结果点集，传出一个含更少点的点集。低精度Harris角检测点用红点标记，亚精度角检测点用绿点标记。

检测圆形时用到了cv2.HoughCircles霍夫曼圆圈检测，霍夫曼算法首先对图像进行canny边缘检测，对边缘中的每一个非0点，通过Sobel算法计算局部梯度。那么计算得到的梯度方向，实际上就是圆切线的法线。三条法线即可确定一个圆心，同理在累加器中对圆心通过的法线进行累加，就得到了圆环的判定。为保证检测精度，我们在操作前还需对图像进行中值滤波处理，尽可能去除噪点（由于这个目标检测对轮廓要求高），否则容易出现圆的误判。此外，还应指定勾勒圆的半径范围，也是为了尽可能减少误判的概率（默认检测50-100大小的圆）。中值滤波的原理会在后部分图像平滑模块中详细展开。

轮廓识别模块同样用到了上文的contours[0]点集，只需再进行approxPolyDP操作，过滤掉同一区域内重复出现的点，即可对目标的轮廓进行简单的判断，可以判断的图形为三角形，矩形，五边形，圆。由于判断逻辑简单，因此输出的图像需具有清晰度较高，或噪点易去除的特点，否则容易判断失误。

1. **图像截取**

对于最简单的方格选择截取，思路是利用cv2.selectROI，跳出对话方框，用户操作截取范围后，返回x, y, w, h ，再对原图选取指定范围内像素点的操作[y:y+h,x:x+w]。迭代阈值分割模块在前几个模块中已经进行了解释，这里不再赘述。

区域增长算法在图形学中已经有涉及到过，这里新颖的方法是，不仅可以手动输入种子的个数，还通过plt.ginput函数手动选择种子在图像中的位置，方便用户对某特定区域分割的需求。这里采用了八邻域区域增长法（顺时针八邻域迭代循环遍历），迭代终止条件是区域内已无相似属性的新点可以遍历，或迭代次数达到阈值（阈值设置的较高，基本能满足分割需要）。除了每个种子周围的区域点集，其余所有像素点点均会在新创立的图像容器中被置为白色，迭代初步结果就是灰白的掩模图。之后，把灰白图中获取到的掩模信息保存到彩色图中，即可实现彩色图的掩模。

人脸框取与人脸图像勾勒实现手段类似，都是通过导入已被训练好的识别库中的识别函数实现，前者为用户提供多人场景下的人脸捕获，后者为用户提供针对单张脸的轮廓提取。这里以后者为例进行说明，这里主要用到了封装好的face\_recognition下的face\_landmarks函数，它为系统提供了详细的人脸识别逻辑，返回一个具有九个区域坐标点集的列表。我们只需要简单的遍历列表，就可以勾勒出包括左右眼，鼻子，下巴，嘴唇等人脸特征。这个功能能为后续人脸美化模块提供详尽的美化区域信息，做到分区域美化的功能，后续美化部分会提及。

1. **噪声添加**

噪声添加部分实现了对椒盐噪声，高斯噪声，随机噪声，泊松噪声四种噪声的添加。所有噪声添加的实现思路都是新建一副与原图同大小的黑色底图（像素点灰度值全为0），利用特定的噪声添加算法在底图上添加噪声后，与原图在有像素点的地方进行融合，完成噪声添加。

椒盐噪声是出现在随机位置、噪点深度基本固定的噪声，本系统默认灰度值添加100-150噪点深度的椒盐噪声。高斯噪声则是几乎每个点上都出现噪声、噪点深度随机的噪声，系统默认生成比率为0.05。随机噪声则是位置与深度都不确定的噪声，模拟随机的过程用到的是numpy库下的random函数，无需进行随机种子初始化。泊松噪声则是符合泊松分布的噪声模型，这个模型适合于描述单位时间内随机事件发生的次数的概率分布。直接调用numpy库下random.poisson生成，lam参数值默认为0.55，图像尺寸一定时，lam值越大，噪声出现频率越大。

1. **空域滤波处理**

这部分分为两大模块，首先是平滑滤波模块，这部分主要用来实现图片的降噪功能，减少图片中噪点的影响。中值滤波对图像进行处理是选定一部分区域（这里默认为5X5区域）像素灰度值求取其中值，用求得的中值代替区域内的像素灰度值，以此来达到平滑的效果，有可能导致图像的不连续性的特点。除了用opencv自带medianBlur函数，这里还自定义了中值滤波处理函数，方便用户进行两种处理的对比。自定义中值滤波处理中，由于有对彩色通道的分解合并，还有未经优化过的排序取中值过程，因此等待时间较长。

cv2.blur函数为均值滤波处理，使用速度较快。均值滤波不能完全消除噪声，只能相对减弱噪声。

cv2.GaussianBlur为高斯滤波处理，处理时，利用某点周围的数值进行高斯模型处理，再将处理的数值作为该点的像素值。高斯模型即二维高斯分布公式。高斯滤波与中值滤波思路类似，有能够更多的保留图像的总体灰度分布特征的特点。

而对于锐化模块，除了之前已经提到的最常用的Canny算子锐化，Sobel算子锐化,还额外添加了Prewitt算子与拉普拉斯算子锐化，不同算子的算子模板（3\*3的矩阵）与卷积手段类似，都是起到保留灰度值变化快的区域，舍弃变换慢的区域的效果。当邻域内像素灰度相同时，模板的卷积运算结果为0；当中心像素灰度高于邻域内其他像素的平均灰度时，模板的卷积运算结果为正数；当中心像素的灰度低于邻域内其他像素的平均灰度时，模板的卷积的负数。

1. **图片特殊美化效果**

特殊美化效果包括四种一键处理功能，包括浮雕，马赛克，卡通化等。

其中浮雕是根据像素与周围像素的差值确定滤波后的像素值，差别较大的像素（边缘点通常像素差别较大），在灰度图中表现为较亮，边缘凸显，形成浮雕状，然后加上一个灰度偏移值128，作为图片的整体底色。与算子锐化相同，也是对图像上的每个点卷积处理，只是锐化要求更低，因此只采用了2\*2的矩阵作为锐化算子。

毛玻璃效果是通过像素点邻域内随机像素点的颜色替代当前像素点的颜色实现的，默认偏移量为5。马赛克效果与毛玻璃的实现手段类似，只是处理幅度（偏移量）更大了，马赛克会将更大的一整块区域用一个像素点灰度值替代，使得图像更难以辨认。在使用马赛克效果时，用户可以自定义动态选择打码区域，而非对整张图进行操作。卡通化效果实现时，包括了边缘检测，减少色彩总数，滤波处理三个步骤 。其中的减少色彩总数，利用到了OpenCV库提供的K-Means聚类算法，我将其封装在color\_quantization函数中，改变k值即可改变色彩总数（默认为9种），从而改变图像卡通化的强弱效果。

加框模块包括两个功能，选择颜色加框将跳出颜色选择对话框，这基于pyqt QColorDialog.getColor语句实现。由此返回的颜色值是html语言框架下的#开头十六进制颜色字符串，因此还需要自定义颜色转换函数，将十六进制字符串转为RGB三通道的颜色分量。选择融合加框允许用户选择加入的框图，考虑到了框图与原图尺寸大小不一致的情况，此时会新建一个numpy黑色底图，尺寸大小取较大的图。之后将两张图的RGB分量分别叠加于黑色底图中。

图像拼接时考虑到了方向与大小不一致的情况，处理手段与上述类似，提示框允许用户动态选择垂直或水平拼接，功能实现基于numpy库下concatenate函数。

1. **相片滤镜与人脸美化**

此部分相片滤镜的实现与上部分特殊美化的手段类似，都是针对全图进行处理，只是这部分的滤镜只会改变图片风格，对图片细节的影响较少，因此把它们单独归为处理相片的滤镜类，供用户参考使用。

怀旧特效是将图像的RGB三个分量分别按照一定比例进行变换的结果，公式为B = int(0.273 \* r + 0.535 \* g + 0.131 \* b)，G = int(0.347 \* r + 0.683 \* g + 0.167 \* b)， R = int(0.395 \* r + 0.763 \* g + 0.188 \* b)。

光照特效是在图片重心处创建一个类似于灯光的光晕特效，图像像素值围绕光照中心点呈圆形范围内的增强。实现方式主要是通过循环遍历图像的各像素点，寻找图像的中心点，再通过计算当前点到光照中心的距离（平面坐标系中两点之间的距离），判断该距离与图像中心圆半径的大小关系，中心圆范围内的图像灰度值增大，范围外的图像灰度值保留，并结合边界范围判断生成最终的光照效果。

流年特效在图像处理中特指将原图像转换为具有时代感或岁月沉淀的特效（模拟相片自然氧化）。它将原始图像的蓝色（B）通道的像素值开根号，再乘以一个权重参数，产生最终的流年效果，其他两个通道像素保留。对蓝色通道操作的公式为B = math.sqrt(img[i,j][0]) \* 12。

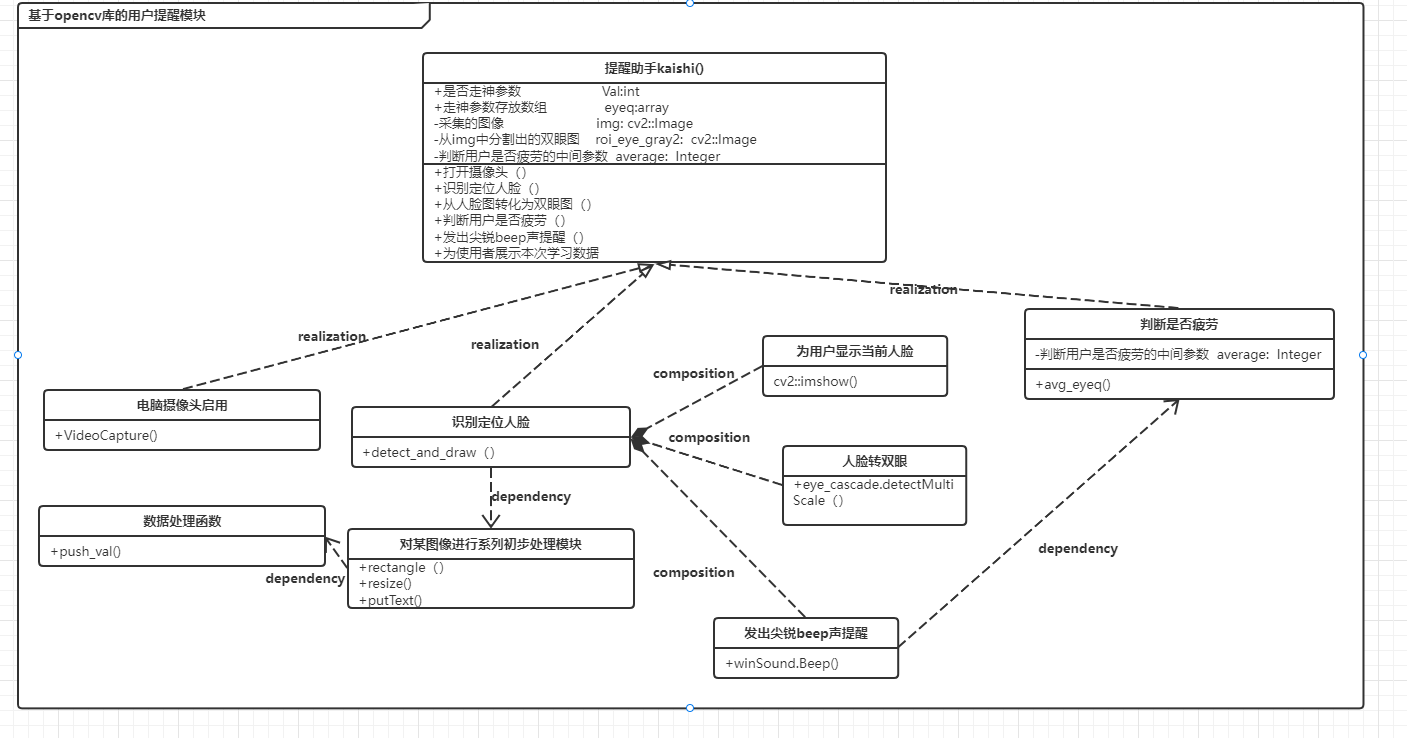
一键美化功能综合了多种图像增强手段，包括对亮度，饱和度，对比度，清晰度的增强，增强手段全部为将图片转为PIL格式，PIL库下ImageEnhance模块下的Brightness，Contrast，Color，Sharpness等方法获取到图片的亮度，对比度等信息，之后对这些信息一一使用enhance增强。对人脸细节磨皮时使用的是效率较低（比绝大多数滤波器都慢）的双边滤波器bilateralFilter，但这个滤波器滤波效果较为柔和，同时可以同时保持较好的边缘信息，适合在人脸美化时运用。由于这个模块考虑到的是一键操作，因此所有参数均为调整好的默认值，如有特殊需求，可以利用图像增强模块中的滑动条工具将亮度等信息一一调试。最后将结果图转化为pyqt空间中的图输出，即可获得到展示结果。

一键化妆功能的实现逻辑延续了人脸轮廓勾勒，同样运用了face\_recognition库下的face\_landmarks工具，获取人脸特征（眼睛，鼻子，下巴）等坐标，之后分区域进行操作。这部分除了用到了之前的ImageDraw类中的line操作用较浅的颜色勾勒轮廓外，还用到了polygon操作，对分割出的封闭多边形区域进行填色处理。一键化妆的总体思路是使左右眉毛颜色更深；用暗红颜色填充上下唇，模拟唇彩效果；双眼区域填充绿色，使得眼睛更深邃；眼眶附近用黑线勾勒，模拟眼线效果。一键化妆模块不仅可以运用于模拟化妆中，还能运用于掩藏人脸面部信息中（调整参数使得掩盖性更强）。

1. **关于与学习侦测**

此部分的访问不需要图片的打开。关于中包含了两个子选项，关于作者中提供了平台名称与版本，作者等信息。

学习侦测模块中主要运用到了cv2下的haarcascade模块中的四个函数，对采集到的彩色图像二值化处理，并主要提取双眼信息。当双眼闭上时，中间参数Val会置零，放入eyeq数组中，并触发主代码while True循环中的winSound.beep发出尖锐声提醒用户。当用户双眼正常睁开时，会利用pop语句，清除上一次中间参数，装入新的信号值，中断尖锐声。由于子函数较多，为此绘制了实现逻辑类图，类图如下所示。



**三、环境配置**

python~=3.8.6

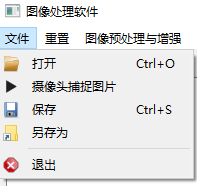
PyCharm~=Community Edition 2021.2.2

opencv-python~=4.5.5.64  
numpy~=1.22.3  
Pillow~=9.1.0  
PyQt5~=5.15.4  
PySimpleGUI~=4.60.0  
matplotlib~=3.5.2

**四、操作说明**

1. **图像存取与显示**

打开->选择任意类型图片文件/摄像头捕捉图片



**五、难点与收获**