滤波算法的磨皮应用

丁菲菲 史桀绮 王君珊

【摘要】尽管“磨皮”这一话题已成为日常生活中随处可见的部分，磨皮相关算法仍然有很大改进空间。本文从流行算法的缺点出发，研究滤波器在磨皮操作中的应用，并将纯粹的滤波算法、局部均方差算法与图像处理中一些常见技巧结合，以达到磨皮同时保留肤质、边缘细节的效果，改进美图秀秀等大众软件的磨皮效果。

【关键词】磨皮　滤波器　图像处理

目录

1 引言

现如今，越来越多的人习惯于在微博、朋友圈等网络媒体上分享个人照片；因此，各种用于代替 photo shop、提供给业余人群的修图软件也得到了极大的发展。但与此同时，人们对于修图软件、算法的要求也急速提高——尤其在于磨皮、滤镜两方面。然而，我们不难发现，现如今的许多软件中，磨皮算法其实还很不完善，大多数软件常常无法达到很好的磨皮效果，反而出现仿佛马赛克的块状斑点；假设将磨皮力度调整到最大，还有可能导致失真。因此，我们着眼于有关磨皮算法的研究，综合人脸识别、滤波算法与计算机视觉中图像处理的部分理论，以期达到最大程度磨皮同时保留真实感的目的。

2 动机

2.1 思路

既然明确在空间领域工作，我们不妨打开 photoshop 来找一些灵感。

假设我们只是 ps 新手。现在，面前的素色画布上有一个黑点，我们将如何去除黑点?——打开任意涂色用笔，颜色设置成背景色并覆盖掉黑点。

那么，假设我们有一个渐变色的画布，上面同样有一个突兀的黑点，又该怎么去除？

我想很多人会在黑点的周围取色，并覆盖黑点。

这样我们就得到了最初的灵感。将人脸看成色彩均匀的画布，而雀斑、痘印是画布上突兀的色块。我们需要做的，只是利用色块周围的肤色，将其覆盖、去除。

现在，我们开始正式思考算法。由磨皮的目的——去除人脸上的雀斑、痘印等而使得皮肤光滑，我们联想到图像处理中常用的滤波器。也即，将想要“磨”去的部分看做噪声，而磨皮的过程就是降噪的过程。进一步说，假设将图片转化至频域，那么我们期望得到的是平滑而规整的波形。

诚然，出去我们上文提到的“利用周围色彩”——基于空间的算法，我们还可以将图片转化到频率范围。但由于我们希望我们的算法能够不加修改的应用于各种自拍场合，对于图片的内容、特点 (尤其是斑点的特点) 没有任何限制，我们无法使用基于时间的滤波器来完成，也难以明确转换至频域后用于处理信号的冲激函数，因此，很难利用维纳滤波等基于频域的滤波器。所以，我们简单地将算法范围确定在空间域的滤波器上。

2.2 相关算法

显然，不管是网络或是正规论文等，都已经出现了许多有关滤波、磨皮的工作。

在网上的各种算法中，大部分使用了双边滤波或快速双边滤波来实现。这种算法实现较为简单，可以简单看做两个参数的高斯模糊；然而，在很多情况下，在面对轮廓线较多、边缘形状复杂 (比如耳朵等部位) 时，双边滤波对于图像的滤波程度过大，导致人脸大部分细节缺失，反而使得图片过于平面化，失真程度过高。而这显然不是我们希望看到的结果。与此同时，我们研究美图秀秀提供的几种磨皮算法，发现在普通磨皮中，采用的算法明显带有高斯模糊的特点，即磨皮区域出现油画状的色块，且整张脸带有磨砂的模糊效果；自然磨皮功能则是利用调高图像的亮度来近似减少斑点，利用肤色和斑点区域色差的减小来近似磨皮美白。这些同样不符合我们的期待。

但在这过程中，我们根据现存磨皮软件的经验，认为空间领域的磨皮算法能够满足我们的目标。同时，我们查找到图像处理相关论文，其中作者提出可以用“局部均方差”相关算法来完成磨皮相关工作。这也促成了我们接下来的实现。

3 具体算法

3.1 前提假设

在介绍具体算法之前，我们先提出几种假设

1. 在大多数情况下，使用我们算法进行处理的人像都是正面、完整的自拍。
2. 照片以 RGB 形式存储 (如 jpg 格式)
3. 只需要对于脸部区域 (不包括手臂等肤色区域) 磨皮

3.2 预处理–识别

在前文中我们已经叙述过，为了实现磨皮的效果，我们将人脸上的雀斑、痘印等看做是图片上的噪声。因此，磨皮等同于一个滤波过程。

然而，想象这样一个情境：照片中的人以一张分布着细小波点的黑白壁纸作为背景。此时，假设我们不加区分的对整张图进行滤波，一定会将背景图片的花纹当做噪声同时去除，进而导致整张图片过分失真。而这不是我们想要看到的。

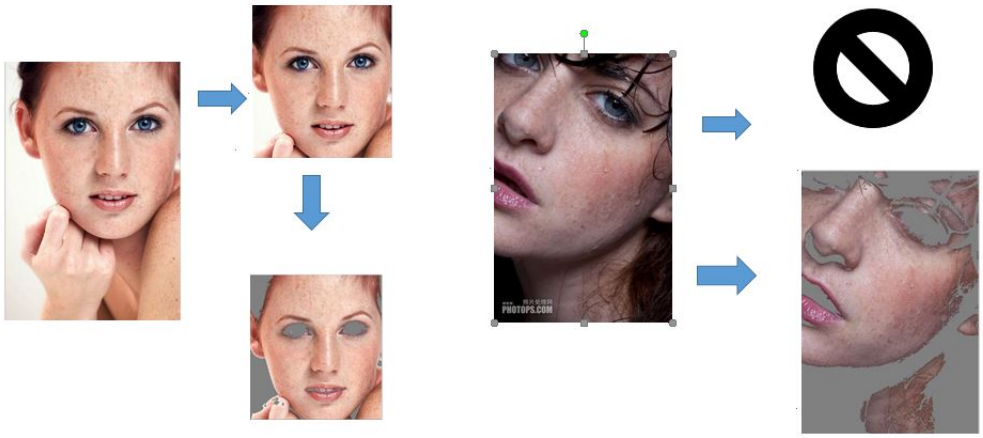
因此，在开始正式工作之前，我们需要首先对人脸进行大致的识别。

在这一步，根据假设3.1，我们不必过分要求人脸识别算法的准确度。因此，我们采用 opencv 提供的级联分类器 CascadeClassifier 来进行简单的检测工作，检测成功率约为 80%。

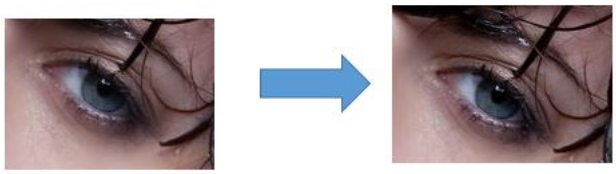
然而，简单的识别并不能够满足我们对于如眼睛、头发这样细节的要求；同时，假设人脸距离摄像头过近、被手势遮挡过多，我们将会检测不到人脸。因此，我们增加一个简单条件：只处理肤色区域的图像。基于 RGB 的肤色阈值 7 如下

1. $R \leq 95, G \leq 40, B \leq 20$
2. $R \leq G, R \leq B$
3. $max(R, G, B) - min(R, G, B) \leq 15$
4. $|R - G| \leq 15$

因此，在正式滤波之前，我们首先摘取人脸区域 (略大于检测出的人脸区域)，并扫描出肤色区域备用。



显然，当我们进行一些基于的扫描后，有关眼睛、头发的细节保留得更加完整。



3.3 磨皮–滤波算法

基于上文的分析，我们考虑这样的算法：对于色差明显的区域，与周边区域进行一定的融合；对于平滑、起伏较小的区域，认为它们是正常皮肤而减小融合比例。但同时，我们必须指出，消去斑点的同时，我们要尽量不对边缘产生破坏。

为了达到这样的目的，我们采用“局部均方差”7 的算法。

假设我们现在对于一幅 N*M 大小灰度图像进行滤波。我们用 x_{ij} 表示位于 (i,j) 处的像素值，在 (2*n+1)*(2*m+1) 窗口内部的局部平均值记为 ave, 及局部均方差记为 v，则求解 ave 与 v 的公式为：

$$ave_{ij} = \frac{1}{2m+1}\frac{1}{2n+1}\sum_{p=n-i}^{n+i}\sum_{q=m-j}^{m+j} x_{pq} \tag{1}$$

$$v_{ij} = \frac{1}{2m+1}\frac{1}{2n+1}\sum_{p=n-i}^{n+i}\sum_{q=m-j}^{m+j} (x_{pq} - ave_{pq})^2 \tag{2}$$

对于 (1) 式，我们可以使用积分图处理。即，在一个 N*M 大小的矩阵中，矩阵每个点存储的值为其左上角 (包括自身) 的所有数值的和。那么，想要求任意一个点为中心、任意大小的窗体的平均值，只需要 (右下角 + 左上角-上方-左侧) 的矩阵值即可。这样，我们将算法的复杂度成功减小到 O(M*N) 级别。

对于公式 (2)，我们可以简单运用方差公式的变形来进行处理。即：方差 = 平方和的平均 - 平均数的平方。从而将公式也简化成易于使用积分图处理的形式。因此，算法的处理时间正比于输入的数据规模。

接下来，我们利用计算得到的均方差产生加权系数
k=$\frac{v_{ij}}{v_{ij}+\sigma}$($\sigma$ 是自行设定的参数)
最终结果为 $y_{ij} = (1-k)ave_{ij} + kx_{ij}$
具体算法流程如下

Algorithm 1 localFilter

Input: 矩阵形式的 RBG 图像,m*n*3

Output: 处理后的图像

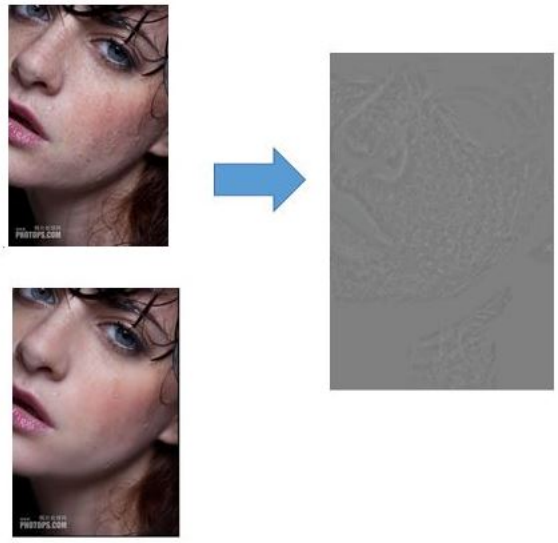
分离三个通道
for $picture \cup \{R, G, B\}$ **do**
  for $i \leftarrow 1tom$ **do**
    for $j \leftarrow 1ton$ **do**
      $sum_{1ij} \leftarrow sum_{1(i-1)j} + sum_{1i(j-1)} - sum_{1(i-1)(j-1)} + picture_{ij}$
      $sum_{2ij} \leftarrow sum_{2(i-1)j} + sum_{2i(j-1)} - sum_{2(i-1)(j-1)} + picture_{ij}^2$
    end for
  end for
  for $i \leftarrow 1tom$ **do**
    for $j \leftarrow 1ton$ **do**
      $ave_{ij} \leftarrow \frac{1}{2m+1}\frac{1}{2n+1}(sum_{1(i+m)(j+n)} + sum_{1(i-m)(j-n)} - sum_{1(i+m)(j-n)} - sum1(i-m)(j+n))$
    end for
  end for
  for $i \leftarrow 1tom$ **do**
    for $j \leftarrow 1ton$ **do**
      $v_{ij} \leftarrow \frac{1}{2m+1}\frac{1}{2n+1}(sum_{2(i+m)(j+n)} + sum_{2(i-m)(j-n)} - sum_{2(i+m)(j-n)} - sum2(i-m)(j+n)) - ave_{ij}^2$
      $k_{ij} \leftarrow \frac{v_{ij}}{v_{ij}+\sigma}$
      $y_{ij} \leftarrow (1-k)ave_{ij} + kx_{ij}$
    end for
  end for
end for
合并三通道

3.4 进一步降噪

基于上一步完成的图片，我们利用图像处理的一个常用手法，进行进一步的降噪——高反差保留。简单来说，我们利用 $lap = localOut - src + 128$ 公式，将上一步得到的图片按像素减去原图并加上中兴灰色值，得到一张浮雕效果的图片。该公式的意义在于，所有平滑区域，即在滤波过程中改变较小的区域，在得到的“浮雕”中均显示为灰色平滑色块；而色彩相差较大的区域，如雀斑、轮廓线，则由于相减后的区别较大而成为突出部分。参数 128 是图像处理中一个重要的参数，被称为中性灰色 (像素范围为 0-255)。在我们即将使用的线性光融合中，掩膜层像素值为 128 的部分不会对底图产生任何影响。

假设我们得到的图片为 lap，在下一步中，我们将 lap 与原图进行融合。

显然，简单的叠加是没有任何改进效果的。在这里，我们采用前文提及的“线性光”的叠加方式：以原图作为基准，与 lap 混合。公式为 $out = src + 2 * lap - 255$ 该公式的意义同样易于理解。当混合色 (即 lap) 数值比中性色 (即 128) 暗时，src 的数值减小，减小基准的亮度；当混合色数值亮于中性色，增加基准的亮度。



通过这样简单的工作，我们进一步抹去了图像上的噪声并保留了边缘。这部分中，我们得到的图片如下：

3.5 后期处理–自然融合

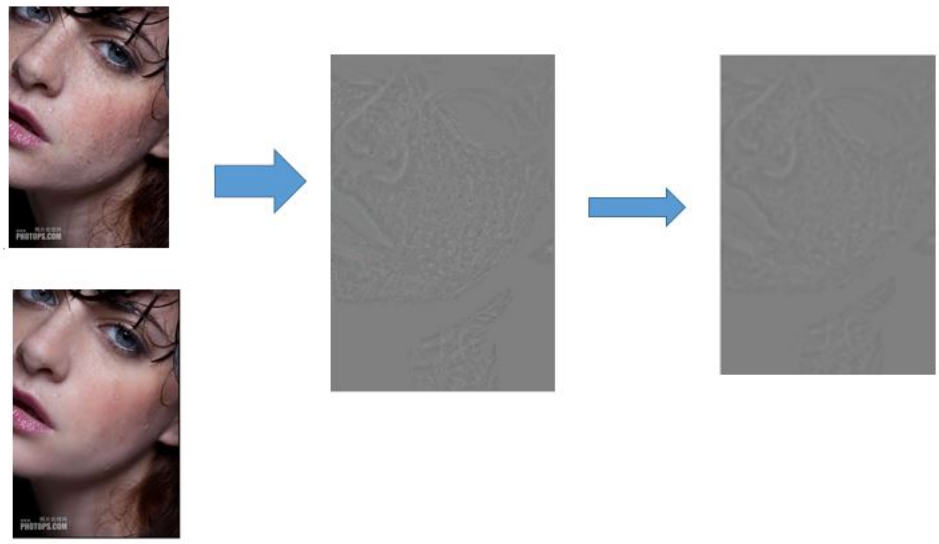
一般来说，磨皮之后的图片之所以与实际人脸相差太大，是因为肤质相差过大导致与真实情况严重不符 (比如粗糙的皮肤磨皮后变得过于平滑、近于油画效果，且丢失自然的腮红等色彩)，或者模糊过度而丢失三维信息 (比如边缘信息丢失、明暗对比降低而如同油画效果)。因此，我们从这两方面来考虑后期修饰。

保留肤质

在上一步的高反差保留部分，我们将 lap 与原图叠加；在这一步中，我们没有区分噪声与脸上的纹路、边缘等肤质相关信息，因此容易发现，产生的结果虽然比作用前“光滑”，但同样显得丢失部分真实性。

因此，我们对这一部分进行改进。最简单的改进，即在线性光叠加前，对 lap 层进行一次高斯模糊，原理为根据空间位置的远近进行加权，并加中心像素的值改变为加权后的和，从而使“浮雕”变得柔和，让 lap 层边缘位置对于图像的影响变小；以丢失部分边缘信息为代价，换取肤质的保留。

显然，此时高斯模糊的半径十分重要。半径过小时，由于进行平均的窗体过小，无法对边缘进行恰当的模糊而无法体现细节，过大又会导致 lap 层效果过于微小，磨皮效果不明显。因此，在多次试验后，我们选择 3*3，5*5，7*7 三个窗口大小，并选取 $\sigma = 5$ 作为模糊参数。



从左到右高斯模糊窗体大小分别为 0，3*3,5*5,7*7；可以明显看出，7*7 的图像上脸颊两侧已经出现粗糙的斑点

从左到右高斯模糊窗体大小分别为 0，3*3,5*5；同时可以观察到，在模糊半径增加的同时，以眼周的纹路为例，图像肤质越发接近真实

同时，我们提供了一个融合函数，将处理后的图片与输入的原图进行简单叠加，以达到保留细节的目的。

高斯模糊的过程大致如下

Algorithm 2 gaussianFilter

Input: 原图片 src，参数 sigma，高斯半径 r

Output: 处理后的图像

sum←0,dst1←src
for x ← 0 to rows-1 **do**
  for y ← 0 to cols-1 **do**
    对于 r*r 大小的区域计算像素 * 空间权重和，三个通道分别计算出 $\text{sum}_{r(b/g)}$
  end for
  存储 $\frac{sum_{r(b/g)}}{sum}$ 的中间值，得到中间图片
end for
for x ← 0 to rows-1 **do**
  for y ← 0 to cols-1 **do**
    对于 r*r 大小的区域计算中间图片像素 * 空间权重和，三个通道分别计算出 $\text{sum}_{r(b/g)}$
  end for
  存储 $\frac{sum_{r(b/g)}}{k}$ 的值，得到目标图片
end for

边缘强化

在使用诸如 photoshop 的软件进行人像处理时，很多人会重复“液化——锐化——液化——锐化”这

样的过程，也就是“磨皮——强化——磨皮——强化”。因此，我们受启发，加入 USM 锐化功能。

在该功能模块中，我们再一次使用高反差对比技术–假设上一步得到的图片为输入图片 filterOut，将其与高斯模糊得到的图片相减并求绝对值。这里我们考虑到锐化程度不应该过大而导致失真，将高斯模糊窗口大小、两方向的参数都设置为 3；由于针对每个像素点，我们仅仅需要“它是否满足一定的条件以成为轮廓边缘”，因此，我们创造高反差矩阵 lowContrastMask，存储内容为“上一步得到图片像素值小于设定的阈值”这一判断语句 (阈值可选，这里我们设置为 1)。

而后，我们首先设置一个 mask=filterOut*(1+ 用户输入参数/100.0f)+gausImg*(-参数/100.0f) 作为中间矩阵, 代表原图与高斯模糊图片的融合程度。并将其对应点放置到“高反差图片值为 0”对应的原图像素，即我们判断的边缘区域处。简单地说，假设我们判断出某个像素点属于边缘，我们就使用 mask 来代替输入图像中的点。

具体过程参照伪代码

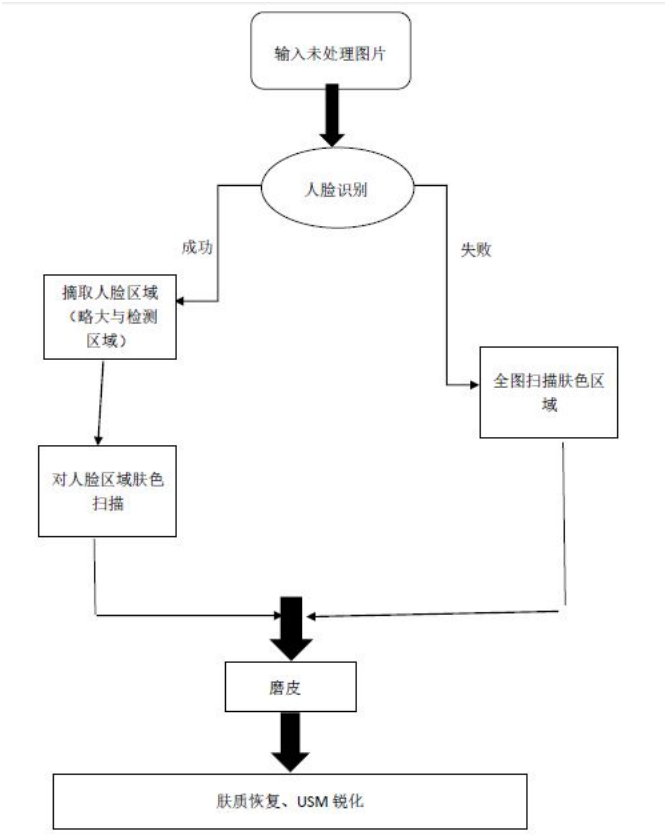
Algorithm 3 USM

Input: 矩阵形式的 RBG 图像 filterOut，用户输入 nAmount

Output: 处理后的图像 dst

1: sigma←3,threshold←1
2: amount← $\frac{nAmount}{100.0f}$
3: GaussianBlur(filterOut, gausImg, Size(), 3, 3)
4: lowContrastMask = $|filterOut - gausImg| < threshold$
5: dst = filterOut*(1 + amount) + gausImg*(-amount)
6: filterOut.copyTo(dst, lowContrastMask)

3.6 总体流程



如图，我们的磨皮算法主要内容即为上述三部分——识别，磨皮，后期处理。伪代码如下

Algorithm 4 process

Input: 矩阵形式的 RBG 图像 src, 高斯模糊窗口大小 s, 融合度 α,USM 锐化参数 nAmount

Output: 处理后的图像 dst

对图像进行人脸识别
if 识别出人脸 **then**
    摘取人脸区域处理
else
    摘取全图处理并发出提示
end if
对摘取区域肤色识别并摘取肤色区域 skin
localFilter(skin,skin) 1
tmp = skin - src + 128
gaussianBlur(tmp, lap, Size(s,s),5,5)
out = src + 2*lap -255
out = $\alpha src + \beta out$
USM(out,nAmount, dst)3

4 效率分析与软件实现

4.1 算法时间复杂度

上文提到，我们的磨皮算法主要内容分为三个部分——识别，磨皮，后期处理，下面逐个分析它们的时间复杂度。不妨设 n 为图像的像素数量。

首先，显然能够发现，由于我们对于每个像素点比较一次 RBG 颜色，肤色识别的时间复杂度为 O(n)。

磨皮部分的局部均方差算法，在前文中已经解释过，由于使用了积分图进行处理，因此只需要一次扫描存储中间值与一次加减操作，时间复杂度也只有 O(n)。

后期处理中除了按像素进行的简单加减，主要运用了高斯模糊算法。虽然高斯模糊的时间复杂度是 O(m*n)，但由于我们选取的 m(即模糊窗口大小) 远小于 n，故时间复杂度也可近似为 O(n)。因此，依托于高斯模糊的锐化操作时间复杂度也是 O(n)。

至于前文提及的人脸识别部分，cascadeclassifier 在实现过程中同样利用积分图进行处理，因此我们简单将其复杂度视为 O(n)。

综上所述，我们算法的理论时间复杂度为 O(n)。

接下来，给出用我们的算法处理几张示例图像所需要的时间

编号	图像大小 (像素)	运行时间 (ms)
1	296*417	1141
2	391*574	1191
3	631*474	1256
4	512*512	1224

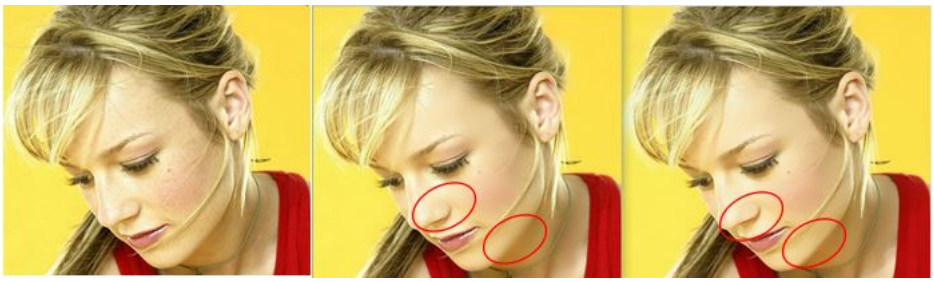
4.2 算法最终效果

除了时间复杂度的对比外，我们给出两组对比图，分别对应大面积磨皮与复杂背景下小范围处理效果。从左到右依次为原图、本文中的算法与美图秀秀磨皮效果。

在本次对比中，我们不难看出，在保留脸部水珠效果、嘴唇纹理与发丝效果等细节的同时，我们的算法对于斑点的去除更加平滑、柔和。尤其在脸侧、鼻头、下颌等区域，几乎可以完全去除雀斑带来的黑色噪声。

在画面整体色调一致、需要磨皮区域与周围色彩相差不大时，我们的算法仍然能够保持优秀的效果。

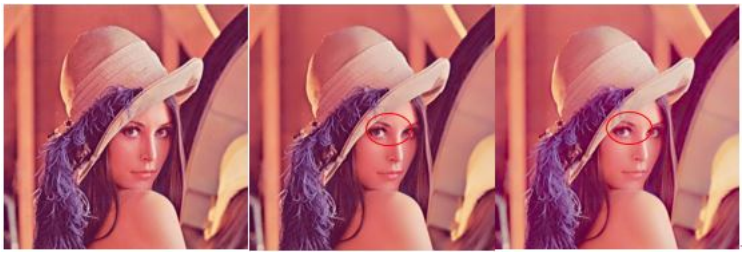
同时，假如我们放大图片观察，还能够发现，由于加入了一些柔和、恢复肤质的效果，我们在诸如脸部轮廓、鼻尖这样拥有明显轮廓线的区域，在牺牲一定对比度的情况下，使得图片效果更加逼真，减小了由于过度磨皮带来的扁平化效果。



对于本身已经不需要过度磨皮、且拥有较为复杂背景的图片，我们的算法同样拥有较好效果。

在本次对比中，本文算法较好保留了人脸、皮肤区域以外的背景部分；同时，由于肤色部分本身已经较为均匀，算法未对图像进行大幅度处理；对比容易看出，美图秀秀处理后的图片整体变白，阴影部分也

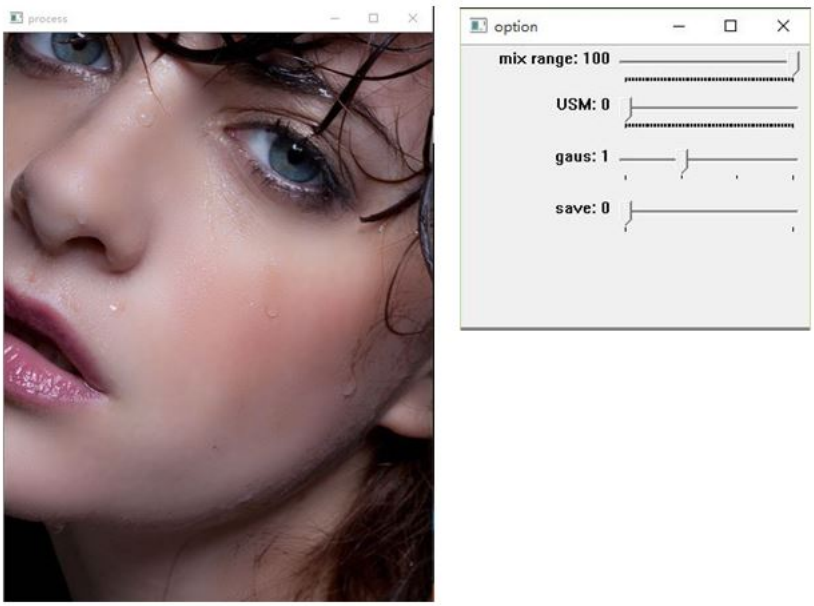
失去强烈对比效果，对于尤为重要的眉眼区域，也因为过度美白而失去细节。



4.3 软件实现

我们提供了一个简单的用户界面。包括：

1. option 窗口，包含四个进度条，分别为上文提到的融合度 (0-100)、USM 锐化程度 (0-100)、高斯参数 (0-3, 对应高斯窗口大小 1,3,5,7) 与是否保存图片 (从 0 变为 1 为保存操作)。
2. process 窗口，实时显示不同参数对应磨皮效果



输入为控制台输入，格式为程序名称 (beauty) 图片名称 (xx.jpg)

同时，利用 androidStudio，我们将算法移植到 android 手机上，具体界面与 PC 端相似；由于手机屏幕大小限制，我们将 option 与 process 窗口融合，进度条放在最上方，下方为图片处理情况

5 未来预期

显然，我们的算法还存在很大的不足。比如，当我们面对一张色斑过于明显、过于密集的图片时，我们的算法效果有很大的可能比不上双边滤波的强力磨皮。同时，假设我们不考虑时间复杂度，BM3D 等算

法也拥有十分漂亮的结果。同样，在前文我们提及过，频率范围上的一些算法，在掌握相关刺激函数时，也能在保证一定效率的同时达到磨皮的目标。

然而，由于一些综合考虑 (比如一般情况下双边滤波常常过度磨皮，而 BM3D 算法过于复杂也过慢，频率相关难以决定参数)，我们最终选择了局部均方。但在未来的工作中，我们将尝试结合空间与频率范围的算法，并尝试利用双边滤波、BM3D 的部分原理，改进我们的算法。最简单的选择是，我们可以添加某种选择与存储功能，使得用户自行选择使用强效磨皮还是自然磨皮来解决。

同时，我们将进一步解决有关人脸识别的问题。比如定位不准，或是磨皮之后的区域与背景区域融合出现突兀边缘的情况。

6 结论

在本文的算法中，我们基于磨皮的理论含义与空间特性，使用一些基础的滤波算法改进现有的磨皮效果。在保证时间效率的同时，我们初步达到了保留细节、肤质同时最大效果磨皮的目标。由于算法的快速、简洁、低要求、最终程序的易上手，我们认为它满足现实自拍的大部分要求。

7 参考文献

1. J Kovač, P Peer, F Solina,$Humanskincolorclustering for facedetection$,ieeexplore.ieee.org,2003

2. J.S. Lee, $Digitalimageenhancementandnoisefilteringbyuseoflocalstatistics$, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(2):165-168, 1980.