计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
42(3):424~430, 2005

应用 PSO 的快速纹理合成算法

张 岩 李文辉 孟 宇 庞云阶
(吉林大学计算机科学与技术学院 长春 130012)
(zhangyanjlu@yahoo.com.cn)

Fast Texture Synthesis Algorithm Using PSO

Zhang Yan, Li Wenhui, Meng Yu, and Pang Yunjie
(*College of Computer Science & Technology, Jilin University, Changchun* 130012)

Abstract Texture synthesis is of importance to computer vision and graphics. The technique has great potential in many applications, especially in image editing, computer animation, real-time rendering for large scale of scenes, realistic and non-photorealisitic rendering etc.. In this paper, a survey of traditional and contemporary texture synthesis methods is presented. Fast texture synthesis algorithm using PSO is an effective texture synthesis algorithm. Particle swarm optimization (PSO) is applied to improve the process of searching and matching in the texture synthesis by patch-based sampling, thus changing the throughout search method of the original algorithm and speeding up the process of synthesis without influencing the quality of the image. It is shown that high-quality texture can be synthesized in several seconds on a midlevel PC, and this algorithm works well for a wide variety of texture ranging from regular to stochastic. How the number of the particles and the iterations influence the speed of the synthesis is analyzed in detail.

Key words texture synthesis; patch-based sampling; particle swarm optimization (PSO)

摘 要 应用 PSO 的快速纹理合成算法是一种高效的纹理合成算法,应用粒子群优化(PSO)算法对基于块采样的纹理合成算法的搜索匹配过程进行了改进,改变了原算法的全遍历搜索过程,在不影响合成质量的前提下加快了合成速度. 本算法对于按序和随机的各种应用都能在一台中等的 PC 机上几秒内合成高质量的纹理. 并对算法执行中的粒子数、迭代次数对合成速度和合成效果的影响进行了详细的分析.

关键词 纹理合成;基于块采样;粒子群优化(PSO)

中图法分类号 TP391.41

1 引 言

纹理合成方法是为了解决纹理映射(texture mapping)中存在的走样问题而提出的. 纹理合成方法大体上可以分为过程纹理合成(procedural texture synthesis)和基于采样的纹理合成(texture synthesis from samples)两种. 但是由于过程纹理合成[1~3]在每合成一种新的纹理时都要调整参数反复实验,非常不方便. 而基于采样的纹理合成技术对给定的小区域纹理样本,根据其自相似性即一小块纹理就能反映整体纹理特点来生成大面积的纹理,可以避免过程纹理合成的缺点,从而越来越成为计算机视觉和图像处理领域研究的热点. Simoncelli 和 Portilla[4,5]利用统计的方法,通过可控金字塔进行纹理合成,对多种纹理都取得了很好的效果. Heeger 和 Bergen[6]利用拉普拉斯和可控金字塔进行纹理合成,使合成纹理的质量得到了提高. De Bonet[7]利用多分辨率金字塔进行纹理合成,采用两个拉普拉斯金字塔及滤波器处理纹理. 在查找匹配时,考虑到已经合成完父层的

收稿日期:2003-08-27;修回日期:2004-03-15
基金项目:国家自然科学基金项目(69883004);国家博士点基金项目(20010183041)

点,从符合条件待选点随机地选一个点来填充. Efros 和 Leung[8]提出了一种更直接的方法,在输出的纹理中设置一些种子,通过给定的邻域在输入样本纹理中选择匹配点,然后选择一点随机填充,效果也很好. Wei 和 Levoy[9,10]提出了一种类似的算法,并采用多分辨率模型进行匹配,利用矢量量化方法对算法进行加速. 以后又有很多文章提出了各自的纹理合成算法,Ashikhmin[11]提出了一种对 Wei 和 Levoy 算法改进的算法,应用到自然纹理中. Battiato 等人[12]用 Antipole 聚类改进了 Wei 和 Levoy 算法,使合成速度和质量都有了很大的提高.

在以往的纹理合成算法中,一般都是采用穷尽法在输入样本纹理中进行搜索寻找匹配点,在符合条件的点中随机选取进行填充,要一个像素点一个像素点地填充,很浪费时间,计算一个很小块的纹理都需要耗费数小时. 虽然以 Wei 和 Levoy 的算法为代表的一类算法对合成速度有了很大的提高,但是自 Xu 等人[13]提出了随机块拼接的快速合成方法后,纹理合成的合成速度与合成质量都得到了很大的提高. 例如 Efors 等人[14]在 2001 年的 SIGGRAPH 会议上,提出了一个简单便捷的基于块缝合的纹理合成算法,通过查找误差最小路径实现各块的拼接,对各种类型的纹理合成的效果都很好. 微软亚洲研究院的基于块采样的实时纹理合成算法[15]与其很类似,他们在块拼接处改为用羽化的方法,效果更好些,而且他们还对算法进行了加速.

在本文中我们提出了一种应用粒子群优化算法(particle swarm optimization,PSO)[16]的快速纹理合成方法,它也是基于块的一种纹理合成算法,与基于块的缝合纹理算法和基于块采样的实时纹理合成算法比较,对于无约束的纹理合成这 3 个算法的采样是类似的. 但是我们的算法在搜索匹配时进行了改进,应用粒子群优化(PSO)算法进行搜索,改变了传统算法的全遍历搜索,使得对于合成同样大小纹理图像时,在合成速度上有了很大的提高,而且合成的质量很高. 虽然基于块采样的实时纹理合成算法也进行了加速,但是最快的加速也要通过实现 3 个加速算法才能达到,大大增加了算法的复杂度.

2　应用 PSO 的快速纹理合成算法

2.1　基于块的算法描述

基于块采样的实时纹理合成算法是基于块纹理合成算法的代表. 它应用输入的样本纹理来合成输出的纹理图像. 图 1(a)中给出了纹理合成过程,把已合成完区域设为父片,在它基础上选取一个宽度为 w_E 的块 E^k_{out} 作为纹理段. 在输入的样本纹理中通过匹配函数寻找匹配的块构成集合 φ_B,然后在 φ_B 中随机选择一个块 B_k 复制到输出纹理. 合成输出纹理的顺序按图 1(b)(c)(d)的顺序来实现,图 1(b)(c)(d)中用 Blending 所指部分是在父片中用来寻找匹配的纹理段的 3 种情况. 对于合成纹理的第 1 个块,从输入的样本纹理中随机地选择一个块进行复制.

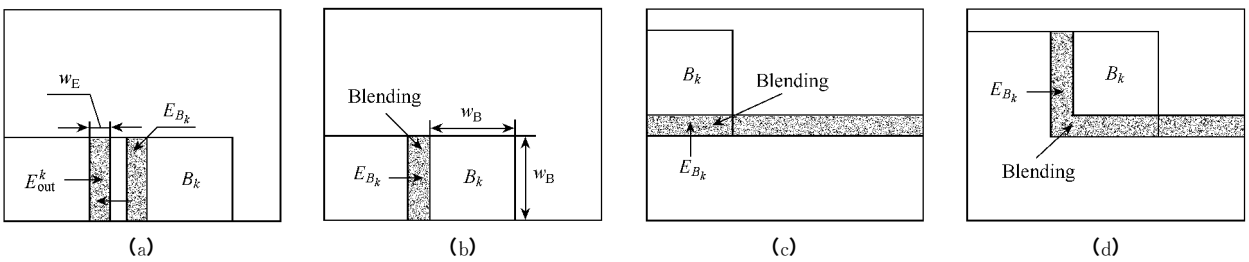


Fig. 1　Patch-based texture synthesis. (a) Texture synthesis process; (b) First configurations for boundary zone matching; (c) Second configurations for boundary zone matching; and (d) Third configurations for boundary zone matching.

图 1　基于块纹理合成. (a) 纹理合成过程;(b) 匹配纹理段第 1 种情况;(c) 匹配纹理段第 2 种情况;(d) 匹配纹理段第 3 种情况

算法中对于一个大小为 $w_{in}\times h_{in}$ 的输入样本纹理,纹理块 w_B 的大小为 $w_B=\lambda\min(w_{in},h_{in})$,这里,$\lambda$ 是随机给出的参数,在(0,1)区间取值. 我们一般情况下取纹理块为 $w_B\times w_B$ 的正方形.

实际上匹配块的搜索是一个高维空间中的对第 k 个最近点搜索问题(an ANN search in high-dimensional space),这个问题已经得到了很好的研究. 正如在文献[15]中所说的要坚持在高维空间中找到精确的最近点,很难找到比平滑搜索(brute-force search)更好的算法,但是因为对于纹理合成,只要找到视觉上相似的就可以,从而可以接受不是最优的解,这样就有很多方法能够达到目的. 在基于块采样的实时纹理合成算法中的基础算法采用了近似最近点[17]的搜索(approximate nearest neighbors)来实现集

合的 φ_B 构造. 而且还应用最优化 kd 树[18](optimized kd-tree)、4 叉树金字塔(quadtree pyramid)、主要分量分析[19](principal components analysis)对算法进行了加速,实现起来过于复杂. 我们对搜索过程进行了改进,对于每个要寻找的纹理块可以把它看成在群中的个体,这样便可应用粒子群优化(PSO)算法[16]进行加速搜索. 在不影响效果的前提下速度也有了很大的提高.

2.2 应用粒子群优化(PSO)的加速纹理合成

2.2.1 粒子群优化算法(PSO)

粒子群优化算法是由 Kennedy 和 Eberhart 于 1995 年提出的,其思想源于对简化的社会系统的模拟[16]. 其主要优点在于算法易于编程实现、算法的收敛性保证以及收敛速度快等.

PSO 首先初始化粒子群包括它的随机位置和速度,并且计算出每个粒子的适应值,然后通过迭代搜索最优解. 对于每个粒子,通过跟踪两个"极值"来更新自己的运动方向. 一个是个体极值,即粒子本身所找到的最优解,用 $pBest$ 表示;另一个是全局极值,即整个种群目前找到的最优解,用 $gBest$ 表示. 当有足够好的适应值或达到最大迭代次数时,算法终止.

粒子根据如下的公式来更新自己的速度和位置:

$$V(t+1) = V(t) + rand() \times c_1 \times (pBest(t) - present(t)) + rand() \times c_2 \times (gBest(t) - present(t)), \quad (1)$$

$$present(t+1) = present(t) + V(t+1), \quad (2)$$

$v(t)$是粒子在时刻 t 的速度;$present(t)$是当前粒子在时刻 t 的位置;$pBest(t)$和 $gBest(t)$如前定义,是时刻 t 的个体极值和全局极值;$rand()$是介于[0,1]之间的随机数;$c_1 - c_2$ 是学习因子,通常 $c_1 = c_2 = 2$.

2.2.2 应用粒子群优化(PSO)的快速纹理合成

在我们的算法中,把粒子的初始位置定义为在输入的样本纹理 I_{in} 中随机给出的若干个像素点,并且给这些粒子一个初始速度,把匹配函数值作为粒子的适应值. 利用 PSO 算法在输入样本纹理中搜索和当前父片中纹理段匹配的纹理块,复制到输出纹理中. 因为不必一定找到最优解,应用这个算法能够很好地找到一个较优的匹配块. 同时,因为 PSO 算法本身没有很多的参数需要调整,使得在执行时,只要根据样本纹理的大小来适当地调整粒子的随机数量和迭代次数,就能很好地控制合成质量和合成速度. 下面对 PSO 算法在本算法中的一些执行细节进行介绍.

(1) 粒子的适应值

首先随机选取 N 个粒子(N 值根据经验选取),对于每个随机的粒子,在当前搜索过程中都有一个适应值. 本算法中适应值的计算公式如下:

$$d(E_{B_k}, E_{out}) = sqrt\{\sum_{i=1}^{A}[(R(p^i_{B_k}) - R(p^i_{out}))^2 + (G(p^i_{B_k}) - G(p^i_{out}))^2 + (B(p^i_{B_k}) - B(p^i_{out}))^2]\}. \quad (3)$$

在式(3)中,E_{B_k}为粒子的当前所在位置决定的纹理段;E_{out}为当前父片中的纹理段;$R()$,$G()$,$B()$表示像素点的三原色值;$p^i_{B_k}$表示在 E_{B_k} 中位置 i 的像素点;p^i_{out}表示在 E_{out} 中位置 i 像素点;A 为这个纹理段中含的像素点数目.

(2) 粒子属性的定义

由于本文工作是二维纹理合成,因而第 i 个粒子的当前位置是在输入的样本纹理中对应位置,记为坐标($PresentX_i$,$PresentY_i$),粒子 i 经历的最好位置记为坐标($LbestX_i$,$LbestY_i$),粒子在这个位置取得的适应值记为 $LBest_i$,所有粒子经历过的最好位置记为坐标($GbestX_i$,$GbestY_i$),粒子在这个位置取得的适应值记为 $GBest$,显然有 $GBest = \min\{LBest_1, LBest_2, \cdots, LBest_n\}$.

对于每个随机的粒子,最初我们都随机地给它一个初始的运动向量 $\boldsymbol{V}_i = (V_{ix}, V_{iy})$,另外定义两个运动向量 $\boldsymbol{L}_i = (L_{ix}, L_{iy})$,$\boldsymbol{G}_i = (G_{ix}, G_{iy})$,定义它们的目的是使得每个粒子能根据当前局部最优解和当前全局最优解的位置改变运动方向和运动速度. 它们的值由以下公式求得:

$$\begin{aligned} L_{ix} &= LbestX_i - PresentX_i, \\ L_{iy} &= LbestY_i - PresentY_i. \end{aligned} \quad (4)$$

$$\begin{aligned} G_{ix} &= GbestX_i - PresentX_i, \\ G_{iy} &= GbestY_i - PresentY_i. \end{aligned} \quad (5)$$

粒子在运动过程中的速度和位置更新公式采用式(1)(2).

我们用 PSO 算法在输入的样本纹理中进行搜索,设定一个比较函数 d_{max},和一个迭代次数(迭代次数是由经验给出),当我们通过 PSO 算法找到的当前全局最优解小于 d_{max},或者达到最大的迭代次数时即停止迭代. 把当前的最优解粒子的位置记录下来,并把它决定的纹理块复制到输出图像中. d_{max} 的定义为下式:

$$d_{max} = \varepsilon \{sqrt[\sum_{i=1}^{A} R(p^i_{out})^2 +$$

$$G(p^i_{\text{out}})^2 + B(p^i_{\text{out}})^2]\Big\}. \quad (6)$$

式(6)中,p^i_{out}为当前父片的纹理段中位置 i 的像素点;A 为这个纹理段中含的像素点数目;$R()$,$G()$,$B()$表示像素点的三原色值;ε 是我们给出的误差系数,是个经验值. 本文实现的程序中取为 0.2.

算法描述:

① 在输入的样本纹理 I_{in}中随机地选择一个大小为 $w_{\text{B}} \times w_{\text{B}}$ 的纹理块复制到输出纹理 I_{out}的左下角. 设 $k = 1$.

② 在当前已合成完的纹理区域中,即父片中选一个宽度为 w_{E} 的纹理片作为纹理段,在输入的样本纹理 I_{in}中应用 PSO 算法寻找和当前父片中纹理段匹配的纹理块.

③ if 应用 PSO 算法在输入样本纹理 I_{in}中能找到解 then

把该位置的粒子决定的纹理块复制到输出的纹理 I_{out}中,$k = k + 1$. 转④;

else 达到了最大的迭代次数;

找一个当前的全局最优解,并把该位置的粒子决定的纹理块复制到输出的纹理 I_{out}中,$k = k + 1$. 转④.

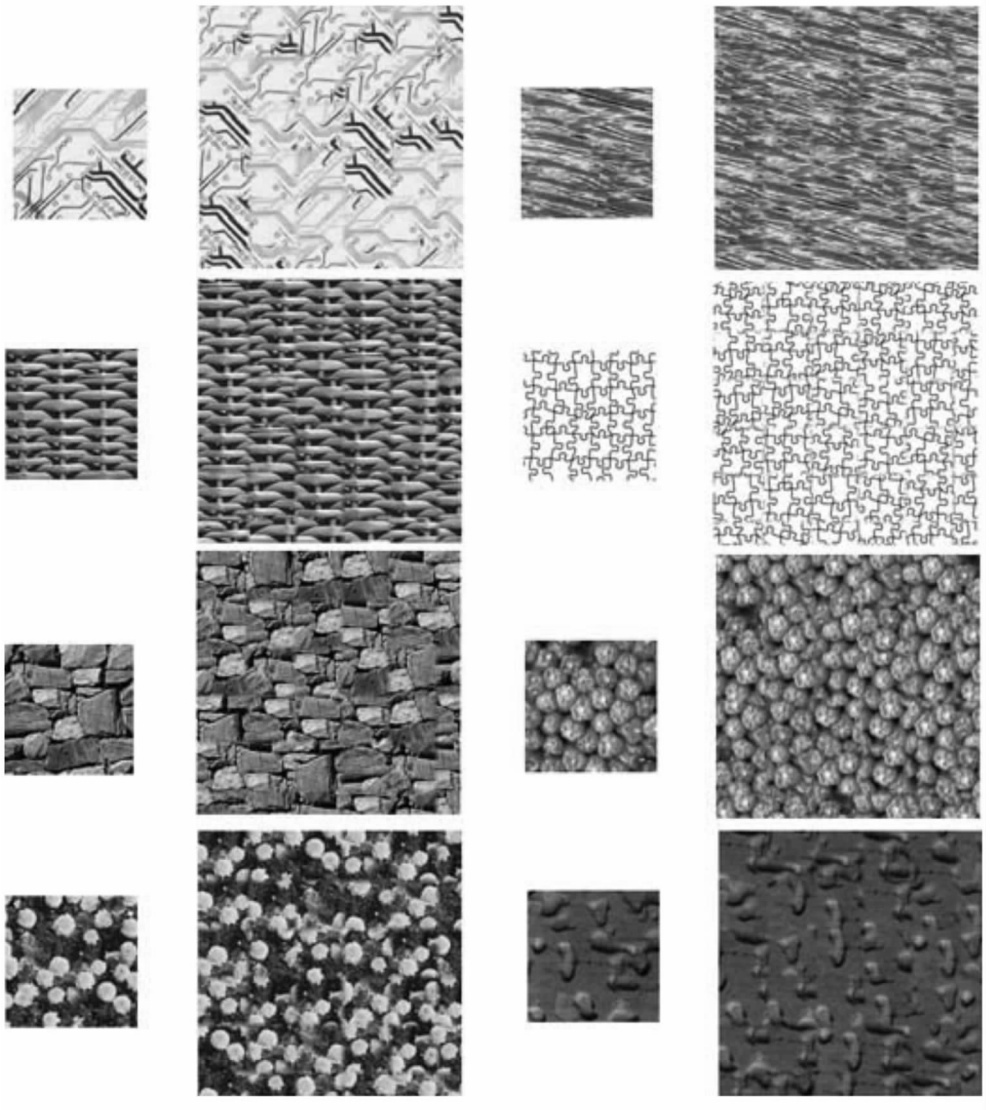
④ 重复②③步,直到输出纹理 I_{out}合成完.

⑤ 应用羽化的方法对对应的纹理段进行融合.

3　合成结果与算法分析

3.1　合成结果

图 2 中给出一些典型的应用我们算法的合成结果,我们算法对从按序到随机的各种应用都能合成



Small images are input sample textures; big images are texture synthesis results.

Fig. 2　Texture synthesis results.

图 2　合成结果

高质量的纹理. 对于输入样本纹理大小为 100×100、合成纹理大小为 200×200 的图像,选取的纹理块大小为 40×40,纹理段宽度 $w_E=4$,应用随机粒子数为 20,迭代次数为 100 次.

3.2 粒子数分析

图 3 给出了粒子随机数对合成效果的影响;表 1 给出了不同粒子数对合成时间的影响. 应用的输入样本纹理大小为 128×128,合成纹理的大小为 200×200. 因为对于 PSO 算法本身的要求,一般粒子数在 20～40 之间[16]就能取得很好的搜索效果,所以在这里我们对粒子随机数为 10,20,40 分别进行了实验.

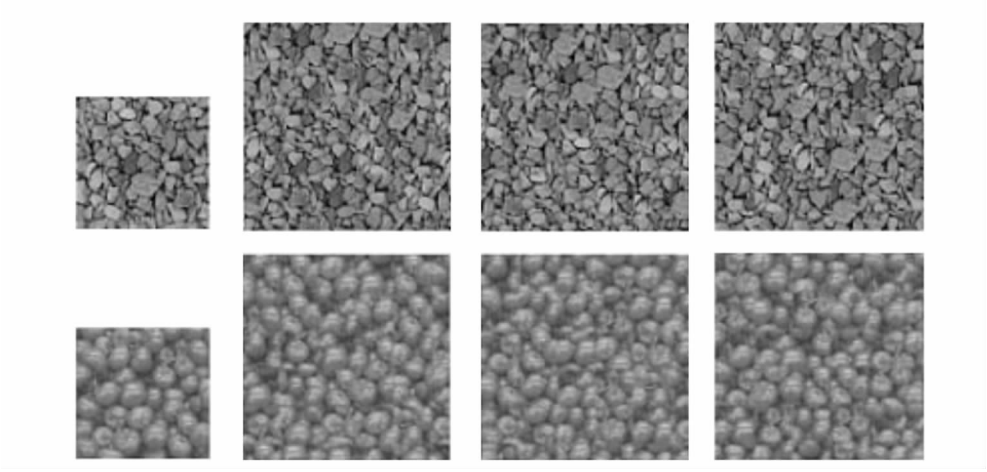


Fig. 3 The effect of different particles. (a) are the input sample textures; (b) uses 10 particles; (c) uses 20 particles; and (d) uses 40 particles.

图 3 粒子数对合成效果的影响. (a) 输入的样本纹理;(b) 粒子数为 10 的效果;(c) 粒子数为 20 的效果;(d) 粒子数为 40 的效果

Table 1 Timing Comparison: Different Particles

表 1 粒子数对合成速度的影响

Particle Counts	Synthesis Time (s)
10	2
20	5
40	11

Note: Timings are measured in seconds on a Dell 4100 Pentium 3 machine with 1GHz CPU for using 350 iterations.

我们从图 3 中可以看出,随着随机粒子数的增加,合成纹理的随机性也在增加. 但对合成质量没有太大的影响,从表 1 可以发现,随着随机粒子数的增加,合成时间也在增加. 当选取的粒子数过多时,有可能会比一般的算法还要慢,选取适当的随机粒子数对快速合成高质量的纹理有很大的影响. 对于一般大小的输入样本纹理,我们把随机粒子数选取为 20 就已经能够取得很好的效果了. 但是当输入样本纹理很小时,我们可以适当地减少粒子数;当输入样本纹理非常大时,我们可以适当地增加粒子数,以保证能够合成高质量的纹理.

3.3 迭代次数分析

当没有足够好的适应值时,PSO 的结束条件就是要达到最大的迭代次数,这样就使得给定的迭代次数对合成速度和合成效果都有一定的影响. 我们在图 4 中给出了迭代次数对效果的影响.

在表 2 中给出了迭代次数对合成速度的影响. 随机的粒子数为 20. 应用的输入样本大小为 128×128,合成纹理大小为 200×200. 对于迭代次数,因

Table 2 Timing Comparison: Fast Texture Synthesis Algorithm Using PSO and Other Algorithms

表 2 不同算法对合成速度的影响

Method	Analysis Time(s)	Synthesis Time(s)	All Time(s)
No acceleration real-time texture synthesis using patch-based sampling[15]	0.0	13	13
QTP + KDTree + PCA[15]	3	2	5
Fast texture synthesis algorithm using PSO (500 iterations)	0.0	9	9
Fast texture synthesis algorithm using PSO (350 iterations)	0.0	5	5
Fast texture synthesis algorithm using PSO (100 iterations)	0.0	1	1

Note: Timings are measured in seconds on a Dell 4100 Pentium 3 machine with 1GHz CPU for using 20 particles.

为原基础算法要求全遍历,128×128 大小的图像要遍历 16384 次.应用 PSO 加速的原则是用尽量少的遍历次数向最优解靠拢,所以我们选取迭代次数为 500,350,100 进行实验.

从图 4 中和表 2 中我们可以看出,迭代次数的变化对合成效果有很小的影响,但对速度有很大的影响.当迭代次数大时,它的合成纹理的随机性比较好些.对于速度,随着迭代次数的减少,合成速度有了很大的提高.如果我们对效果没有更高的要求,可以通过更改迭代次数来提高合成速度.对于迭代次数的选取,应该由输入的样本纹理大小来决定.

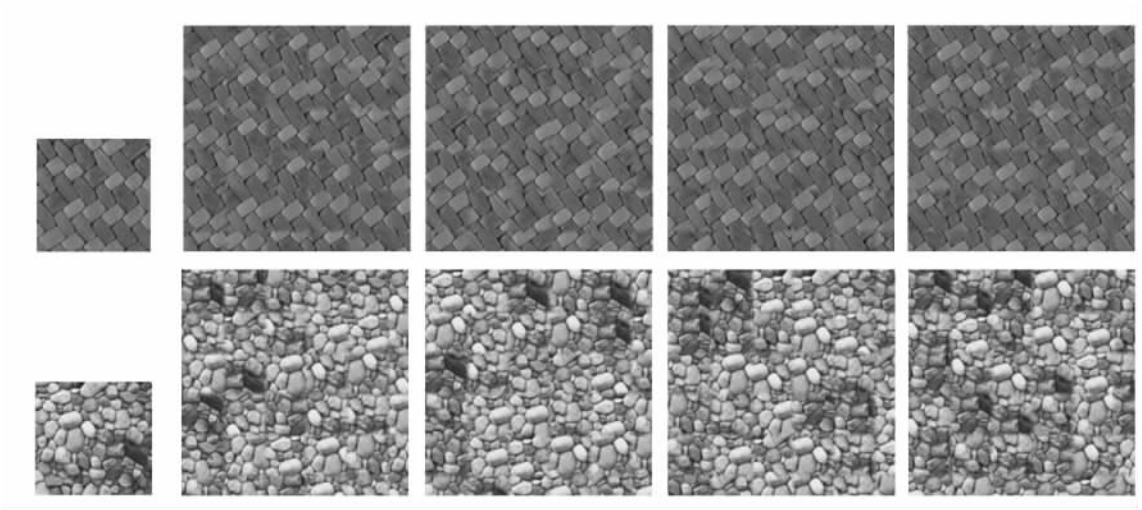


Fig. 4　The effect of different iterations. (a) are the input sample textures; (b) are results generated by patch-based texture synthesis; (c) uses 500 particles; (d) uses 350 particles; and (e) uses 100 particles.

图 4　迭代次数对合成效果的影响.(a) 输入的样本纹理;(b) 原基础算法合成的结果;(c) 迭代 500 次合成的结果;(d) 迭代 350 次合成的结果;(e) 迭代 100 次合成的结果

4　总结与展望

我们提出了应用 PSO 的快速纹理合成算法.该算法能够适应各种纹理,并且速度快,质量高,实现简单.但是本算法对于不同大小的样本纹理,要根据经验人为地指定迭代次数.

在未来的工作中,我们将改进比较函数,使算法能够更好地收敛.并且进一步扩展算法,使它应用到更广的领域中,例如纹理传输、约束的多样图纹理合成等方面.

参　考　文　献

1　A. Witkin, M. Kass. Reaction-diffusion textures. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 1991. 299～308

2　J. Dorsey, A. Edelman, J. Legakis, *et al*.. Modeling and rending of weathered stone. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 1999. 225～234

3　S. Worley. A cellular texture basis function. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 1996. 291～294

4　E. P. Simoncelli, J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In: The 5th Int'l Conf. on Image Processing. Los Angeles: CA Press, 1998. 62～66

5　J. Portilla, E. P. Simoncelli. Texture modeling and synthesis is using joint statistics of complex wavelet coefficients. In: IEEE Workshop on Statistical and Computational Theories of Vision. Fort Collins: CO Press, 1999. 22～23

6　D. J. Heeger, J. R. Bergen. Pyramid-based texture analysis/synthesis. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 1995. 229～238

7　J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 1997. 361～368

8　A. A. Efros, T. K. Leung. Texture synthesis by non-parametric sampling. In: The Int'l Conf. on Computer Vision. New York: ACM Press, 1999. 1033～1038

9　L. Y. Wei, M. Levoy. Fast texture synthesis using tree-structured vector quantization. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 2000. 479～488

10　L. Y. Wei. Texture synthesis by fixed neighborhood searching: [Ph. D. dissertation]. Stanford: Stanford University, 2002

11　M. Ashikhmin. Synthesizing natural textures. In: 2001 ACM Symp. on Interactive 3D Graphics. New York: ACM Press, 2001. 217～226

12　S. Battiato, A. Pulvirenti, D. R. Recupero. Antipole clustering for fast texture synthesis. WSCG'2003, Plzen, Czech Republic,

2003
13 Y. Xu, B. Guo, H.-Y. Shum. Fast and memory efficient texture synthesis. Microsoft Research, Tech. Rep.: MSR-TR-2000-32, 2000
14 A. A. Efors, W. T. Freeman. Image quilting for texture synthesis and transfer. In: Proc. of ACM SIGGRAPH. New York: ACM Press, 2001. 341~347
15 L. Liang, C. Liu, Y Xu, *et al*.. Real-time texture synthesis using patch-based sampling. Microsoft Research, Tech. Rep.: MSRTR-2001-40, 2001
16 J. Kennedy, R. C. Eberhart. Particle swarm optimization. The IEEE Int'l Conf. on Neural Networks, Piscataway, NJ, 1995
17 S. Arya, D. M. Mount, N. S. Netanyanhu, *et al*.. An optimal algorithm for approximate nearest neighbor searching. Journal of ACM, 1998, 45(6): 891~923
18 D. M. Mount. AAN programming manual. http://www.cs.umd.edu/~mount/ANN, 1998
19 I. T. Jollife. Principal Component Analysis. New York: Springer-Verlag, 1986

Zhang Yan, born in 1980. Ph.D. student in College of Computer Science and Technology, Jilin University, whose major research interests include computer graphics, image processing, etc..

张岩,1980 年生,博士研究生,主要研究方向为计算机图形学、计算机图像处理.

Li Wenhui, born in 1961. Professor in College of Computer Science and Technology, Jilin University, doctor supervisor, whose major research interests include computer graphics, image processing, and CAD.

李文辉,1961 年生,教授,博士生导师,主要研究方向为计算机图形学、图像处理、智能 CAD.

Meng Yu, born in 1981. Ph.D. student in College of Computer Science and Technology, Jinlin University, whose major research interests include computer graphics and image processing.

孟宇,1981 年生,博士研究生,主要研究方向为计算机图形学、计算机图像处理.

Pang Yunjie, born in 1939. Professor in College of Computer Science and Technology, Jilin University, doctor supervisor, whose major research interests include computer graphics, image processing, and artificial intelligence.

庞云阶,1939 年生,教授,博士生导师,主要研究方向为计算机图形学、图像处理、人工智能.

Research Background

This paper is supported by the National Natural Science Foundation of China under the Grant No.69883004; National Doctor Station Fund No.20010183041.

Texture synthesis is of great importance to computer vision and graphics. We present a patch-based texture synthesis method by using a particle swarm optimization algorithm to search for an approximate best match patch.

As mentioned, we are not aiming at always finding the best match due to texture synthesis's special feature: synthesized texture should look like the sample texture and keep the randomness of textures. The PSO algorithm will give us an approximate best location. Now we apply the PSO algorithm to texture synthesis. Randomly set a number of positions in input image I and treat these points as virtual particles. These particles can be thought of as virtual points of the image and each particle determines a patch by setting the left upper corner of the patch as the position of this particle. When the particles travel through the image, we compare the patches they determine with the synthesized area and find an approximate best match patch (an approximate best fit position of the particles).

We show that high-quality texture can be synthesized in several seconds on a midlevel PC, and this algorithm works well for a wide variety of texture ranging from regular to stochastic. We have analyzed how the number of the particles and the iterations influences the speed of the synthesis.