基于描绘单幅图像的三维树木交互建模系统*

刘一松，许祥虎，曾兰玲，詹永照
（江苏大学 计算机科学与通信工程学院，江苏 镇江 212013）

摘　要：为了提高树木建模的真实感与可交互性，提出了一种基于单幅树木图像或直接手绘的交互式植物建模系统。系统根据植物形态学规律构建了标准三维树木的模板。通过描绘图像中树的主要枝干，结合叶序周规律，基于三维模板树将其从二维图像变换到三维模型，并通过弯曲、增加和删除枝干以及添加树叶等交互式操作生成和修改三维树木模型。实验结果表明，该系统能便捷快速地生成真实感较强的三维树木。

关键词：植物建模；植物形态学；分形；交互式
中图分类号：TP391.41　　**文献标志码**：A　　**文章编号**：1001-3695(2012)06-2364-04
doi:10.3969/j.issn.1001-3695.2012.06.097

3D trees interactive modeling system by sketching according to single image

LIU Yi-song, XU Xiang-hu, ZENG Lan-ling, ZHAN Yong-zhao
(*School of Computer Science & Telecommunication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: To improve the interactivity and realist of the tree models, this paper presented an interactive system for designing 3D models of trees by sketching according to single image of the original tree or freehand sketches. It created a standard 3D tree model based on the principle of plant morphology. By sketching original tree's primary branches, the original tree could be transformed from 2D image to 3D model according to the botanical phyllotaxis discipline. It generated and modified the 3D tree model using interactive operations as bending, adding, deleting the branches, and adding leaves etc. The results show that this method is quick and easy for generating 3D trees models.

Key words: plant modeling; plant morphology; fractal; interactive

在虚拟现实技术迅速发展的今天，植物的仿真模拟已经应用在许多方面，如游戏电影等娱乐业、农业研究、城市美化和一些防自然灾害的研究等。然而由于植物种类、外界环境和人为修剪等一系列原因，导致植物的几何形态颇为复杂，所以植物的建模与仿真一直都是计算机图形学中一个具有挑战性的课题[1]。由于植物的形态特征非常复杂，很难找到一个完整的生长机模型来描述其特征。目前许多学者提出了图像方法，根据不同角度拍摄的植物图像对植物进行三维重建。Quan 等人[2]提出了需要很少交互的重建方法，通过对目标树木进行 360°照片的拍摄，然后进行采样提取点云，完成叶子和枝干分开建模，但系统耗时比较大。Reche 等人[3]提出了一种体积的方法，假设树是由很多小体积构成以及图像中的每个像素点都是由许多枝干或树叶的投影构成，进而计算出每个体积的不透明度，并在每个体积上贴纹理达到三维重建。该方法同样耗时很大，平均时间达到 1 h，而且只适用于稀疏类型的植物。Neubert 等人[4]提出了一种结合图像和粒子系统的方法，该方法能够在较短的时间内生成与图像匹配的植物模型，但是同样不适合茂密的树型。

在植物的建模中，可交互的设计是非常重要的。Lintermann 等人[5]提出了图形化的交互界面使得用户能直观地设计树木的 L 系统规则，但用户必须对规则有一定的了解。Okabe 等人[6]提出了手画树的模型，即通过鼠标或者手画板绘制的二维图形生成三维树。该方法虽然采用了基于例子的编辑操作，减少了交互量，但只有少量的叶片排序方式，并且从手绘 2D 变换到 3D 时只是遵循枝干间的间隙最大化这一简单原则，没有考虑树木的本身形态特性，如叶片和枝干的排序规律，绘制结果缺乏真实感。Ijiri 等人[7,8]设计了一个花卉交互式建模系统，由于对手绘的处理较为简单，加上花卉形状的复杂性，该系统只能生成规则的花瓣，因而生成的花卉显得较为单调。

针对图形方法难以寻找完整的生长机模型的问题，以及由于图像方法处理枝干间和叶片间的遮挡导致提取植物特征过于复杂，本文结合图形和图像技术，根据植物形态学规律用分形方法建立标准的模板树。在此基础上，依据叶序周规律设计了一种只需描绘单幅目标树图像或直接手绘就能生成其逼真的三维模型的交互式植物建模系统。

1　系统建模过程

系统建模过程如图 1 所示。用户先根据目标树图像（图 1(a)）设置生成模板树的参数。如果用户对目标树了解较多，可以设置根枝干的长度和粗细、树木的递归深度、叶序周、子枝干数、子枝干的起始比例等参数；如果了解偏少，可以设置只含有一棵根枝干的模板树以及叶序周等参数（图 1(b)）。然后

收稿日期：2011-10-26；**修回日期**：2011-12-10　　**基金项目**：国家自然科学基金资助项目(60673190)；公安部科技创新资助项目(2010YYCXGDST066)；江苏大学高级人才科研启动基金资助项目(10JDG063)；江苏省高校自然科学基金资助项目(03KJD520075)

作者简介：刘一松(1966-)，男，副教授，博士，主要研究方向为计算机图形学、虚拟现实(liuyisong@ujs.edu.cn)；许祥虎(1987-)，男，硕士，主要研究方向为植物建模；曾兰玲(1973-)，女，副教授，博士，主要研究方向为计算机图形学；詹永照(1962-)，男，教授，博士，主要研究方向为计算机图形学、分布式计算等.

用鼠标描绘图像自动生成带有弯曲枝干的目标树三维模型,如图 1(c)(d)。最后通过添加树叶的交互操作生成三维树,如图 1(e)~(g)。

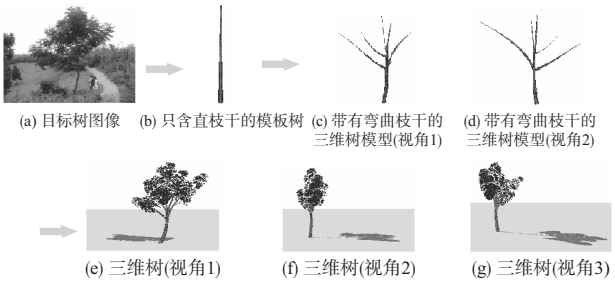


图1 建模过程

2 模板树建模算法

系统首先根据植物形态学原理应用分形方法生成一棵标准的模板树。根据之前的研究发现,子枝干的长度与其在父枝干中的位置到父枝干末端的长度满足黄金分割比,子枝干与父枝干的夹角与 90°满足黄金分割比[9]。另外,树的分枝和叶子的分布还满足叶序周。所谓叶序周就是从起点叶到终点叶之间的螺旋线绕茎周数,叶序周的表示方式为 a/b,其中 a 表示叶序周,b 表示在叶序周内出现的叶片数目。系统将黄金分割比和叶序周引用到树木枝干排序中。模板树就是基于植物形态学原理在不考虑任何外界环境下用分形方法生成的,用户可以通过交互接口修改模板树参数。另外,为了方便用户对不同图像的树木完成建模,系统根据树木枝干的形态结构,将其分成单轴分枝和合轴分枝类型。

2.1 枝干模型

因为模板树不考虑任何外界条件,所以枝干都用直干类型,不弯曲,如图 2 所示。系统将直干分成 DIVID 段,对于单轴类型的,末端半径为 0,对于合轴类型的,末端半径 $radius_{end}$ 根据经验公式为 $radius/\sqrt{n}$(n 表示 n 分轴类型,radius 表示枝干根部半径)。系统假设枝干粗细均匀变化,则第 i($1 \leqslant i \leqslant DIVID$)段的半径为$(DIVID - i + 1) \times (radius - radius_{end})/DIVID$,每段的长度为 length/DIVID(其中 length 为枝干长度),每段均用六棱柱模拟。

2.2 枝干生成算法

在一棵完整的模板树的设计中,每个枝干有其父枝干(除了根枝干)和子枝干(除了叶子)。子枝干与父枝干的分叉角度 angle(图 3)采用黄金分割比形态学规律,单轴用 34.4°,合轴用 56.6°。枝干的旋转角 rotation_angle(图 4)遵循叶序周规律。

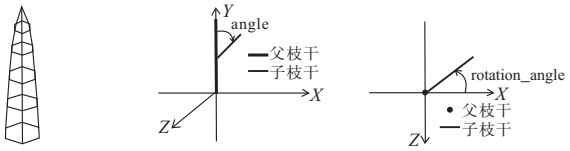


图2 直干类型枝干　图3 子枝干与父枝干的角度　图4 子枝干的旋转角

由用户设定树的初始参数,如树的长度、半径、层次数、叶序周、子枝干数、子枝干起始比例等。其中长度 length 和半径 radius 控制树的高度和粗细,层次数 depth 控制树的最大迭代次数,叶序周 a/b 控制枝干及叶子排序,子枝干数 childcount 控制每个枝干有多少个子枝干,子枝干起始比例 ratio 控制树的枝干疏密度(图 5)。

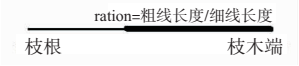


图5 子枝干的起始比例

枝干生成算法描述如下:

tree_generate(length, radius, depth, a, b, childcount, ratio)

Step1 如果 depth = = 0

{

generate_leaf(& length); //生成树叶,见 3.3 节

return;

}

Step2 循环计算第 i 个子枝干的数据($1 \leqslant i \leqslant$ childcount)

Step2.1 对第 i 个子枝干赋值

Step 2.1.1 在主枝干中的位置 pos 为(1 - ratio) + i × ratio/childcount;

Step 2.1.2 如果枝干是单轴类型,长度为 length × (1 - pos × 0.618),否则为合轴类型,长度为 length × lengthScale(lengthScale 是父枝干与子枝干的比例,在系统中取 0.8);

Step 2.1.3 如果枝干是单轴类型,半径为 radius × (1 - pos × 0.618),否则为合轴类型,半径为 pos × ($radius_{end}$(radius) + radius;

Step 2.1.4 如果枝干是单轴树,分叉角度 angle 为 34.4°,否则合轴类型,angle 取 5.6°;

Step 2.1.5 旋转角度 rotation_angle 为$(i \times \frac{a}{b} - \lfloor i \times \frac{a}{b} \rfloor) \times 360°$。

Step2.2 递归深度减 1 并与第 i 个枝干的数据一起作为参数递归调用 tree_generate。

按照上述算法用户可以初始设置 ratio = 0 可生成合轴模板树。为了避免多次生成数据,系统将生成和绘制分开,绘制算法描述如下:

tree_render(branchNo)

//假设枝干都有编号,branchNo 代表当前枝干编号

Step1 将父枝干段的模型矩阵压栈(模型矩阵为旋转和平移仿射变换的组合);

Step2 按第 branchNo 号枝干的 angle 和 rotation_angle 设置绕 Z 轴和 Y 轴旋转的模型矩阵:

$$\begin{bmatrix} c_2 & 0 & s_2 & 0 \\ 0 & 1 & 0 & 0 \\ -s_2 & 0 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中:c_1 为 cos(angle);s_1 为 sin(angle);c_2 为 cos(rotation_angle);s_2 为 sin(rotation_angle);

Step3 对第 i 段枝干段循环($1 \leqslant i \leqslant DIVID$)

Step3.1 如果存在子枝干在主枝干中的位置 pos 满足$(i-1)/DIVID < pos \leqslant i/DIVID$,依次调用 tree_render(childNo);

Step3.2 设置沿 Y 轴正方向平移 length/DIVID 个单位(枝干段的长度)的模型矩阵:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & length/DIVID \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step3.3 绘制第 i 段枝干段;

Step4 将模型矩阵出栈。

采用 OpenGL 库里的模型矩阵方法绘制的枝干连接处一般会有间隙,本文采用自己设计的向量方法对其进行了修正。其主要思路是使枝干段的上下表面都平行于 XOZ 平面,这样在枝干的衔接处就能达到无间隙的效果。向量方法描述如下:

a)在 XOZ 平面上以 O 点为圆心,1 为半径作一个正六边

形 G,其顶点为 $P_i(0\leqslant i\leqslant 5)$。

b)绘制一个枝干段,它的起点为 P(其下表面中心点,也是其上一个枝干段的上表面中心点),长度为 L,单位方向为 l,则它的上表面中心点 $P'=P+l\times L$,构成上下六边形表面的点为 $R\times P_i+P$ 和 $R'\times P_i+P'$(其中 $0\leqslant i\leqslant 5$,R 为该枝干段的下表面半径,R'为其上表面半径)。

c)分别对每个枝干段执行步骤 b)。

2.3　叶片生成算法

叶片相当于数据结构中树的叶节点,长在无分枝的细枝上,与生成子枝干不同的是要对细枝的长度加长,这样树可以吸收到更多的阳光。树叶用纹理融合技术,将带有 alpha 值的源 TGA 图像与一块边长为 leafScale 的正方形进行融合。算法描述如下:

generate_leaf (& length)//length 为细枝的长度
修改 length,将 length 增长 ε 倍,在系统中 $\varepsilon=1.2$;
将模型矩阵压入栈中;
打开融合;
设置纹理模式;
设置融合参数;
绘制一个正方形多边形,设置每个顶点的纹理坐标;
关闭融合;
模型矩阵出栈。

2.4　模板树实例

利用以上所建立的模板树模型,通过设置相关参数,可以很方便地生成各种模板树。表 1 为三种模板树实例的生成参数,图 6 为所生成三种模板树实例。

表 1　三种模板树的生成参数

模板树	类型	depth	ratio	length	radius	a	b	childcount
左	单轴	4	0.7	12	0.4	3	8	8
中	合轴	6	0	5	0.4	1	3	3
右	单轴	4	1	12	0.4	5	8	10

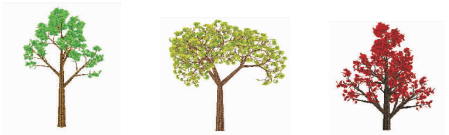


图6 三种模板树实例

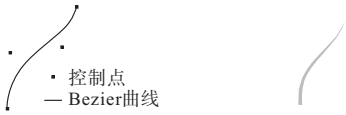
3　基于图像的交互过程

系统通过交互式技术提取图像中树木的枝干结构,并基于此结构调整模板树的分枝形态特征,生成与图像相似的三维树木模型。图像中反映的是现实世界中的树,模板树仅仅考虑在无任何外界影响下生长的树,而现实中存在很多外界因素(重力、光照等),所以枝干不可能是直干,用户可以通过系统的交互接口或者用鼠标描绘图像中的枝干生成弯曲枝干。

3.1　弯曲枝干的生成算法

在计算机图形中,Bézier 曲线被广泛地用来为平滑曲线建立模型。由于只需要少量控制点就可以形成各种弯曲曲线的造型,本文选择 Bezier 曲线模拟枝干弯曲,以实现通过系统的交互接口生成弯曲枝干,实验证明是可行的(图 7)。

系统通过鼠标描绘图像来生成新枝干。在描绘的过程中所记录的点不能都作为控制点,因为点太多会导致处理速度慢,另外描绘过程中可能由于手的抖动会形成锯齿,所以必须对所记录的点作些简单的处理。假设整个描绘过程中记录了



(a) 控制点控制曲线弯曲　(b) 对应控制点生成弯曲枝干

图7 Bezier曲线模拟弯曲枝干

m 个点,从这 m 个点中选取 n 个点作为控制枝干弯曲的曲线控制点,其中第 i 个点为$\lfloor \frac{m}{n-1}\times i\rfloor(0\leqslant i<n)$,通过这样的简单处理能够得到较好的效果(图 8)。

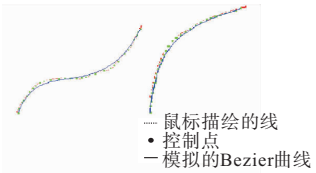


图8 描绘枝干曲线获得控制点(n=30时情况)

在绘制弯曲枝干过程中,不是像直枝干那样仅在开始绘制时设置一次绕 X 和 Z 轴旋转的模型矩阵,而必须在绘制每个枝干段时都设置一次。枝干段之间的夹角可根据式(1)得到。式(1)中 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 分别为第 i 个和第 $i+1$ 个枝干段的向量($0\leqslant i\leqslant$(DIVID − 1)),它们都在 XOY 平面内(当 $i=0$ 时取 $\boldsymbol{a}=(0,1,0)$),且 $\boldsymbol{e}_z=(0,0,1)$。根据枝干段之间的夹角就可以绘制弯曲的枝干。

$$\begin{cases}\cos^{-1}\left(\dfrac{\boldsymbol{a}\cdot\boldsymbol{b}}{|\boldsymbol{a}|\times|\boldsymbol{a}|}\right)\times\dfrac{180^\circ}{\pi} & (\boldsymbol{b}\times\boldsymbol{a})\cdot\boldsymbol{e}_z\\ -\cos^{-1}\left(\dfrac{\boldsymbol{a}\cdot\boldsymbol{b}}{|\boldsymbol{a}|\times|\boldsymbol{a}|}\right)\times\dfrac{180^\circ}{\pi} & \text{其他}\end{cases}\tag{1}$$

3.2　添加枝干的算法

根据图像描绘新枝干时必须为其选择父枝干,系统选择距离新枝干的根部最近的已绘制的枝干作为其父枝干。如果新枝干与几个枝干具有相同的最短距离时,则从中随机选择一个,如果是第一次描绘,则以新枝干作为根枝干,如图 9 所示。

根据式(2),可以求出空间中的一点 P(新枝干的下表面的中心点)到线段 AB(A 和 B 分别是父枝干段上下表面的中心点)的最短距离。要计算新枝干到一个已存在父枝干的最短距离,可以通过式(2)对父枝干的每个枝干段求最短距离,从中选择最小值作为新枝干到父枝干的最短距离。

$$\begin{cases}|AP| & AP\cdot AB\leqslant 0\\ |BP| & BP\cdot BA\leqslant 0\\ \sqrt{|AP|^2-\left(AP\cdot\dfrac{AB}{|AB|}\right)^2} & \text{其他}\end{cases}\tag{2}$$

对于式(2)中的第一种和第二种情况,由于能恰好找到父枝干上的一个控制点作为插入点,所以能够准确找到插入位置。对于式(2)中的第三种情况,需要增加父枝干上的控制点以确定插入的位置,增加控制点的方法是通过在 A 和 B 点中间插入 C 点,$C=\lambda\times A+(1-\lambda)\times B$,其中 λ 为 P 在线段 AB 上的投影点与 B 点的距离占线段 AB 长度的比例。

通过图像描绘的枝干只是三维空间到平面的投影,借助叶序周规律将其再变换到三维空间。如图 10 所示,假设平面 α(XOY 平面)和 β(旋转后的平面)的夹角为 θ(平面的旋转角度,也就是枝干的旋转角,$0\leqslant\theta<\pi/2$),α 和 β 平面交线为 l_1(平行于 Y 轴),平面 β 内存在一条有向线段 l_2(枝干旋转后在平面 β 中的线段),l_2 在平面 α 内的投影为有向线段 l_3(枝干在 XOY 平面中的手绘有向线段),则 θ 与 $|l_2|$ 满足式(3)。也就是说,式(3)中 l_1 和 l_3 是已知量,根据 $|l_2|$ 可以算出 θ,同样根据 θ

可以算出 $|l_2|$。

$$|l_2| = \sqrt{\frac{\left|\frac{l_1}{|l_1|}\times l_3\right|^2}{\cos^2\theta} + \left(|l_3|\cdot\frac{l_1}{|l_1|}\right)^2} \tag{3}$$

图9 插入新枝干时父枝干的选择

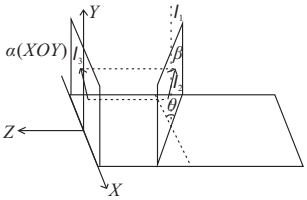


图10 平面β到平面α的投影

将 XOZ 平面以 X 和 Z 轴分为区域 0 ~ 3 的四个区域,每个区域中旋转角范围依次为[0,90)、[90,180)、[180,270)和[270,360)。根据叶序周规律计算所有可能的旋转角,并将这些旋转角分配到四个区域内。

如果描绘的枝干在 XOZ 平面的投影的 X 坐标为负值,就在区域 1 和 2 内选择旋转角,否则在区域 0 和 3 内选择。为了吸收更好的光照,假设枝干长度最大化,枝干长度的上限是其父枝干的 ε 倍(可根据实际情况自行修改,在实验时单轴类型取 0.6,合轴类型取 0.8)。根据 ε 和式(3)计算推荐角度,如果存在旋转角(根据叶序周规律生成)与推荐角度相等就直接选取该旋转角,否则就从旋转角中选择最接近推荐角度的两个角度(分别大于和小于推荐角度),再从这两个角度中选择一个其枝干长度变化率 $|(l_2-\varepsilon\times l_3)/\varepsilon\times l_3|$ 较小的旋转角。确定了旋转角 θ,根据式(3),可最终确定 l_2。从而实现了通过图像描绘的枝干从平面到三维空间的变换。

为了使生成的模型更加符合目标树图像,系统实现了对模型中的枝干(子树)进行添加和删除的操作。用户可以在树中一枝干的某个枝干段处添加新枝干及其子枝干,从而生成合并树,如图 11 所示。用户还可以删除树中某个枝干及其子枝干。

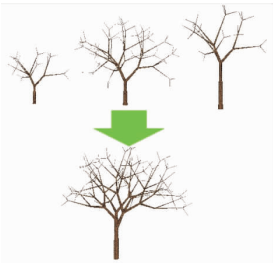


图 11　合并树

3.3　局部参数调整

用模板或者描绘图像生成的枝干很难完全满足要求,所以局部参数调整是必需的。系统设置了对某一枝干进行调整的交互接口(图 12),通过修改参数可对所选择的枝干进行相应的调整。

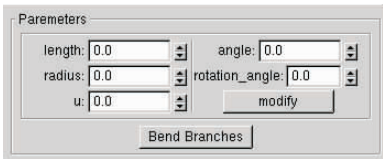


图 12　局部参数调整面板

图 12 中,length 为选择枝干的长度(length > 0);radius 为半径,代表枝干粗细(radius > 0);u 代表被选择的枝干在父枝干中的位置($0 \leqslant u \leqslant 1$);angle($0° < \text{angle} < 180°$,见图 3)和 rotation_angle($0° \leqslant \text{rotation_angle} < 360°$,见图 4),分别反映父枝干与子枝干的夹角以及旋转角。

3.4　叶子生成

叶子是植物建模的重要部分,为了减少建模时间,系统建立了包含多种叶片的叶子库,用户可以从模型库选择叶子类型以及通过叶序周改变叶子的排列方式,控制面板如图 13 所示。通过 leaf_type 可以选择叶子种类,系统中设置了单轴、合轴和柳枝,通过子枝干起始比例 ratio 可以设置树叶的生长范围,通过 leaf_count 可以设置在 ratio 范围内的叶子数。对于柳条状的末梢及树叶,只需更改枝条的 Bézier 曲线控制点,系统采用的是(0,0,0),(0.1,-0.05,0),(0.1,-0.4,0),(0.1,-0.8,0),如图 14 所示。

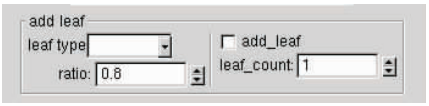


图 13　叶子添加控制面板

(a) 在鼠标选定位置增加单个叶子簇　(b) 以叶序周为3/8增加三个叶子簇　(c) 增加一个柳条状的末梢及树叶

图14 叶子生成

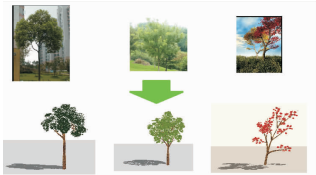
此外,为了增加模型的真实感,系统中添加叶子是带有细枝的,不是单纯地添加叶片,通过控制面板(图 13)设置参数,然后与枝干生成算法类似产生一棵树作为叶子簇添加到枝干中。

4　结束语

本文针对图形和图像方法的不足,提出了基于单幅图像交互生成三维树木的算法。系统定义了单轴和合轴两种基本类型,用分形方法生成标准的模板树,用交互接口或者鼠标描绘图像中的枝干形成弯曲枝干,通过叶序周规律将图像中的二维树变换成三维,并建立了多种叶子库以供选择。通过与文献[7]中所提供的结果(图 15)比较可以发现,采用本文提出的方法可以使建模结果更加符合参考图像,并遵循植物的生长规律。比较结果如图 15 和 16 所示。

(a) 参考图像　(b) CPFG系统生成结果　(c)XFrog 系统生成结果　(d) Okabe 系统生成结果

图 15　对比结果



(a)　(b)　(c)

图 16　结果图(上一排为真实照片,下一排为系统生成的结果)

以上实验是在 VC++6.0 平台用 OpenGL 库,在配置为 AMD Turion64 MT-30 1.6 GHz CPU,768 MB 内存,NVIDIA GeForce Go 6200 256 MB 显卡的机器上开发的。图 16(a)和(b)中的照片是在校园中自拍的,(c)中的照片是从 (下转第 2378 页)