基于DBTNN算法的场景调度策略研究

杨东梅[1,2] 印桂生[1] 赖初荣[2]
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[1] (哈尔滨工程大学科技园 哈尔滨 150001)[2]

摘 要 构建了一种能有效描述大规模虚拟环境的场景图和基于该场景图的场景多叉树,以此为基础提出了基于改进的动态二叉树神经网络(Dynamical Binary-tree Based Neural Network,DBTNN)进行场景调度的新方法,并给出了神经网络挖掘装配过程中视点变化的规律。同时,通过该网络输出预测下一步或下几步的视点状态信息,使场景调度具有一定的容错性,提高了场景调度的实时性并使调度更加智能化。最后给出了不同场景的测试结果,表明该算法对于大规模复杂场景有非常好的优化效果。

关键词 虚拟装配,场景调度,动态二叉树
中图法分类号 TP311 **文献标识码** A

Research on Scene Dispatch Strategy Based on DBTNN Algorithm

YANG Dong-mei[1,2] YIN Gui-sheng[1] LAI Chu-rong[2]
(Computer Science & Technology College, Harbin Engineering University, Harbin 150001, China)[1]
(National Science Park, Harbin Engineering University, Harbin 150001, China)[2]

Abstract A new scene graph that can effectively describe large-scale virtual environment and a scene graph based Multi Space Partition tree(MBSP) were proposed. Proposed scene dispatch strategy of Dynamical Binary-tree Based Neural Network(DBTNN), and gave the nerve network to dig the viewpoint change rule during the assembly process. At the same time, forecasted viewpoint state of the next step or the next steps by the network output, so that the scene had a fault-tolerant scheduling, improved scheduling of real-time scenes and more intelligent scheduling. Finally, the test results of different scenes show that the algorithm for large-scale optimization of complex scenes has very good results.

Keywords Virtual assembly, Scene scheduler, Dynamical binary-tree

在虚拟环境中,场景调度是虚拟场景管理的重要组成部分,对场景建模、碰撞检测和可见性剔除等问题都有重要影响。如何在内存和图形渲染能力受限的条件下实现虚拟环境的建模和交互,是大规模虚拟环境管理和实时生成技术的难点,其本质在于如何从大规模的复杂虚拟环境中根据化身的位置来构造需要渲染的场景。在基于几何模型的虚拟装配系统中,描述精细模型往往要花费几万甚至几十万个多边形,要占用数十兆磁盘空间[1]。对这样的模型进行显示,帧的数据处理量会非常庞大,甚至超过一般计算机处理能力的极限,很难达到令人满意的实时效果。同时,由于装配空间相对狭小,会发生频繁的碰撞,因此场景调度策略的合理设计对进行虚拟装配至关重要。

神经网络存储模糊信息的能力较好,可以用来学习并记忆虚拟装配过程中的信息。当训练好的神经网络稳定后,就可以利用该网络的输出信息调度装配场景。本文在建立大规模虚拟环境的场景多叉树模型的基础上,利用径向基神经网络挖掘虚拟装配过程中视点状态变化的规律,采用基于改进的动态二叉树神经网络(Dynamical Binary-tree Based Neural Network,DBTNN)进行场景调度,以提高场景显示的实时性和交互性,同时本文方法可以扩展到虚拟现实其他领域的场景调度中。

1 场景图的构建

场景管理技术按其目的可划分为面向性能和面向交互两大类。面向交互的场景管理方法主要为场景图法(scene graph)描述虚拟环境,其主要优点是以对象为单位对虚拟环境进行层次式表示,可以真实地描述虚拟环境中对象的组织结构,较容易修改虚拟场景,并能支持重用,提高图形软件的开发速度,还能方便场景数据在不同VR系统之间的交换。因此,场景图法在WTK,Java3D等VR开发工具和VR系统中得到了广泛应用。面向性能的场景管理方法主要有二叉空间剖分、八叉空间剖分、单元-入口方法。二叉空间剖分算法适用于复杂的室内场景,基于单元-入口的算法适用于多房间的室内漫游情况,基于八叉树的算法则更适用于复杂的大场景。这类方法的最大优点是剔除与碰撞干涉检测的效率较高[2]。

基于以上对虚拟环境中的对象所进行的研究,采用面向对象的方法来描述大规模虚拟环境。环境对象是由多个子空

到稿日期:2009-10-19 返修日期:2009-12-31 本文受国家自然科学基金资助项目(90718003),省科技厅攻关项目(GC05A115)资助。
杨东梅(1979—),女,博士生,主要研究方向为虚拟现实技术,E-mail:ydm411@sohu.com;印桂生(1964—),男,博士生导师,CCF会员,主要研究方向为虚拟现实与数据库技术;赖初荣(1977—),男,硕士生,主要研究方向为计算机网络与数据库技术。

间对象构成的。封闭空间有 2 个组对象，分别描述外部场景与内部场景。开放空间只有 1 个组对象，此组对象又由交换组和变换组对象组合而成，变换组对象以形体为其子对象。通过环境、空间、组、形体对象的层次组合可描述任何复杂的大规模场景虚拟环境。最终场景图构成一个多叉树，如图 1 所示。

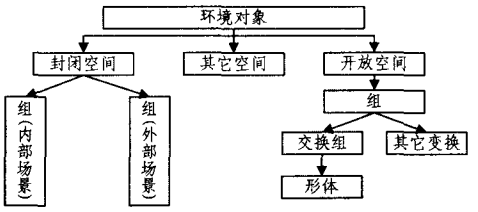

图 1　大规模虚拟环境的场景图

由于多叉树可能将一个场景对象分割到若干个子空间中，如果想获取某个场景对象所包含面的顶点坐标，就必须遍历整个多叉树，从而增加了系统的运行时间。且用多叉树剖分建立的树中虽然记录了场景中所有对象的几何信息，但是在剖分场景时将对象的几何信息无规律地切割到若干子空间中，使我们无法高效地利用对象几何信息。此外，用多叉树处理动态物体，最直观的方法是当物体运动时，将其从多叉树中删除，并插入到新的位置上。这会涉及到多叉树结构的更改，因此其实时性往往难以保证[3-5]。

对场景管理技术的研究很多，如文献[3]中构建了一种能有效描述大规模虚拟环境的场景图和基于该场景图的场景二叉树，以实现场景的管理。文献[4]中综合利用了面向对象概念和八叉树算法，构建了一种适合处理动态场景的交互树来进行场景管理。但文献[3]主要是从虚拟环境漫游的角度来考虑虚拟环境的场景管理，未深入研究虚拟环境中动态对象的管理；而文献[4]中构建的关系树依赖于场景设计者的人工干预，在处理复杂场景时工作量较大。

因此，本文设计了一种基于动态二叉树的映射树，即基于动态二叉树的神经网络（Dynamical Binary-tree Based Neural Network，DBTNN），它是一种结合神经网络和树结构层次分类的算法。构造树型结构的神经元分布，具有神经网络无监督等特性，同时具有树型层次结构的优点。

2　基于径向基神经网络的场景调度

2.1　构造径向基神经网络

神经网络通过训练可以存储环境的经验知识。这里经验知识指的是视点状态的变化规律，它以权值的形式存储在权值矩阵中。径向基函数（Radial Basis Function，RBF）网络是以函数逼近理论为基础而构造的一类前向网络，它将低维空间映射到多维空间中寻找训练数据的最佳拟合平面。在逼近能力、分类能力和学习速度等方面径向基函数均优于 BP 网络。网络的结构如图 2 所示。

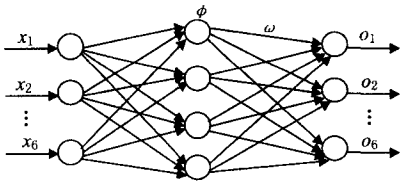

图 2　径向基函数神经网络

实例$\{(x_i,d_i)\}_{i=1}^{N}$作为训练样本，其中向量 x_i 为网络的输入，表示视点的当前实际状态；d_i 为 x_i 的实际后续状态。o_i 为网络的输出，x_i 表示预测的后续视点状态。N 为样本个数，网络只有一个隐含层。w_{ij} 为隐含层第 i 个神经元到第 j 个神经元的连接权值。

基函数采用高斯函数：

$$\phi(x,\sigma)=\exp[-||x-c_i||^2/\sigma^2] \tag{1}$$

式中，$||x-c_i||^2$ 为 x 到 c_i 的距离；c_i，σ 分别为径向基函数的中心和宽度，采用 K-均值聚类算法加以确定[6]，即

$$C_j=1/numb(\theta_j)\sum_{x_j\in\theta_j}X_j \tag{2}$$

$$\sigma=d_m/\sqrt{2M} \tag{3}$$

式中，θ_j 为聚类子集，$numb(\theta_j)$ 为 θ_j 的样本数目，d_m 为各中心之间的最大距离，M 为中心数。

收集训练样本时，假设 100 步操作可以完成某一零件的装配，则共经历了 101 个状态（包括初始状态），共产生了 100 个训练样本$\{(x_i,d_i)\}_{i=1}^{100}$，训练样本采用集中式方式通过神经网络，根据样本确定 c_i，σ。采用试凑法确定隐含层神经元个数，先设隐含层含有较少的神经元，然后逐渐增加个数，当达到网络输出精度要求后就停止，这样网络具有最小的结构。

考虑神经网络的计算成本，采用实际装配中收集训练样本、脱离环境时训练网络的方式进行训练，记录相应零件的神经网络结构和权值矩阵。训练完成后使用神经网络时，输入输出所需的时间非常小，可以满足实时性的需要。

2.2　场景调度过程

2.2.1　DBTBNN 算法的思想

DBTNN 算法采用传统的单方向搜索方式，在每个内部节点，把径向基神经网络的输入样本向量$\{(x_i,d_i)\}_{i=1}^{N}$和该节点的两个孩子节点的权重向量做比较，选择权重向量更接近输入样本向量的孩子节点作为下一步搜索节点，直到找到一个叶节点，即场景调度过程中当前视点的后续状态 d_i。

对该获胜叶节点进行分裂（splitting）或剪除（pruning）操作，从而调整树结构。若该获胜叶节点误差值较大，则该获胜叶节点将分裂生成操作新节点；同样，如果该获胜叶节点误差值太大，且树中节点数目超过预定的值，则进行剪除操作。按照这种树调整方法，不断调整连接根节点与获胜叶节点路径上的神经元的权值，以反映当前输入样本$\{(x_i,d_i)\}_{i=1}^{N}$。

通过判断总误差值 E 来确定是否结束算法：当总误差值 E 变化不大时，算法结束；否则对输入样本进行新一轮学习训练。

在某种意义上，DBTNN1 算法中用来分裂或剪除操作的叶节点总是最接近当前输入样本的获胜叶节点，这通常会形成某些叶节点总是该层次上的获胜叶节点，使树的结构严重失去平衡，很容易导致误分类或权值的调整没有向聚类中心靠近。所以本文提出了双路径搜索 DBTNN2 算法，它不是总选择最接近输入样本的叶节点作为最终获胜叶节点，而是考虑了节点所在树的层次，采用双路径进行比较搜索[7,8]。

在网络训练过程中，根据输入的样本搜索树直到找到叶节点。这时，该输入样本被映射到这个叶节点神经元，从而实现了聚类，并根据输入样本和叶节点神经元的权重向量计算该节点误差值 e_j。

2.2.2　DBTNN 算法的基本操作

(1) 生长

初始状态下，树一般只有一个根节点神经元。根据输入

的样本连续增加两个子节点，直至达到所设定的最大节点数目 M。接着，DBTNN 算法在增加新的子节点之前需要删除两个树的子节点。当获胜叶节点 J 的误差值 e_j 超过预定的分裂阈值 T 时，该节点进行分裂操作，从而生成两个新的子节点，对新生成的叶节点进行初始化：

1）初始化左孩子节点的权重向量为 θ_j，初始化右孩子的权重向量为 $(1+\lambda)\theta_j$，λ 为一个非常小的正常数；

2）每个孩子节点的误差值初始化为 $e_{j,2}$；

3）每个孩子节点的计数器初始化为 1。

与其他决策树相比，DBTNN 算法不用通过完整地搜索树的所有叶节点来确定哪个叶节点即将分裂。

（2）剪除

树通常都是以一种贪婪的方式来生长神经元，即在叶节点的误差值超过预定的分裂阈值 E 时才会最大限度地生长神经元。而当树中的神经元数目已达到预定值 M 时，可以通过剪除机制来减少错误分裂所造成的影响。从树中删除误差值最小的节点，使树中的神经元数目减少，这样树中误差值很大的获胜叶节点可以继续进行分裂来生长新的神经元节点。对于每一个训练样本，当树中的神经元数目达到预定值 M，且获胜叶节点的误差值大于预定分裂阈值 T 时，即 $e_j > T$，DBTNN 算法寻找具有最小误差值的叶节点。同时获胜叶节点 J 按原方式进行分裂，生成新的节点[9]。

2.2.3 DBTNN 算法的基本流程

DBTNN 神经网络是一种通过增量学习来调整权重向量的树型结构聚类算法，算法过程如图 3 所示。

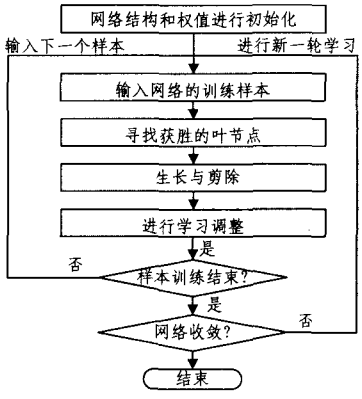

图 3 DBTNN 算法流程图

3 仿真结果

虚拟装配中场景的渲染是基于 Windows XP 平台，在主流的 PC 平台上，分辨率为 1024×768。仿真平台的配置如下表 1 所列。

表 1 仿真平台配置

配件	型号
CPU	Inter@Core(TM)2 3.00GHz
硬盘	WD180G,7200 转
内存	2.00GB
显卡	NVIDIA GeForce2 6600GT

分别对不同装配场景（主要是几何节点数量的变化）的画面帧数进行仿真测试，对装配场景分别采用线性检测和 DBTNN 算法进行测试，评估场景渲染效率的改进情况，仿真测试结果如表 2 所列。

表 2 线性检测与 DBTNN 检测的比较

场景环境的几何节点	线性检测（未采用 DBTNN 算法）的帧数	采用 DBTNN 算法的帧数	提高的效率
80 个几何节点	92fps	108fps	17.4%
260 个几何节点	54fps	76fps	40.7%
1200 个几何节点	29fps	48fps	65.5%

从表 2 中可以看出，随着装配场景中节点数量的增加，帧数提高的效率越为明显。这是因为 DBTNN 算法会使原有的节点数量增加（增加了空间节点的数量），所以对于节点数量较少的场景来说，线性检测和 DBTNN 检测的差别并不明显，但场景中几何节点一旦增加，如第三行所示的场景中几何节点数增加到 1000 个以上后，帧数竟有 50%以上的提高。这说明，本文提出的算法 DBTNN 对大规模复杂场景进行场景调度有非常好的优化效果，能够大大提高原有的渲染帧数，达到了交互的实时要求，较好地解决了场景调入的“跳闪”现象。

结束语 本文针对虚拟装配中视点信息变化的特点，在建立大规模虚拟环境场景树的基础上，设计了一种动态二叉树神经算法 DBTNN。并提出双路径搜索 DBTNN2 算法，它不是总选择最接近输入样本的叶节点作为最终获胜叶节点，而是考虑了节点在树所在的层次，采用双路径进行比较搜索。同时，在虚拟现实的其他领域，也可以应用该策略进行场景调度。

如何进一步提高场景实时渲染的速度，增强场景的真实感，并在对场景节点的各种协同浏览和协同编辑下仍能达到协同、实时，还需要我们做进一步的研究。

参 考 文 献

[1] 郝爱民，赵沁平. 一种虚拟室内场景调度策略与碰撞处理方法[J]. 计算机工程与应用，2002，7：107-111

[2] Sowizral H A, Deering M F. The Java 3 D API and Virtual Reality [J]. Computer Graphics and Applications, 2008, 19(3): 12-15

[3] 刘雁翎，诸昌铃. 一种适合处理动态场景的交互树[J]. 计算机应用，2001，21(11)：7-9

[4] 罗自荣，常明，肖人彬. 面向虚拟环境的场景管理关键技术及其实现研究[J]. 系统仿真学报，2007(6)：891-897

[5] 罗冠，郝重阳，淮永建. 虚拟现实引擎的设计与实现[J]. 计算机学报，2005，24(11)：1264-1269

[6] Jing liang. Progressive Geometry Encoder Using Octree-based Space Partitioning[C]// IEEE International Conference on Multimedia and Expo. 2004

[7] 张群洪，陈崇成. 一种改进的动态二叉树的自组织神经网络算法[J]. 计算机应用，2007(9)：2262-2265

[8] 喻罡，崔杜武，王竹荣. 虚拟漫游系统中调度算法的研究与实现[J]. 计算机工程，2002，27(12)：115-118

[9] MAO Yong-cai, HU Qi-ying. Random process[M]. Xidian University Press, 1998