一种基于八叉剖分的近似曲率的边折叠简化算法*

张 果, 刘旭敏
(首都师范大学 信息工程学院, 北京 100048)

摘 要: 为了提高三角网格模型简化的速度,满足实时显示的要求,并且有效地克服边折叠简化算法在低分辨率的状态下易丢失模型重要几何特征的问题,提出了一种基于八叉剖分的近似曲率的边折叠简化算法。采用八叉树结构自适应地分割网格模型空间,同时在各个区域中采用近似曲率的边折叠算法并行地进行边折叠操作。实验证明,该算法取得了不错的效果。

关键词: 网格简化; 边折叠; 近似曲率; 形状特征; 八叉树
中图分类号: TP394.41 **文献标志码**: A **文章编号**: 1001-3695(2010)05-1955-04
doi:10.3969/j.issn.1001-3695.2010.05.101

Similar curvature edge collapse simplification based on octree

ZHANG Guo, LIU Xu-min
(*College of Information Engineering*, *Capital Normal University*, *Beijing* 100048, *China*)

Abstract: To improve the speed of triangle mesh simplification, achieve the requirement of real-time display, and reserve more important shape features in the low-level model, the paper presented an improved triangle mesh simplification algorithm. It used octree structure to divide the triangle mesh model into different areas, and then used similar curvature edge-collapse simplification algorithm to simplify each area in parallel. The experimental results show that the improved algorithm obtains a better effect.

Key words: mesh simplification; edge-collapse; similar curvature; shape features; octree

0 引言

在计算机图形学和几何造型中,物体表面常用多边形网格模型尤其是三角网格模型描述,这已经是几何造型中最直接最简单的方法。但是,直接由高科技设备创建的三角网格模型都很复杂,表面分割精细,数据量十分庞大,给物体实时显示带来了很大的困难。因而,三角网格模型简化技术的研究已经成为该领域的研究热点并出现了许多研究成果。国内外研究成果大致可以归结为顶点聚类法、区域合并法、顶点删除法、边折叠法、三角形折叠简化方法和小波分解法等。由于边折叠法具有简单、方便、折叠误差小的特点,它成为一种常用的几何模型简化方法。

网格模型简化的目标是在尽量保持模型逼真度的前提下,大幅度地减少三角形面片的数目。所以,一个好的模型简化方法应具有快速的特点,这是几何模型实时显示的基本要求,也应具有保持原始模型特征的特点。

Garland 等人[1]于 1997 年提出基于二次误差测度的边折叠简化算法(简称 QEM 算法),以顶点到相关三角平面距离的平方和作为误差测度,算法速度快,而且能生成高质量的简化模型。但是 QEM 算法的误差测度标准过于单一,所产生的简化网格都是均匀的,在低分辨率的状态下往往丢失模型重要几何特征,从而导致视觉上的失真。刘晓利等人[2]于 2005 年在 Garland 算法基础上引入尖特征度的概念,在不增加时间复杂度的情况下,较大程度地保持了简化模型的外观。但是该算法简化后模型的几何误差值较大,同时由于对表面变化反映不够细致和全面,大规模简化后仍然会丢失很多重要的几何信息;而且采用这种方法,每一个顶点、边和三角形都要建立数据结构,对于无法调入内存的网格数据来说,这是不可能的。因此仅用这种简化方法是无法直接对超大规模网格模型进行简化的,在那些对物体实时显示要求比较高的场合,仅用该算法也是不够的。

鉴于研究中存在的这些问题,提出了一种基于近似曲率的边折叠简化算法来维持模型的几何特征;采用八叉树的结构来自适应地分割网格模型空间,不仅克服了无法直接对超大规模网格模型进行简化的问题,采用分块的原理对每个分割后的区域单独调入内存进行简化,而且对于可以直接调入内存的模型,对各个分割后的区域并行地进行边折叠操作,以此来提高三角网格模型简化的速度,达到实时显示的要求。实验表明该简化方法取得了不错的效果。

1 相关知识

边折叠简化算法首先计算模型中所有边在折叠时的代价,然

收稿日期: 2009-09-07; **修回日期**: 2009-10-26 **基金项目**: 国家自然科学基金资助项目(60873006);北京市教育委员会科技发展计划重点资助项目(KZ200710028014);北京市自然科学基金资助项目(4082009)

作者简介:张果(1985-),女,河南洛阳人,硕士研究生,主要研究方向为计算机图形学(zg226@163.com);刘旭敏(1956-),女,辽宁锦州人,教授,主要研究方向为计算机图形学、数据挖掘.

后按折叠代价从小到大进行排序。每次从队头取出一条边进行折叠,当对该边折叠之后,相关联的边都从优先级队列中删除,重新计算其折叠的代价,并且将满足折叠条件的边按优先级顺序插入队列,继续进行边折叠,直到没有边满足折叠条件为止。图 1 为边折叠前后的比较图。

Garland 算法是基于二次误差测度的边折叠算法,以顶点到相关三角平面距离的平方和作为误差测度,算法速度快,简化质量较高,成为一种经典的边折叠算法。

该方法对每个顶点 $v=[v_x \quad v_y \quad v_z]^{\mathrm{T}}$ 定义其误差为 v 与其相关联平面集合 planes(v)的距离平方和。这个误差测度可以写成二次型形式:

$$\Delta(v)=\sum_{p\in \mathrm{plane}(v)}(p^{\mathrm{T}}v)^2=\sum_{p\in \mathrm{plane}(v)}v^{\mathrm{T}}(pp^{\mathrm{T}})v=v^{\mathrm{T}}\Big(\sum_{p\in \mathrm{plane}(v)}\boldsymbol{K}_p\Big)v \tag{1}$$

其中:p 是方程 $ax+by+cz+d=0(a^2+b^2+c^2=1)$定义的与 v 相关联三角形所在平面;$\boldsymbol{K}_p$ 是平面 p 的基本误差二次型,即

$$\boldsymbol{K}_p=pp^{\mathrm{T}}=\begin{bmatrix}a^2 & ab & ac & ad\\ ab & b^2 & bc & bd\\ ac & bc & c^2 & cd\\ ad & bd & cd & d^2\end{bmatrix} \tag{2}$$

把 $\boldsymbol{Q}_v=\sum_{p\in \mathrm{plane}(v)}\boldsymbol{K}_p$ 称为顶点 v 的二次误差测度矩阵。当进行边折叠$(v_i,v_j)\to v'$时,边折叠代价为

$$\Delta(v')=v'^{\mathrm{T}}(\boldsymbol{Q}_i+\boldsymbol{Q}_j)v' \tag{3}$$

折叠后 v'的二次误差测度矩阵为

$$\boldsymbol{Q}_{v'}=\boldsymbol{Q}_i+\boldsymbol{Q}_j=\begin{bmatrix}q_{11} & q_{12} & q_{13} & q_{14}\\ q_{12} & q_{22} & q_{23} & q_{24}\\ q_{13} & q_{23} & q_{33} & q_{34}\\ q_{14} & q_{24} & q_{34} & q_{44}\end{bmatrix} \tag{4}$$

这种方法速度快,简化后模型表面误差均值较低。但由于只考虑距离的度量,网格顶点分布均匀,在大规模简化后,模型表面较为尖锐的棱角等重要几何特征丢失。

为了克服网格分布过于均匀,不能突出模型重要特征的缺点,刘晓利提出尖特征度(sharp degree)的概念。首先将相关联的两个面的二面角(两个面的外法向夹角)大于给定阈值 θ 的边定义为尖特征边,则原始顶点的尖特征度定义为与该顶点关联的所有尖特征边的个数。加入尖特征度的二次误差测度表示为

$$\Delta^L(v)=\Delta(v)+c_{\mathrm{sharp}}\cdot\ \mathrm{sharp}=v^{\mathrm{T}}\Big(\sum_{p\in \mathrm{plane}(v)}\boldsymbol{K}_p\Big)v+c_{\mathrm{sharp}}\cdot\ \mathrm{sharp}=v^{\mathrm{T}}\Big(\sum_{p\in \mathrm{planes}(v)}\boldsymbol{K}_p+\boldsymbol{K}_{\mathrm{sharp}}\Big)v \tag{5}$$

其中:$\boldsymbol{K}_{\mathrm{sharp}}=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & c_{\mathrm{sharp}}\cdot\ \mathrm{sharp}\end{bmatrix}$;sharp 是顶点 v 的尖特征度;c_{sharp}是根据经验设定的一个惩罚系数,刘晓利将其大概取为 Garland 误差均值的 1/10。顶点 v 的二次误差测度矩阵为 $\boldsymbol{Q}_v=\sum_{p\in \mathrm{planes}(v)}\boldsymbol{K}_p+\boldsymbol{K}_{\mathrm{sharp}}$。由于尖特征度反映了局部表面变化,该方法能够在很大程度上保持模型尖锐特征。然而 c_{sharp}的设定需要事先知道 Garland 原始二次误差测度每次简化的误差均值,同时阈值 θ 也是经验值,这给实际应用带来较大的困难。另外,由于在误差测度中直接加入惩罚项,使得模型原始顶点的初始误差不为 0。同时该误差测度不能反映小于 θ 二面角的影响,大于 θ 的二面角对误差测度贡献相同,而且顶点尖特征度是小于等于顶点度数的整数值,对表面变化反映不够充分。

2　一种基于近似曲率的边折叠简化算法

为了弥补刘晓利算法的不足,提出了近似曲率这个概念来反映三角网格模型中顶点处的弯曲程度,并将其作为权值嵌入到二次误差测度中,以此来改变边折叠的顺序,同时并不改变二次型的基本性质以及初始误差值。

2.1　顶点的近似曲率的计算

在三角网格模型中,由于平滑区域中三角形的顶点处的弯曲程度一般都较小,而具有尖锐特征的区域中,其顶点处的弯曲程度一般都较大,所以用顶点的近似曲率作为这个自适应权值的主要依据,能更好地保持模型的几何特征。

由于被简化的多边形表面是由一系列的三角形构成的,它不是二阶可微的曲面图形,从理论上来说,它不存在曲率,但它可以看做是光滑表面的分段近似[3]。

改进的算法用式(6)来代表顶点处的近似曲率,它表示模型中顶点处的弯曲程度:

$$K_s=\frac{\sum_{v_i\in \mathrm{neiverts}(v_s)}c_{si}}{m_i} \tag{6}$$

其中:c_{si}是顶点 v_s 和 v_i 的法向量夹角的余弦值;v_i 是与 v_s 相关的顶点即 $v_i\in \mathrm{neiverts}(v_s)$;$m_i$ 是与顶点 v_s 相关的顶点个数;K_s 表示了 v_s 与其相关顶点的法向量之间的夹角余弦值的平均值。

$$c_{si}=\cos(\boldsymbol{n}_s,\boldsymbol{n}_i)=\frac{\boldsymbol{n}_s\cdot\boldsymbol{n}_i}{|\boldsymbol{n}_s||\boldsymbol{n}_i|} \tag{7}$$

其中:$\boldsymbol{n}_s$ 是顶点 v_s 的法向量,$\boldsymbol{n}_i$ 是顶点 v_i 的法向量。由公式可看出,c_{si}的值越大,这两个顶点法向量之间的夹角就越小,顶点处的弯曲程度就越大,近似曲率就越大;c_{si}的值越小,这两个顶点法向量之间的夹角就越大,顶点处的弯曲程度就越小,近似曲率就越小。这里顶点法向量用与其相关的三角平面的法向量的和来表示。

K 值的大小反映了顶点 v_s 邻域内的近似曲率的大小。K 值越大,顶点 v_s 的邻域内的近似曲率越大,模型表面的几何变化越明显;K 值越小,顶点 v_s 邻域内的近似曲率越小,模型表面的几何变化就越不明显。

2.2　算法的改进

为了使得二次误差测度在能够度量距离偏差的情况下,能够反映模型局部表面几何变化,用顶点的近似曲率作为自适应权值嵌入到二次误差测度中,由此在式(3)的基础上得到式(8):

$$\Delta(v')=v'^{\mathrm{T}}(\boldsymbol{Q}_s+\boldsymbol{Q}_t)v'-\frac{1}{2}(K_s+K_T)=v'^{\mathrm{T}}(\boldsymbol{Q}_s+\boldsymbol{Q}_t)v'-\frac{1}{2}\Big(\frac{\sum_{v_i\in v_s.\,\mathrm{neiverts}}c_{si}}{m_i}+\frac{\sum_{v_j\in v_t.\,\mathrm{neiverts}}c_{tj}}{m_j}\Big) \tag{8}$$

其中:前半部分 $v'^{\mathrm{T}}(\boldsymbol{Q}_s+\boldsymbol{Q}_t)v'$表示顶点 v'到与其相关三角平面的距离平方和;括号中的两项分别给出了顶点 v_s 和 v_t 与其相关顶点的法向量之间的夹角余弦值的平均值,这个值的大小反映了顶点 v_s、v_t 的邻域内的近似曲率的大小。因此这两项的平均值越大,说明在边(v_s,v_t)的邻域内网格表面的弯曲程度越大,几何形状变化也越明显,于是这条边的删除就更应该置后。

另外,由于所嵌入的自适应权值总是在[-1,1]中变化,为防止它们在边长很大的情况下失去作用,对上式进一步修改如下:

$$\Delta(v') = v'^{\mathrm{T}}(Q_s + Q_t)[1 - \frac{1}{2}(K_s + K_T)]v' =$$

$$v'^{\mathrm{T}}(Q_s + Q_t)[1 - \frac{1}{2}(\frac{\sum_{v_i \in v_s.\,\mathrm{neiverts}} c_{si}}{m_i} + \frac{\sum_{v_j \in v_t.\,\mathrm{neiverts}} c_{tj}}{m_j})]v' \qquad (9)$$

有了以上的边权定义,简化时网格的几何特征就能够得到较好的保持。

3　采用八叉树结构进行自适应区域划分

对于超大规模网格模型或者对物体实时显示要求比较高的场合,可以采用将原始模型分解成多个区域的方法。对于超大规模网格模型,只要对分解后的每个区域单独调入内存进行简化操作即可,在这里主要考虑对实时显示要求比较高的场合。首先采用八叉树结构来自适应地分割网格模型空间;然后对每个分割后的区域采用上述的边折叠简化算法并行地进行简化,在简化过程中保持这些区域的边界;最后将已经简化的区域无缝地连接起来形成一个简化模型。此方法有效地提高了三角网格模型简化的速度。

物体的八叉树表示是一种层次数据结构。该算法将含有整个场景的空间立方体作为根节点,按三个方向的中间剖面将其分割成八个子立方体网格,组织成一棵八叉树。若某一子空间网格中所含面片数大于给定阈值则继续剖分,直到满足规定的面片数为止。

采用八叉树表示物体最大的缺点是它占用内存很多,这是因为每一体元都是立方体,且体元各表面分别与三个坐标平面平行。为了减小八叉树表示所需的空间存储量,提出了线性八叉树的表示方法。所谓线性八叉树即用一个可变长度的一维数组来存储一棵八叉树。

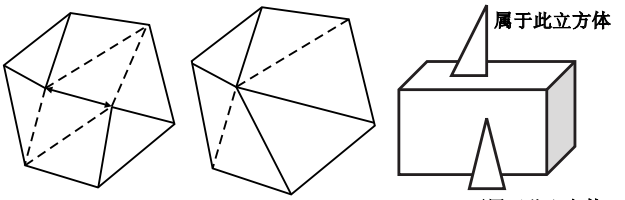
3.1　八叉树数据结构

本文采用八叉树结构对三角网格模型进行自适应划分的目的,就是把原始网格模型划分为不同的区域,而每个区域又是一个单独的网格模型,然后再对每个区域并行地进行边折叠操作。为了以后网格简化过程中可以方便地调用每个区域网格模型,本文的结构体中有一个表示网格模型的变量,以此来表示区域划分以后的子网格模型。

```
vector〈OcTreeNode〉 _treelist;// 线性八叉树链表
typedef struct OcTreeNode //八叉树节点
{
    //存储三角面片的 vector 容器
    vector〈triangle〉NodeTri;
    //存储顶点的 vector 容器
    vector〈vertex〉NodeVert;
    int numOfTriangle;//三角形个数
    int vertexnum;//顶点个数
    //孩子节点指针
    struct OcTreeNode * pChild[8];
     mesh * _mesh;//子网格模型
}OCTREENODE;
```

3.2　三角面片与立方体网格包含关系的确定

由上面论述知道三角面片容器建立的关键是确定三角面片与子空间网格的包含关系。这里以文献[4] 给出的不等式方法为基础。如图2 所示,首先确定子立方体中顶点的最大值和最小值:$V_{\min}(x_{\min}, y_{\min}, z_{\min})$,$V_{\max}(x_{\max}, y_{\max}, z_{\max})$。对于给定的三角面片 $T_1T_2T_3$ 作这样的判断:如果 $V_{\min} < T_i < V_{\max}, V_{\min} < T_j < V_{\max}, T_k > V_{\max}$ 或 $T_k < V_{\min}$,则认为此三角面片位于该子立方体内,并将其加入对应的三角形容器内;如果 $V_{\min} < T_i < V_{\max}, T_j > V_{\max}, T_k > V_{\max}$ 或 $T_k < V_{\min}$,则认为此三角面片不在该立方体内,不对其进行处理。

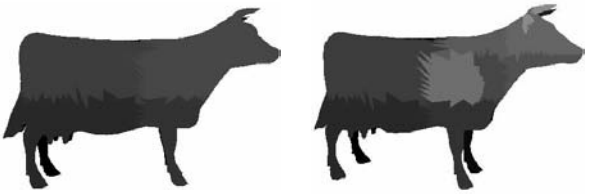


(a)边折叠前　(b)边折叠后

图1　边折叠前后的比较

图2　三角面片与立方体网格包含关系

图3(a)为牛模型经过八叉剖分以后的效果图。如果第一次八叉剖分以后每个子立方体三角面片的数目大于对应的阈值,则对该子立方体网格继续剖分。图3(b)为对牛模型的第一个子立方体继续进行网格剖分的效果图。



(a)第一次八叉剖分效果图　(b)第一子立方体继续八叉剖分效果图

图3　牛模型八叉剖分后效果图

4　算法描述

下面给出了本文算法的整体描述。与原始的边折叠算法相比,其主要在边折叠操作前增加了空间八叉剖分,也就是区域划分处理[5],然后再在每个区域采用基于近似曲率的边折叠算法并行地进行简化。这不仅减少了三角网格简化的时间,达到实时显示的要求,而且维持了原始三角网格模型的几何特征。

a)读取文件;

b)对载入的文件采用八叉树结构进行分割,将三角网格模型划分为八个子立方体;

c)如果每个子立方体中三角面片的数目小于对应的阈值则停止剖分,否则对该子立方体网格继续剖分,直到每个子空间所含面片数小于给定阈值;

d)每个子立方体对应一个独立的子网格模型,在每个子网格模型中分别计算各边的二次误差矩阵,并计算折叠代价及折叠后新顶点的位置;

e)根据折叠代价分别建立各个区域折叠代价的堆栈;

f)并行地在每个区域按折叠代价的大小分别进行一次边折叠操作;

g)若三角形的数目已经达到简化要求,结束,得到简化模型;否则,局部修改折叠后的二次误差矩阵、折叠代价及新顶点的位置信息,并转步骤e)。

5　区域的无缝连接

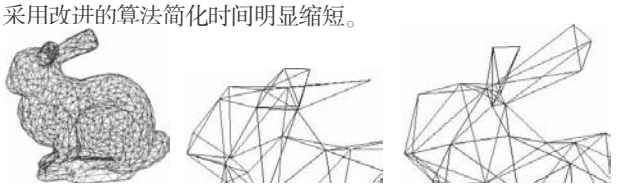
按上述的算法可知,采用八叉剖分的方法来对三角网格模型进行简化,是并行地对剖分后的每个区域进行三角网格简化,然后将已经简化的区域无缝地连接起来形成一个简化模型。因此如何将已经简化的区域无缝地连接起来成为一个重点研究的问题。本

文采用的方法是对每一个区域分别找到其边界边,为每个边界边生成一个虚拟平面,这个平面通过该边界边并垂直于边界边所在的三角平面。给该虚拟平面规定一个重要度作为该平面基本误差二次型的权值,然后累加到该边界边端点初始的二次误差测度矩阵中。由于虚拟平面重要度比较大,边界边的折叠代价相对比较高,边界可以得到很好的保持。但是,考虑到多次边折叠操作之后,边界边的边折叠代价相对比较高,致使边界处的三角片过密,影响效果,因此设置一个阈值。当边折叠的次数大于这个阈值时,停止对模型的继续简化,保存这个简化模型,然后重新对这个简化模型进行八叉剖分,从而改变每个子网格模型的边界边,更好地实现区域的无缝连接。

6　结果分析

为了验证算法的效果,用 Visual C ++ 6.0 实现了该算法,并用 OpenGL 完成图形的绘制。

图 4 ~6 分别为各种原始模型采用改进的算法与原始算法所得到的简化模型的对比图。很明显,在图 4 中采用改进的算法,当模型简化到 134 个三角面片时,兔子耳朵的几何特征得到了很好的保持,而采用刘晓利算法,兔子耳朵的几何特征很大一部分已经被简化;在图 5 中,采用改进的算法,牛模型的形状特征得到了很好的维持,尤其是牛的角,几何特征保持得很不错,而采用刘晓利算法,牛模型角的几何特征已经不太尖锐;在图 6 中,当蚂蚁模型简化 35% 的时候,采用刘晓利算法,蚂蚁的触须已经开始断裂,而采用改进的算法却得到了不错的效果。从表 1 和 2 还可以看出,采用改进的算法简化时间明显缩短。



(a)原始模型1850　(b)刘晓利算法134　(c)改进算法134

图4　兔子模型的部分特征对比

(a)原始模型5804　(b)刘晓利算法270　(c)改进算法270

图5　牛模型的特征对比

(a)原始模型　(b)刘晓利算法(简化35%)　(c)改进算法(简化35%)

图6　蚂蚁模型的特征对比

表 1　牛模型八叉剖分前后简化时间的对比表　s

	简化 10%	简化 30%	简化 50%	简化 80%
未剖分	8.905	9.031	9.172	9.264
剖分	1.203	1.318	1.337	1.347

表 2　兔子模型八叉剖分前后简化时间的对比表　s

	简化 10%	简化 30%	简化 50%	简化 80%
未剖分	4.218	4.233	4.249	4. 312
剖分	0.651	0.653	0.655	0.663

通过实验结果可以看出,采用改进的算法不仅模型的细节特征得到了很好的维持,而且简化速度也得到了大大提高。

7　结束语

为了达到网格模型简化的目标,在尽量保持模型逼真度的前提下,所用的时间应尽可能地少,以此来达到实时显示的目的。文章首先采用八叉树结构来自适应地分割网格模型空间,然后在各个子网格模型并行地进行边折叠操作,以此来提高三角网格模型简化的速度;同时为了能够更好地维持模型的几何特征,引入了近似曲率的概念,把它作为自适应权值嵌入到二次误差测度中,有效地克服了刘晓利算法简化模型几何误差值较大、对表面变化反映不够充分的问题。实验证明,该简化方法取得了不错的效果。

参考文献:

[1] KHO Y, GARLAND M. User-guided simplification[C]//Proc of ACM Symposium on Interactive 3D Graphics. 2003:123-126.

[2] 刘晓利,刘则毅,高朋东,等. 基于尖特征度的边折叠简化算法[J]. 软件学报,2005,16(5):669-675.

[3] 郭震宇. 三角网格模型的简化技术及多细节层次模型[D]. 大连:大连理工大学,2006.

[4] 彭群生,鲍虎军,金小刚. 计算机真实感图形的算法基础[M]. 北京:科学出版社,2002.

[5] 全红艳,张田文,董宇欣. 一种基于区域分割的几何模型简化方法[J]. 计算机学报,2006,29(10):1834-1842.

(上接第 1954 页)

[2] SAMAL A, IYENGAR P A. Automatic recognition and analysis of human faces and facial expressions:a survey[J]. Pattern Recognition,1992,25(1):65-77.

[3] VALENTIN D, ABDI H, O' TOOLE A J. Connectionist model of face processing:a survey [J]. Pattern Recognition,1994,27(9):1209-1230.

[4] HONG Z. Algebraic feature extraction of image for recognition[J]. Pattern Recognition,1991,24(3):211-219.

[5] LIU K, YANG J Y, WANG H F, *et al*. A generalization optimal set of discriminate vectors[J]. Pattern Recognition,1992,25(7):731-739.

[6] 王蕴红,谭铁牛,朱勇. 基于奇异值分解和数据融合的脸像鉴别[J]. 计算机学报,2000,23(6):649-653.

[7] 甘俊英, 张有为. 一种基于奇异值特征的神经网络人脸识别新途径[J]. 电子学报,2004,32(1):170-173.

[8] TIAN Y, TAN T, WANG Y H, *et al*. Do singular values contain adequate information for face recognition[J]. Pattern Recognition,2003,36(6):649-655.

[9] 杜干,朱雯君. 基于局部奇异值分解和模糊决策的人脸识别方法[J]. 中国图象图形学报,2006,11(10):1457-1459.

[10] 高全学,梁彦,潘泉,等. SVD 用于人脸识别存在的问题及解决方法[J]. 中国图象图形学报,2006,11(12):1785-1791.

[11] 邓聚龙. 灰色系统理论教程[M]. 武汉: 武汉理工大学出版社, 1990.

[12] 王唯一,任以君,张明,等. 彩色序列图像的人脸分割与识别技术研究[J]. 微计算机信息,2008,9(3):278-279.

[13] 王聃,贾云伟,林福严. 人脸识别系统中的特征提取[J]. 微计算机信息,2005,7(3):224-225.

[14] FENG G C, YUEN P C. Multi-cues eye detection on gray intensity image [J]. Pattern Recognition,2001,34(10):1033-1046.