纹理合成中基于非标量距离度量的Graph Cut方法

邹 昆[1] 韩国强[2] 沃 焱[2] 张见威[2]
(电子科技大学中山学院计算机工程系 中山 528402)[1]
(华南理工大学计算机科学与工程学院 广州 510006)[2]

摘 要 在逐块纹理合成中,Graph Cut方法被广泛用于优化块间重叠区域的像素取值。传统Graph Cut方法采用的累积距离度量,使得切割路径趋于走捷径而穿过高误差区域。针对此问题,提出了一种基于非标量距离度量的Graph Cut方法。提出了一种基于该度量的高效最小割算法,并证明了其最优性;讨论了改进Graph Cut方法的规则性问题,并给出了解决方法。实验结果表明,由改进的Graph Cut方法所得到的切割路径更加曲折和平滑,块边界隐蔽性更强。

关键词 纹理合成,Graph Cut,非标量距离度量,最小割
中图法分类号 TP391.41 **文献标识码** A

Graph Cut Method Based on Non-scalar Distance Metric for Texture Synthesis

ZOU Kun[1] HAN Guo-qiang[2] WO Yan[2] ZHANG Jian-wei[2]
(Department of Computer Engineering, China Zhongshan Institute, University of Electronic Science and Technology, Zhongshan 528402, China)[1]
(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China)[2]

Abstract Graph cut technique is widely used in patch-based texture synthesis algorithms to optimize patch boundaries. The traditional graph cut method is based on the cumulative distance metric which sometimes leads the path to taking short cuts through high cost areas. To overcome this problem, a graph cut method based on the non-scalar distance metric was proposed. A minimum cut algorithm based on this metric was presented, and its optimality was proved. The regularity problem of the improved graph cut method was discussed and a solution was provided. Experimental results show that the cutting paths by the improved graph cut method are smoother and more circuitous, and the patch seams are less obvious.

Keywords Texture synthesis, Graph cut, Non-scalar distance metric, Minimum cut

1 引言

基于样图的纹理合成是近年发展起来的一种纹理生成技术,它避免了纹理映射中存在的接缝和扭曲问题,也无需像过程纹理合成那样进行繁琐的参数选取,算法适用范围也更广,因此成为计算机图形学、数字图像处理和计算机视觉领域的一个研究热点。该问题可以描述为:给定一个有限的纹理样本,生成任意大的与样本视觉上相似但又有一定差别的纹理图像。

在众多纹理合成算法中,基于Markov随机场模型的区域增长算法是当前主流。根据合成单元的大小可分为逐点合成算法[1-3]和逐块合成算法[4-10]。逐点合成算法由于每次仅合成一个点,合成速度较慢,且对于大尺度的结构特征保持得不好,因此逐块合成算法发展了起来,这类算法的共同点是:每次合成一个图像块,然后用一定的方式将各个块融合在一起。

与逐点合成算法相比,逐块合成算法速度更快,且能保持块内的结构特征,但块间容易出现接缝,因此需采用一定方法来避免或消除接缝,才能得到高质量的纹理。Xu等[4]提出的混沌拼贴合成算法以及Ashikhmin[3]提出的基于相关性搜索的合成算法利用人眼的视觉掩蔽特性来隐藏接缝,缺点是只适用于合成带有高频细节的纹理。Liang等[5]采用羽化的方法来融合块间重叠区域的像素,但会造成一定程度的模糊,在具有强边缘特征的纹理中尤其明显。Efros等[6]采用最小误差边界切割(Minimum Error Boundary Cut, MEBC)来确定重叠区域块边界,该方法可以最小化块重叠区域误差,但要求重叠区域的几何形状是规则的。Kwatra等[7]将Graph Cut技术引入纹理合成,用于确定最优块边界,该方法适用于任何几何拓扑结构的重叠区域,同时可以对已合成区域进行优化。Nealen等[8,9]提出了一种混合合成算法,在逐块合成之后利用逐点方法对重叠区域中误差较大的点重新进行合成。

上述5种方法中,前两种的应用范围有局限性,最后一种

到稿日期:2009-03-25 返修日期:2009-06-10 本文受国家自然科学基金(60573019),广东省自然科学基金博士科研启动基金(8452840301001693),电子科技大学中山学院科研启动基金(408YKQ08)资助。

邹 昆(1980—),男,博士,讲师,CCF会员,主要研究方向为图像处理等,E-mail:zoukun1980@gmail.com;韩国强(1962—),男,教授,博士生导师,主要研究方向为多媒体数据压缩等;沃 焱(1975—),女,博士,副教授,主要研究方向为图像处理等;张见威(1969—),女,博士,副教授,主要研究方向为图像处理等。

是后处理操作，可联合其他方法使用。MEBC 和 Graph Cut 方法都是用来确定最优块边界，而 Graph Cut 方法适用范围更广。

现有的 Graph Cut 方法存在以下两个问题：一是和 MEBC 方法一样，最小化的是累积误差，这会使切割路径倾向于更短的线路，从而可能穿过误差较大的区域；二是所采用的能量函数不能保证满足“规则性”[11]，使得所得到的切割路径有时并不是最优的。

针对以上问题，本文提出了一种改进的 Graph Cut 方法。本方法采用了基于非标量距离度量的最小切割，使得切割路径尽量避开高误差区域，且得到的块边界更加曲折，从而更难被人眼察觉。文中提出了一种基于此度量的最小切割算法，并对其最优性进行了证明。另外，改进的 Graph Cut 方法所得的切割路径最优性已不能够由规则性来保证，文中对此进行了讨论，并给出了有效的解决方法。改进的 Graph Cut 方法继承了传统 Graph Cut 方法的所有优点，且得到的合成纹理质量更高。

2 传统 Graph Cut 方法

Graph Cut 能量最小化方法是由 Boykov 等[12]在 2001 年提出的。近年它被广泛用于解决计算机视觉中的各种标记问题。标记的能量通常为以下形式：

$$E(f)=\sum_{\{p,q\}\in N}V_{p,q}(f_p,f_q)+\sum_{p\in P}D_p(f_p) \tag{1}$$

其中，p,q 为像素位置，P 为像素位置集合，N 为邻域系统，f 为标记，函数 D 用于惩罚标记值与观测值的差异，函数 V 用于惩罚邻域像素间标记的不一致性。最小化该能量的方法是不断地进行标记更替操作，直到能量不能再减少为止。在每次迭代过程中，针对某个标记或某对标记建立相应的图，使得对该图的最小切割所对应的标记状态能够最小化能量。

2003 年，Kwatra 等[7]将 Graph Cut 技术引入纹理合成。其合成过程大致如下：将输入样本每次以不同的位移放置到输出图中，然后在重叠区域计算一条误差最小的切割路径来决定最终的输出区域。这样，输出图中的像素值就可以由其所属样本块的贴块位移来决定。如果将贴块位移作为标记，则纹理合成可以转化为标记问题。因只需考虑相邻像素间的光滑一致性，所以能量函数仅包含式(1)中的 V 项：

$$E(f)=\sum_{\{p,q\}\in N}V_{p,q}(f_p,f_q) \tag{2}$$

当样本以某一位移放置到输出图中后，处于重叠区域的像素或保持其原有值，或更新为新块中的相应像素值，这正如 Graph Cut 能量最小化中的 α-expansion 移动[12]，因此可以建立相应的图，图中边的权值用式(2)计算，然后用最大流/最小切来决定最小误差边界。

传统 Graph Cut 方法采用的是累积距离度量，即最小化的是块边界上的累积误差：

$$|p|=\sum_{i=1}^{n}|e_i|,\ e_i\in p$$

其中，$|p|$ 为块边界切割路径 p 的误差，$|e_i|$ 为 p 中边 e_i 的权值，n 为边的数目。要使 $|p|$ 尽可能小，一方面每条边的误差 $|e_i|$ 要尽可能小，另一方面边的数目 n 要尽可能小。前者是纹理合成所期望的，但后者会使切割路径趋于走捷径而穿过高误差区域，从而给合成结果带来显著的缺陷。

3 改进的 Graph Cut 方法

3.1 非标量距离度量

这里所采用的非标量距离度量最早是由 Pai 和 Reissell[13]引入的，用于解决机器人的移动路线规划问题。由于穿过高代价区域对机器人的健康是有害的，因此这种距离度量鼓励行进路线避开高代价区域，而不惜绕远路。

该距离度量可描述如下。给定两条路径 p_a 和 p_b，将其所含边分别按权值(即距离或误差)从大到小顺序排列：

$$p_a=\{e_{a1},e_{a2},\cdots,e_{am}\}\ |e_{a1}|\geqslant|e_{a2}|\geqslant\cdots\geqslant|e_{am}|$$

$$p_b=\{e_{b1},e_{b2},\cdots,e_{bn}\}\ |e_{b1}|\geqslant|e_{b2}|\geqslant\cdots\geqslant|e_{bn}|$$

其中，$|e|$ 表示边 e 的权值，m 和 n 分别为两条路径所含边的数目。则 p_a 和 p_b 的距离比较可用伪码形式表示如下(假设已通过加入 0 权值边的方式使两条路径的边数相等)：

```
for i=1 to max(m,n)
    if |e_ai| ≠ |e_bi| break;
    if |e_ai| > |e_bi| then |p_a| > |p_b|;
    if |e_ai| < |e_bi| then |p_a| < |p_b|;
    if |e_ai| = |e_bi| then |p_a| = |p_b|;
```

即先比较两条路径中的最大边权值。若相等，则比较第二大边权值。依此类推，直到出现不相等，此时权值较大的边所属路径的距离也较长。如果每对边的权值都相等，则两条路径距离相等。

在用于纹理合成的 Graph Cut 方法中，建图时边的权值与该边作为块边界一部分时两边像素的不一致性相对应。权值越大，所对应的不一致越明显，因此需防止切割路径穿过高权值边。这与机器人的移动规划问题类似，因此非标量距离度量更加适合 Graph Cut。

3.2 基于非标量距离度量的最小割算法

先假定不改变 Graph Cut 建图方法[7]，此时要实现基于非标量距离度量的 Graph Cut，则需找到基于此度量的最小切割路径。该问题可描述为：给定含有两个端结点(源结点 s 和终结点 t)的图，找到一条最短切割路径(基于非标量距离度量)，将该图划分为两个部分，分别包含 s 和 t，如图 1 所示。其中灰色区域为块间重叠区域，含有 4×4 个像素(用方块表示)，虚线表示一条切割线路。

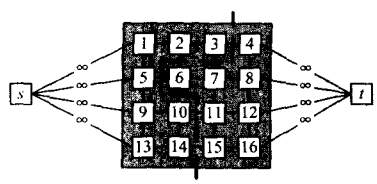

图 1 最小割示意图

3.2.1 算法描述

算法基本思想如下：每次迭代过程中，寻找一条从 s 到 t 的路径。若找到，则将路径上最小权值边切掉(如有多条最小权值边，仅切掉一条)；若找不到，则算法终止。此时连接 s 和 t 所属区域的边构成切割路径。

根据 Ford&Fulkerson 定理[14]，传统的最小割与最大流计算是等价的。基于非标量距离度量的最小割与最大流是不等价的，但它与基于增广路径的最大流算法一样，都建立在路径搜索的基础上，算法效率也都取决于路径搜索的效率。因此，我们修改了 Boykov 和 Kolmogorov 提出的最大流算

法[15]，使其成为一种高效的基于非标量距离度量的最小割算法。

算法保持了两棵搜索树 S 和 T，如图 2 所示。搜索树 S 中的结点用斜线填充，搜索树 T 中的结点用网格线填充。S 和 T 中的边用中等粗线条表示，S 和 T 不重叠，P 表示内部结点，A 表示活动结点(外部结点)，自由结点无填充。活动结点可以吸收相邻的自由结点，加入其所属的树。当活动结点与另一棵树的活动结点相邻时，则找到一条从 s 到 t 的路径，在图中显示为最粗的灰浅色。

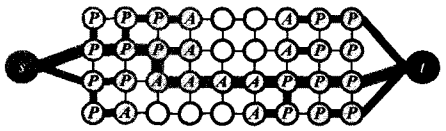

图 2　当找到一条路径时的搜索树示意图[15]

算法初始状态为 S 仅含 s，T 仅含 t，活动结点集合 A 仅包含 s 和 t，孤儿结点集合 O 为空。算法不断重复 3 个步骤。

Step 1　生长阶段：S 和 T 不断生长，直至发现一条从 s 到 t 的路径。具体过程为如下。

```
while A≠Φ
  选取一个活动结点 p∈A
  for each q∈N_p//p 与 q 相邻,即存在连接二者的边
    if TREE(q)=Φ then
      TREE(q):=TREE(p)
      PARENT(q):=p
      A:=A∪{q}
    if TREE(q)≠Φ and TREE(q)≠TREE(p) then
        return P=PATH_{s→t}
  end for
  remove p from A
end while
return P=Φ
```

Step 2　切割阶段：切掉所发现路径上的一条最小权值边(切割后可能打断搜索树从而形成森林)。设所切掉的边为 $e(p,q)$(p 更靠近 s 结点)，则

```
if TREE(p)=TREE(q)=S then
    PARENT(q):=Φ,O:=O∪{q}
if TREE(p)=TREE(q)=T then
    PARENT(p):=Φ,O:=O∪{p}
```

Step 3　恢复阶段：恢复 S 和 T 的结构。试图为 O 中每个结点 p 寻找新的父结点，该结点必须与 p 相邻，且属于同一棵树，并要求其起始结点为 s 或 t；若未找到，则将 p 转为自由结点，若有子结点，则将子结点加入 O。

```
while O≠Φ
    选择一个孤儿结点 p∈O
    for each q∈N_p
        if TREE(q)=TREE(p)andq 祖先为 s 或 t then
            PARENT(p):=q,break
    end for
    if PARENT(p)=Φ then //没找到父结点
        for each q∈N_p
            if TREE(q)=TREE(p) and
            PARENT(q):=p then
                PARENT(q):=Φ,O:=O∪{q}
        end for
        TREE(p)=Φ,A:=A-{p}
end while
```

当 Step 1 中找到路径为空时，算法结束，此时图被划分为两部分 G_s 和 G_t。在被切掉的边中，端点属于不同部分的边构成最小割 C。

本算法复杂度与文献[15]相同。根据文献[15]，在最差情况下，算法比传统的基于增广的最大流算法的复杂度更高，但在通常的计算机视觉相关应用中实际性能更好。

3.2.2　算法最优性证明

3.2.1 节中算法具有以下最优性：

(1) 算法是一种合法的图割算法；

(2) 算法可以最小化切割路径中最大的边权值；

(3) 当搜索到的路径上的每条最小权值边都唯一时，算法采用非标量度量的完整实现，即算法所得到的切割路径是基于该非标量距离度量的最短路径。

证明：

(1) 当算法终止时，不存在从 s 到 t 的路径。这样至少将所建图 G 划分为两个区域，且 s 和 t 处于不同的区域($s\in G_s$，$t\in G_t$)。假设存在第 3 个区域 G_{other}，由于算法对每条找到的路径仅切掉一条边，因此在使 G_{other} 独立出去的那次切割之前：

a) 仅有一条边将 G_{other} 与含有 s 和 t 的区域相连；

b) 存在从 s 到 t 的路径穿过 G_{other}；

由 b)可以推出至少存在 2 条边将 G_{other} 与含有 s 和 t 的区域相连，与 a)矛盾，因此假设不成立。因此算法仅将图 G 划分为两个区域，且 s 和 t 属于不同的区域，所以算法是一种合法的图割算法。

(2) 设集合 P 包含了所有从 s 到 t 的路径(由算法找到的路径集合是 P 的一个子集)。在合法的图割中，每条从 s 到 t 的路径 $p\in P$ 上都至少有一条边被切。最理想情况是所有被切边都是对应路径上权值最小的边，这样该理想切割路径上的最大的边权值为：

$$m=\max\{p.minEdgeCost \mid p\in P\} \tag{3}$$

设由算法所得到的切割路径上最大的边权值为 m'，则 $m'\geqslant m$。另一方面，由于每次切掉的都是所找到路径的最小权值边，因此 m' 必为某条路径上的最小边权值，即

$$\exists\, p\in P,\ p.minEdgeCost=m'$$

由式(3)可知 $m'\leqslant m$。所以 $m'=m$，即算法可以最小化切割路径的最大边权值。

(3) 设由算法所得到的切割路径为：

$$C=\{e_1,e_2,\cdots,e_m\}\quad |e_1|\geqslant|e_2|\geqslant\cdots\geqslant|e_m|$$

假设存在另一条距离更短的切割路径，即

$$\exists C'=\{e_1',e_2',\cdots,e_n'\}\quad |e_1'|\geqslant|e_2'|\geqslant\cdots\geqslant|e_n'|$$
$$|C'|<|C|$$

则 $\exists\, 1<i\leqslant\min(m,n)+1$ 使得 $|e_i|>|e_i'|$ 且 $\forall j<i, |e_j|>|e_j'|$。$i\neq1$ 是因为 e_1 是最大代价边，之前已证明其权值已最小化。

设与 e_i 对应的搜索路径是 p_i，不失一般性，假设等权值的相同边在 C 和 C' 中的排列相同，即如果 $|e_a|=|e_b|$，则在 C 中 e_a 排在 e_b 前面，而 $e_a,e_b\in C'$，则在 C' 中 e_a 还是排在 e_b 前

面。

下面用归纳法证明$|C'|\geqslant|C|$。

先证$e_1=e_1'$。由之前证明，$|e_1|$已经最小化了，而$|C'|<|C|$，所以$|e_1|=|e_1'|$；如果$e_1\neq e_1'$，则在C'中必存在另一条边$e_j'\in p_1$，由于p_1中最小代价边唯一，这样$|e_j'|>|e_1|$，从而e_j'排在e_1'前面，这显然不可能，所以$e_1=e_1'$。

假设对于$k>1$，$\forall j<k, e_j=e_j'$，下面证明$e_k=e_k'$：由$|C'|<|C|$可知，$|e_k'|\leqslant|e_k|$；如果$e_k\neq e_k'$（含e'_k不存在情形），则在C'中必存在另一条边$e_j'\in p_k$，且$|e_j'|>|e_k|\geqslant|e_k'|$，从而$e_j'$排在$e_k'$前面，由假设知$e_j=e_j'$，从而在$p_k$上有两条边属于$C$，这种路径多次跨越子图$G_s$和$G_t$，形如图3中粗直线路径所示，虚线为切割线，斜线填充结点构成G_s，网格线填充结点构成G_t。按照文中提出的路径搜索算法是不可能得到这种路径的，因为两棵搜索树一旦"接触"就形成一条路径，然后立即切割。所以$e_k=e_k'$。

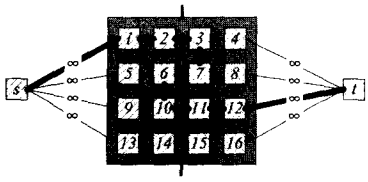

图3　一条路径含多条切割边的示意图

由归纳法可知C中所有边与C'中前m个边相等，这样$|C'|\geqslant|C|$，与$|C'|<|C|$矛盾。所以当每条搜索到的路径上的最小权值边都唯一时，算法所得到的切割路径是最短路径。

当某些搜索到的路径上存在多个最小权值边时，所得到的切割路径可能不是最优的，但由于每次切掉的都是路径上的最小权值边，所得到的切割路径也是一个次优解，因此切割效果较好。

3.3　改进 Graph Cut 方法中的建图问题

传统 Graph Cut 方法采用的是基于累积距离度量的最小割，这与计算从源结点s到终结点t的最大流等价。而在最大流计算中，将一条$s\rightarrow t$路径上所有边的权值同时增大或减少一个常数，不会影响最小割结果，文献[11]中的通用建图方法正是利用了这一点。

改进的 Graph Cut 方法采用的是基于非标量距离度量的最小割，它与最大流计算不等价。只有在同时增大或减少所有边权值的情况下，切割结果才不受影响。因此不能采用文献[11]中的建图方法，而直接采用文献[7]中的建图方法，边权值用式(2)计算(具体形式可有多种)。

相邻像素标记不同时的建图方式如图4所示。p和q为相邻像素结点，标记分别为f_p和f_q，α为新标记，a为辅助结点。在传统 Graph Cut 方法中，要求边$e(p,a)$和$e(a,q)$不能同时被切，这等价于文献[11]中的"规则性"。然而对于本文所提出的改进 Graph Cut 方法，仅仅满足"规则性"是不够的。例如，假设$V(f_p,\alpha)=2, V(f_q,\alpha)=3, V(f_p,f_q)=4$，显然是满足"规则性"的，但由于$V(f_p,\alpha)$和$V(f_q,\alpha)$均比$V(f_p,f_q)$小，仍可能成为$s\rightarrow t$路径上的最小权值边而均被切掉，从而使切掉边的权值与对应的标记能量不一致。

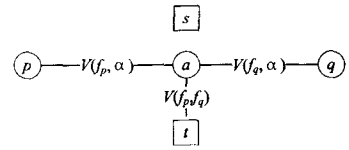

图4　相邻像素标记不同时的建图方式

通过在切割时引入控制，使$e(p,a)$和$e(a,q)$不同时被切。当待切边(假设为$e(p,a)$)与辅助结点相连时，判断与辅助结点相连的另一条边$e(a,q)$是否被切。如果没有被切，则直接切掉边$e(p,a)$；否则，通过计算局部匹配误差来判断最终切掉哪条边：

$$Error(p,\alpha)=\sum_{r\in N_p}V_{p,r}(\alpha,f_r)$$

其中，N_p为p的邻域，其它符号意义与式(2)相同。$Error(p,\alpha)$表示像素p标记更换为α后与周围像素的匹配情况，与切掉边$e(a,q)$的情形相对应，$Error(q,\alpha)$与切掉边$e(p,a)$的情形相对应。如果$Error(p,\alpha)\leqslant Error(q,\alpha)$，说明在$e(p,a)$和$e(a,q)$不同时被切的情况下，切掉边$e(a,q)$误差更小，这样在当前路径上找到权值第二小的边(也可能为与$e(p,a)$权值相等的另一条边)进行切割；如果$Error(p,\alpha)>Error(q,\alpha)$，则切掉边$e(p,a)$，然后连接已切边$e(a,q)$。

连接已切边$e(a,q)$后还需作相关处理(此时a必与t相连)：

if $TREE(q)\neq\Phi$ and $TREE(q)\neq TREE(a)$ then

　　return $P=PATH_{s\rightarrow t}$

if $TREE(q)=\Phi$ then

　　$TREE(q):=TREE(a), PARENT(q):=a, A:=A\cup\{q\}$

4　实验与分析

本文提出的改进 Graph Cut 方法是用来决定块间最优边界的，可用于多种逐块纹理合成算法中。以下实验建立在前期工作[10](对文献[7]工作的改进)的基础上，Graph Cut 部分分别采用原方法和改进 Graph Cut 方法进行比较。实验环境为 P4 3.4G CPU，1G 内存，使用 VC 6.0 编程。

图5给出了传统 Graph Cut 和改进 Graph Cut 方法的切割路径和效果对比。为公平起见，块和贴块位移均相同。其中，(a)和(b)分别为传统 Graph Cut 方法所得到的合成结果及对应的切割路径(用蓝线表示)，(c)和(d)分别为改进 Graph Cut 方法的合成结果及对应的切割路径。可以看到：(a)中存在明显的不连续，而(c)中则没有；(b)中的路径较直且有棱角，而(d)中的路径更加曲折。因此，改进的 Graph Cut 方法使得块间过渡更加平滑，且由于人眼对于光滑而曲折的边界不敏感，(d)中边界的隐蔽性更好，更不容易被人眼察觉。

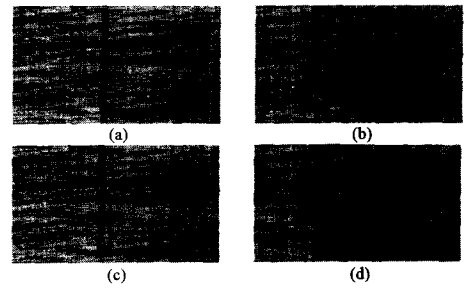

图5　传统 Graph Cut 和改进 Graph Cut 的切割路径和效果对比

图 6 给出了传统 Graph Cut 和改进 Graph Cut 方法所得合成结果的对比，上方小图是样本纹理，下方左边是传统 Graph Cut 方法的合成结果，右边是改进 Graph Cut 方法的合成结果，所有输出图像均为 360×360 大小。合成时第一次的贴块位移相同，每次选取匹配误差最小的位移进行贴块。可以看到，改进 Graph Cut 方法所得到的块边界更加平滑，合成纹理质量更高。

一般来说，对于具有低频和局部特征的纹理，如图 6 中的 OCEAN 和 WOOD，改进效果更明显（传统 Graph Cut 方法合成的 OCEAN 纹理存在大量明显的不一致，WOOD 纹理存在两处明显的切割边界，而由改进的 Graph Cut 方法合成的纹理无以上现象）；而对于高频纹理，如 BAMBOO 和 BARK，由于视觉掩蔽效果的存在，块间接缝不明显，效果提升也不明显（仔细观察仍可看到传统 Graph Cut 方法合成结果中的瑕疵，在图中用红圈标出）。

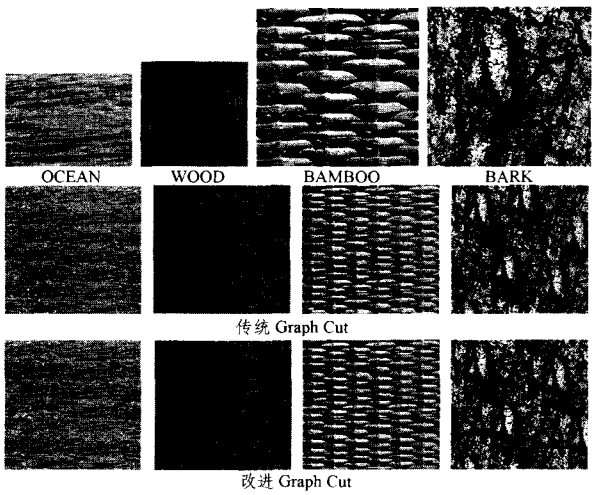

图 6 传统 Graph Cut 和改进 Graph Cut 合成结果对比

表 1 列出了在合成图 6 中纹理时两种合成算法所用的贴块数和时间。对于 OCEAN 和 BARK 纹理，基于改进 Graph Cut 的合成算法的平均贴块时间稍长，而其他两个纹理的平均贴块时间稍短，这说明影响合成速度的最主要因素还是贴块位移搜索时间，Graph Cut 方法的改进并没有使合成速度变慢。从理论上说，改进的 Graph Cut 方法中的最小割算法复杂度与原最小割算法相同，但对辅助边的切割控制会增加一定运算量，这取决于连接已切边的次数。

表 1 图 6 中的合成时间数据

样本纹理	基于传统 Graph Cut 的合成算法		基于改进 Graph Cut 的合成算法	
	总贴块数	合成时间	总贴块数	合成时间
OCEAN	20	0.661s	25	0.932s
WOOD	33	1.095s	36	1.061s
BAMBOO	13	1.240s	13	1.105s
BARK	13	1.030s	13	1.449s

结束语 本文提出了一种改进的 Graph Cut 方法，采用了基于非标量距离度量的最小割，这使得切割路径能够尽量避开高误差区域，且得到的路径更加曲折和平滑，因此块边界的切割质量更高，更难被人眼察觉。受传统基于增广路径的最大流算法的启发，提出了一种基于该非标量距离度量的高效最小割算法，并证明了其最优性。文中分析了改进的 Graph Cut 方法中的建图问题，并给出了相应解决方案。作为一种通用的优化块间重叠区域的方法，改进的 Graph Cut 适用于任何可采用传统 Graph Cut 的纹理合成算法中。

参 考 文 献

[1] Efros A A, Lung T K. Texture synthesis by non - parametric sampling[C]//Proceedings of IEEE International Conference on Computer Vision. Corfu, 1999: 1033-1038

[2] Wei L Y, Levoy M. Fast texture synthesis using tree-structured vector quantization[C]//Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH. Los Angeles, 2000: 479-488

[3] Ashikhmin M. Synthesizing natural textures[C]//Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics. Los Angeles, 2001: 217-226

[4] Xu Y Q, Guo B, Shum H Y. Chaos mosaic: fast and memory efficient texture synthesis[R]. MSR-TR-2000-32. Redmond: Microsoft Research, 2000

[5] Liang L, Liu C, Xu Y Q, et al. Real-time texture synthesis using patch-based sampling [J]. ACM Transactions on Graphics, 2001, 20(3): 127-150

[6] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer[C]//Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH. Los Angeles, 2001: 341-347

[7] Kwatra V, Schodl A, Essa I, et al. Graphcut textures: image and video synthesis using graph cuts[J]. ACM Transactions on Graphics, 2003, 22(3): 277-286

[8] Nealen A, Alexa M. Hybrid texture synthesis[C]//Proceedings of the 14th Eurographics workshop on Rendering. Leuven, 2003: 97-105

[9] Nealen A, Alexa M. Fast and high quality overlap repair for patch-based texture synthesis[C]//Proceedings of the Computer Graphics International. Washington, 2004: 582-585

[10] 邹昆，韩国强，李闻，等. 基于 Graph Cut 的快速纹理合成算法[J]. 计算机辅助设计与图形学学报，2008，20(5)：652-658

[11] Kolmogorov V, Zabih R. What energy functions can be minimized via graph cuts[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(2): 147-159

[12] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(11): 1222-1239

[13] Pai D K, Reissell L M. Multiresolution rough terrain motion planning[J]. IEEE Transactions on Robotics and Automation, 1998, 14(1): 19-33

[14] Ford L, Fulkerson D. Flows in networks[M]. Princeton University Press, 1962

[15] Boykov Y, Kolmogorov V. An experimental comparison of min-cut / max-flow algorithms for energy minimization in vision[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(9): 1124-1137