研究论文

Michael Feischl 和 Hubert Hackl*

通过最优网格细化实现自适应图像压缩

https://doi.org/10.1515/sample-YYYY-XXXX 收稿日期：YYYY年MM月DD日；修订日期：YYYY年MM月DD日；录用日期：YYYY年MM月DD日

arXiv:2304.01640v1 [math.NA] 4 Apr 2023

摘要： JPEG算法是图像压缩领域事实上的标准。我们研究了能否利用自适应网格细化来优化压缩比，并提出了一种新的自适应图像压缩算法。我们证明，对于给定的误差范数，该算法以很高的概率生成准最优细分网格。该细分网格所需的存储开销极小，因此可实现高效的压缩算法。实验结果表明，对于许多图像，在视觉质量没有明显损失的情况下，新算法能够达到优于标准 JPEG 压缩的压缩比。

这项工作的数学核心表明，在假设图像像素受到小幅加性高斯噪声影响时，Binev最优树逼近算法以高概率适用于图像压缩。

关键词： 自适应网格细化、最优性、图像压缩、JPEG

1 引言

JPEG算法几十年来一直是标准的图像压缩算法之一。它将任意给定的图像分解为由 8×8 个像素块组成的固定网格。在每个块上，图像都会经过离散二维余弦变换。随后，利用量化矩阵 Q 对得到的系数序列进行量化，并执行舍入步骤；该步骤的设计方式是使较高频率以较低精度存储（或完全省略）。与图像的位图表示相比，这显著减少了存储空间。对于通常不会出现明显颜色或亮度变化的照片而言，这种压缩方法尤其有效。本研究旨在以最优方式自适应生成背景网格，而非使用固定的块大小 8×8，从而提高压缩率。网格细化算法的灵感来源于用于近似偏微分方程的自适应网格细化，尤其是自适应有限元法。

偏微分方程的自适应网格细化最早于 20 世纪 80 年代和 90 年代针对有限元方法进行了数学分析。I. M. Babuška 通过开发有限元方法的首批后验误差估计量 [BM87]，掀起了一股数学研究热潮。这开辟了数值分析的一个新领域，并促成了大量研究活动；这里仅提及著作 [AO00、Ver13、Rep08]。十多年后，Dörfler [Dör96] 基于误差估计量，针对二维情形提出了首个严格的网格细化策略；又过了五年，Morin、Nochetto 和 Siebert [MNS00] 给出了首个无条件收敛性证明。

自适应策略的最优性，即渐近意义下能够生成最优网格这一事实，最早由 Babuška和Vogelius 在一维情形下证明 [BV84]；对于 $d \geq 2$，Binev、Dahmen 和 DeVore 于 2004 年针对泊松模型问题的自适应有限元法给出了证明 [BDD04]。在此基础上，Stevenson [Ste07]以及 Cascon、Kreuzer、Nochetto 和 Siebert [CKNS08] 简化了算法并扩展了问题类别。我们

Michael Feischl, 维也纳工业大学，分析与科学计算研究所，奥地利，michael.feischl@tuwien.ac.at***通讯作者：Hubert Hackl,** 维也纳工业大学，分析与科学计算研究所，奥地利，hu- bert.hackl@tuwien.ac.at

关于自适应网格细化的概述，请参阅 [CFPP14, Fei22] 。本研究受到自适应树逼近的启发；自适应树逼近是 [BD04] 提出并在 [Bin18]中扩展到hp-设置的一类自适应算法的重要构建模块（尤其适用于需要粗化网格的时变问题）。自适应树逼近生成的网格能够以近乎最优的精度与成本比逼近给定的目标函数。我们使用该算法进行 JPEG 压缩：在矩形网格上逼近目标图像（可视为在每个像素上保持常数的函数），并在每个矩形上通过离散二维余弦变换对图像进行标准量化处理，这与传统 JPEG 压缩的做法相同。

我们提出两种算法：第一种算法接收给定的图像 I，并生成一个向量 S_I，该向量对压缩图像进行编码。第二种解码算法通过（几乎）逆转向量 S_I 中的压缩步骤，生成原始图像的近似图像 I_{app}。这些算法满足以下性质：1. I_{app} 是 I 的近似，即满足

$$\|I - I_{\mathrm{app}}\| < \tau$$

对于某个范数和预设容差 τ。2. S_I 所需的存储空间是（近似）最小的。3. 编码和解码算法的运行时间是（近似）最优的。

文献中还可以找到其他几种自适应版本的 JPEG：在 [EAS14 中，RW96], 会自适应地选择每个 8×8 子网格上的压缩质量。在 [TTQ09], 中，通过一种基于插值误差的贪心算法来选择自适应网格；该算法也用于 [UJQT12] 中的图像压缩，但并未证明该算法具有（近似）最优性（也无法预期其具有这种性质）。

本文其余部分安排如下：第 2 节中，我们为一般范数建立自适应算法。第 3 节中，我们将该算法应用于 L^2-范数，并推导出一个近最优性结果。第 4 节讨论压缩图像的有效存储，第 5 节则在实际图像上测试该算法。

1.1 JPEG 压缩算法

JPEG 压缩算法最早于 20 世纪 90 年代提出（并在 [Sta94] 中标准化），此后成为图像压缩的事实标准。

压缩算法将图像转换到 YCbCr 色彩空间，该空间将颜色分为蓝色色差、红色色差以及经过伽马校正的亮度通道。由于人眼对亮度变化比对颜色变化更加敏感，因此各通道采用不同的质量参数进行压缩。不过，所有通道使用的基本算法相同：图像被划分为由 8×8 个像素组成的块，并对这些块进行离散余弦变换，得到一个包含系数的8×8 矩阵。该矩阵中的每个元素都除以同尺寸的所谓量化矩阵 Q 中对应的元素。这个矩阵由 JPEG 标准预先定义，用于控制算法的压缩比。随后将相除的结果舍入到最接近的整数，这通常会使高频系数中出现许多零。得到的整数序列再通过无损游程编码算法进一步压缩。请注意，舍入步骤是压缩算法中唯一不可逆的步骤，因此也是 JPEG 产生质量损失的原因（更多细节和参考文献见例如 [HLN+18] ）。

2 自适应压缩算法

下面，我们将推导自适应 JPEG 压缩的数学表述。

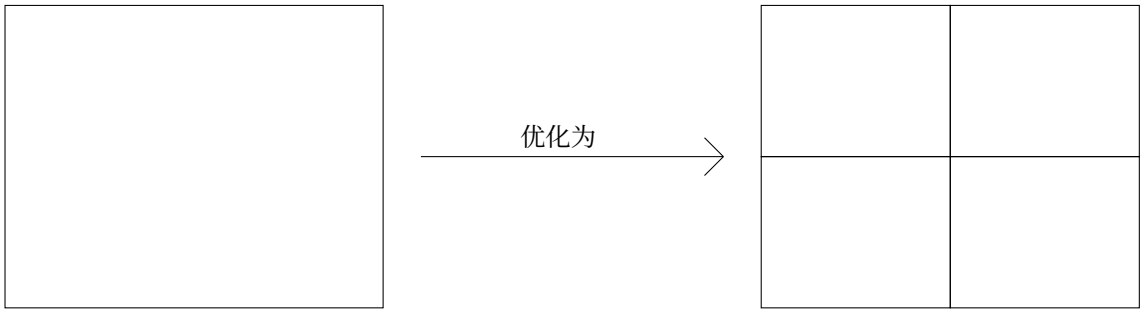

图 1： 元素通过对其边进行两次二分细化为四个子元素。

2.1 图像的表示方式

我们假设这幅图像的高度为 h 、宽度为 w 像素，可表示为一个函数:

$$I : \{1, ..., h\} \times \{1, ..., w\} \to \mathbb{R}^3,$$

其中， $I(i,\ j)$ 对应于像素（$i,\ j$）的 RGB 值，它们可能会被归一化为 $[0, 1]$（不过，具体的编码方式并不影响下文）。像素的坐标经过设定，使得像素（$1,\ 1$）是左上角像素，而（$h,\ w$）是图像的右下角像素，因此我们可以将图像视为一个大小为 $h \times w$、值属于 $\mathbb{R}^3$ 的矩阵。

获取近似图像 I_{app} 的主要思想是将图像划分为较小的元素，并通过经典的 JPEG 策略对这些元素进行近似。

为此，我们将元素 R 定义为$\{1, \dots, h\} \times \{1, \dots, w\}$ 的矩形子集，其纵横比与原始图像相同，即

$$R := \{j_R, \dots, j_R + h_R - 1\} \times \{k_R, \dots, k_R + w_R - 1\} \quad \text{with } h_R/w_R = h/w.$$

纵横比条件并非绝对必要，但它使网格的高效存储更为容易，因为只需存储一个维度和元素的位置即可。不失一般性，我们可以假设 w 和 h是 2 的幂，以确保存在可容许的元素。由元素R组成且构成$\{1, \dots, h\} \times \{1, \dots, w\}$ 不相交并集的集合称为网格 $\mathcal{T}$。对于网格 $\mathcal{T}$，我们考虑在该网格上计算得到的近似图像 $I_{\text{app},\mathcal{T}} = I_{\text{app}}$ ； 上下文明确时，我们省略其对 $\mathcal{T}$的依赖。用 $\mathcal{T}_0$表示初始网格，它仅由一个元素$\mathcal{T}_0 = \{R_0\}$ 组成，并覆盖整个图像，即 $R_0 = \{1, \dots, h\} \times \{1, \dots, w\}$。

最后，我们用 $I|_R$ 表示图像在给定元素 R 上的限制算子。为了生成最优适配的网格，我们考虑一种通过二分法实现的简单细化规则，如图 1 所示。给定网格 $\mathcal{T}$，细化元素 $R \in \mathcal{T}$ 会生成一个新网格 $\mathcal{T}'$，使得

$$\mathcal{T}' \setminus \mathcal{T} = \{R_1, \dots, R_4\},$$

即，我们不考虑网格闭包（因为下面的逼近问题完全是局部的）。注意，由于对元素纵横比的假设，所有新创建的元素在几何上都与 R_0 相似。

如果网格 $\mathcal{T}$可以通过对 $\mathcal{T}$执行细化步骤得到，则称网格 $\mathcal{T}$较粗 于网格 $\mathcal{T}'$。此时也称 $\mathcal{T}'$ 比 $\mathcal{T}$ 更细，并记为 $\mathcal{T} < \mathcal{T}'$ 或 $\mathcal{T}' > \mathcal{T}$。基于这一概念，我们进一步定义集合。

$$\mathbb{T} := \big\{\mathcal{T} : \mathcal{T}_0 < \mathcal{T}\big\}$$

and

$$\mathbb{T}_n := \big\{\mathcal{T} \in \mathbb{T} : \#\mathcal{T} - \#\mathcal{T}_0 \le n\big\}.$$

请注意，尽管两个网格可能无法比较，我们所选的细化规则仍能保证局部嵌套性。以下结果众所周知，这里重述一遍以方便读者。

引理 1. 设 $\mathcal{T}\prime\mathcal{T} \in \mathbb{T}$。任意元素 $R \in \mathcal{T}$ 均满足以下条件之一： $R \subseteq R'$，其中某个 $R' \in \mathcal{T}\prime o \quad r$

$$R = \bigcup_{\substack{R' \in \mathcal{T}' \\ R' \subseteq R}} R'. \tag{1}$$

证明。如果 (1) 不成立，则必定存在某个元素 $R' \in \mathcal{T}\prime$ 满足 $R \cap R' \neq \emptyset$ 和 $R' \not\subseteq R$。由于 R 和R' 均由 R_0 开始生成，因此我们得出结论 $R \subseteq R'$。 □

2.2 自适应算法

我们采用第 1.1 节所述的 JPEG 策略，将图像转换到 YCbCr 色彩基底，从而得到三个图像分量 I^Y、I^{C_b} 和 I^{C_r}。我们通过计算由像素构成的 2×2块的均值，将 C_b 和 C_r 分量下采样至图像尺寸的一半。然后，将下文的自适应算法（算法 2）应用于各个分量。

自适应算法的总体策略是从 $\mathcal{T}_0$ 开始，迭代细化网格，直到给定范数下的压缩误差足够小。为此，我们需要能够引导细化的局部误差。

定义 1. 令 $\|.\|$ 为我们在大小为 $h \times w$, I 的图像空间上选定的范数， $h \times w$ 为大小为 $\mathcal{T}$ 的图像，且 $\mathcal{T}$ 为一个网格。我们定义：
(i) 局部误差

$$\eta(R) := \|(I - I_{\mathrm{app},\mathcal{T}})|_R\| \qquad \forall R \in \mathcal{T} \tag{2}$$

(ii) 全局误差

$$E(\mathcal{T}) := \|(I - I_{\mathrm{app},\mathcal{T}})\| \tag{3}$$

(iii) 对 n 个元素的最佳逼近误差

$$E_n := \min_{\mathcal{T} \in \mathbb{T}_n} E(\mathcal{T}).$$

由于下面的自适应算法本质上是一种贪心算法，对于大多数误差范数 $\|\cdot\|$，这些直接的定义无法得到最优结果。因此，有必要先考虑一种修正误差，该误差最早由 [BD04] 提出并分析。

我们用 $\widetilde{\eta}$ 表示修正局部误差，并令初始元素 R_0 的 $\widetilde{\eta}(R_0) = \eta(R_0)$。令 $\mathcal{T} \in \mathbb{T}$，并假设 $\widetilde{\eta}(R)$ 已经定义。我们将子元素 $R_i,\ i = 1,..,4$ 的 $\widetilde{\eta}(R_i)$ 定义为

$$\widetilde{\eta}(R_i)^2 := \frac{\sum_{i=1}^4 \eta(R_i)^2}{\eta(R)^2 + \widetilde{\eta}(R)^2} \widetilde{\eta}(R)^2.$$

这意味着 $\widetilde{\eta}$ 在兄弟元素之间是常数。这个定义背后的一般思想是，通过这里引入的与父元素之间的关系，修正误差隐式地取决于元素层级，即生成该元素所需的细化次数。这使算法具有非局部行为，而标准贪心算法无法实现这种行为；事实证明，这对于证明过程 [BD04]的最优性至关重要。

现在我们可以给出自适应算法的表述。

算法 2. 输入： 大小为 $h \times w$ 的图像分量 I，其中 h、w 均为 2 的幂；初始网格$\mathcal{T}_0$、容差 $\tau > 0$、计数器 $\ell = 0$
当 $E(\mathcal{T}_\ell) > \tau$ 时，执行：
1. 计算所有 $R \in \mathcal{T}_\ell$ 的修正误差 $\widetilde{\eta}(R)$。2. 若满足 $\widetilde{\eta}(R) = \max_{R' \in \mathcal{T}_\ell} \widetilde{\eta}(R')$
且 $\min\{h_R, w_R\} \geq 16$，则标记 $R \in \mathcal{T}_\ell$。3. 细化标记元素以生成 $\mathcal{T}_{\ell+1}$，
并设置 $\ell = \ell + 1$。
输出： 网格序列 $\mathcal{T}_\ell$。

备注 1。请注意，该算法不会生成小于 8×8 的元素 R，也就是说，它生成的网格不会比标准 JPEG 网格更精细。

本节的主要目标是运用 [BD04]中建立的框架，证明算法 2 的近最优性。

定义 2. 设某个算法在某个（经过自适应细化的）网格 $\mathcal{T}_I \in \mathbb{T}$上，计算给定图像 I 的一个近似。若存在正常数 C_1、C_2，使得对于 $n, k \in \mathbb{N}$ 满足 $k \leq C_1 n$，则称该算法是近最优的（就细化次数而言）。

$$\mathcal{T}_I \in \mathbb{T}_k \quad \Longrightarrow \quad E(\mathcal{T}_I) \leq C_2 E_n.$$

这些常数是普适的，即不依赖于图像 I。

3 L L^2 范数下的自适应 JPEG 压缩

本节中，我们使用算法 2，其中 $\|\cdot\|$ 表示以 $(wh)^{-1/2}$ 加权的 L^2 范数，即像素总数的倒数平方根。一个自然的问题是，L^2 范数是否恰当地反映了人类对图像质量损失的容忍度（答案很可能是否定的，参见例如 [WB06]）；关于此问题的进一步讨论和实验，请参阅第 3.3 节和第 5 节。

3.1 网格元素上的图像压缩

给定网格 $\mathcal{T}$上的近似图像的生成方式与原始 JPEG算法类似，即我们使用离散余弦变换（DCT）和量化矩阵 Q 对图像进行量化。我们使用 JPEG 标准 [Sta94]中定义的量化矩阵：

$$Q := \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}.$$

下面，我们将引入一些算子来形式化自适应压缩：设 R 为一个大小为 $h \times w$的元素。我们定义一个限制算子，将其限制到 R的左上角 8×8 块

$$\begin{aligned} \mathrm{TL}_R &\colon \mathbb{R}^{h \times w} \to \mathbb{R}^{8 \times 8} \\ &(a_{ij})_{1 \leq i \leq h, 1 \leq j \leq w} \mapsto (a_{ij})_{1 \leq i,j \leq 8} \end{aligned} \tag{4a}$$

以及它的右逆，即通过补零将 $\mathbb{R}^{8\times 8}$ 嵌入算子到 $\mathbb{R}^{h\times w}$ 中，

$$\mathrm{TL}_R^{-1} \colon \mathbb{R}^{8\times 8} \to \mathbb{R}^{h\times w}, \quad \mathrm{TL}_R \mathrm{TL}_R^{-1} = \mathrm{Id} \colon \mathbb{R}^{8\times 8} \to \mathbb{R}^{8\times 8}. \tag{4b}$$

此外，令 DCT_R 表示作用于 $\mathbb{R}^{h\times w}$ 上的二维离散余弦变换，IDCT_R 表示其逆变换，即

$$\begin{aligned} &\mathrm{DCT}_R : \mathbb{R}^{h\times w} \to \mathbb{R}^{h\times w} \\ &(a_{ij})_{1\leq i\leq h, 1\leq j\leq w} \mapsto \Big(\frac{2}{\sqrt{hw}} C(i)C(j) \sum_{k=0}^{h-1} \sum_{l=0}^{w-1} a_{kl} \cos\frac{(2k+1)i\pi}{2h} \cos\frac{(2l+1)j\pi}{2w}\Big)_{1\leq i\leq h, 1\leq j\leq w} \end{aligned} \tag{4c}$$

$$\mathrm{IDCT}_R : \mathbb{R}^{h\times w} \to \mathbb{R}^{h\times w}$$
$$(a_{ij})_{1\le i\le h,1\le j\le w} \mapsto \Big(\frac{2}{\sqrt{hw}}\sum_{k=0}^{h-1}\sum_{l=0}^{w-1} C(k)C(l)a_{kl}\cos\frac{(2k+1)i\pi}{2h}\cos\frac{(2l+1)j\pi}{2w}\Big)_{1\le i\le h,1\le j\le w}, \tag{4d}$$

其中，若 $n=0$，则 $C(n)=1/\sqrt{2}$；否则 $C(n)=1$。

注意，这些算子仅取决于 R 的大小，而与 R 在图像中的位置无关。由于算法 2 中不会出现大小小于 8×8 的元素 R，因此 TL_R 始终有定义。下文中，令符号 $\odot$ 和 $./$ 分别表示同尺寸矩阵的逐项乘法和逐项除法。给定图像 I 和网格 $\mathcal{T}$，我们通过以下方式计算近似图像 $I_{\mathrm{app},\,\mathcal{T}}^{\mathrm{final}}$

$$I_{\mathrm{app},\mathcal{T}}^{\mathrm{final}}|_R = \mathrm{IDCT}_R\Big(\mathrm{TL}_R^{-1}\big(Q\odot\mathrm{round}(\mathrm{TL}_R(\mathrm{DCT}_R(I|_R))./Q)\big)\Big) \quad \text{for all } R\in\mathcal{T}. \tag{5}$$

然而，对于自适应过程，我们将忽略量化，因此仅以这种方式计算最终网格对应的近似图像。对于自适应过程中出现的网格 $\mathcal{T}'<\mathcal{T}$，我们只需将图像限制为每个元素 $R\in\mathcal{T}'$上的前 8×8 个 DCT 系数，从而计算近似图像 $I_{\mathrm{app},\,\mathcal{T}'}$，即:

$$I_{\mathrm{app},\mathcal{T}'}|_R = \mathrm{IDCT}_R\Big(\mathrm{TL}_R^{-1}(\mathrm{TL}_R(\mathrm{DCT}(I|_R)))\Big) \quad \text{for all } R\in\mathcal{T}'. \tag{6}$$

在接下来的章节中，我们将只使用 (6)；当目标是高效存储并重建近似图像时，再回到 (5)。因此，(2)–(3) 中定义的局部误差和全局误差可写为

$$\eta(R)^2 := \frac{1}{hw}\sum_{(i,j)\in R}(I_{\mathrm{app},\mathcal{T}}(i,j)-I(i,j))^2$$

and

$$E(\mathcal{T})^2 := \sum_{R'\in\mathcal{T}}\eta(R')^2 = \frac{1}{hw}\sum_{(i,j)\in I}(I_{\mathrm{app},\mathcal{T}}(i,j)-I(i,j))^2.$$

3.2 细化性质

算法 2 最优性的关键在于所谓的 细化性质 （文献中也称为估计量的次可加性 ）：存在一个通用常数 $C_0>0$ 使得

$$\sum_{i=1}^{4}\eta(R_i)^2 \le C_0\eta(R)^2 \qquad \text{with } R_1,..,R_4 \text{ children of } R \quad \text{for all } R\in\mathcal{T}\in\mathbb{T}. \tag{7}$$

本质上，细化性质 (7) 表明，细化网格不会显著增加近似误差。遗憾的是，如图 2 所示，在我们的应用中，这一结论一般并不成立。不过，我们证明，在对图像分布作出某些假设的条件下，对于给定图像，细化性质 (7) 以很高的概率成立。

引理 3. 令 $\Phi_{k,\lambda}$ 表示自由度为 $k\in\mathbb{N}$、非中心参数为 $\lambda\ge 0$的非中心卡方分布的累积分布函数。则，对于所有 $0\le x<\infty$ 和所有 $\lambda\ge\lambda'$，有 $\Phi_{k,\lambda}(x)\le\Phi_{k,\lambda'}(x)$。

证明。 我们使用众所周知的恒等式 $\Phi_{k,\lambda}(x) = 1-Q_{k/2}(\sqrt{\lambda},\sqrt{x})$，其中 Q 表示 Marcum Q 函数。根据 [Esp68，公式 (2)], , 该导数可表示为

$$\partial_\mu Q_{k/2}(\mu,y) = \mu\big(Q_{k/2+1}(\mu,y)-Q_{k/2}(\mu,y)\big).$$

结合 [ATM09，公式（6）], 可得

$$\partial_\lambda\Phi_{k,\lambda}(x) = \partial_\lambda(1-Q_{k/2}(\sqrt{\lambda},\sqrt{x})) = -\frac{1}{2\sqrt{\lambda}}\left(\sqrt{\lambda}\left(\frac{\sqrt{x}}{\sqrt{\lambda}}\right)^{k/2}\mathrm{e}^{-\frac{\lambda+x}{2}}I_{k/2}(\sqrt{x\lambda})\right), \tag{8}$$

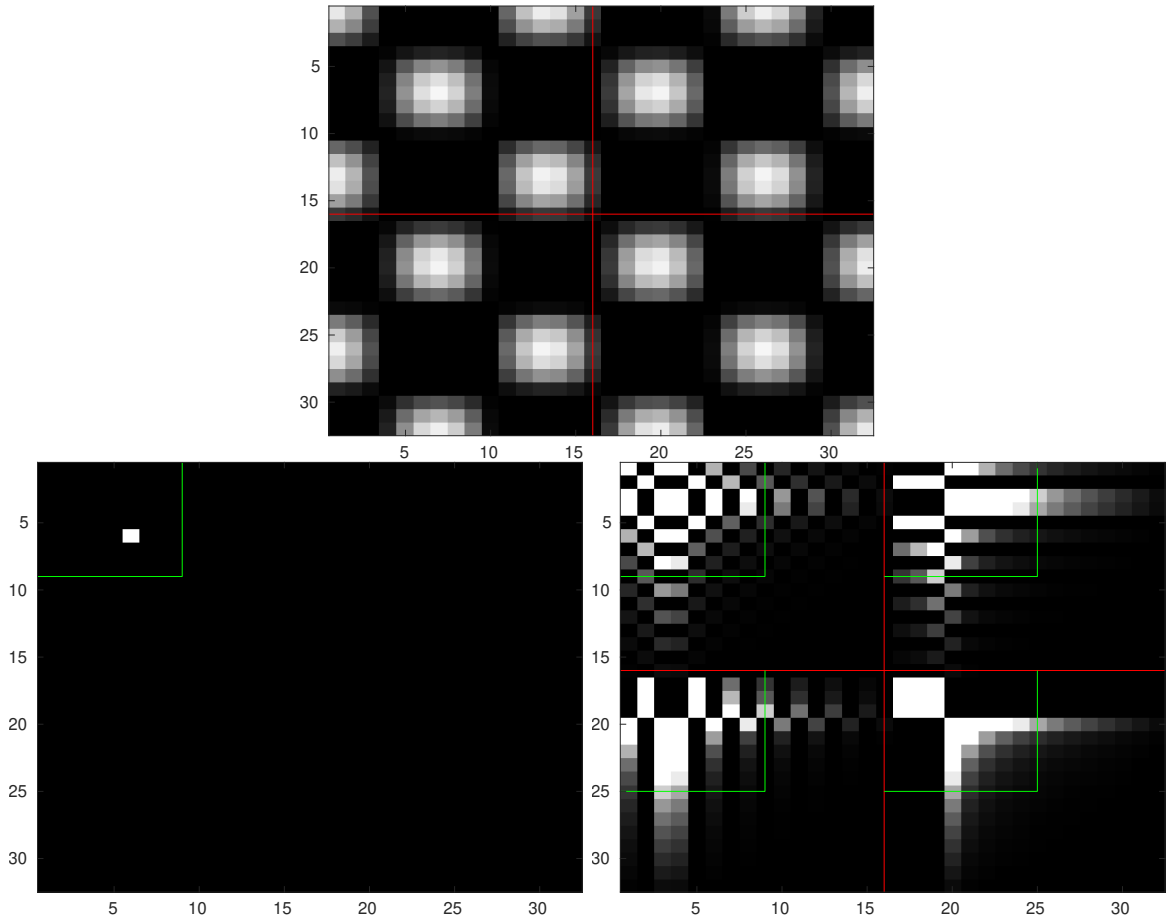

图 2： 左下：元素 R 的系数矩阵 $\mathrm{DCT}_R(I|_R)$，大小为 32×32，且仅有一个非零系数 (5，5)，幅值为 15。上方：对应的图像 $I|_R$，其子元素 R_1、.....、R_4 以红色标出。右下：系数矩阵 $\mathrm{DCT}_{R_i}^{-1}(I|_{R_i})$。绿色线条标出不对 $\eta(R)$ 或 $\eta(R_i)$ 作出贡献的 8×8 系数所在的块。因此，对于此例，(7) 的右侧为零（左侧图像中绿色块之外没有非零系数），而左侧不为零。

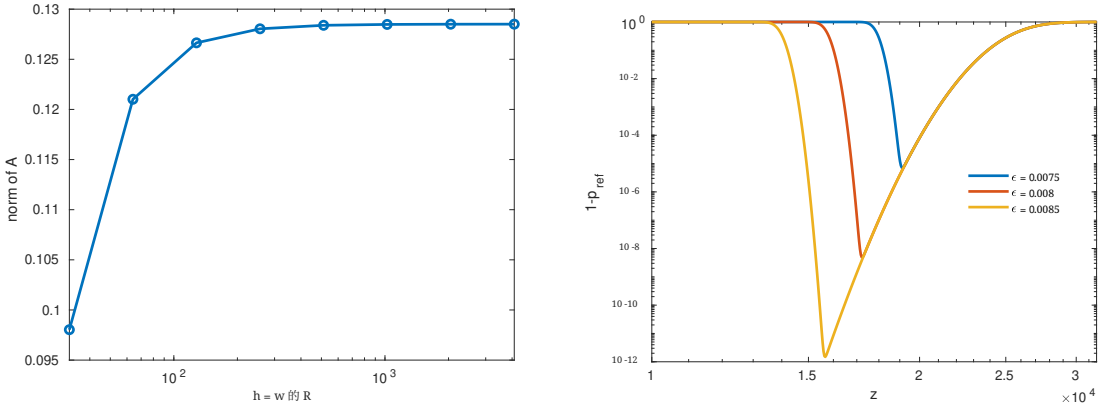

图 3： 左图：不同尺寸的 A 范数。右图：不同 ε 和 $z \geq$ 值下 $1 - p_{\mathrm{ref}}$ 的上界 0.

其中，第一类修正贝塞尔函数定义为

$$I_{k/2} := \sum_{k=0}^{\infty} \frac{1}{k!\Gamma(k+k/2+1)} \left(\frac{x}{2}\right)^{2k+k/2}.$$

这表明 $\partial_\lambda \Phi_{k,\lambda}(x) \leq 0$ 对所有 $0 \leq x < \infty$ 均成立，并由此完成证明。 □

定理 4. 设 $I \in \mathbb{R}^{h\times w}$ 表示原始图像，$\widetilde{I} \in \mathbb{R}^{h\times w}$ 表示通过向第 i 个像素添加 $\varepsilon\zeta_i$ 而得到的随机失真，其中 ζ_i 为独立同分布的标准正态随机数。那么，对于所有C、$z \geq 0$，细化性质(7) 以概率

$$p_{\text{ref}} \geq \left(1 - \Phi_{448,0}\left(\frac{\delta^2 z}{C^2 - \delta^2}\right)\right)\Phi_{64,\varepsilon^{-2}}(z) \quad \textit{for all } C, z \geq 0$$

且对所有 $R \in \mathcal{T} \in \mathbb{T}$均成立。常数 δ 满足 $\delta \leq 1$，但对于实际图像尺寸，其数值可追踪至$\delta \leq 0.13$。

证明。设 R 表示一个元素，$\widetilde{R}$ 表示 R的一个子元素。回顾上文 (4) 中定义的算子 $\mathrm{TL}_{\widetilde{R}}$、$\mathrm{TL}^{-1}_{\widetilde{R}}$、$\mathrm{DCT}_{\widetilde{R}}$ 和 $\mathrm{IDCT}_{\widetilde{R}}$。对于 $i \in \{1,..,4\}$，我们考虑以下算子

$$\begin{aligned} &A_{\widetilde{R}} \colon R \to \widetilde{R} \\ &A_{\widetilde{R}} := (\mathrm{Id} - \mathrm{TL}^{-1}_{\widetilde{R}} \circ \mathrm{TL}_{\widetilde{R}}) \circ \mathrm{DCT}_{\widetilde{R}} \circ \mathrm{P}_{\widetilde{R}} \circ \mathrm{IDCT}_R \circ \mathrm{TL}^{-1}_R \circ \mathrm{TL}_R \circ \mathrm{DCT}_R, \end{aligned}$$

其中，$\mathrm{P}_{\widetilde{R}}$ 表示 R 到 R 的限制算子，以及

$$\begin{aligned} &A \colon R \to R \\ &A(I|_R)|_{R_i} := A_{R_i}(I|_R), \quad \text{for } i = 1, \ldots, 4. \end{aligned}$$

如下文所示，算子 A_R 刻画了粗单元 R 与细单元 R_i上的误差之差。由于算子 A_{R_i} 由正交投影、嵌入算子和（逆）离散余弦变换复合而成，因此它们在 R、R_i中一致有界。给定 R_i 和 R，我们可以通过计算相应的矩阵范数，精确求出它们的范数。显然，有$\|A\| \leq \delta = 1$，但我们可以针对各种实际图像尺寸数值跟踪该范数（结果见图 3）。这表明 $\delta = 0.13$ 是一个切合实际的上界，并且与 $\delta = 1$相比，改进了定理陈述中的常数。请注意，证明中其余所有论证在采用朴素上界 $\delta = 1$时仍然成立。

我们旨在说明，对于大多数图像，细化性质 (7) 均成立。根据假设，对于独立同分布的标准正态随机变量$\xi_{i,j}$，有$\widetilde{I}(i,j) = I(i,j) + \varepsilon\xi_{i,j}$。注意，$I$ 是一个随机变量，其 L^2-范数的分布服从非中心卡方分布。局部误差 $\eta(R_i)$可以视为 I_{R_i} 与 R_i上相应近似图像之差。因此，对于 R的所有子节点 R，我们计算

$$\begin{aligned} \sum_{i=1}^{4} \eta(R_i)^2 &= \sum_{i=1}^{4} \|(\mathrm{Id} - \mathrm{TL}^{-1}_{R_i}\mathrm{TL}_{R_i})\mathrm{DCT}_{R_i}\mathrm{P}_{R_i}(\widetilde{I})\|_2^2 \\ &\leq \sum_{i=1}^{4} 2\|(\mathrm{Id} - \mathrm{TL}^{-1}_{R_i}\mathrm{TL}_{R_i})\mathrm{DCT}_{R_i}\mathrm{P}_{R_i}\mathrm{IDCT}_R(\mathrm{Id} - \mathrm{TL}^{-1}_{R}\mathrm{TL}_{R})\mathrm{DCT}_R(\widetilde{I})\|_2^2 \\ &\quad + 2\|A(\widetilde{I})\|_2^2 \\ &\leq \sum_{i=1}^{4} 2\|\mathrm{P}_{R_i}\mathrm{IDCT}_R(\mathrm{Id} - \mathrm{TL}^{-1}_{R}\mathrm{TL}_{R})\mathrm{DCT}_R(\widetilde{I})\|_2^2 \\ &\quad + 2\|A(\widetilde{I})\|_2^2 \\ &\leq 2\|(\mathrm{Id} - \mathrm{TL}^{-1}_{R}\mathrm{TL}_{R})\mathrm{DCT}_R(\widetilde{I})\|_2^2 + 2\delta\|\widetilde{I}\|_2^2 = 2\eta(R)^2 + 2\delta\|\widetilde{I}\|_2^2, \end{aligned}$$

其中我们使用了三角不等式和恒等式

$$\mathrm{Id} = \mathrm{IDCT}_R(\mathrm{Id} - \mathrm{TL}^{-1}_{R}\mathrm{TL}_{R})\mathrm{DCT}_R + \mathrm{IDCT}_R(\mathrm{TL}^{-1}_{R}\mathrm{TL}_{R}(\mathrm{DCT}_R)).$$

接下来，我们证明对于 $C > 0$

$$\delta/C\|\widetilde{I}\|_2 = \delta/C\|\mathrm{DCT}_R(\widetilde{I})\|_2 \le \|(\mathrm{Id} - \mathrm{TL}_R^{-1}\mathrm{TL}_R)\mathrm{DCT}_R(\widetilde{I})\|_2 = \eta(R) \tag{9}$$

以高概率成立。我们用 $0 \le p_{\mathrm{ref}} \le 1$ 表示该概率。注意，将噪声加到 $\sim I$ 上，等价于将噪声加到 $\mathrm{DCT}_R(I)$ 上，因为离散余弦变换只是正交归一基的变换。因此，接下来只需估计其范数

$$\Big((\mathrm{Id} - \mathrm{TL}_R^{-1}\mathrm{TL}_R)\widetilde{J}\Big)_{ij} = \begin{cases} 0 & i \le 8 \text{ or } j \le 8, \\ \widetilde{J}_{ij} & \text{else}, \end{cases}$$

其中 $\widetilde{J} := (\mathrm{DCT}_R(I)_{i,j} + \varepsilon\xi_{i,j})$。因此，只需估计以下情况的概率

$$\delta/C\|\widetilde{J}\|_2 \le \|(\mathrm{Id} - \mathrm{TL}_R^{-1}\mathrm{TL}_R)\widetilde{J}\|_2.$$

令 $A = \big\{(i,j) \in \mathbb{N}^2 \colon 1 \le i \le h_R, 1 \le j \le w_R\big\}$ 且 $B := A \setminus \{1,\dots,8\}^2$，可将其改写为

$$p_{\mathrm{ref}} = \mathbb{P}\Big(\delta^2/C^2 \sum_{(i,j)\in A} (\mathrm{DCT}_R(I)_{i,j} + \varepsilon\xi_{i,j})^2 \le \sum_{(i,j)\in B} (\mathrm{DCT}_R(I)_{i,j} + \varepsilon\xi_{i,j})^2\Big).$$

不失一般性，我们可以假设 $\|I\|_2 = 1$，因此 $\sum_{1\le i,j\le 8} \mathrm{DCT}_R(I)_{i,j}^2 \le 1$，由此得到

$$p_{\mathrm{ref}} \ge \mathbb{P}\Big(\delta^2/C^2 X \le (1 - \delta^2/C^2) Y\Big),$$

其中 $X = \sum_{1\le i,j\le 8} (R\,I)_{ij}/\varepsilon + \xi_{ij})^2$ $Y = \sum_{(i,j)\in B} \varepsilon^{-1} (R\,I)_{ij} + \xi_{ij}^2$ 是相互独立的随机变量：$\mu = \delta^2/C^2$，ϕ_X 和 ϕ_Y 分别为 X 和 Y 的概率密度函数和累积分布函数。由此得到

$$\begin{aligned} p_{\mathrm{ref}} &= \int_0^\infty \int_{\mu x/(1-\mu)}^\infty \phi_Y(y)\,dy\phi_X(x)\,dx = \int_0^\infty \Big(1 - \Phi_Y(\frac{\mu x}{1-\mu})\Big)\phi_X(x)\,dx \\ &\ge \int_0^\infty \Big(1 - \Phi_{448,0}(\frac{\mu x}{1-\mu})\Big)\phi_X(x)\,dx, \end{aligned}$$

其中，最后一个不等式用到了引理 3，此外还用到了 Y 至少包含 $32 \cdot 16 - 64 = 448$ 个项这一事实。这是因为算法 2 只会细化满足 $h_R,\ w_R \ge 16$ 的元素。对于 $h_R = w_R = 16$，根据定义，7 的左侧为零。由于根据假设，w_R 和 h_R 都是 2 的幂，因此下一个最小元素 R 满足 $\max\{h_R,\ w_R\} \ge 32$，从而 $\#B \ge 32 \cdot 16 - 64 = 448$。对于 $z \ge 0$，我们可以估计

$$\begin{aligned} p_{\mathrm{ref}} &\ge 1 - \int_0^\infty \Phi_{448,0}(\frac{\mu x}{1-\mu})\phi_{64,0}(x)\,dx \ge 1 - \int_0^z \Phi_{448,0}(\frac{\mu x}{1-\mu})\phi_X(x)\,dx - (1 - \Phi_{64,\varepsilon^{-2}}(z)) \\ &\ge (1 - \Phi_{448,0}\Big(\frac{\mu z}{1-\mu}\Big))\Phi_{64,\varepsilon^{-2}}(z). \end{aligned}$$

综合这些估计，我们以概率 p_{ref} 得到

$$\sum_{i=1}^4 \eta(\widetilde{R}_i)^2 \le (2 + 2C)\eta(R)^2.$$

证明至此完成。 □

非中心卡方分布的累积分布函数（CDF）可见于例如 [JH99]，也可通过 Matlab 中的命令 `ncx2cdf` 进行数值计算。因此，我们可以评估概率界

在 $\delta = 0.13$ 和 $C = 1$ 取现实值时，定理 4 可得此概率界。这意味着以概率满足细化性质 (7)，其中 $C_0 = 4$

$$p_{\text{ref}} \geq \begin{cases} 1 - 7.3 \cdot 10^{-6} & \varepsilon = 0.0075, \\ 1 - 5.4 \cdot 10^{-9} & \varepsilon = 0.008, \\ 1 - 1.6 \cdot 10^{-12} & \varepsilon = 0.0085. \end{cases}$$

关于 z，定理 4 中 p_{ref} 的下界表现见图 3。

备注 2. 我们注意到，定理 4 表明，噪声幅度为 $\varepsilon = 0.0075$ 时，以 99.99% 的概率满足细化性质 (7)，其中 $C_0 = 4$。我们还要强调，即使噪声幅度为 0.015，这种噪声也不会被肉眼察觉，如图像 4 所示。这表明，不满足 (7) 的图像只是罕见的例外。



图 4： 上：原始图像。下：添加了振幅为 0.015 的额外噪声后的原始图像。图像来自 unsplash.com，摄影师为 Kadyn Pierce。

以下结果阐述了算法 2 的最优性。

定理 5. 设 I 使得细化性质 (7) 成立。那么，对于算法 2 的所有迭代 $\ell = 0$、1、……，网格 $\mathcal{T}_\ell$ 满足

$$E(\mathcal{T}_\ell) \le 12(2+2C)^2 E_m \quad \text{for all } m \le \#\mathcal{T}_\ell/10, \tag{10}$$

其中，$C > 0$ 是定理 4 中的常数。为创建 $\mathcal{T}_\ell$，算法至多使用 $12(2+2C)^2(\#\mathcal{T}_\ell - \#\mathcal{T}_0 + 1)$ 次算术运算，并计算 η_ℓ。

证明。由于细化性质 (7) 成立，该结论可由 [BD04，定理 5.2], 推出，其中子元素的数量为 $K = 4$。我们注意到，我们对 $\mathbb{T}_n$ 采用了略有不同的定义，因此 E_n也有所不同。在 [BD04] 中，$\mathbb{T}_n$ 表示恰好经过 n 次细化所生成的网格集合，而我们统计的是元素数量。由于每次细化恰好生成三个新元素，细化次数对应于 $(\#\mathcal{T} - \#\mathcal{T}_0)/3$。因此，这两种定义等价，上述结论得证。

□

备注 3。 利用备注 2 中的 $\varepsilon = 0.0075$ 和 $C = 1$，定理 4–5 推出，对于所有 $m \le \#\mathcal{T}_\ell/10$，$E(\mathcal{T}_\ell) \le 192E_m$ 的成立概率超过 99。99%。

注 4. 在算法 2 中，使用最大的 $\widetilde{\eta}(R)$ 可能需要复杂度为 $O(N\log N)$ 的排序过程。为了使操作次数保持在 $O(N)$ 阶，我们可以使用二进制分箱代替排序。相应地修改结果及其证明即可，定理估计中的绝对常数会有所变化。

3.3 其他范数下的自适应压缩

一种可能更适合捕捉图像质量损失的范数是 BV-范数。对此的一种启发式解释是，人眼对尖锐的颜色／亮度变化更敏感，而这些变化无法通过 L^2-范数捕捉。函数 $u \in BV \subset L^1((0,h]\times(0,w])$ 的 BV-范数定义为

$$\|u\|_{BV} := \|u\|_1 + V(u), \tag{11}$$

其中

$$V(u) := \sup\left\{ \int_{(0,h]\times(0,w]} u(x)\mathrm{div}(\phi(x))dx : \phi \in C_c^1((0,h]\times(0,w],\mathbb{R}^n),\ \|\phi\|_\infty \le 1 \right\}$$

表示 u 的全变差。对于弱可微函数 $w \in L^1$，即 $w \in W^{2,\,1}$ BV范数由w 给出。

$$\|w\|_{BV} = \|w\|_1 + \|\nabla w\|_1.$$

然而，在当前情形下，图像 I 和 I_{app} 表示在图像的每个像素 □ 上恒定的函数。对于这样的函数 u，其总变差计算为

$$V(u) = \sup_{\|\phi\|_\infty\le 1} \sum_{\square\in I} \int_{\square} u\mathrm{div}(\phi)\,dx = \sup_{\|\phi\|_\infty\le 1} \sum_{\square\in I} \int_{\partial\square\cap I^\circ} un\cdot\phi\,dx = \sum_{e\in F^\circ} \|[u]\|_{L^1(e)},$$

其中，n 表示 $\partial\square$上的单位法向量，I° 表示 $\mathbb{R}^2$ 中由 I的像素占据的开子集，F° 表示像素之间的内部界面集合。我们还使用常见记号$[u]|_e = u|_\square - u|_{\square'}$表示 u 在两个像素 □ 和 □′之间的界面 $e = \square\cap\square' \in F^\circ$ 上的跳跃。因此，近似误差（$I - I_{\mathrm{app}}$）的 BV 范数可写为

$$\|I - I_{\mathrm{app}}\|_{BV} = \frac{1}{hw}\Big(\|I - I_{\mathrm{app}}\|_1 + \sum_{e\in F^\circ} \big|[I - I_{\mathrm{app}}]|_e\big|\Big).$$

新的范数会产生新的局部误差和全局误差。

$$\eta(R) = \|(I - I_{\mathrm{app}})|_R\|_{BV} \quad \text{and} \quad E(\mathcal{T}) = \sum_{R\in\mathcal{T}} \eta(R) \tag{12}$$

可用于驱动算法 2。

注释 5. 由于我们考虑的是有限维空间上的范数，因此可以立即得到 L^2-范数与 BV-范数等价。因此，对于大多数图像，细化性质 (7) 对 BV-范数也成立，近最优性结果定理 5 亦然。然而， BV-范数对应的 (7) 中的常数取决于范数等价常数，而后者又取决于图像大小。直接计算该常数似乎非常困难，因为这需要研究离散余弦变换作为作用于 BV 上的算子时的映射性质。

4 自适应压缩的高效存储

必须存储算法 2 生成的自适应网格，以便重建压缩图像。与使用预定义网格的标准 JPEG 压缩相比，这可能会产生额外开销。不过，我们将证明，采用一些简单的优化后，这种开销并不显著。

4.1 存储 I_{app}

算法 2 分别应用于各个颜色分量 $X \in \{Y,\ C_b,\ C_r\}$，并生成相应的网格$\mathcal{T}^X$。由于自适应算法中忽略了 JPEG 压缩的舍入步骤，因此对于 I_{app}、 I_{app}^X的每个元素 $R \in \mathcal{T}^X$，我们按以下方式压缩 X 分量：

– 计算 $F^X(R) := \text{round}(\text{TL}_R(\text{DCT}_R(I^X|_R))./Q)$ – 按照图 5 所示的顺序，将 $F^X(R)$ 中的条目存储到向量 S_R^X中，直至最后一个非零条目

请注意，离散余弦变换会返回一个由基函数系数组成的矩阵，其中靠近底部和右侧的系数对应于频率更高的基函数。典型图像中对应高频基函数的系数通常较小，这意味着它们在舍入后会变为零并被舍弃。

要从向量 S_R^X重建 I_{app}^X，我们还需要存储 I_{app}的宽度 w、I_{app} 的比率 h/w 以及 $R \in \mathcal{T}^X$的宽度。此外，我们还需要一些记作 sep 的分隔位，以便标记 S_I中下一个元素的起始位置。令人惊讶的是，无需存储元素 R 在 I中的位置。为此，我们对网格 $\mathcal{T}^X$中的元素 R 引入一种全序。回想一下，元素 R 由其左上角像素的位置和宽度确定。因此，我们只需对这些元素的左上角像素排序：

我们定义像素（$i,\ j$）上的字典序 $<$，以 I 为依据

$$(i,j) < (i',j') \quad \Longleftrightarrow \quad (i < i') \text{ or } \big((i = i') \text{ and } (j < j')\big). \tag{13}$$

1	3	6	10	15	21	28	36
2	5	9	14	20	27	35	43
4	8	13	19	26	34	42	49
7	12	18	25	33	41	48	54
11	17	24	32	40	47	53	58
16	23	31	39	46	52	57	61
22	30	38	45	51	56	60	63
29	37	44	50	55	59	62	64

Fi图 5： 矩阵 ent 的枚举可视化 N ries

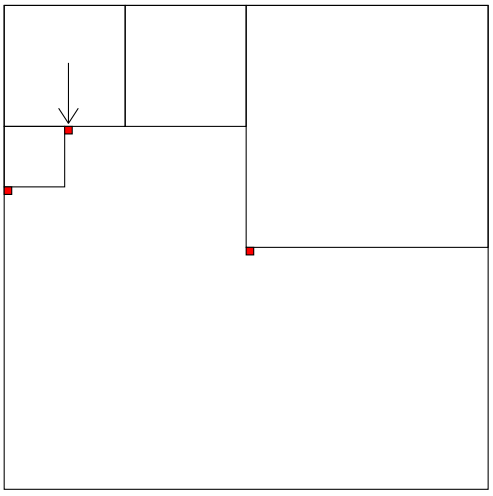

图 6：可视化显示 $<$-最小空像素的候选列表。箭头标示最小值。

随后，我们对元素 $\mathcal{T}^X = \{R_1^X, \ldots R_{n_X}^X\}$ 进行排序，使其左上角像素（j_{R_i}, k_{R_i}）按照 $<$ 升序排列。

考虑向量 S_I，它对应近似图像 I_{app} 及相应网格 $\mathcal{T}^X$，网格由元素 R_i^X、$i = 1$、.....、n_X 组成，其中 $X \in \{Y, C_b, C_r\}$。那么向量 S_I 的形式为

$$\begin{aligned} S_I = \Big(& \frac{h}{w}, w, \operatorname{width}(R_1^Y), S_{R_1^Y}^Y, \operatorname{sep}, \operatorname{width}(R_2^Y), S_{R_2^Y}^Y, \operatorname{sep}, \ldots, \operatorname{width}(R_{n_Y}^Y), S_{R_{n_Y}^Y}^Y, \\ & \operatorname{width}(R_1^{C_b}), S_{R_1^{C_b}}^{C_b}, \operatorname{sep}, \ldots, \operatorname{width}(R_{n_{C_b}}^{C_b}), S_{R_{n_{C_b}}^{C_b}}^{C_b}, \\ & \operatorname{width}(R_1^{C_r}), S_{R_1^{C_r}}^{C_r}, \operatorname{sep}, \ldots, \operatorname{width}(R_{n_{C_r}}^{C_r}), S_{R_{n_{C_r}}^{C_r}}^{C_r} \Big). \end{aligned}$$

在存储数据之前，对 S_I 应用进一步的无损游程压缩步骤 [Hop21]。这与标准 JPEG 压缩类似，可去除 S_R^X中通常会出现的大量零。

4.2 压缩图像的重建

要重建 I_{app}，我们首先对 S_I进行无损解压缩。从 S_I 中移除并存储 h/w和 w。然后，分别重建每个颜色分量 $X \in \{Y, C_b, C_r\}$。重建过程如下：

算法 6. 1. 初始化一个大小为 $h \times w$ 的空 I_{app}^X。
2. 在 I_{app}^X 中查找 $<$最小的空像素 (i, j)，如果不存在空像素，则移至下一个颜色分量。3. 从 S_I 中移除第一个可用块 (宽度(R^X) $S_{R^X}^X$)。4. 计算 $w_{R^X} =$宽度(R^X)和 $h_{R^X} = w_{R^X} \frac{h}{w}$。5. 设置

$$I_{\text{app}}^X|_{\{i,\ldots,i+h_{R^X}-1\}\times\{j,\ldots,j+w_{R^X}-1\}} = \operatorname{IDCT}_R(\operatorname{TL}_{R^X}^{-1}(Q \odot F^X(R^X))).$$

并转到步骤 *2*。

注释 6。 如果以朴素方式查找算法 6 步骤 2 中的最小像素，代价相当高。然而，我们注意到，该最小值始终位于先前重建的元素 R 的角像素旁边（更准确地说，是紧邻右上角和左下角像素，见图 6）。维护这些候选像素的列表可以显著减少搜索工作量。

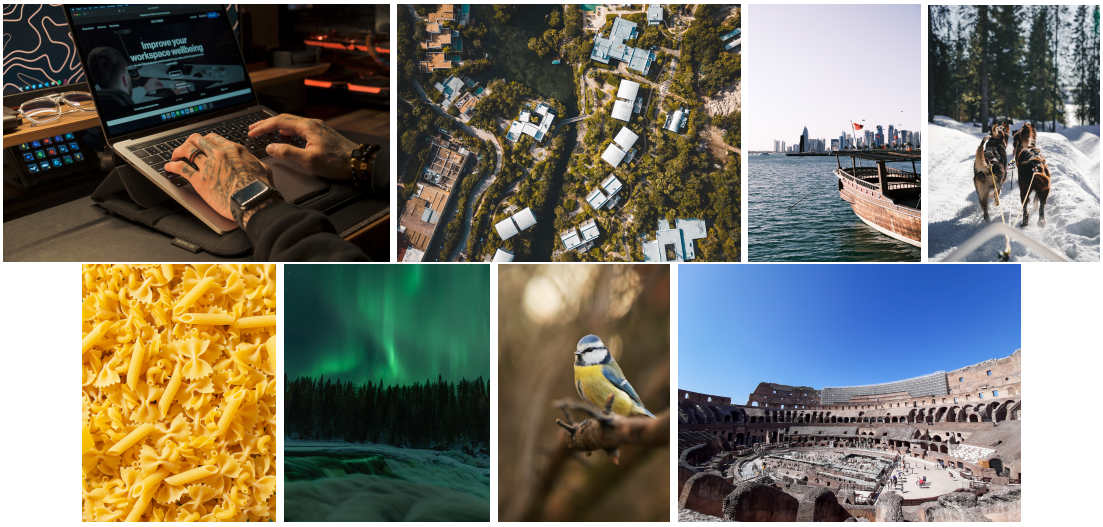

图 7： 从左到右、从上到下：来自 unsplash.com 的测试图像（摄影师及尺寸）：(1) Kadyn Pierce（4000x6000），(2) WillianJusten de Vasconcellos（3024x4032），(3) Emre Raizyq（6240x4160），(4) Kevin Charit（5780x3853），(5) Vishwa-jeet Nishad（2560x1920），(6) Christoph Nolte（2400x1920），(7) Lukasz Rawa（960x640），(8) Mark Stuckey（1440x1920）（从左到右、从上到下）。

5 自适应压缩算法的实际性能

我们使用算法 2 进行少量实验，以测试其在图 7 所示真实图像上的性能。如图 9–10 所示，我们选择算法 2 中的容差 τ，使原始图像与压缩图像之间不存在能够立即察觉的质量差异。图 8 比较了原始 JPEG 文件与使用算法 2 进行自适应重新压缩后的存储大小。

我们发现，自适应压缩可显著减少较大 JPEG 文件的存储空间，而对较小文件的改进则不明显。由于算法 2 不会生成小于 8×8的元素，因此可获得的最精细网格与标准 JPEG 算法相同。因此，从启发式角度来看，任意给定图像所需的存储空间至多与 JPEG 大致相同，不过要严格证明最坏情况下的结论并非易事。

6 结论

我们通过实际实验和数学最优性结果，证明了自适应 JPEG 压缩的可行性。从效率的角度看，采用更精细的均匀网格 $\mathcal{T}_0$作为起点是合理的。算法2和算法 6都可以轻松地针对初始网格 $R \in \mathcal{T}_0$中的元素进行并行化。

该方法还可以通过其他几种方式扩展：一种初步设想是将算法应用于视频，而不是图像。为此，我们需要在由三维长方体单元 R 组成的网格上运行算法 2（每个长方体都可以拆分为八个子长方体），并用离散三维余弦变换（或单独处理时间坐标的变体）来近似 $I|_R$。

该方法的另一种扩展是应用 Binev 的 hp近似算法 [Bin18]。与（有限元方法中采用的）提高多项式次数不同， p细化可以在每个元素 R上局部修改量化矩阵，也可以改变为每个 R存储的频率数量。

最后，如果能用一种更精细的方法来替代局部误差中的范数$\|\cdot\|$，以评判图像的主观质量损失，将会很有意思。

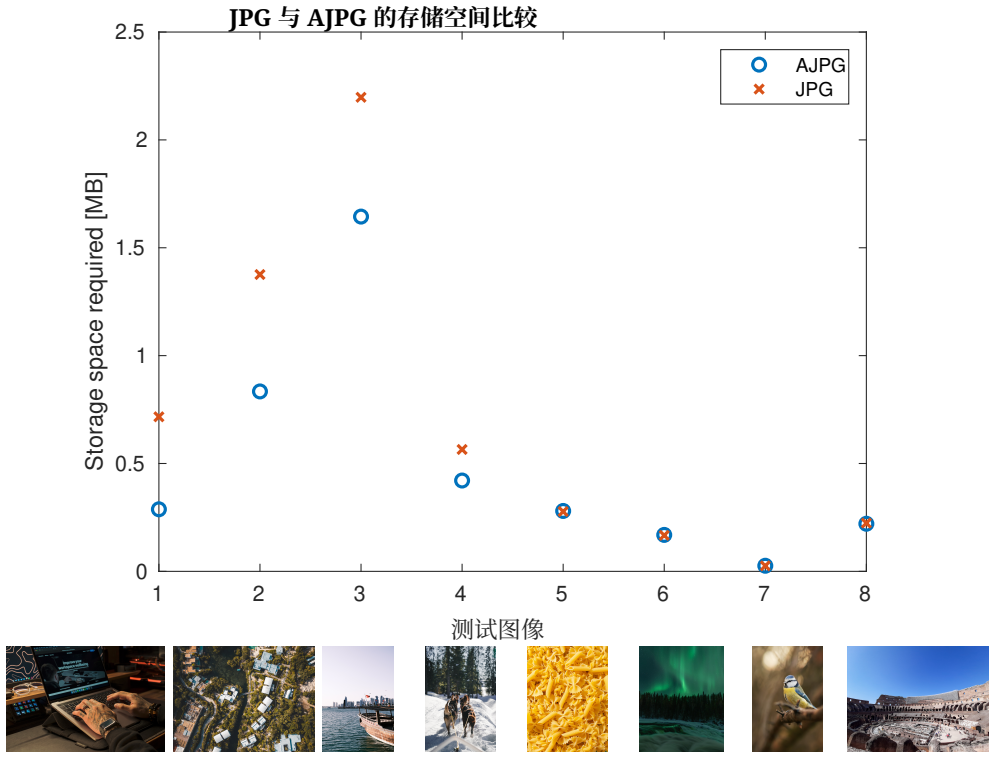

图 8： 原始 JPEG 文件（JPG）及其使用算法 2 进行自适应重新压缩后的文件（AJPG）所需的存储空间。

资助： 本研究由德国研究联合会（DFG）资助，项目编号 258734477（SFB 1173）；同时也由奥地利科学基金（FWF）通过特别研究计划“驯服 PDE 系统的复杂性”（资助项目 SFB F65）资助。

参考文献

[AO00] M. Ainsworth 和 J. T. Oden。有限元分析中的后验误差估计。《纯粹与应用数学》（纽约）。Wiley-Interscience，纽约，2000 年。[ATM09] A. Annamalai、C. Tellambura 和 J. Matyjas。广义 Marcum q 函数 qm(a, b) 的一种新变体：具有分数阶 m 及其应用。载于*2009 年第 6 届 IEEE 消费者通信与网络会议*，第 1–5 页，2009 年。[BD04] Peter Binev 和 Ronald DeVore。自适应树逼近中的快速计算。*Numer. Math.*，97(2)：193–217，2004 年。[BDD04] P. Binev、W. Dahmen 和 R. DeVore。具有收敛率的自适应有限元方法。*Numer. Math.*，97(2)：219–268，2004 年。[Bin18] Peter Binev。用于 hp 自适应的树逼近。*SIAM J. Numer.Anal.*，56(6)：3346–3357，2018 年。[BM87] I. Babuska 和 A. Miller。一种带后验误差估计的反馈有限元方法：第一部分，有限元方法及后验误差估计器的一些基本性质。*Comput. Methods Appl. Mech.Engrg.*，61(1)：1–40，1987 年。[BV84] I. Babuska 和 M. Vogelius。一维边值问题的反馈与自适应有限元求解。*Numer. Math.*，44(1)：75–102，1984 年。[CFPP14] C. Carstensen、M. Feischl、M. Page 和 D. Praetorius。自适应公理。*Comput. Math.Appl.*，67(6)：1195–1253，2014 年。[CKNS08] J. M. Cascon、C. Kreuzer、R. H. Nochetto 和 K. G. Siebert。自适应有限元方法的准最优收敛率。*SIAMJ. Numer.Anal.*，46(5)：2524–2550，2008 年。

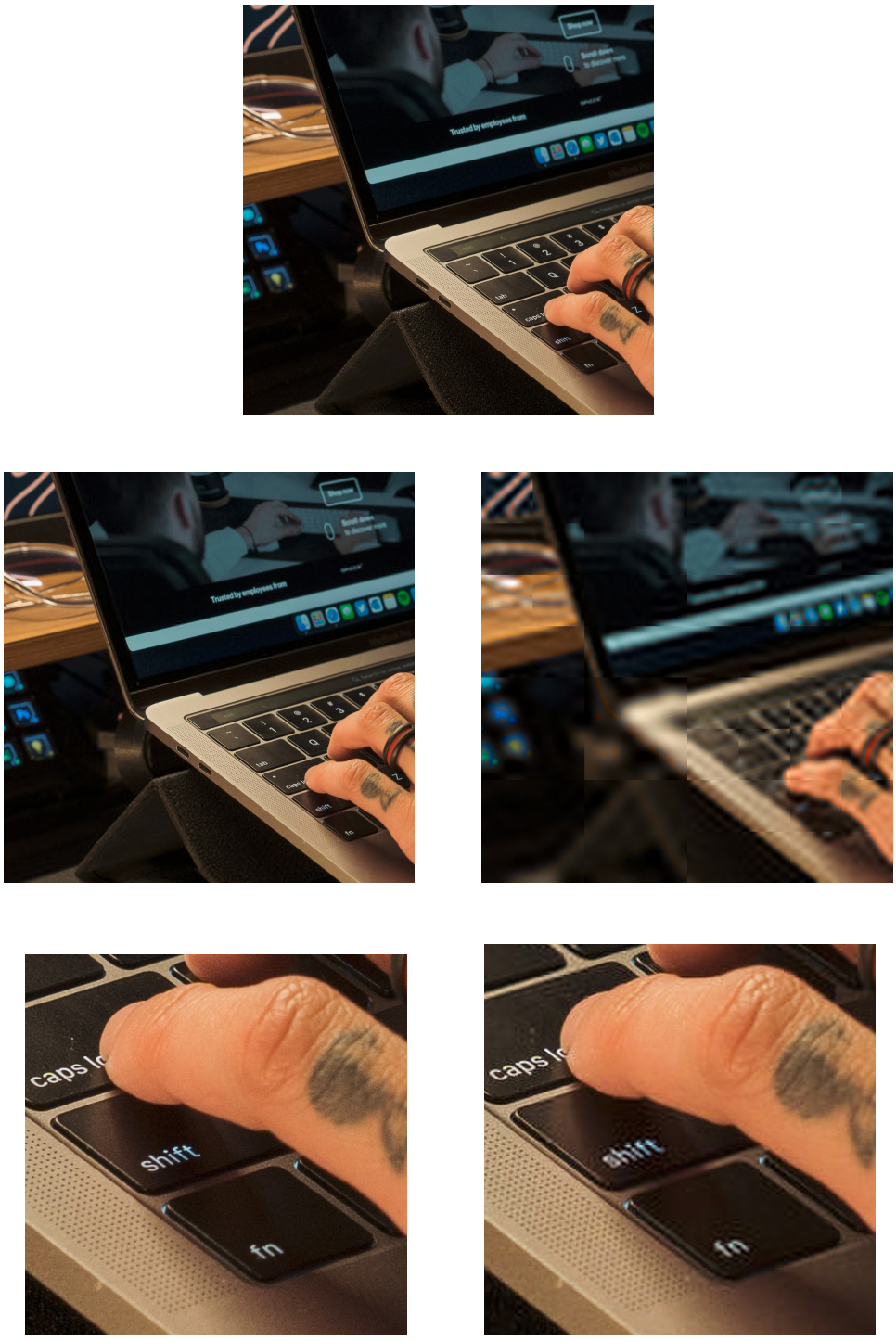

图9： 原始图像（顶部）与自适应算法针对小型和大型τ生成的两个输出（中部）的比较。图8中使用的原始图像特写（左下）和重新压缩图像特写（右下）。图像来自 unsplash.com，艺术家为 Kadyn Pierce。



图 10： 原始图像（上）与自适应算法在小型和大型τ条件下的两种输出（中）的比较。原始图像特写（左下）和重新压缩图像（右下）用于图 8。图像来自 unsplash.com，摄影师为 Willian Justen de Vasconcellos。



图 11： 原始图像（上）与针对 L^2 范数的自适应算法输出（中左）和 BV 范数的自适应算法输出（中右）的比较。底部一行比较了 L^2 输出（左）与 BV 输出（右）的局部细节，结果表明 BV 范数可能更为适用。选择相应的阈值参数，使两种输出占用的存储空间几乎相同。图像来自 unsplash.com，摄影师为 Kevin Charit。

[Dör96] Willy Dörfler。泊松方程的收敛自适应算法。SIAM 数值分析杂志，33(3)：1106–1124，1996。[EAS14] Ferda Ernawan、Nur Azman Abu 和 Nanna Suryana。使用心理视觉模型的自适应 JPEG 图像压缩。《先进科学快报》，20：26–31，2014。[Esp68] R. Esposito。关于 Marcum Q 函数偏微分的评论。第 56 卷，第 2195–2195 页，1968。[Fei22] Michael Feischl。inf-sup 稳定性蕴含拟正交性。《数学计算》，91(337)：2059–2094，2022。[HLN+18] Graham Hudson、Alain Léger、Birger Niss、István Sebestyén 和 Jørgen Vaaben。JPEG-1 标准 25 年：成功的过去、现在和未来原因。《电子成像杂志》，27(4)：040901，2018。[Hop21] J. Hopkins。压缩例程，MATLAB Central 文件交换平台，访问日期：2022 年 5 月 31 日，2021。[JH99] Karl-Heinz Klösener Joachim Hartung，Bärbel Elpelt。统计学。Oldenbourg，1999。[MNS00] P. Morin、R. H. Nochetto 和 K. G. Siebert。数据振荡与自适应有限元法的收敛性。SIAM 数值分析杂志，38(2)：466–488，2000。[Rep08] S. Repin。偏微分方程的后验估计，Radon 计算与应用数学丛书第 4 卷。Walter de Gruyter GmbH & Co. KG，柏林，2008。[RW96] R. Rosenholtz 和 A.B. Watson。感知自适应 JPEG 编码。载于《第三届 IEEE 国际图像处理会议论文集》，第 1 卷，第 901–904 页，第 1 卷，1996。[Sta94] ISO 标准。ISO/IEC 10918-1:1994 信息技术——连续色调静态图像的数字压缩与编码：要求与指南，1994。[Ste07] R. Stevenson。标准自适应有限元法的最优性。《计算数学基础》，7(2)：245–269，2007。[TTQ09] U Kin Tak、Zesheng Tang 和 Dongxu Qi。数字图像的非均匀矩形分区编码及其应用。载于《2009年国际信息与自动化会议》，第 995–999 页，2009。[UJQT12] KinTak U.、Nian Ji、Dongxu Qi 和 Zesheng Tang。基于非均匀矩形分区的 JPEG 自适应量化技术。载于 Ying Zhang 编，《未来无线网络与信息系统》，第 179–187 页，柏林、海德堡，2012。Springer 柏林·海德堡。[Ver13] R. Verfürth。有限元方法的后验误差估计技术。《数值数学与科学计算》。牛津大学出版社，牛津，2013。[WB06] Zhou Wang 和 Alan Conrad Bovik。现代图像质量评估。载于《现代图像质量评估》，2006。