基于预测梯度的图像插值算法

陆志芳 [1] 钟宝江 [1]

摘 要 提出一种新的非线性图像插值算法, 称为基于预测梯度的图像插值 (Image interpolation with predicted gradients, PGI). 首先沿用现有的边缘对比度引导的图像插值 (Contrast-guided image interpolation, CGI) 算法思想对低分辨率图像中的边缘进行扩散处理, 然后预测高分辨率图像中未知像素的性质, 最后对边缘像素采用一维有方向的插值, 对非边缘像素采用二维无方向的插值. 与通常的非线性图像插值算法相比, 新算法对图像边缘信息的理解更为完善. 与 CGI 算法相比, 由于梯度预测策略的使用, PGI 算法能够更有效地确定未知像素的相关性质 (是否为边缘像素, 以及是边缘像素时其边缘方向). 实验结果表明, PGI 算法无论在视觉效果还是客观性测评指标方面均优于现有的图像插值算法. 此外, 在对彩色图像进行插值时, 本文将通常的 RGB 颜色空间转化为 Lab 颜色空间, 不仅减少了伪彩色的生成, 而且降低了算法的时间复杂度.

关键词 图像插值, 预测梯度, 对比度, 梯度, 边缘

引用格式 陆志芳, 钟宝江. 基于预测梯度的图像插值算法. 自动化学报, 2018, **44**(6): 1072−1085

DOI 10.16383/j.aas.2017.c160793

Image Interpolation With Predicted Gradients

LU Zhi-Fang[1] ZHONG Bao-Jiang[1]

Abstract A new nonlinear image interpolation algorithm is proposed, referred to as image interpolation with predicted gradients (PGI). First, the idea of contrast-guided image interpolation (CGI) is employed to diffuse the edges in the low-resolution (LR) image. Then, unknown pixels in the high-resolution (HR) image are predicted. Finally, a 1-D directional filter is employed to process edge pixels while a 2-D directionless filter is used to interpolate non-edge pixels. Compared to the common nonlinear image interpolation algorithms, the new algorithm has a better interpretation of image edges. Compared to the CGI, the PGI can predict the property of unknown pixels more precisely (including whether an unknown pixel is an edge pixel or not, and its direction if it is). Experimental results show that PGI has a better performance than the existing algorithms, either with respect to visual effect or in terms of objective criteria. In addition to interpolate color images, the usual RGB space needs to be converted to the Lab space. As a result, pseudo-color can be suppressed and the computational complexity is reduced.

Key words Image interpolation, predicted gradients, contrast, gradients, edge

Citation Lu Zhi-Fang, Zhong Bao-Jiang. Image interpolation with predicted gradients. *Acta Automatica Sinica*, 2018, **44**(6): 1072−1085

图像插值广泛应用于图像处理的各个领域, 例如人脸检测、军事雷达图像处理、医学图像分析以及超高清电视应用等[1−2], 其目的是将一幅图像从它的低分辨率 (Low resolution, LR) 版本变成相应的高分辨率 (High resolution, HR) 版本. 例如, 在超高清电视的应用中, 大部分的电视节目还是标清信号, 将这些低分辨率的图像信号转为超高分辨率的图像信号成为各电视生产企业正在采取的一种手段. 这样的信号转换如何实现, 就是图像插值研究工作者需要致力解决的问题.

在过去的几十年中, 人们提出了很多图像插值算法. 这些算法从技术思想方面可概略地分为两类: 一类是基于局部数据 (Local-data-based) 的插值方法[3−18], 另一类是基于样例 (Example-based) 的插值方法[19−22]. 两者的主要区别在于, 基于局部数据的插值方法根据待插值图像的当前像素, 利用数学方法直接进行插值; 而基于样例的插值方法则首先通过训练和学习建立起低分辨率图像和高分辨率图像之间的一种映射关系, 然后利用图像块匹配和替换来完成插值. 这两类插值算法各具特色, 分别有不同的优势和局限性.

本文主要研究和考虑基于局部数据的插值方法 (除特别说明外, 以下章节中提到的插值算法均为这类算法). 与基于样例的插值方法相比, 这类方法一般具有较低的时间和空间复杂度. 从建模特征方面, 这类方法又可细分为线性插值方法和非线性插

收稿日期 2016-12-13 录用日期 2017-05-11
Manuscript received December 13, 2016; accepted May 11, 2017
国家自然科学基金 (61572341), 苏州大学 "东吴学者计划" 资助
Supported by National Natural Science Foundation of China (61572341) and Soochow Scholars Program of Soochow University
本文责任编委 桑农
Recommended by Associate Editor SANG Nong
1. 苏州大学计算机科学与技术学院 苏州 215006
1. College of Computer Science and Technology, Soochow University, Suzhou 215006

值方法. 两者的不同点在于, 线性方法在插值过程中采用同一个插值内核, 这种做法会使得图像中的边缘变得模糊不清, 达不到高清图像的视觉效果; 非线性方法则会对图像不同部分采用不同的插值策略, 其主要的出发点就是为了更好地处理图像在边缘处的插值效果.

本文提出一种非线性插值算法, 称为 “基于预测梯度的图像插值 (Image interpolation with predicted gradients, PGI) 算法”. 与现有的非线性插值方法相比, PGI 算法能够更有效地区分图像的边缘像素和非边缘像素. 具体来说, 在通过梯度信息判断某一像素是否是边缘像素时, 我们提出了一种梯度预测策略, 提高了像素的区分精度, 最终提升了图像的插值效果. 此外, 在对彩色图像进行插值时, 我们将通常的 RGB 颜色空间转换到更符合人眼感知特性的 Lab 颜色空间. 在 Lab 颜色空间上执行新算法, 不仅减少了图像的伪彩色, 而且降低了图像的插值时间.

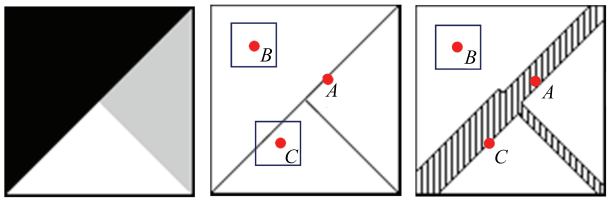
1　现有的图像插值算法

现有的基于局部数据的图像插值方法中, 线性插值主要包括: 最近邻插值法、双线性插值法、双三次插值法[3] 以及其他一些改进算法[4−6]. 非线性插值方法主要包括: 基于协方差矩阵的方法[7−8], 基于小波系数的方法[9−10] 以及基于边缘信息的方法[11−15]. 虽然这些非线性图像插值方法在阐述各自的技术思想时采用了不同的方式, 但其主要针对的任务是一致的, 即如何有效地解决图像中边缘处像素的插值问题. 实际上, 对于非边缘处像素的插值即便是传统的线性插值方法已经能够取得很好的效果, 而正是出于对边缘像素插值效果的考虑, 才推动了当前非线性插值方法的持续发展.

在通常的非线性图像插值方法中, 对 “边缘” 的理解和认识与现有的边缘检测方法 (例如 Canny 算法) 是一致的, 即将梯度超过给定阈值且通过非极大值抑制的像素点连接而成的线条称为边缘, 其宽度一般只有一到两个像素. 在插值过程中, 这类边缘像素将得到 “保护”, 例如采用与边缘方向一致的一维方向插值[23]; 而对于非边缘像素, 一般会采用与线性插值法相同的策略 (例如双三次插值) 处理. 此时, 对于边缘附近的非边缘像素, 其插值所涉及到的支持领域将跨越边缘, 导致插值得到的图像边缘仍然会存在一定程度的模糊化.

对此, 2013 年 Wei 和 Ma[14] 在其研究工作中给出了对图像边缘一种新的理解与阐述: 在非线性插值方法中, 仅对传统意义上一到两个像素宽的边缘进行 “保护” 是不够的, 为了保证插值效果, 需要同时考虑到距离边缘一定范围内的非边缘像素并对它们进行类似的保护 (例如采用边缘导向的一维方向插值, 而不是采用无方向的二维插值), 这一做法可称为 “边缘扩散” 或 “边缘带状化”. 特别地, 边缘扩散的程度与边缘对比度 (即边缘两侧像素的灰度值差异) 成正比, 即边缘对比度越大, 扩散后的带状化边缘越宽. 基于以上考虑, Wei 和 Ma 提出了一种边缘对比度引导的图像插值 (Contrast-guided image interpolation, CGI) 算法[14].

图 1 示例了以上两种不同图像边缘阐述方法的差异. 图 1 (a) 给出了一幅结构比较简单的图像, 图 1 (b) 呈现的是通常的非线性图像插值方法所考虑的边缘. 这类方法能够保证边缘上的像素 (例如点 A) 清晰地出现在 HR 图像中. 对于点 B 和点 C, 则会被当作非边缘像素进行二维无方向插值. 此时, 对于点 B 来说, 由于无方向插值涉及的支持领域完全位于边缘的同一边 (像素差异值较小), 插值效果能够得到保证. 而对于点 C 来说, 由于插值的支持领域跨越边缘, 便有可能产生严重的误差. 图 1 (c) 显示的是 CGI 算法所考虑的边缘, 根据其边缘扩散原则, 对于图 1 (a) 的图像, 黑色与白色的对比度明显大于灰色与白色的对比度, 因而扩散后的带状化边缘宽度也不一样. 在图 1 (c) 中, 由于点 C 被扩散后的边缘涵盖, 因此在插值过程中将得到保护, 进而产生与通常的非线性插值方法不同的插值效果.



(a) 测试图像
(a) The test image

(b) 通常非线性插值算法考虑的边缘
(b) Edges interpreted by usual nonlinear interpolation methods

(c) CGI 算法考虑的边缘
(c) Edges interpreted by CGI

图 1　对图像边缘的两种不同阐述方式

Fig. 1　Two types of interpretations of image edges

在 CGI 算法中, 边缘扩散是通过一种隐式方式即像素的梯度扩散来实现的[14].

考虑如下能量函数:

$$E(u_\theta) = E_d(u_\theta) + \lambda E_s(u_\theta) \tag{1}$$

其中, λ 是一个正则参数, 默认值为 0.2. u_θ 是 θ 方向上扩散之后的梯度值, $E_d(u_\theta)$ 是数据保真项, 目的是使扩散后的边缘保持原有的特性, 形式为

$$E_d(u_\theta) = \iint \left[U_\theta^2(u_\theta - U_\theta)^2\right] \mathrm{d}x\mathrm{d}y \tag{2}$$

其中, U_θ 是 θ 方向上的梯度值, $E_s(u_\theta)$ 是平滑项,

用来促进边缘的扩散, 抑制非边缘的扩散, 表示为

$$E_s(u_\theta)=\iint\left[(u_\theta)_x^2+(u_\theta)_y^2\right]\mathrm{d}x\mathrm{d}y \tag{3}$$

根据式 (1)∼(3), 可得

$$E(u_\theta)=\iint\left\{U_\theta^2(u_\theta-U_\theta)^2+\lambda\left[(u_\theta)_x^2+(u_\theta)_y^2\right]\right\}\mathrm{d}x\mathrm{d}y \tag{4}$$

为方便起见, 将式 (4) 的积分函数转化为

$$F[x,y,u_\theta,(u_\theta)_x,(u_\theta)_y]=U_\theta^2(u_\theta-U_\theta)^2+\lambda[(u_\theta)_x^2+(u_\theta)_y^2] \tag{5}$$

此时, 可将 $E(u_\theta)$ 求解最小值的过程转化为

$$\frac{\partial F}{\partial u_\theta}-\frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial F}{\partial(u_\theta)_x}-\frac{\mathrm{d}}{\mathrm{d}y}\frac{\partial F}{\partial(u_\theta)_y}=0 \tag{6}$$

即

$$U_\theta^2(u_\theta-U_\theta)-\lambda\nabla^2u_\theta=0 \tag{7}$$

其中, ∇^2 是拉普拉斯算子, ∇^2u_θ 表示为

$$\nabla^2u_\theta=(u_\theta)_{xx}+(u_\theta)_{yy} \tag{8}$$

直接求解方程 (7) 比较困难, 经过分析可得, 待求解的方程中, 存在两个连续的空间变量 x 和 y, 因此, 可以通过迭代的方式求解, 即引入一个连续的时间变量参数 t, 根据

$$\frac{\partial u_\theta(x,y,t)}{\partial t}=U_\theta^2[u_\theta(x,y,t)-U_\theta]-\lambda\nabla^2u_\theta(x,y,t) \tag{9}$$

$$\frac{\partial u_\theta(x,y,t)}{\partial t}\approx u_\theta^{n+1}(i,j)-u_\theta^n(i,j) \tag{10}$$

$$\nabla^2u_\theta(x,y,t)\approx u_\theta^n(i+1,j)+u_\theta^n(i,j+1)+u_\theta^n(i-1,j)+u_\theta^n(i,j-1)-4u_\theta^n(i,j) \tag{11}$$

可得

$$u_\theta^{n+1}(i,j)=u_\theta^n(i,j)+U_\theta^2(i,j)[u_\theta^n(i,j)-U_\theta(i,j)]-\lambda[u_\theta^n(i+1,j)+u_\theta^n(i,j+1)+u_\theta^n(i-1,j)+u_\theta^n(i,j-1)-4u_\theta^n(i,j)] \tag{12}$$

其中, n 代表迭代次数, 一般取不超过 10 的正整数.

由于边缘扩散思想的引入, CGI 算法提升了图像边缘的插值视觉效果[14]. 鉴于 CGI 算法的优越性, 本文将沿用其思想来理解图像边缘, 即在插值过程中考虑扩散后的边缘而非传统意义上的 Canny 边缘.

2 基于预测梯度的图像插值

2.1 基本思想

将待插值的 LR 图像记为 $I^{(l)}(i,j)$, $1\le i\le M$, $1\le j\le N$, 即原始图像的大小是 $M\times N$. 将插值得到的 HR 图像记为 $I^{(h)}(i,j)$, 其大小为 $m\times n$. 本文主要讨论 2 倍插值的情况, 即 $m=2M$, $n=2N$, 其他倍数插值可基于已有的 2 倍插值的结果进行拓展[8].

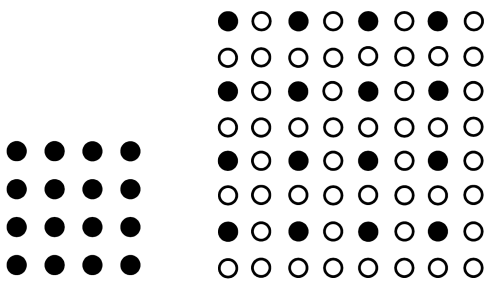
在插值开始时, 我们首先检测待插值的 LR 图像的边缘, 并对其进行扩散处理. 基于扩散后的边缘, 可以判断出 LR 图像中每个像素的性质. 这里, 像素的性质包括两个方面:

1) 是边缘像素还是非边缘像素;

2) 若是边缘像素, 其所在边缘的方向.

为了得到 HR 图像, 我们首先会将 LR 图像中 (i,j) 位置上的像素 (连同其性质) 直接复制到 HR 图像中 $(2i-1,2j-1)$ 位置上, 如图 2 所示. 具体来说

$$I^{(h)}(2i-1,2j-1)=I^{(l)}(i,j) \tag{13}$$



(a) LR 图像 (a) The LR image　　(b) HR 图像 (b) The HR image

图 2 插值示意图, 其中实心点表示已知的像素 (从 LR 图像中直接复制得到), 空心点表示未知的像素

Fig. 2 Illustration of the interpolation process (the dots denote the known pixels and the circles denote the missing pixels)

为了插值出 HR 图像中剩余位置的未知像素, 需要首先根据已知像素判断其性质, 再确定使用一维方向插值 (当该像素为边缘像素时) 还是二维无方向插值 (当该像素为非边缘像素时).

在 CGI 算法中, 未知像素性质的判断策略是非常简单的, 形式为

$$\begin{cases}u_\theta^{(h)}(2i-1,2j)=u_\theta^{(h)}(2i-1,2j-1)\\ u_\theta^{(h)}(2i,2j-1)=u_\theta^{(h)}(2i-1,2j-1)\\ u_\theta^{(h)}(2i,2j)=u_\theta^{(h)}(2i-1,2j-1)\end{cases} \tag{14}$$

从而, $(2i-1,2j)$, $(2i,2j-1)$ 以及 $(2i,2j)$ 位置上的未知像素的性质与 $(2i-1,2j-1)$ 位置上的已知像素的性质是完全相同的. 换句话说, CGI 算法采用了最近邻策略来判定 HR 图像中未知像素的性质.

显然, CGI 算法对于未知像素性质的判断是很粗糙的. 当图像的边缘信息比较丰富时 (例如多条边缘相互交叉), 会出现像素的属性 (是否为边缘像素) 或边缘的方向混乱不清, 从而影响最终的插值效果. 为了有效地判断出 HR 图像中未知像素的性质, 我们提出了梯度预测策略.

2.2　梯度预测

考虑一幅 LR 图像 $I^{(l)}(i,j)$, 像素点 (i,j) 在方向 θ 上的梯度值记为

$$U_\theta^{(l)}(i,j)=|\nabla_\theta^{(l)}I^{(l)}(i,j)| \tag{15}$$

其中, $\nabla_\theta^{(l)}$ 是方向导数. 在离散情形下, 可以用卷积来计算方向导数, 形如

$$\nabla_\theta^{(l)}I^{l}(i,j)=S_\theta * I^{(l)}(i,j) \tag{16}$$

其中, $*$ 表示卷积操作, S_θ 表示边缘检测算子所对应的掩模. 在连续情形下, 边缘的方向是任意的, 即 θ 的取值在 $(0°,180°]$ 之间. 然而对于离散的数字图像而言, 边缘方向只需考虑 4 种情况, 即 $\theta=0°$, $\theta=45°$, $\theta=90°$ 以及 $\theta=135°$, 如图 3 所示.

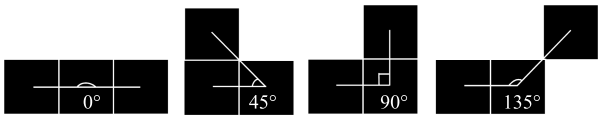


图 3　数字图像的边缘方向示意图

Fig. 3　Illustration of edge directions in digital image

考虑到边缘检测精度和程序运行效率两方面的因素, 可采用 4 个 3×3 的卷积掩模来计算图像在 $0°$, $45°$, $90°$ 以及 $135°$ 这 4 个方向上的梯度近似值[14]. 形式为

$$U_0^{(l)}(i,j)=\left|\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}*I^{(l)}(i,j)\right| \tag{17}$$

$$U_{45}^{(l)}(i,j)=\left|\begin{bmatrix}0&0&1\\-1&0&1\\-1&-1&0\end{bmatrix}*I^{(l)}(i,j)\right| \tag{18}$$

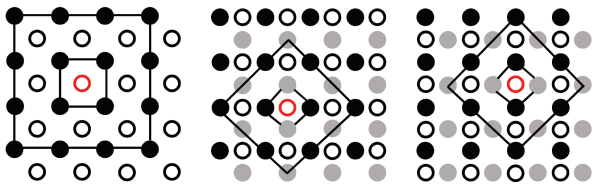
$$U_{90}^{(l)}(i,j)=\left|\begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix}*I^{(l)}(i,j)\right| \tag{19}$$

$$U_{135}^{(l)}(i,j)=\left|\begin{bmatrix}1&1&0\\1&0&-1\\0&-1&-1\end{bmatrix}*I^{(l)}(i,j)\right| \tag{20}$$

为了得到扩散的梯度, 根据前文阐述的扩散思想将计算得到的 $U_\theta^{(l)}(i,j)$ 进行扩散处理, 得到梯度的扩散值 $u_\theta^{(l)}(i,j)$. 为得到 HR 图像上每个像素的扩散梯度 $u_\theta^{(h)}(i,j)$, 首先将 LR 图像中 (i,j) 位置上的梯度直接复制到 HR 图像中 $(2i-1,2j-1)$ 位置上, 形式为

$$u_\theta^{(h)}(2i-1,2j-1)=u_\theta^{(l)}(i,j) \tag{21}$$

对于剩余位置上的未知像素, 我们考虑如下的梯度预测策略: 以该像素为中心, 在其周边区域选择最相邻的 16 个已知梯度的像素, 通过双三次插值来预测当前未知像素的梯度. 这里, "最相邻" 准则是为了尽可能提高梯度的预测精度. 而在插值过程中, 新 "已知梯度" 的像素会按顺序不断生成. 为此, 遴选已知梯度像素的过程需要按照一定的次序进行 (先计算对角方向, 再计算水平和竖直方向), 这是由于在计算水平和竖直方向上像素梯度时需要利用对角方向上像素的梯度. 如果第一步预测水平和竖直方向上的梯度, 此时, 16 个像素梯度中有 8 个梯度是对角方向上的梯度, 它们是未知的, 无法实现, 因此必须将对角方向上像素梯度的预测放在第一步. 在两倍插值过程中, 我们首先会将 LR 图像中的 (i,j) 位置上的像素直接复制到 HR 图像中 $(2i-1,2j-1)$ 位置上, 此时处于位置 $(2i,2j)$ 上的像素就是对角像素, 处于位置 $(2i-1,2j)$ 上的像素就是水平像素, 处于位置 $(2i,2j-1)$ 上的像素就是竖直像素. 具体的预测策略如图 4 所示.



(a) 对角方向上像素梯度的预测
(a) Gradients prediction of the diagonal pixels

(b) 水平方向上像素梯度的预测
(b) Gradients prediction of the row pixels

(c) 竖直方向上像素梯度的预测
(c) Gradients prediction of the column pixels

图 4　梯度预测示意图

Fig. 4　Illustration of gradients prediction

1) 若是对角方向上的像素 $I^{(h)}(a,b)$, 则其对应的扩散梯度 $u_\theta^{(h)}(a,b)$ 所选择的已知像素的梯度如图 4 (a) 所示:

$$u_\theta^{(h)}(a-1,b-1),\qquad u_\theta^{(h)}(a-1,b+1)$$

$$
\begin{aligned}
&u_\theta^{(h)}(a+1,b+1), \quad u_\theta^{(h)}(a+1,b-1)\\
&u_\theta^{(h)}(a-3,b-3), \quad u_\theta^{(h)}(a-3,b-1)\\
&u_\theta^{(h)}(a-3,b+1), \quad u_\theta^{(h)}(a-3,b+3)\\
&u_\theta^{(h)}(a-1,b+3), \quad u_\theta^{(h)}(a+1,b+3)\\
&u_\theta^{(h)}(a+3,b+3), \quad u_\theta^{(h)}(a+1,b+1)\\
&u_\theta^{(h)}(a+3,b-1), \quad u_\theta^{(h)}(a+3,b-3)\\
&u_\theta^{(h)}(a+1,b-3), \quad u_\theta^{(h)}(a-1,b-3)
\end{aligned}
$$

2) 若是水平方向上的像素 $I^{(h)}(a-1,b)$, 则其对应的扩散梯度 $u_\theta^{(h)}(a-1,b)$ 所选择的已知像素的梯度如图 4 (b) 所示:

$$
\begin{aligned}
&u_\theta^{(h)}(a-2,b), \quad u_\theta^{(h)}(a-1,b+1)\\
&u_\theta^{(h)}(a,b), \quad u_\theta^{(h)}(a-1,b-1)\\
&u_\theta^{(h)}(a-4,b), \quad u_\theta^{(h)}(a-3,b+1)\\
&u_\theta^{(h)}(a-2,b+2), \quad u_\theta^{(h)}(a-1,b+3)\\
&u_\theta^{(h)}(a,b+2), \quad u_\theta^{(h)}(a+1,b+1)\\
&u_\theta^{(h)}(a+2,b), \quad u_\theta^{(h)}(a+1,b-1)\\
&u_\theta^{(h)}(a,b-2), \quad u_\theta^{(h)}(a-1,b-3)\\
&u_\theta^{(h)}(a-2,b-2), \quad u_\theta^{(h)}(a-3,b-1)
\end{aligned}
$$

3) 若是垂直方向上的像素 $I^{(h)}(a,b-1)$, 则其对应的扩散梯度 $u_\theta^{(h)}(a,b-1)$ 所选择的已知像素的梯度如图 4 (c) 所示:

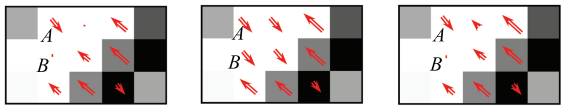
$$
\begin{aligned}
&u_\theta^{(h)}(a-1,b-1), \quad u_\theta^{(h)}(a,b)\\
&u_\theta^{(h)}(a+1,b-1), \quad u_\theta^{(h)}(a,b-2)\\
&u_\theta^{(h)}(a-3,b-1), \quad u_\theta^{(h)}(a-2,b)\\
&u_\theta^{(h)}(a-1,b+1), \quad u_\theta^{(h)}(a,b+2)\\
&u_\theta^{(h)}(a+1,b+1), \quad u_\theta^{(h)}(a+2,b)\\
&u_\theta^{(h)}(a+3,b-1), \quad u_\theta^{(h)}(a+2,b-2)\\
&u_\theta^{(h)}(a+1,b-3), \quad u_\theta^{(h)}(a,b-4)\\
&u_\theta^{(h)}(a-1,b-3), \quad u_\theta^{(h)}(a-2,b-2)
\end{aligned}
$$

其中, $a=2i$, $b=2j$.

根据上述预测策略, 我们可以得到扩散处理后的梯度, 从而能够确定 HR 图像上每个像素的性质.

与 CGI 算法中使用的最近邻梯度预测方式相比, 以上所考虑的双三次梯度预测方式能够获得更高的数值精度. 具体地说, 最近邻插值预测的相对误差为 $\mathrm{O}(h)$, 而双三次插值预测的相对误差为 $\mathrm{O}(h^3)$, 其中 h 为插值的采样步长参数[3].

图 5 给出了新的梯度预测方式与 CGI 算法中梯度预测方式对同一幅测试图像梯度预测结果的细节比较. 图 5 (a) 是测试图像各像素的真实梯度值和方向, 图 5 (b) 是 CGI 算法计算出来的梯度及方向, 图 5 (c) 是用本文提出的梯度预测方法计算出的梯度和方向. 如图 5 (b) 所示, 对于像素 B, 由于 CGI 算法采用最近邻策略, 使得像素 B 的梯度和方向和像素 A 相同 (方向都是朝右下角, $\theta=135°$). 此时, 会将像素 B 作为 135° 方向上的边缘像素进行插值. 但是, 从图 5 (a) 的原图梯度可以看到, 像素 B 其实是个非边缘像素. 如图 5 (c) 所示, 由于 PGI 算法采用待插像素周围 16 个已知像素的梯度值, 使得其可以判断出像素 B 是个非边缘像素, 从而采用非边缘像素的插值策略. 与 CGI 算法相比, PGI 算法计算出的像素梯度无论是从大小还是方向上都更接近真实值.

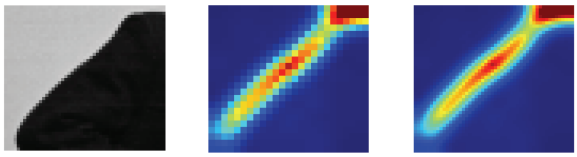


(a) 实际梯度 (a) The ground-truth | (b) CGI 算法的预测结果 (b) Prediction result of CGI | (c) 本文算法的预测结果 (c) Prediction result of ours

图 5 预测结果的细节比较, 箭头方向表示像素梯度方向, 箭头长短表示像素梯度大小

Fig. 5 The detail comparison of prediction results, where the direction and the length of arrow represent the direction and the size of gradient, respectively

图 6 是本文梯度计算方式与 CGI 算法中梯度计算方式对同一幅测试图像梯度预测结果的整体比较. 可以看出, 我们所预测出的梯度能量图比 CGI 算法所得到的更自然和平滑. 在数值结果上, 本文算法的预测结果与真实值的误差 (0.95) 小于 CGI 算法的误差 (1.36).



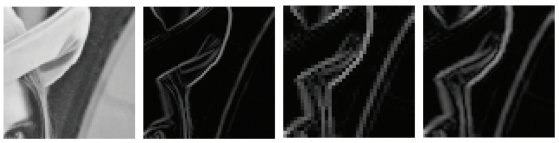
(a) 测试图像 (a) The ground-truth | (b) CGI 算法的预测结果 (误差为1.36) (b) Prediction result of CGI where the error is 1.36 | (c) PGI 算法的预测结果 (误差为0.95) (c) Prediction result of ours where the error is 0.95

图 6 预测结果的整体比较

Fig. 6 The unified comparison of prediction results

图 7 是本文与 CGI 算法中两种梯度计算方式对同一幅测试图像进行梯度预测的可视化结果比较. 图 7 (a) 是测试图像; 图 7 (b) 是测试图像的梯度可视化结果; 图 7 (c) 是利用 CGI 算法从低分辨率 LR

图像上预测高分辨率 HR 图像的梯度可视化结果; 图 7 (d) 是利用本文提出的梯度预测策略从低分辨率 LR 图像上预测高分辨率 HR 图像的梯度可视化结果. 可以看到, CGI 算法预测出的梯度比较粗糙, 与原图梯度相比具有明显的误差. 特别是在边缘信息丰富的局部, 预测误差导致了边缘相互影响, 模糊不清, 给后继的插值带来困难. 本文算法预测的梯度可视化效果图则与原图相差无几.



(a) 测试图像 (a) The test image　(b) 原图梯度 (b) The ground-truth　(c) CGI 预测的梯度 (c) Prediction result of CGI　(d) 本文算法的预测结果 (d) Prediction result of ours

图 7　预测梯度的可视化结果比较

Fig. 7　A visual comparison of predicted gradients

2.3　图像插值

基于预测后的梯度, 可以判断出 HR 图像中每个未知像素的性质 $P(e,\theta)$, 具体地,

1) 此像素是边缘像素 (此时, $e=Y$) 还是非边缘像素 (此时, $e=N$).

2) 若 $e=Y$, 则要确定当前像素所在边缘的方向 (即 $\theta=0°$, $\theta=45°$, $\theta=90°$ 或 $\theta=135°$); 若 $e=N$, 则 $\theta=NA$ (即 Not applicable, 不可用). 这里, 利用梯度判断像素性质的具体做法为:

1) 对于 HR 图像上的每个像素, 我们可以得到其 4 个方向上的梯度, 分别记为 $u_0^{(h)}$, $u_{45}^{(h)}$, $u_{90}^{(h)}$, $u_{135}^{(h)}$.

2) 将这 4 个方向分成两组正交方向进行比较, 其中, $u_0^{(h)}$ 和 $u_{90}^{(h)}$ 是一组, $u_{45}^{(h)}$ 和 $u_{135}^{(h)}$ 是另一组.

3) 对于对角方向上的像素, 若 $|u_{45}^{(h)}-u_{135}^{(h)}|\geq T$, 说明此像素是边缘像素, 即 $e=Y$, 此时

$$\begin{cases}\theta=135°, & 若\ u_{45}^{(h)}\geq u_{135}^{(h)}\\ \theta=45°, & 若\ u_{45}^{(h)}<u_{135}^{(h)}\end{cases}$$

4) 若 $|u_{45}^{(h)}-u_{135}^{(h)}|<T$, 说明此像素是非边缘像素, 即 $e=N$, 此时 $\theta=NA$.

5) 对于水平和垂直方向上的像素, 若 $|u_0^{(h)}-u_{90}^{(h)}|\geq T$, 说明此像素是边缘像素, 即 $e=Y$, 此时

$$\begin{cases}\theta=90°, & 若\ u_0^{(h)}\geq u_{90}^{(h)}\\ \theta=0°, & 若\ u_0^{(h)}<u_{90}^{(h)}\end{cases}$$

6) 若 $|u_0^{(h)}-u_{90}^{(h)}|<T$, 说明此像素是非边缘像素, 即 $e=N$, 此时 $\theta=NA$.

本文算法的结构如图 8 所示. 首先, 对 LR 图像进行边缘像素的检测, 为保护边缘附近的非边缘像素, 需要对边缘进行扩散处理, 得到 $u_\theta^{(l)}$. 然后, 基于 LR 图像的扩散梯度 $u_\theta^{(l)}$ 预测 HR 图像的扩散梯度 $u_\theta^{(h)}$. 接着, 基于得到的扩散梯度 $u_\theta^{(h)}$ 预测 HR 图像上每个像素的性质 $P(e,\theta)$. 最后, 根据预测的像素性质区分 HR 图像上的边缘像素和非边缘像素, 并对它们分别进行处理, 即对边缘像素采用一维方向插值, 对非边缘像素采用二维无方向插值.

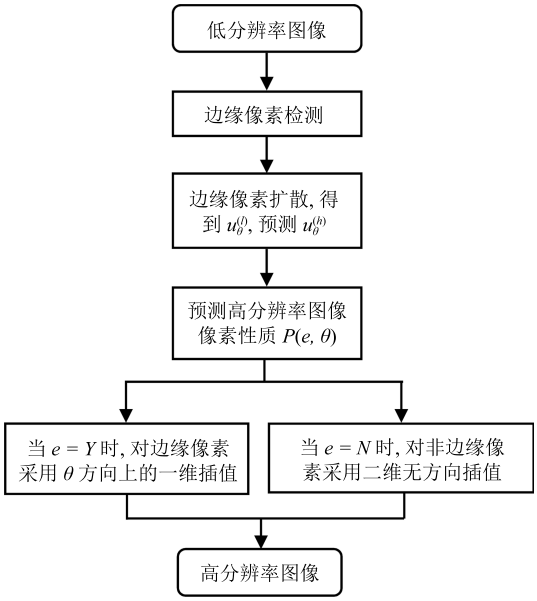


图 8　PGI 算法插值流程图

Fig. 8　The flowchart of PGI

在插值过程中, 由于待插像素所处位置的不同, 也需要按照一定的次序进行 (先计算对角方向, 再计算水平和竖直方向), 这是由于在计算水平和竖直方向上像素灰度值时需要利用对角方向上像素的灰度值. 因而, 第一步先根据 LR 图像上的灰度值 $I^{(l)}(i,j)$ 计算 HR 图像对角方向上像素的灰度值 $I^{(h)}(2i,2j)$; 第二步再根据原始灰度值 $I^{(l)}(i,j)$ 以及第一步得到的 $I^{(h)}(2i,2j)$ 分别计算水平方向和竖直方向上像素的灰度值 $I^{(h)}(2i-1,2j)$ 以及 $I^{(h)}(2i,2j-1)$. 插值过程中, 对于每一个非边缘像素, 利用二维无方向插值进行处理, 本文采用传统的双三次插值算法作为二维无方向插值算法, 对于位置 $(i+u,j+v)$ 处像素的灰度值 (u 和 v 都是不大于 1 的正浮点数), 其计算公式为

$$I(i+u,j+v)=ABC \tag{22}$$

其中,

$$A=\begin{bmatrix}S(u+1) & S(u+0) & S(u-1) & S(u-2)\end{bmatrix}$$

$$B=\begin{bmatrix} I(i-1,j-1) & I(i-1,j+0) & I(i-1,j+1) & I(i-1,j+2)\\ I(i+0,j-1) & I(i+0,j+0) & I(i+0,j+1) & I(i+0,j+2)\\ I(i+1,j-1) & I(i+1,j+0) & I(i+1,j+1) & I(i+1,j+2)\\ I(i+2,j-1) & I(i+2,j+0) & I(i+2,j+1) & I(i+2,j+2) \end{bmatrix}$$

$$C=\begin{bmatrix} S(v+1) & S(v+0) & S(v-1) & S(v-2) \end{bmatrix}^{\mathrm{T}}$$

$$S(x)=\begin{cases} 1-2|x|^2+|x|^3, & 0\le|x|<1\\ 4-8|x|+5|x|^2-|x|^3, & 1\le|x|<2\\ 0, & |x|\ge 2 \end{cases}$$

利用式 (22) 就可以对图像上的每一个非边缘像素进行插值处理. 对于边缘像素, 利用

$$I=\omega(I_a+I_b)+(0.5-\omega)(I_c+I_d) \tag{23}$$

进行处理. 这里, ω 是可调参数, 基于仿真实验结果, 文中的值为 0.575, 此值是根据 151 幅灰度图像进行训练得到的最佳结果; I 表示待求像素的灰度值; I_a, I_b, I_c 以及 I_d 表示已知像素的灰度值, 其确定规则为:

1) 对角像素插值

a) 若 $u_{45}^{(h)}-u_{135}^{(h)}>T$, 则 $\theta=135^\circ$, 此时

$$I_a=I^{(h)}(i-1,j-1),\ \ I_b=I^{(h)}(i+1,j+1)$$
$$I_c=I^{(h)}(i-3,j-3),\ \ I_d=I^{(h)}(i+3,j+3)$$

b) 若 $u_{135}^{(h)}-u_{45}^{(h)}>T$, 则 $\theta=45^\circ$, 此时

$$I_a=I^{(h)}(i+1,j-1),\ \ I_b=I^{(h)}(i-1,j+1)$$
$$I_c=I^{(h)}(i+3,j-3),\ \ I_d=I^{(h)}(i-3,j+3)$$

2) 水平和垂直像素插值

a) 若 $u_{0}^{(h)}-u_{90}^{(h)}>T$, 则 $\theta=90^\circ$, 此时

$$I_a=I^{(h)}(i-1,j),\quad I_b=I^{(h)}(i+1,j)$$
$$I_c=I^{(h)}(i-3,j),\quad I_d=I^{(h)}(i+3,j)$$

b) 若 $u_{90}^{(h)}-u_{0}^{(h)}>T$, 则 $\theta=0^\circ$, 此时

$$I_a=I^{(h)}(i,j-1),\quad I_b=I^{(h)}(i,j+1)$$
$$I_c=I^{(h)}(i,j-3),\quad I_d=I^{(h)}(i,j+3)$$

这里, T 是阈值, 取值为 0.008.

图 9 给出了 CGI 算法和 PGI 算法的整体思想比较示意图. 可以看出, 两者之间的区别主要在于如何从 LR 图像的扩散梯度计算 HR 图像的扩散梯度, 进而估计 HR 图像的边缘. CGI 算法利用的是最近邻策略, PGI 算法利用的则是基于未知像素周边已知像素的梯度预测策略. 两者之间的实验结果比较将在第 3 节的实验部分详细描述.

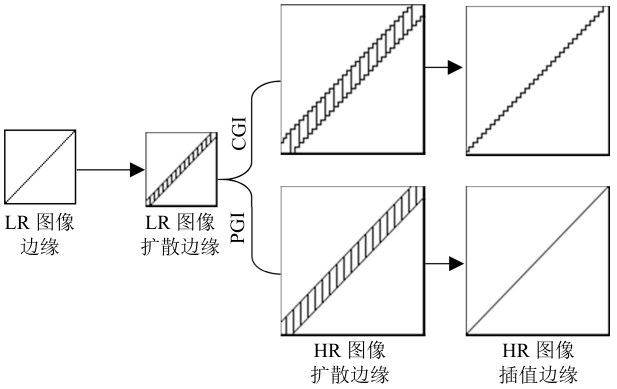


图 9 基于 LR 图像像素性质推测 HR 图像上各像素性质的思想概要图

Fig. 9 The characteristics prediction of pixels on HR based on the characteristics of pixels on LR

2.4 彩色图像插值

目前, 人们主要关注对灰度图像的插值, 对彩色图像插值的研究相对较少, 通常做法是对各个颜色通道采用灰度图像插值算法. 然而, 同一幅彩色图像在不同的颜色空间中可能会有不同的表现形式, 选择在哪个颜色空间处理, 如何处理是一个需要确定的基本问题. 例如, RGB 是一个叠加颜色系统, 在此空间上, 一幅彩色图像可以表示为如下模型:

$$I:\Omega\to\mathbf{R}_+^3=\{(r,g,b):r,g,b\ge 0\} \tag{24}$$

其中, r, g, b 分别表示图像的 3 个颜色通道. 这三个通道同时包含图像的亮度特征和色度特征, 具有较强的相关性. 对它们分别进行处理时, 在边缘处各个通道插值得到的结果相互叠加易产生伪彩色, 降低图像的插值质量.

相比于 RGB 颜色空间, Lab 颜色空间是一种与设备无关的颜色系统, 它通过数字化的方法描述人眼的视觉感应. 在 Lab 颜色空间中, 亮度特征和色度特征是分开的, 即其具有二维独立分布的特点. 因此, 我们可以分别提取图像的亮度特征和色度特征. 同时, Lab 颜色空间的色彩空间范围大于 RGB 颜色空间, 即 RGB 颜色空间可以描述的色彩信息均可以在 Lab 颜色空间得以映射. 图 10 给出了三幅彩色图像分别在 RGB 和 Lab 颜色空间上, 人眼的感知变化差异. 为简单起见, 我们利用 “欧几里得距离” 来描述两幅彩色图像之间的颜色差异. 在 RGB 及 Lab 颜色空间中, 颜色差异的计算公式为

(a) RGB 空间
(a) RGB color space

(b) Lab 空间
(b) Lab color space

图 10　不同颜色空间的颜色差异比较
Fig. 10　The comparison of contrast in different color spaces

$$R_{1,2} = \sum_{c\in\{r,g,b\}} (c_1 - c_2)^2 \tag{25}$$

$$\widetilde{R}_{1,2} = \sum_{\widetilde{c}\in\{l,a,b\}} (\widetilde{c}_1 - \widetilde{c}_2)^2 \tag{26}$$

其中, c 代表 RGB 颜色空间上的一个通道分量; $\widetilde{c}$ 代表 Lab 颜色空间上的一个通道分量. 图 10 (a) 显示在 RGB 颜色空间上, 黑色和灰色之间的对比度 ($R_{g,b} = 255$) 等于黑色与红色之间的对比度 ($R_{b,r} = 255$). 实际上, 对于人眼而言, $R_{g,b}$ 应该小于 $R_{b,r}$. 图 9 (b) 显示在 Lab 颜色空间上, 它们两者之间的关系与人眼感知是一致的 ($\widetilde{R}_{g,b} = 61$, $\widetilde{R}_{b,r} = 120$). 所以, 上述实验结果表明将彩色图像转到 Lab 颜色空间上处理更合理.

图 11 给出了不同颜色空间上彩色图像的插值过程示意图. 现有算法通常在 RGB 空间上分别对彩色图像的 r, g, b 三个通道运用灰度图像的插值算法, 然后合成一幅彩色图像. 考虑到人眼对亮度的敏感程度大于对色度的敏感程度以及插值的效率, 在 Lab 空间上对彩色图像进行插值时, 我们仅对亮度通道 L 采用本文提出的 PGI 插值算法, 而对色度通道采用传统双三次插值. 虽然从 RGB 颜色空间转换到 Lab 颜色空间需要一定的时间消耗, 但是这些时间消耗远小于在色度空间上选用传统插值方法所节省的时间. 例如, 对于一幅 256 像素 × 256 像素的彩色图像, 颜色空间转换大约为 0.16 s, 利用传统双三次插值处理一个通道的时间大约是 0.04 s, 利用 PGI 插值算法处理一个通道的时间大约为 0.7 s, 因此转换到 Lab 空间处理此幅图像的时间大约为 0.94 s (0.16 + 0.7 + 0.04 + 0.04 = 0.94), 而直接在 RGB 空间上处理所需的时间大约为 2.1 s (0.7 + 0.7 + 0.7 = 2.1). 实验结果证明, 对亮度和色度通道分开处理, 不但可以节省大量时间, 而且可以减少插值图像的伪彩色.

3　实验结果

为了测评本文算法, 我们选取了当前 8 种具有代表性的算法进行比较, 分别是: 双三次插值 (Bicubic) 法[3]、边缘导向插值 (NEDI) 法[7]、线性最小均方误差估计插值 (LMMSE) 法[15]、软判决自适应插值 (SAI) 法[8]、对称同态预测插值 (SME) 法[21]、非局部自回归插值 (NARM) 法[19] (该算法虽然是基于样例的插值方法, 但样例数据来源于待插值图像本身, 因此与本文研究的基于局部数据的插值算法存在可比性)、对比度引导的图像插值 (CGI) 法[14] 以及快速边缘扩散插值 (CED) 法[24]. 实验环境如下: 程序编写所基于的平台是 MATLAB, 版本是 2014, CPU 是 2.30 GHz 的 Inter Core, 内存为 4 GB. 各算法都是算法提出者编写的源代码, 且各个参数配置均是算法提出者所推荐的默认参数.

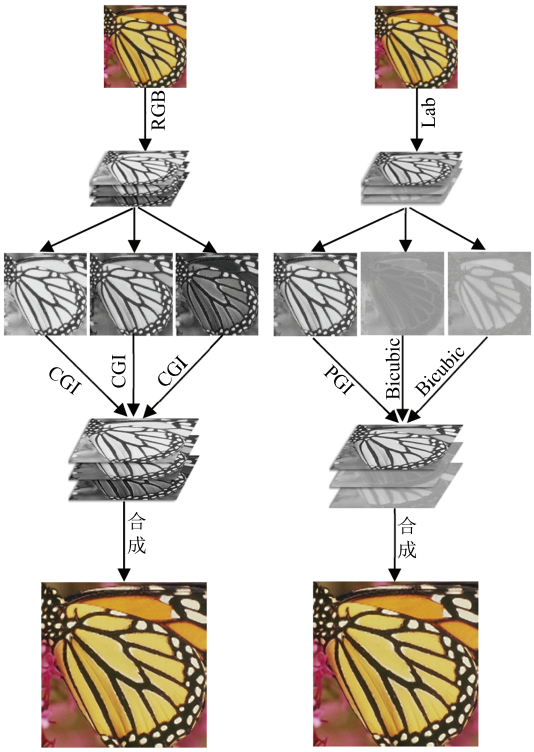


图 11　不同颜色空间上彩色图像插值示意图
Fig. 11　Demonstration of color image interpolation in different color spaces

实验中用到的 12 幅测试图像如图 12 所示, 它们分别为: Airplane (512 像素 × 384 像素)、Bike (512 像素 × 768 像素)、Boats (512 像素 × 512 像素)、Bridge (512 像素 × 512 像素)、Butterfly (288 像素 × 480 像素), Cameraman (256 像素 × 256 像素)、Fence (256 像素 × 256 像素)、House (256 像素 × 256 像素)、Lena (256 像素 × 256 像素)、Parthenon (459 像素 × 292 像素)、Station (256 像素 × 256 像素) 以及 Wheel (160 像素 × 300 像素). 这 12 幅图像是图像插值领域常用的图像, 内容包含纹理和边缘等重要信息, 具有参考价值. 例如 Lena 图像, 其鉴别度较高, 平整的区块、清晰细致的纹路、

渐渐变化的光影、颜色的深浅层次等, 使它在验证影响处理演绎法则时, 相当有成效. Cameraman 和 House 这两幅图像是国际标准测试图像, 用于此, 比较有说服力. 图像 Fence, Bike 以及 Boats 的纹理部分可以用来比较不同图像插值算法对纹理处理的优越性. Station, Wheel, Butterfly 以及 Lena 图像上拥有丰富的边缘信息, 特别是这些图像上存在目前常用插值算法所不能解决的多条边缘相交时的情况, 用于此, 也便于各算法之间进行插值效果比较. Airplane, Bridge 以及 Parthenon 也是图像插值领域常用的测试图像. 为了能将各算法得到的插值图像与对应的参考图像进行比较, 文章采用的 LR 图像都是通过将参考图像进行隔行隔列降采样得到的. 这样对 LR 图像进行 2 倍插值所得到的 HR 图像就与参考图像大小相同, 进而可以对各算法插值的结果进行客观数据的比较.

表 1 是以上 9 种不同插值算法所获得插值图像与参考图像的峰值信噪比 (Peak signal to noise ratio, PSNR). 为便于比较, 表格中最后一行的平均值为各算法相对于双三次插值算法所提高的结果 (峰值信噪比增益). 从表 1 可以看出, 对于测试的 12 幅图像, 由于梯度预测思想的引入, 本文算法的 PSNR 值高于 CGI 算法. 对于 Station, Cameraman, Wheel 以及 Butterfly 这四幅图像, 新算法能够取得最高的 PSNR 值. SAI 算法、SME 算法以及 NARM 算法也能在其他一些测试图像上取得最高值, 但是从表 1 最后一行数据可以看到, 在平均 PSNR 值上本文算法和 SME 算法相当 (相对双三次插值的平均增益同为 0.82), 优于其余 7 种算法.

表 2 是各算法处理 12 幅灰度图像的结构相似性 (Structural similarity index, SSIM) 比较. 可以看到, 由于本文算法在插值过程中对图像边缘的仔细处理, 使得新算法 PGI 在边缘信息比较丰富的 Station, Wheel 以及 Butterfly 这三幅测试图像上能够取得最大的 SSIM 值. 在平均 SSIM 值上, 本文算法略低于 NARM 算法, 但优于其余 7 种算法.

表 3 是各算法处理 12 幅灰度图像的边缘保持性 (Edge preserve index, EPI) 比较. 可以看到, 本

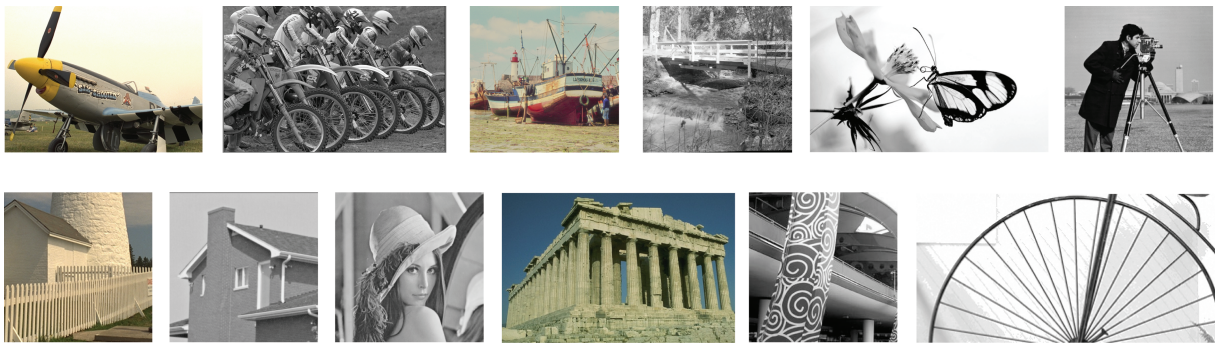


图 12 实验中用到的 12 幅测试图像

Fig. 12 Twelve test images used for simulation experiments

表 1 不同插值算法基于 PSNR 的比较 (dB)

Table 1 A comparison of different interpolation methods with respect to the PSNR (dB)

测试图像	Bicubic (1981)	NEDI (2001)	LMMSE (2006)	SAI (2008)	SME (2010)	NARM (2013)	CGI (2013)	CED (2016)	本文算法
Airplane	29.28	29.76	29.86	30.38	30.29	30.41	30.03	29.97	30.05
Bike	25.96	25.99	26.06	26.99	26.79	26.97	26.71	26.58	26.71
Boats	29.64	29.57	29.66	30.00	30.06	30.26	29.81	29.78	29.79
Bridge	25.85	25.72	25.68	25.94	25.88	25.86	25.70	25.76	25.66
Butterfly	26.17	26.88	26.44	27.40	27.39	27.32	27.68	27.46	27.68
Cameraman	25.26	25.38	25.55	25.77	26.06	25.78	25.75	25.82	25.82
Fence	23.08	21.68	23.09	22.28	23.10	23.21	23.15	23.16	23.14
House	32.06	31.84	32.47	32.73	33.08	33.23	32.70	32.57	32.76
Lena	30.19	30.57	30.50	31.34	30.94	31.38	31.07	31.03	31.11
Parthenon	25.65	25.38	25.74	25.65	25.71	25.81	25.66	25.69	25.65
Station	24.65	25.04	25.07	25.94	26.03	26.16	26.28	26.23	26.39
Wheel	19.59	21.06	19.64	21.53	21.94	20.76	22.38	22.28	22.44
峰值信噪比增益	0	0.12	0.20	0.71	0.82	0.81	0.79	0.74	0.82

表 2 不同插值算法基于 SSIM 的比较 (dB)

Table 2 A comparison of different interpolation methods with respect to the SSIM (dB)

测试图像	Bicubic (1981)	NEDI (2001)	LMMSE (2006)	SAI (2008)	SME (2010)	NARM (2013)	CGI (2013)	CED (2016)	本文算法
Airplane	0.9261	0.9311	0.9330	0.9374	0.9357	0.9410	0.9348	0.9336	0.9355
Bike	0.8610	0.8494	0.8593	0.8782	0.8741	0.8807	0.8739	0.8702	0.8740
Boats	0.8764	0.8735	0.8752	0.8825	0.8842	0.8888	0.8794	0.8788	0.8796
Bridge	0.7982	0.7823	0.7875	0.7992	0.7989	0.8015	0.7932	0.7941	0.7923
Butterfly	0.9508	0.9562	0.9531	0.9621	0.9599	0.9634	0.9638	0.9626	0.9641
Cameraman	0.8649	0.8647	0.8692	0.8732	0.8730	0.8779	0.8717	0.8711	0.8724
Fence	0.7604	0.7411	0.7573	0.7518	0.7654	0.7734	0.7647	0.7647	0.7649
House	0.8747	0.8722	0.8755	0.8757	0.8793	0.8819	0.8781	0.8773	0.8781
Lena	0.9114	0.9129	0.9118	0.9239	0.9191	0.9243	0.9208	0.9203	0.9217
Parthenon	0.7894	0.7719	0.7886	0.7863	0.7847	0.7919	0.7878	0.7877	0.7873
Station	0.8928	0.9023	0.9028	0.9160	0.9187	0.9216	0.9219	0.9208	0.9235
Wheel	0.7723	0.8227	0.7686	0.8415	0.8406	0.8308	0.8619	0.8584	0.8639
平均值	0.8565	0.8567	0.8568	0.8690	0.8695	0.8731	0.8710	0.8700	0.8714

表 3 不同插值算法基于 EPI 的比较 (dB)

Table 3 A comparison of different interpolation methods with respect to the EPI (dB)

测试图像	Bicubic (1981)	NEDI (2001)	LMMSE (2006)	SAI (2008)	SME (2010)	NARM (2013)	CGI (2013)	CED (2016)	本文算法
Airplane	0.7776	0.7793	0.7452	0.7629	0.7946	0.7730	0.8055	0.8050	0.8080
Bike	0.7725	0.8059	0.7469	0.7808	0.7917	0.8027	0.8267	0.8257	0.8323
Boats	0.7473	0.7359	0.7052	0.7279	0.7671	0.7209	0.7607	0.7586	0.7640
Bridge	0.7009	0.6802	0.6648	0.6855	0.7039	0.7055	0.7149	0.7090	0.7218
Butterfly	0.8406	0.8713	0.8175	0.8516	0.8691	0.8657	0.8856	0.8863	0.8874
Cameraman	0.7342	0.7212	0.6902	0.7099	0.7562	0.7234	0.7528	0.7554	0.7542
Fence	0.7015	0.7901	0.6738	0.7213	0.7309	0.7227	0.7118	0.7110	0.7141
House	0.7508	0.7429	0.7213	0.7400	0.7723	0.7231	0.7580	0.7581	0.7594
Lena	0.7928	0.8078	0.7711	0.7928	0.8075	0.7946	0.8236	0.8234	0.8273
Parthenon	0.7018	0.6996	0.6632	0.6803	0.7175	0.6884	0.7159	0.7142	0.7191
Station	0.8475	0.8781	0.8125	0.8433	0.8684	0.8656	0.8979	0.8968	0.9003
Wheel	0.7310	0.8214	0.6785	0.8036	0.7912	0.7455	0.8159	0.8186	0.8178
平均值	0.7582	0.7778	0.7242	0.7583	0.7809	0.7609	0.7891	0.7885	0.7921

文算法与其他 8 种算法相比, 除了在测试图像 Cameraman 和 Wheel 上结果略低于算法 CED 外, 无论是在其他 10 幅测试图像上还是所有 12 幅测试图像的平均值, 结果都高于其他 8 种算法, 说明本文算法在图像的边缘保持性方面能够取得良好的性质.

表 4 是各算法处理 12 幅灰度图像的 CPU 时间比较. 因为 Bicubic 算法是一种不需要考虑图像结构的线性插值算法, 所以在速度上是最快的. CED 算法利用高斯模糊对边缘进行扩散, 其速度仅次于 Bicubic 算法. 由于梯度预测思想的引入, 新算法 PGI 的时间略高于 CGI 算法, 但两种算法是在一个数量级上的. NARM 虽能够取得较好的 PSNR 值, 但需要对每幅输入图像预先建立一个低分辨率图像和高分辨率图像之间的映射关系, 然后进行图像插值, 因此, 该算法的速度是 9 种算法中最慢的, 其平均时间大约是本文算法所消耗时间的 300 倍. SME 算法虽然在平均 PSNR 上表现优秀, 但在所耗费的平均时间方面处于明显的劣势, 约为本文算法的 100 倍. 对于 SAI 算法, 由于原作者只提供了 C 代码, 所以此处不参与比较.

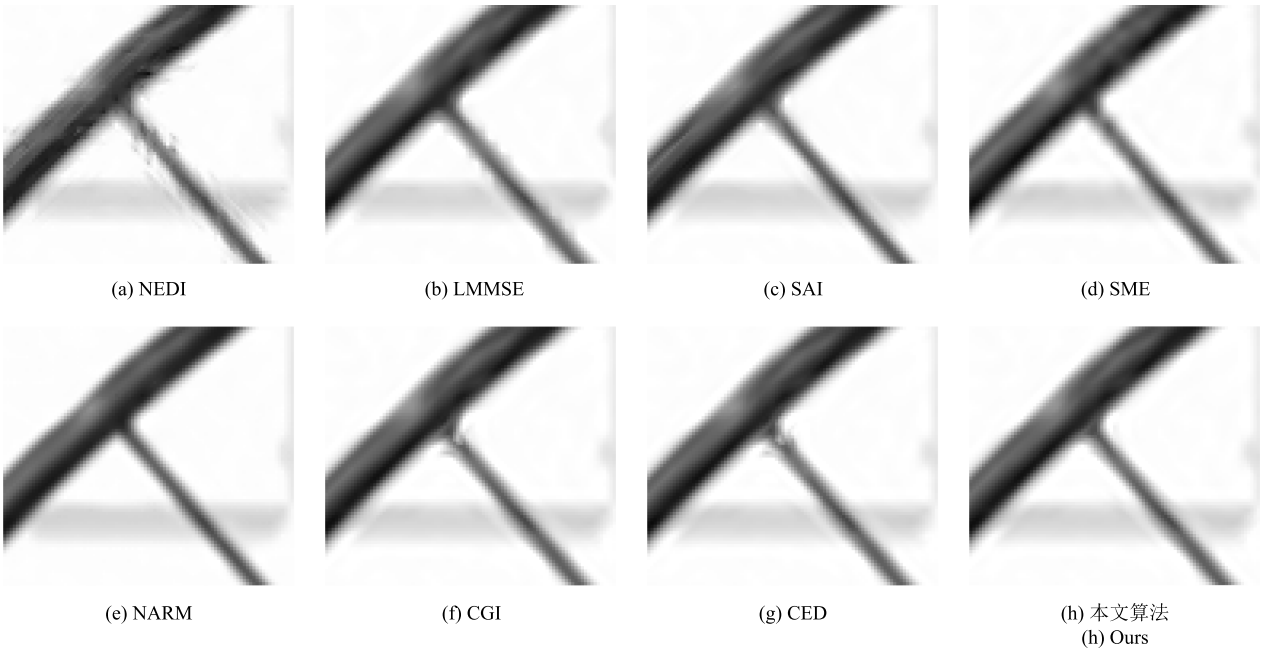
在主观测评方面, 我们比较了各算法在真实边缘保持、虚假边缘抑制和噪声抑制等方面的表现. 总的来说, 早期文献的算法 (Bicubic[3]、NEDI[7] 等) 容易因振铃现象产生虚假边缘, 新算法和近期文献的算法 (SAI[8]、NARM[19] 等) 均可以给出主观效果良好的 HR 图像.

图 13 给出了除 Bicubic 算法外其他 8 种算法对测试图像 Wheel 插值的主观效果比较. 为了能够

表 4 不同算法的插值时间比较 (s)

Table 4 A comparison of different interpolation methods with respect to the CPU time (s)

测试图像	Bicubic (1981)	NEDI (2001)	LMMSE (2006)	SME (2010)	NARM (2013)	CGI (2013)	CED (2016)	本文算法
Airplane	0.052	12.547	11.501	170.446	387.140	1.662	1.178	1.715
Bike	0.048	28.306	23.947	399.348	1014.306	3.530	2.572	3.766
Boats	0.030	17.657	14.464	235.319	580.087	2.126	1.554	2.279
Bridge	0.030	17.886	14.339	233.430	708.391	2.285	1.633	3.113
Butterfly	0.016	7.175	7.024	109.869	233.126	1.006	0.754	1.133
Cameraman	0.011	3.906	3.506	52.474	127.869	0.518	0.399	0.527
Fence	0.008	4.658	3.484	53.359	154.776	0.540	0.395	0.572
House	0.008	3.780	3.480	52.230	141.061	0.539	0.379	0.549
Lena	0.007	3.990	3.483	52.295	157.384	0.545	0.427	0.598
Parthenon	0.018	8.543	7.192	116.360	332.498	1.136	0.812	1.218
Station	0.009	4.092	3.516	52.557	145.134	0.551	0.427	0.618
Wheel	0.008	2.764	2.309	35.045	98.709	0.375	0.304	0.430
平均值	0.020	9.609	8.187	130.228	340.040	1.234	0.903	1.377



(a) NEDI (b) LMMSE (c) SAI (d) SME

(e) NARM (f) CGI (g) CED (h) 本文算法 (h) Ours

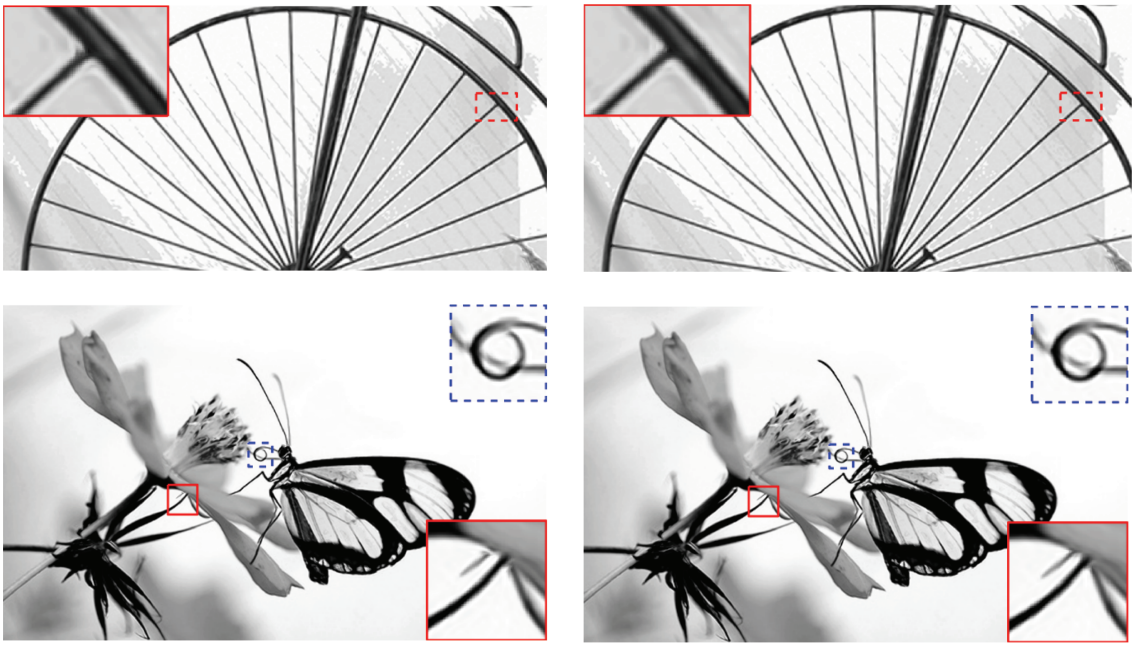
图 13 各算法对 Wheel 图像的插值结果比较

Fig. 13 Comparison of interpolation results on Wheel by different interpolation methods

更清楚地看出不同算法的插值细节, 我们对测试图像进行 4 倍插值. 从图 13 可以看出, NEDI 算法、LMMSE 算法以及 SME 算法在边缘处都出现了锯齿状, 降低了图像的视觉效果. CGI 算法和 CED 算法虽然对边缘的处理比较平滑, 但是却在边缘交界处出现了虚假边缘. 本文算法不仅能够较平滑地处理图像边缘, 而且避免了虚假边缘的产生. NARM 算法虽然能够取得与 PGI 算法相似的视觉效果, 但从表 4 可以看到, 它消耗的 CPU 时间相对较长.

图 14 给出了本文算法和 CGI 算法对两幅测试图像进行 4 倍插值的主观效果比较. 可以看出, CGI 算法对图像上多条边缘交界的地方处理效果不太理想, 出现虚假边缘或边缘缺失的问题. 由于我们在 HR 图像像素性质判断时, 引入梯度预测策略, 能够更精确地判断像素性质, 因此, 很好地解决了上述 CGI 算法在插值中遇到的问题.

图 15 比较了测试的 9 种算法在真实边缘保持和虚假边缘抑制方面的表现. 测试图像为 Bike, 为方便演示, 图中只给出一个局部. 从图 15 可以看出, 算法 Bicubic、LMMSE、SME 以及 CGI 在红色方

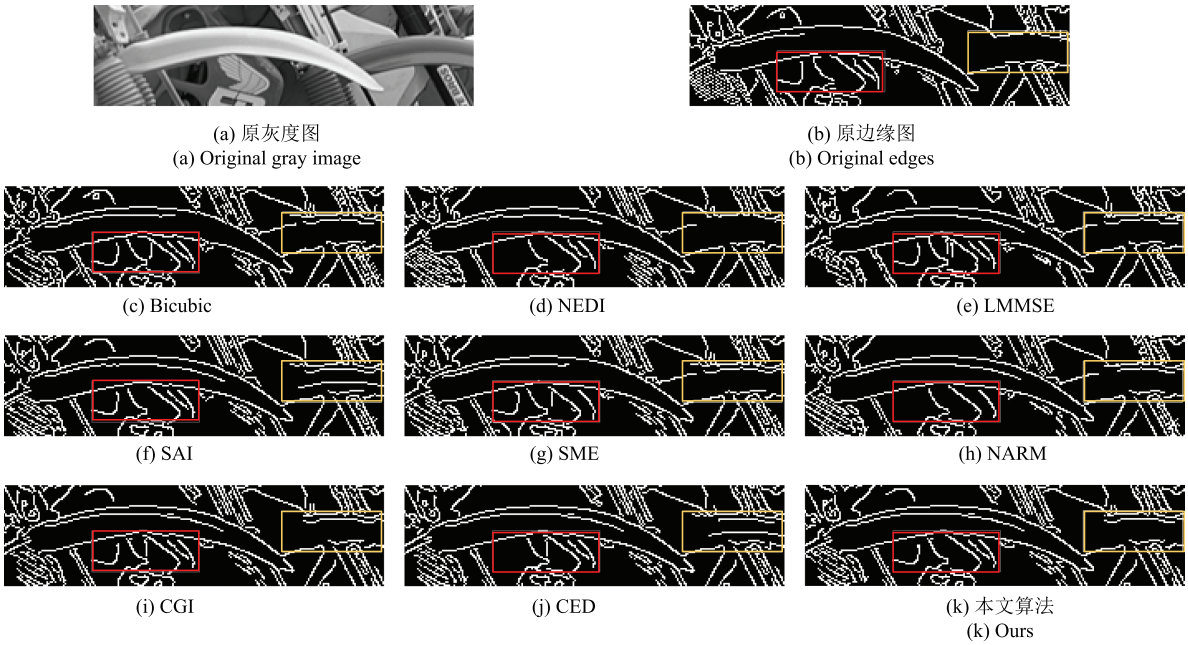


(a) CGI 算法
(a) CGI

(b) 本文算法
(b) Ours

图 14　本文算法和 CGI 法对测试图像的 4 倍插值结果比较

Fig. 14　The comparison of test images via CGI and ours with an interpolation factor 4×4



(a) 原灰度图
(a) Original gray image

(b) 原边缘图
(b) Original edges

(c) Bicubic　(d) NEDI　(e) LMMSE

(f) SAI　(g) SME　(h) NARM

(i) CGI　(j) CED

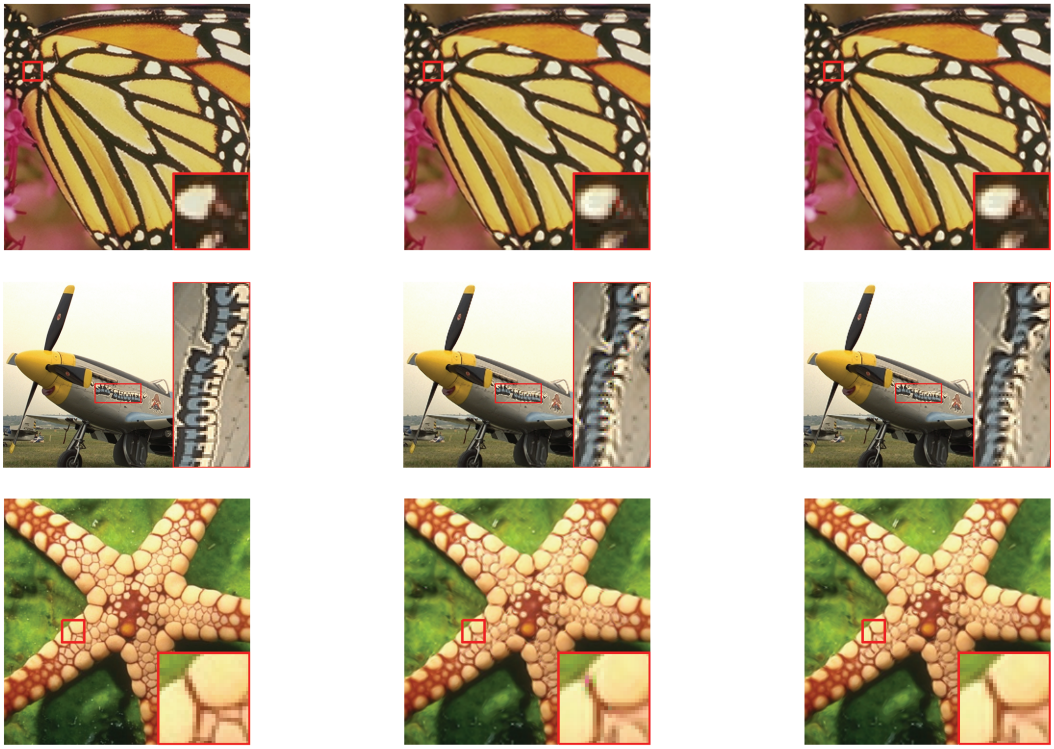
(k) 本文算法
(k) Ours

图 15　各算法插值得到的图像边缘比较

Fig. 15　Comparison of interpolation results on edges by different interpolation methods

框所示的区域内出现了虚假边缘; 算法 SAI 在黄色方框内出现了虚假边缘; 算法 CED 在两个区域内都出现虚假边缘; 算法 NEDI 和 NARM 虽然在上述方框内未出现虚假边缘, 但是却在红色方框内出现了边缘缺失现象; 而本文算法 PGI 无论是在虚假边缘抑制还是在真实边缘保持方面都表现良好.

图 16 是 CGI 算法和新算法对彩色图像的插值结果比较. 文中用到了 Butterfly、Starfish 和 Airplane 这 3 幅测试图像, 其中 Butterfly 以及 Starfish 这两幅图像是文献 [16] 中用到的测试图像, Airplane 是文献 [7] 中用到的测试图像. 可以看出, 将彩色图像的插值转换到与人眼感知相符的 Lab 颜色



(a) 原图
(a) Original image

(b) CGI 插值结果
(b) CGI

(c) 本文插值结果
(c) Ours

图 16 本文算法和 CGI 对彩色图像的插值结果比较

Fig. 16 The comparison of color images via CGI and ours

表 5 CGI 算法和 PGI 算法对彩色图像进行插值的 PSNR (dB) 和 CPU 时间 (s) 比较

Table 5 A comparison of different interpolation methods with respect to the PSNR (dB) and the CPU time (s)

彩色图像	Butterfly		Airplane		Starfish	
	PSNR	CPU time	PSNR	CPU time	PSNR	CPU time
CGI	27.59	1.5	29.94	4.1	28.91	1.4
本文算法	27.75	0.6	29.95	1.4	29.01	0.6

空间, 并对图像的亮度和色度分量分别用不同的插值技术处理, 可以减少彩色图像插值时边缘处伪彩色情况的发生, 提高了图像的插值效果. 同样, 从表 5 的数值结果可以得到相同的结论. 另外, 从表 5 可以看到, 对人眼比较敏感的亮度分量采用新算法, 不太敏感的色度分量采用时间复杂度较低的 Bicubic 插值算法, 最终的插值时间大约是 CGI 算法时间的一半.

4 结论

对图像进行高分辨率插值的关键在于对边缘的处理. 通常的非线性插值算法对边缘的理解与边缘检测方法类似, 即 1～2 个像素宽的线条, 从而插值后会发生边缘模糊的现象. 为此, 本文沿用了边缘对比度引导算法的思想, 首先对低分辨率图像中的边缘做扩散处理, 得到 “带状化” 边缘, 用以引导对高分辨率图像的插值. 在计算高分辨率图像中未知性质的像素时, 我们提出了一种梯度预测策略, 即在未知像素周围选择最相邻的 16 个已知梯度的像素, 以此估算出当前像素的梯度. 通过这种策略, 可以对未知像素的性质 (是否是边缘像素, 以及是边缘像素时其边缘方向) 进行更有效的判断. 在具体插值时, 对边缘像素采用有方向的一维插值, 而对非边缘像素采用二维无方向插值. 实验结果表明, 本文提出的新算法可以取得比现有算法更好的插值效果. 在对彩色图像进行插值时, 本文将通常的 RGB 颜色空间转化为 Lab 颜色空间. 这一做法的主要考虑是 Lab 颜色空间更符合人类的视觉特性, 从而可以减少伪彩色的生成. 此外, 由于仅需在 L 颜色通道中执行基

于梯度预测的图像插值, 因此可以显著降低彩色图像插值的计算复杂度.

References

1 El-Khamy S E, Hadhoud M M, Dessouky M I, Salam B M, Abd El-Samie F E. Efficient implementation of image interpolation as an inverse problem. *Digital Signal Processing*, 2005, **15**(2): 137−152

2 Thévenaz P, Blu T, Unser M. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 2000, **19**(7): 739−758

3 Keys R. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1981, **29**(6): 1153−1160

4 Hwang J W, Lee S L. Adaptive image interpolation based on local gradient features. *IEEE Signal Processing Letters*, 2004, **11**(3): 359−362

5 Shi J Z, Reichenbach S E. Image interpolation by two-dimensional parametric cubic convolution. *IEEE Transactions on Image Processing*, 2006, **15**(7): 1857−1870

6 Pang Zhi-Yong, Tan Hong-Zhou, Chen Di-Hu. An improved low-cost adaptive bicubic interpolation arithmetic and VLSI implementation. *Acta Automatica Sinica*, 2013, **39**(4): 407−417
(庞志勇, 谭洪舟, 陈弟虎. 一种改进的低成本自适应双三次插值算法及 VLSI 实现. 自动化学报, 2013, **39**(4): 407−417)

7 Li X, Orchard M T. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 2001, **10**(10): 1521−1527

8 Zhang X J, Wu X L. Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation. *IEEE Transactions on Image Processing*, 2008, **17**(6): 887−896

9 Chang S G, Cvetkovic Z, Vetterli M. Locally adaptive wavelet-based image interpolation. *IEEE Transactions on Image Processing*, 2006, **15**(6): 1471−1485

10 Huang Hai-Yun, Qi Fei-Hu, Chen Jian, Yao Zhi-Hong. A wavelet-based interpolation of medical images. *Acta Automatica Sinica*, 2002, **28**(5): 722−728
(黄海赟, 戚飞虎, 陈剑, 姚志洪. 基于小波的医学图像插值. 自动化学报, 2002, **28**(5): 722−728)

11 Yang Yun-Feng, Su Zhi-Xun, Hu Jin-Yan. A new edge-holding algorithm of image interpolation. *Journal of Image and Graphics*, 2005, **10**(10): 1248−1251
(杨云峰, 苏志勋, 胡金燕. 一种保持边缘特征的图像插值方法. 中国图象图形学报, 2005, **10**(10): 1248−1251)

12 Jensen K, Anastassiou D. Subpixel edge localization and the interpolation of still images. *IEEE Transactions on Image Processing*, 1995, **4**(3): 285−295

13 Sun H B, Zhang F W, Zheng N N. An edge-based adaptive image interpolation and its VLSI architecture. In: Proceedings of the 2012 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). Hollywood, CA, USA: IEEE, 2012. 1−6

14 Wei Z, Ma K K. Contrast-guided image interpolation. *IEEE Transactions on Image Processing*, 2013, **22**(11): 4271−4285

15 Zhang L, Wu X L. An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Transactions on Image Processing*, 2006, **15**(8): 2226−2238

16 Yamaguchi T, Ikehara M. The quick and high quality image interpolation for single image using multi-filtering and weighted mean. In: Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP). Phoenix, AZ, USA: IEEE, 2016. 2841−2845

17 Dong C Q, Portilla J. Maximum likelihood interpolation for aliasing-aware image restoration. In: Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP). Phoenix, AZ, USA: IEEE, 2016. 564−568

18 Zhu S Y, Zeng B, Zeng L Y, Gabbouj M. Image interpolation based on non-local geometric similarities and directional gradients. *IEEE Transactions on Multimedia*, 2016, **18**(9): 1707−1719

19 Dong W S, Zhang L, Lukac R, Shi G M. Sparse representation based image interpolation with nonlocal autoregressive modeling. *IEEE Transactions on Image Processing*, 2013, **22**(4): 1382−1394

20 Huang J J, Siu W C, Liu T R. Fast image interpolation via random forests. *IEEE Transactions on Image Processing*, 2015, **24**(10): 3232−3245

21 Mallat S, Yu G S. Super-resolution with sparse mixing estimators. *IEEE Transactions on Image Processing*, 2010, **19**(11): 2889−2900

22 Romano Y, Protter M, Elad M. Single image interpolation via adaptive nonlocal sparsity-based modeling. *IEEE Transactions on Image Processing*, 2014, **23**(7): 3085−3098

23 Hou H, Andrews H. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1978, **26**(6): 508−517

24 Ye W, Ma K K. Convolutional edge diffusion for fast contrast-guided image interpolation. *IEEE Signal Processing Letters*, 2016, **23**(9): 1260−1264

陆志芳 苏州大学计算机科学与技术学院硕士研究生. 2014 年获得苏州大学计算机科学与技术学院学士学位. 主要研究方向为图像处理, 计算机视觉.
E-mail: 20144227015@stu.suda.edu.cn
(**LU Zhi-Fang** Master student at the College of Computer Science and Technology, Soochow University. She received her bachelor degree from Soochow University in 2014. Her research interest covers image processing and computer vision.)

钟宝江 苏州大学计算机科学与技术学院教授. 主要研究方向为计算机视觉, 图像分析与理解. 本文通信作者.
E-mail: bjzhong@suda.edu.cn
(**ZHONG Bao-Jiang** Professor at the College of Computer Science and Technology, Soochow University. His research interest covers computer vision, image analysis and understanding. Corresponding author of this paper.)