基于改进A*算法的狭窄空间装配路径规划

何　磊，曹　虎，陈　雷
（航空工业成都飞机工业（集团）有限责任公司总装厂，成都 610091）

[摘要] 针对飞机总装过程中部分狭窄区域存在装配路径仿真困难，提出三维空间下改进 A* 算法，进行装配路径规划。将 A* 算法由二维平面推广到三维空间，同时考虑安装物体的尺寸及旋转对装配路径的影响。对安装空间和安装物体进行网格化处理，形成地图映射。根据安装物体在三维空间中的移动和旋转成本，建立评价函数，利用改进 A* 算法进行启发式搜索，获得装配路径节点。利用 CATIA 二次开发技术，在 CATIA 装配环境中进行算法实现和路径仿真。仿真结果表明该方法可以有效控制安装物体的移动及旋转，避开在狭窄空间中的障碍物，生成装配路径。

关键词： A* 算法；启发式函数；装配路径规划；装配路径仿真；CATIA
DOI:10.16080/j.issn1671-833x.2018.12.065

何　磊
助理工程师，研究方向为数字化装配、数字化工艺设计、智能车间构建。

现代飞机是一个复杂的系统工程，由飞机机体、机电及航电成品、管路系统、电气系统和液压系统等构成，各子系统按照功能需求以一定次序分布在飞机机体中。从制造和维护的角度，飞机又可划分为不同的舱位，每个舱位中交错分布有各子系统。

飞机制造过程中，需要在总装配阶段完成飞机各舱位内的线束敷设、导管安装、机电及航电成品安装等工作。各项敷设及安装工作受到飞机舱位空间限制，操作空间通常较狭小。

飞机总装配过程中，需要在各舱位中以一定次序将不同子系统的相关部件分别安装到位，同时需满足一定间隙要求，各部件间不能发生干涉。各子系统的部件交错分布，又存在安装依赖要求，协调关系十分复杂。

在以图纸为依据的飞机制造过程中，由于无法表达舱位内各部件间复杂的空间关系，需要制造物理样机，在物理样机上进行拆装试验后才能确定协调关系和安装路径。

随着计算机技术的发展，飞机制造逐渐进入到以数字模型为依据的时代，可以使用 CATIA、DELMIA 等工程软件构建虚拟样机，通过在虚拟样机中进行拆装仿真可以较为便捷地确定协调关系和安装路径，但仍需要耗费大量时间和精力进行仿真工作。同时装配路径仿真的结果可以作为输出物，指导现场工人进行机上装配工作，相较纸质工艺文件，操作指导性更强，更加直观，能够有效提高装配质量和装配效率。

随着飞机研制周期的不断缩短、状态更迭愈加频繁。装配路径仿真工作已成为制约飞机快速研制及装配质量提升的瓶颈。因此开展狭窄空间中的装配路径规划算法及实现研究是十分有益的。

飞机舱位中部件或成品的安装实际上是一个在狭窄空间中进行路径规划的过程，安装过程中需要考虑部件或成品的大小、姿态、安装位置、

安装路线和安装环境中的障碍物等因素。

针对总装配阶段的路径规划主要解决以下两个问题：

（1）实现三维装配空间中的路径规划。此处的三维装配空间是一个全向三维空间，物体可以沿 6 个自由度移动、旋转，且装配空间中的障碍物分布限制较少，可以从各个方向出现。不同于已有的汽车或者飞行器路径规划，障碍物通常位于地面，且有一定支撑。

（2）实现狭窄空间中的路径规划。此处的狭窄空间表明可通过区域的尺寸与进行路径规划物体的尺寸处于同一量级。通常进行路径规划的物体不能被简化为质点、球体或立方体。因为在狭窄空间的路径规划中，物体各方向上的尺寸差异对路径生成非常重要，常需要调整物体以某一特定角度才能通过障碍区域。

A* 算法及其应用

随着人工智能技术的发展，在路径规划领域出现了多种算法。其中 A* 算法是一种在有多个节点路径的二维平面上，求解出最低通过成本的算法。该算法结合了 Best-First 搜索算法和 Dijkstra 算法的优点，在进行启发式搜索提高算法效率的同时，可以找到一条基于评估函数的最优路径。通过对评估函数进行扩充和改进可以适应不同路径规划情形，因此 A* 算法具有较强适应性，基于以上优点，A* 及其改进算法在路径规划领域应用十分广泛。但原始 A* 算法仅限于在二维平面进行计算，将路径规划中的物体视为质点，未考虑物体的空间大小及姿态变换。

A* 算法作为一种较为高效、准确的寻路算法，在许多领域得到应用，尤其是在二维平面路径规划中，应用更加普遍。如在智能交通领域，利用 A* 算法在静态路网中求解最短路径问题 [1]。由于游戏中的地图通常为二维平面，因此多采用 A* 算法作为游戏中 AI 的寻路算法 [2]。同时，A* 算法还可运用在移动机器人的路径规划中 [3]。

但是如何将 A* 算法由二维平面推广到三维空间，许多学者做出了有益探索。

对于三维游戏场景中的障碍表达，陈疆等 [4] 提出一种新的表达方案，仍采用矩阵表达三维空间，对 3D 空间中大量的冗余数据进行了压缩，提高了数据存储和处理效率，但只能表示单层障碍物，对于多个障碍物叠加无法处理。

在无人机路径规划领域，利用稀疏 A* 搜索算法，并将地形的高清信息纳入考虑，能够有效进行地形回避，航线结果优于二维航迹 [5]，但是该算法只是简单对高度信息给出限定值，并没有完全实现三维空间中的路径规划算法。利用基于动态时间规整（DTW）距离的改进 A* 算法分层做二维航迹规划，在此基础上，提出粒子沉降法规划航迹点的高度信息生成三维航迹，显著减少了算法计算时间 [6]，该算法对于地形障碍、武装威胁等位于地面的需避让区域拥有较好的三维路径规划效果，而对于装配空间中可能从各个方向出现的障碍物则无法处理。

南加州大学的 Nash 等 [7] 与卡耐基梅隆大学的 Carsten 等 [8] 将 A* 算法从二维空间推广到三维空间，并分别提出了三维网格空间中任意角度下的 A* 搜索算法（图 1）但上述两种算法仅适用于物体本身相对于障碍物大小忽略不计，即在路径规划时将物体看作质点，并未解决狭窄空间中的路径规划问题。

在三维仿真领域，A* 算法也有一定应用，如利用 A* 算法实现三维模塑互连器件（MID）的交互式三维自动布线 [9]，但是该算法只能沿空间转折平面进行搜索，不能解决三维立体空间中的路径规划问题。

田立中等 [10] 对于数字装配技术中利用 A* 算法进行装配路径规划进行了探索，给出了变换坐标的原则，证明了动态坐标的 A* 搜索算法的收敛性，并进行了复杂性分析，但

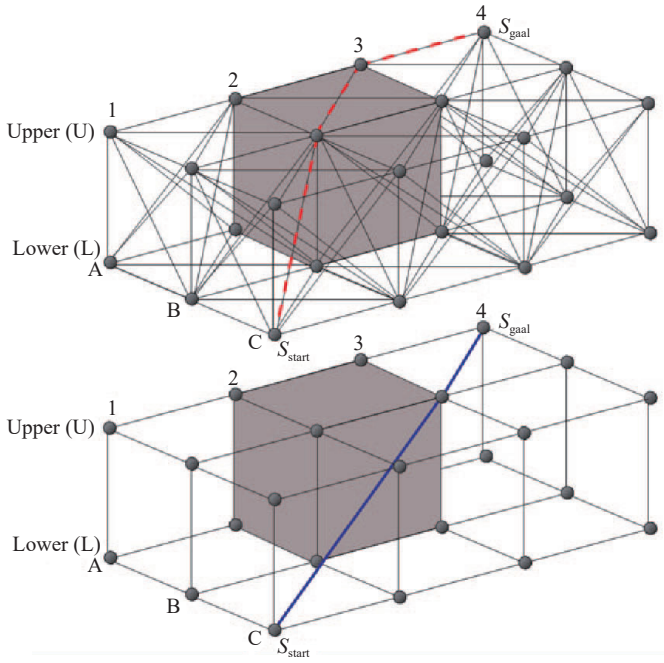


图1　网格化空间：图像边界构造的最短路径（上方）与向量路径构造的最短路径（下方）

Fig.1　Cubic grid: shortest path formed by graph edges (top) vs. shortest vertex path (bottom)

并未进行算法实现和仿真验证。

综上所述对 A* 算法进行改进后，可将其拓展到三维空间，但是为了适应装配空间的路径规划要求，必须将路径规划中物体的空间大小、姿态变换纳入考虑。将改进后的 A* 算法应用到飞机总装配阶段的安装仿真中，可实现部件或成品的安装路径自动生成，大大降低由技术人员手动进行仿真的人力成本，同时缩短飞机研制时间，对于推进飞机总装配阶段的自动化工艺设计有一定意义。

构建三维空间搜索地图

A* 算法是一种在网格化地图上进行搜索的算法，假定每次物体只能在网格化地图上沿某一方向移动单位距离。在二维平面问题中，搜索区域限制在二维网格地图上。在装配空间中的路径规划问题中，需要首先构建三维空间搜索地图。

因此在应用 A* 算法前，需要先将安装空间、障碍物、安装成品或部件进行网格化处理。

1 安装空间网格化

如图 2 所示，在 CATIA 等三维建模软件中，数字模型拥有一个初始坐标系，数字模型的各种特征建立以初始坐标系为基准展开。装配模型的初始坐标系又称为世界坐标系，如图 3 所示，将数字模型加载进装配模型中时，通过数字模型初始坐标系与世界坐标系的相对位置关系可以确定数字模型在装配环境中的位置。

对于三维安装空间，可以通过三维数组 Map(i, j, k) 对其进行离散化表示，三维数组中每一成员对应于安装空间中一个单元体。当三维数组中的某一成员被置为 0 时，表示对应安装空间中单元体未被障碍物占据，当三维数组中的某一成员被置为 1 时，表示该位所对应单元体已被障碍物占据。将所有被障碍物占据单元体所对应三维数组成员置为 1，即可得到进行路径规划所需地图。

2 障碍物网格化

为得到路径规划所需地图，需要将被障碍物占据单元体所对应三维数组成员置为 1，即需要对障碍物进行网格化。

三维建模软件中，数字模型由多个特征构成，模型具有完整的几何信息和拓扑信息，但无法网格化表示对数字模型所占据空间。

如图 4 所示，可构建一个单元体阵列，将该单元体阵列与数字模型进行相交检测，与数字模型相交的单元体即表示被占据，可在三维数组中将对应成员置为 1。深度方向上一个层级检测完毕后，将单元体阵列偏移一个单元距离循环检测，通过这种方式可将数字模型转换为网格化表示。

安装部件或成品数字模型网格化的过程与障碍物网格化类似。由于安装部件或成品在进行路径规划时会进行位置和姿态变换，因此在网格化时需获取其初始位置和姿态信息。CATIA 中，数字模型的位置和姿态通过一个含有 12 个元素的数组表示。如图 5 所示，数组中的前 9 个元素为可划分为 3 组，每组 3 个元素，分别表示单元坐标轴在世界坐标系 X_O、Y_O 和 Z_O 轴上的投影量，数组中最后 3 个元素表示数字模型在世界坐标系中的 X_O、Y_O 和 Z_O 上的位置值。

$$\begin{pmatrix} X_i & X_j & X_k \\ Y_i & Y_j & Y_k \\ Z_i & Z_j & Z_k \\ X & Y & Z \end{pmatrix}$$

式中，X_i 表示模型单位坐标轴 X 在世界坐标系 X_O 上的投影量，其余各量依次类推。

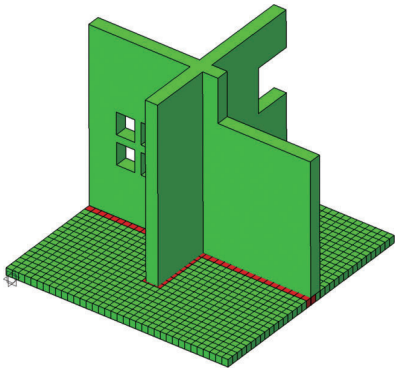


图4 障碍物网格化
Fig.4 Meshed barrier

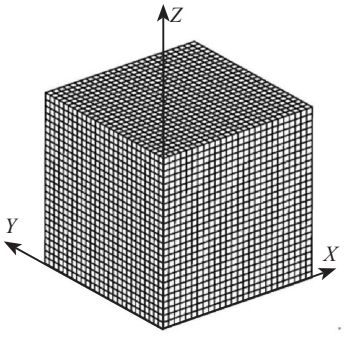


图2 网格化空间
Fig.2 Meshed space

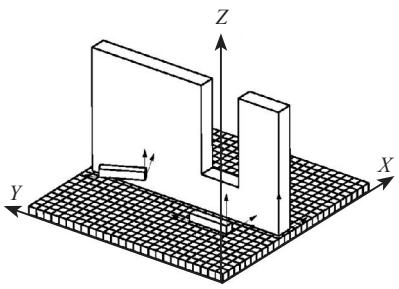


图3 世界坐标系与模型坐标系
Fig.3 Word coordinate and model coordinate

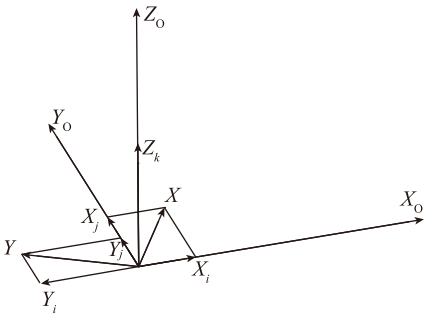


图5 姿态的投影表示
Fig.5 Project of model attitude

对 A* 算法的改进

1 将 A* 算法推广到三维

A* 算法是一种启发式搜索算法，可用于在有多个节点的地图中，寻找出最低通过成本的路径。

在该算法中，以 $g(n)$ 表示从起点到任意顶点 n 的实际距离，$h(n)$ 表示任意顶点 n 到目标顶点的估算距离，估算距离的值根据所采用的评估函数的不同而变化，则 A* 算法的估算函数为：

$$f(n)=g(n)+h(n) \quad (1)$$

常见的 $h(n)$ 评估函数包括欧几里得距离、曼哈顿距离和切比雪夫距离等。

以下以三维空间中的切比雪夫距离为例。提出推广到三维后 A* 算法的评估函数。

若三维空间中有两个点 P、Q，其坐标分量分别为 P_i 及 Q_i，则两者之间的切比雪夫距离定义如下：

$$D_{\text{Chebyshev}}(P,Q):=\max_i(|P_i-Q_i|) \quad (2)$$

假设起点 P 及终点 Q 坐标分别为：$P(0,3,0)$，$Q(5,0,6)$，依据上式则这两点间的切比雪夫距离为 6。

图 6 所示，这里的切比雪夫距离表示从 P 点出发，每次移动一个单元距离（既可平行移动，也可斜向移动，但只能向相邻单元移动），移动到 Q 点所需的最小步数。对于平面问题，每次可移动的相邻单元格有 8 个，对于三维问题，每次可移动的相邻单元格有 26 个。

切比雪夫距离仅得出从起点移动到终点所需最小步数，评估函数中使用到的是路径的距离，因此还需根据切比雪夫距离计算得出实际的路径距离。

$$D_{\text{Real}}(P,Q):=D_{\text{Real}}(P,M)+D_{\text{Real}}(M,N)+D_{\text{Real}}(N,Q) \quad (3)$$

根据上式可以得出：

$$D_{\text{Transform}}(P,Q):=\sqrt{3}\min_i(|P_i-Q_i|)+\sqrt{2}(\text{mid}_i(|P_i-Q_i|)-\min_i(|P_i-Q_i|))+(\max_i(|P_i-Q_i|)-\text{mid}_i(|P_i-Q_i|)) \quad (4)$$

2 狭窄空间装配中的 A* 算法评估函数改进

对于狭窄空间中的装配，还需考虑装配体姿态的变化，因此评估函数除要考虑路径移动成本外，还需考虑姿态变换成本，在路径规划过程中尽量减少姿态变换。

根据 A* 算法及其应用，可用 3×3 矩阵表示数字模型的姿态信息，则 P 点与 Q 点姿态变换成本可表示为两个矩阵相减，对结果矩阵各元素的绝对值求和。

$$\boldsymbol{R}=\begin{pmatrix}P_{x_i} & P_{x_j} & P_{x_k}\\ P_{y_i} & P_{y_j} & P_{y_k}\\ P_{z_i} & P_{z_j} & P_{z_k}\end{pmatrix}-\begin{pmatrix}Q_{x_i} & Q_{x_j} & Q_{x_k}\\ Q_{y_i} & Q_{y_j} & Q_{y_k}\\ Q_{z_i} & Q_{z_j} & Q_{z_k}\end{pmatrix} \quad (5)$$

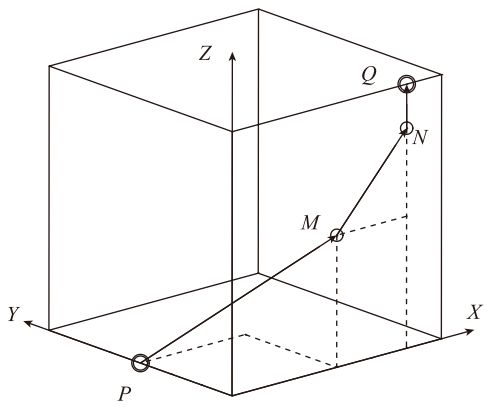


图6 三维空间下的切比雪夫距离

Fig.6 Chebyshev distance in 3D space

P_{x_i} 表示 P 点模型单位坐标轴 X 在世界坐标系 X_O 上的投影量，其余各量依次类推。

$$D_{\text{Rotate}}(P,Q):=\sum_{\substack{0\leqslant m\leqslant 3\\ 0\leqslant n\leqslant 3}}|R(m,n)| \quad (6)$$

则 P、Q 两点间的路径通过成本为：

$$h(P,Q):=D_{\text{Transform}}(P,Q)+D_{\text{Rotate}}(P,Q) \quad (7)$$

以下为 A* 搜索实现伪代码：

```
Main ()
  open:=closed:=Ø;
  g(s_start ):=0;
  parent(s_start):=s_start;
  open.Insert(s_start, g(s_start)+h(s_start))
  While  open ≠Ø Do
    s:=open.Pop();
    [SetVertex(s)];
    If s = s_goal Then
        Return " 已找到路径 ";
        Construct_Path(s);
    closed:=closed ∪ {s};
    Foreach s' ∈ nghbr_vis (s) Do
        If s'∉ closed Then
          If s'∉ open Then
            g(s'):=∞;
            parent (s')=NULL;
          UpdateVertex (s, s');
    Return " 路径未找到 ";
End

UpdateVertex (s, s')
    g_old:=g(s');
    ComputeCost (s, s' );
    If g(s') < g_old Then
        If s'∉ open Then
          open.Remove (s');
    open.Insert (s', g(s')+h(s'))
End
ComputeCost (s, s')
    If g(s')+h(s, s')<g(s') Then
        parent (s'):=s;
        g(s')=g(s)+h(s, s');
End
```

其中，open.Insert (s, x) 函数向

open 列表中插入一个向量点 s，向量点 s 的 key 值为 x，open.Remove (s) 函数从 open 列表中移除 s，open.Pop() 函数从 open 列表中返回一个向量点，该向量点具有最小的 key 值。

$nghbr_{vis}$ (s) 函数返回一个列表，该列表中为 s 向量点的周围节点，$nghbr_{vis}$ (s) 函数的定义如下：

```
nghbrvis (s)
 For i = -1 To 1
  For j = -1 To 1
   For k = -1 To 1
    Foreach m ∈ s Do
     If Map(m. x+i, m. y+j, m. z+k)=0
      Flag = True
     Else
      Flag = Flase
      Exit For
     If Flag = True
       nghbrvis.Insert (s,v)
     Else
      Foreach r ∈ RotateMatrix Do
       Foreach m ∈ s Do
         m'=Rotate(m,r)
           If Map(m'.x+i, m'. y+j,
m'.z+k)=0
         Flag = True
       Else
         Flag = Flase
         Exit For
       If Flag = True
         nghbrvis.Insert (s,r)
   Reture nghbrvis
End
```

其中，Map(i, j, k) 函数表示在第一章构建的数字地图中位置为 i, j, k 的点是否被障碍物占据。被占据返回 1，否则返回 0。

RotateMatrix 表示一个变换矩阵列表，列表中以位姿矩阵的方式存储了装配体可能的位姿方向。

$nghbr_{vis}$.Insert (s, v) 函数表示向 $nghbr_{vis}$ 列表中添加一个向量点，其位置和姿态变化为 v。Rotate($\boldsymbol{m}$, $\boldsymbol{r}$) 表示将向量 $\boldsymbol{m}$ 按照姿态变换矩阵 $\boldsymbol{r}$ 进行变换。

$$\mathrm{Rotate}(\boldsymbol{m},\boldsymbol{r})=\begin{pmatrix} r_{x_i} & r_{x_j} & r_{x_k} \\ r_{y_i} & r_{y_j} & r_{y_k} \\ r_{z_i} & r_{z_j} & r_{z_k} \end{pmatrix}^{\mathrm{T}} \times \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} \quad (8)$$

A* 搜索完毕后调用 Construct_Path(s) 函数绘制路径，函数从 s 开始绘制，先绘制 s 与 s 的父节点间的路径，随后迭代绘制 s 父节点与 s 父节点的父节点间的路径，直至起点，则整个路径绘制完毕。

基于 CATIA 二次开发的算法实现

CATIA 作为一款辅助三维设计软件，除提供三维建模、装配仿真功能外，还提供了强大的二次开发接口和工具，如 VBA(Visual Basic in Application) 及 CAA(Component Application Architecture) 等，用户可以利用这些接口和工具编写自动化脚本和插件，完成 CATIA 中原本无法实现的功能，本文中的算法实现利用 CATIA 附带的 VBA 工具完成。关于 CATIA 附带 VBA 工具的使用可以参考 CATIA 自带的 V5Automation 文件。

如前文所述，网格化时需要利用单元体与数字模型进行相交检测，可通过对 CATIA 空间分析模块中的碰撞检测工具进行定制后完成。

如图 7 所示，碰撞检测工具会检测两个装配体之间干涉和接触的部分，将选择 1 设为单元体矩阵，将选择 2 设为障碍物，并利用碰撞检测工具进行检测，可以得到单元体矩阵中被障碍物占据的单元体位置，即对障碍物进行了网格化处理，如图 8 所示。上述功能可以通过对 CATIA 提供的 VBA 接口中 Clashes 对象进行相应操作后完成。

图7 碰撞检测工具
Fig.7 Collision detect tool

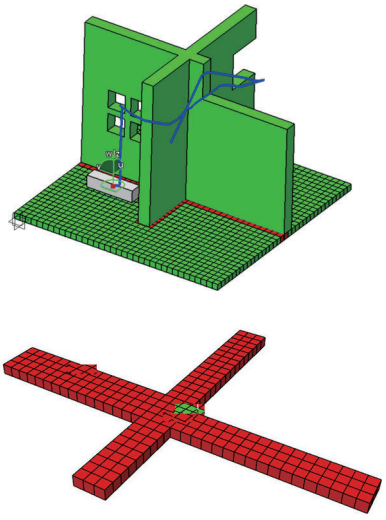


图8 碰撞检测
Fig.8 Collision detect

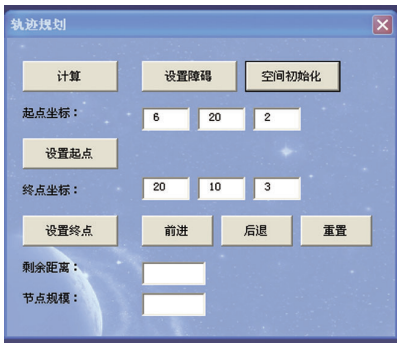


图9 软件界面
Fig.9 Software UI

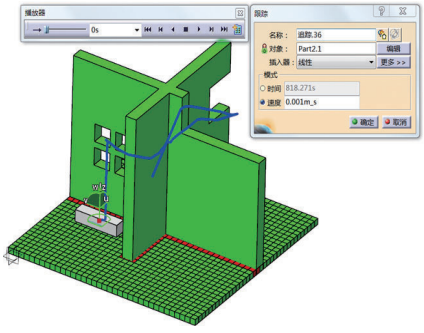
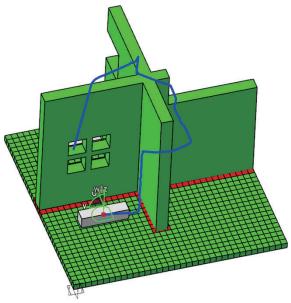
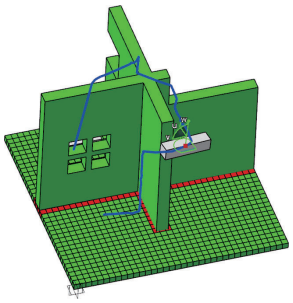


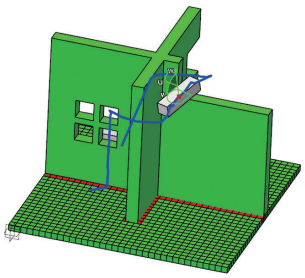
图10 装配路径仿真
Fig.10 Assembly path simulation



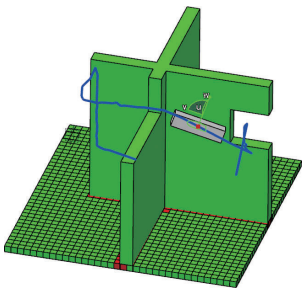
（a）第 1 步



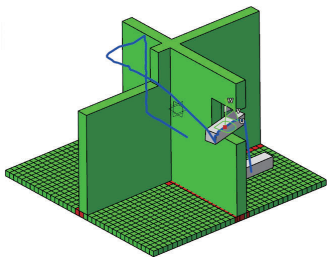
（b）第 2 步



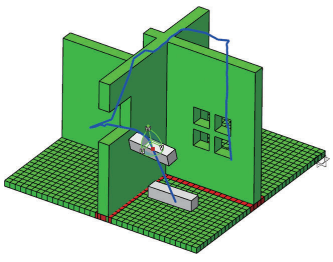
（c）第 3 步



（d）第 4 步



（e）第 5 步



（f）第 6 步

图11　装配路径仿真结果

Fig.11　Result of assembly path simulation

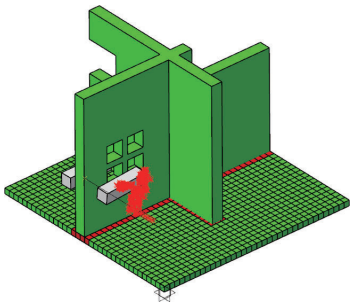
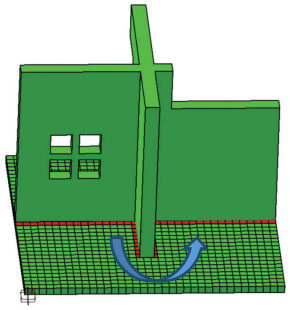
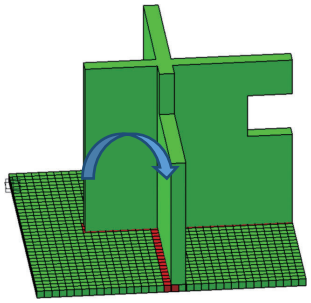


图12　CATIA自身的路径寻找算法仿真结果

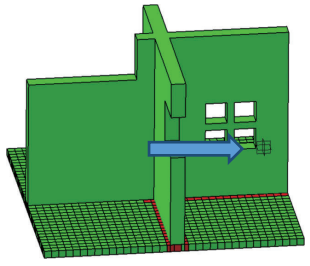
Fig.12　Result of CATIA path routing algorithm



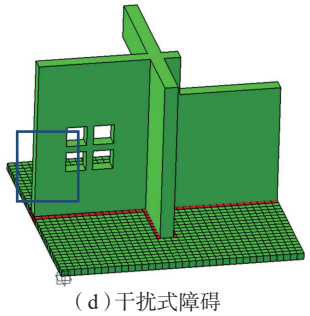
（a）绕过式障碍



（b）跨越式障碍



（c）穿越式障碍



（d）干扰式障碍

图13　4种障碍形式

Fig.13　Four type barrier

利用 CATIA 提供的 VB 编辑器可以设计交互界面，实现基于触发的程序控制，可提供与软件原生界面类似的交互体验，本文编写的算法实现交互界面如图 9 所示。

在 CATIA 中进行路径规划时按照设置起点、设置终点、设置障碍物和计算的步骤进行。程序计算完毕后，如图 10 所示，会以 CATIA 提供的路径仿真格式存储在装配模型中，便于仿真和查看。

如图 11 所示可通过 CATIA 提供的路径仿真工具对路径规划结果进行播放，播放器的功能包括暂停、回退、单帧播放、播放速度控制等，可以对仿真结果做详细的查看和分析。

算法比较

对推广到三维空间后的 A* 算法、改进后的三维 A* 算法及 CATIA 软件自身的路径寻找算法进行横向比较。主要从计算效率、对障碍环境的适应程度及路径规划质量等方面进行比较。3 种算法的特点如下：

（1）推广到三维空间后的 A* 算法，可以快速在障碍环境中进行路径规划，但进行路径规划的物体仅能进行平移，无法进行旋转，某些障碍区域无法通过。

（2）改进后的三维 A* 算法，可

以快速在障碍环境中进行路径规划，进行路径规划的物体可以进行平移和旋转，能通过各种类型的障碍，适应性强。

（3）CATIA 自身的路径寻找算法，给定起点和终点后，自动根据障碍环境进行路径规划，计算速度较快，容易陷入局部死循环无法退出（图 12）。

硬件运行环境如下：Windows 7 操作系统，CPU 为 Intel E5-2620，显卡为 NVIDIA Quadro K2200，运行内存为 32G。软件版本为 CATIA V5R21。

如图 13 所示，通过对装配空间中的障碍信息进行归纳总结，可分为 4 类：

（1）绕过式障碍：通过此类障碍时，物体在一个方向上受到尺寸限制，在另一个方向上不受限制。

（2）跨越式障碍：通过此类障碍时，物体在一个方向上受到尺寸限制，在另一个方向上受到一定限制。

（3）穿越式障碍：通过此类障碍时，物体在两个方向上都受到尺寸限制。

（4）干扰式障碍：此类障碍的通过尺寸小于或接近物体最小尺寸，如果不能准确识别，会对路径规划造成干扰。

路径规划物体设定物体宽度为网格单元距离 2 倍，物体长宽比为 4。

如图 14 所示，起点位于 Start 区，

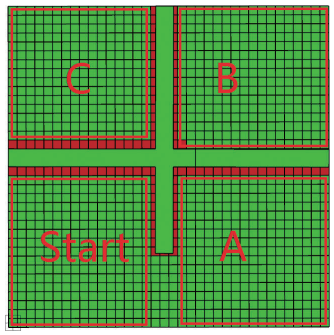


图14　路径规划区域划分

Fig.14　Area of path routing

表1　改进后的三维A*算法测试结果　　min

终点区域	5 倍通过距离	4 倍通过距离	3 倍通过距离	2 倍通过距离
C 区	36	42	50	56
B 区	3	13	21	25
A 区	0.2	0.3	0.8	1

注：通过距离倍数是指障碍空间中最小通过距离与进行路径规划物体宽度的比值。

表2　推广到三维空间后的A*算法测试结果　　min

终点区域	5 倍通过距离耗时	4 倍通过距离耗时	3 倍通过距离耗时	2 倍通过距离耗时
C 区	10	13	N/A	N/A
B 区	2	4	21	N/A
A 区	0.2	0.3	0.6	0.7

注：N/A 表示路径规划算法无法得出结果。

表3　CATIA自身的路径寻找算法测试结果　　s

终点区域	5 倍通过距离耗时	4 倍通过距离耗时	3 倍通过距离耗时	2 倍通过距离耗时
C 区	N/A	N/A	N/A	N/A
B 区	20	35	55	72
A 区	6	7	7	8

终点依次位于 *A*、*B*、*C* 3 个区域，3 种算法的路径规划测试结果如表 1~3 所示。

通过上述算法比较可知：

（1）当障碍环境中仅包含绕过式和跨越式障碍时，CATIA 自身的路径寻找算法效率最高，能快速进行路径规划。

（2）当障碍环境中包含以上 4 种类型的障碍时，CATIA 自身的路径寻找算法由于陷入局部循环，通常无法完成路径规划任务。

（3）当障碍区域的可通过尺寸大于物体长度时，推广到三维空间后的 A* 算法能够以较高效率完成路径规划任务。

（4）当障碍区域的可通过尺寸介于物体长度和宽度之间时，仅有改进后的三维 A* 算法能够完成路径规划任务，因为物体需要旋转角度后才能通过障碍区域。

结论

将 A* 搜索算法进行推广，将其拓展到三维区域，并考虑物体的空间大小及姿态对轨迹规划的影响。利用 CATIA 提供的二次开发接口 (VBA) 实现改进 A* 搜索算法，并在 CATIA 装配体环境中进行验证。

验证结果表明，改进后的 A* 算法能够较好地在三维狭窄空间中进行轨迹规划，通过控制物体的姿态变换，能够穿越原本仅依靠平移操作无法穿越的区域。

上述操作与工艺人员在飞机装配工艺设计过程中所进行的仿真工作是类似的，将该方法应用到飞机装

配工艺设计中，能够降低工艺人员手工仿真的工作量。

本文仅讨论单一物体路径规划问题，实际上飞机装配过程中同一舱位有多个物体需进行路径规划，可将算法进一步扩展到多个物体的路径规划上。同时结合多个物体间的安装依赖规则定义、安装逻辑定义，能够实现装配工艺过程的自动化设计，仿真结果自动生成，仿真结果可作为现场生产指导依据，对于缩短飞机研制流程，提高飞机现场生产效率有一定作用。

参 考 文 献

[1] 朱云虹，袁一．基于改进 A* 算法的最优路径搜索 [J]. 计算机技术与发展，2018 (4): 1–6.

ZHU Yunhong, YUAN Yi. An optimal path search based on the improved A* algorithm [J]. Computer Technology and Development, 2018(4): 1–6

[2] 晋国卿．游戏开发中智能寻径方法的应用研究 [D]. 南昌：南昌大学，2013.

JIN Guoqing. Application research of artificial intelligence path finding in games development[D]. Nanchang : Nanchang University, 2013.

[3] 孙炜，吕云峰，唐宏伟，等．基于一种改进 A* 算法的移动机器人路径规划 [J]. 湖南大学学报 (自然科学版), 2017, 44(4): 94–101.

SUN Wei, L Ü Yunfeng, TANG Hongwei, et al. Mobile robot path planning based on an improved A* algorithm[J]. Journal of Hunan University (Natural Sciences), 2017, 44(4): 94–101.

[4] 陈疆，郭克华，梁琳．一类新的网络游戏场景 3D 障碍信息表达方案 [J]. 中南大学学报 (自然科学版), 2012, 43 (7): 183–188.

CHEN Jiang , GUO Kehua , LIANG Lin. A novel approach for 3D barriers information expression in network game scene[J]. Journal of Central South University (Science and Technology) , 2012, 43(7): 183–188.

[5] 李春华，郑昌文，周成平，等．一种三维航迹快速搜索方法 [J]. 宇航学报，2002(3): 13–17.

LI Chunhua, ZHENG Changwen, ZHOU Chengping, et al. Fast search algorithm for 3D-route planning[J]. Journal of Astronautics, 2002(3): 13–17.

[6] 杨润洲，丁勇，张承果．基于 DTW 的改进 A* 算法在航迹规划中的应用 [J]. 电光与控制，2016, 23(6): 5–10, 26.

YANG Runzhou, DING Yong, ZHANG Chengguo. Application of the improved A* algorithm based on dtw in route planning[J]. Electronics Optics & Control, 2016, 23(6): 5–10, 26.

[7] NASH A, KOENIG S, TOVEY C A. Lazy Theta*: any-angle path planning and path length analysis in 3D[J]. Symposium on Combinatorial Search, 2010, 18(3): 299–307.

[8] CARSTEN J, FERGUSON D, STENTZ A. 3D Field D: Improved path planning and replanning in three dimensions[C]// EEE/RSJ International Conference on Intelligen Robots & Systems. Beijing, 2006.

[9] 展慧娴，卓勇，吴轩，等．基于 A~* 算法 MID 三维布线的研究与实现 [J]. 厦门大学学报 (自然科学版), 2015, 54(6): 888–892.

ZHAN Huixian, ZHUO Yong, WU Xuan, et a1. The realization of 3D routing based on A* algorithm in MID design[J]. Journal of Xiamen University (Natural Science), 2015, 54(6): 888–892.

[10] 田立中，付宜利，马玉林，等．装配路径规划中基于动态坐标的 A~* 搜索算法 [J]. 计算机集成制造系统，2002, 8(4): 316–319.

TIAN Lizhong, FU Yili, MA Yulin, et al. A* search arithmetic based on dynamic coordinate in assembly path plan[J]. Computer Integrated Manufacturing Systems, 2002, 8(4): 316–319.

通讯作者：何磊，E–mail：fenglinwansu@126.com。

Narrow Space Assembly Path Planning Based on Improved A* Algorithm

HE Lei, CAO Hu, CHEN Lei

(Final Assembly Workshop, AVIC Chengdu Aircraft Industry (Group) Co., Ltd, Chengdu 610091, China)

[ABSTRACT] Combining with the assembly path simulating difficulty of some narrow space area in final assembly situation, the improved A* algorithm in 3D space is adopted in assembly path planning. Improve the A* algorithm from 2D plane to 3D space, and the influence of size and rotate of assembly part on path planning is considered. Mesh the assembly space and part to make a map. According to the move and rotate cost of assembly part in 3D space to build the evaluate function, by using the improved A* algorithm to heuristic searching to find the assembly path node. By using the automation development of CATIA, realizing the algorithm and path simulation in CATIA assembly environment. According to the simulating result, the method can effectively control the move and rotation of assembly part, avoid the barrier in narrow space and generate the assembly path.

Keywords: A* algorithm; Heuristic function; Assembly path planning; Assembly path simulation; CATIA

（责编　大漠）