

《数 字 图 象 处 理》

综合作业 2

综合报告

班级：自 45

姓名：林子坤

学号：2014011541

目录

一、项目简介.....	3
二、程序属性.....	3
三、需求分析与设计思路.....	4
四、方法原理.....	4
4.1 同态滤波	4
4.2 形态学处理	5
4.3 坎尼边缘检测	6
五、算法设计.....	6
5.1 步骤一：去除背景.....	7
5.1.1 分离颜色（函数：ColorSeparate.m）	7
5.1.2 去除绿色背景（函数：GBAdjustion.m）	7
5.1.3 坎尼边缘检测（函数：CannyEdge.m）	7
5.1.4 提取背景（函数：Background.m）	8
5.2 去除反光	8
5.2.1 方法 1：简单颜色拼贴法（函数：Simple.m）	8
5.2.2 方法 2：多次色彩处理法（函数：Color.m）	9
5.2.3 方法 3：一次同态滤波法（函数：HomoRGB.m、HomoFilter.m）	10
5.2.4 方法 4：同态滤波升级法（函数：Advanced.m，并基于方法 3 的两个函数）	11
5.2.5 方法 5：去除反光 RGB 成分法（函数：RidBGRFilter.m）	11
六、输出结果与讨论	12
七、实验中遇到的问题和解决方案	13
参考文献.....	14

综合作业 2 去反光

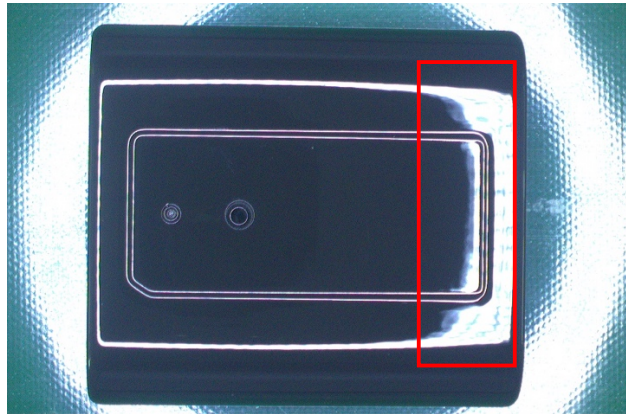
综合报告

林子坤

(自动化系 自 45 班 2014011541)

一、项目简介

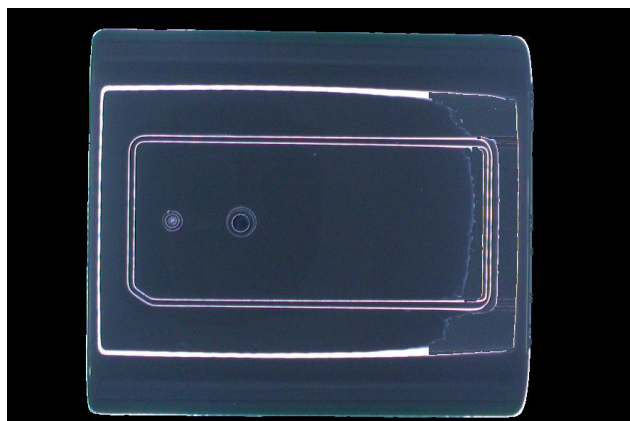
本次项目的功能是对下图工件中反光较为严重的区域（红框区域）实现去反光，并将其与背景分离，并确保图像尽可能自然。



二、程序属性

本程序使用 Matlab(R2016a, maci64)编写，“.m”文件支持的运行环境是所有装有 Matlab 的计算机。

以下是一张较为理想的生成图片。



三、需求分析与设计思路

在生活中，我们经常会遇到如题中所示的图像，由于光照问题，图像的动态范围很大，整张图像的某一部分灰度值特别大（接近纯白色[255,255,255]）。而我们要解决的问题正是如何将这片失真的区域恢复成为原有的颜色。

初读此题，本人产生了以下想法：首先，由于该工件的颜色比较均匀，符合规律，因此我们也不难推断出反光严重的区域的颜色与工件主体的颜色是差不多的，因此我们可以通过“颜色移植”的方法，直接将工件中一部分颜色剪贴到那些需要去除反光的区域上。

其次，一幅图像可以表示为其照射分量和反射分量的乘积，使用同态滤波器可以更好地控制照射分量和反射分量，即衰减高频（反射）的贡献，而增加低频（照射）的贡献，从而有效地滤除图片中的反光部分。因此，我们可以通过“同态滤波”，抑制图像中颜色较亮的部分。

最后，对于去除背景的选做任务，我产生的设想是：对于所给的十张图片，其背景几乎都是照射的白光和几乎同一色系的绿色，且这种色系的绿色在工件上几乎没有出现。因此，我们可以先将这种色系的绿色全部改为白色，从而背景几乎均变为白色，方便对其进行连通域分析并完全去除。

四、方法原理

4.1 同态滤波

注：本部分内容参考了冈萨雷斯《数字图像处理》第4章“4.9.6 同态滤波”一小节中的内容。

一幅图像 $f(x, y)$ 可以表示为其照射分量 $i(x, y)$ 和反射分量 $r(x, y)$ 的乘积，即

$$f(x, y) = i(x, y) r(x, y)$$

其中，图像的照射分量通常由慢的空间变化来表征，而反射分量往往引起突变，特别是在不同物体的连接部分。这些特性导致图像取对数后的傅里叶变换的低频成分与照射想联系，而高频成分与反射相联系，这便是我们设计同态滤波器的基础。

我们定义

$$z(x, y) = \ln i(x, y) + \ln r(x, y)$$

则有

$$\mathfrak{F}\{z(x, y)\} = \mathfrak{F}\{\ln i(x, y)\} + \mathfrak{F}\{\ln r(x, y)\}$$

或

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

我们用同态滤波器 $H(u, v)$ 对 $Z(u, v)$ 进行滤波，故有

$$S(u, v) = H(u, v)Z(u, v) = H(u, v)F_i(u, v) + H(u, v)F_r(u, v)$$

在空间域中，滤波后的图像是

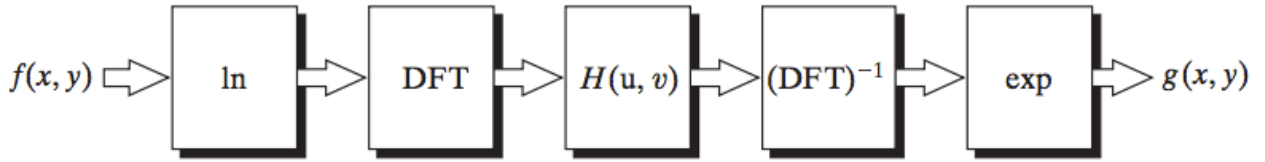
$$\begin{aligned} s(x, y) &= \mathfrak{F}^{-1}\{H(u, v)F_i(u, v)\} + \mathfrak{F}^{-1}\{H(u, v)F_r(u, v)\} \\ &= i'(x, y) + r'(x, y) \end{aligned}$$

最后，因为 $z(x, y)$ 是通过取输入图像的自然对数形成的，我们通过取滤波后的结果的指数这一反处理来形成输出图像。

$$g(x, y) = e^{s(x, y)} = e^{i'(x, y)} e^{r'(x, y)} = i_0(x, y) r_0(x, y)$$

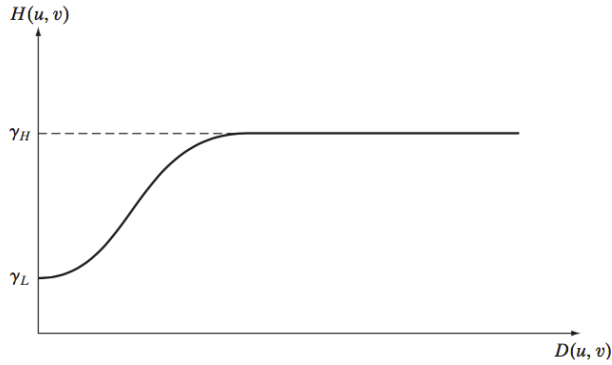
其中 $i_0(x, y)$ 和 $r_0(x, y)$ 是输出（处理后）图像的照射和反射分量。

综上所述，同态滤波方法的流程步骤如下图所示。



同态滤波方法的关键在于照射分量和反射分量的分离，其实现形式为 $Z(u, v) = F_i(u, v) + F_r(u, v)$ 。如式 $S(u, v) = H(u, v)Z(u, v) = H(u, v)F_i(u, v) + H(u, v)F_r(u, v)$ 指出的那样，同态滤波函数 $H(u, v)$ 可分别对这些分量进行操作。

同态滤波器需要制定一个滤波器函数，它可用不同的可控方法影响傅里叶变换的低频和低频分量。这个滤波器函数的剖面图如下图所示。



它可用形式稍微变换一下的高斯高通滤波器来得到。

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{-c[D^2(u, v)/D_0^2]}] + \gamma_L$$

4.2 形态学处理

注：本部分内容参考了冈萨雷斯《数字图像处理》第9章“9.2 腐蚀和膨胀”与“9.3 开操作与闭操作”两小节的内容。

在形态学图像处理中，“腐蚀”和“膨胀”是两种常用的处理方式。其中，“腐蚀”能够缩小或细化二值图像中的物体，能够将小于结构元的图像细节从图像中滤除，从而缩小一幅图像中的组成部分；“膨胀”能够“增长”或“粗化”二值图像中的物体，从而扩大一幅图像的组成部分，这种特殊的方式和粗化的宽度由所用的结构元来控制。

“开操作”指的是先腐蚀、后膨胀，其作用是平滑物体的轮廓，断开较窄的狭颈并消除细的突出物。“闭操作”指的是先膨胀、后腐蚀，也能够平滑轮廓的一部分，并能够弥合较窄的间断和细长的沟壑，消除小的孔洞，填补轮廓线中的断裂。

在 Matlab 中，可以使用 `strel` 函数来设置结构元的形状（如矩形、圆形）与尺寸（如矩形边长、圆形半径），使用 `imdilate` 函数根据设定好的结构元对图像进行膨胀，使用 `imerode` 函数根据设定好的结构元对图像进行腐蚀。

4.3 坎尼边缘检测

注：本部分内容参考了冈萨雷斯《数字图像处理》第 10 章“10.2.6 更先进的边缘检测技术”一小节中“坎尼边缘检测器”一部分的内容。

边缘检测是基于灰度突变来分割图像的最常用的方法。而坎尼边缘检测算法就是用一个台阶边缘模型推导的。在图像中，一阶导数的幅度可用于检测图像中的某个点处是否存在一个边缘；同样，二阶导数的符号可用于确定一个边缘像素位于该边缘的暗的一侧还是亮的一侧。

坎尼边缘检测基于低错误率、边缘点被很好地定位、单边的边缘点响应，是迄今为止的常用边缘检测器中最优秀的一种。主要通过以下步骤来完成：

首先，用一个高斯滤波器平滑输入图像。

其次，计算梯度幅值图像和角度图像。

再次，对梯度幅值图像应用非最大抑制，其本质是指定边缘法线的许多离散方向（梯度向量），从而细化那些边缘。

最后，用双阈值处理来减少伪边缘点，在这一步骤中，坎尼算法通过特殊的滞后阈值来改进“伪边缘”去除问题。阈值处理后，通过连接分析来检测并连接边缘。

在 Matlab 中，可以使用 `edge` 函数来调用“canny”边缘检测器。

五、算法设计

本程序分为去除背景和去除反光两个步骤。其中，去除背景将使用坎尼边缘检测和形态学图像处理的算法；去除图像的反光常常有几种目的，一是通过去除反光来获得并凸显工件真正的边缘，二是通过去除反光来还原工件原有的颜色，因此本程序去除反光的过程将针对不同的目的需求，前后共计采用了五种方法，各有优劣，将在下文进行阐述。

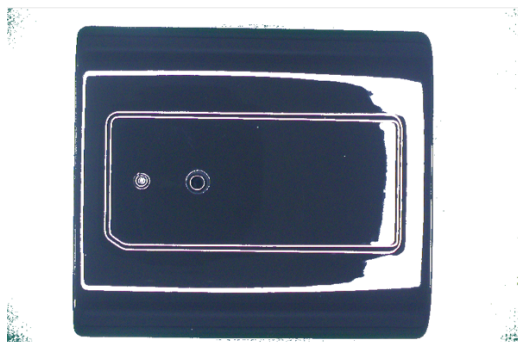
5.1 步骤一：去除背景

5.1.1 分离颜色（函数：ColorSeparate.m）

由于彩色图像有 RGBA 四个通道，则这三个图像均为 $3264 \times 2448 \times 4$ 的三维数组矩阵。而许多时候，图像的滤波处理仅能够对一个通道的图像进行操作。为了能够对这三个通道的色彩分别处理，程序将通过 ColorSeparate.m 函数将彩色图像中的三通道分别提取出来。

5.1.2 去除绿色背景（函数：GBAdjustion.m）

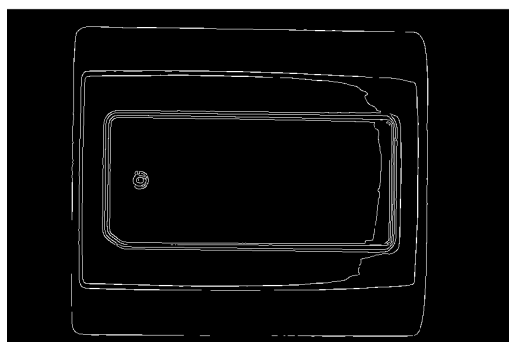
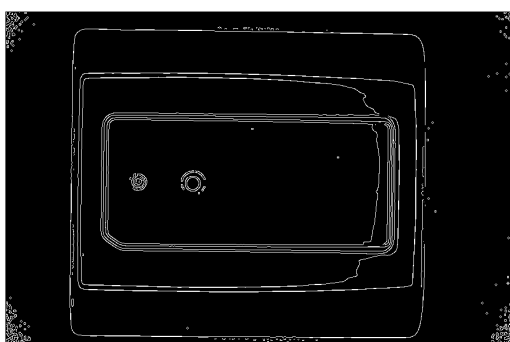
如上文所言，本图片的背景较为单一，颜色范围较为狭窄，因此可以通过一个简单的算法先行将绿色部分去除。经过多次尝试，本人发现 G+B 数值在 230~400 之间的点能够基本覆盖背景中的绿色，且与工件上的颜色几乎没有重合。因此可以将一部分赋值为白色。操作结果如下图所示。可以看出，绿色部分的背景几乎完全被去除。



5.1.3 坎尼边缘检测（函数：CannyEdge.m）

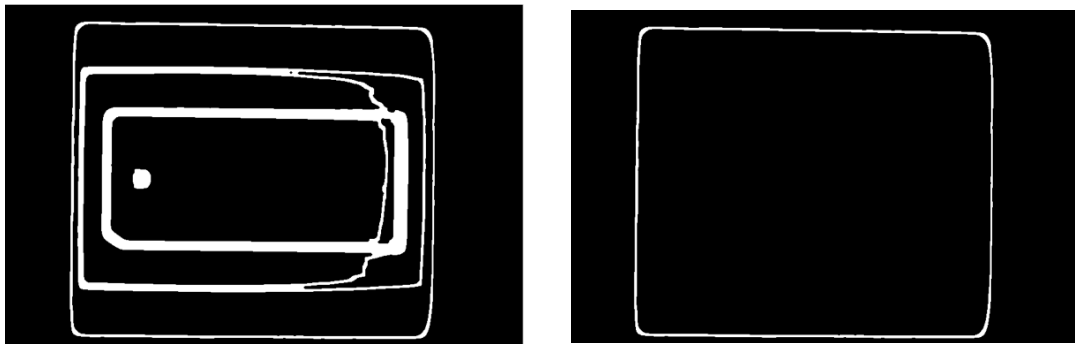
对于以上步骤处理过后的图像，其背景部分几乎完全联结。对上图进行黑白二值化处理，进行坎尼边缘检测，结果如左下图所示。

随后，利用连通域算法，消除边角处的小区域，操作结果如右下图所示。

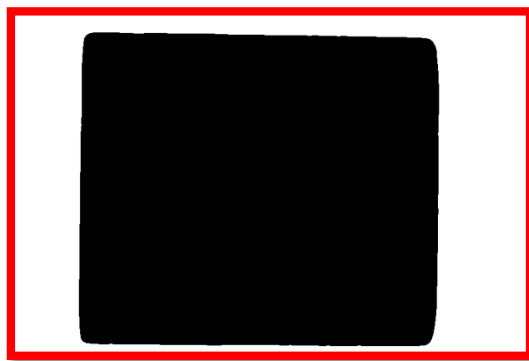


其后，对这些右上图进行闭运算操作，使得工件的边缘较为连续，能够形成一个完整的连通域，操作结果如左下图所示。

通过 regionprops 函数获取左下图的连通域信息，提取出最大的连通域即工件边缘所在的的连通域如右下图所示。



至此可以看出，右上图被工件边缘分为两个黑色的连通域，对其取反后再次进行连通域分析，可以方便地分别取出两个连通域。例如，下图白色部分为取出的背景部分连通域。（为了与背景白色区分，在下图的边缘加了红色边框）。



5.1.4 提取背景（函数：Background.m）

将原图与上图提取出的背景部分掩膜进行矩阵元素分别相乘，即可提取出图中的背景部分。获得结果如下所示。而工件部分将在下一步骤中进行进一步处理。



5.2 去除反光

根据不同的去除目标需求，去除反光共计采用以下五种方法

5.2.1 方法 1：简单颜色拼贴法（函数：Simple.m）

正如名字所述，简单颜色拼贴法的算法十分简单，其主要思想是将反光严重区域用其它颜色拼贴。对此，本人产生了几种想法：

- （1）将工件有效部分的 RGB 值进行平均，并将这个平均值赋给反光严重区域。
- （2）将反光严重区域的 RGB 值乘以一定的比例，从而降低其白色分量。

(3) 将反光严重区域的 RGB 值用工件有效部分的一个区域进行复制粘贴。

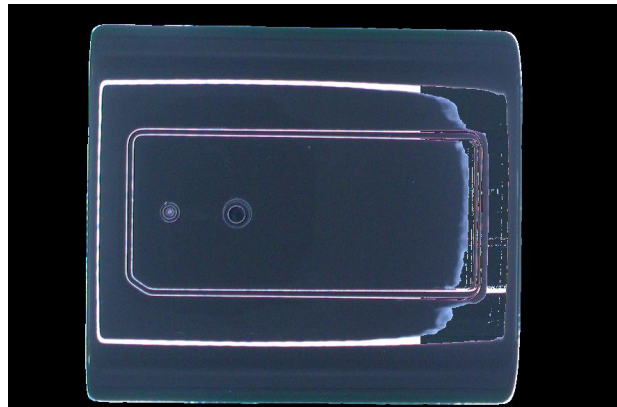
经过尝试，本人发现方法 (1) (2) 并不能得到较好的结果。分析原因如下：

(1) 工件有效部分并不是颜色均匀的，仔细观察以下 5×5 区域可以发现工件是由红绿蓝各种色调的颜色组合而成，看起来是有些“粗糙”的。如果直接将同一个值赋给反光区域，则会使这个区域的颜色“光滑”得太不自然，因此方法 (1) 获得的效果并不理想。



(2) 工件反光区域的反光过于严重，以至于基本上无法从这片反光（接近纯白色 [255,255,255]）中读取必要的原有颜色信息，因此若是简单地乘上一个比例，其所得的结果其实差不多，同样会使这个区域的颜色“光滑”得不自然，因此方法 (2) 获得的效果也并不理想。

因此，本人使用工件反光部分左侧 150 格的颜色信息作为新的输出颜色。若恰巧碰上白色边缘，则将纵坐标下移 75 格再取颜色。获得的效果如下图所示。



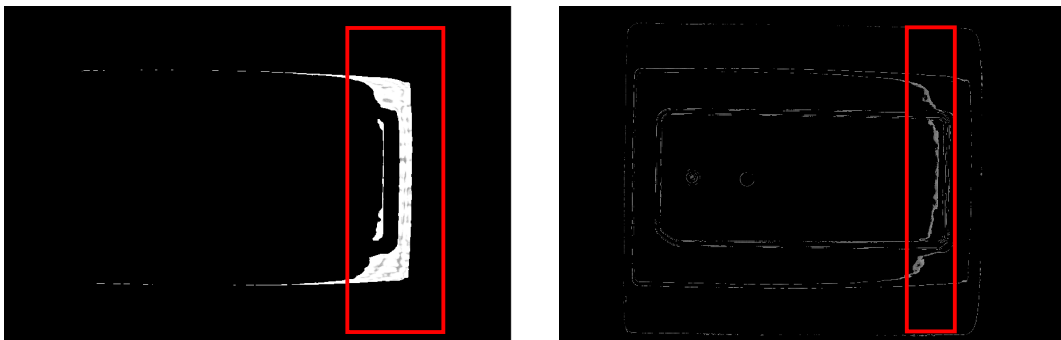
本方法算法较为简单，但也容易发现其中存在的问题：这种“一刀切”的方法对于较浅的反光没法有效地处理，对于有效的工件白边也造成了一定程度的伤害；并且，由于视觉效果影响，覆盖在工件反光区域上的颜色看起来偏暗。这个现象启发了我：仅仅设置一个阈值是完全不够的，应该分别对每个颜色范围进行分别处理，并且将处理后的结果乘以一个常数从而适应视觉效果，这便是“多次色彩处理法”的基本思想。

5.2.2 方法 2：多次色彩处理法（函数：Color.m）

为了针对反光区域的各种颜色进行分批次处理，本方法主要分为以下四个步骤：

(1) 针对生成的黑白图像 >0.75 的白色部分，即工件右侧的大片白边部分（左下图红框部分）进行处理。

(2) 针对处理后的图像，灰度值在 $0.5 \sim 0.7$ 的白色部分，即反光部分的左侧边缘（右下图红框部分）进行处理。

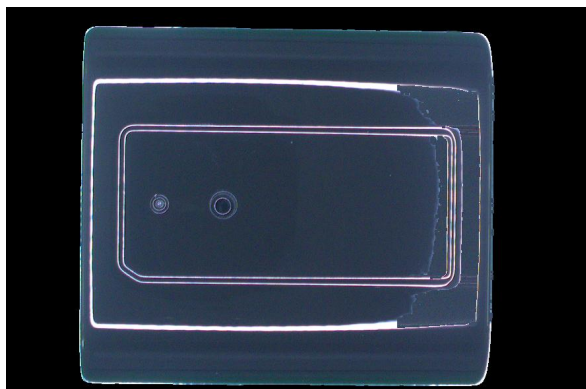


(3) 针对处理后的图像，生成的黑白图像 >0.75 的白色部分和灰度值在 $0.35\sim0.5$ 的白色部分，即下面两图的红框部分进行处理。



每一步的处理方式大同小异：提取出相应部分的反光区域、通过形态学处理获得有效的处理范围、用工件有效部分的一个区域进行复制粘贴并将 RGB 值乘以一个系数从而适合这个区域视觉效果。

最终获得的效果如下图所示。

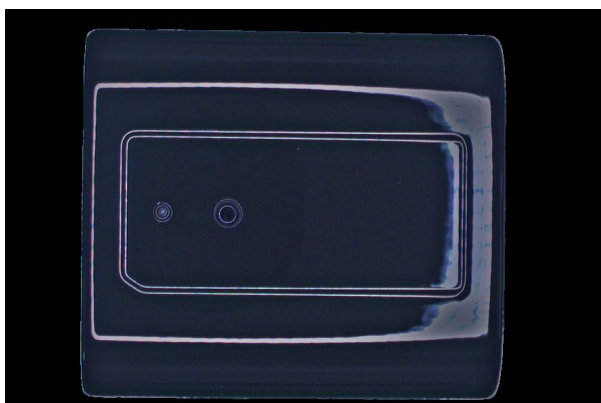


5.2.3 方法 3：一次同态滤波法（函数：HomoRGB.m、HomoFilter.m）

对于图片中的 RGB 三个通道分别使用上文所提到的高斯滤波器近似同态滤波器，公式如下，随后将三个通道组合起来获得处理后的图像。

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{-c[D^2(u, v)/D_0^2]}] + \gamma_L$$

经过多次尝试，使用经验值 $\gamma_H = 1.35$, $\gamma_L = 0.5$ ，最终获得的效果如下图所示。

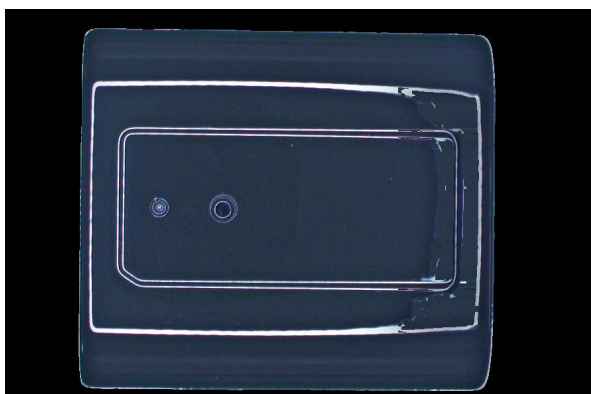


可以看出,进行一次同态滤波能够有效地削弱反光区域的影响,大大降低肉眼感觉的“亮度”。但是,我也发现了两个问题:首先,随着反光区域亮度的大幅削弱,工件有效部分的亮度也随之有小幅变暗。其次,由于原图反光部分太大、反光太强,几乎不能够从反光部分获取到什么有效的信息,因此还需要使用一定的颜色拼贴来弥补这些信息,这便是下面一种方法“同态滤波升级法”的设计初衷。

5.2.4 方法 4: 同态滤波升级法 (函数: `Advanced.m`, 并基于方法 3 的两个函数)

“同态滤波升级法”的基本思想是“二次同态滤波”与“颜色拼贴”相结合,即在一定的颜色拼贴效果达成之后再次进行同态滤波,在同态滤波结果的基础上再次进行颜色拼贴,从而逐步趋近最好的效果。

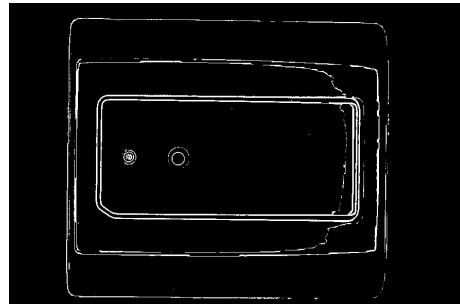
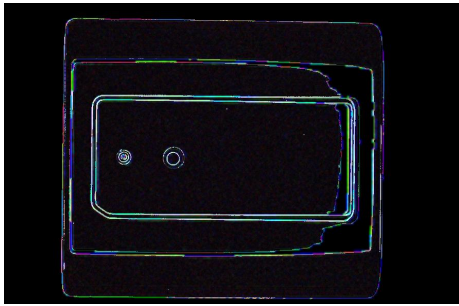
其中,进行颜色拼贴的思路过程与方法 2 基本相同,进行二次同态滤波的思路过程与方法 3 基本相同,使用经验值 $\gamma_H = 0.8$, $\gamma_L = 0.6$ 。最终获得的效果图如下图所示。



5.2.5 方法 5: 去除反光 RGB 成分法 (函数: `RidBGRFilter.m`)

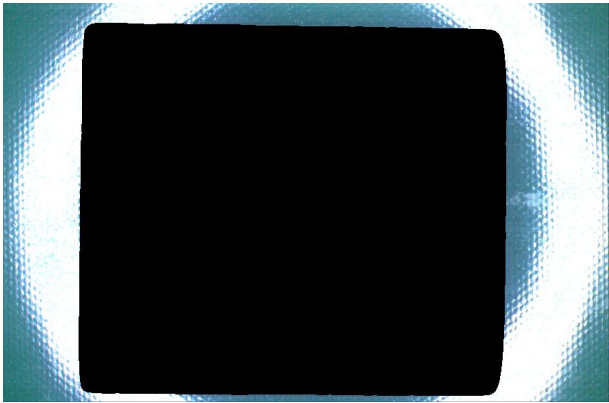
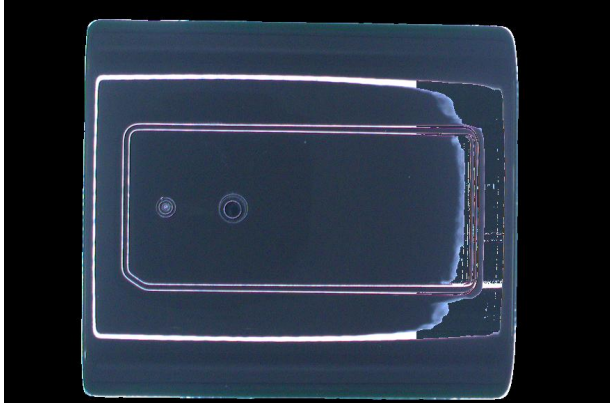
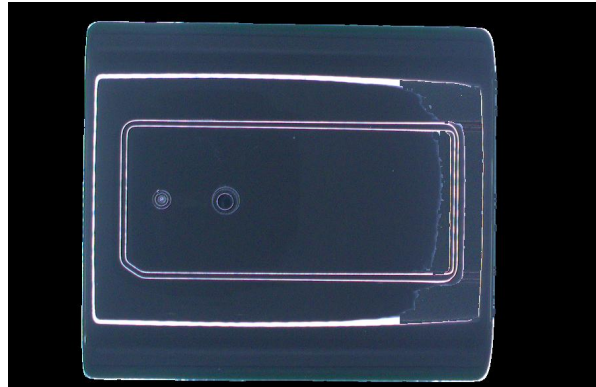
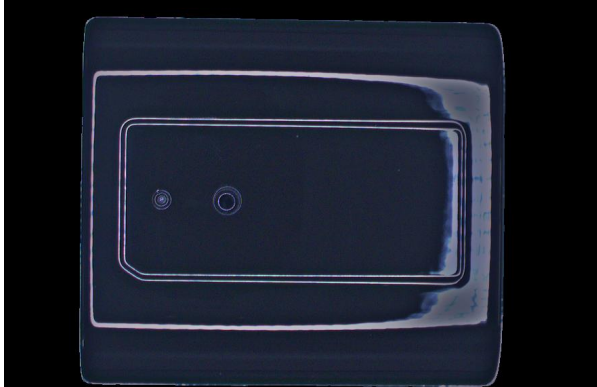
“去除反光 RGB 成分法”的基本思想是对于光照不均的图像去除不均匀的区域,并暗化图像。其中估计图像背景的照度,通过对 RGB 三个通道分别取 3×3 、 4×4 、 5×5 大小图像块中的最小值做照度,并通过双三次插值将其扩展为与原始图像大小相同的矩阵。在最后,从原始图像中减去以上矩阵去除高光照。

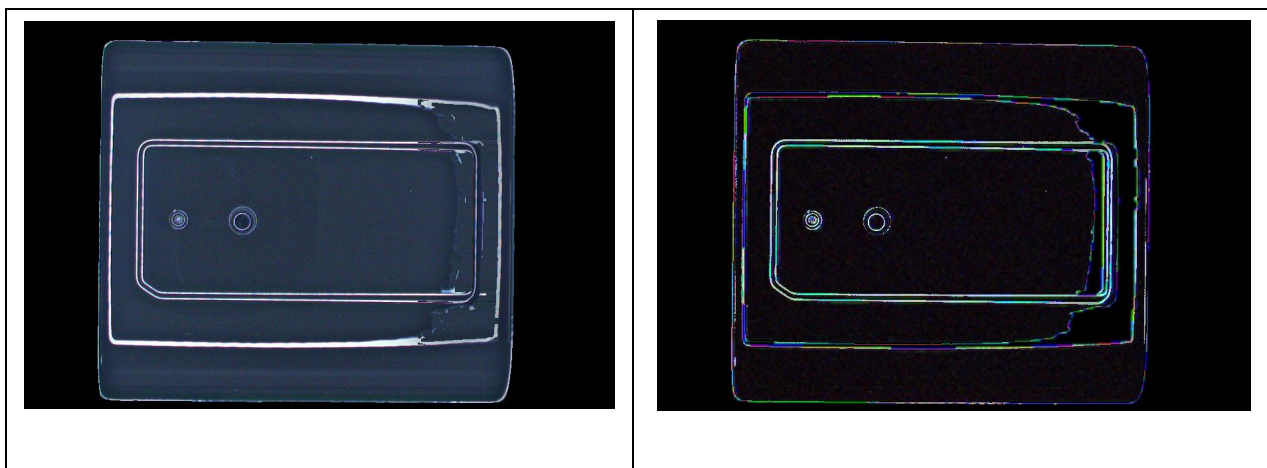
最终,反光几乎被一次性完全去除,且工件的边缘信息得到较为有效的保留,但工件的颜色存在较大的失真。最终获得的效果图如左下图所示,若对其做阈值为 0.3 的黑白图转换,最终获得的效果如右下图所示,工件部分的轮廓形态基本上得到了保留。



六、输出结果与讨论

输出效果图如下表所示。（以图 1 为例，其余输出效果图请参阅“处理结果”文件夹）

分离所得背景（1background.jpg）	简单颜色拼贴法（1simple.jpg）
	
多次色彩处理法（1color.jpg）	一次同态滤波法（1homo.jpg）
	
同态滤波升级法（1advanced.jpg）	去除反光 RGB 成分法（1rid.jpg）



通过对比和上文的算法分析，总结出各种方法各自的优缺点如下表所示。（😊表示该属性较优秀，😐表示该属性较一般，😞表示该属性相对较弱）

方法\属性	算法简单程度	有效白边保留	反光去除效果	颜色不失真
简单颜色拼贴	😊	😐	😊	😐
多次色彩处理	😞	😐	😊	😊
一次同态滤波	😐	😊	😐	😊
同态滤波升级	😞	😊	😊	😊
去除反光 RGB 成分	😐	😊	😊	😞

七、实验中遇到的问题和解决方案

本次实验过程并不是一帆风顺的，期间不免遇到了各种各样的问题。

首先，完成这次作业的过程中，“调参”成为了一个十分重要的部分。为了获得每个准确的阈值，为了尽可能保证每张图的处理效果较为良好（尤其是噪点较多的图 3），在完成本次程序过程中，以“‘运行’键被点击了接近千次”来形容调参次数应该不为过。但是看似简单的“调参”过程却时常让我们联想到数字图象处理课程中的相关知识，可谓“结合知识来调参”。对一些取值范围很大的参数，通过一两次试运行的结果和“二分法”等巧妙算法的辅助，可能的调参时间被不断缩短。

其次，这次作业实现过程也是一个不断“否定”的过程。在每次尝试一种方法后，我经常认为这种方法所得到的结果“真不错”。然而通过一定的思考我又能想出新的解决方法，可能会获得更好的效果。将这些方法加以尝试后，我总能有一种获得新大陆的快感。

最后，由于编程能力有限，在运行 Matlab 程序时还经常会出现各种办法。我的解决方案是将错误信息放到搜索引擎里搜索，并且输出相关参数来判断具体的错误是出在哪一个过程中，最终总能够排查出相应问题。

参考文献

- [1] Digital Image Processing_3rd, Rafael C.Gonzalez
- [2] <http://www.cnblogs.com/einyboy/archive/2012/08/03/2621820.html>
- [3] <http://www.ilovematlab.cn/thread-21954-1-1.html>
- [4] <http://www.linuxidc.com/Linux/2014-06/103669.htm>
- [5] http://wenku.baidu.com/link?url=g_D2hot-QEU5UDwaga1mNDmaYXopFPO9VTKGHCcxCmLBKG3XQTeIzhJ9SU7pRtdgGKQD0TCUg49dFZYISO8cla4HBjL-5UdSiZfyA2Mv4g7