Fast Image/Video Upsampling 读书报告

一、相关工作

[Baker and Kanade2000; Tipping and Bishop 2002] 尝试利用亚像素替代的方法从多个低分辨率的图像中得到一个高分辨率图像，和 Fast Image/Video Upsampling 中的框架最后一步像素替代模型相同，但只用了一个图像输入。

[Freeman et al. 2002] 使用基于学习的图像超分辨率策略和识别方法实现了高分辨率信息的推断。在图像上采样后像素缺失的填补具有很重要的意义。

Kopf et al. [2007] 对在图像平面上获得的结果进行上采样。首先在低分辨率下计算该解，然后使用联合双边滤波进行上采样。

[Avidan and Shamir 2007] 不通过缩放来调整图片的大小，而是通过将原图像中“不那么重要的”（我的理解：非边缘像素）的像素进行插入。

[Baker and Kanade 2000] 通过模拟图像形成的逆过程来完成本论文。（从高像素到低像素的逆过程）

二、算法过程

论文提出了一种基于反馈控制的，简单且高效率的上采样方法来自动提高图像的分辨率，并且该方法可以自然而然地可以运用到视频中去。通俗来说，就是将图像/视频进行放大并提高清晰度。那么问题来了，对于单图片输入来说，如何得到一个放大的图像，如何对放大图像缺失的像素进行填补呢？答案是显而易见的：从已知的像素中进行推断并填补到周围像素中。但随之而来的问题

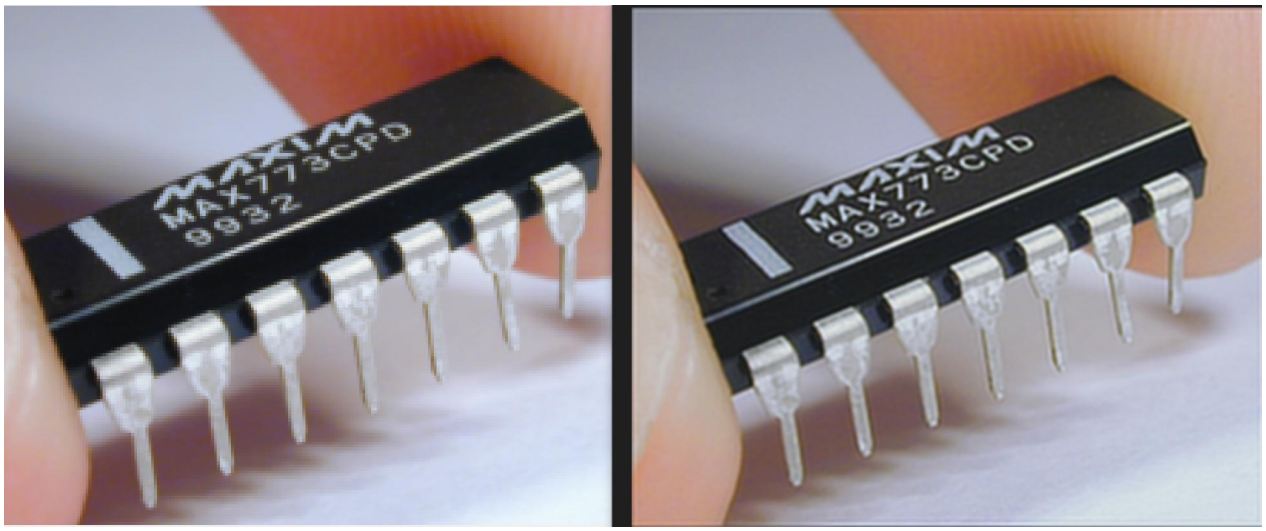
是，如果知道填补的结果是否合理呢？首先，论文中提到，上采样是指将图像上采样到更高分辨率，是一种把低分辨率图像采样成高分辨率图像的技术手段。单图像上采样最简单的办法就是使用如 bilinear,bicubic 和 Lanczos 算法等，这些算法容易实现且跑的很快，但会出现的问题就是如果使用一个较大的因子时会产生一个模糊的结果，而基于实例的方法对计算机和所提供的实例要求很高。因此该论文的目标就是在解决这些问题的同时，以 Baker 和 Kanade 所提出的图像形成过程为基础，并将该过程进行逆转。

Feedback-Control Upsampling 的过程：

输入一个低分辨率的图像，首先将 RGB 转换为 YUV 的颜色空间，然后只对 Y 通道进行上采样，然后使用双三次插值算法将其变成所需要的分辨率，然后将这个得到图像，在反馈控制循环中进行不断迭代，图像不断变清晰。而反馈控制循环包括三个部分，分别是非盲去卷积，重复卷积和像素抽取。在像素抽取的过程中，可以理解为对高分辨率图像 H 采样 d 倍得到 L，也就是说图像 H 中每 d 个像素只保留一个最后得到 L，因此在循环中，如果上采样因子为 n，我们将会用 L 中的（i，j）替代掉 H 中的（$n \times i+1, n \times j+1$）。这样做的用处在于：（1）模拟了图像形成过程中均匀像素抽取过程（2）在重复卷积的过程中，不断地将去卷积和循环的结果进行传播，使得 H 中缺失的像素信息，会被相邻像素代替，这样最终得到的图像 H 相当于是一个高斯滤波处理过后的图像。

三、代码

为了对整个算法的过程有更加深入的了解，而文章没有提供源代码，网上也没有关于 python 的代码，因此我用 python 对这该算法进行了复现。（其中数学部分没有办法实现，因此在 GitHub 找到有相关代码并使用 Matlab 进行计算（该部分代码来自于 https://github.com/zzeitt/forFastUpsampling-reproduce），其余 python 代码实现均由本人完成。得到的结果如下：**（左图是没有经过任何处理得到的图像，右图为经过本人 python 处理后的图像）**



四、评价

优点：（1）利用 YUV 通道进行计算，可以同时兼容灰度图和彩色图的。

（2）用 L 中的（i，j）替代掉 H 中的（$n \times i+1, n \times j+1$），保留的低分辨率图像信息。

缺点：

（1）处理速度慢，虽然论文中说部分的处理时间只花在少量的快速傅里叶变换（FFT）操作上，但运用其提供的 exe 执行文件处理需要 GPU 参与。并且通过对迭代过程中后台的输出，发现真正影响速度的是预处理部分，包括将 RGB 转换成 YUV 和双三次插值算法进行上采样。

（2）由于没有源代码，根据我们自己写的 python 代码还得到这个缺点：如上图所示，肉眼可见的，图像的分辨率得到的很大的提高（但会出现**色差**，原因推测是在 python 和 matlab 中对于 RGB 和 YUV（Ycbcr）色彩空间有一定误差造成）但是会在图像的边缘处出现黑色像素。我们推测以下原因：这是由于在像素替代的过程中，作者假定卷积核为固定的高斯核，而使用 filter2D 函数对

边缘处理是补 0，而像素没有成功在边缘处进行替代，从而出现黑点。

（3）由于该算法采用的是**非盲去卷积方法**，因此需要在已知 PSF 的情况下才可以恢复出清晰的图像。而文中将该模糊核假定为高斯核（但并没有给出证明或原因）而实际上很多时候，我们无法提前测定 PSF，只有一张模糊的照片。

针对上述我们发现的问题提出改进：

针对缺点（3）：在反馈控制循环中，我认为可以利用**盲去卷积**的方法去模糊可能得到更好的结果并且适用范围更加广泛。或者先利用变分贝叶斯，估计卷积核 K 之后再采用非盲去卷积的方法得到清晰图像。

针对缺点（2）：纵观整个算法的过程，我认为整个过程就是在对图像放大后进行边缘提取与增强。因此提出理论可行建议：先对图像进行高通滤波处理后得到高频部分，再添加回原图上从而强化边缘达到锐化的效果。

由于复现论文代码花费太多时间，因此这两个改进还在理论阶段。