基于自适应八叉树分割点云的表面模型重建

杨 客 张志毅* 董 艳
(西北农林科技大学信息工程学院 陕西 杨凌 712100)

摘 要 传统的三角网生长法进行点云数据表面模型重建时,搜索第三点耗时太长,导致重建效率很低。采用自适应八叉树划分算法将点云数据分割成相互覆盖的子域,在每个子域内进行三角网格重建,避免网格拼接的过程;采用最大角最小化原则进行三角网格优化;并运用三角面片定向的方法进行网格法向量一致化处理。实验结果表明,该方法极大地提高了表面模型重建的效率,形成的网格质量也很好,能够较好地体现模型的细节特征,鲁棒性好。

关键词 点云 表面模型重建 自适应八叉树 三角网生长法

中图分类号 TP37 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2013.06.023

SURFACE MODEL RECONSTRUCTION BASED ON POINT CLOUD SUBDIVISION WITH ADAPTIVE OCTREE

Yang Ke Zhang Zhiyi* Dong Yan
(*College of Information Engineering, Northwest A&F University, Yangling 712100, Shaanxi, China*)

Abstract In surface model reconstruction with regard to point cloud data, traditional triangulation growth method has very low efficiency because it takes too long time in searching the third vertex. The adaptive octree subdivision algorithm is used in this paper. Point cloud data are divided into subdomains covering each other, and the triangular grids are reconstructed in every subdomain, thus the process of the grid stitching is avoided. The produced triangular grids are optimised using the principle of minimising the maximum angle. A triangular facet orientation method is used for uniformisation of the normal vectors of grids. Experimental results show that the efficiency of the surface model reconstruction is greatly improved by using this method, and the quality of the triangular grids produced are very good as well, the detail features of the model are better reflected, and the algorithm is robust.

Keywords Point cloud Surface model reconstruction Adaptive octree Triangulation growth method

0 引 言

近年来,随着基于视觉和多视图像三维模型获取技术的发展,基于点云的曲面重建技术广泛应用于逆向工程、计算机视觉、虚拟现实等领域。作为常用的曲面重建方法之一的三角剖分技术,既是热点又是难点问题。一般情况下,在众多三角剖分算法中,Delaunay 三角剖分构建的网格质量是最优的[1,2]。通常 Delaunay 三角剖分算法[3]分为:三角网生长法,逐点插入法和分割-合并算法。Green、Sibson[4]于 1978 年提出了生长算法,实现了平面内点集的三角剖分;随后 Bowyer、Watson 等人[5,6]将平面三角剖分推广到空间;直接的三角网生长法由于搜索第 3 点所消耗的时间过长,许多学者对该算法进行了改进。董洪伟[7]于 2005 年在基于增量扩散法的思想提出了一种三角网格重建算法,在进行第三点搜索时采用了探测邻域规则,提高了搜索效率;邓曙光、刘刚[8]于 2006 年采用了直线分割式正负区的原理搜索第三点,每次只搜索点集中的部分点,在一定程度上提高了构网效率。

八叉树空间划分算法是常用的空间划分算法之一,能极大地提高模型重建效率。李泽宇等人[9]于 2000 年提出了一种基于八叉树的三维散乱数据点法向量估计方法。该方法利用八叉树结构建立点与点之间的拓扑关系,从而可以方便、快速的计算散乱点的 m 邻域,大大提高了计算效率;Ohtake 等人[10]于 2003 年将自适应八叉树分割的方法应用于隐式曲面重建,快速、有效地实现了点云数据的模型重建;伍军等人[11]于 2008 年借助八叉树空间划分算法,将搜索区域限制在临近的立方体的点集中,快速实现了三角网格重建。在进行分割时,当空间平均点数 k 取值很小时可以提高剖分效率,但可能导致漏洞产生,并且会造成一定量的无效计算;陈伟、刘肖琳[12]于 2009 年提出了基于空间栅格划分的三角剖分算法,该算法提高了剖分效率,并且保留了细节特征,但改进后的算法存在三角面片的冗余。

本文采用了自适应八叉树空间划分算法,将整体点云数据划分成较小的子域。设定这些子域相互覆盖,然后在每个子域内进行三角网格重建。最后对生成网格进行网格优化与法向量一致化处理,形成对整个点集的表面模型重建。该算法避免了网格拼接的过程,快速、有效地实现了模型的表面重建。

收稿日期:2012-06-28。教育部留学回国人员科研启动费项目(K314020901);中央高校基本科研业务费专项资金资助项目(Z10902 1004)。杨客,硕士生,主研领域:计算机图形学。张志毅,副教授。董艳,硕士生。

1 三角网生长法

基于生长法的三角剖分算法的基本思想是首先从点云数据中确定一个种子三角形;然后以该种子三角形的三条边为基础,搜索第三个点以构成新的三角形;以新构成的三角形为基础向四周扩展,最终完成对整体点云数据的三角网格重建。这种算法的效率比较低,时间复杂度为 $O(N^2)$,其中 N 为点云数据中点的数目。

本文基于三角网生长法对点云数据进行三角网格重建,在以边为基础进行扩展时设置了一些限定条件。具体网格重建步骤如下:

Step1 随机选取点云数据 P 中一点 P_1;

Step2 从点云数据 P 中计算出距离 P_1 最近的点 P_2;

Step3 从点云数据 P 中搜索第三个点 P_3,使得夹角 $P_1P_3P_2$ 最大;

Step4 构建初始的边列表 P_1P_2、P_1P_3、P_2P_3,以及初始的三角形列表 $P_1P_2P_3$;

Step5 如果边列表不为空,执行如下步骤:

1) 取出边列表中一条边 P_1P_2;

2) 搜索满足如下条件的点:点 P_4 满足 step3 中条件,并且 P_4 与边 P_1P_2 构成的三角形 $P_1P_2P_4$ 与三角形 $P_1P_2P_3$ 的二面角大于 30^0;

3) 将与新搜索到的点 P_4 所构建的边和三角形,分别添加到边列表和三角形列表中;

4) 删除被两个三角形共用的边 P_1P_2;

5) 重复执行 Step5;

Step6 列表为空,算法结束。

三角网生长法重建点云模型的的伪代码如下:

```
Read_Data ( ) ;                          //读取点云数据 P
Rand_Data( ) ;                           //从 P 中随机选取一点 P1
Nearest_Vertex( ) ;                      //求出距离 P1 最近的点 P2
Maxangle_Vertex( ) ;                     //求出使得夹角 P1P3P2 最大的点 P3
Initialize_Edge( ) ;                     //初始化边列表
Initialize_Triangle( ) ;                 //初始化三角形列表
while ( edge! =null) do                  //如果边列表不为空,执行如下步骤
Optimal_Vertex ( ) ;                     //搜索满足条件点 P4
Add_Edge( ) ;                            //将新构建的边加入边列表
Add_Triangle ( ) ;                       //将新构建的三角形加入到三角形列表
Delete_Edge ( ) ;                        //删除被两个三角形公用的边
endwhile
Output ( ) ;                             //算法结束,输出结果
```

2 自适应八叉树空间划分

由于对大量点云数据进行处理会耗时过大,通常会采用一定的空间分割算法将点云数据进行分割后再处理。八叉树空间分割算法是常用的空间分割算法之一,但通常的八叉树分割次数与形成的子域数目都由递归深度决定,这样可能会造成不必要的空间和时间的浪费。比如在分割过程中,由于同一级的节点的分割是同步的,但是点云数据并不是均匀的分布于各子域内,这样便可能造成对一个空的子域进行不必要的分割。

为了解决这个问题,本文采用了自适应八叉树空间划分算法,将点云数据分割成相互有覆盖的较小的子域,然后在每个子域内进行三角网格重建。每个节点划分过程停止的条件是该子域中的点云数目小于自适应参数 K。这样就有效避免了不必要的分割过程,同时也为后面的网格重建过程节省了时间和空间。K 的取值直接影响网格重建的效率和生成的网格质量(当 K 过大时,每个子域内的点数过多,虽然生成的网格质量会比较好,但效率就会很低;当 K 过小时,生成的网格质量会受到一定影响,效率也就会较快,但过度的细分也会导致效率下降)。设 K 为单元子域内点数目的平均值,则自适应八叉树分割的时间复杂度约为 $O(N^2/K)$,分割后三角网格重建的时间复杂度约为 $O(KN)$。为使整体消耗时间最小,本文定义了一个时间函数 $f(x)$。

$$f(x) = N^2/K + K \cdot N \tag{1}$$

对式(1)求极值可以得到 $K = \sqrt{N}$。因数据点分布不均匀,导致平均数 K 不能准确反应自适应效率,我们在试验中加入影响系数。经多次试验发现,该影响系数取二分之一时重建速度最快,重建结果较好。

给定输入点云数据 $P = \{P_i\}_{i=1}^{N} \subset R^3$,首先计算该点集中的点在 x,y,z 三个方向上的最大值和最小值 $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$;然后由这些数据构造初始的包含整个点集的长方体包围盒,并将其作为八叉树分割的根节点;然后,根据设置的分割条件将根节点平均分割成八个等尺寸的小长方体,称为根节点的孩子节点;孩子节点又被进一步划分成八个更小尺寸的长方体,直至达到分割停止条件,最后生成对整体点云数据的自适应空间划分。

八叉树分割过程如图 1 所示。

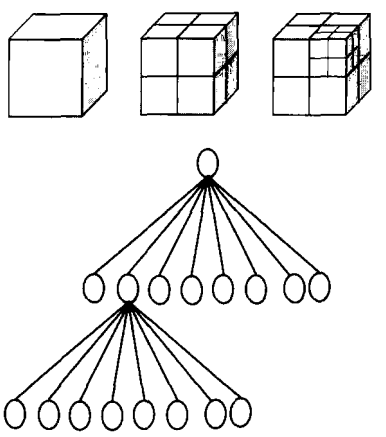

图 1 八叉树分割示意图

本文为每一个长方体定义了一个支撑半径 R:

$$R = \alpha d \tag{2}$$

式中,d 为长方体的对角线长度,α 为自定义的一个常数。当 α 大于 0.5 时,便能确保每个以长方体中心为圆心,以 R 为半径的子域相互有覆盖,本文取 α 为 0.6。分割形成的子域示意图如图 2 所示。

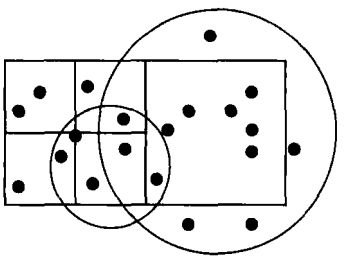

图 2 子域示意图

自适应八叉树空间划分的伪代码如下:

```
//定义八叉树结构体
struct OctreeNode
```

```
}
  //定义八叉树的八个子节点
  OctreeNode * Top_Front_Right, * Top_Front_Left;
  OctreeNode * Bottom_Front_Right, * Bottom_Front_Left;
  OctreeNode * Top_Back_Right, * Top_Back_Left;
  OctreeNode * Bottom_Back_Right, * Bottom_Back_Left;
};
//构造自适应八叉树
Creat_Octree( OctreeNode root)
{
  Sum_Cover( ) ;                    //求出该节点子域内的点云数目
  if( Sum_Cover > K)
//如果该子域内的点云数目大于 K 值,则继续划分
{
    Create_Octree( root→Top_Front_Right) ;
    Create_Octree( root→Top_Front_Left) ;
    Create_Octree( root→Bottom_Front_Right) ;
    Create_Octree( root→Bottom_Front_Left) ;
    Create_Octree( root→Top_Back_Right) ;
    Create_Octree( root→Top_Back_Left) ;
    Create_Octree( root→Bottom_Back_Right) ;
    Create_Octree( root→Bottom_Back_Left) ;
  }
}
```

3　网格优化

3.1　重复网格去除

采用自适应八叉树方法进行空间划分,由于分割子域之间有相互覆盖的点集,这样虽然避免了网格拼接的过程,但是形成的三角网格会有一定的重复三角形。去除重复的三角形过程如下:

已知一个三角形 ABC,寻找与其三个顶点都相同的三角形(包括 ABC、ACB、BAC、BCA、CAB、CBA 六种情况),保留其中一个,将剩余的三角形全部删除。

3.2　规范化网格

因重建后的网格,会存在少量正则度较差的网格,本文采用了一种常用的优化方法规范化网格。对于每一个有公共边的两个三角形所组成的空间四边形,应用最大角最小化的原则进行规范化,具体步骤如下:

1) 计算出空间四边形的最大角;

2) 若空间四边形存在两个对角线,如图 3(b)所示,则保留分割最大角的那条对角线;

3) 若空间四边形只存在一条对角线,则需要分情况考虑:若对角线未分割四边形最大角,如图 3(a)所示,则舍弃该对角线,连接另外一条对角线;若对角线本身已分割四边形的最大角,如图 3(c)所示,则该四边形即为规范四边形,不作处理。

网格规范化示意图如图 3 所示。

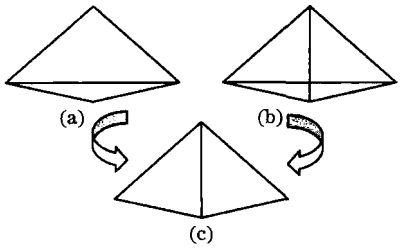

图 3　网格规范化示意图

网格优化的伪代码如下:

```
//去除重复网格
Read_Triangle ( );                          //读取三角网格数据
Same_Triangle ( );              //寻找三个顶点都相同的三角网格
Delete_Same ( );        //保留其中一个,将其余重复三角网格删除
//规范化网格
MaxAangle_Tetradron ( );                  //计算空间四边形的最大角
if( Both( ) )                               //如果两条对角线都存在
      Delete_LittleDiagonal ( );
                              //删除没有分割最大角的那条对角线
else if( BigDiagonal == null)
          //如果分割最大角的那条对角线不存在,执行下述过程
{
      Delete_LittleDiagonal ( );  //删除没有分割最大角的那条对角线
      Add_BigDiagonal ( );              //连接分割最大角的对角线
}
else
      continue;         //如果只有分割最大角的对角线,不作处理
```

4　法向量一致化

因算法是在分割的每个子域内进行三角网格重建,生成的三角面的法向量可能是不一致的,这样会给后期的绘制等处理操作带来困难。本文采用了三角面片定向的方法进行法向量一致化处理。一致化过程如下:

首先选取一个三角形 T_1,记顶点分别为 A、B、C,排列顺序为 ABC;找到与该三角形共边的三角形 T_2,假使其顶点为 A、B、D(或 A、C、E 或 B、C、F),则不管 A、B、D 的原来排序如何,将其顶点排序变为 BAD。这样由一个种子三角形依次扩散,直至遍历所有的三角形,完成了法向量的一致化。三角形法向量一致后的顶点排序如图 4 所示。

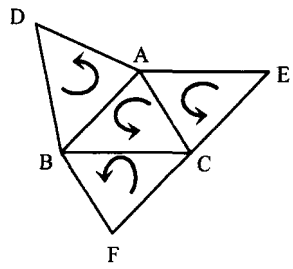

图 4　法向量一致化后的顶点排序

法向量一致化的伪代码如下:

```
while( not all triangles are marked) do
//如果有三角形尚未进行法向量一致化,执行下述步骤
Select_Triangle ( );                          //选取一个三角形 T1
CommonEdge_Triangle( );              //寻找与 T1 共边的三角形 T2
Order ( );              //将三角形 T2 重新排序,完成法向量一致化
endwhile;
```

5　实验结果与分析

本文的算法采用了大量的数据进行验证,生成的实验结果都是很好的。本文所使用的计算机环境为:CPU AMD Athlon(tm)Ⅱ×3 2.90G,内存 2.0G,操作系统 Win7,编程平台 VC ++ 6.0。实验数据来源:台湾大学资讯工程学系。实验数据大小:Slice 99.7KB,Pig 108KB,Dinosaur 197KB,Horse 290KB,Bunny 1093KB,Cow 1520KB。

以下是部分模型数据的实验结果。

表 1 原始三角网生长法

点云数据	点数	三角形数	时间(s)
Slices	2852	5735	11.1
Pig	3522	7209	15.3
Dinosaur	6002	12975	42.6
Horse	8229	16447	82.9
Bunny	35947	69463	1471.6
Cow	49667	99376	2802.4

表 2 改进后的三角网生长法

点云数据	点数	三角形数	时间(s)
Slices	2852	5780	0.3
Pig	3522	7297	0.6
Dinosaur	6002	13107	1.1
Horse	8229	16615	1.5
Bunny	35947	69762	11.0
Cow	49667	99857	31.4

由表 1 与表 2 实验结果对比可知,与原始的三角网生长法相比,改进后的算法极大地提高了重建效率,并且随着点云数据的增加,提高的倍数总体呈增长趋势。实验结果分析如图 5 所示。

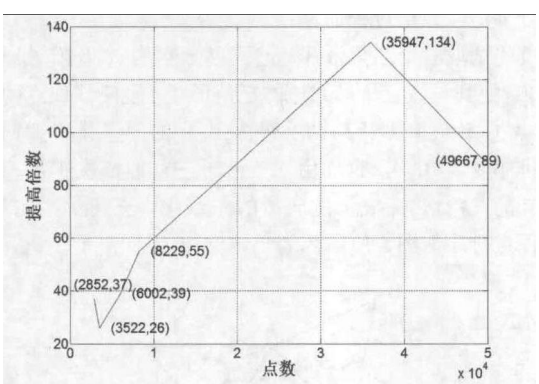

图 5 点数与提高倍数关系图

其他算法实验数据如下:

表 3 文献[11]的实验数据

点云数据	点数	三角形数	时间(s)
Bunny	8146	17125	5
Horse	19851	40441	22
Dinosaur	23982	48522	26

表 4 文献[12]的实验数据

点云数据	点数	三角形数	时间(s)
大卫	6502	21904	2.1
马	5489	17080	1.7

由表 2 - 表 4 实验数据对比可知,与其他算法相比,本文改进后算法在重建效率上有明显的优势。在点数大致相同的情况下(文献[11]中的 8000 点左右;文献[12]中的 6000 点左右),本文算法重建的时间大约为文献[11]的 1/3,约为文献[12]的 1/2。

在构建的网格质量方面,本文的算法也较文献[12]更优些。本文算法重建生成的三角网格数量与原始点云数目的比例约为 2:1,而文献[12]中所生成的三角网格数量与原始点云数目的比例约为 3:1。

因当一个顶点被六个三角形共用时,形成的网格是最优的。这时,顶点在每个三角形内的作用域约为该三角形面积的 1/3,一个顶点的全部作用域大约为两个三角形的面积。所以,当三角形数目与顶点比例约为 2:1 时,形成的三角网格是最优的。顶点作用域的示意图如图 6 所示。

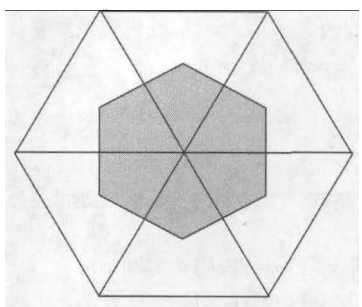

图 6 顶点作用域示意图

模型表面重建的结果如图 7 - 图 10 所示。图中,左上角图为模型线框图,左下角图为其局部放大图;右上角图为模型平面着色图,右下角图为其局部放大图。

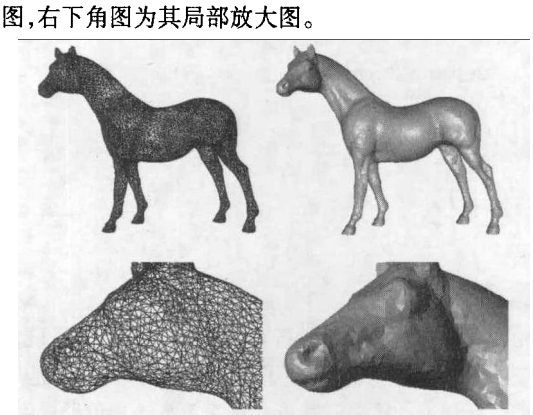

图 7 Horse 模型重建结果

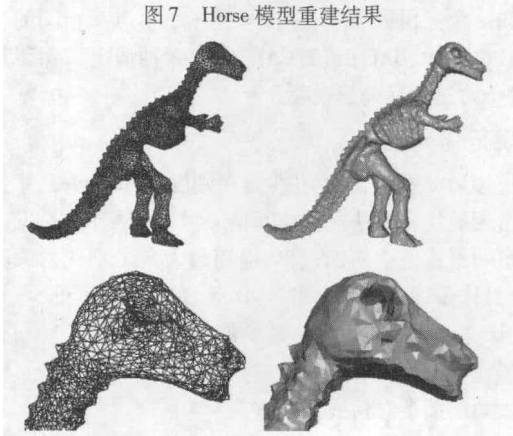

图 8 Dinosaur 模型重建结果

图 9 Bunny 模型重建结果

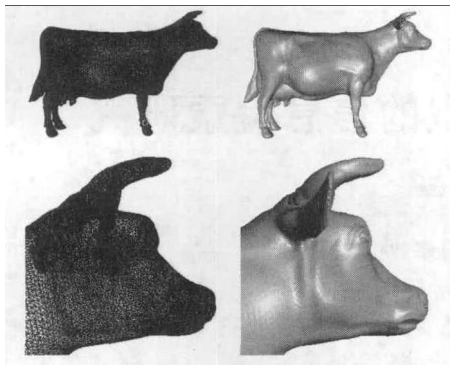

图10　Cow模型重建结果

由图7-图10可知,本文算法能很好地重建点云模型表面,细节特征明显,能适用于多种不同的模型,有较强的鲁棒性。

6 结　语

本文针对普通的三角网生长法对点云数据进行表面模型重建耗时长、效率低等缺点,采用了自适应八叉树空间分割的思想。将点云数据分割成较小的子域,然后在每个分割子域内进行三角网格重建。通过网格优化方法进行重复网格去除和网格规范化。最后对生成的网格进行法向量一致化,形成对整体点云数据的表面重建。实验结果表明,该算法明显地提高了模型重建的效率,生成的三角网格也比较规范,能够体现点云模型的细节特征,鲁棒性较好。但是本文算法也存在一定的不足之处,与传统的三角网生长法相比,本文改进后的算法生成的三角网格有一定的冗余网格。这是在今后的研究中需要考虑和解决的问题。

参 考 文 献

[1] Miles R E. Probability distribution of a network of triangles(a solution to problem 67-15)[J]. Society for Industrial and Applied Mathematics, 1969, 11(3): 399-422.

[2] Lingas A. The greedy and Delaunay triangulations are not bad in the average case[J]. Information Processing Letters, 1986, 22(1): 25-31.

[3] 余杰,吕品,郑昌文. Delaunay三角网构建方法比较研究[J]. 中国图象图形学报,2010,15(8):1158-1167.

[4] Green P J, Sibson R. Computing dirichlet tessellations in the plane[J]. The Computer Journal, 1978, 21(2): 168-173.

[5] Bowyer A. Computing dirichlet tessellations[J]. The Computer Journal, 1981, 24(2): 162-166.

[6] Watson D F. Computing the n-dimensional delaunay tessellation with application to Voronoi polytopes[J]. The Computer Journal, 1981, 24(2): 167-172.

[7] 董洪伟. 散乱点云的三角网格重构[J]. 计算机工程,2005,31(15):30-32.

[8] 邓曙光,刘刚. 一种TIN生成算法的改进与实现[J]. 计算机时代, 2006, 2(1): 1-2.

[9] 李泽宇,李德华,胡汉平,等. 基于八叉树的三维散乱点的法矢的估计[J]. 计算机与数字工程,2000,28(4):44-65.

[10] Ohtake Y, Belyaew A, Alexa M, et al. Multi-level partition of unity implicits [J]. ACM Transactions on Graphics, 2003, 22(3): 463-470.

[11] 伍军,杨杰,秦红星. 基于广度搜索的增量式点云表面重建[J]. 上海交通大学学报, 2008, 42(10): 1740-1744.

[12] 陈伟,刘肖琳. 一种快速三维散乱点云的三角剖分算法[J]. 计算机仿真,2009,26(9): 338-341.

(上接第3页)

集而导致分类准确率较低的问题,本文提出了一种基于改变率自适应分类的多类隐写分析方法。该方法使用SVR估计待测图像的改变率,进而根据改变率自适应地选择分类器,从而去除了多嵌入率因素对分类结果的干扰。实验结果表明,所提方法相较于目前准确率最高的文献[6]方法,真阳性率和真阴性率均有所提高,平均提高约2%~3%,特别是在嵌入率较低的情况下,提高幅度可达5%以上。

参 考 文 献

[1] 王朔中, 张新鹏, 张卫明. 以数字图像为载体的隐写分析研究进展[J]. 计算机学报, 2009, 32(7):1247-1263.

[2] Lubenko I, Ker A D. Steganalysis using logistic regression[C]//Proceedings of the Society of Photo-optical Instrumentation Engineers, CA, USA, Jan 24-46, Bellingham: SPIE Press:0K01-0K11.

[3] Pevny T, Fridrich J, Ker A D. From blind to quantitative steganalysis [J]. IEEE Transactions on Information and Security, 2012, 7(4): 445-454.

[4] Pevny T, Fridrich J. Benchmarking for steganography[C]//Proceedings of 10th International Workshop on Information Hiding, Santa Barbara, CA, USA, Berlin: Springer-Verlag, 2008:251-267.

[5] 黄聪,宣国荣,高建炯,等. 基于模式识别的多类隐写分析[J]. 计算机工程与应用,2006(27):43-45.

[6] Pevny T, Fridrich J. Multiclass detector of current steganographic method for JPEG format[J]. IEEE Transactions on Information and Security. 2008, 3(4):635-650.

[7] Pevny T, Fridrich J. Merging Markov and DCT features for multi-class JPEG steganalysis[C]//Proceedings of the Society of Photo-optical Instrumentation Engineers, San Jose, CA, USA, Jan. 29-Feb. 1, Bellingham: SPIE Press, 2008:301-314.

[8] Scholkopf B, Smola A. Learning with kernels: support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning)[M]. The MIT Press, 2001.

[9] Sallee P. Model-based steganography[C]//2nd International Workshop on Digital Watermarking, Seoul, South Korea, Oct 20-22, Berlin: Springer-Verlag, 2004:174-188.

[10] Westfeld A. High capacity despite better steganalysis (F5-a steganographic algorithm)[C]//Proceedings of 4th International Workshop on Information Hiding, Pittsburgh, PA, USA, Apr 25-27, Berlin: Springer-Verlag, 2001, 289-302.

[11] Kim Y, Duric Z, Dana Richards. Modified matrix encoding technique for minimal distortion steganography[C]//Proceedings of 8th International Workshop on Information Hiding, Alexandria, VA, USA Jul 10-12, Berlin: Springer-Verlag, 2006:314-327.

[12] JP Hide&Seek[OL]. 2011. http://linux01.gwdg.de/~alatham/stego.html.

[13] Hetzl S, Mutzel P. A graph-theoretic approach to steganography[C]//9th International Conference on Communications and Multimedia Security, Salzburg, Australia, Sep 19-21, Berlin: Springer-Verlag, 2005:119-128.

[14] Dong J, Wang W, Tan T. Multi-class blind steganalysis based on image run-length analysis[C]//8th International Workshop on Digital Watermarking, Univ Surrey, Guildford, England, Berlin: Springer-Verlag, 2009:199-210.