基于 WSSD的不规则图像块快速匹配

钟 凡[1] 莫铭臻[1] 秦学英[1), 2)] 彭群生[1]
[1)] (浙江大学 CAD&CG国家重点实验室, 杭州 310027) [2)] (山东大学计算机学院, 济南 250101)

摘 要 传统的图像块匹配加速算法都要求待匹配的图像块具有预先定义好的形状。但有时候由于数据损坏、丢失等原因,待匹配块的形状是不规则的(如图像修复)。针对这种情况,提出了一种无损精度的不规则块匹配加速算法,将不规则块匹配扩展为一求最小加权平方差和(WSSD)的问题,块的形状间接地通过每个像素的权重来控制,这使得图像块都能被统一地当成矩形块。为了进行加速,提出了用快速傅里叶变换(FFT)计算 WSSD的方法。并利用待匹配块及其权重在傅里叶变换过程中需大面积补零的特殊性改进了 FFT算法,在不损失精度的前提下,进一步降低了其复杂度。最后以图像修复为例,说明 WSSD是比 SSD更一般的图像块相似度,并为各种图像块匹配的应用提供了一种统一的处理框架。

关键词 不规则 块匹配 图像修复
中图法分类号: TP301.4 **文献标志码**: A **文章编号**: 1006-8961(2010)03-495-07

Fast Irregular Image Patch Matching Based on WSSD

ZHONG Fan[1)], MO Mingzhen[1)], QIN Xueying[1), 2)], PENG Qunsheng[1)]
[1)] (*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou* 310027)
[2)] (*Department of Computer Science, Shandong University, Jinan* 250101)

Abstract Traditional methods for fast patch matching can deal with only image patches with predefined shapes. However, in some cases (e.g. image completion) the patch shape is irregular and different from patch to patch due to destroyed or missing data. In this paper we propose an efficient method for accurate irregular patch matching. We formulate irregular patch matching as a problem to find the minimum weighted SSD (WSSD), and the shape of patches is controlled indirectly with the weights of pixels. In this way all patches can be taken to be rectangular, and then the computation of WSSD can be accelerated with fast Fourier transform (FFT). By taking advantage of the property that large area of patch should be padded with zero. We improve the FFT algorithm and further improve its performance without sacrificing accuracy. At the end of this paper we take image completion as an example to show that WSSD is a more general measure of patch similarity than SSD, and can serve as a uniform framework for various applications that involve image patch matching.

Keywords irregular; patch matching; image completion

0 引 言

图像块匹配是计算机图形学、图像处理及计算机视觉里广泛使用的基本技术,在纹理合成[1-2]、图像风格转换[3]、图像修复[4]和物体检测[5]等领域都有很重要的应用。本文中,称待匹配的图像块为模板块;将与之进行匹配的图像为样本图;样本图中包含很多大小形状与模板块相同的样本块,因此可以把样本图看成是样本块的集合。给定样本图和模板

基金项目: 国家重点基础研究发展规划(973)项目(2002CB312101);国家高技术研究发展计划(863)项目(2007AA01Z326)
收稿日期: 2008-09-23;改回日期: 2009-01-16
第一作者简介: 钟 凡(1982—),男。浙江大学数学系博士研究生。主要研究方向为实时视频分割,图像、视频编辑。E-mail: zhongfan@cad.zju.edu.cn

块,块匹配的目的是找出与模板块最相似的样本块。图像块之间的相似度通常通过向量的 Minkowski距离来衡量。d 维向量 $\vec{u}=\{u_0, u_1, \cdots, u_{d-1}\}$和 $\vec{v}=\{v_0, v_1, \cdots, v_{d-1}\}$的 p 阶 Minkowski距离 (L_p距离)定义如下:

$$L_p(\vec{u},\vec{v})=\left(\sum_{i=0}^{d-1}|u_i-v_i|^p\right)^{\frac{1}{p}} \tag{1}$$

L_1距离即绝对差和 (SAD), L_2距离等价于平方差和 (SSD),这是两种最常用的图像块相似度。当图像块较大的时候, L_p距离的计算会比较费时,又由于样本块通常都很多,因此简单地通过逐一比较来匹配会非常慢。在很多情况下这样的速度是无法容忍的,所以必须寻求块匹配的加速算法。

SSD的计算可采用积分图和 FFT来加速[5-6],积分图用于计算平方项的和,而 FFT则用于计算模板块与候选块的相关性 (correlation)。基于样本细分的方法[1-2,7]先对样本块进行 k-d树状的聚类和剖分,树的每一个结点代表一个聚类中心,自顶向下每个结点所包含的向量个数逐渐减少。搜索的时候则根据模板块与结点中心的距离判断该进入哪棵子树。虽然这种方法搜索时加速效果明显,但当样本量较大时对它们进行聚类和剖分会很慢。在实际中较常用的加速算法是近似最近邻搜索 (ANN)[8]。ANN也是基于对样本的树状剖分,不同的是它是对向量的每一维单独进行的,即每一维都将样本空间一分为二。ANN的加速一般都可达到两个数量级以上,并且其复杂度随着向量维数的增长较慢。最近几年出现了一种实时块匹配的新方法[9],该方法将图像块投射到一组 Walsh-Hadamard基上进行比较。Walsh-Hadamard基实际上为样本块建立了一种排序规则,使得在匹配的时候与模板块差异较大的样本块可以很快地被排除掉。

上述加速算法都要求模板块与样本块具有相同的形状,这一方面是因为采用的是 L_p距离;另一方面则是由于这些算法本身的限制,积分图和 FFT都要求图像块是矩形的;其他几种方法都需要先对样本块按某种规则组织或排序,若模板块与样本块形状不一样,则无法在预先构造好的结构中进行搜索。但是有时候,由于数据丢失、损坏等原因,用来匹配的模板块是不完整的,即模板块的像素只与样本块像素的某个子集对应,并且每个模板块的形状都不一样,称这种情况为不规则块匹配。比如在基于块的纹理合成和图像修复中,目标区域中既包含已知像素又包含未知像素,已知像素被当作模板块进行匹配,匹配的结果被用于填补未知像素。显然,之前的块匹配加速算法是不能用于不规则块匹配的。一些基于全局优化的纹理合成和图像修复算法[1-2,10]首先随机地初始化未知像素,然后通过不断迭代来改善效果。虽然这样做避免了进行不规则块匹配,但随机初始化并不总是合理的,并且迭代也很费时。

针对上述问题,提出了一种无损精度的不规则块匹配加速算法。首先将不规则块匹配扩展为求最小加权平方差和 (Weighted SSD, WSSD)的问题。权重可用于描述模板块的形状,未知像素的权重为零,而已知像素的权重非零。这使得模板块和样本块都能被统一地当成矩形块来处理。为了加速块匹配,扩展了用 FFT计算 SSD的方法,使得 WSSD的计算变得很快。利用模板块及其权重在傅里叶变换的过程中需大面积补零的特性,还改进了 FFT算法,在不损失精度的情况下,进一步降低了其算法复杂度。最后将以图像修复为例,说明 WSSD不仅可处理不规则块,而且还能够满足不同的应用目的,它实际上为图像修复、结构性纹理合成等各种不同的应用提供了一种统一的解决方案。

1 WSSD图像块匹配及其加速

1.1 不规则块匹配与 WSSD

以不带下标的符号表示整张图像的属性,而以带加标的符号表示某一像素的属性。记样本图为大小 $M\times N$ 的图像 $\boldsymbol{I}$,其中某一像素为 $\boldsymbol{I}_{u,v}$。模板块是大小为 $m\times n$ 的图像 $\boldsymbol{P}$,其中某一像素为 $\boldsymbol{P}_{ij}$。

为了方便处理彩色图像,颜色值都被当成向量,其维数等于图像的通道数 C。于是模板块与样本块的平方差和可通过式 (2)来计算:

$$SSD_{u,v}=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\|\boldsymbol{I}_{u+i,v+j}-\boldsymbol{P}_{ij}\|^2 \tag{2}$$

由于在这里 SSD 仅用于比较图像块的差异,因此式 (2)与 L_2距离是等价的。为了处理不规则的模板,定义模板块中已知像素的集合为 Ω,则 SSD 应计算为

$$SSD_{u,v}=\sum_{p_{ij}\in\Omega}\|\boldsymbol{I}_{u+i,v+j}-\boldsymbol{P}_{ij}\|^2=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[p_{ij}\in\Omega]\|\boldsymbol{I}_{u+i,v+j}-\boldsymbol{P}_{ij}\|^2 \tag{3}$$

式中, $[p_{ij} \in \Omega] \in \{0, 1\}$是二值函数, 表示像素 p_{ij} 的颜色是否已知。这相当于为每个像素赋予了一个权重, 使得未知像素不会对结果产生任何影响。更一般地, 可以将式(3)写为如下的形式:

$$WSSD_{u,v} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W_{ij} \| \boldsymbol{I}_{u+i,v+j} - \boldsymbol{P}_{ij} \|^2 \tag{4}$$

式中, W_{ij}为像素 p_{ij}的权重, 可以取为二值函数, 也可以具有其他任何形式。式(4)即加权平方差和(WSSD)。通过设置权重可以消除未知像素对匹配的影响, 于是不规则模板块便可以统一地当成矩形块来处理, 这使得用 FFT加速不规则图像块的匹配成为可能。此外, 为每个像素赋予一个权重使得匹配过程可以包含更多的信息, 从而使匹配结果更符合要求。

1.2　基于 FFT的 WSSD加速计算

通过逐块计算 WSSD来进行图像块匹配的复杂度为 $O(MNmn)$。为了提高匹配速度, 必须寻求加速算法。由于改变 WSSD的权重实际上相当于改变了距离的度量方式, 并且每个模板块的权重都可能不一样, 因此基于样本空间剖分的加速算法都不适用。幸运的是, WSSD使得图像块都可被当成矩形块, 这使得它的计算可以通过 FFT来加速。与 SSD的 FFT加速算法一样[6], 主要利用了 FFT可快速计算图像相关性(correlation)的性质。将式(4)中的平方项展开即得:

$$\begin{aligned} WSSD_{u,v} = & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W_{ij} (\boldsymbol{I}^{\mathrm{T}}_{u+i,v+j}\boldsymbol{I}_{u+i,v+j} + \boldsymbol{P}^{\mathrm{T}}_{ij}P_{ij} - \\ & \boldsymbol{I}^{\mathrm{T}}_{u+i,v+j}\boldsymbol{P}_{ij} - \boldsymbol{P}^{\mathrm{T}}_{ij}I_{u+i,v+j}) = \\ & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W_{ij} \| \boldsymbol{I}_{u+i,v+j} \|^2 + \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (W_{ij} \times \\ & \| \times \boldsymbol{P}_{ij} \|^2) - \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} 2W_{ij}\boldsymbol{P}^{\mathrm{T}}_{ij}\boldsymbol{I}_{u+i,v+j} \end{aligned} \tag{5}$$

式中, 第 1项可视为 W 和 $\| \boldsymbol{I}. \|^2$的相关性, 记为 $C(W, \| \boldsymbol{I}. \|^2)$。注意 $\| \boldsymbol{I}. \|$表示对样本图的每个像素的颜色取模, 而非对样本图 $\boldsymbol{I}$取模。第 2项为与 u, v无关的常数, 不会影响匹配结果, 记为 E。将第 3项中向量的内积展开便得到:

$$\begin{aligned} & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W_{ij}\boldsymbol{P}^{\mathrm{T}}_{ij}\boldsymbol{I}_{u+i,v+j} = \\ & \sum_{k=0}^{C-1}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} 2W_{ij}\boldsymbol{P}_{ijk}\boldsymbol{I}_{u+i,v+j,k} = \\ & \sum_{k=0}^{C-1}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (W_{ij}\boldsymbol{P}_{ijk})(2\boldsymbol{I}_{u+i,v+j,k}) = \\ & \sum_{k=0}^{C-1} C(W \cdot \boldsymbol{P}_k, 2\boldsymbol{I}_k)_{u,v} \end{aligned} \tag{6}$$

式中, 下标 k表示图像的通道; P_k, I_k 分别是模板块和样本图的第 k个通道; $\cdot$ 表示逐像素相乘。可见式(5)中除了常数项外, 其余两项都可以表示为图像的相关性。于是模板块与所有样本块的 WSSD可按式(7)计算:

$$WSSD = C(W, \| \boldsymbol{I}. \|^2) - \sum_{k=0}^{C-1} C(W \cdot \boldsymbol{P}_k, 2\boldsymbol{I}_k) + E \tag{7}$$

由于相关性的计算可以用 FFT加速, 因此通过式(7)计算 WSSD会更快。具体地, 以 $C(W, \| \boldsymbol{I}. \|^2)$为例, 它的 FFT加速计算可表示为

$$C(W, \| \boldsymbol{I}. \|^2) = \mathfrak{I}^{-1}[\mathfrak{I}^*(W) \cdot \mathfrak{I}(\| \boldsymbol{I}. \|^2)] \tag{8}$$

式中, $\mathfrak{I}$表示傅里叶变换, $\mathfrak{I}^*$ 表示傅里叶变换后再取共轭, $\mathfrak{I}^{-1}$为傅里叶逆变换。$C(W \cdot \boldsymbol{P}_k, 2\boldsymbol{I}_k)$的计算也类似。注意在作傅里叶变换之前, 模板块 $\boldsymbol{P}$ 及其权重 W 都需要补零成 $M \times N$ 的图像。将式(8)代入式(7)既得:

$$\begin{aligned} WSSD = & \mathfrak{I}^{-1}[\mathfrak{I}^*(W) \cdot \mathfrak{I}(\| \boldsymbol{I}. \|^2)] - \\ & \sum_{k=0}^{C-1} \mathfrak{I}^{-1}[\mathfrak{I}^*(W \cdot \boldsymbol{P}_k) \cdot \mathfrak{I}(2\boldsymbol{I}_k)] + E = \\ & \mathfrak{I}^{-1}[\mathfrak{I}^*(W) \cdot \mathfrak{I}(\| \boldsymbol{I}. \|^2) - \\ & \sum_{k=0}^{C-1} \mathfrak{I}^*(W \cdot \boldsymbol{P}_k) \cdot \mathfrak{I}(2\boldsymbol{I}_k)] + E \end{aligned} \tag{9}$$

式(9)的推导利用了傅里叶逆变换对加法的分配律, 这样便可以少做 C 次逆变换。虽然将各个颜色通道分别计算 WSSD再相加的结果是一样的, 但那样做需要 $2C$ 次傅里叶变换和 C 次傅里叶逆变换, 直接用向量进行推导使得一些运算能够被合并。由于对 $M \times N$ 的图像, 傅里叶变换和傅里叶逆变换的复杂度都是 $O(MN \log_2 MN)$, 图像通道数 C 可视为常数, E 在实际计算时可忽略, 因此通过式(9)计算 WSSD的总复杂度为 $O(MN \log_2 MN)$。对较大的模板块, 这会比直接计算快很多。

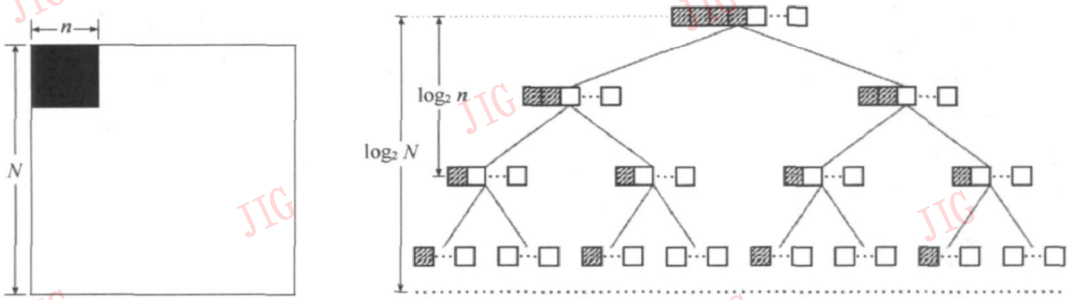
注意在式(9)中, 因为 $\| \boldsymbol{I}. \|^2$ 和 $2\boldsymbol{I}_k$, $k = 0, 1, \ldots, C-1$ 都只与样本图有关而与模板块及其权重无关, 所以对它们的傅里叶变换只需要在设定样本图的时候进行。为了找出给定模板块的最佳匹配, 还需要进行 $C+1$次傅里叶变换和 1次傅里叶逆变换。可见傅里叶变换的速度对上述加速算法的效果至关重要。

1.3　减小 FFT的计算量

虽然 FFT已经是非常高效的算法了, 但对块匹

配而言一般的 FFT算法并不是最优的。利用模板块及其权重都需要大面积补零的特殊性, FFT算法的复杂度可以进一步降低, 并且不会损失精度。

为了方便描述算法, 假设样本图和模板块都是长宽相等的图像, 其大小分别为 $N \times N$ 和 $n \times n$。对模板块补零后用 FFT进行傅里叶变换的复杂度为 $O(N^2 \log_2 N^2)$, 但注意到其中大部分元素都是零, 如图 1(a)所示, 左上角的填充区域为模板块包含的数据, 其他区域都为零。对这些全为零的区域进行计算是不必要的, 利用这一性质可以进一步降低 FFT的复杂度。



(a) 模板块补零后的图像 (b) 拆分输入序列, 深度大于$\log_2 n$的层会包含很多全为零的序列

图 1 减小 FFT的计算量

Fig. 1 Reduce the computation cost of FFT

利用傅里叶变换在维数上的可分离性, 多维傅里叶变换可以分解为多个 1维傅里叶变换来进行, 因此接下来仅考虑 1维傅里叶变换。1维离散函数 $f(x)$, $x = 0, 1, 2, \cdots, N-1$的傅里叶变换定义为

$$F(t) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, e^{-i2\pi tx/N} \tag{10}$$

$$t = 0, 1, 2, \cdots, N-1$$

如图 1(a)所示, 在模板块补零后的图像中, 一行或者一列要么全部元素都是零, 要么只有前 n个元素非零。对于前者, $F(t) \equiv 0$; 对于后者, 虽然非零元素已经很少, 但直接用式 (10)进行计算其复杂度仍为 $O(Nn)$, 一般情况下要比采用 FFT加速后的复杂度 $O(N \log_2 N)$更大。

首先简要介绍一下 FFT算法。设 $N = 2K_1$, 则 $f(x)$的傅里叶变换可以拆分成两个长度为 K_1的函数 $f(2x)$和 $f(2x+1)$来进行[11]:

$$F(t) = \frac{1}{2}\left[F_{\text{even}}(t) + F_{\text{odd}}(t)\, e^{-i2\pi t/N}\right]$$

$$F(t+K_1) = \frac{1}{2}\left[F_{\text{even}}(t) - F_{\text{odd}}(t)\, e^{-i2\pi t/N}\right] \tag{11}$$

$$F_{\text{even}}(t) = \frac{1}{K_1}\sum_{x=0}^{K_1-1} f(2x)\, e^{-i2\pi tx/K_1}$$

$$F_{\text{odd}}(t) = \frac{1}{K_1}\sum_{x=0}^{K_1-1} f(2x+1)\, e^{-i2\pi tx/K_1}$$

式中, $t = 0, 1, 2, \cdots, K_1 - 1$。可见 $F_{\text{even}}(t)$和 $F_{\text{odd}}(t)$分别是函数 $f(2x)$和 $f(2x+1)$的傅里叶变换, 它们可以继续拆分, 直到仅包含一个元素为止。此时函数的傅里叶变换即其本身。

利用这个性质, FFT采用自底向上的方式计算输入函数的傅里叶变换; 在每一层上的复杂度都为 $O(N)$, 一共有 $\log_2(N+1)$层, 因此最终的复杂度为 $O(N \log_2 N)$。

当只有前 n个元素非零时, 上述过程并不需要从最底层开始。如图 1(b)所示, 每一个方格代表一个元素, 每个结点代表一个序列; 填充的方格表示非零元素。可见在第 $\log_2 n$层时序列即只包含一个非零元素, 继续拆分会得到很多全为零的序列, 因此是不必要的。当输入序列仅有第 1个元素非零时, 函数的傅里叶变换可以很容易地得到:

$$F'(t) = \frac{1}{K_l} f'(x)\, e^{-i2\pi tx/K_l} \tag{12}$$

$$t = 0, 1, 2, \cdots, K_l - 1$$

式中, K_l表示第 1层上函数的长度。给定输入图像的大小, 式 (12)中的复数 $e^{-i2\pi tx/K_l}$可以预先计算好; 然后对任意只有前 n个元素非零的输入, 自底向上的过程可以直接从第 $\log_2 n$层开始, 这样复杂度便被降到了 $O(N \log_2 n)$。文中以 FFT* 表示改进后的傅里叶变换算法, 以与一般的 FFT算法区别。

注意在上述讨论中假设 N, n都是 2的整数次方。在这种情况下 FFT算法速度最快, 并且也最容易实现。若 N不是 2的整数次方, 则需要对样本图补零; 若 n不是 2的整数次方, 则可以把模板块中一

些填补的像素当成非零来处理。

对 2维图像, 需要在行和列上分别进行 1维的傅里叶变换。模板块补零后只有前 n行包含非零元素, 其余的行可直接忽略; 又由于输入都是实数, 因此每一行变换后都是关于它们的中心共轭对称的, 即 $F(t)=F^*(N-t)$, $t=1, 2, \cdots, \frac{N}{2}-1$(注意 $F(0)$不包括在内), 并且易知列变换是保持这种对称性的。

利用这个性质, 可以只对前$\frac{N}{2}+1$列进行变换, 然后根据对称性就能得到其余列上的结果。所以该算法最终的复杂度为 $O\left(\left(n+\frac{N}{2}\right)N\log_2 n\right)$。在实际应用中, 因为模板块的大小通常都是相对固定的, 所以 n可以被当作是常数; 又由于 n一般都比 N要小得多, 因此上述算法的复杂度可近似地认为是 $O(N^2)$, 实际加速效果将在下一节中给出。

2　实验与讨论

实验以一台 2.2 GHz CPU 的个人电脑为平台, 并用 C++实现了提出的不规则块匹配加速算法。由于采用 FFT算法加速图像卷积的计算, 以及对 FFT算法的改进都不会损失精度, 因此采用本文方法计算得到的 WSSD以及块匹配的结果也都是精确的。接下来仅从速度上对本文方法进行评估。

表 1是不同图像块大小下 FFT* 与 FFT效率的比较。FFT* 的速度与模板块的大小有关, 模板块越小, 非零区域就越小, 速度便越快; 而 FFT始终将整张图像当成非零区域来处理, 因此其速度仅与图像大小有关。从表 1可以看出, 模板块越小, FFT* 相对于 FFT的加速效果越明显 (快 5~10倍)。注意加速比与非零区域占整张图像的比率并不成正比, 而是成对数比:

$$\frac{N^2\log_2 N^2}{(n+N/2)N\log_2 n}\approx 4\log_n N \qquad (13)$$

式中, 分号上下分别为 FFT与 FFT* 的复杂度, 因此 FFT* 的加速效果并不如直观上那么明显, 毕竟 FFT已经是非常高效的算法了。

表 2是对不规则 (WSSD)块匹配加速的效果。实验中所用的输入都是 3通道的彩色图像, 根据第 1.2节的结论每一次需要 4次正向变换和 1次逆变换, 其中正向变换都用 FFT* 进行。表 2同时给出了用简单方法逐块比较的运行时间, 对比可知对 8×8的块用本文的方法可加速 5倍, 对 16×16的块可加速约 20倍, 对 32×32的块可加速约 80倍。模板块越大, 块匹配的加速效果越明显。逐块比较的复杂度为 $O(N^2n^2)$, 傅里叶逆变换的复杂度为 $O(N^2\log_2 N^2)$, 因此对 3通道的图像, 加速比可计算为

$$\frac{3N^2n^2}{N^2\log_2 N^2+4(n+N/2)N\log_2 n}\approx\frac{3n^2}{2\log_2 N n} \qquad (14)$$

式中, 分号上下分别为逐块比较与本文加速算法的复杂度, 其比值即加速比。当 $n=8$, $N=512$时约为 8, 比实际结果偏高, 这主要是因为傅里叶变换需要用浮点数运算, 而直接计算 WSSD可以只用整数运算 (权重可通过乘上一个较大的数转换为整数)。从表 2可以看出加速算法的速度受模板块大小的影响很小, 这种性质使得它更适合做较大块的匹配。考虑到图像块匹配具有非常广泛的应用, 已将实验中所用程序整理成了 C++库 FPM (fast patch matching)。

表 1　FFT* 与 FFT运行时间的比较

Tab. 1　Time cost (ms) of FFT* and FFT

/ms

N/n	FFT*				FFT
	8	16	32	64	
512	3.52	4.38	5.47	7.11	36.3
1 024	17.2	20.6	24.7	30.1	158
2 048	78.4	90.0	106	124	672

表 2　块匹配加速与不加速的时间对比

Tab. 2　Time cost (ms) without/with acceleration

/ms

N/n	时间 (逐块比较/FFT* 加速)		
	8	16	32
256	32.4/4.85	112/5.00	403/5.15
512	128/26.0	484/26.3	1 859/27.0
1 024	516/106	2 000/106	7 914/107

3　在图像修复中的应用

与纹理合成一样, 早期的图像修复算法都是基于像素的[12], 但这样做不仅速度很慢, 而且只能修复较小的破损区域, 因此在实际中应用较多的是基于块的方法[4]。块匹配通常都是根据已知区域像

素的 SSD, 由于这些像素在模板块中的位置是不固定的, 因此这是一个不规则块匹配问题。

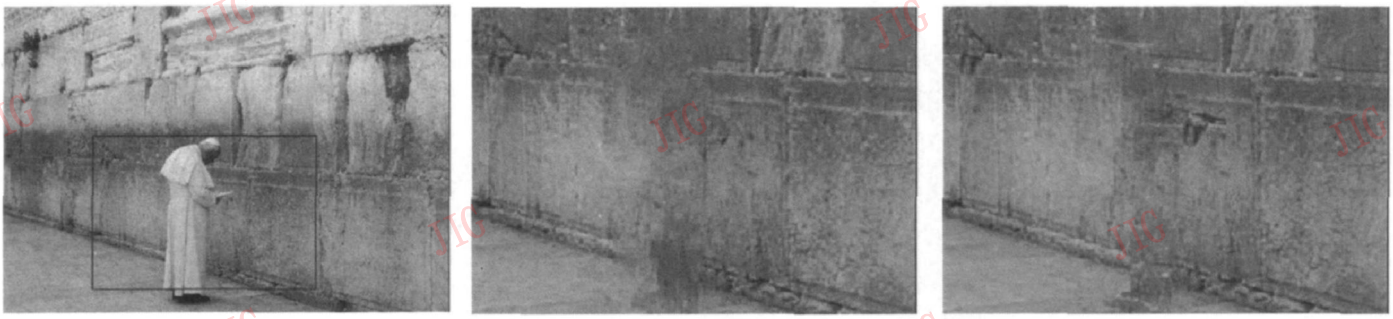
以 Criminisi等人的方法[4]为例。该方法的目标是要保持图像的结构, 通过为破损区域边界上每个像素计算一个优先级, 可以使得结构性较强的部分先被修复, 从而可在一定程度上避免结构被破坏。这一小节的目标首先是要对块匹配进行加速; 其次是通过调整 WSSD的权重, 以使得搜索到的图像块在结构上与已知区域更一致。为此, 可采用如下的权重函数:

$$W_{ij}=\begin{cases} w_0+\lambda\log(1+\|\nabla_{ij}\|^2) & \text{若 } p_{ij}\in\Omega \\ 0 & \text{其他} \end{cases} \tag{15}$$

其中, ∇_{ij}为像素 p_{ij}处图像的梯度; Ω为所有已知像素的集合。w_0, λ为常数。w_0是权重的下限, 用于防止平滑区域的权重太小; λ可以控制图像结构信息对匹配的影响程度。

通过式(15)计算的权重同时解决了不规则块匹配与结构性保持的问题。由于未知像素的权重为零, 因此尽管它们都参与了 WSSD的运算, 但不会对匹配结果产生任何影响; 当 λ> 0时, 已知像素中梯度越大的像素权重越大, 这使得块匹配过程能够优先考虑图像的结构信息, 从而更好地保持图像的结构; 而当 λ= 0时即退化为不规则块的 SSD匹配。图 2给出了按上述方法进行图像修复的结果, 图 2(a)为原图, 图 2(b)(c)为将矩形框内人物移除后再修复的结果; 图 2(b)中 λ= 0, 相当于 SSD匹配; 而图 2(c)中 λ> 0, 表示图像结构特征的像素将获得较大的权重, 因此较好地保持了图像的结构。可见在块匹配过程中考虑图像的结构信息有助于改善修复的效果。实验中所用的图像块大小为 16× 16, 如表 2所示, 此时大约可加速 20倍。

上述方法也可用于结构性纹理合成, 以达到保持纹理结构的目的。更一般地, 由于 WSSD以及提出的加速算法对其权重都没有任何限制, 因此可以结合具体的应用采用不同的权重函数, 比如在计算权重时可以同时考虑像素的空间位置、颜色分布等信息, 以使得匹配结果更符合要求。这实际上为各种需进行图像块匹配的应用提供了一种统一的处理框架。



(a) 原图 (b) ω_0=1, λ=0 (c) ω_0=1, λ=1

图 2 利用 WSSD块匹配进行图像修复

Fig. 2 Image completion with WSSD patch matching

4 结 论

提出了一种针对不规则图像块匹配的加速算法。将不规则块匹配扩展为一求最小加权平方差和(WSSD)的问题。WSSD是比 SSD更一般的图像相似度计算方式, 通过设置权重可以控制每个像素对匹配结果的影响, 这使得图像块可以被统一地当成矩形块来处理。为了加速块匹配, 提出了利用 FFT快速计算 WSSD的方法, 并将其扩展到向量表示的形式, 这使得彩色图像块的匹配变得更有效率。利用模板块及其权重在做傅里叶变换时需大面积补零的特殊性, 还改进了 FFT算法, 避免了大量不必要的运算, 进一步提高了块匹配的速度。提出的加速算法不会损失 WSSD以及块匹配的精度, 因此具有更广泛的适应性。最后以图像修复为例说明了 WSSD块匹配不仅可用于处理不规则图像块, 而且还为各种图像块匹配的应用, 如图像修复、结构性纹理合成等, 提供了一种统一的处理框架。

本文方法在模板块越大时加速效果越明显, 但当模板块较小时其加速效果还不能与规则块匹配的加速算法(如 ANN)相比。将来的工作可以考虑借鉴这些方法的思想, 以进一步提高不规则块匹配的速度。

参考文献 (References)

[1] Wei L Y, Levoy M. Fast texture synthesis using tree structured vector quantization[C]//Proceedings of ACM SIGGRAPH. New York, USA: ACM Press/Addison-Wesley Publishing, 2000: 479-488.

[2] Kwatra V, Essa I, Bobick A, et al. Texture optimization for example-based synthesis[J]. ACM Transactions on Graphics, 2005, 24(3): 795-802.

[3] Hertzmann A, Jacobs C E, Oliver N, et al. Image analogies [C]// Proceedings of ACM SIGGRAPH. New York, USA: ACM Press, 2001: 327-340.

[4] Criminisi A, Pérez P, Toyama K. Object removal by exemplar-based inpainting [C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. New York, USA: IEEE Computer Society, 2003: 721-728.

[5] Viola P, Jones M. Robust real-time object detection [J]. International Journal of Computer Vision, 2001, 57(2): 137-154.

[6] Kilthau S L, Drew M, Moller T. Full search content independent block matching based on the fast fourier transform [C]// Proceedings of International Conference on Image Processing. New York, USA: McGraw-Hill Book Company, 2002: 669-672.

[7] Dellaert F, Kwatra V, Oh S M. Mixture trees for modeling and fast conditional sampling with applications in vision and graphics [C]//Proceedings of IEEE Computer Vision and Pattern Recognition. New York, USA: IEEE Computer Society, 2005: 619-624.

[8] Arya S, Mount D M, Netanyahu N S, et al. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions [J]. Journal of the ACM, 1998, 45(6): 891-923.

[9] He-Or Y, He-Or H. Real time pattern matching using projection kernels[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(9): 1430-1445.

[10] Komodakis N, Tziritas G. Image completion using global optimization[C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. New York, USA: IEEE Computer Society, 2006: 442-452.

[11] Rafael C, Gonzalez, Richard E W. Digital Image Processing (Second Edition) [M]. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1987: 150-213.

[12] Bertalm M, Sapiro G, Caselles V, et al. Image inpainting[C]// Proceedings of ACM SIGGRAPH. New York, USA: ACM Press/ Addison-Wesley Publishing, 2000: 417-424.