

数字图像处理-课程大作业-人脸美颜

姓名：陈彦旭

班级：无24

模块介绍

相机畸变矫正

模块 `calibrate.py` :

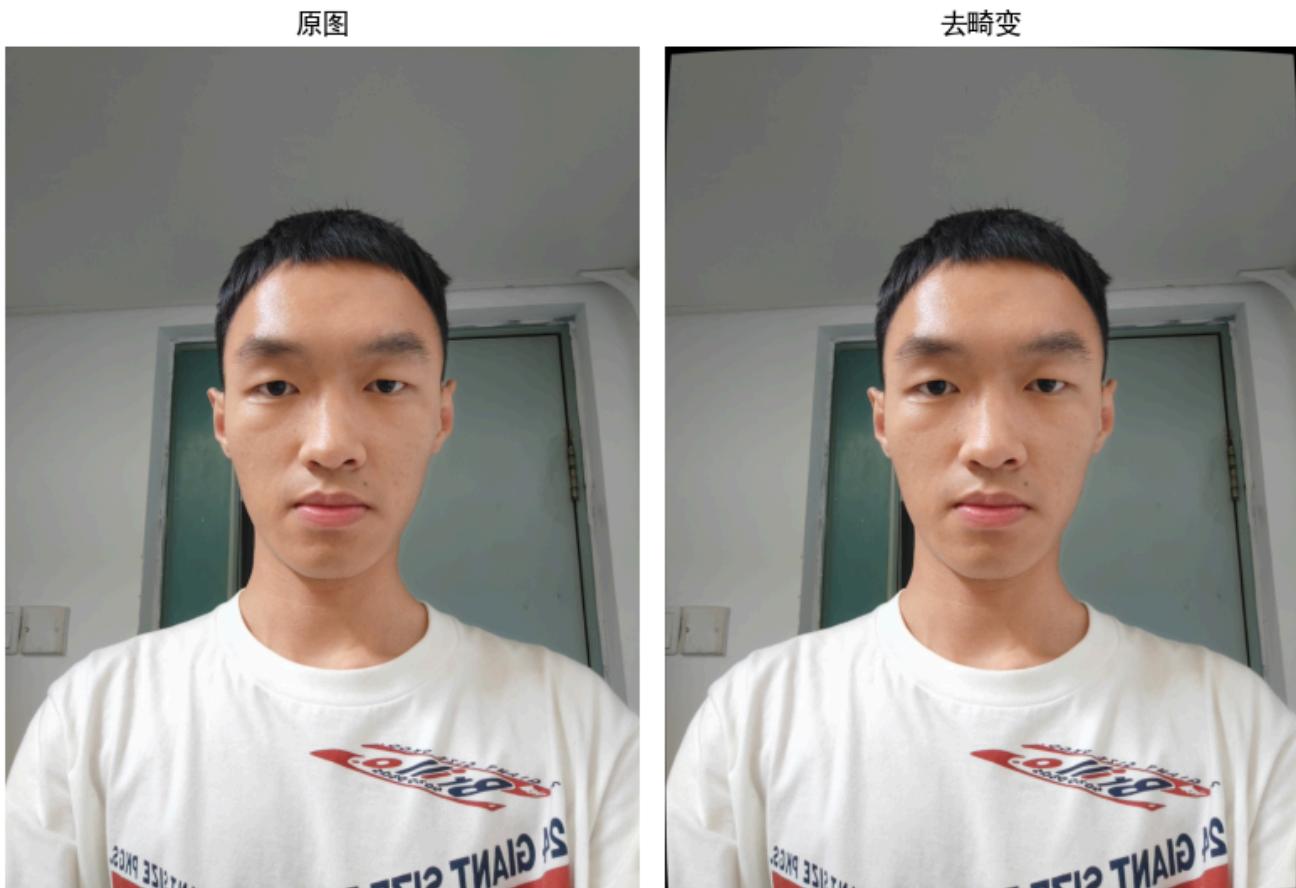
`detect_corners()` 函数，加载棋盘格图片，检测和细化图像中角点，返回各个图像中角点在真实世界坐标系中的位置，和在图像坐标系中的位置。

`calibrate_camera()` 函数，使用 `detect_corners()` 函数返回的角点位置，计算相机内参矩阵和畸变系数。

`undistort_image()` 函数，接受一张存在畸变的图像，根据内参矩阵和畸变系数计算出去畸变的图像。

拍摄棋盘格图片时应该包含角点分布在图片中间、边缘、角落等不同位置，还要包含不同棋盘格旋转角度、不同拍摄角度、不同距离等情况，才能尽可能精确求解相机内参矩阵和畸变系数。如果棋盘格的位置、角度等信息变化较小，求解时单映射矩阵 H 信息可能重复或不够丰富，条件约束较弱，或者说矩阵比较“奇异”，导致数值计算结果不够稳定。

该模块测试效果如下：



可见，使用标定相机后得到的内参矩阵和畸变系数，对自拍图像进行去畸变，图像边缘处出现了类似于枕型失真的现象，说明手机镜头直接拍摄的照片存在桶形失真。

`cv2.getOptimalNewCameraMatrix()` 函数中 `alpha` 参数可以控制去畸变后图像中“黑边”与原始图像保留的比例，如果设置为1，则保留所有原始图像的像素，图像边缘处会出现黑边；如果设置为0，则只保留那些在所有重映射中都有有效像素的位置——裁掉尽可能多的黑边。为了不影响图像亮度分布和后续直方图均衡化等全图处理，我选择设置 `alpha=0`，尽量避免黑色区域的影响。

图像预处理

模块 `preprocess.py`：

色彩空间转换：

1. `rgb_to_hsi()` 函数，将 RGB 图像转换为 HSI 图像。
2. `hsi_to_rgb()` 函数，将 HSI 图像转换为 RGB 图像。

对同一图像进行测试，连续使用 `rgb_to_hsi()` 和 `hsi_to_rgb()`，得到的图像与原图像均方误差 MSE 不超过 0.200。

2种图像高频提升：

1. `laplacian_sharpen()` 函数，拉普拉斯锐化。
2. `unsharp_mask()` 函数，反锐化掩膜。先通过高斯滤波模糊图像，去除高频信息，然后原始图像减去模糊图像，得到高频掩膜，最后把高频掩膜按照一定比例加回到原始图像，实现高频增强。

理论上，相比于拉普拉斯锐化，反锐化掩膜可以更自然地增强图像细节，避免过度锐化。但实际上现在的手机前置摄像头分辨率已经较高，尤其是人脸部分与背景部分的对比足够明显，高频提升的增强边缘的效果，对于后面边缘检测带来的改进并不明显，在视觉上也没有明显差异。

2种直方图均衡化，用于增强图像对比度：

1. `histogram_equalization()` 函数，仅对 HSV 的 V 通道进行全局均衡化。
2. `cclahe_equalization()` 函数，仅对 HSV 的 V 通道进行局部自适应均衡化。

实际测试发现，使用全局直方图均衡化时，大面积的背景影响了直方图的分布，导致均衡化之后人脸区域对比度过大，直接偏离了美颜的目标。

采用 CLAHE 自适应直方图均衡化，将图像划分成多个小块分别进行直方图均衡化，可以增强局部对比度和避免过度放大噪声。我的手机前置摄像头的图像分辨率为 (2448, 3264)，测试中发现对图像划分的网格数量为 (200, 200) 时较为合适，如果网格较多则会局部对比度过高，像漫画风格一样；如果网格数量较少则会整体偏暗，就像粘上了许多黑色污渍。

测试效果如下：



人脸检测与分割

模块 `detect_face.py` :

`get_face_mask()` 函数，检测人脸区域，得到二值化掩膜。先对 HSV 和 YCrCb 空间做阈值分割得到一个 mask，再单独对 Cr 通道做大津算法阈值分割得到另一个 mask，将两个 mask 取并集。再经过开运算、闭运算、区域连通等形态学操作去除噪点、填补空洞。最后膨胀一次，使 mask 向外延伸保证完全覆盖人脸，并使用高斯模糊平滑边缘，最后得到二值化掩膜 `face_mask`。

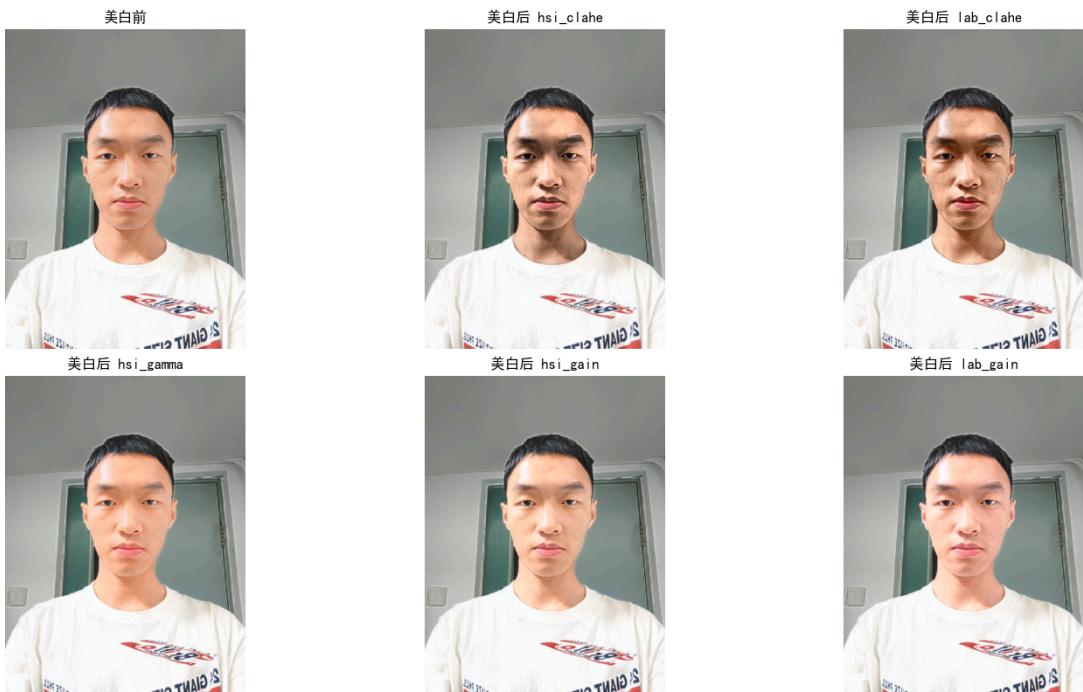
两种边缘检测：

1. `sobel_edge()` 函数，Sobel 算子边缘检测。
2. `canny_edge()` 函数，Canny 算子边缘检测。

人脸美颜

模块 `beautify.py` :

对于美白功能，可以考虑 HSI 和 LAB 色彩空间入手，因为 I 通道和 L 通道分别是亮度/明度的单独描述，其中 L 通道更接近于人眼的感知。有几种可能的方法：对 I 通道或 L 通道做 CLAHE 均衡化，增加对比度，提升亮色的比例；对 I 通道做伽马变换，提升亮色比例，抑制暗色比例；对 I 通道或 L 通道乘以一个扩大增益系数。



经过测试对比发现，如果对 L 通道做伽马变换，结果是“灾难性”的，色彩完全失真，原因可能是 L 通道对亮度的描述是非线性的，叠加上非线性的伽马变换之后，色彩映射关系“扭曲”。而“对 L 通道乘增益”函数 `whiten_lab_gain()` 与“对 I 通道做伽马变换”`whiten_hsi_gamma()` 函数效果较好，最终选择了 `whiten_lab_gain()`。不像均衡化需要对全图进行操作，这两个函数都是对像素的点操作，因此直接对面部掩膜 mask 区域操作即可，然后转换回 RGB 空间。

对于磨皮功能，去除面部区域的瑕疵和噪点。需要在平滑的同时保留边缘信息，因此使用保边滤波，选择引导滤波或者双边滤波。

函数 `smooth_skin_bilateral` 为双边滤波的实现，在邻域加权平均时，不仅考虑空间距离的权重也考虑像素值相似性的权重。

函数 `smooth_skin_guided` 为引导滤波的实现，引导图像为输入图像本身，通过最小化输入图像和输出图像的差异，在平滑区域内继续滤波，在边缘处保留信息。

经过测试，从直观感觉上来说，引导滤波效果稍好，保留边缘的同时对面部瑕疵去除效果更好。



对于瘦脸功能，仅依靠水平方向仿射变换的压缩实现，只改变脸的胖瘦，不改变脸的高度。

瘦脸函数 `slim_face()`，面部掩膜的作用仅仅是确定放缩变换的关系，`face_mask` 区域左端和右端横坐标分别为 x_{\min}, x_{\max} ，面部中心的横坐标估计为两者的平均值 $x_{\text{center}} = (x_{\min} + x_{\max})/2$ 。

如果设置瘦脸的比例为 $g < 1.0$ ，那么原图像中面部区域所在的竖状条带，即横坐标范围为 $[x_{\min}, x_{\max}]$ 的区域，宽度为 $W_{\text{face}} = x_{\max} - x_{\min}$ ，以竖轴 $x = x_{\text{center}}$ 为中心，向里压缩。

左右边界横坐标分别映射至： $x_L = x_{\text{center}} - W_{\text{face}}/2 * g$ 和 $x_R = x_{\text{center}} + W_{\text{face}}/2 * g$ 。压缩后人脸区域横坐标范围变为 $[x_L, x_R]$ 。左右两半背景区域也需要扩张，分别从 $[0, x_{\min}]$ 和 $[x_{\max}, W]$ 映射到 $[0, x_L]$ 和 $[x_R, W]$ 。

最后使用 `cv2.remap()` 构建坐标映射关系。区域边界处像素坐标连续保证边界处图像连续，区域内部使用双线性插值，最后使用高斯滤波进一步平滑。

瘦脸之后，脸部区域变形，因此不能忘记对 `face_mask` 也做 `slim_face()` 瘦脸，保证同步变化。

对于大眼功能，原本通过对原图像 `face_mask` 区域内做边缘检测和形态学操作，找到两个眼睛的连通域，计算出眼睛中心坐标和半径，在使用径向放缩的仿射变换实现大眼，但是效果并不理想，形态学操作的参数不好控制，眼部其他区域对形态学操作干扰较大。

换了一种思路的大眼，在面部区域中估算出两个眼睛的区域，用矩形框的四个坐标描述，对在矩形区域内部进行放缩的仿射变换。在此思路上有三种方法：

(1) 第一种只对新眼睛区域做映射，即使使用高斯模糊边缘，但与其他区域连接处突兀跳变，就像对原来的眼睛放大后直接贴回原图。如下图：



(2) 第二种类似于瘦脸，眼睛区域扩大，同时挤压其他背景处变小。, 虽然边界连续，但是由于与脸部连接，人脸区域也相应变胖变高，如下图：



(3) 眼睛区域范围不变，根据放大系数反向计算出放大后大区域对应的放大前的小区域，然后建立坐标映射关系进行放大。这样保证除了眼睛以外人脸其他区域不会改变，相当于取出原来眼睛区域的一个子集然后放大，这种方法需要在边缘处进行一定模糊，而且放大倍数不能过大，否则眼球会溢出眼睛区域的矩形框而截断。最终选择的是该种方法，为函数 `enlarge_eyes()`。

完整流程

完整的人脸美颜操作流水线步骤为：

1. 读取原始图像。
2. `undistort_image()` 去畸变。
3. 预处理：中值滤波 `cv2.medianBlur()`，高斯滤波 `cv2.GaussianBlur()`，高频提升 `unsharp_mask()`，CLAHE 均衡化 `clahe_equalization()`。
4. `get_face_mask()` 分割人脸区域，得到面部掩膜。
5. `whiten_lab_gain()` 美白。
6. `slim_face()` 瘦脸，同时也对掩膜瘦脸。
7. `enlarge_eyes()` 大眼。
8. `smooth_skin_guided()` 磨皮。
9. 构造一个羽化掩膜，将美颜后的人脸区域和原始图像的背景区域无缝融合，减小美颜过程中对背景的影响。

该过程各个步骤结果为：

原图



去畸变



中值滤波



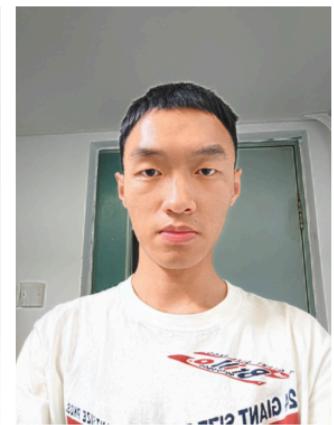
高斯滤波



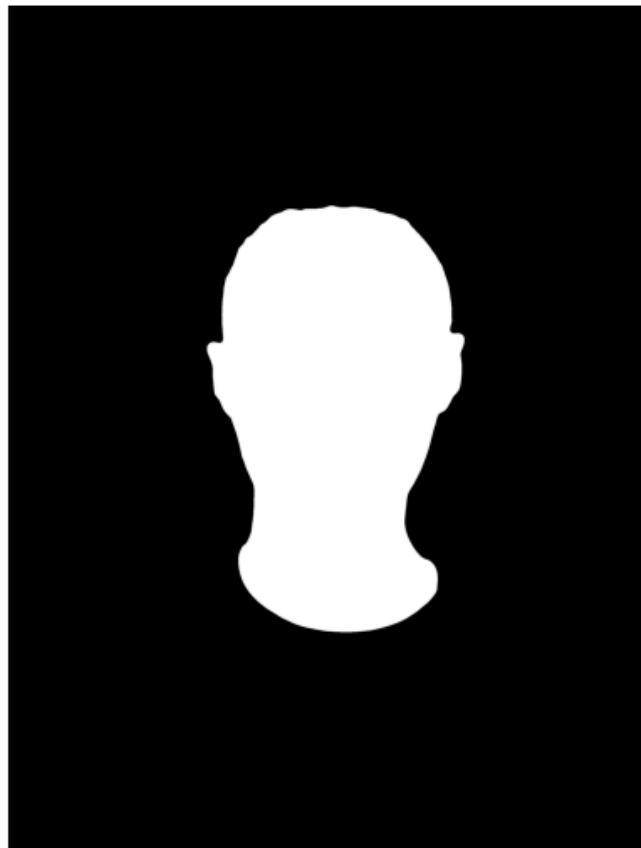
高频提升 反锐化掩膜



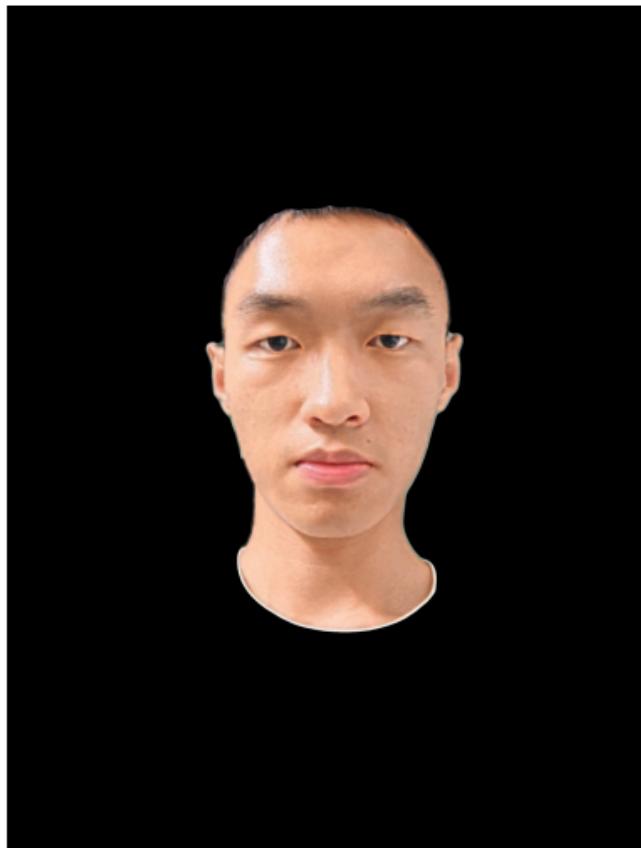
CLAHE 均衡化



人脸掩膜



人脸区域



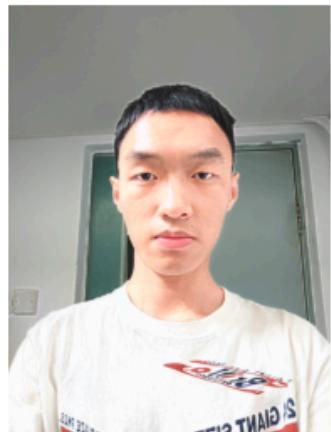
美白前



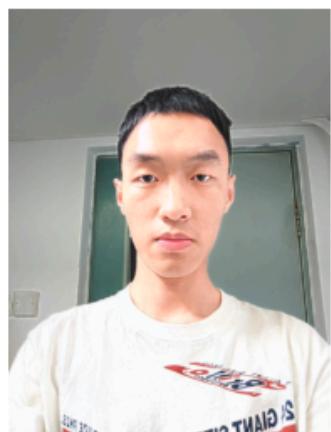
美白后



瘦脸前



瘦脸后



大眼前



大眼后



磨皮前



磨皮后





对于其他图片的处理，见 `images` 目录中的 JPG 文件，其中文件名为类似于 `p1_1.jpg` 的为相机拍摄的原始图像，文件名带有后缀 `_new.jpg` 的为经过美颜处理之后的图片。

可见，大量不同场景下经过美颜后的图片效果各异。效果较好的图片例如 `p1_7_new.jpg`, `p3_3_new.jpg`, `p4_5_new.jpg` 且原图大多数为背景较暗的图片。效果较差的图片中，背景及其亮度的干扰占主要因素，有些图片中背景颜色会被检测为人脸颜色，或者在进行人脸检测时没有办法通过形态学操作完全去除，导致人脸区域包含背景，在后续进行美白时部分背景亮度也会随之增大，形成“撕裂感”，例如 `p1_4_new.jpg`。瘦脸和大眼操作由于需要用到区域坐标范围，导致瘦脸效果不理想，以及眼睛区域定位偏差很大。另外，由于脖子也会被认为是人脸的一部分（因为颜色接近），以及不同人脸形状的差异，导致眼睛区域的难以准确定位，而本方案采用的固定参数，使得有些图片定位的人眼区域发生错位，经过放大后反而会挤压眼睛导致眼睛变形或变得更小，如 `p1_5_new.jpg`, `p2_4_new.jpg`；而有些图片由于背景的干扰或者手势的遮挡，甚至没有定位到眼睛，没有实现大眼，例如 `p4_9_new.jpg`。总体来说，这一套美颜算法受到背景、光照、遮挡以及个体差异，难以保证使用同一套参数实现稳定和理想的结果，鲁棒性不高，但是仍然在一些图像中得到了算法的较好结果和验证，证明了数字图像处理方法的可行性。