A Closed Form Solution to Natural Image Matting

Anat Levin Dani Lischinski Yair Weiss

School of Computer Science and Engineering

The Hebrew University of Jerusalem

{alevin,danix,yweiss}@cs.huji.ac.il

自然背景图像抠图的一个闭环解决方案

原文链接：

http://webee.technion.ac.il/people/anat.levin/papers/Matting-Levin-Lischinski-Weiss-CVPR06.pdf

译文出处：https://github.com/hyifan/TranslatePapers

摘要

交互式数字抠图（interactive digital matting），是基于有限的用户输入从图像中提取前景对象的过程，是图像和视频编辑中的一项重要任务。从计算机视觉的角度来看，这项任务是非常具有挑战性的，因为它是严重不适定的（massively ill-posed）-在每个像素，我们必须估计前景和背景颜色，以及从单一颜色测量得到的前景不透明度（“alpha 遮罩”）。当前的方法要么将估计限制在图像的一小部分，要么基于已知像素附近的像素估计前景和背景颜色，要么用 alpha 估计通过交替的前景和背景颜色估计迭代进行非线性估计。

本文提出了一种自然图像抠图(natural image matting)的闭环解决方案(a closed form solution)。我们从对前景和背景色的局部平滑（local smoothness）假设中推导出一个代价函数，并证明在所得到的表达式中，可以通过分析消除前景和背景色，得到 α 的二次代价函数。这使得我们可以通过解一个稀疏的线性方程组来找到全局最优的 alpha 遮罩。此外，闭环公式允许我们通过分析稀疏矩阵的特征向量来预测解的性质，稀疏矩阵与光谱图像分割算法中使用的矩阵密切相关。我们将证明，可以使用令人惊讶的少量用户输入从自然图像中获得高质量的遮罩（matte）。

1.介绍

自然图像抠图和合成在图片和视频编辑中具有重要意义。形式上，图像抠图的方法是以图像 I 作为输入，假定图像 I 是前景图像 F 和背景图像 B 的组合。第 i 个元素的颜色假定为相应的前景颜色和背景颜色的线性组合，

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \qquad (1)$$

其中 α_i 是第 i 个元素的前景透明度。在自然图像抠图中，合成公式(1)右边的所有变量都是未知的。因此，在一个有三通道的颜色图像中，每个像素都有 3 个公式和 7 个未知数。

显然，这是一个欠约束的问题，需要用户的交互来提取一个好的遮罩。最新的方法要求用户提供一个三值图(trimap)作为起点，图 1(e)显示了一个示例。三值图是将图像分成 3 个区域的一个粗略(通常是手绘) 分割：前景（显示为白色）、背景（显示为黑色）、未知区域（显示为灰色）。给定一副三值图，这些方法通常同时求解 F、B 和 α。这通常是通过迭代非线性优化来完成的，即将 F 和 B 的估计与 α 的估计交替进行。事实上，这意味着想有好的效果，需要三值图绘出的未知区域尽可能小。因此，基于三值图的方法通常难以处理大部分混合的像素或前景对象有很多孔的情况。在这种具有挑战性的情况下，不可避免地需要大量的经验和用户交互来构造一个三值图使之产生一个好的遮罩。三值图交互的另一个问题是，用户不能直接影响图像最重要部分的遮罩：混合像素。

在这篇论文中，我们提出了一种新的闭环解决方案用于从自然图像中提取 alpha 遮罩。我们从前景颜色 F 和背景颜色 B 的局部平滑假设中推导出一个代价函数，并证明在所得的表达式中它可以通过分析消除 F 和 B，从而得到 α 的二次代价函数。本文产生的 alpha 遮罩是该代价函数的全局最优解，可以通过解一个稀疏线性系统得到。由于我们的方法可以直接计算 α 而不需要对 F 和 B 进行可靠的估计，因此很少的用户输入（例如一组稀疏的涂鸦）通常足以提取高质量的遮罩。此外，我们的闭环公式可以通过检查稀疏矩阵的特征向量来了解和预测解的特性，其中稀疏矩阵与光谱图片分割算法中使用的矩阵密切相关。除了为我们的方法提供坚实的理论基础外，这种分析还可以为用户提供关于图像涂鸦应放置在何处的有用提示。

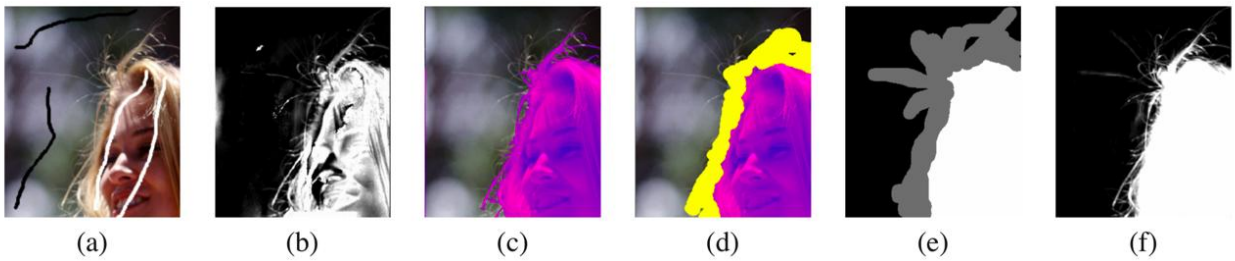


图 1

(a)具有稀疏约束的图像：白色涂鸦表示前景，黑色涂鸦表示背景。将这种稀疏输入应用于贝叶斯抠图(Bayesian matting) 会产生一个完全错误的遮罩(b)。前景提取算法，如文献[9,11]，产生一个硬分割（hard segmentation）(c)。从硬分割自动生成的三值图可能会丢失精细特征(d)。在这种情况下，需要精细的手绘三值图(e)，以产生合理的遮罩(f)。(图片来自文献[15])

1.1 前期工作

大多数现有的自然图像抠图的方法要求输入图像带有三值图，将每个像素标记为前景、背景或未知。这些方法的目的是求解合成方程（1）中的未知像素。这通常是通过利用 F 和 B 上的一些局部规律性假设来预测未知区域中每个像素的值来实现的。在 Corel KnockOut 算法中，假设 F 和 B 是平滑的，并通过基于已知前景和背景像素的加权平均值（越近的像素接收到更高的权重）来预测。一些算法假设局部前景和背景来自相对简单的颜色分布。也许这些方法中最成功的是贝叶斯抠图（Bayesian matting）算法，该算

法混合了定向高斯用于学习局部分布，α、F 和 B 被估计为给定分布中最可能的一个。当前景和背景的颜色分布不重叠，且三值图中未知区域较小时，这种方法可以很好地工作。如图 1(b)所示，一组稀疏的约束可能导致完全错误的遮罩。相比之下，虽然我们的方法对于 F 和 B 进行了确定的平滑的假设，但在提取遮罩之前，它不涉及估计这些函数的值。

泊松抠图(Poisson matting)方法也期望三值图作为其输入的一部分 ,并通过用遮罩梯度场(gradient field) 和 Dirichlet 边界条件求解泊松等式来计算混合区域中的 alpha 遮罩。在全局泊松抠图方法中，可以通过采用合成方程的梯度和忽略 F、B 的梯度来将遮罩梯度场近似为$\nabla I/(F - B)$ 。通过求解某些函数，这些函数的导数与近似的遮罩梯度场接近，来找到遮罩。当 F 和 B 在未知区域内不够平滑时，得到的遮罩可能不正确，可能需要对遮罩梯度场进行附加的局部操作，以获得满意的解。这些交互式优化过程称为局部泊松抠图（local Possion matting ）。正如我们将看到的，我们的方法对 F 和 B 的行为作出了较弱的假设，这通常会导致更精确的遮罩。

近年来 ,人们提出了几种从背景中提取前景对象的成功的方法。这些方法都将用户指定的简单约束(如涂鸦或边界矩阵）转换为最小切割问题。解决最小切割问题会导致一个硬的二进制分割，而不是一个分数 alpha 遮罩（图 1c ）。由于腐蚀（erosion），硬分割可以转化为三值图，但这仍然会遗留一些细微或模糊的特征(图 1d)。虽然 Rother 等人通过在硬边界周围的窄条带中 ,拟合参数 alphp 轮廓来执行边界抠图，但这更类似于羽化，而不是完全 alphp 抠图，因为不能以这种方式处理宽模糊区域。

我们的方法与 Levin 等人的着色方法和 Grady 等人的 random walk alpha 抠图方法密切相关。这两种方法都是通过最小化二次代价函数将涂鸦约束传播到整个图像。在这里，我们应用了类似的策略，但是我们修改了假设和代价函数，以便更好地适应抠图问题。

Wang 和 Cohen 最近提出了另一种基于涂鸦的用于交互抠图的界面。从表示少量背景和前景像素的一些涂鸦开始，他们使用置信传播（belief propagation）迭代估计图像中每个像素的未知数。虽然这些方法产生了一些令人印象深刻的结果，但它的缺点是采用了一个昂贵的迭代非线形优化过程，该过程可能收敛到不同的局部极小值。

2.推导

为了清晰的说明，我们首先推导了灰度图像的 alpha 抠图的闭环解决方案。该结果将扩展到第 2.1 节中的彩色图像。

如前所述，抠图问题是一个严重的欠约束问题。因此，需要对 F、B 和（或）α 的性质进行一些假设。为了推导灰度情况的解决方案，我们假设 F 和 B 在每个像素周围的一个小窗口上都近似常量。注意，假设 F 和 B 是局部平滑的并不意味着输入图像 I 是局部平滑的，因为 α 的不连续可以导致 I 的不连续。这些假设在第 2.1 节会稍微放宽，允许我们重写（1）将 α 表示为图像 I 的线形函数：

$$\alpha_i \approx aI_i + b, \quad \forall i \in w \tag{2}$$

其中$a = \frac{1}{F-B}$ ，$b = -\frac{B}{F-B}$ ，w 是一个小图像窗口。这些线形操作与之前使用过的相似。本文的目标是寻找 α、a 和 b，最小化代价函数：

$$J(\alpha, a, b) = \sum_{j \in I} (\sum_{i \in w_j} (\alpha_i - a_j I_j - b_j)^2 + \varepsilon {a_j}^2) \quad (3)$$

其中w_j是像素 j 附近的小窗口。

上面的代价函数包括 a 上的正则化项。增加这个项的一个原因是数值稳定性。例如，如果图像在第 j 个窗口中是常量，则在没有先验的情况下无法唯一确定a_j和b_j。此外，最小化范数 a 会使得解偏向于更平滑的 alpha 遮罩（因为$a_j = 0$意味着 α 在第 j 个窗口上是常量）。

在我们的实现中，我们通常使用 3×3 像素的窗口。因为我们在每个像素周围放置一个窗口，所以（3）中的窗口w_j重叠。正是这种特性使得信息能够在相邻像素之间传播。代价函数是 α、a 和 b 的二次函数，对于一个有 N 个像素的图像有 3N 个未知数。幸运的是，正如我们下面所展示的，a 和 b 能从（3）中消除，只留下含 N 个未知数的二次代价函数：像素的 alpha 值。

定理 1：

定义J(α)为

$$J(\alpha) = \min_{a,b} J(\alpha, a, b)$$

那么

$$J(\alpha) = \alpha^T L \alpha \quad (4)$$

其中 L 是一个 N×N 矩阵，它的第(i,j)元素是

$$\sum_{k|(i,j) \in w_k} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + \frac{1}{\frac{\varepsilon}{|w_k|} + {\sigma_k}^2} (I_i - \mu_k)(I_j - \mu_k) \right) \right) \quad (5)$$

其中 δ_{ij} 是克罗内克函数（Kronecker delta），μ_k 和 ${\sigma_k}^2$ 是 k 周围的窗口 w_k 中各像素颜色的平均值和方差，$|w_k|$ 是该窗口中的像素个数。

证明：

用矩阵表示法重写式（3）可得

$$J(\alpha, a, b) = \sum_k \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \overline{\alpha}_k \right\|^2 \quad (6)$$

对于每个窗口 w_k，G_k 定义为 $(|w_k| + 1) \times 2$ 的矩阵。对于每个 $i \in w_k$，G_k 包含了一行 $[I_i, 1]$，且 G_k 的最后一行是 $[\sqrt{\varepsilon}, 0]$。对于给定的遮罩 α，我们定义 $\overline{\alpha}_k$ 是 $(|w_k| + 1) \times 1$ 的向量，对于每个 $i \in w_k$，定义为 α_i，它的最后一个元素是 0。$\overline{\alpha}_k$ 和 G_k 的元素都是按顺序对应的。

对于给定的遮罩 α，每个窗口 w_k 中的最优对 a_k^*，b_k^* 是最小二乘问题的最优解：

$$(a_k^*, b_k^*) = \operatorname{argmin} \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \overline{\alpha}_k \right\| \quad (7)$$

$$= ({G_k}^T G_k)^{-1} {G_k}^T \overline{\alpha}_k \quad (8)$$

将这个解代入等式（6），引入 $\bar{G}_k = I - G_k(G_k{}^T G_k)^{-1} G_k{}^T$，我们得到

$$J(\alpha) = \sum_k \bar{\alpha}_k{}^T \bar{G}_k{}^T \bar{G}_k \bar{\alpha}_k$$

进一步的代数运算表明 $\bar{G}_k{}^T \bar{G}_k$ 的第(i,j)个元素可以表示为

$$\delta_{ij} - \frac{1}{|w_k|}\left(1 + \frac{1}{\frac{\varepsilon}{|w_k|} + \sigma_k{}^2}(I_i - \mu_k)(I_j - \mu_k)\right)$$

对 k 求和可以得到式（5）

2.1 彩色图像

将代价函数应用于彩色图像的一个简单方法是将灰度代价函数分别应用于每个通道。或者，我们可以将线形模型（2）替换为 4D 线形模型：

$$\alpha_i \approx \sum_c a^c I_i^c + b, \quad \forall i \in w \tag{9}$$

这种组合线形模型的优点在于，它放宽了我们先前的假设，即 F 和 B 在每个窗口上都是常数。相反，如我们下面所示，假设在一个小窗口中，F 和 B 中的每一个都是两种颜色的线性混合物就足够了；换句话说，小窗口的值 F_i 位于 RGB 颜色空间单独的一行：$F_i = \beta_i F_1 + (1 - \beta_i)F_2$，背景值 B_i 也是如此。下面我们将这个假设称为色彩线模型（color line model）。

这样的模型是有用的，因为它捕捉的是，例如，在一个具有常量反照率（albedo）的表面上的阴影变化。另一个例子是，窗口包含同属于背景或前景的两个颜色一致的区域之间的边缘。此外，Omer 和 Werman 证明，在许多自然图像中，RGB 空间中的像素颜色倾向于形成相对少量的细长簇。虽然这些簇不是直线，但它们的轮廓基本上是局部线形的。

> 译者注：
>
> Richard Szeliski《计算机视觉—算法与应用》一书对色彩线模型有一个很通俗的解释：
>
> Levin, Lischinskiand Weiss(2008)没有去算出每个像素的前景色和背景色，而只是假设前景色和背景色在局部上分别可以近似是由两种颜色混合而成，这被称为色彩线模型。。

定理 2:

如果窗口中的前景和背景颜色满足色彩线模型，有表达式：

$$\alpha_i = \sum_c a^c I_i^c + b, \quad \forall i \in w$$

证明：

用线性组合

$$F_i = \beta_i{}^F F_1 + (1 - \beta_i{}^F)F_2$$

和

$$B_i = \beta_i^B B_1 + (1-\beta_i^B)B_2$$

替换等式（1），其中 F_1、F_2、B_1、B_2 是小窗口上的常量，可以得到：

$$I_i^c = \alpha_i\left(\beta_i^F F_1^c + \left(1-\beta_i^F\right)F_2^c\right) + (1-\alpha_i)\left(\beta_i^B B_1^c + \left(1-\beta_i^B\right)B_2^c\right)$$

设 H 是一个 3×3 矩阵，它的第 c 行是$[F_2^c + B_2^c, F_1^c - F_2^c, B_1^c - B_2^c]$。那么上式可以写成：

$$H\begin{bmatrix}\alpha_i \\ \alpha_i\beta_i^F \\ (1-\alpha_i)\beta_i^B\end{bmatrix} = I_i - B_2$$

其中，I_i 和 B_2 是 3×1 向量，表示 3 个颜色通道。我们将 H^{-1} 的第一行的元素记为 a^1, a^2, a^3，将 H^{-1} 的第一行的向量与 B_2 作标量乘法得到 b。最终得到 $\alpha_i = \sum_c a^c I_i + b$ 。

> 译者注：
>
> 我计算的 H 的第 c 行是$[F_2^c - B_2^c, F_1^c - F_2^c, B_1^c - B_2^c]$，与论文不同，计算结果与论文相同的小伙伴请赐教。

使用 4D 线性模型（9），我们定义了如下用于 RGB 图像抠图的代价函数：

$$J(\alpha, a, b) = \sum_{j\in I}\left(\sum_{i\in w_j}\left(\alpha_i - \sum_c a_j^c I_i^c - b_j\right)^2 + \varepsilon\sum_c a_j^{c\,2}\right) \qquad (10)$$

与灰度情况类似，a^c 和 b 可以从代价函数中消除，从而生成关于未知数 α 的二次代价函数：

$$J(\alpha) = \alpha^T L\alpha \qquad (11)$$

这里 L 是一个 N×N 矩阵，它的第(i,j)元素值为：

$$\sum_{k|(i,j)\in w_k}\left(\delta_{ij} - \frac{1}{|w_k|}\left(1 + (I_i-\mu_k)\left(\sum_k + \frac{\varepsilon}{|w_k|}I_3\right)^{-1}(I_j-\mu_k)\right)\right) \qquad (12)$$

其中 $\sum_k$ 是一个 3×3 的像素颜色的协方差矩阵，μ_k 是窗口 w_k 中各像素颜色的 3×1 平均值向量，I_3 是 3×3 单位矩阵。

我们将方程（5）和（12）中的矩阵 L 称为抠图拉普拉斯（matting Laplacian）。注意，L 的每一行中的元素总和为 0，因此 L 的零空间（nullspace）包括常量向量。如果使用 ε=0，L 的零空间也包括 I 的每个颜色通道。

3.约束和用户界面

在我们的系统中，用于提供的对抠图的约束是通过基于涂鸦的 GUI 提供的。用户使用背景画笔（在我们的示例中为黑色涂鸦）表示背景像素（α=0），使用前景画笔（白色涂鸦）表示前景像素（α=1）。

为了提取与用户涂鸦匹配的 alpha 遮罩，我们需解出：

$$\alpha = \operatorname{argmin}\alpha^T L\alpha, \qquad \text{s.t. } \alpha_i = s_i, \forall i \in S \qquad (13)$$

其中 S 是一组涂鸦像素，s_i 是涂鸦指示的值。

定理 3:

设I为根据（1）由F和B生成的图像，α*表示真正的alphp遮罩。如果F和B满足每个局部窗口 w_k 中的色彩线模型，并且用户指定的约束 S 与 α*一致，则 α*是系统（13）的最优解，其中 L 用 ε=0 构造。

证明：

由于 ε=0，如果在每个窗口 w_k 中都满足色彩线模型，则根据定义（10），有 J(α*,a,b)=0，因此有 $J(\alpha^*) = \alpha^{*T} L \alpha^* = 0$。

我们在图 2 中演示了这一点。第一幅图（图 2(a)）是一个人造的例子，它是用计算机模拟（单色）烟雾与包含几个色带的简单背景合成的图片，满足色彩线模型。黑白涂鸦显示了输入约束。我们的方法提取的遮罩（图 2(b)）与真值（ground truth）遮罩完全一致。第二个例子（图 2(c)）是一张真实的图像，具有相当均匀的前景和背景颜色。通过只涂鸦出两个黑白点，可以提取出高质量的遮罩（图 2(d)）。

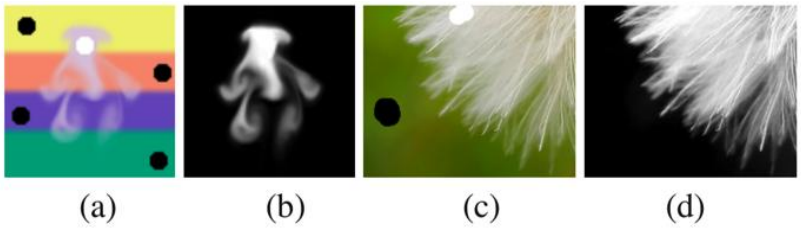


图 2

抠图例子。(a,c)经过涂鸦的黑色图像。(b,d)提取的遮罩。

4.光谱分析

抠图拉普拉斯矩阵 L 是一个对称正定矩阵，从定理 1 及其证明中可以看出。该矩阵也可以写成 L=D-W，其中 D 是对角矩阵 $D(i,i) = \sum_j W(i,j)$，W 是对称矩阵，其非对角项由（12）定义。因此，矩阵 L 是用光谱方法分割的拉普拉斯图，但具有（12）给出的新型亲和函数。为了方便对比，定义亲和函数的典型方法（即在标准化切割图像分割算法中[13]）是设置

$$W_G(i,j) = e^{-\|I_i - I_j\|^2/\sigma^2} \qquad (14)$$

其中σ是一个全局常量（通常手动选择）。对于邻近像素当它们具有相似的颜色时亲和度很强，当它们的色差远大于σ时亲和度接近于零。random walk 抠图算法对抠图问题使用了相似的亲和函数，但在对两个像素的颜色进行线形变换后，取的是两个像素的颜色距离。该变换依赖于图像，并使用流形学习技术进行估计。

相反，通过将抠图拉普拉斯改写为 L=D-W，我们得到以下亲和函数，称之为“抠图亲和度”（matting affinity）：

$$W_M(i,j) = \sum_{k|(i,j)\in w_k} \frac{1}{|w_k|}\left(1 + (I_i - \mu_k)\left(\sum_k + \frac{\varepsilon}{|w_k|} I_3\right)^{-1} (I_j - \mu_k)\right) \quad (15)$$

为了获得抠图亲和度的直观感受，考虑一个刚好包含两种颜色（例如，理想边缘）的图像补丁。在这种情况下，可以证明，同一个颜色的两个像素之间的亲和度随距离而降低，而不同颜色的像素之间亲和度为 0。一般来说，我们得到一个与标准亲和函数相似的情况：颜色相似的邻近像素具有高亲和度，而颜色不同的邻近像素具有低亲和度。但是，请注意，抠图亲和度没有全局缩放参数σ，而是使用平均值和方差的局部估计。正如我们随后所展示的，这种自适应特性导致了性能的显著提高。在文献[16]中也进行了类似的观察，这表明局部调整缩放参数可以改善图像分割结果。

为了比较两个亲和函数，我们检查了相应拉普拉斯的最小特征向量，因为这些特征向量被光谱分割算法用于图像分割。

图 3 展示了两个示例图像上的两个拉普拉斯矩阵的第二个最小特征向量（在两种情况下，第一个最小特征向量都是常量图像）。第一个例子是一个带有不同颜色同心圆的简单图像。在这种情况下，区域之间的边界非常简单，两个拉普拉斯矩阵都能正确地捕捉到转换。第二个例子是孔雀的形象。全局σ特征向量（由归一化切割算法使用）无法捕捉到孔雀尾羽和背景之间的复杂模糊边界。相比之下，抠图拉普拉斯的特性向量很好地将孔雀与背景区分开来。本例的抠图拉普拉斯用 $\varepsilon=0.0001$ 计算。

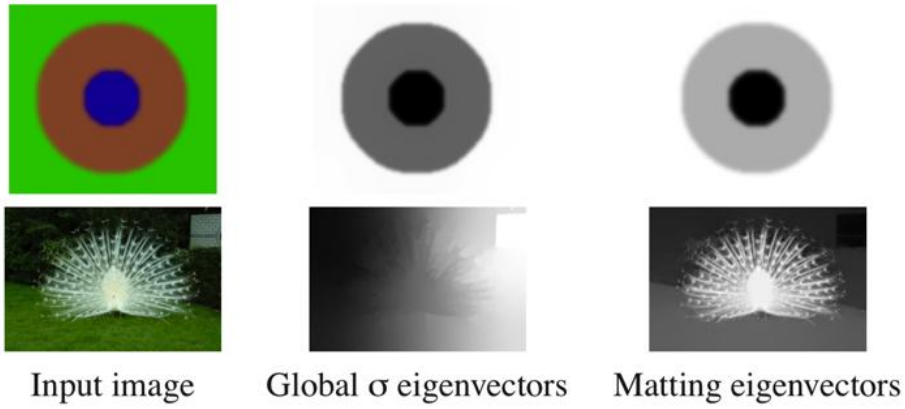


图 3

使用不同拉普拉斯的最小特征向量。

4.1 作为向导的特征向量

虽然在没有用户输入的情况下，抠图拉普拉斯矩阵是不适定的，但是抠图拉普拉斯矩阵包含了大量关于图像的信息，甚至在没有提供任何涂鸦的情况下，如前一节所示。

这表明。查看抠图拉普拉斯的最小特征向量可以指导用户在哪里放置涂鸦。例如，在最小特性向量为分段常数（piecewise constant）的相同区域中，提取的遮罩往往是分段常数。如果特征向量图像中某个段内的值是一致的，那么该段内的单个涂鸦就足以将所有值传播到整个段。另一方面，特性向量值不太一致的区域对应于图像中更“困难”的区域，这表明可能需要在那个区域进行更多的涂鸦。

更准确地说，可以通过检查抠图拉普拉斯的较小的特征向量来预测 alpha 遮罩，因为（13）的最优解将在很大程度上由较小特征向量掌握。事实上，可以约束最优解的权重，将其分配给较大的特征向量，作为相应特征值比率的函数。

图 4 说明了特征向量如何引导一个涂鸦的过程。通过检查两个最小特性向量（图 4(a-b)），我们在每个显示出一致的特征向量值的区域内放置一个涂鸦（图 4(c)）。得到如图 4(d)所示的遮罩。请注意，图 4(c)的涂鸦是我们第一次尝试在此图像上放置涂鸦。

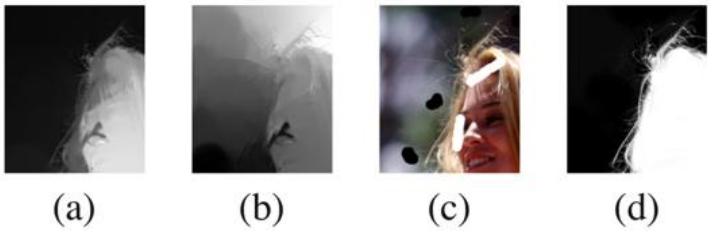


图 4

用于引导用户涂鸦位置(c)的最小特征向量(a-b)。提取的遮罩(d)。

5.结果

我们在这里展示了通过提供的涂鸦约束来最小化抠图拉普拉斯函数，以提取 alpha 遮罩的闭环解决方案。由于抠图拉普拉斯中的 alpha 是二次的，可以通过解一组稀疏的线形方程精确地求出最小值。我们通常使用 3×3 的窗口定义抠图拉普拉斯。当前景和背景颜色分布不太复杂时，使用更宽的窗口是有帮助的。但是，使用更宽的窗口会增加计算时间，因为生成的结果系统不那么稀疏。为了克服这个问题，我们考虑了线形系数（9），它与图像的 alpha 遮罩相关。在高分辨率图像上使用宽窗口获得的系数，与在较低分辨率图像上使用较小窗口获得的系数相似。因此，我们可以在低分辨率图像上使用 3×3 窗口来求解 alpha 遮罩，并计算与低分辨率图像相关的线形系数。之后，我们对线性系数进行插值，并将其应用到更高分辨率的图像上。使用这种方法获得的 alpha 遮罩与直接在具有高分辨率图像上获得的遮罩结果相似。更多细节见文献[8]。

对于这里显示的结果，我们使用 matlab 的直接解算器（“反斜杠”操作符）求解抠图系统，在 2.8GHz CPU 上 200×300 图像需要 20 秒。由于内存的限制，无法使用 matlab 解算器处理大图像。为了克服这一点，我们采用了从“粗”到“细”的方案。我们对图像和涂鸦进行了缩小采样，并以较低的分辨率进行求解。然后将重建的 alpha 插入到更精细的分辨率中，并对 alpha 值进行阈值化，将 alphp 接近 0 或 1 的像素视为更精细的分辨率中的约束。约束像素可以从系统中消除，从而减小系统大小。我们还为遮罩提取实现了一个多网格解算器。即便是对于非常大的图像多网格解算器也能在几秒内运行，但遮罩质量略有下降。

我们在这里只显示提取的 alpha 遮罩。注意，未来在新背景上合成图像，我们还需要解出 F。当遮罩

被找到以后，可以直接从方程（10）中解出 a^c 和 b 系数，并从中提取前景和背景。然而，我们发现，为获得更好的前景和背景估计，可以通过求解 F 和 B 中的一组新的线形方程，或可以通过在方程（1）中引入 F 和 B 上的一些显式平滑先验推导出。关于前景重建以及一些合成结果的更多信息，可以在文献[8]中找到。

图 5 显示了使用我们的技术对文献[15]中具有挑战性的两个图像提取的遮罩，并将我们的结果与其他几种最新的算法进行了比较。可以看出，我们对这些示例的结果在质量上与文献[15]的结果相当，尽管我们使用的算法要简单得多。全局泊松抠图虽然在三值图上性能很好，但无法从稀疏的“涂鸦”中提取好的遮罩。random walk 抠图算法同样是用最小化拉普拉斯，但是使用具有全局缩放参数的亲和函数，因此对孔雀图像有特殊的困难。

为了获得这些算法的定量比较，我们用带有真值 alpha 遮罩的合成图像做了一个实验。我们从图 6(h)所示的图像中随机抽取 2000 个子图像。我们使用每个子图像作为背景，并在上面使用两个不同的 alpha 遮罩合成统一的前景图像：第一个遮罩是计算机模拟的烟雾，其中大部分是部分透明的；另一个遮罩是一个圆的一部分，大部分是不透明的，且带有羽状边界。遮罩如图 6(c)所示。因此，我们获得了 4000 张合成图像，其中两张如图 6(a)所示。在这组图像上，我们比较了四种抠图算法的性能：Wang 和 Cohen、全局泊松抠图、random walk 抠图和我们自己的算法（使用 3×3 无金字塔窗口）。所有的算法都提供三值图作为输入。图 6(a,d-g)显示了三值图的示例以及不同方法产生的结果。对于每一种算法，我们测量了提取的遮罩与真值之间的绝对误差之和。图 6(i,j)绘制了四种算法作为背景平滑度函数的平均误差（具体来说，我们测量了平均梯度强度，分为 10 个单元）。烟雾遮罩的误差如图 6(i)所示，圆形遮罩的误差如图 6(j)所示。当背景平滑时，所有算法都能很好地处理这两个遮罩。当背景包含强渐变时，全局泊松抠图的性能较差（请记住，它假定背景和背景渐变可以忽略不计）。在其余算法中，我们的算法始终产生最精确的结果。

图 7 显示了一个例子（来自文献[15]），其中 Wang 和 Cohen 的方法由于前景和背景之间的颜色模糊而未能从涂鸦中提取出一个好的遮罩。然而，当提供一个三值图，同样的方法能产生一个可接受的遮罩。同一套涂鸦，使用我们的方法产生了一个更清洁，但也不是完美的遮罩，但是添加少量额外的涂鸦可以获得更好的遮罩。

图 8 显示了另一个示例（来自文献[14]的考拉图像的特写），其中背景和前景颜色之间存在模糊。在这种情况下，我们的方法产生的遮罩明显优于 Wang-Cohen 方法产生的遮罩。为了更好地理解这种情况的原因，我们显示了一个来自 F 和 B 的涂鸦的有代表性的 RGB 柱状图。背景中的一些像素比另一些像素（例如 8(b)中标记为红色的并由 8(d)中的箭头指示）更适合前景颜色模型。因此，在第一阶段，这些像素被归类为具有高度确定性的前景。一旦出现此错误，它只会在该像素附近进一步加强错误的决定，最终导致 alpha 遮罩中出现白色块状。

由于我们的方法没有使用 F 和 B 的全局颜色模型，因此它可以处理如图 8 所示的模糊情况。然而，也存在一些非常相同的原因，我们的方法未能产生一个准确的遮罩。图 9 显示了一个女演员在两种颜色背景

前。即使黑色涂鸦 B 覆盖了这两种颜色，生成的遮罩仍包含了部分背景（头发和左边肩膀之间）。在这种情况下，用户必须在该区域添加另一个涂鸦 B。

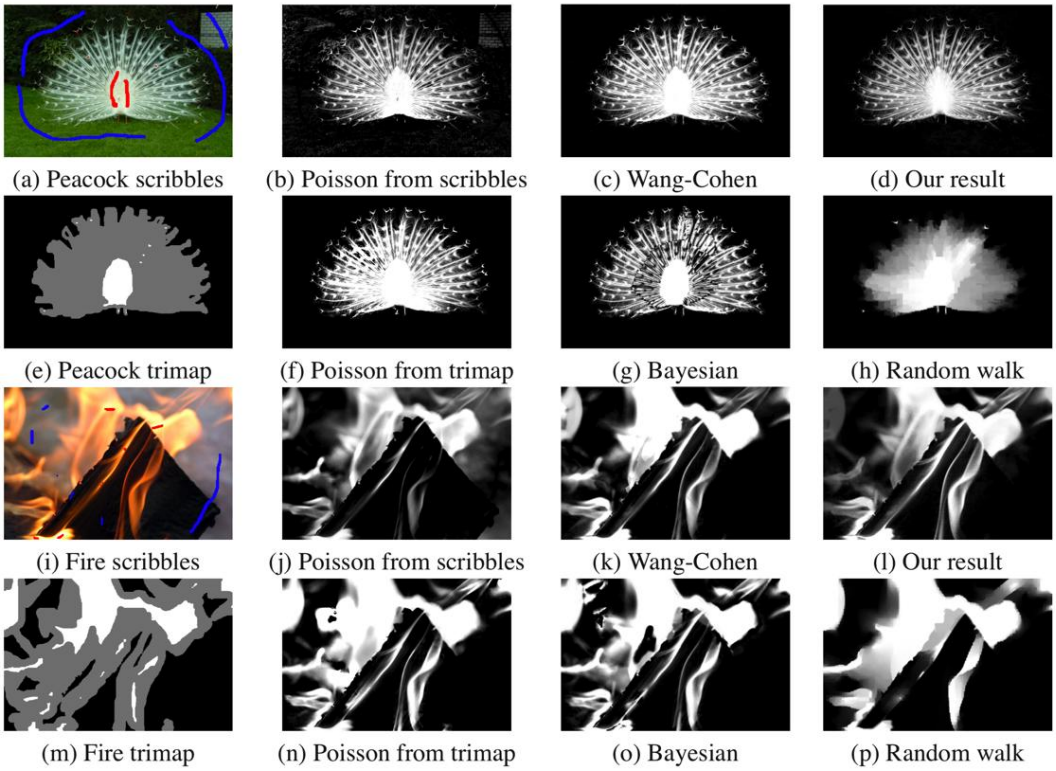


图 5

使用不同算法提取的 alpha 遮罩的比较。图像(a,c,e,g,i,k,m,o)来自文献[15]。其余图像是用我们自己的方法生成的。

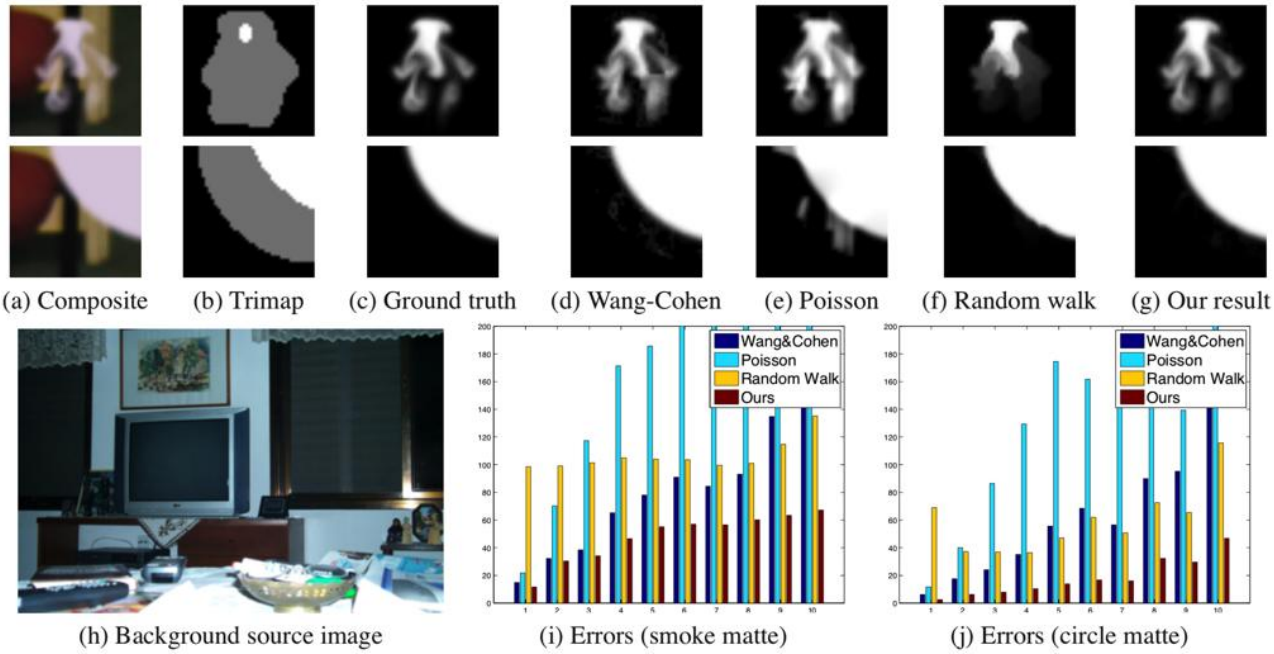


图 6

使用两个真值遮罩（ground truth matte）进行定量比较。误差(i)和(j)是使用背景的平均梯度强度作为函数绘制出的，分为 10 个单元。为了产生这些结果，我们使用了各自的方法和原始论文中指定的参数值来实现。

6.讨论

抠图和合成是图像和视频编辑的核心任务，对计算机视觉提出了重大挑战。虽然根据定义，这个过程需要用户交互，但随着用户输入量的减少，大多数现有算法的性能会迅速下降。本文在基于前景和背景颜色平滑的假设上，引入了一个代价函数，并给出了如何通过分析消除前景和背景颜色，得到 alpha 的二次代价函数。所得的代价函数与光谱方法得到的代价函数相似，但具有一个新的亲和函数，该函数是由抠图问题（matting problem）的公式推导出来的。通过解一组稀疏的线形方程组，可以有效地找到我们的代价函数的全局最小值。我们对真实图像和合成图像的实验表明，我们的算法明显优于其他不是从抠图问题推导出的二次代价函数的算法。我们的实验还表明，我们的结果与更复杂的非线性的代价函数比，是有竞争力的。然而，与以前的非线性方法相比，我们可以在几秒钟内得到解，我们可以分析证明我们的解的性质，并通过分析我们的算子的特征向量，为用户交互提供指导。

虽然我们的方法假定前景和背景颜色是平滑的，但是它不假定每个段的全局颜色分布。我们的实验表明，我们的全局平滑假设通常适用于自然图像。然而，将我们的公式扩展到包括对这两个部分的附加假设（例如，全局模型、局部纹理模型等）是很有趣的。其目标是结合更复杂的前景和背景模型，但仍然可以使用简单的数值线性代数获得高质量的结果。

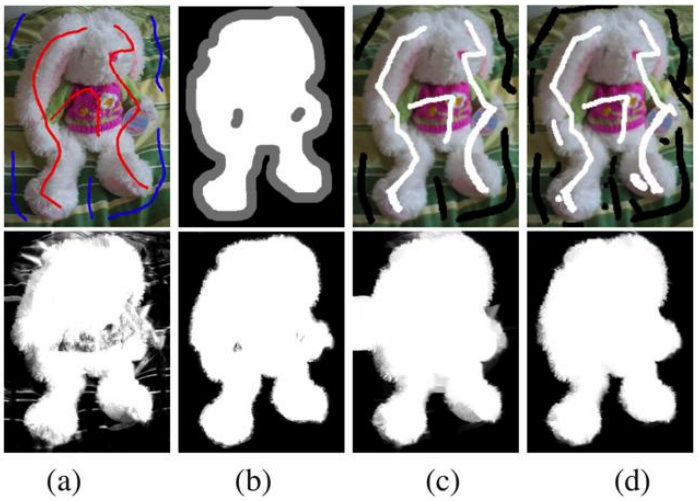
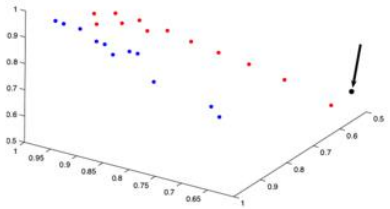


图 7

一个在背景和前景间存在颜色模糊的例子（来自文献[15]）。(a)来自文献[15]的涂鸦和遮罩；(b)来自文献[15]的使用三值图的结果；(c)我们的涂鸦和产生的结果与(a)中的相似；(d)我们增加了一些额外的涂鸦和产生的结果。

(a) Scribbles (b) Wang-Cohen (c) Our result



(d) RGB histogram of F (red) and B (blue) pixels.

图 8

F 和 B 有歧义的例子。

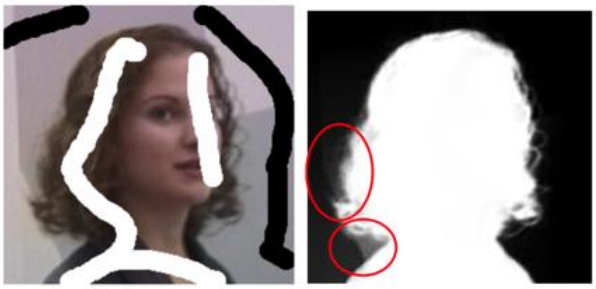


图 9

由于缺少颜色模型而失败。