曲面上任意两点的近似最短路径算法研究

余晓容 杨晓东 申长雨
(郑州大学工学院橡塑模具国家工程研究中心,郑州 450002)

摘 要 为了提高曲面上任意两点间近似最短路径的计算效率,提出了求解曲面上任意两点间近似最短路径的算法,该算法首先利用三角形网格模型表示曲面,并形成相应的带权图结构,然后采用 FSPA(快速最短路径法)动态计算带权图上两点的最短路径,再通过迭代细分最短路径周围的三角形网格上的边,最后由这些边构造新的子图来不断逼近曲面上两点间的最短路径。为验证该算法效果,还给出了该算法两个应用实例。应用结果表明,该算法效率高,容易实现,并可用网格尺寸和细分参数 γ 来控制近似精度。

关键词 曲面 三角形网格模型 最短路径
中图法分类号:TP391.72 **文献标识码**: A **文章编号**: 1006-8961(2005)07-0900-05

Approximate Shortest Path on a Curve Surface

YU Xiao-rong, YANG Xiao-dong, SHEN Chang-yu
(*NERC of Mold and Die, Zhengzhou University, Zhengzhou* 450002)

Abstract A new algorithm is proposed for calculating the approximate shortest path on curve surface which can be represented by a triangle mesh model and form a weighted graph. This method mainly use FSPA (faster shortest path algorithm) to calculate shortest path on weighted graph on the basis of edge subdivision technology. By iteratively subdividing triangle edge that adjoins the shortest path and constructing new subgraph, a shortest path of high approximation accuracy can be achieved. The algorithm is efficient and easy to be implemented. The approximation accuracy can be improved by adjusting the mesh size and edge subdivision parameter γ. Two applications of the algorithm are demonstrated.
Keywords curve surface, triangle mesh model, shortest path

1 引 言

随着计算机辅助几何设计的发展,各种曲面造型广泛应用于航空、航天、造船、汽车、模具等各项领域,但由于曲面本身的复杂性,因此在实际工作中很难对曲面,特别是带权的曲面进行测量。例如在塑料制件的设计中,由于通常要根据估算的制品最大流长比(流动长度和厚度的比值)来选择相应的塑料材料或更改制品设计,使其满足可成型性[1],因此不可避免地要遇上计算两点之间的最短路径问题,特别对于测量凹凸曲面上两点间最短路径很困难,因为对于凹曲面上两点的最短路径无法通过两点牵线来测量,即使对于凸曲面,由于估算流长比时,还要考虑厚度的因素,两点牵线也不准确。

考虑到三角形网格模型是常用于表示复杂曲面的模型,本文采用三角形网格模型来表示曲面,这样就把曲面上任意两点间的最短路径问题转变成求解三角形网格模型上两点间的最短路径问题。目前在三角形网格模型上求解最短路径的算法,按精度可分为精确的最短路径算法[2,3]和近似最短路径算法[4~6]两类。由于精确最短路径算法在时间和空间上要耗费很高的代价,如 Sharir 等提出的算法时间复杂度为 $O(n^3\lg n)$[2],但在很多实际应用中并不要

基金项目:国家高技术研究发展计划"863"项目(2002AA336120)
收稿日期:2004-02-18;**改回日期**:2005-01-19
第一作者简介:余晓容(1974~),女,讲师。2004 年获郑州大学材料加工工程专业博士学位。主要研究方向为塑料成型过程的计算机辅助分析与优化设计。已发表论文 20 余篇,其中被 EI 收录 6 篇,SCI 收录 1 篇。E-mail: xryu@zzu.edu.cn

求精确值,因此本文关注近似最短路径算法,即以精度的降低来换取时间和空间复杂度的降低。通常求解近似最短路径的算法是利用边细分技术,即先通过对模型的多边形进行细分,然后构造细分后的带权图,再利用带权图两点间的最短路径算法来进行求解。各算法的差别主要在于带权图上最短路径算法、细分方法和带权图的构造过程不同,以及是否迭代求解。比较优秀的近似最短路径算法为 Kanai 等提出的算法[4],该算法的基本思想是,首先对网格模型进行细分加密,并由新边、新点和原来模型上的节点生成带权图 G,然后采用 Dijkstra 算法计算 G 上两点间的最短路径,再由最短路径经过的点和边以及相邻边构成子图 $G^{(1)}$,最后通过迭代细分最短路径所在子图上相邻的原三角形网格模型的边来构成新的子图,以便不断缩小搜索范围,把宽带变成细带,最终成为直线。该算法的不足在于近似精度在很大程度上取决于第 1 次构成的子图 $G^{(1)}$,若近似路径有一部分不落在图 $G^{(1)}$ 上,则无论如何调整边的细分参数都无法得到令人满意的近似解,而且由于它先对整个网格模型加密,因此增大了计算规模,影响了计算效率。

本文给出的三角形网格模型上两点间的近似最短路径算法,沿用了文献[4]的细分及迭代逼近的基本思想,即通过对近似最短路径周围的三角形网格的边不断细分,将细分的网格生成新的子图,迭代地在子图内求解近似最短路径。但采用的带权图上最短路径的算法、三角形细分方法和带权图的构造过程与文献[4]和文献[6]介绍的均有所不同,它更容易实现,而且效率更高。

2　最短路径计算

本算法的基本思路是:首先采用三角形网格生成软件来生成曲面的三角形网格模型,再由三角形网格模型上的节点和边来构成图,并以曲面的特性构成边的权值(以计算塑件流长比为例,将边所在单元的厚度平均值作为边的厚度,以边的厚度的倒数作为边的权值),进而生成一个带权图 G;然后采用 FSPA(faster shortest path algorithm)法动态计算带权图上任意两点之间的最短距离来得到最初近似最短路径;最后通过细分最短路径周围的三角形网格边来生成新的带权子图 $G^{(1)}$,再次调用 FSPA 法计算新的最短路径,该迭代过程不断反复,使得近似最短路径最终趋向真实最短路径。

2.1　带权图中两点间的最短路径

关于单源最短路径的算法,比较经典的是 Dijkstra 算法,它的时间复杂度是 $O(n^2)$,n 为图中的节点数。在实际应用中,通常是应用以邻接表为存储结构的改进的 Dijkstra 算法,它的时间复杂度为 $O(N_{edge}\lg(N_{edge}))$[6],N_{edge} 为图 G 的边数。考虑到在搜索近似最短路径的过程中,要不断进行带权图上两点间的最短路径计算,因此提高带权图上两点间最短路径算法效率可显著提高近似最短路径算法效率。据此,本文提出 FSPA 法,并以邻接表作为图的存储结构,采用动态优化思想[7]计算图 G 中的起始点 v_{start} 到终点 v_{end} 之间的最短路径和最短距离。算法步骤如下:

(1) for all v in V, $L(v)\leftarrow\infty$, $LabelP[v]\leftarrow$ false, $Path[v]\leftarrow -1$

(2)把 v_{start} 放入空队列 ***Queue*** 中,$LabelP[v_{start}]\leftarrow$true

(3)While 队列 ***Queue*** 不为空 do begin

(4)从队列 ***Queue*** 中取出队首元素 v_i

(5)$LabelP[v_i]\leftarrow$false

(6)for each $v_j\in AdjointP[v_i]$ do

(7)if $(L(v_j)>L(v_i)+w(v_i,v_j))$ then begin

(8)$L(v_j)\leftarrow L(v_i)+w(v_i,v_j)$, $Path[v_j]\leftarrow v_i$

(9)if($LabelP[v_j]$ = false) then begin

(10)把 v_j 放入 ***Queue*** 的队尾,$LabelP[v_j]\leftarrow$true

end

end

end

其中,v 为节点;V 是节点的集合;***LabelP*** 为标记节点 v 是否在队列的数组;***Queue*** 为先进先出队列;L 数组用于存放从源点到各点的带权最短路径长度值;***Path*** 数组记录了对应节点的上一节点标号,由终点 v_{end} 反推可以得到最短路径;***AdjointP*** 为节点的邻接表;$w(v_i,v_j)$ 为节点 v_i 到节点 v_j 的带权长度值,如果 v_i 到 v_j 之间没有边相连,则 $w(v_i,v_j)=\infty$;否则为两点的边长与权重的乘积。本算法构造了一个先进先出的队列 ***Queue*** 用来存放待优化的点,并通过顺序取出队首点,以队首点 v_i 的当前路径长度值 $L(v_i)$ 去优化 v_i 节点邻接数组 $AdjointP[v_i]$ 中各点的路径长度值 $L(v_j)$。若 $L(v_j)$ 的值变小,且标记值 $LabelP[v_j]$ 为 false,则把 j 放入队列尾部。这样不断从队列 ***Queue*** 中取出点进行优化,直至不存在可优

化的点,即此时队列为空,就得到源点到其他各点的最短路径值。

FSPA 算法的时间复杂度为 $O(N_{edge})$。由于在实际的应用中,总有 $N_{edge}<10n$,所以一般 $N_{edge}\ll n^2$。可见 FSPA 法的时间复杂度远低于 Dijkstra 算法的时间复杂度 $O(n^2)$。本文以一个节点数为2 875,单元数为5 560,边数为8 434的平板的三角形网格模型为例,采用 Dijkstra 算法计算 188 对点的最短路径,一共花费了 70.23s,平均计算一次花费 0.373 5s,而采用 FSPA 算法,在相同机型、相同条件下,一共花费了 1.203 2s,平均花费时间为 0.006 64s,约为 Dijkstra 算法的 17‰,而对于大型网格,它的效率更远远优于 Dijkstra 算法。

2.2 子图生成

对带权图 G 采用 FSPA 法得到的近似最短路径实际上是三角形网格模型中首尾相连的边的集合,而并不是实际网格面上的最短路径。图 1 的实线部分是在图 G 中采用 FSPA 算法得到的近似路径,它与虚线表示的实际路径还有很大差距。因此,本文沿用文献[4]的基本思想,将最短路径所经过的顶点周围的三角形单元的边进行细分,并由细分后得到的顶点、边和最短路径所经过的顶点、边来形成新的子图,然后调用 FSPA 算法计算子图中的新的最短路径和最短路径值,这个迭代过程不断进行,直到满足收敛条件:

$$|L^{(k)}-L^{(k+1)}|<\varepsilon$$

其中,ε 是容许误差,由用户给出。此时得到的最短路径为模型上两点间近似最短路径。下面给出根据子图 $G^{(k)}$ 的最短路径 $P^{(k)}$ 产生子图 $G^{(k+1)}$ 的方法。

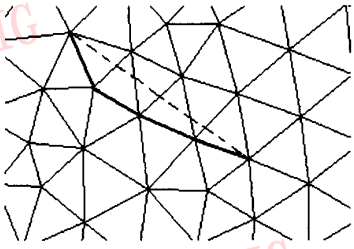


图 1 最初近似最短路径

Fig. 1 Initial approximate shortest path

(1)将子图 $G^{(k)}$ 的最短路径 $P^{(k)}$ 所经过的节点加入到子图 $G^{(k+1)}$;

(2)对于 $P^{(k)}$ 上任意节点 v_i,若其相邻点 v_j 满足 $v_j\in G^{(k)}$,且边 $e=\{v_i,v_j\}$ 满足 $Original(e)\neq -1$,则把点 v_j 和边 e 加入子图 $G^{(k+1)}$,并做上相应的标记 $LableE[Original(e)]=1$;

(3)对于最短路径上任意节点 $v_i\in G$,若其相邻点 v_j 满足 $v_j\in G$,且边 $e=\{v_i,v_j\}$ 满足 $LableE[Original(e)]=0$,则把点 v_j 和边 e 加入子图 $G^{(k+1)}$;

(4)用计数器 K 记录此时子图 $G^{(k+1)}$ 的边数;

(5)顺序取出子图 $G^{(k+1)}$ 中的边 $e=\{v_i,v_j\}$,再对边 e 进行细分产生新的节点和边,并把这些新边和点加入到子图 $G^{(k+1)}$ 中,同时从子图中删去边 e;

(6)计数器 K 减一,若 K 不为零,则返回步骤(5);

(7)对于节点对 $\langle v_i,v_j\rangle\in G^{(k+1)}$,当 $e=\{v_i,v_j\}\notin G^{(k+1)}$,并且 v_i、v_j 是同一三角形网格上不同边上的点时,则把边 e 加入子图 $G^{(k+1)}$;

其中,$G^{(k)}$ 为第 k 次迭代生成的子图,$G^{(0)}=G$;若边 e 在图 G 的边上,则函数 $Original(e)$ 返回边 e 在图 G 上的边号,否则返回 -1。***LabelE*** 数组标记图 G 上的边是否已加入子图。

文献[4]生成的新节点和边都在上一子图范围内,这样带来的问题是:求解近似路径的结果完全依赖第 1 次得到的子图。本文给出的新节点生成方法由于考虑了最短路径上的点为原图 G 上的点的情况,从而有效地克服了文献[4]的不足,且简化了文献[6]提出的针对陷阱点的处理对策。图 2 给出了由步骤 1 ~ 步骤 5 得到的子图点集示意图,其中,边的细分数为 3。粗黑实线为由图 G 得到的最短路径 $P^{(0)}$,实心点为由最短路径 $P^{(0)}$ 得到的新节点;虚线为由阴影表示的子图 $G^{(1)}$ 得到的第 2 条最短路径 $P^{(1)}$,实心矩形为由 $P^{(1)}$ 得到的新节点。

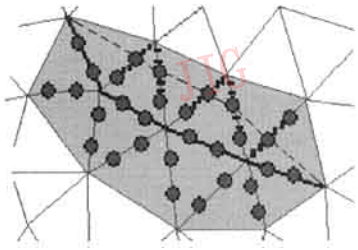


图 2 子图点集生成示意图

Fig. 2 The generation of subdividing vertices

在步骤 5 中细分边 e 时,其所生成新的节点数直接由用户指定的参数 γ 决定,即生成 $\gamma-1$ 个沿边 e 均匀分布的新的顶点。若对近似程度要求高,则

增加 γ 值,这将使每条边上的细分点增加,这样相应的每一步迭代效率要低。考虑到一般网格模型中出现狭边的机会不多,且各边生成的节点数基本一致,所以本文在生成新的顶点时,直接根据用户指定的 γ 值来生成新的顶点,这样就可以提高计算效率。图 3 给出了子图边集的示意图,图中点 A 和点 B 为图 $G^{(k+1)}$ 的最短路径上的点,且 A、B、C 属于原图 G 上的节点,AD、DE、EB、AF、FG、GC 为原边上新生成的边,细实线为由不在同一条边上细分点相连生成的边。文献[6]中,为了缩减子图规模,采取不增加边 AD、DE、EB、AF、FG、GC 的策略,这实际上并未达到缩减子图规模的目的,因为节点总数并未减小,而且还要相应地增加一些辅助操作来记录这些信息。

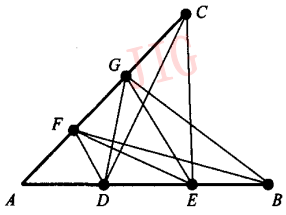
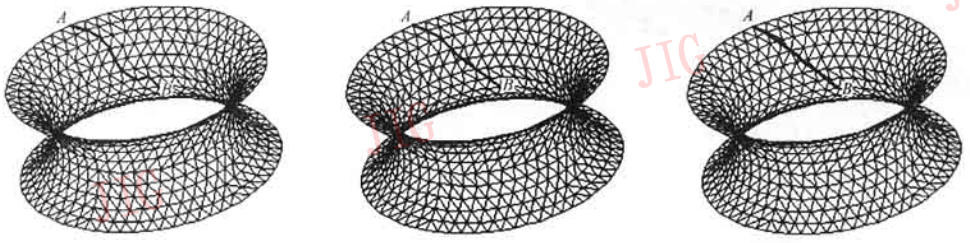


图 3　子图边集生成示意图

Fig. 3　The generation of edges of subgraph

图 4 给出了一个旋转曲面上 A(50mm,90mm,-86.603mm) B(66.535mm,35.673mm,-28.9mm)两点间的最短路径搜索过程示意图,其采用的三角型网格模型如图 4 所示,节点数为 628,单元数为 1 130。相应的计算参数设置为 $\gamma=3$;$\varepsilon=0.05$,迭代 4 次后结束。图 4(a)~图 4(c)中粗实线分别为由带权图 G、$G^{(1)}$ 和 $G^{(4)}$ 得到的最短路径。它们的路径长度分别为 89.927 898mm、84.789 062mm 和 84.040 175mm。当参数设置为 $\gamma=7$;$\varepsilon=0.000\,001$ 时,迭代 6 次收敛,路径长度为84.038 286mm。



(a) 初始近似最短路径　　(b) 第 1 次迭代结果　　(c) 第 4 次迭代结果

图 4　最短路径生成过程图

Fig. 4　The generation of shortest path

3　算例分析

为了验证本文提出算法的运行效率,利用两个应用领域的算例进行了验证。

3.1　算例 1

图 5 是一个球面体的三角形网格模型,其相应的节点数为3 674,三角形单元为7 344。本文以它为例来说明本算法在绘制局部区域边界中的应用。它解决的问题是根据用户给出的控制点来绘出相应的区域边界。本算例是计算球面体上由 4 个点控制的区域边界。它的实现过程为:①依次输入 4 个控制点 A、B、C 和 D;②由三角形网格模型生成带权图 G;③由 G 一次计算得到 AB、BC、CD、DA 之间的最短路径 $P_{AB}^{(0)}$、$P_{BC}^{(0)}$、$P_{CD}^{(0)}$ 和 $P_{DA}^{(0)}$;④生成相应子图 $G_{AB}^{(1)}$、$G_{BC}^{(1)}$、$G_{CD}^{(1)}$ 和 $G_{DA}^{(1)}$;⑤子图各自进入迭代细分过程,直到得到最终的近似最短路径 P_{AB}、P_{BC}、P_{CD} 和 P_{DA};⑥把 4 条路径首尾相连,即得到 4 个点决定的区域边界。

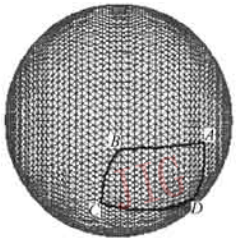


图 5　球面体四边形区域生成示意图

Fig. 5　The generation of quadrangle on sphere

3.2　算例 2

对于注塑成型的塑料制件,在制件设计时,一般需要计算制件的最大流长比,并根据所选择的塑料材料和注射机数据判断该制件是否能成型(即熔体

是否能充满型腔),然后才能综合各方面的因素来决定是更换材料或注射机型号,还是修改模具设计、移动浇口位置或更改制件设计,可见在模具制造前,估算最大流长比非常重要。对于简单制件,设计师通过实物测量即可很快估算流长比,但对于复杂制件,人们则很难由测量的方式来得到两点间的最短距离,更何况还要考虑厚度的因素。采用本文的算法,由于可以非常快捷算出制件上任意两点的流长比,从而从根本上解决了这个问题。

图 6 为摩托车的汽车后视镜的三角形网格模型,其相应的节点数为2 921,单元数为5 878。图中粗实线为 A、B 两点间的最短路径,相应的最短距离即 A、B 两点间的流动长度和厚度的比值为 115.15。

图 6 摩托车后视镜

Fig. 6 Rearview mirror of motorcycle

4 结 论

本文采用三角形网格模型表示复杂曲面,从而把求曲面上任意两点间的最短路径问题转变成求解三角形网格模型上两点间的最短路径问题。本文算法是采用 FSPA 法来动态计算带权图上两点间的最短路径,并利用迭代细分思想来构造子图。该法的优势在于:

(1)算法效率高。由于算法是由最短路径可能通过的区域来构造相应的带权图,因此可大大缩小子图规模。采用 FSPA 法计算带权图上两点的最短路径,其时间复杂度为 $O(N_{edge})$。在一般的应用中,由于总有 $N_{edge} \ll n^2$,所以该算法的时间复杂度远低于 Dijkstra 算法的时间复杂度 $O(n^2)$。

(2)近似精度可控。本文算法可以通过调整细分边参数 γ 值和改善网格质量等措施来提高曲面上两点间最短路径的逼近度。

(3)算法易理解和易于程序实现。

本文给出的算法对计算机辅助几何设计在测量中的应用具有重要的作用,现已应用于塑料制件的最大流长比计算。本算法还可应用于地理信息系统、机器人等其他相关领域。

参考文献(References)

1 Georg Menges, Walter Micaeli, Paul Mohren. How to make injection molds(3rd)[M]. Munich German: Carl Hanser Verlag, 2001. [[德]Georg Menges, Walter Micaeli, Paul Mohren. 著,闫光荣,许鹤峰等译. 注射模具制造工程[M]. 北京:化学工业出版社,2003:135.]

2 Sharir M, Schorr A. On shortest path in polyhedral spaces[J]. SIAM Journal on Computing, 1986, **15**(1):193~215.

3 Chen J, Han Y. Shortest path on the polyhedron [A]. In: Proceedings of the 6th ACM Symposium on Computer Geometry[C], Berkley, California, USA, 1990:360~369.

4 Kanai T, Suzuki H. Approximate shortest path on a polyhedral surface and its applications[J]. Computer-Aided Design, 2001, **33**(11): 801~811.

5 Lanthier M, Maheshwari A, Stack J-R. Approximating weighted shortest paths on polyhedral surfaces[A]. In: Proceedings of the 13th ACM Symposium on Computer Geometry[C], Nice, France, 1997:272~283.

6 Zhang Liyan, Wu Xi. Approximate shortest path on triangular mesh surface[J]. Journal of Computer Aided Design & Computer Graphics, 2003, **15**(5):592~597. [张丽艳,吴熹. 三角网格模型上任意两点间的近似最短路径算法研究[J]. 计算机辅助设计与图形学学报, 2003, **15**(5):592~597.]

7 Duan Fanding. A Faster Algorithm for Shortest-Path—SPFA[J]. Journal of Southwest Jiaotong University, 1994, **29**(2):207~212. [段凡丁. 关于最短路径的 SPFA 快速算法[J]. 西南交通大学学报,1994, **29**(2):207~212.]