Journal of System Simulation

Volume 31 | Issue 3 Article 16

11-20-2019

Dynamic Interference Detection Algorithm for Bearing with Scar

Jinyuan Jia
School of Software Engineering, Tongji University, Shanghai 201804, China;

Naihao Zheng
School of Software Engineering, Tongji University, Shanghai 201804, China;

Dynamic Interference Detection Algorithm for Bearing with Scar

Abstract

Abstract: To solve the dynamic interference detection problem of scar bearing in visual assembly, *an efficient dynamic interference inspection algorithm for scar bearing* is proposed. The first step is to perform vertex management on the 3D reconstructed scar bearing model through the quad-tree; the second step is to select the appropriate algorithm for the calculation of the maximum interference distance by considering bearing shape and its motion characteristics. The research shows that the algorithm can quickly and efficiently test the dynamic interference and calculate the interference distance for the bearing with scar, which can effectively improve the efficiency of bearing interference inspection and provide a new idea for bearing interference detection.

Keywords

bearing wear, interference detection, simulation experiment, real-time calculation

Recommended Citation

Jia Jinyuan, Zheng Naihao. Dynamic Interference Detection Algorithm for Bearing with Scar[J]. Journal of System Simulation, 2019, 31(3): 502-510.

第 31 卷第 3 期 系统仿真学报© Vol. 31 No. 3
2019 年 3 月 Journal of System Simulation Mar., 2019

带疤痕轴承的动态干涉检验算法

贾金原，郑乃豪
（同济大学软件学院，上海 201804）

摘要：为解决虚拟装配中带疤痕轴承的动态干涉检验问题，提出了一套*高效的带疤痕轴承的动态干涉检验算法*。第一步通过四叉树对三维重建的带疤痕轴承模型进行顶点管理，第二步结合轴承形态和运动特征，选择适合的算法进行最大干涉距离的计算。研究表明，算法可以快速高效准确的对带疤痕轴承的动态干涉进行检验并计算干涉距离，可以有效提高轴承干涉检验的效率，为轴承干涉检验提供了一种新思路。

关键词：轴承疤痕；干涉检验；模拟实验；实时计算
中图分类号：TP391.4 文献标识码：A 文章编号：1004-731X (2019) 03-0502-09
DOI: 10.16182/j.issn1004731x.joss.18-VR0740

Dynamic Interference Detection Algorithm for Bearing with Scar

Jia Jinyuan, Zheng Naihao

(School of Software Engineering, Tongji University, Shanghai 201804, China)

Abstract: To solve the dynamic interference detection problem of scar bearing in visual assembly, *an efficient dynamic interference inspection algorithm for scar bearing* is proposed. The first step is to perform vertex management on the 3D reconstructed scar bearing model through the quad-tree; the second step is to select the appropriate algorithm for the calculation of the maximum interference distance by considering bearing shape and its motion characteristics. The research shows that the algorithm can quickly and efficiently test the dynamic interference and calculate the interference distance for the bearing with scar, which can effectively improve the efficiency of bearing interference inspection and provide a new idea for bearing interference detection.

Keywords: bearing wear; interference detection; simulation experiment; real-time calculation

引言

列车轴承磨损是影响铁路安全运行的关键因素之一[1]。为解决带疤痕轴承的动态干涉检验问题，本文选择对车辆传动轴中出现较多的滚子轴承的干涉进行研究。滚子轴承的特点是，轴承内部的所有滚子和内外圈的轴线的延长线共面或交于一点，这种设计保证了结构的稳定性。

在使用过程中，轴承在负责传动的同时还会受到来自所承载结构重量产生的巨大压力。压力过大时，轴承部件会出现形变，甚至出现相互嵌入的情况。即轴承的部件之间发生干涉。

轴承干涉会使轴承损耗速度大幅提升，长时间的干涉还会使轴承产生疤痕。有疤痕的轴承很难保持平稳顺滑的运动，产生巨大的安全隐患。

轴切法是工业界普遍使用的解决干涉模拟实验的方法。轴切法中，待测算模型沿垂直轴线方向

收稿日期：2018-08-09 修回日期：2018-11-03；
基金项目：中央基本业务经费基金(0200219153，2100219066)；
作者简介：贾金原(1963-)，男，山东乐陵，博士，教授，研究方向为 Web Graphics、分布式虚拟现实；郑乃豪(1994-)，男，山西晋城，硕士，研究方向为 Web3D 轻量化技术。

进行细分得到薄片，再使用这些忽略厚度的薄片进行干涉检验。此方法可以得到较为准确的结果，但时间复杂度较高(O(n^2)，n 为细分粒度)，无法满足高效计算的需求。

虚拟装配技术是工业界常用的解决此类问题的技术[2]。但由于在虚拟装配中使用的模型面数较多，传统的基于空间平面的碰撞检测算法的计算效率便会大打折扣。高效率的实现干涉检验是现阶段虚拟装配技术研究的重点之一。在现有研究中，从真实世界获取的物体，通常使用点云对模型进行重建，再使用模型的点云信息对其最近距离进行计算[3]。Kim Y J 对当前在机器人领域常用的空间体碰撞检测进行了分析[4]。除此之外，还有基于物体位置信息的算法也可以高效实现动态干涉检验[5]。

只考虑轴承的形态信息，可以将其视为空间几何体。空间几何体最大干涉距离的计算，前人有了丰富的研究。Aurelien B 提出了一套空间点到几何体的解决方案[6]。空间中的柱体和锥体，空间点与其最近距离总是与轴线共面的，从而可以快速得到结果。Busé L 提出了一套使用低维数据集来计算标准柱体和锥体空间相对位置信息的方法[7]。对于空间中的可变形几何体，一般通过构建包围盒的方法来进行处理。Qian K 提出了使用球状面包围网格表面进行碰撞检测的算法[8]。而对于传统的基于包围盒的碰撞检测方法，近年的研究重点集中在通过并行优化线性算法来提高算法效率[9-10]。

由于疤痕区域的存在，现有的标准几何体碰撞检测算法直接应用于带疤痕轴承干涉检验的难度较高。而任意空间体最近距离算法，没有对模型本身的形态特征进行考虑。对于柱形轴承的标准区域计算会产生额外的开销。此外，现有的干涉检验算法，一般只考虑是否发生干涉，不对干涉位置和干涉距离进行计算。

因此，本文研究是非常必要的。本文提出了一种带疤痕轴承动态最大干涉距离算法，其特点在于可以准确高效的计算出，带疤痕轴承在运动中的干涉情况和最大干涉距离。针对该算法，本文从模型的三维重建，模型的顶点管理，模型顶点的动态干涉算法几部分入手进行了相关研究。

1 算法概述

本文提出的高效带疤痕轴承的动态干涉检验算法，主要分为初始化和动态实时计算两部分。

初始化过程分为 4 步。第一步根据模型数据进行三维重建，同时将疤痕信息添加到模型当中。第二步根据上一步建模结果，计算理想状态下可能出现最大干涉点对的模型表面区域。第三步，使用四叉树对模型顶点信息进行管理并针对计算过程进行优化。第四步，将模型放入场景，并计算模型开始运动前的最大干涉距离信息，供动态计算过程使用。

动态实时计算过程分为 2 步。第一步，利用上一时刻的最大干涉点对位置推算当前时刻的理想最大干涉区域范围，判断当前时刻的理想最大干涉区域内是否包括疤痕。第二步，根据第一步得到的结果，选择不同的算法对当前时刻最大干涉距离信息进行计算。若包括疤痕区域，使用带疤痕柱体轴承最大干涉距离算法，若不包括疤痕区域，则使用标准柱体轴承最大干涉距离算法。

2 初始化过程

2.1 问题分析

常见的柱形滚子轴承的结构如图 1 所示。通过对柱体轴承结构进行分析，柱体轴承发生干涉时，由于轴承本身的刚体特性，干涉的深度一定是远小于轴承本身的径向的。换言之，随着干涉深度的增加柱体轴承构件轴线间的距离是单调减少的。同样的，相互干涉部件表面点构成的点对，在两构件轴线所限定的区域内部，其空间距离也不断减小。当两点对呈干涉状态时，其空间距离可以看做负值。由此，本文要解决的问题由轴承部件的最大干涉距离计算转化为限定条件下的空间多面体最近距离计算问题。

DOI: 10.16182/j.issn1004731x.joss.18-VR0740

第 31 卷第 3 期 系统仿真学报 Vol. 31 No. 3
2019 年 3 月 Journal of System Simulation Mar., 2019

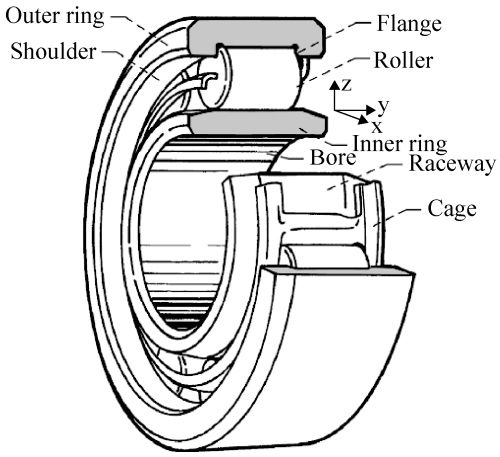


图 1 柱形滚子轴承结构图[11]

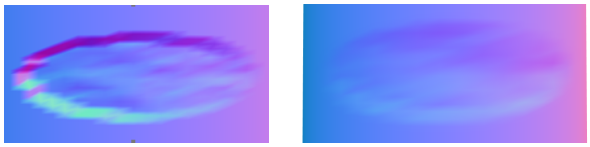
Fig. 1 Schematic diagram of cylindrical roller bearing[11]

2.2 模型重建

本文实验中使用的轴承使用网格模型进行表示。主要出于以下两方面考虑。其一，可以完整的保存顶点连接关系和拓扑结构，可以较为容易的对模型中的顶点进行访问和修改。其二，可以通过插值算法提升模型顶点数和面数。在保证形态特征和结构的前提下获得更为精确的模型。

轴承构件基础网格构建完成后，需要对轴承表面的疤痕区域进行重建。通过对大量破损轴承部件进行分析，疤痕可以抽象为轴承表面点发生在径向上的随机扰动。在多数情况下，这个扰动是向轴承构件结构内部发生的。由此，构建疤痕区域时，只需计算疤痕在轴承径向的高度图，即可准确的将疤痕在轴承模型上进行重建。疤痕区域的径向位移，可以使用 diamond-square 算法进行拟合。通过该算法可以得到疤痕区域内每个顶点的高度，即为疤痕区域顶点的径向位移。

模型构建的最后一步，需要将生成好的疤痕与轴承构件模型进行结合。这一过程分为三步来实现。第一步，确定疤痕区域顶点与模型表面顶点的映射。目的在于确定疤痕区域在模型表面的位置和形状。第二步，模型表面顶点按照对应的疤痕区域顶点的高度值，进行径向位移。最后一步，需要对疤痕区域边界进行滤波。Diamond-square 算法生成的区域在边界的变化非常剧烈。若仅简单的将轴承模型顶点进行径向位移，显然这样的结果是与实际轴承疤痕不相符的，如图 2(a)所示。为解决这一问题，使用中值滤波对疤痕边界进行处理，从而保证疤痕区域和正常轴承区域的连接平滑，如图 2(b)所示。至此，本文实验使用的模型构建完毕。



(a) 未滤波 (b) 滤波后

图 2 轴承模型疤痕滤波效果图

Fig. 2 Scar filtering effect diagram of bearing model

2.3 理想最大干涉范围计算

通过对实际轴承干涉过程的观察和分析，轴承运动时部件会在垂直轴向的平面内发生随机扰动，且在多数情况下，轴承构件的轴线是共面的。同时由于轴承的刚体特性，干涉深度远小于轴承径向距离。由此，结合经验参数，我们可以对轴承在发生干涉时可能出现干涉点的最大范围进行计算。

接下来，使用图 3 模型对理想最大干涉区域的计算过程进行描述。其中 A 和 B 为两个柱型轴承部件，高度分别为 h_a 和 h_b，半径分别为 r 和 R。两端面圆心分别为 T_aB_a 和 T_bB_b。

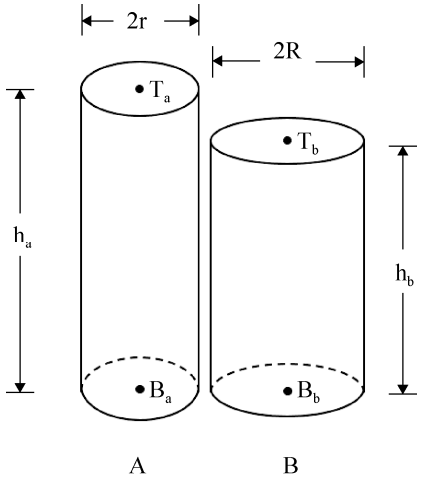
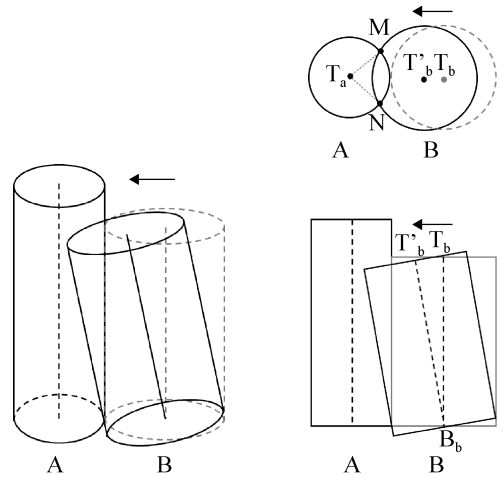


图 3 柱型轴承模型示例

Fig. 3 Cylindrical bearing model example

轴承在正常运动时，轴线是平行的。出现偏转

时，两部件轴线会产生夹角 α。α 即为偏转角，可以量化的表现出当前状态下的干涉情况。在图 4 中，偏转角为 $\angle T`_b B_b T_b$。产生干涉时，轴承 B 向轴承 A 发生偏转，如图 4(b)所示。达到极限状态时，α 也达到最大值，此时 α 为最大偏转角。通过对轴承干涉的实际情况进行分析，α 的角度非常小。由此，我们可以认为干涉的深度为 $h_b \tan\alpha$，即 $T'_b T_b$ 的长度为 $h_b \tan\alpha$。且偏转后的轴承 B 的顶面在俯视图中的投影依然为圆。



(a) 立体图 (b) 俯视图和正视图

图 4 理想最大干涉范围示意图

Fig. 4 Ideal maximum interference range

对图 4(b)中俯视图进行分析。明显的，对于轴承部件干涉最大距离所在点对，一定不会超过 MN 所界定的柱面区域。柱面区域的范围可通过 $\angle MCN$ 的大小进行量化。在三角形 $MCT`_b$ 中，$\angle MCT`_b$ 是 $\angle MCN$ 半角，MC 的长度为 r，$MT`_b$ 的长度为 $R$，$CT`_b$ 的长度为 $(R+r)-h_b\tan\alpha$。令 $\angle MCN=\theta$，根据余弦定理：

$$\theta = 2\arccos\left(\frac{\sin^2\alpha h_b^2 - 2\sin^2\alpha h_b(R+r) + 2R(R+r)}{2Rr}\right) \quad (1)$$

在计算时，只需将模型参数信息代入上式，即可获得理想最大干涉范围。在实际的计算过程中，每一时刻的计算只需考虑理想最大干涉范围内的顶点，将计算范围从模型的所有顶点降低到范围内的顶点。提升了算法效率。

2.4 顶点四叉树

对于任何一个二维平面，都可以使用四叉树对其顶点进行有效的数据管理。树的深度，决定了二维平面数据的划分粒度。同样的，对于二维的网格，亦可使用四叉树进行管理。提高对模型内部顶点信息的访问速度的同时，可以保存更多的模型信息。

构建四叉树的方法比较简单，在此不再赘述。需要着重说明的是，为保证存储内容不产生冗余，在进行划分时，尽量使得四叉树对应区域不与模型本身的网格重合。此外所有顶点都被添加到顶点树中后，需要对位于疤痕区域的所有叶子节点进行标记，所有包含疤痕子节点的根节点也需要被标记。至此，顶点树的构建初步完成。

每步计算过程中，都对完整的顶点树进行遍历，无疑会造成巨大的额外开销，降低算法的效率。为解决这一问题，在顶点树构建完成后，还需要针对计算过程进行结构优化。

第一步为划分计算层。通过计算得到的理想最大干涉范围，可以确定每一时刻最大干涉点对出现的范围。由此，在进行最大干涉距离的计算时，只需从所有叶子节点都在理想最大干涉范围内的最高层根节点开始计算即可。本算法中，这样的节点被称为计算节点。包含计算节点的四叉树层次为计算层。对于柱形轴承构件，最大理想干涉范围是确定的，且四叉树划分是均匀的。因此计算层可以在开始实时计算过程前通过计算进行确定。

第二步为抽象节点计算。在计算过程中，对较高层节点对应区域进行比较计算，粗略的结果即可满足需求。如果对其所有子节点进行分析，无疑会造成巨大的计算开销。为解决这一问题，顶点树处理的第二步，对计算层以下的所有非叶子节点进行抽象，使用其四个子节点取均值，得到当前父节点的虚拟顶点坐标。此时，每一个非叶子节点都可以视作一个独立的子节点。同层内的比较计算开销大大下降。通过这一过程，在保证模型形态结构的情况下，有效的提升了计算速度。至此，本文算法所

DOI: 10.16182/j.issn1004731x.joss.18-VR0740

第 31 卷第 3 期 系统仿真学报 Vol. 31 No. 3
2019 年 3 月 Journal of System Simulation Mar., 2019

使用的四叉树构建完成。

2.5 初始状态最大干涉距离计算

在进行动态干涉检验之前，需要对初始状态下的轴承最大干涉距离进行计算，以供动态计算第一时刻进行使用。此时没有前一时刻的结果参考，因此需要对轴承构件的整体构件进行分析，具体步骤如下：

输入：模型顶点树。

输出：最大干涉距离及点对。

Step 1 假定当前模型均为标准柱形，使用标准柱体的最大干涉距离算法求解。

Step 2 根据结果判定当前时刻的理想最大干涉范围。

Step 3 遍历范围内的所有顶点，查看是否有疤痕节点

1) 若有疤痕节点，执行下一步。

2) 若无疤痕节点，直接第一步计算得到的最大干涉点对。

Step 4 对疤痕区域重新计算，得到初始状态下的最大干涉点对。

3 动态实时计算

3.1 基于模型运动的算法选择

考虑一种较为简单的情况。完好无疤痕的柱形轴承，以顶面底面圆心为轴进行旋转的模拟实验时。最大干涉距离点对所在空间位置不会随着轴承的运动发生变化。但点对在轴承表面的位置发生了变化。此时，可以通过轴承旋转的角度计算出任意时刻的最大干涉距离点对。

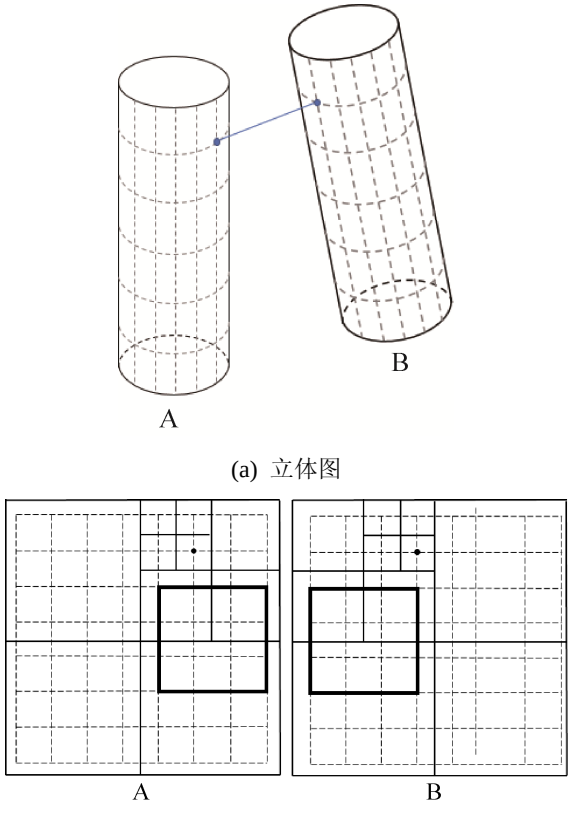
当柱形轴承增加在垂直轴向的随机扰动时，直接对某一时刻的最大干涉点对进行计算较为困难。但根据理想最大干涉范围，依然可以确定当前时刻最大干涉点对出现的可能范围。此时，若范围内包含疤痕区域，则使用基于时序的递归算法进行计算，若不包含疤痕区域，则使用标准柱体最大干涉距离算法进行计算。

3.2 基于时序的最大干涉距离算法

通过上一时刻的最大干涉点对信息，可以确定当前时刻最大干涉点对可能出现的区域。这一区域在顶点树上对应的计算层节点即为当前时刻的待测算节点。对所有的待测算节点进行计算，即可求解得到当前时刻的最大干涉点对及干涉距离。接下来结合示例对算法进行介绍。

图 5 中展示了动态实时计算中当前时刻的初始状态。左侧为柱体的实际效果，右侧为侧面展开后的局部顶点和四叉树信息。

图 5 中黑点表示上一时刻的最大干涉距离点对所在位置。网格图中的粗线框为疤痕顶点所在区域。



(a) 立体图

(b) 侧面展开图

图 5 动态实时计算起始状态示意图

Fig. 5 Dynamic real-time calculation start state diagram

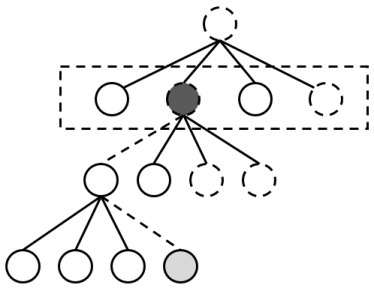
获取上一时刻最大干涉点对信息后，结合顶点

四叉树开始计算。四叉树中虚线圆表示当前节点对应模型表面区域有疤痕。深灰色填充表示待计算节点，浅灰色填充表示前一时刻的最大干涉点对所在位置。黑色填充表示当前时刻的最大干涉点对所在位置，虚线框表示所有的计算层节点。以图 5 中柱形轴承 A 为例展示计算过程。

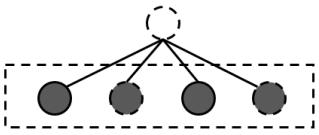
第一步，根据上一时刻最大干涉点对所在区域标定当前时刻计算层待测算节点位置。如图 6(a)所示虚线路径。

第二步，根据理想最大干涉区域，标记其他所有可能出现最大干涉距离的节点。

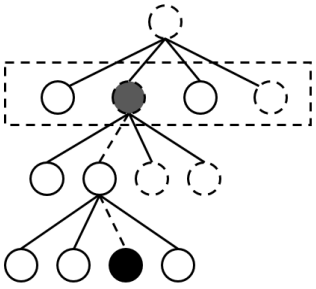
第三步，对所有的待测算节点进行计算。从计算层开始，直到叶子节点层，得到当前时刻最大干涉点对。如图 6(c)所示虚线路径。



(a) 第一步



(b) 第二步



(c) 第三步

图 6 动态实时计算示意图

Fig. 6 Dynamic real-time calculation process diagram

3.3 标准柱形轴承的最大干涉距离算法

当两柱形轴承当前时刻的理想最大干涉区域都不包含疤痕时，可以将直接使用标准柱形轴承的最大干涉距离算法进行计算。对于柱体间的最大干涉距离计算，根据柱体相对位置的不同，可能会出现多种情况。根据轴承结构和运动规律，对其进行分析。

(1) 当两柱体相离，即没有互相干涉时。此时最大干涉距离变为两圆柱最近距离，且最近距离点对出现在非端面圆周的柱面上。该连线会经过两圆柱轴线，同时这条线段所在的直线与轴线的交点也是轴线最近距离的所在位置。

(2) 当两柱体相离。且最近距离的点对出现在一个或者两个圆柱的端面圆周上时，此时的最近距离的计算变为端面圆环到柱体侧面或者两端面圆环之间的最近距离。实际上也会出现端面圆环到端面的情况，但是不会出现在运动模拟的过程中，此处不予考虑。

(3) 两柱体相互干涉时，上述两种情况也可能发生。由于轴承的刚体特性，当最大干涉距离点对出现在柱面时，依然可以看作是两轴线的最近距离所在的直线与柱面的交点。

(4) 若干涉时最大干涉距离出现在端面圆周，从干涉的定义出发，需要计算的内容即为圆柱端面圆环到柱面的距离[12]。

由此，本算法提供了一套具有鲁棒性的柱形轴承最大干涉距离的解决方案，流程如下：

输入：模型数据信息。

输出：最大干涉距离。

Step 1 计算两柱形轴承构件轴线段的最近距离。

Step 2 法向量验证。

1) 如果两柱形轴承构件轴线段最近距离所在直线与其相互垂直，此时最近距离所在直线与两柱形轴承构件侧面的交点即为最大干涉点对。执行 Step 4。

2) 如果轴线最近距离所在直线方向向量与两轴线段不垂直，执行下一步。

DOI: 10.16182/j.issn1004731x.joss.18-VR0740

第 31 卷第 3 期 系统仿真学报 Vol. 31 No. 3
2019 年 3 月 Journal of System Simulation Mar., 2019

Step 3 使用四个端面分别计算到另外一个轴承轴线的最大干涉点对。

Step 4 得到当前时刻最大干涉点对。

4 实验设计

4.1 实验数据结构

实验中，需要用到两组数据，对其结构进行简要分析。

第一组数据为模型结构和顶点数据，结构如下所示：

bearing.parameters = {radius, height, radialSegments, heightSegments, position, rotation, vertices,};

radius 保存柱体轴承底面半径，为 float 型变量；height 保存柱体轴承底面高度，为 float 型变量；radialSegments 保存柱体轴承圆周细分粒度，为 int 型变量；heightSegments 保存柱体轴承高度细分粒度，为 int 型变量；position 保存柱体轴承空间位置，使用保存 xyz 三维坐标的三维向量表示；rotation 保存柱体轴承空间旋转信息，用三维向量保存，分别保存了柱体轴承沿 xyz 三坐标轴旋转的角度；vertices 保存柱形轴承模型顶点坐标，使用保存 xyz 三维坐标的三维向量表示。

第二组数据为顶点四叉树信息，结构如下所示：

quadTree.parameters={scarLabel, calLevelLabel, parent, child, corner, visualVertex,};

scarLabel 为节点是否为疤痕节点的标记信息，为 bool 变量；calLevelLable 为节点是否为计算层节点的标记信息，为 bool 变量；parent 保存当前节点的父节点信息，为顶点索引，若根节点不存在父节点，此属性为空；child 保存当前节点的子节点信息，为长度是 4 的数组，数组中保存子节点的索引，若叶子节点不存在子节点，此属性为空；corner 保存父节点对应区域的四角顶点的信息，为长度是 4 的数组，数组中保存顶点索引和坐标信息，叶子节点不存在对应区域，此属性为空；visualVertex 保存节点的虚拟节点信息，若不存在虚拟节点，此属性为空。

4.2 算法复杂度分析

动态计算过程中，根据情况会采用不同的算法，在此分别进行讨论。对于标准柱形轴承的最大干涉距离，无论模型顶点数目，只需要进行一次计算即可得到结果。此时，算法复杂度为 O(1)。对于基于时序的最大干涉距离算法，每一时刻需要选取理想最大干涉范围中的顶点进行计算。最坏的情况是，计算层位于根节点的下一层。此时，每一层需要进行 42 即 16 次计算，需要计算的层数为四叉树的总层数-1。四叉树的总层数可以通过柱形轴承细分粒度进行计算，即(n-1)。由此可得，此时算法复杂度为 O(16*(n-1))。当四叉树层数较多时，可以认为当前状态下的算法复杂度为 O(n)。

相较于传统的轴切法，本算法在最不理想的情况下，也可以保证 O(n)的复杂度，远低于轴切法的 $O(n^2)$。

4.3 对比实验设计

在之前的研究中，还没有针对带疤痕的柱形轴承的最近距离的计算有较为高效和成熟的算法。由此，实验设计中，将本文算法与近似情景中较为成熟的算法进行了对照试验，测试算法的性能。试验主要分为三部分进行。

第一部分，将本算法与参考文献[6]中的算法进行对照试验，对本算法在单次计算最大干涉距离时的效率进行评估。

第二部分，将本算法与参考文献[6]中的算法在精确度上进行比较。构建位置固定的轴承模型对后，调整轴承模型的顶点数量，对最大干涉距离的结果进行评估。

第三部分，将本算法与几个较为成熟的实时碰撞检测算法进行对照试验，对本算法在实时计算过程中的效率进行评估。常用于碰撞检测的 BVH 算法效率极高，但在本实验中不适用。若对模型区域

使用包围盒进行计算，很难反映出模型本身的形态特征。若构建顶点级的包围盒，在单次计算最大干涉距离计算过程中计算次数过多。故没有对 BVH 算法进行比较。

试验设备 CPU 为 i7-6700K，16G 内存，显卡为 NVIDIA 1070 8G。第一部分实验使用 JavaScript 进行实现。第二部分实验使用 C++进行实现。

4.4 实验结果分析

表 1 为第一部分实验结果，通过比较分析可知。本文所提出的算法在计算效率上，较原有算法有一定的提升，单次计算的时间开销有一定的优化。

表 1 单次计算所用时间
Tab. 1 Times spent in a single calculation /ms

模型顶点数	基础算法	文献[6]算法	本文算法
32^2	7.617 5	1.033 8	0.8
36^2	12.447 5	1.659 3	0.822 5
40^2	18.81	2.478 8	0.835
44^2	27.11	3.543 1	0.945
48^2	38.455	4.990 6	1.287 5
52^2	52.597 5	6.790 3	1.34
56^2	70.517 5	9.064 8	1.535
60^2	92.13	11.803 3	2.147 5
64^2	119.722 5	15.291 9	3.705

表 2 为第二部分实验结果。通过比较分析可知，本文算法计算结果精度受模型的顶点数目影响较大。当实验模型顶点较少，较为粗糙时，模型中两相邻顶点的距离较大。最大干涉点对出现在模型表面非顶点的区域时，便会产生误差。随着模型顶点数的上升，这一影响逐渐减弱。

表 3 为第三部分实验结果，其中蛮力法作为测试的基准数据。现有算法和本文算法都可以高效的得到场景内物体的干涉检验情况。现有算法在碰撞体数量较小时可以达到很高的计算效率。随着场景内碰撞体数量提升，对比试验中所选择的两种碰撞检测算法效率大幅度下降。本文提出的算法，随着模型的复杂度上升，帧数变化较为平缓。

表 2 计算结果比较
Tab. 2 Comparison of calculation results /mm

模型顶点数	基础算法	文献[6]算法	本文算法
32^2	40.239 5	40.292 3	40.360 1
36^2	40.216	40.253 2	40.314 5
40^2	40.203 6	40.234 7	40.288 7
44^2	40.191 7	40.202 8	40.255
48^2	40.173	40.191 2	40.224 1
52^2	40.158 5	40.173 1	40.192 2
56^2	40.140 4	40.143 7	40.163 6
60^2	40.122 9	40.129 1	40.138 3
64^2	40.107 8	40.109 5	40.112 1

表 3 动态计算显示帧数
Tab. 3 Dynamic calculation /fps

模型顶点数/碰撞体数量	基础算法	本文算法	Sweep and prune	Loose octree
32^2/256	54	60	60	60
36^2/324	41	57	60	60
40^2/400	27	56	60	60
44^2/484	19	56	60	60
48^2/576	14	55	60	60
52^2/676	10	55	60	53
56^2/784	8	54	60	45
60^2/900	6	53	56	39
64^2/1024	5	53	45	35

4.5 改进与优化

通过对实验结果的分析，本算法相较于传统的最大干涉距离计算方法有着巨大的性能提升。同时也暴露出一些问题。其一，当前算法仅在计算标准柱形轴承最大干涉距离时可以达到较高效率。当轴承结构为复合结构或圆台时，计算效率会有所下降，并且基于轴线的计算方式不再适用。在接下来的工作中，将尝试对现有的方法进行扩展，对圆台形状的轴承动态干涉的高效计算方式进行研究。

此外，本文算法可以从两部分入手，进行并行优化。其一，在基于时序的最大干涉距离算法中，四叉树每一层顶点进行的计算都是相互独立的，在保证所有数据只保存一次的情况下，可以同时进行 4 对顶点的最大干涉距离计算。并行后，寻找当前时刻最近点对的计算效率可以提升 4 倍。其二，在实际的应用场景中，轴承中需要进行干涉检验的构

第 31 卷第 3 期 系统仿真学报 Vol. 31 No. 3
2019 年 3 月 Journal of System Simulation Mar., 2019

件对数量较多，且运动状态相互独立。将同一时刻发生的相互独立的干涉检验流程并行执行，可以进一步有效的提升算法效率。

5 结论

为了解决带疤痕柱体轴承的干涉检验问题，本文提出了一套基于带疤痕柱体模型的动态干涉检验算法，对模型的顶点组织结构和高效干涉检验算法开展了相关研究，通过设计实验以及与其他同类型算法对比，证明了算法的效率和准确性。主要得出以下结论：

1) 本文算法相较于传统轴切法在单次计算时间上有着巨大提升。

2) 本文算法处理粗糙模型时结果准确性会受到影响，但当模型精度上升时，计算结果准确性会大幅上升。

参考文献：

[1] Cao H, Fan F, Zhou K, et al. Wheel-bearing fault diagnosis of trains using empirical wavelet transform[J]. Measurement (S0263-2241), 2016, 82: 439-449. Doi：10.1016/j.measurement.2016.01.023

[2] 郑轶, 宁汝新, 刘检华, 等. 虚拟装配关键技术及其发展[J]. 系统仿真学报, 2006, 18(3):649-654.
Zheng Yi, Ning R X, Liu J H, et al. Survey on Key Techniques of Virtual Assembly[J]. Journal of System Simulation, 2006, 18(3): 649-654.

[3] Schauer J, Nüchter A. Collision detection between point clouds using an efficient kd tree implementation[J]. Advanced Engineering Informatics (S1474-0346), 2015, 29(3): 440-458.

[4] Kim Y J, Lin M C, Manocha D. Collision Detection[C]// Goswami A., Vadakkepat P. Humanoid Robotics: A Reference. Dordrecht: Springer, 2017: 1-24.

[5] Bender J, Müller M, Otaduy M A, et al. A survey on position‐based simulation methods in computer graphics[C]// Computer graphics forum. 2014, 33(6): 228-251.

[6] Aurelien B, Eric G. Fast Distance Computation Between a Point and Cylinders, Cones, Line-Swept Spheres and Cone-Spheres[J]. Journal of Graphics Tools (S2165-3488), 2004, 9(2): 11-19.

[7] Busé L, Galligo A, Zhang J. Extraction of cylinders and cones from minimal point sets[J]. Graphical Models (S1524-0703), 2016, 86: 1-12.

[8] Qian K, Yang X, Zhang J, et al. An Adaptive Spherical Collision Detection and Resolution Method for Deformable Object Simulation[C]// Computer-Aided Design and Computer Graphics (CAD/Graphics), 2015 14th International Conference on. IEEE, 2015: 8-17.

[9] Wong T H, Leach G, Zambetta F. An adaptive octree grid for GPU-based collision detection of deformable objects[J]. The Visual Computer (S1432-2315), 2014, 30(6/7/8): 729-738.

[10] Cai P, Indhumathi C, Cai Y, et al. Collision Detection Using Axis Aligned Bounding Boxes[M]// Simulations, Serious Games and Their Applications. Singapore, Springer, 2014:1-14.

[11] Oswald F B, Zaretsky E V, Poplawski J V. Interference-fit life factors for roller bearings[J]. Tribology Transactions (S1547-397X), 2009, 52(4): 415-426.

[12] David E. Distance to Circles in 3D [EB/OL]. (2015-05-31) [2018-06-29]. https://www.geometrictools.com/ Documentation/DistanceToCircle3.pdf.

