一个根据几何信息对场景对象重组织以加速显示的方法

周毅敏 李光耀
(同济大学电信学院CAD中心 上海 200080)

摘 要 大规模场景图形应用和交互系统常常需要同时处理大量的场景对象。虽然场景结构对每帧的显示速度有很大的影响,但近几年来的研究和关注的重点已不是场景对象的组织结构,而主要集中在诸如物体表面的简化以及每个对象的表示等一致性特征上。场景中的物体几何信息的一致性十分易于理解,文中提出一个基于这些几何信息的分析方法以提升特定场景的显示效率。这个方法主要是将场景组织成层级结构。还提出了这个方法的许多变化形式,其中有些仅使用场景对象的几何信息,有些利用了场景固有的几何性质,而有些则同时使用了这些信息。最后还展示了一些统计信息,以证实文中的方法对图形交互应用效能的提升。

关键词 几何信息分析,聚类层级分析,群,剔除
中图法分类号 TP301 **文献标识码** A

Method for Accelerating Display Speed by Organizing Scene Objects via Geometric Data Analysis

ZHOU Yi-min LI Guang-yao
(CAD Center of Electronics and Information College, Tongji University, Shanghai 200080, China)

Abstract Graphical applications and interactive systems, especially applied in large scene simulation, are often required to display large quantities of objects synchronously. But not much attention has been focused on the organization of the scene structure which plays an important role in determining the speed of displaying the whole scene each frame respectively while much has been spent on surface simplification and other coherency within each object's representation. Though the coherency within the scene is so simple to understand, this thesis will still present some also simple methodology which can be used to factually improve the efficiency in displaying the customized scene via applying geometric data analysis or related method. We presented a new method for representing a hierarchy of scene objects which partitions the scene into groups of objects termed as clouds of objects. The method has several variants, some of which applies only geometric data analysis, some of which just applies scene's inherent geometrical property and others which applies both. Finally, we presented some statistics information to verify improvement by applying our method in graphical interactive systems.

Keywords Geometrical data analysis, AHC, Cloud, Culling

1 引言

现代的图形系统需要显示包含大量对象的场景。有些大城市的场景可能很轻易就包含上千万个甚至更多的对象。然而,如果每一帧都需要处理这么多对象对图形系统来说是个很大的挑战。必须找到一种方法尤其是在交互实时应用时决定每一帧需要处理哪些对象而不需要处理哪些。近年来提出的许多技术都是为了从大量的场景对象中选出哪些最终需要被处理。而这些技术一般都是在3D引擎渲染过程的对象剔除阶段对对象进行处理。1970年,HSR[1]提出了隐藏面剔除算法,另一种剔除方法被称为视锥体剔除,它是剔除视锥体之外不需要被显示的所有对象。在极端情况下的另一种方法——遮挡剔除是将被其它对象遮挡的对象从场景中剔除。在上述的视锥体剔除中已经使用了高效的层级技术[2,3]。随着GPU加速技术的流行,又有些文献提出了基于GPU的剔除层级计算的方法[21]。

基于层级技术这种概念,本文提出了一种基于几何信息分析[4]的新方法。每个区域中的相关对象会形成一个群体,我们称为对象群。对区域中的对象进行分类,这样一个较大的群可以被分割为多个子群。可以通过设置区域的大小或者区域中对象的数量确定最终不可分割的子群的大小。本文的工作主要由以下几部分组成:(1)提出了一个分割对象的层级结构,(2)将经典和标准的分类方法与(1)中提出的方法进行比较,(3)提出一种用于指导分割过程的修正方法用于控制对象群的形状。本文提出的层级模型可以应用于各种场合如光线跟踪、碰撞检测、在线和离线查询等,非常适用于城市等场景中。这个方法在城市穿越[5]、静态场景的飞行模拟等应用中较大地提升了显示速度。下面集中讨论这个模型的精确表示以及对象群层级的高效生成算法。

2 对象层级

对于实时系统和3D引擎来说,对象都是具体的对象,而

周毅敏(1980—),男,博士生,主要研究方向为图形图像、虚拟现实、城市仿真,E-mail:1010080053@tongji.edu.cn;李光耀(1953—),男,博士生导师,主要研究方向为图形图像、虚拟现实、城市仿真。

不是几何对象。因此一个对象可能包括几个、几十个甚至更多的几何对象。这是很自然的事实，属于同一个对象的几何对象一般位置相近，反映出了场景的结构信息。但这未必能反映出场景的所有结构信息，因为这些信息还包括具体对象之间的联系。如果一个场景包含数百个建筑，属于同一个街区的建筑相互之间就有联系，而不同街区的建筑之间则没有这种联系。根据这种联系可以得到我们关于对象层级的概念。相近的对象属于同一个对象群，而相隔较远的对象属于不同的对象群。由于对象是以层级的关系组织起来的，位于下层的对象相隔较近，而位于上层的对象相距较远。

2.1 欧几里得距离

整个分类的目标非常容易理解。分类的目标是为了构造对象群，使得在同一个对象群中的物体相互距离尽可能地近，而在不同对象群中的物体相隔尽可能远。分类的目标尽可能地保证相隔较远的对象不会过早地被分在一个群中，而相隔较近的对象不会被错误地分在不同的对象群中。在最初阶段，每个对象独立地被认为自成一群。分类时相近的对象被加入越来越大的对象群。换言之，每个对象群变得越来越大而对象群的数目则变得越来越少。最后所有的对象群被集中成一个代表整个场景的对象群。每一步选择相距最近的两个对象群混合成一个更大的对象群，见图 1。在图 1 中，对象 o 和 o' 是相距最近的两个对象。

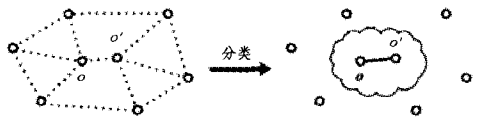


图 1 欧几里得分类的基本概念

当这个迭代过程结束后，我们得到一个二叉树，树的每个叶子结点对应于最初场景中的每个对象。当我们把两个对象合并以后，创建一个新的结点来表示它，原来的两个结点成为它的结点。这个方法被成功地运用于即时[17,18]视觉依赖重建中，但从未被应用于本文涉及的领域。但是本方法仅利用欧几里得距离也有缺点，层级中在特定位置的对象群可能具有一个不规则的形状。然而，这个缺憾在某种程度上可以由本文第 3 节偏差矫正提出的方法弥补。

2.2 群分类算法

如前所述，我们使用一个迭代步骤对对象进行分类以创建一个层级二叉树。首先定义一个在每个迭代步骤用到的公式[4]。给定单个对象或对象群 o 和 o'，o 和 o' 的中心位于 M^o 和 $M^{o'}$。对于对象群，这里抽象意义上的中心在通常意义下及我们的算法中被定义为立体几何中的重心。两个对象或对象群合并后的中心是很容易由式(1)[4]求出来的。w^o 是对象 o 的权值，$w^{o'}$ 是对象 o' 的权值，在这里我们假定它们是相同的。

$$M^d=\frac{w_oM^o+w_o{}'M^{o'}}{w_o+w_{o'}} \tag{1}$$

中心位于 M^o 和 $M^{o'}$ 的方差系数用 $\delta(o,o')$ 表示(又被称为惯性系数)，被定义为

$$\delta(o,o')=\frac{w_ow_{o'}}{w_o+w_{o'}}dis(M^oM^{o'})^2 \tag{2}$$

$dis(M^o,M^{o'})$ 是对象 M^o 和 $M^{o'}$ 之间的距离，我们可以定义 δ 为两群之间的聚类指标。对象群 d 是通过合并两个对象群 c' 和 c"得到的且聚类指标符合下述公式，

任意对象群 $c\in C,c'\in C$ 和任意对象群 $c\in C-\{c',c''\}$：

$$\delta(c,d)=\frac{(w_c+w_{c'})\delta(c,c')+(w_c+w_{c''})\delta(c,c')-w_c\delta(c',c'')}{w_c+w_{c'}+w_{c''}} \tag{3}$$

根据上述定义的数式来定义每步迭代的算法形式。下面给出算法的伪码。

算法 1 构建层级群

1. 计算每对对象间的方差系数 δ，并加入到集合 S 中
2. while S 非空
3. 从集合 S 中取出最小元素，假定它为 δ(c′,c″)
4. 从 S 中删除所有与 c′和 c″相关的 δ
5. 将 c′和 c″合并为一个更大的 c
6. 对任意 d∈S 计算 δ(c,d)
7. 将 c′和 c″合并的结点 c 作为层级树 T 的新结点

endWhile

8. 返回 T

算法 2 创建当前层级(ConstructCurrentLevel)

1. 创建结点 S
2. 选出算法中创建的 T 的最后 levelNodesCount 个结点
3. 构建层级群，假定为 t[1...levelNodesCount]
4. 对任意 i∈[1,levelNodesCount]
5. s[i]←ConstructCurrentLevel(t[i],levelNodesCount)
6. 将所有的 s[i]作为 S 的子结点
7. endFor
8. 返回 S

对于算法 1，中心思想是对任意两个对象群之间的最小 δ 的贪心查找，这通过维护一个这些 δ 的堆来实现。通过不断地对堆进行插入和删除操作来实现整个层级对象群树的构建。维护堆的每个操作(插入、删除)需要 O(log$n$)的时间，共需进行 O(n^2)个维护堆的操作。整个算法的运行时间是 O(n^2log$n$)，空间复杂度是 O(n^2)。

对于算法 2，很容易看出它的运行时间为 O(hn)，其中 h 是场景对象的最大层级数。

3 偏差矫正

上述分类算法的指导思想是希望每一步求出的对象群具有较小的方差值，迭代步骤就是以此为目标。每一次选择所有对象群之间最小的 δ 值。式(3)可以被解释为当把两个对象群合并为一个较大的对象群时，其中另一个群与新群的群间方差减去这个对象群与原来的两个群的群间方差，也就是生成的新群 c 的方差增量。当我们选择最小 δ 值的时候，在某种程度上最小化了同样多个群生成新群的最小方差增量。

但是这个目的不是我们所需的唯一目标。我们还需要控制对象群的形状，希望对象群有一个紧凑的形状，希望它的表面积最小，或尽可能接近于球形。也就是在给定对象群的体积下尽可能减小它的表面积。上述算法将对象群分类为一个尖锐的形状，可能是过于强调了某个方向的效果而忽略了其它方向的影响。这样的操作可能会延长交互应用的处理时间，然而在文献[19]有一个易于控制偏差矫正的简单方法，即增加一个形状控制项。类似地，我们定义不规则因子 γ 为凸的表面积和它的体积的平方的比值。

$$\gamma=\frac{S^3}{36\pi V^2} \tag{4}$$

这个技巧已经被广泛地应用于各个领域。在合并两个对

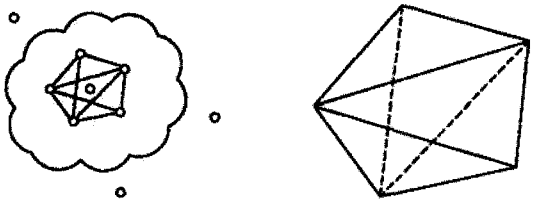
象群的时候，两个群的不规则系数 γ 分别为 γ_1 和 γ_2。定义形状惩罚值为合并操作的不规则系数的改变量。

$$\varepsilon_{shape}=\frac{\gamma-\max(\gamma_1,\gamma_2)}{\gamma}\min(\delta_1,\delta_2) \tag{5}$$

所以 ε_{shape} 是正是负由不规则系数是增加还是减少确定。堆中最后需要比较的改变值为

$$\delta+\alpha\times\varepsilon_{shape} \tag{6}$$

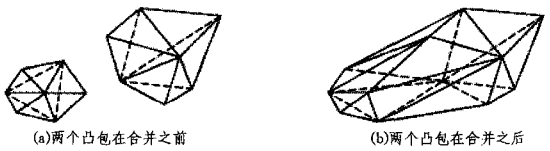
正如在群分类算法中提到的，$\delta(c,d)$ 的计算需要常量的代价，位于算法 1 的第 6 行。剩下的问题是我们如何在每一步中计算 ε_{shape} 的值并将它与 δ 相结合。一般来说，对象群的形状可以由它内部的某些对象来表示，从几何术语来说就是这些对象的重心位于的空间中的位置的点的凸包。这些点可以近似地被认为是对象的重心，因为在图形学中只考虑对象的形状所以可以认为是对象的几何重心，或者可以从某种程度上忽略每个对象的体积。我们如果不能忽略对象的体积或者不同对象的体积差别很大，那么仍然可以间接地用对象的包围盒作为凸包算法的基本元素以避免考虑复杂的多边形细节。这样做只会给我们的算法增加常量的复杂度，如图 2 所示。



(a) 框线表示一个对象群　　(b) 这个对象群所形成的凸包

图 2　一个对象群和它的凸包

图 2(a)表示在某一步中进行分类，图 2(b)的框线连接了代表对象群的凸包的对象。当凸包建立以后，它的表面积和体积可以非常轻易地计算出来，它是群中对象数的线性复杂度。这是因为具有 n 个顶点的凸包至多有 $3n-6$ 条边和 $2n-4$ 个面，这可以通过著名的欧拉公式计算得到。最后需要考虑的是凸包的联合，如图 3 所示。本文的实现中使用 quick-hull 算法[20]。所以算法 1 第 5 行改为“merge c′ and c″ into a larger cloud c and merge their convex hulls into one”。quickhull 算法的时间复杂度为 $O(n\log n)$，这里 n 是凸包上的顶点数，因此算法的整个运行时间变为 $O(n^3\log^2 n)$。



(a)两个凸包在合并之前　　(b)两个凸包在合并之后

图 3　两个凸包的合并

4　层级网格分类

假定在一个空间中，对象是均匀分布的。在这样一个理想的空间中，不需要使用复杂的 AHC[4] 算法。我们可以使用一个简化的算法创建一个场景对象的层级模型。这个方法来自于层级网格[16]，最初被用于图形学中的模型创建。首先把空间分割为几个层级的小网格。将最精细的网格中的对象合并为一个对象群，然后将比最精细网格层次高一级的网格中的对象群合并为一个对象群。重复这个操作，直到最终将所有对象合并为一个代表整个场景的对象群。特殊的最简单的层级网格就是八叉树[15]，如图 4 所示。

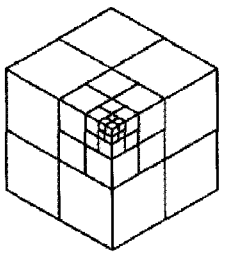


图 4　空间的层级网格分割之一八叉树的空间分割

层级网格非常易于理解，它的运行时间为 $O(n+2^h)$。其中 h 是场景的层级数。它非常适合于对象均匀分布的场合。但是当对象聚集在某些区域时，如果处理得不好，生成的网格或者树可能是很不平衡的。本文的实验数据中，我们使用的八叉树模型所建立的层级数近似地与 AHC 算法所建立的层级数相当，它们的效率比较如图 7 所示。

5　方法的应用

5.1　本方法在遮挡剔除中的应用

我们的方法可以应用于一些计算机图形应用中。第一个应用实例是软件遮挡剔除。在计算机图形应用中，当摄像机或观察点接近某些对象时，这些对象可能会遮挡住远离观察点的一些对象，因此系统不需要显示它们。如果只是少数几个对象被遮挡，那么所节省的时间几乎可以忽略不计，但当场景中有大量的对象被遮挡并被合理剔除时就可以节省相当可观的时间。在某些情况下需要并且可以使用 PVS。但是当计算每个视单元的 PVS 时，使用本文提出的算法对场景进行分类形成对象层级而不是随机或手动地形成对象层级。这是本文的方法在遮挡剔除中的应用。

5.2　本方法在 3D 实时系统中的应用

本文的方法也可运用在一些实时系统[21]中，正如之前的部分所提到的，可以运用在静态场景的飞行模拟和穿越漫游等实时系统中。对于飞行模拟系统，我们进行了悬挂在固定位置的不规则形状的小行星群的飞行模拟，这种场景经常出现在游戏中。我们在场景中同时应用了 AHC 算法和层级网格分类方法，以及偏差矫正。每个方法的平均对象剔除阶段使用的时间如图 5 所示(图中仅给出了对象数，整个场景的面片数则可按照每个对象有 500～2500 个面推算出)。在这个图中可以看到不使用矫正偏差的 AHC 算法确实提高了剔除阶段的效率。应用层级网格较大幅度提高了剔除的效率，因为它完全利用了场景内在的一致性，生成的对象均匀分布在场景中。但是通常情况下，均匀分布的场景并不多见。

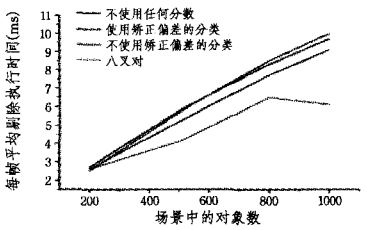
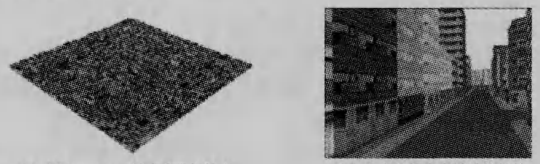


图 5　每种分类方法剔除阶段所用的时间

对于穿越漫游系统，我们使用 3ds max 和它的插件 Ghosttown 模拟了上百个甚至上千个建筑的城市场景(每个建筑模型对象约包含 5～20 个面)。城市场景和穿越场景的剔除阶段平均所用时间如图 7 所示。我们在飞行模拟系统中

应用了提到的每种方法以及八叉树。城市的场景和在其中漫游的场景如图 6 所示。自然的分类方法是根据对象所属的街区来划分。在图中可以看到分类算法得到了一个相对较好的结果，对象剔除阶段的运行效率提升了约 30%，达到了使用八叉树的效率。由于硬件和 CAD 软件的限制，场景中的对象数量都不是很大，但通过我们的分析可以预期在几十万甚至更多对象的场景中会得到类似的效果，只要我们设置合适的参数如最大层级数。



(a) 用Ghosttown创建的虚拟城市　(b) 在虚拟城市中漫游的场景

图 6　虚拟城市场景

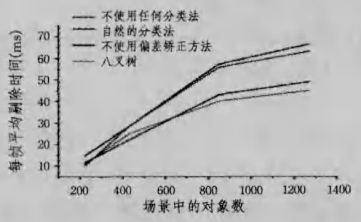
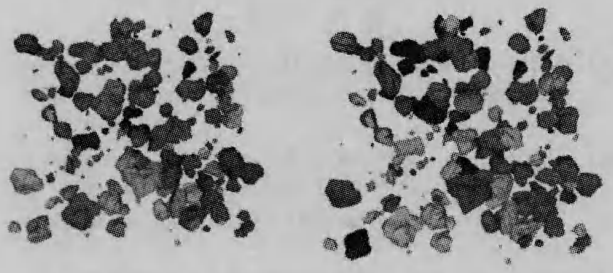
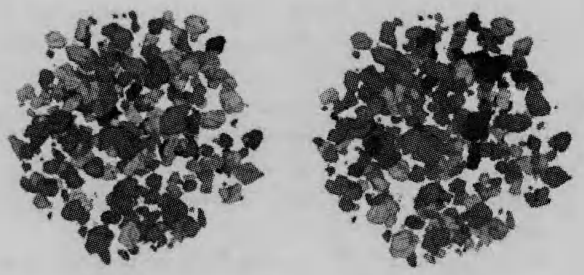


图 7　城市场景中每种分类算法剔除阶段平均所用的时间



(a) AHC 算法生成的立方体空间内 200 个对象形成的 16 个对象群　(b) AHC 算法结合矫正偏差生成的立方体空间内 200 个对象形成的 16 个对象群



(c) AHC 算法生成的球形空间内 500 个对象形成的 25 个对象群　(d) AHC 算法结合矫正偏差生成的球形空间内 500 个对象形成的 25 个对象群

图 8　AHC 算法生成的对象群

在上述两项实验的结果数据中，虽然未明确列出实时漫游场景的帧显示速度，这是由于实验环境的硬件设备相对落后，在现在的环境下已不具代表性，但是在我们观察得到的帧速率数据中已经发现，运用我们的预处理方法比不运用提升了 10%左右。值得注意的是，根据算法复杂度理论和图形学相关技术及理论，我们的实验结果中所得到的效率的提升在绝大多数软硬件环境，包括今后出现的环境中都是可复制和再现的。而我们这项技术提升效率的最关键环节是在对象或表面剔除阶段。AHC 算法生成的对象群如图 8 所示。

6　应用结果

图 7 描述了本文的分类算法在不同规模对象数场景中的运行效率。图 8(b)和图 8(d)展示了使用偏差矫正的 AHC 生成的更规则的形状的对象群。但是图 8(c)和 8(d)的分类结果不如图 8(a)和图 8(b)，原因是后者具有更密的对象分布。分类算法的效率计算均以运行在 2.53 GHz Intel i3 CPU 和 8GB 内存的系统为准。如果不使用偏差矫正，需要 $O(n^2 \log n)$的运行时间，否则需要 $O(n^3 \log^2 n)$。根据我们的实验结果，不使用偏差矫正的 AHC 算法效率更高，而使用矫正偏差则得到较好的分类视觉效果。第二部分是场景树的构建，需要 $O(hn)$的运行时间，其中 h 是场景的最大层级数。对于整个算法来说，这个过程的运行时间几乎可以忽略。

结束语　本文提出了一个可行的分类算法，即完全利用分布在场景中的对象的内在一致性，将场景对象分类成层级对象群。在此基础上还提出了一种偏差矫正方法用于控制对象群的形状。这种算法可以应用于文中列出的各种实际应用中，并且确实加快了 3D 实时系统的显示速度。本文提出的算法独立于观察空间一次分类就可以用于场景的任何平移或旋转变换中，这不同于标准的方法如八叉树或 kd 树。当然本文的方法也可以做一些改进，比如对每个对象赋予不同的权值，在对象群中应用 LOD 技术以及运用 Delaunay 三角化方法对场景作预处理等等。

参 考 文 献

[1] Sutherland I E, Sproull R F, Schumaker R A. A characterization of ten hidden surface algorithms[J]. ACM Computer Surveys, 1974, 6(1): 1-55

[2] Clark J H. Hierarchical geometric models for visible surface algorithms[J]. Communications of the ACM, 1976, 19(10): 547-554

[3] Kumar S, Manocha D, Garrett W F, et al. Hierarchical back-face computation[J]. Computers & Graphics, 1999, 23(5): 681-692

[4] Roux B L, Rouanet H. Geometric Data Analysis: From Correspondence Analysis to Structured Data Analysis[M]. Kluwer Academic Publishers, 2004

[5] Cohen-Or D, Chrysanthou Y, Silva C, et al. A survey of visibility for walkthrough applications[J]. IEEE Tr. Vis. and Comp. Graphics, 2003, 9(3): 412-431

[6] Wonka P, Schmalstieg D. Occluder shadows for fast walkthroughs of urban environments[J]. Eurographics, 1999, 18(3): 51-60

[7] Schauer G, Dorsey J, Decoret X, et al. Conservative volumetric visibility with occluder fusion[C]// SIGGRAPH 2000 Conference Proceedings. July 2000: 229-238

[8] Surdasky O, Gotsman C. Dynamic scene occlusion culling [J]. IEEE Trans. on Visualization and Computer Graphics, 1999, 5(1): 13-29

[9] Batagelo H, Wu S. Dynamic scene occlusion culling using a regular grid[C]// Proceedings of the XV Brazilian Symposium on Computer Graphics and Image Processing. 2002: 43-50

[10] Lu T, Chang C. Distributed visibility culling technique for com-

plex scene rendering[J]. Visual Languages and Computing, 2005,16(5):455-479

[11] Laine S. A general algorithm for output-sensitive visibility preprocessing[C]//Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games. 2005:31-40

[12] Djeu P, Keely S, Hunt W. Accelerating shadow rays using volumetric occluders and modi_ed kd-tree traversal[C]//Proceedings of the Conference on High Performance Graphics 2009. 2009:69-76

[13] Kao C, Tsai R. Properties of a level set algorithm for the visibility problems[J]. Scientific Computing, 2008, 35(2):170-191

[14] Choi B, Chang B, Ihm I. Technical section: Construction of efficient kd-trees for static scenes using voxel-visibility heuristic [J]. Computers and Graphics, 2012, 36(1):38-48

[15] Jackins C, Tanimoto S L. Oct-trees and their use in representing 3-d objects[J]. *Computer Graphics and Image Processing*, 1980, 14:249-270

[16] Reinhard E, Smits B, Hansen C. Dynamic acceleration structures for interactive ray tracing[C]//Eurographics Workshop on Rendering. 2000:299-306

[17] Luebke D, Erikson C. View-dependent simplification of arbitrary polygonal environments // SIGGRAPH 97 Proc. August 1997: 199-208

[18] Xia J C, Varshney A. Dynamic view-dependent simplification for polygonal models[C]//Proceedings of Visualization '96. October 1996:327-334

[19] Garland M, Willmott A, Heckbert. Hierarchical face clustering on polygonal surfaces[C]//Proceedings of the ACM Symposium on Interactive 3D Graphics. 2001:49-58

[20] Barber C B, Dobkin D P, Huhdanpaa H T. The quickhull algorithm for convex hulls[J]. ACM Trans. on Mathematical Software, 1996, 22(4):469-483

[21] Wong S K, Cheng Yu-chun, Lii S Y. GPU Ray Tracing Based on Reduced Bounding Volume Hierarchies[C]//Proceedings of the *Ninth International Conference on Computer Graphics, Imaging* and Visualization. 2012:1-6

[22] Wang Rui. Qian Xue-lei. OpenSceneGraph 3 Cookbook[M]. Packt Publishing, March 2012

(上接第 269 页)

82%，而 TF-IDF 仅仅为 68%。因此我们可以得出结论，SS-IDF 比 TF-IDF 在精度与召回率方面具有更好的平衡性。

表 1 部分实验参与者信息

用户	性别	年龄	专业	研究方向
User1	男	25	计算机科学与技术	J2EE 架构及技术
User2	女	22	软件工程	大数据处理
User3	男	23	信息与通信工程	无线通信技术
…	…	…	…	…
User8	男	24	计算机技术	人工智能 …
…	…	…	…	

表 2 TF-IDF 算法和 SS-IDF 算法在 cutoff=0.65 平均实验结果

性能指标	IF-IDF	SS-IDF
Accuracy	78.3%	80.8%
Precision	77.2%	77.9%
Recall	21.8%	35.7%
Specificity	97.5%	94.6%
F1-measure	38.9%	48.6%

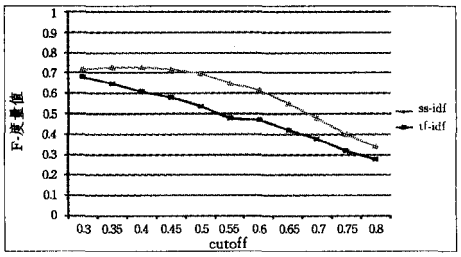


图 2 TF-IDF 算法和 SS-IDF 算法在不同 cutoff 下的 F-性能指标

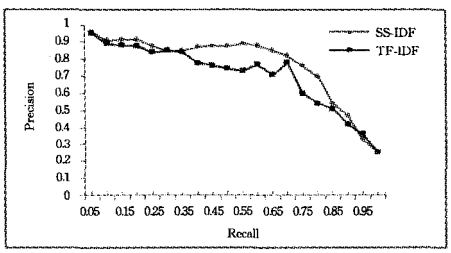


图 3 PR 曲线

结束语 为了改善传统上用于新闻项目推荐的词频逆文档频率(TF-IDF)加权技术，本文提出了一种新方法。本文的方法基于概念和它们的语义相似性，从中得到新闻项目之间的相似性。在新闻项目推荐中通过比较性能指标，本文提出的算法性能优于传统的词频逆文档频率(TF-IDF)加权技术。

参 考 文 献

[1] 华秀丽，朱巧明，李培峰. 语义分析与词频统计相结合的中文文本相似度量方法研究[J]. 计算机应用研究，2011，29(3)：834-836

[2] Goossen F, Jntema W, Frasincar F, et al. News Personalization using the CF-IDF Semantic Recommender[C]//Proc of the International Conference on Web Intelligence, Mining and Semantics. 2011

[3] 黄承慧，印鉴，侯昉. 一种结合词项语义信息和 TF-IDF 方法的文本相似性度量方法[J]. 计算机学报，2011，34(5)：857-863

[4] 李明涛，罗军勇，尹美娟，等. 结合词义的文本特征权重计算方法[J]. 计算机应用，2012，32(5)：1355-1358

[5] Toutanova K, Klein D, Manning C D, et al. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network[C]//Proc of " NAACL". 2003:173-180

[6] Jensen A S, Boss N S. Dty similarity[OL]. http://damn. dk/similarity/javadoc/model/similarity/Lesk. html, 2008

[7] Lextek: Onix Text Retrieval Toolkit {API Reference. http://www. lextek. com/manuals/onix/stopwords1. html (2011)(stop word)

[8] Jiang J J, Conrath D W. Semantic Similarity Basedon Corpus Statistics and Lexical Taxonomy[J]. Proc of 10th International Conference on Research in Computational Linguistics, 1997, 19 (33)

[9] Fellbaum C. WordNet: an electronic lexical database [OL]. WordNet is available from http://www. cogsci. princeton. edu/wn, 2010

[10] Resnik P. Using Information Content to Evaluate Semantic Similarity in a Taxonomy[C]//Proc of the 14th International Joint Conference on Artificial Intelligence. 1995, 11:448-453

[11] Wu Zhi-biao, Palmer M. Verb Semantics and Lexical Selection [C]//Proc of 32nd Annual Meeting on Association for Computational Linguistics. 1994:133-138