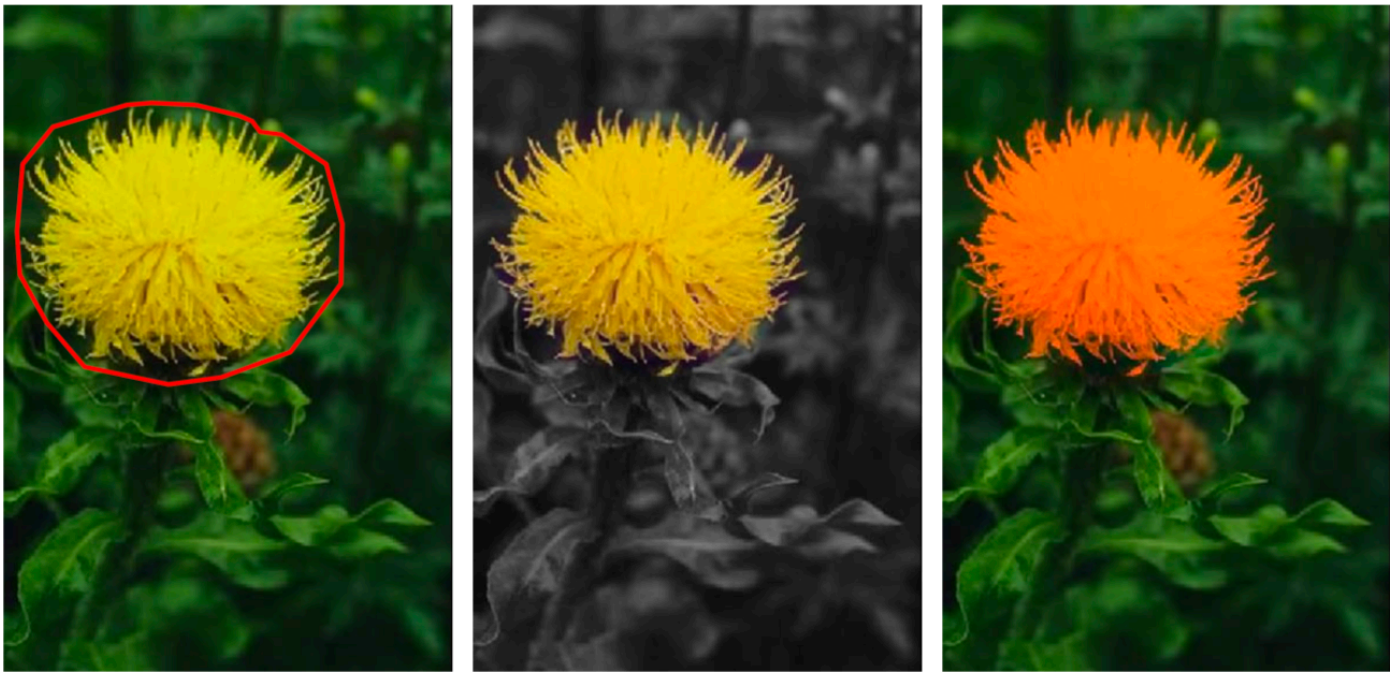
泊松图像编辑



本文同步发表在我的微信公众号“计算摄影学”，欢迎扫码关注

http://weixin.qq.com/r/Ey4lPRPEhE77rVde93sB (二维码自动识别)

【转载请注明来源和作者】

正如在8. 图像处理的应用 - 动作放大中所讲述的动作放大技术当初让我大吃一惊一样，在图像合成和融合这个领域也有让我一接触到就感觉很震撼的技术，这就是泊松融合，以及后面还会讲述的GradientShop技术。

一. 泊松融合的惊人之处

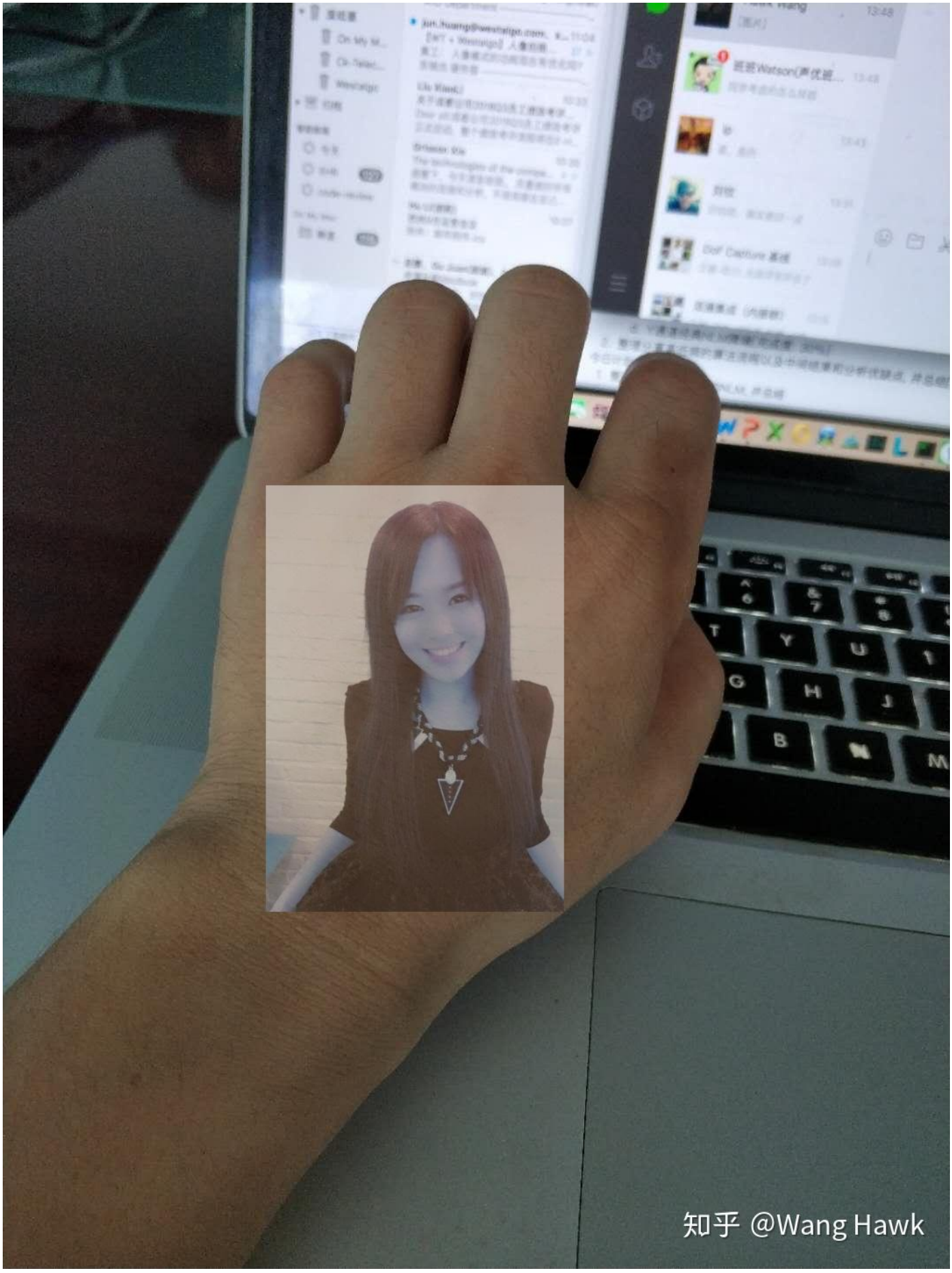
让我们看一个有趣的例子：如果我想把苍老师纹到我的手上，应该怎么利用图像融合技术?

我们之前已经学到了从cut-and-paste到多频带融合等图像的合成和融合技术。它们各自都有一些缺点。

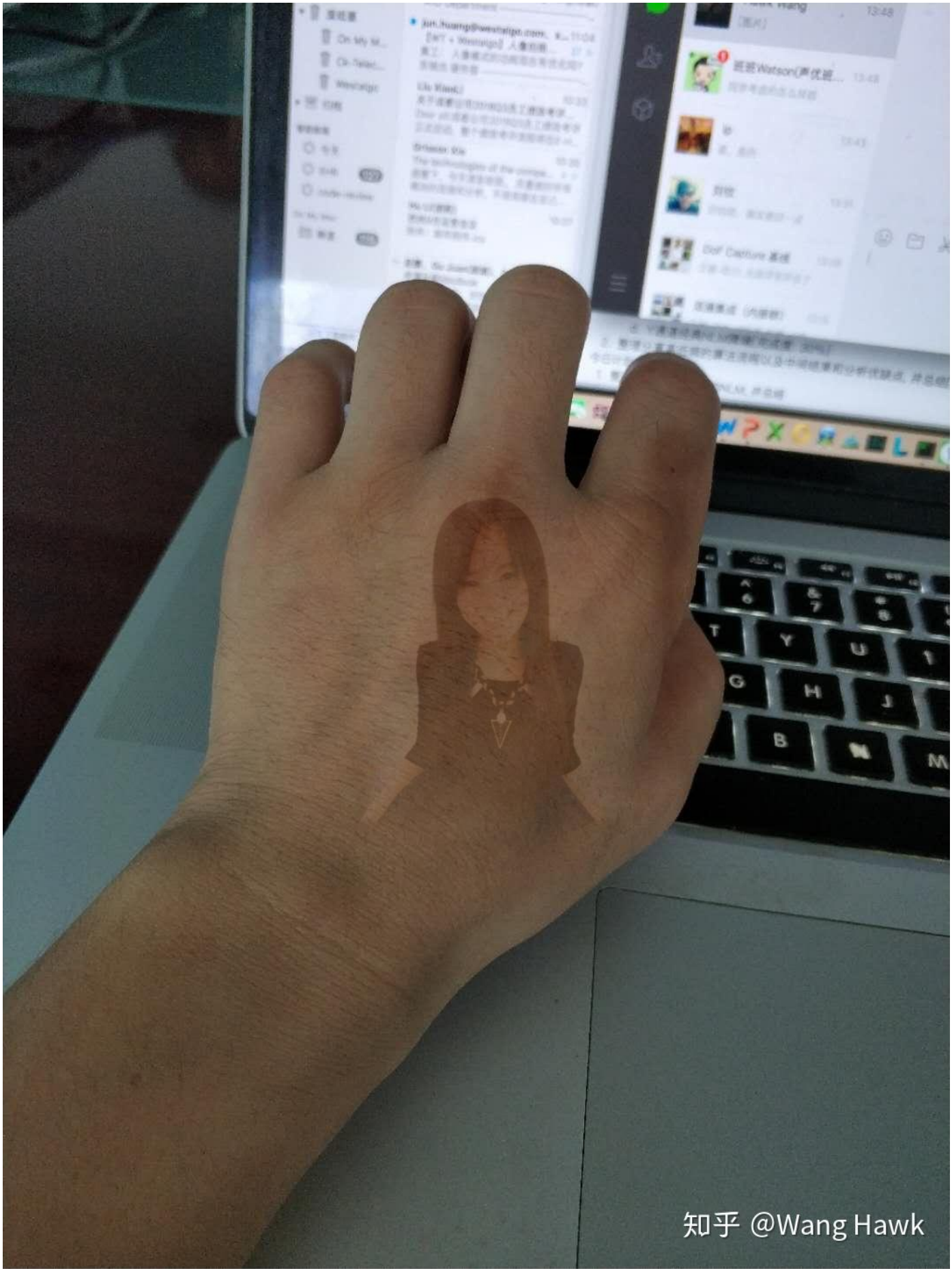
如果用cut-and-paste或者alpha融合，很明显我们需要非常准确的从左图中抠出苍老师的人像。如果抠的不够准确，或者干脆偷懒直接是矩形的图像区域输入，就会是这样的效果. (这里我把苍老师的权重设置为0.2，鄙人的手的权重设为0.8)


知乎 @Wang Hawk

那么用多频带融合呢？得到的结果也无法达到要求，甚至更是难看：


知乎 @Wang Hawk

而我们今天所讲的泊松融合，或者更广义来讲是泊松图像编辑，则可以很轻松的得到如下的结果，而且完全不需要去做非常准确的抠图：

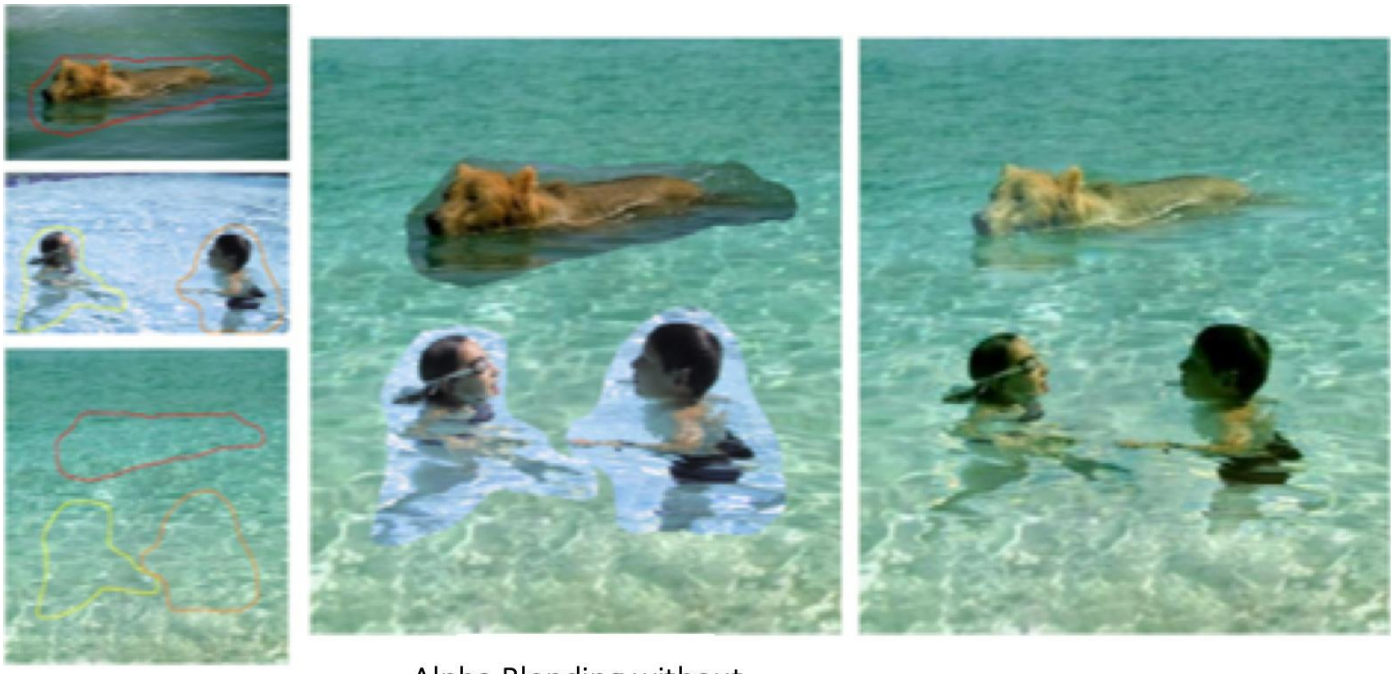

知乎 @Wang Hawk

我再给大家看更多泊松融合相比传统融合方式的优势，以及更多泊松融合的应用。

图像合成应用

对于cut-and-paste，我们已经看到它很容易在融合时出现明显的截断感。而我们今天所讲的泊松融合，则可以得到非常自然的融合结果，例如：

而我所讲的Alpha融合则很明显需要非常准确的用户操作或者准确的抠图，把想要融合的目标区域标示出来。如若不然就会出现很奇怪的效果。泊松融合则根本不需要精准的抠图，就可以得到很自然的结果：



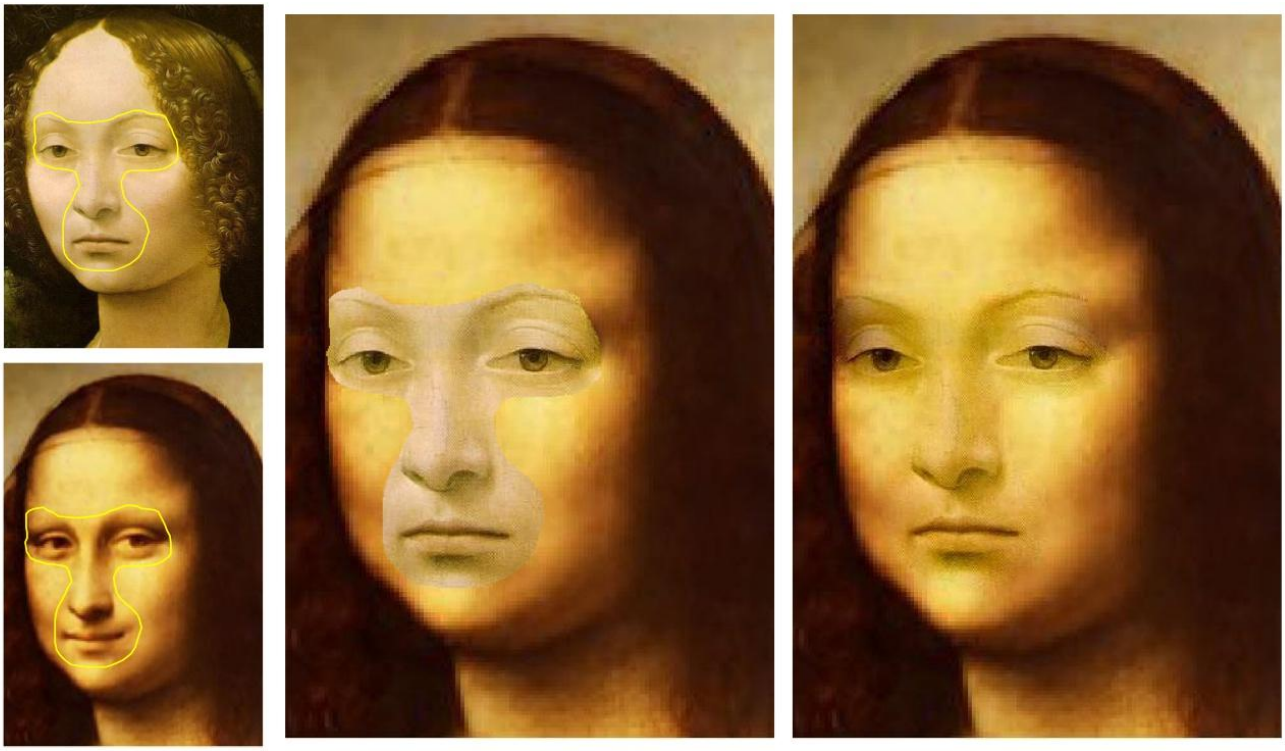
至于多频带融合，尽管它有时可以得到比较好的自然的融合结果，但各位也已经通过我开篇所展示的例子看到了它的缺陷性。不仅如此，它还有更多不足：

第一：每一层融合时的权重是由这一层的缩放尺度所决定的，这不见得是最优化的策略。

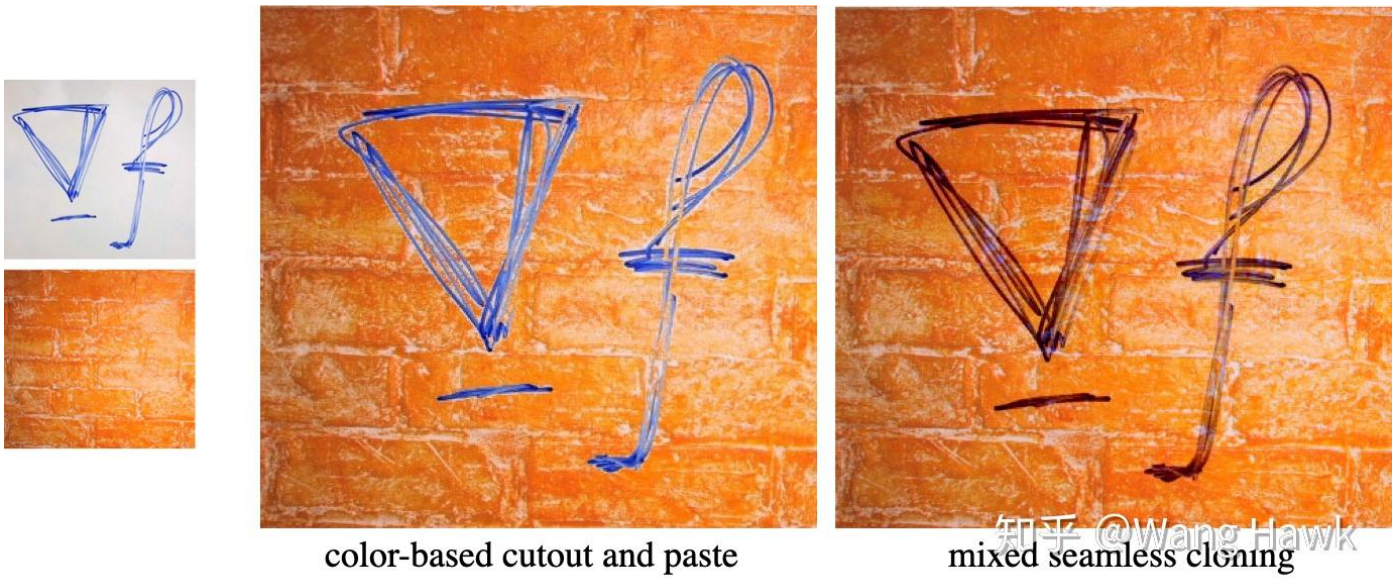
第二：在金字塔的上层，两个图像的融合会使得在原图像中相隔较远的像素的信息混叠到一起，这有时候会是不正确的行为。

第三：同时多频带融合仅仅提供图像融合的功能，而今天将要介绍的泊松图像编辑则不仅提供了融合图像的功能，还能提供更多我将要介绍的功能。

泊松融合在用户所指定的区域比较随意和粗糙时尤其有用：

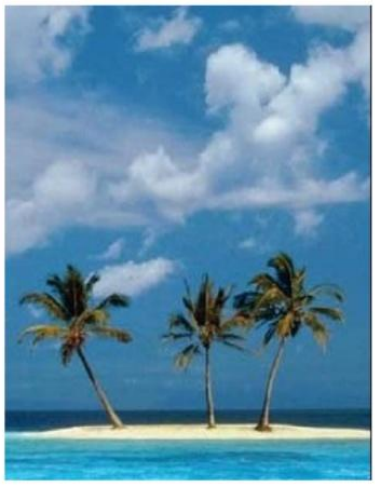


如果源图像内容包含空洞区域，如下图的字符，传统方法需要非常精确的抠取有效内容（下图中）。而泊松融合却只需要框取源对象所在的区域即可（下图右）。



如果源对象是部分透明的也没关系，泊松融合很自然的就可以将源目标的显著内容拷贝到目标图像。

source

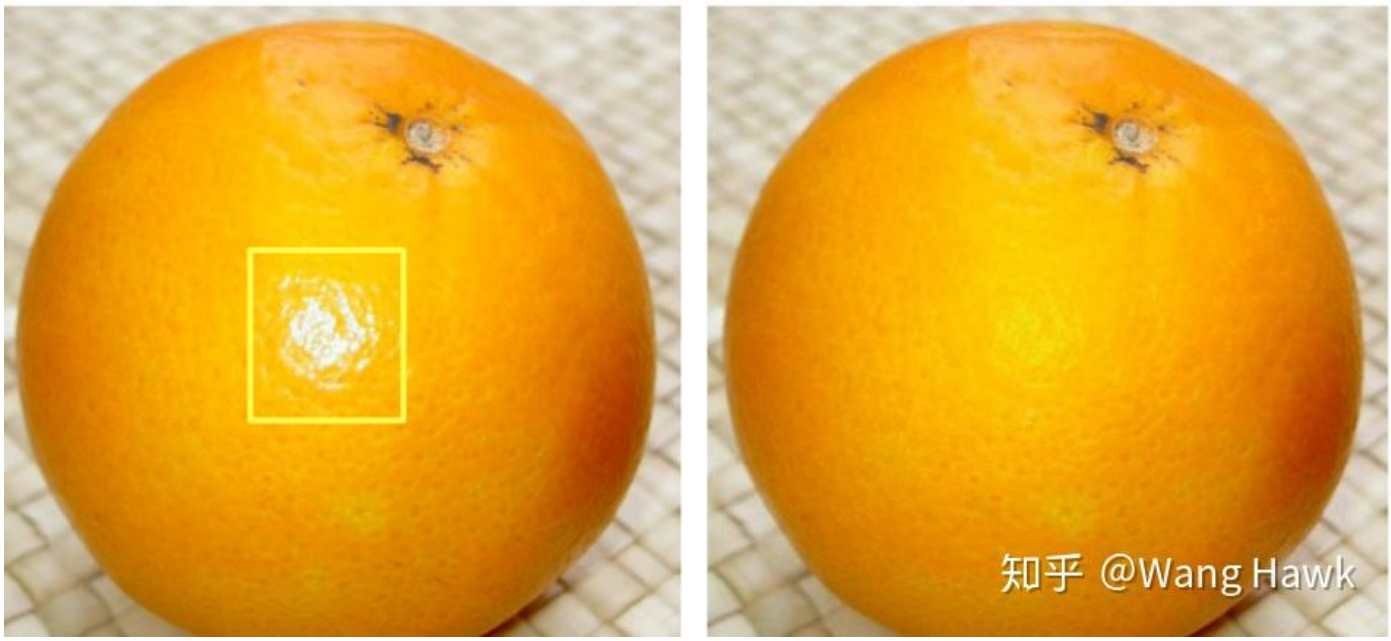


destination

图像编辑处理应用

泊松融合仅仅是泊松图像编辑中的一个功能，它还可以很容易的处理目标图像的纹理，例如下图，可以看到右图中脸部变得平坦了。这个过程中用户并不需要指明一个非常准确的人脸掩膜，只需要给出大概范围即可。

下图中水果表明上的高光区域被无缝的处理掉了，看起来毫无处理的痕迹。



下面这个场景中，只要大概的圈定一个范围就可以对图像的主题和图像内容，就可以选择性的对背景和前景进行处理，看起来毫无破绽。

二. 泊松融合的原理

讲了这么多泊松图像编辑的好处，那么到底它的原理是怎样的呢？为什么叫做泊松图像编辑呢？我们先来看看最基本的泊松融合。

2.1 基本的泊松融合

我们先对这个问题做一些基本符号定义。下图描述了在图像合成过程中的一些关键量的定义：

Definitions and notation

Notation

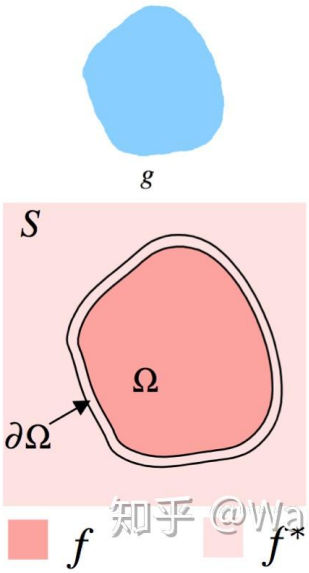
g: source function

S: destination

Ω: destination domain

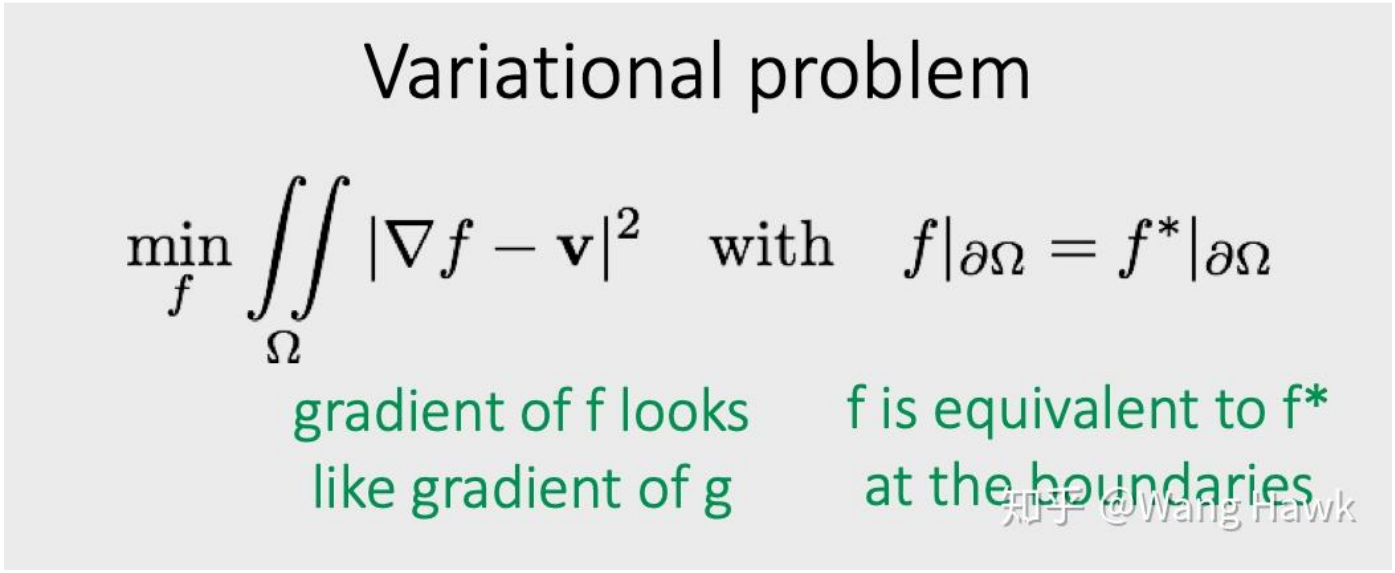
f: interpolant function

f*: destination function



图像合成是要把源图像g插入到目标图像S的区域Ω中，这个区域具有边界∂Ω 。目标图像的像素值和像素位置的关系可以用一个函数f*来描述。而合成后的图像在区域里的像素值和像素位置的关系则用函数f来描述。因此这里的g，S，Ω，∂Ω，f*等都是已知量，唯有f是待求解的函数（即图像）。我们希望f会使得图像合成后的边界过渡非常自然，而不会出现如我一开篇所举的传统融合方式结果那么突兀。

泊松融合的论文作者指出，要做到这一点，其实是需要求解一个变分问题：



这里：∇*f *指的是图像函数f的梯度

$$\nabla . = [\frac{\partial .}{\partial x}, \frac{\partial .}{\partial y}]$$

v在原论文中是指一个**引导向量场（guidance field）**，当用于图像合成时，它指的就是源图像的梯度。这意味着上面的变分方程是指在Ω 的区域内，f的梯度和源图像的梯度一致，而在Ω 的边缘处f的值则和源图像f*的值一致。这个变分方程的解是如下**泊松方程** 在Dirichlet边界条件时的解，这也是为什么我们的融合方式叫做**泊松融合** 。

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

这里进一步解释一下数学知识，这个方程中的几个关键和符号说明如下图：

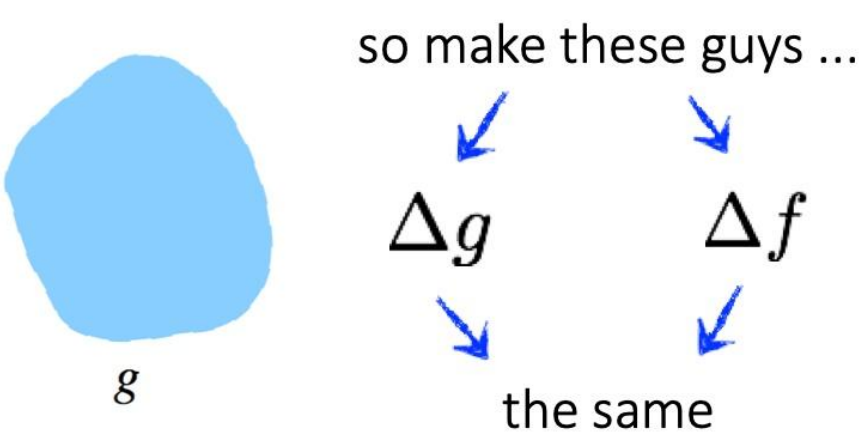
Gradient $\mathbf{v} = (u, v) = \nabla g$

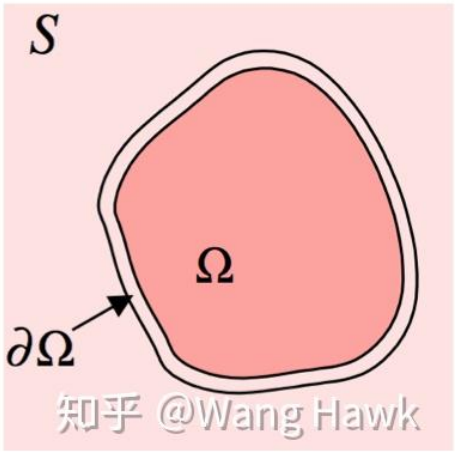
Laplacian $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Divergence $\operatorname{div} \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$

$$\begin{aligned}\operatorname{div} \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &= \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \\ &= \Delta g\end{aligned}$$

因此用人类语言描述上面的泊松方程，即是 f的拉普拉斯滤波结果和g的拉普拉斯滤波结果一致，且f的边界值和f*的边界值一致。





源论文中的关键公式为：

$$\Delta f = \Delta g \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}. \tag{10}$$

这意味着对于f和g的每个像素的拉普拉斯算子作用结果都一致：

$$\Delta f_p = \Delta g_p$$

由于拉普拉斯算子是这样的：

Laplace filter

0	1	0
1	-4	1
0	1	0

所以就是：

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

注意这里g是已知量，f是需要求取的函数，其中仅有边界值是已知的。

求解泊松方程是一件非常有技巧的事情，很多计算机视觉/图像处理方面的初学者都会卡在这一环上，这导致即便前面的原理都看懂了，也无法真正实现函数。这里我讲讲CMU课程中提到的一种方法，它的关键是将上述泊松方程表示为线性向量形式 ***Af=b***

linear equation of N variables

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

one for each pixel in destination

In vector form:

$$[0 \cdots 1 \cdots 1 \; 4 \; 1 \cdots 1 \cdots 0] \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} \Delta g_1 \\ \vdots \\ \Delta g_{q_1} \\ \vdots \\ \Delta g_{q_2} \\ \Delta g_p \\ \Delta g_{q_3} \\ \vdots \\ \Delta g_{q_4} \\ \vdots \\ \Delta g_N \end{bmatrix}$$

Linear system of equations

$$Af = b$$

等号的右边是图像g中每一个像素的拉普拉斯滤波结果Δgp，这很容易理解。未知函数f的每个元素构成了等号左边的列向量。而系数矩阵A则描述了最关键的f的拉普拉斯滤波算子。

我把A写成下面的形式你可能更容易理解（别忘了矩阵和向量的乘法的基本过程，可以稍稍复习下大一的基本知识）

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & -1 & \cdots & -1 & 4 & -1 & \cdots & -1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

对于每一个元素fp，它的拉普拉斯算子作用结果是4fp 减去p的所有4邻域像素值的和。因此当p不是边缘点时，这个矩阵的每一行有5个非零值。而当我们将f表示为列向量时，其邻域像素的位置是不连续的，因此上面的矩阵中每一行的-1所在的位置也是不连续的。当然，如果是边缘点，则还需要更加仔细的处理边缘，此时其对应的系数行中非零值的个数可能小于5。

如果你能够成功的将系数矩阵A表示出来，并将源图像g的拉普拉斯滤波结果算出，那么求解上述问题就变成了求解线性最小二乘问题了

如果你使用的是Matlab，一般来说只需要用：

```
f = A \ b
```

即可得到结果。

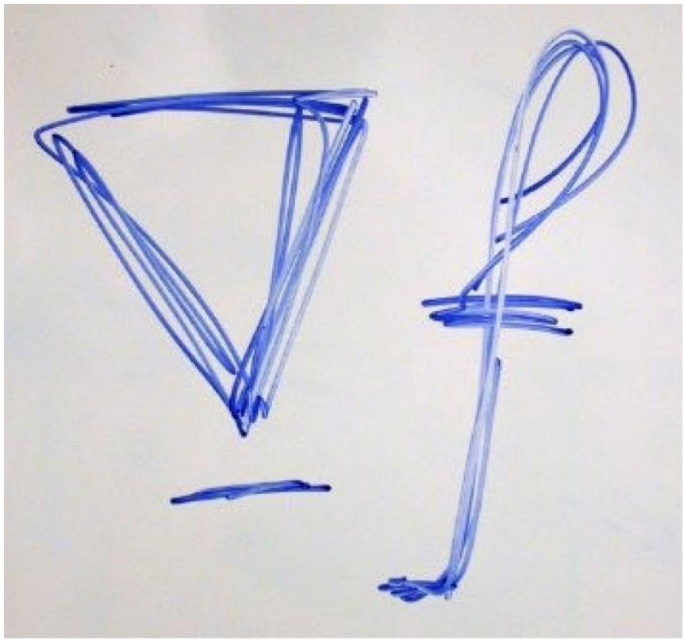
而如果你用的是Python，在scipy中也提供了很多线性方程求解函数。特别要注意的是，你可以看到上面的A矩阵里面有很大部分都是零值，因此当图像较大时一定要用稀疏矩阵来存储之，因此不管用任何语言来实现都需要采用对应的系数矩阵和函数库。

2.2 混合梯度的泊松融合

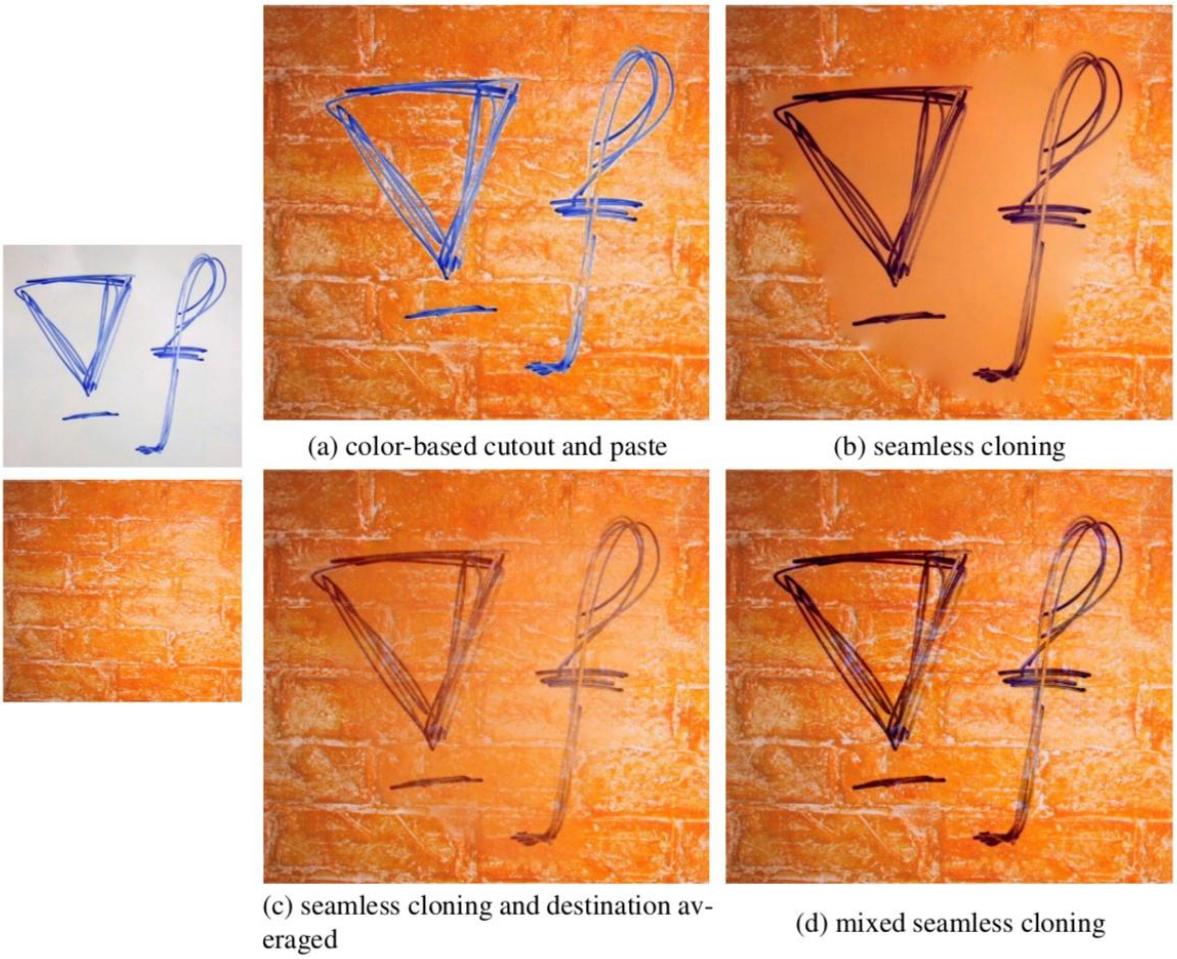
在基本的泊松融合中，我们的前提是：

$$\mathbf{v} = \nabla g, \tag{9}$$

即我们完全采纳源图像的梯度作为引导场。但有的时候，这样并不能得到很好的结果。有时候我们要同时兼顾源图像的梯度场和目标图像的梯度场。例如原论文中提到了一个例子是将下面两张图像融合到一起：



由于源图像中包含空洞，因此完全信任源图像的结果将不会很自然，如果你理解了上述的原理你会猜测到生成的图像在文字周围也会是梯度为0的一片模糊状。当然你可以将这个结果和目标图像进行加权融合，但也不会得到很漂亮的图像，还增加了计算量。而混合梯度的泊松融合则是为了解决此问题的。大家可以看看比较结果（引用自原论文）



(a) color-based cutout and paste (b) seamless cloning (c) seamless cloning and destination averaged (d) mixed seamless cloning

Figure 6: **Inserting objects with holes**. (a) The classic method, color-based selection and alpha masking might be time consuming and often leaves an undesirable halo; (b-c) seamless cloning, even averaged with the original image, is not effective; (d) mixed seamless cloning based on a loose selection proves effective.

这里的关键是这样设置v：

$$\text{for all } \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})|, \\ \nabla g(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{12}$$

即对于某个需要融合的像素，如果源图像的梯度幅值相比目标图像的梯度幅值更强，则设置v为源梯度，否则设置其为目标图像的梯度。

这一招尤其是在需要将源图像中的对象非常近的合成到目标图像中另外一个对象时有用：

Figure 8: **Inserting one object close to another**. With seamless cloning, an object in the destination image touching the selected region Ω bleeds into it. Bleeding is inhibited by using mixed gradients as the guidance field.

2.3 泊松图像编辑

在图像编辑中，源图像也是目标图像，而上面提到的泊松方程是一个通用的表达式：

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

正如原论文所说，其本质是在做Guided Interpolation：

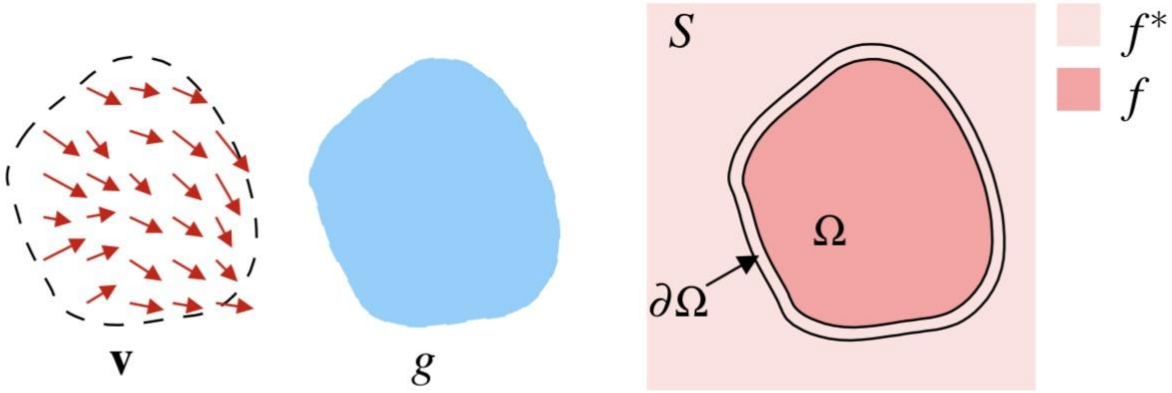


Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

当需要进行特别的图像编辑时，我们只需要设计特别的v即可，我们来看几个例子：

2.3.1 图像纹理平滑

这里我们希望得到如下结果

Figure 9: **Texture flattening**. By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect.

这里就只需要设置v为：

$$\text{for all } \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = M(\mathbf{x}) \nabla f^*(\mathbf{x}), \tag{14}$$

其中M是一个二值掩膜，它使得v中只保留了目标图像中非常显著的梯度。比较好的选择是只保留图像的边缘处的梯度。保留的梯度越少，最终的图像越平滑。

2.3.2 局部亮度改变

这里的效果如下图所示：

Figure 10: **Local illumination changes**. Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.

此时的v就更加复杂了：

$$\mathbf{v} = \alpha^{\beta} |\nabla f^*|^{-\beta} \nabla f^*, \tag{16}$$

总之，设计不同的引导场v可以得到不同的结果。

三. 总结

今天给大家介绍了一种惊人的图像融合技术：泊松融合，以及其衍生的泊松图像编辑。它能得到非常好的融合效果，也能够对图像进行很自然的编辑，而且不像传统方法那么费事。大家可以开动脑洞，自己想想有没有什么更好的用处。我帮直男想的招就是可以试试将自己的女神纹（Blend）到手臂上，身上，体恤上什么的，用于表白

正如以前一样，我也会随后放上跟这篇文章相关的python notebook，展示其中的一些基本思想，请大家等待这篇文章的更新信息。

跟这一系列专题文章相关的Notebook可以从 https://github.com/yourwanghao/CMUComputationalPhotography.git获取

有进一步问题可以点下面链接向我咨询

Wang Hawk15 次咨询5.015597 次赞同去咨询

参考资料：

这一篇文章的绝大部分素材来自于

[1] CMU 2017 Fall Computational Photography Course 15-463, Lecture 7

[2] P. P´erez, M. Gangnet, and A. Blake. Poisson image editing. In ACM Transactions on graphics (TOG), volume 22, pages 313–318. ACM, 2003.

[3] Richard Szeliski, Computer Vision : Algorithms and Applications

文章中，苍老师的图像来源于网络，如有侵权请及时提醒我更换。 图像原始地址为：

http://image.hnol.net/c/2013-03/13/09/4477cd4e580a21e35f7e1a9cd20b77ca-2995785.jpg