摘要

本文列举了 Python 处理图像的实用工具或框架, 简述了前景背景分离的原理, 经典算法和常用框架, 并尝试使用这些框架尝试处理了图片.

1 常用的图像处理库

Python 中有许多常用的图像处理库[1], 这其中有部分是从科学计算, 矩阵处理中通用库; 也有专注于图像处理的软件包.

1.1 科学计算库

1.1.1 NumPy

NumPy 是 Python 编程中的核心库之一，提供对数组的支持。图像本质上是一个标准的 NumPy 数组，包含数据点的像素。因此，通过使用基本的 NumPy 操作，如切片、蒙版和花式索引，可以修改图像的像素值。可以使用 skimage 加载图像，使用 Matplotlib 显示图像。

1.1.2 SciPy

SciPy 是 Python 的另一个核心科学模块 (和 NumPy 一样)，可以用于基本的图像操作和处理任务。特别是子模块 scipy.ndimage 提供了在 n 维 NumPy 数组上操作的函数。该软件包目前包括线性和非线性滤波、二元形态、B-spline 插值和对象测量的函数。

1.1.3 Matplotlib

Matplotlib 主要用于二维可视化，但它也可以用于图像处理。虽然它不支持所有的文件格式，但 Matplotlib 可以有效地调整图像，从中提取信息。

1.2 图像库

1.2.1 scikit-image

scikit-image 是一个开源的 Python 包。它实现了用于研究、教育和工业应用的算法和实用程序。它是一个相当简单和直接的库，即使对于那些刚接触 Python 生态系统的人来说也是如此。其代码是高质量的，经过同行评审的，由一个活跃的志愿者社区编写。

scikit-image 通过转换原始图片，使用 NumPy 数组作[1]为图像对象。而且由于 NumPy 是用 C 语言编程建立的，所以速度非常快，使其成为图像处理的十分理想的库。

[1]这些 NumPy 数组既可以是整数（有符号或无符号），也可以是浮点数。

1.2.2 Mahotas

Mahotas 是一个 Python 的计算机视觉和图像处理和操作库。一个库是一个函数和方法的集合，它允许你执行许多操作，而不需要编写数百行代码。Mahotas 包含了许多用数组操作的算法，目前 Mahotas 有 100 多个用于图像处理和计算机视觉的函数，并且还在不断增加。

Mahotas 在寻找图像中的重复图案方面提供了很好的解决方案，例如"Where's Wally Problem " 就可以用 Mahotas 轻松解决。

1.2.3 Pillow

许多非科学的 Python 项目都利用 PIL 或 Pillow。例如，Python 网络框架 Django 使用 PIL 来表示数据库中的图像区域。如果你需要做一些快速但不优雅的图像处理，PIL 和 Pillow 有它们的位置，但如果需要认真地学习图像处理、计算机视觉和图像搜索引擎，往往会其他有更好的选择.

1.2.4 OpenCV

OpenCV[2]是计算机视觉应用最广泛的库之一。OpenCV 支持多种编程语言，如 C++、Python、Java 等，并可在 Windows、Linux、OS X、Android 和 iOS 等不同平台上使用。基于 CUDA 和 OpenCL 的高速 GPU 操作接口也在积极开发中。

OpenCV-Python 是 OpenCV 的 Python API。OpenCV-Python 不仅速度快，因为后台由 C/C++ 编写的代码组成，而且它还易于编码和部署（由于前台的 Python 封装器）。这使得它成为执行计算密集型计算机视觉程序的绝佳选择。

1.2.5 SimpleCV

SimpleCV 是另一个用于构建计算机视觉应用程序的开源框架，它提供了对 OpenCV 等多个功能强大的计算机视觉库的访问，但不必了解位深、文件格式、色彩空间等。它提供了对 OpenCV 等几个高功能计算机视觉库的访问，但不必了解位深、文件格式、色彩空间等。它的学习曲线大大小于 OpenCV 的学习曲线，正如其名称所言，它使得计算机视觉变得简单。

1.2.6 h5py

h5py 库是 Python 中存储大型数值数据集的库。它提供了对 NumPy 数组的支持。如果你有一个大型的数据集用 NumPy 数组表示，而且它不适合放在内存中，或者你想要高效、持久地存储 NumPy 数组，那么 h5py 是你的最佳选择。整个数据集永远不需要一次性从磁盘上完全加载，而且内存占用非常小，即使是数千个特征向量也是如此。

[2]Open Source Computer Vision Library

2 前景背景分离技术

影像去背（Image Matting）是一种很常见的图像处理应用[2]。简单来说，Matting 就是保留图片中需要的像素，删除其他部分。所以可以将图片分为前景和背景两部分。借由计算前景的颜色和透明度，将前景从影像中撷取出来的技术，就被称为 Image Matting, 可用于替换背景、影像合成、视觉特效，在电影工业中被广泛地使用。影像中的每个像素会有代表其前景透明度的值，称作阿尔法值（Alpha），一张影像中所有阿尔法值的集合称作阿尔法遮罩（Alpha Matte），将影像被遮罩所涵盖的部分取出即可完成前景的分离。许多图像编辑和电影后期制作应用都依赖于自然图像去背作为处理步骤之一。matting 算法的任务是准确估计图像或视频序列中前景物体的不透明度。

影像去背的主要工作就是求得精确的阿尔法遮罩 (alpha matte)，而影像常有无法人工标示的部分，例如：人的发丝或是动态模糊的部分，一种简单的解决方法是先人工标定出影像的“Trimap”，再由算法计算出阿尔法遮罩以完成影像去背。

概括出来就一个公式

$$\underset{\text{图像}}{I} = \alpha \cdot \underset{\text{前景}}{F} + (1-\alpha) \cdot \underset{\text{背景}}{B}$$

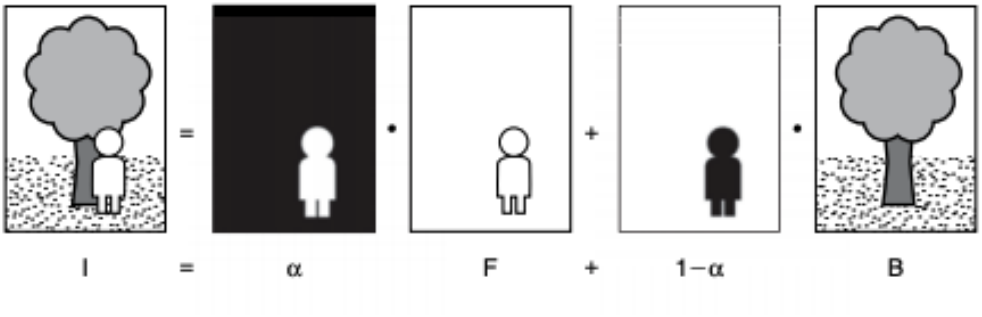


图 1: $I = \alpha \cdot F + (1-\alpha) \cdot B$ 的形象解释

当然，因为 F, B 和 α 都是未知的，要把这么多未知项都求出来显然很不容易。所以就需要增加一些附加的约束，通常，这种约束以 Trimap 的形式给出。

融合系数 α 是一个介于 0 到 1 之间的分数，它给出了前景和背景在待处理影象中所占的比例。显然，对于确定的背景部分，$\alpha = 0$；对于确定的前景部分，$\alpha = 1$。在前景与背景相互融合的边缘部分，α 基于 0 到 1 之间。而这正是最终要求解的核心问题。

2.1 术语

2.1.1 color space

色彩空间, 主要有 RGB, CMYK 和 HSV. 当然也有灰度图. 一个在二维空间的点 (图片) 在 RGB 色彩空间下可以表示为

$$I(x,y) = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

这样一个向量

RGB 三原色光模式（RGB color model），又称 RGB 颜色模型或红绿蓝颜色模型，是一种加色模型，将红（Red）、绿（Green）、蓝（Blue）三原色的色光以不同的比例相加，以合成产生各种色彩光。若加上透明度通道可形成 RGBA

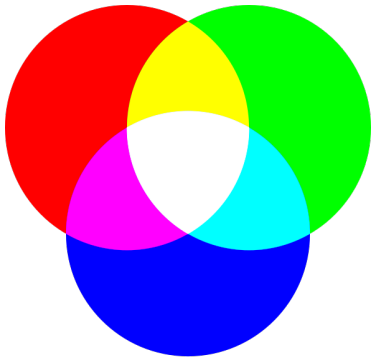


图 2: RGB

CMYK 印刷四分色模式（CMYK）是彩色印刷时采用的一种套色模式，利用色料的三原色混色原理，加上黑色油墨，共计四种颜色混合叠加，形成所谓“全彩印刷”。

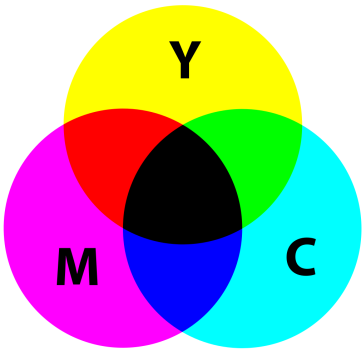


图 3: CMYK

- **C**yan 青色，常被误称为“天蓝色”或“湛蓝”

- **M**agenta 洋红色，又称为“品红色”
- **Y**ellow 黄色
- blac**K** 黑色，此处缩写使用最后一个字母 K 而非开头的 B，是因为在整体色彩学中已经将 B 给了 RGB 的 Blue 蓝色。

HSV 和 HSL HSL 和 HSV 都是一种将 RGB 色彩模型中的点在圆柱坐标系中的表示法。这两种表示法试图做到比基于笛卡尔坐标系的几何结构 RGB 更加直观。

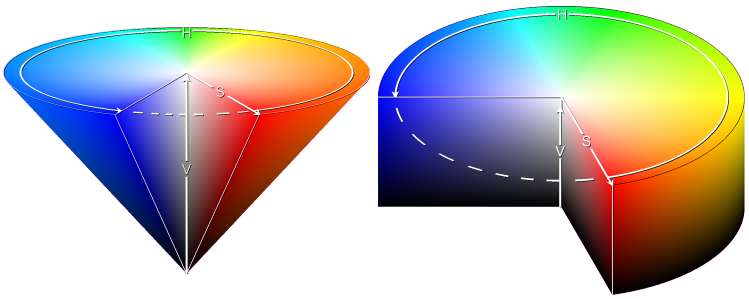


图 4: HSV(左图) 和 HSL(右图)

HSL 即色相、饱和度、亮度（英语：Hue, Saturation, Lightness）。HSV 即色相、饱和度、明度（英语：Hue, Saturation, Value），又称 HSB，其中 B 即英语：Brightness。

- 色相（H）是色彩的基本属性，就是平常所说的颜色名称，如红色、黄色等。
- 饱和度（S）是指色彩的纯度，越高色彩越纯，低则逐渐变灰，取 0-100% 的数值。
- 明度（V），亮度（L），取 0-100%。

2.1.2 alpha matte

透明度遮罩[2] (阿尔法遮罩)，是代表对于影像前景透明度的遮罩，大小和影像相同，遮罩中每个像素的值为相应的影像像素的阿尔法值（Alpha）。阿尔法值为 1 代表该像素属于前景，0 则代表该像素属于背景。阿尔法值也可能介于 0,1 之间，表示对应到的影像像素为半透明，例如烟雾、动态模糊。

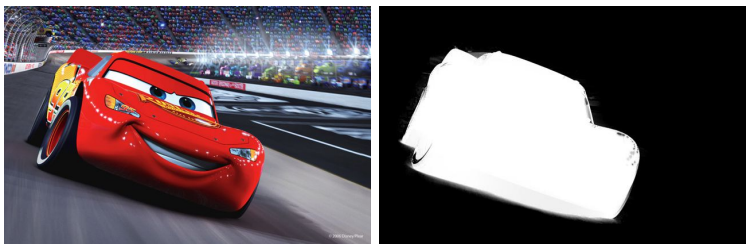


图 5: Alpha matte (右图)

2.1.3 trimap

又被译作三分图, 三值模板[2]. 一张影像的 “Trimap”，是指将影像中的每个像素划分为三种区域：前景（Foreground）、背景（Background）和待确认（Unknown）。算法会将标定好的前景和背景当成已知，再借由颜色等资讯将待确认区域中的像素标为前景或背景。

图 6: Trimap (右图)

右图中黑色为背景, 白色为前景, 而灰色就是待确认部分.

2.1.4 scribble

简单标记[2], 随便在原图划拉两下. 黑色表示背景, 白色表示前景.

图 7: scribble (右图)

2.2 算法

2.2.1 取样法

取样法 (Sample-based image matting) 会对影像的局部区域取样，从已知资讯计算出该区域的阿尔法遮罩。常见的方法有 Bayesian Matting。

Bayesian Matting 以贝叶斯计算最大后验概率，对前景、背景及阿尔法值同步进行优化，使用有向高斯共变异数（Oriented Gaussian Covariances）能有效的推估颜色的分布。贝氏去背的

优点是几率模型简单又符合直觉，去背效果良好。其限制是需要良好的“Trimap”，当影像的前后景关系变得复杂时，贝叶斯去背的效果会显著下降

在融合方程中，已知的只有 C，而 F, B 和 α 都是未知的。于是可以从条件概率的角度去考虑这个问题，即给定 C 时，F, B 和 α 的联合概率应为

$$\begin{aligned} P(F, B, \alpha \mid C) &= \frac{P(C \mid F, B, \alpha) \cdot P(F, B, \alpha)}{P(C)} \\ &= \frac{P(C \mid F, B, \alpha) \cdot P(F) \cdot P(B) \cdot P(\alpha)}{P(C)} \end{aligned}$$

其中第一个等号是根据贝叶斯公式得到的，第二个等号则是考虑 F, B 和 α 是彼此独立的。上式表明 Matting 问题可以被转化为已知待计算画素颜色 C 的情况下，如何估计它的 F, B 和 α 的值以最大化后验概率 $P(F, B, \alpha \mid C)$ 的问题。

上述等式中的右端项，需要通过取样统计的方式进行估计，而这种估计结果的准确性，很大程度上决定了算法的融合质量。具体来说，算法采用一个连续滑动的视窗对邻域进行取样，视窗从未知区域和已知区域之间的两条边开始向内逐轮廓推进，计算过程也随之推进。

2.2.2 传播法

传播法 (Propagation image matting) 是指借由分析整张影像的特性，像是颜色、梯度等，来直接求得整张影像遮罩的方法。常见的方法有 Poisson Matting、Robust Matting 等。

Poisson Matting 帕松去背分为两步，一是从影像中算出遮罩近似的梯度场，二是借由解帕松等式（Poisson Equation），从遮罩的梯度场求得遮罩。当前景和背景颜色接近时，帕松去背容易出错，此时能够以更多的使用者输入，用区域的帕松去背来进行优化。

Robust Matting 通常在有品质好的“Trimap”的情况下，取样法可以得到较好的遮罩；但在前后景关系复杂，“Trimap”品质不佳时，取样法的效果会迅速衰减。稳健性去背景法会先进行取样，得到遮罩后再进行优化，进而结合取样法和传播法的好处。

2.2.3 机器学习

大多数现有的去背方法需要绿幕背景或手动创建的”Trimap” 来产生良好的去背。自动的、无”Trimap” 的方法正在出现，但其质量仍然不够好。某些基于机器学习的 matting 算法，要求用户在拍摄时额外拍摄一张没有主体的背景照片。这一步需要少量的操作，但远比创建”Trimap” 耗时少。

- KNN Matting - IEEE
- Deep Image Matting
- Background Matting: The World is Your Green Screen
- alphamatting.com

2.3 辅助方法

在实际操作中，可以借由增加观测资讯，让去背景变得更加容易。

2.3.1 红外线

在拍摄时，同时使用一般镜头和红外线镜头，借由红外线照片所得到的资讯，将同时拍摄的一般照片当中的人与背景分离。

2.3.2 闪光

在拍摄时，拍下使用闪光灯和没有使用闪光灯各一张，闪光灯会明显改变前景的亮度，但对背景的影响较小，借由分析有无闪光的两张照片，来完成影像去背。

2.4 影像前景背景分离技术

如果可以给定多张图像 (视频), 我们就可以利用运动分析 (Motion analysis) 和物体追踪 (Object tracking) 来进行前景背景分离.

前景检测 (Foreground detection) 是计算机视觉和图像处理领域的主要任务之一，其目的是检测图像序列的变化。背景去除 (Background subtraction) 就是任何允许提取图像的前景以进行进一步处理（物体识别等）的技术。

许多应用不需要知道视频序列中运动演变的所有信息，而只需要知道场景的变化信息，因为图像的兴趣区是其前景的物体（人、车、文字等）。在图像预处理阶段（可能包括图像去噪、形态学[3]等后期处理）后，需要进行物体定位，这可能会利用这种技术。

前景检测根据前景发生的这些变化将前景和背景分开。它是一套典型的分析用固定摄像机实时记录的视频序列的技术。

2.4.1 算法

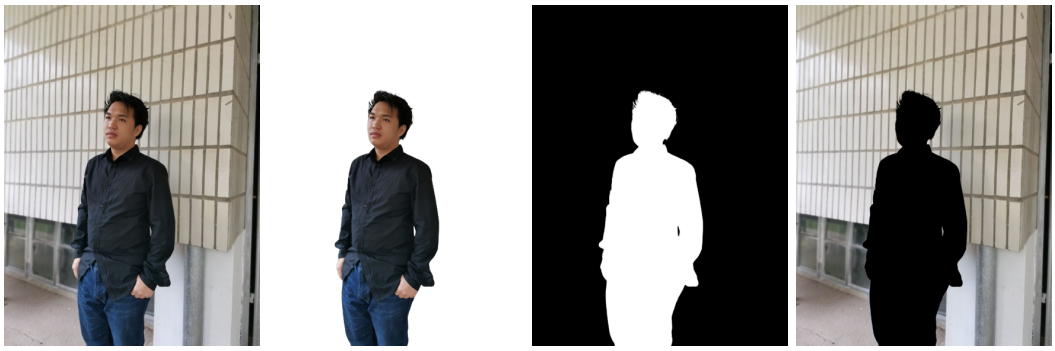
OpenCV 中自带有几个算法和示例代码

- MOG
- MOG2
- GMG
- KNN
- CNT

[3] Mathematical morphology

3 前景背景分离实践

本次实践使用了removebg的 API 生成的图片作为测试基准, 如图 8所展示的效果.



(a) 原图 (b) Cutout (c) Alpha 遮罩 (d) 背景图

图 8: 使用 removebg API 的处理示例

因得到的背景图像的实用价值较前景来说较低, 故以下省略了分离之后的背景的展示 (如图 8最右图), 而增加一项使用新的背景图来替代原图的测试 (如图 9).

(a) 原图 (b) 背景图 (c) 合成图像

图 9: 背景图处理示例

3.1 GrabCut

GrabCut 算法[3] 利用了图像中的纹理（颜色）信息和边界（反差）信息，只要少量的用户交互操作即可得到比较好的分割结果。只需要在目标外面画一个框，把目标框住，它就可以完成良好的分割; 因为 GrabCut 是按颜色分布和边缘对比度来分割图片的，对一些常见的与此原则相悖的图片，效果确实不好。比如前景人物的帽子、鞋、墨镜，通常颜色跟前景主体有较大区别；再如前景中的孔，有可能由于颜色区分和边缘的对比度不足，导致边缘的惩罚占上风，而没有扣出来背景。

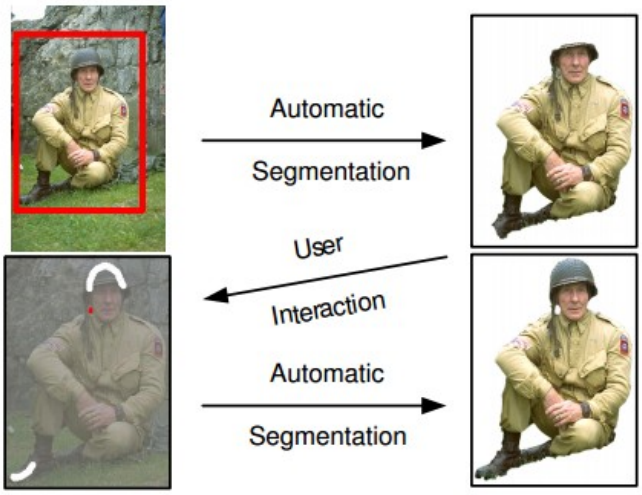


图 10: GrabCut 的 User Edit 方法

所以，GrabCut 还是保留了人工修正的操作，定义了两种标记：绝对是背景和可能是前景。对分割错误人工修正后 (如图 10)[4]，分割还是可以比较准确的。

参考官方文档 Foreground Extraction using GrabCut Algorithm，代码见`grabcut.py`

(a) 原图　(b) 带范围框的原图　(c) Alpha 遮罩　(d) 合成图像

图 11: GrabCut 算法的处理结果

3.2 PyMatting

Github 文档见A Python Library for Alpha Matting[4].

给定一个输入图像和一个手绘或者自动生成的 Trimap，使用 Alpha 去背方法估计前景对象的 Alpha 通道 (Alpha 透明度遮罩)，然后可以组成不同的背景。

PyMatting 提供以下 Alpha 去背方法

- Closed Form Alpha Matting
- Large Kernel Matting

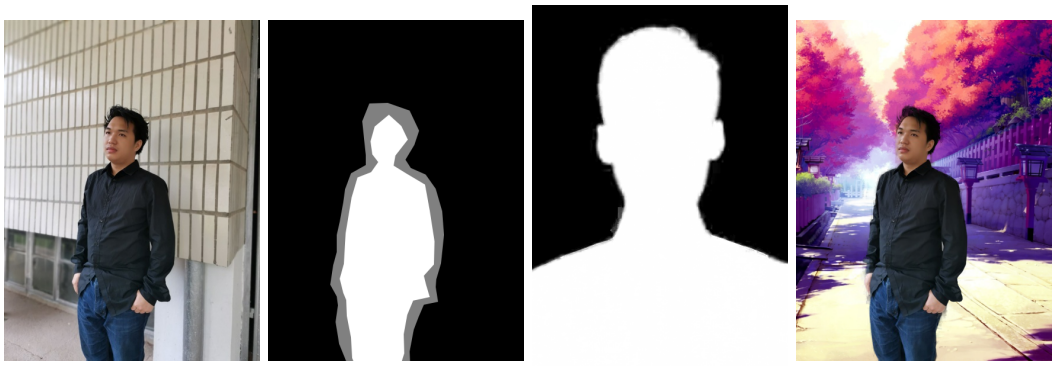
[4]时间所限，本次实践对 user edited 修正方法尚未涉及

- KNN Matting
- Learning Based Digital Matting
- Random Walk Matting

和以下前景估计实现

- Closed Form Foreground Estimation
- Fast Multi-Level Foreground Estimation

本示例中使用了 Closed Form Alpha Matting[5] 方法[5] (由于时间所限, 尚未测试其他方法), 代码见`pymat.py`



(a) 原图 (b) Trimap (c) Alpha 遮罩 (d) 合成图像

图 12: PyMatting 库的处理结果

3.2.1 Trimap 的处理

通常我们得到的 Trimap 可能会包含不止三种[6]种颜色. 直接丢进去 PyMatting 会报错, 故需要进行预处理得到只有三种颜色的 Trimap. PyMatting 已经自带有`fix_trimap`函数. 就是根据

$$T_{\text{fixed}} = \begin{cases} 0, & T < \text{上阈值} \\ \frac{1}{2}, & \text{其他} \\ 1. & T > \text{下阈值} \end{cases}$$

计算得到的

3.2.2 背景提取

值得注意的是 PyMatting 中的`foreground.estimate_foreground_ml module`函数[7]可以返回经过估计的背景 (而非留黑), 如图 13所示.

[5] `alpha.estimate_alpha_cf`函数
[6] 黑, 白和 50% 灰之外可能会因为图像处理软件的抗锯齿算法混进去其他颜色
[7] 即上文提到的 Fast Multi-Level Foreground Estimation 的实现

图 13: 左图为原图, 右图为估计背景

3.3 Close-Form Matting

Github 文档见Closed-Form Matting

就是上文 PyMatting 库的 Closed Form Alpha Matting[5] 另一个库的实现, 可以做到根据 scribble 就能进行前景背景分离, 虽然最后的效果不如前两者理想.

代码见`closed_form.py`

(a) 原图 (b) Scribble (c) Alpha 遮罩 (d) 合成图像

图 14: Close-Form Matting 库的处理结果

3.4 性能比较

本次主要从处理时间和结构相似性指标两个方面对以上介绍的三种方法进行性能上的比较. 当然还有其他指标没有涉及到, 如处理时的内存占用以及 MSE, PSNR 等参数.

3.4.1 处理时间

使用 Powershell 自带命令`Measure-Command`进行计时. 结果如图 15.

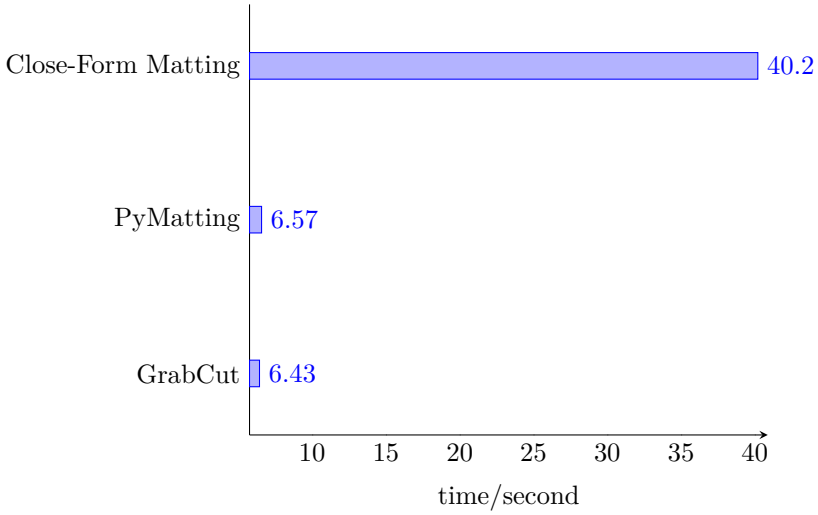


图 15: 处理所需时间 (越短越好)

3.4.2 SSIM

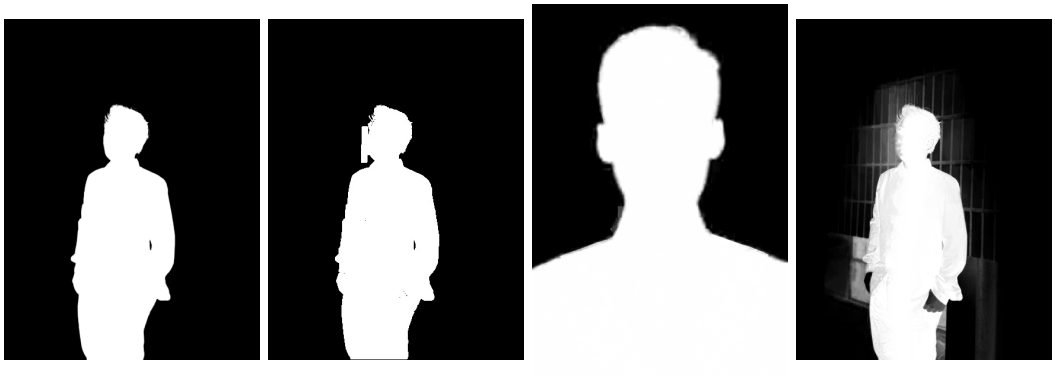
结构相似性指标（structural similarity index，SSIM index)[6] 是一种用以衡量两张数位影像相似程度的指标。当两张影像其中一张为无失真影像，另一张为失真后的影像，二者的结构相似性可以看成是失真影像的影像品质衡量指标。

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

- μ 表示均值
- σ 表示方差 (协方差)

scikit-image 已经包含了`compare_ssim` 函数. 将基准图像的 Alpha 遮罩 (图 8c) 和其余算法的结果进行对比, 结果如图 18.

代码见`compare.py`



(a) 基准图像 (b) GrabCut (c) PyMatting (d) Close-Form Matting

图 16: Alpha 遮罩对比

(a) GrabCut (b) PyMatting (c) Close-Form Matting

图 17: SSIM 差值图对比

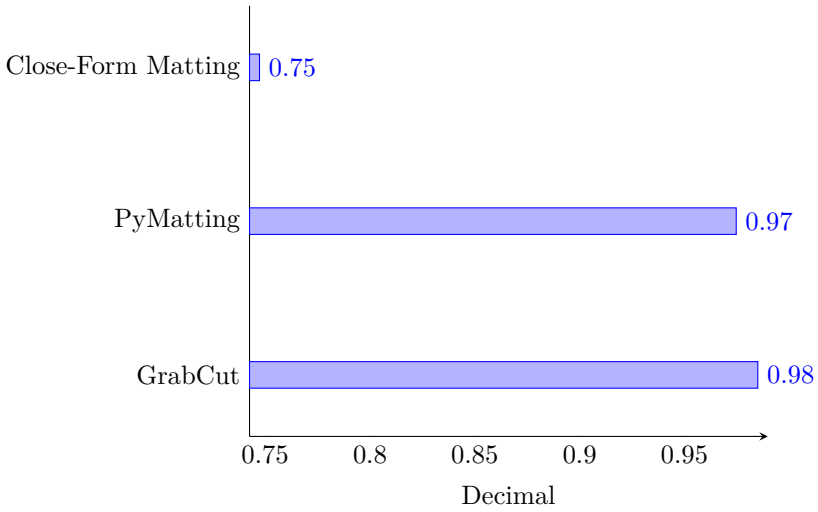


图 18: 与基准图像的 SSIM (越高越好)

3.5 小结

对上面介绍的工具进行优缺点的总结

3.5.1 PyMatting

PyMatting 作为一个综合工具, 在利用基于 Close-Form 的 Trimap 算法的基础上其效果在三个库中应该是最好的. 缺点在于 Trimap 的绘制还是稍显麻烦.

3.5.2 GrabCut

GrabCut 出现了多截取了一部分的情况, 但也可以通过用户的补充输入来去除, 以达到更好的效果.

3.5.3 Close-Form Matting

而最后一个同样利用利用基于 Closed Form Alpha Matting, 但 Scribble 得到的信息可以说是比 Trimap 少很多的, 能达到如此效果也实属不易, 但其优点在于输入简单. 缺点在于运行时间仍可以再优化.

4 结语

由于时间所限, 本文未详细讨论利用机器学习 (如 TensorFlow, Keras) 来进行前景背景分离, 亦没有基于对影像视频的运动检测的前景分离进行分析.

我个人更加注重 Python 在 Web 开发中的应用; 使用 Python 语言编写的 Flask 作为 Web 服务器，也能够运行 Python 语言编写的 Web 程序。可以使以上技术在 Web 后端实现处理, 前端处理用户输入, 如此则可以更加便利地使用这些前景背景分离的工具, 实现实用价值.

参考文献

[1] PANDEY P. 10 Python image manipulation tools[Z]. https://opensource.com/article/19/3/python-image-manipulation-tools. 2018.

[2] 潘俊林. 图像前背景分离的研究与实现[D]. 复旦大学.

[3] ROTHER C, KOLMOGOROV V, BLAKE A. GrabCut -Interactive Foreground Extraction using Iterated Graph Cuts[J/OL]. ACM Transactions on Graphics (SIGGRAPH), 2004. https://www.microsoft.com/en-us/research/publication/grabcut-interactive-foreground-extraction-using-iterated-graph-cuts/.

[4] GERMER T, UELWER T, CONRAD S, et al. PyMatting: A Python Library for Alpha Matting[Z]. 2020. arXiv: 2003.12382 [cs.CV].

[5] LEVIN A, LISCHINSKI D, WEISS Y. A Closed-Form Solution to Natural Image Matting[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 30(2): 228-242.

[6] WANG Z, BOVIK A C, SHEIKH H R, et al. Image Quality Assessment: From Error Visibility to Structural Similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600-612. DOI: 10.1109/TIP.2003.819861.