计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200300131

基于异步机制的 Gazebo 仿真优化研究

曾 蕾[1] 李 豪[2] 林宇斐[2] 张 帅[2]
1 天津(滨海)军民融合人工智能创新中心 天津 300457
2 军事科学院 北京 100000
(tjpu_zenglei@sina.com)

摘 要 在大规模机器人仿真过程中,为了保证仿真精度,通常采用基于时间步的推进机制。这种机制下,虽然可以通过调整仿真时间步灵活控制仿真精度,但当仿真规模较大时,在仿真循环的每次迭代中需要采用同步阻塞的方式,执行大量用于更新位姿或状态的插件代码,从而导致仿真性能降低。针对这一大规模机器人仿真所面临的精度与性能之间的矛盾,提出了一种基于异步策略的优化方案,并在流行的机器人仿真器 Gazebo 中对优化方案进行了设计实现,最后基于 rosflight 固定翼无人机案例,验证了方案的有效性。实验结果表明,对于 100 架固定翼无人机的仿真,采用异步策略优化重构后,仿真加速比达到了 5.0 以上。

关键词:Gazebo;ROS;异步策略;优化重构;高精度;实时仿真;大规模
中图法分类号 TP391.9

Study on Simulation Optimization of Gazebo Based on Asynchronous Mechanism

ZENG Lei[1],LI Hao[2],LIN Yu-fei[2] and ZHANG Shuai[2]
1 Tianjin Artificial Intelligence Innovation Center,Tianjin 300457,China
2 The Academy of Military Science,Beijing 100000,China

Abstract In the process of large-scale robot simulation,in order to ensure the accuracy of simulation,the propulsion mechanism based on time step is usually adopted. In this mechanism,the simulation accuracy can be flexibly controlled by adjusting the simulation time step. However,when the simulation scale is large,a large number of plug-in codes for updating posture or state need to be executed in the way of synchronous blocking in each iteration of the simulation cycle,which reduces the performance of the simulation. In order to solve the contradiction between the accuracy and performance of this large-scale robot simulation,an optimization scheme based on asynchronous strategy is proposed,and the optimization scheme is designed and implemented in the popular robot simulator Gazebo. Finally,the validity of the scheme is verified based on the case of the fixed wing of rosflight UAV. The experimental results show that the acceleration ratio of the simulation is over 5.0 after the asynchronous strategy is used to optimize the simulation of 100 fixed wing UAVs.

Keywords Gazebo,ROS,Asynchronous strategy,Optimal reconstruction,High-precision,Real time simulation,Large-scale

1 引言

近年来,机器人技术发展迅猛,在军事、国防、民用上都得到了广泛的应用。然而,机器人造价高昂,为规避和减少不必要的损失,仿真在机器人相关研究中扮演了重要角色。通过对机器人的结构和功能进行建模仿真,可以快速有效地测试新概念、新策略和新算法,从而为控制器原型的快速实现、机器人评估设计、虚拟传感器的模拟提供支撑[1-4]。随着群体概念的兴起,机器人集群仿真受到了越来越多的关注和研究[5-8]。

在机器人集群仿真过程中,通常采用基于时间步的仿真推进机制,通过对时间步的调整,灵活控制仿真精度。一般地,仿真时间步越小,仿真精度越高,同时,仿真计算量也越多,进而导致仿真性能越差。然而,随着集群仿真规模的不断增加,仿真精度与性能之间的矛盾逐渐突出。如何在保证仿真精度的前提下,提高机器人集群仿真性能是研究的热点问题[10-11]。

Gazebo 是一款在工业和学术界都受到广泛关注的开源机器人平台[1-4],可以支持大规模集群的仿真。为了保证仿真精度,Gazebo 提供了 ODE,Bullet,Dart,Simbody 等多种底层物理引擎支持[8]。此外,Gazebo 提供了灵活的插件机制,为机器人集群仿真的建模、控制以及可视化等提供了丰富的接口。值得一提的是,通过与机器人操作系统(ROS)的紧密连接,形成的 Gazebo-ROS 仿真开发框架[1]为机器人集群等复杂系统的开发、测试以及应用带来了极大的便利。

本文基于 Gazebo-ROS 仿真框架,针对大规模机器人集群仿真面临的性能问题,提出了一种基于异步策略的仿真优化方案。在启动初始化阶段,加载注册外部插件并设置插件功能逻辑代码调用频率及模型状态话题发布频率;随后在仿真主线程外,额外启动并维护两个子线程,分别用来周期性调用执行外部扩展插件注册的功能逻辑代码及发布模型状态话题信息,进而解耦了话题发布与 Gazebo 物理求解过程的关联,从而使话题发布不再直接影响 Gazebo 物理求解的执行。

通信作者:李豪(lihao@nudt.edu.cn)

最后，基于 rosflight 固定翼无人机案例，设计实现了大规模无人机仿真测试实验，验证了该优化方案的正确性与有效性。

2 平台简介

2.1 Gazebo

利用 Gazebo 创建三维动态多机器人环境，可以重现下一代移动机器人将遇到的复杂世界，进而通过仿真来精确再现机器人可能遇到的动态环境。在仿真过程中，所有模拟对象都具有质量、速度、摩擦力和其他属性。通过这些属性，Gazebo 可以模拟真实世界中推、拉、翻、搬运物体等动作。将这些简单的动作进行组合进而可以形成更为复杂的行为，例如觅食等。在仿真世界中，机器人本身是由刚体通过关节连接而成的动态结构，力和力矩都可以施加到表面和关节上，以产生运动和与环境的相互作用。世界本身是由景观、突出的建筑和其他用户创建的对象来描述的。

为了实现模拟对象可视化与物理求解过程的解耦，提高仿真性能，Gazebo 采用了 C/S 架构，如图 1 所示。Gazebo 启动后，存在 gzserver 与 gzclient 两个进程。其中 gzserver 负责仿真物理模拟和解算，gzclient 负责仿真渲染显示。gzserver 与 gzclient 通过话题订阅与发布机制实现信息的交互。

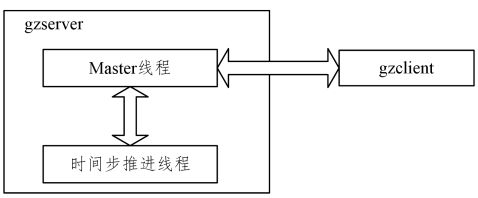

图 1 Gazebo C/S 架构

Fig. 1 C/S architecture of Gazebo

为了实现模拟对象物理学解算，Gazebo 封装了 ODE，Bullet，Dart 等物理引擎，采用基于时间步的迭代循环进行仿真的演算推进。每个时间步迭代中，Gazebo 主要完成预处理、碰撞检测、物理更新、消息处理等 4 部分功能。

首先，在预处理阶段，Gazebo 利用扩展的功能插件实现与外部控制器的通信，既接收控制器指令，施加到模拟对象上用于该时间步内的迭代模拟，又可以根据当前模拟对象状态信息解算生成相关的功能性消息（如 GPS，IMU 等）并发送给控制器供其使用。Gazebo 基于微内核架构实现了插件机制，如图 2 所示。

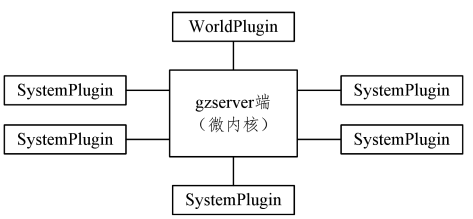

图 2 基于微内核架构的插件机制

Fig. 2 Plugin mechanism based on microkernel architecture

Gazebo 提供了 ModelPlugin，SensorPlugin，WorldPlugin，SystemPlugin，GUIPlugin，VisualPlugin 6 类扩展插件。利用这些插件 Gazebo 可以扩展定制各种功能，如利用 SystemPlugin 插件可以实现 Gazebo 与 Ros 的连通，利用 SensorPlugin 插件可以实现功能多样的传感器插件（摄像机、激光雷达、声纳等），利用 ModelPlugin 可以实现模拟对象与外部控制器之间的通信等。

为了实现上述插件功能，Gazebo 实现并维护了一套同步事件接口。在 Gazebo 初始化阶段，加载各类扩展插件并向其注册事件回调函数，并在每个时间步迭代过程中以同步阻塞的方式回调执行。其中，预处理阶段主要是 ModelPlugin 插件注册的回调函数的周期性调用执行。

其次，在碰撞检测与物理更新阶段，Gazebo 利用底层封装的物理引擎实现模拟对象的物理力学解算过程。根据不同的仿真场景需求，可以配置使用不同的物理引擎。比如，ODE 主要用于刚体间的物理学求解，而 Bullet 不仅可以用于刚体间物理学求解，也可以用于流体等可变形体之间的物理模拟求解。

最后，在消息处理阶段，Gazebo 会根据模拟对象的状态信息生成位姿等话题信息并按照一定的频率发布出去，供 gzclient、传感器等订阅渲染使用。

由于封装了 ODE 等物理引擎，Gazebo 仿真逻辑与 ODE 基本应用执行流程相似，如图 3 所示。在 Gazebo 仿真流程中，仿真更新预处理模块通过同步事件机制实现了外部注册回调函数的回调执行，通过该模块施加力和力矩等控制命令来实现对仿真世界中仿真刚体的人为控制，也可以基于当前仿真世界中仿真刚体的状态生成外部程序所需的信息，如里程信息、GPS 信息等。消息处理模块则在每步仿真迭代结束后，向 Gazebo Master 发步仿真模型最新状态信息以供 gzclient 等客户端程序订阅渲染。按照采样定理，通常外部控制指令与仿真状态信息按一定周期更新即可满足特定条件下的高保真仿真。

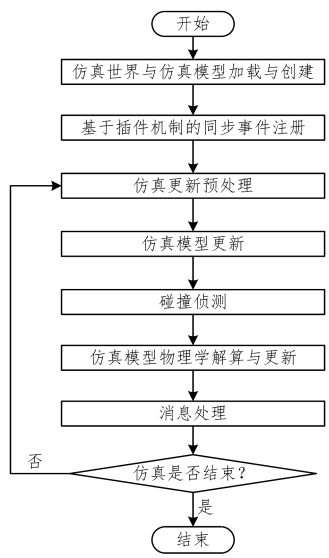

图 3 Gazebo 仿真流程

Fig. 3 Implementation flow chart of Gazebo

2.2 ROS

ROS 通常用来进行机器人控制及通信，用来驱动真实机器人执行各种任务。然而，在实际应用场景中，通常利用 ROS API 开发的机器人控制程序在仿真环境中进行有效性验证，进而防止由于程序的不成熟而带来不必要的损失。Gazebo 因高精度、实时性、支持 3D 场景的优点而被机器人领域的众多研究人员所喜爱。Gazebo-ROS 架构下的仿真场景也被大量采用和实践，进而用于验证和指导机器人程序开发。

Gazebo-ROS 仿真场景下，通常利用 ROS API 来进行机器人控制程序开发（如 rosflight 飞控程序[18]等），Gazebo 用来构建机器人仿真模型及仿真环境，进而对现实生活中的物理

现象进行模拟和仿真，其整体框架如图 4 所示。

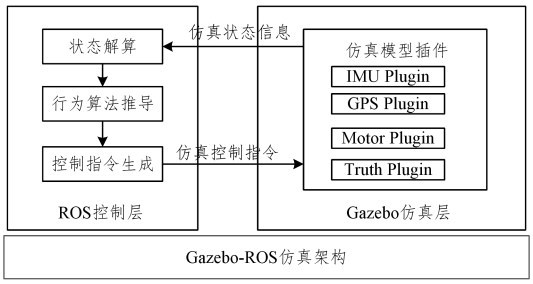

图 4　Gazebo-ROS 架构图

Fig. 4　Gazebo-ROS architecture

从图 4 中可以看出，Gazebo-ROS 架构下仿真的大致流程为：1）Gazebo 通过仿真模型插件实时更新模型状态信息；2）ROS控制层定时获取仿真模型状态信息并进行状态信息解算；3）行为算法推导节点获取解算后的状态信息并进行行为推导计算；4）控制指令生成节点获取推导产生的行为信息来生成行为控制指令；5）Gazebo 仿真插件获取行为控制指令用于控制仿真模型的运动。

3　基于异步策略的 Gazebo 迭代循环优化重构

3.1　Gazebo-ROS 架构下仿真插件的分析

如前所述，Gazebo 可以与通用机器人操作系统 ROS 方便地进行通信与交互，从而为高精度实时 3D 仿真提供了一种解决方案：利用 Gazebo 开发碰撞检测、物理更新等底层物理解算，再基于 ROS 来进行机器人功能模块，如运动算法的开发。这样既降低了开发难度，又缩短了仿真与实物之间的鸿沟。然而，Gazebo-ROS 架构下实时仿真所支持的机器人规模非常有限，这为大规模无人群体实时仿真带来了挑战。

rosflight 固定翼无人机基准包是典型的 Gazebo-ROS 架构下仿真应用案例，其仿真架构如图 5 所示。ROS 与 Gazebo 各司其职。其中，ROS 负责从 Gazebo 中获取模拟对象状态信息（如 Truth 话题）并解算生成控制指令（如 Command 话题）。Gazebo 则通过扩展插件（如 Aerodynamics 插件）将控制指令解算成力和力矩，然后施加到固定翼无人机上，用于无人机物理现象模拟求解，进而使固定翼无人机做出仿真动作。

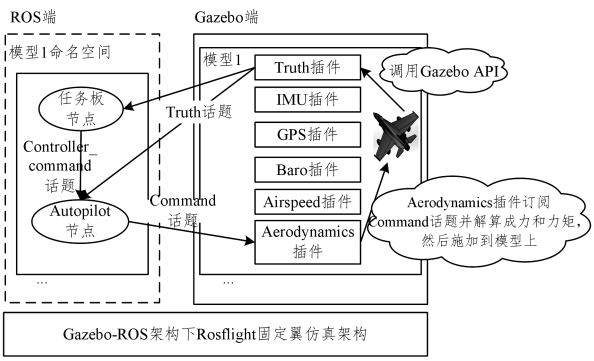

图 5　Gazebo-ROS 架构下 Rosflight 固定翼仿真架构

Fig. 5　Rosflight fixed wing simulation architecture based on Gazebo-ROS architecture

图 5 中 Truth 话题由 Truth 插件注册的回调函数生成并发布，Command 话题则由 Aerodynamics 插件注册的回调函数订阅并将其解算成力和力矩。Truth 话题（包含仿真模型位姿等信息）在实际应用中通常按照一定频率进行发布供 ROS 端控制器订阅使用，故可以将 Truth 插件注册的回调函数利用异步事件调用执行。而 Aerodynamics 插件注册的回调函数则需在每个时间步内均调用执行，以便随时订阅来自 ROS 端控制器的 Command 话题，以实现高保真模拟。图 5 中 IMU 插件、GPS 插件、Baro 插件、Airspeed 插件与 Truth 插件类似，周期性地根据仿真模型状态信息生成相关话题（如里程、GPS 信息等）并发布供 ROS 端控制器订阅使用。因此，可以采用异步事件机制优化重构上述插件。Aerodynamics 等插件的主要功能是订阅来自 ROS 端的控制指令话题进行解算，并将解算后的结果反馈给模型以影响模型的动作。因此，Aerodynamics 等插件的逻辑应保持不变，以保证仿真的正确性。

3.2　异步事件优化设计

如前所述，Gazebo 提供了 6 类扩展插件，其中，ModelPlugin 插件在模拟对象与 ROS 控制器间信息交互起到了桥梁的作用。然而，其向 Gazebo 注册的回调函数在每个时间步迭代预处理阶段以同步阻塞的方式调用执行，大大影响了仿真性能。因此，基于异步策略在 Gazebo 中增加异步事件接口支持，每个时间步迭代预处理阶段以异步方式回调执行 ModelPlugin 插件注册的函数，进而避免了仿真主线程核心物理求解过程的迭代循环。

增加异步事件接口后，Gazebo 仿真流程如图 6 所示。Gazebo 启动后，首先加载仿真世界与仿真模型，并基于插件机制向 Gazebo 注册同步事件与异步事件，设定事件周期。随后，Gazebo 同时开启和维护两个线程，一个为仿真主线程，另一个为异步事件子线程。仿真主线程主要用于循环执行基于时间步的模拟对象物理求解过程，并负责向异步事件线程发送异步触发信号，按需触发异步事件的执行。异步事件线程则负责监听仿真主线程发送过来的异步信号，接到异步信号后，首先判断是否满足异步事件执行频率，若满足则进行异步事件注册回调函数的调用执行，若不满足则继续等待监听来自仿真主线程的异步事件信号。

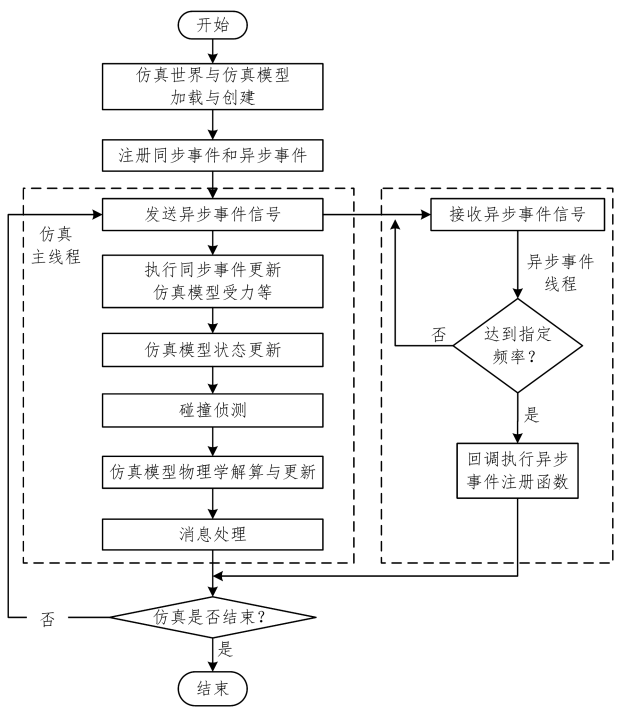

图 6　基于异步事件的 Gazebo 仿真流程

Fig. 6　Gazebo simulation process based on asynchronous events

为了进一步描述异步事件优化执行逻辑，给出了该优化方案下的伪代码，如图7、图8所示。图7描述了Gazebo迭代循环主线程的伪代码，图8描述了异步事件处理子线程的伪代码。

```
Thread 1  gazebo simulation iteration loop
Input: world、robots、commands
Output: robots' pose
1. loadWorldAndModels() //加载仿真世界和模型
2. registerAndSetupEvent() //注册并设置预处理事件
3. while meetIterLoop() //判断是否继续仿真
4. do
5.      sendAsynSignal() //向异步线程发送触发信号
6.      execSyncCallback() //执行同步事件回调
7.      updatePoseForModels() //更新模型位姿等
8.      collisionDetect() //碰撞检测阶段
9.      physicSimAndUpdate() //动力学响应阶段
10.     processMsg() //一步迭代结束，发布位姿信息
11. end
```

图7 Gazebo迭代循环主线程伪代码

Fig. 7 Pseudo code of Gazebo iteration loop in main thread

```
Thread 2  asynchronous event processing loop
Input: asynchronous signal
Output: execution of asynchronous event callback function
1. while meetIterLoop() //判断是否继续仿真
2. do
3.   if signalIsReceived() //判断触发信号是否到来
4.   then
5.     if frequencyIsReached() //频率是否满足
6.     then//回调执行异步事件逻辑
7.       call the function that asynchronous event register
8.     else
9.       continue
10.  else
11.    continue
12. end
```

图8 异步事件处理子线程伪代码

Fig. 8 Pseudo code of asynchronous event in child thread

3.3 消息处理异步化

如前所述，Gazebo由gzserver与gzclient两部分组成。gzserver主要负责仿真模型的基于时间步物理求解推进，gzclient仅仅负责可视化渲染。两者通过话题订阅与发布方式进行信息交互，gzclient订阅gzserver在每步迭代结束周期性生成发布的仿真模型状态信息来进行可视化渲染显示，gzserver则订阅gzclient按需生成发布的各种配置话题信息，来实现仿真过程的动态配置。

消息处理过程主要是gzserver在每步迭代结束后周期性生成与发布仿真模型位姿(包括位置和朝向)、时间戳等话题信息，这一过程是以同步阻塞方式完成的，在一定程度上降低了仿真性能。与异步事件优化类似，消息处理的话题只要按照一定频率发布即可满足gzclient可视化渲染需求，因此，可以采用异步策略优化重构消息处理过程。消息处理异步化后，Gazebo仿真流程如图9所示。

类似异步事件优化，消息处理异步化单独开启一个线程执行消息处理逻辑，其执行流程如下：

1)等待接收异步信号；

2)异步信号到来，判断是否达到指定频率，若是执行步骤3)，若否转到步骤1)；

3)发布仿真最新状态信息；

4)判断仿真结束条件是否满足，若是结束线程执行，若否转到步骤1)。

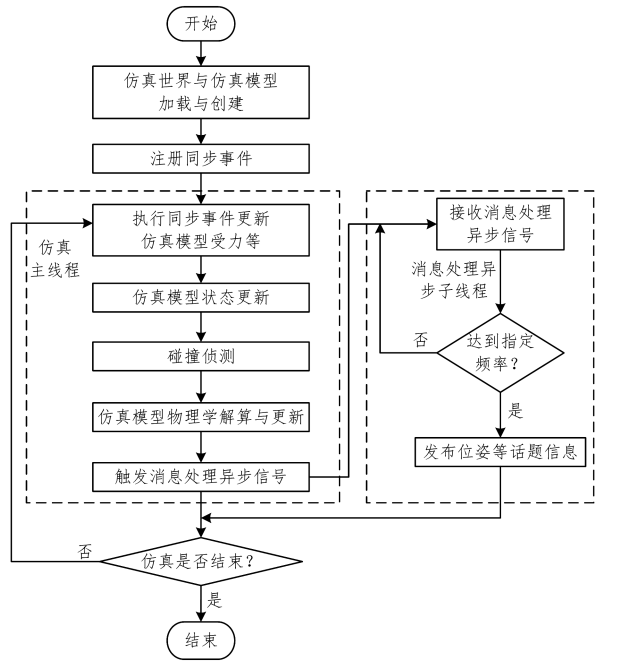

图9 消息处理异步化仿真流程

Fig. 9 Message processing simulation flow based on asynchronous strategy

为了更加详细地描述消息处理异步化逻辑，图10给出该方案下子线程处理逻辑的伪代码。

```
Thread 3  message processing loop based on asynchronous mechanism
Input: asynchronous signal
Output: advise robots' pose topic(location, orientation, time stamp and so on)
1. while meetIterLoop()
2. do
3.   if signalIsReceived() //判断是否继续仿真
4.   then
5.     if frequencyIsReached()//达到频率?
6.     then //发布机器人位姿信息
7.       advise robots' pose topic
8.     else
9.       continue
10.  else
11.      continue
12. end
```

图10 消息处理异步化伪代码

Fig. 10 Pseudo code of message processing

消息处理异步化后，其代码执行逻辑不再阻塞仿真主线程的执行。但由于多线程对CPU、内存等硬件资源存在竞争关系，异步化后的逻辑代码执行仍会间接影响仿真性能。

在一次仿真过程中同时开始异步事件及消息处理异步化优化后，在仿真主线程外会另外开启两个异步逻辑子线程，分别用于执行异步事件及消息处理异步化代码逻辑。此时，仿真流程如图11所示。

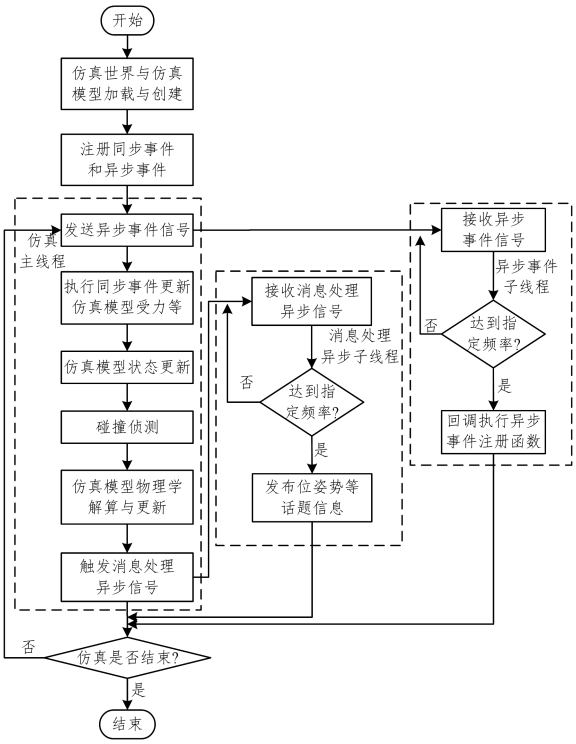

图 11　同时开启异步事件及消息处理异步化仿真的流程图

Fig. 11　Flow chart of asynchronous event and message processing based on asynchronous strategy

4　实验测试及数据分析

为了验证基于异步策略优化重构方案的有效性，基于 rosflight 基准案例包进行了不同架次规模验证测试实验。rosflight 软件包是为了加速微型飞行器自主能力的研发而开发的一套飞行控制框架[18]。其整合了机器人操作系统、开源软件及硬件，使自动驾驶仪的代码开发变得更简单、更高效。rosflight 提供了两种无人机模型——固定翼及四旋翼无人机，并且融合了实物固件功能包及仿真功能包。Rosflight 软件包的详细介绍[1)]及其源码[2)]请参考相关网址。

所有的测试实验在 Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz 共 6 个物理核心 12 个逻辑核心、DDR4 64GiB 内存、GeForce GTX 1050Ti 独立显卡硬件平台，Ubuntu 16.04 及 Ros Kinetic 等系统软件平台上，以 1ms 的时间步仿真迭代运行 100 000 步，统计出仿真耗时、仿真实时率(Real Time Factor，简称 RTF，其定义为仿真时间与真实时间比)等性能数据。图 12 给出了 100 架次 rosflight 固定翼无人机仿真场景图。

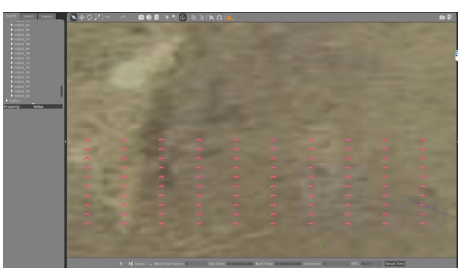

图 12　100 架次 Rosflight 固定翼仿真场景

Fig. 12　Simulation scene of 100 Rosflight fixed wing UAVs

针对无摄像头固定翼无人机、带摄像头固定翼无人机两种仿真模型构建基准测试案例，案例中 rosflight 固定翼做简单的向前向上飞行动作，并按以下 4 种场景进行对比实验：1)未作优化(case 1)；2)开启消息处理异步化优化(case 2)；3)开启异步事件机制优化(case 3)；4)同时开启消息处理异步化、异步事件机制优化(case 4)。4 种场景下的收集的数据均在开启仿真 GUI 情况下获取。

图 13、图 14 分别为无摄像机无人机仿真 *RTF*、仿真总耗时随无人机架次变化趋势图。从图中看出，相比原始逻辑，异步消息与异步事件机制优化策略分别有不同层次的加速效果，其中异步事件加速效果尤为明显。当同时开启异步消息与异步事件机制优化策略时，仿真性能表现最好。其中，100 架次无人机仿真优化后，*RTF* 从原来的 0.53 加速到了 2.98，加速比为 5.62。

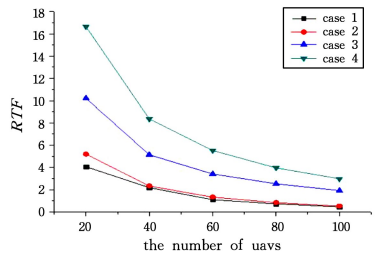

图 13　无 Camera 无人机仿真 *RTF* 随飞机架次变化图

Fig. 13　*RTF* line chart of UAVs without Camera

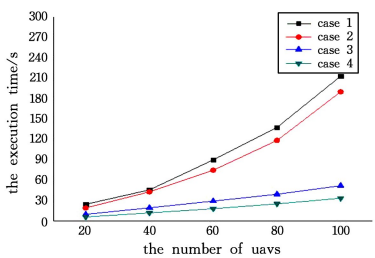

图 14　无 Camera 无人机仿真总耗时随飞机架次变化图

Fig. 14　Execution time of UAVs without Camera

图 15、图 16 分别给出了带摄像头无人机仿真 *RTF*、仿真总耗时随无人机架次变化趋势图。从图中看出，虽然仿真模型因多了摄像头传感器而变得更加复杂，但异步消息与异步事件机制优化策略仍取得了不同层次的加速效果。类似无摄像头仿真模型，同时开启异步消息与异步事件机制优化策略时，仿真性能表现最好。此时，100 架次无人机仿真优化后，*RTF* 从原来的 0.4 加速到了 2.17，加速比为 5.43。

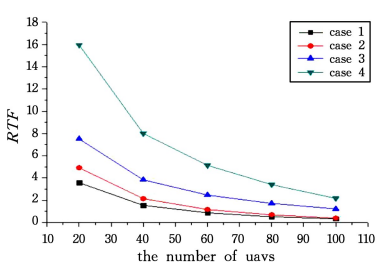

图 15　带 Camera 无人机仿真 *RTF* 随飞机架次变化图

Fig. 15　*RTF* line chart of UAVs with Camera

1) http://wiki.ros.org/rosflight

2) https://github.com/rosflight

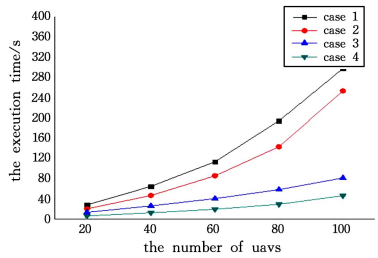

图 16 带 Camera 无人机仿真总耗时随飞机架次变化图

Fig. 16 Execution time of UAVs with Camera

异步事件优化与每个仿真模型挂载的扩展功能插件数量及插件注册回调函数的执行代码量有关。当外挂功能插件数量越多，回调函数执行代码量越大时，利用异步事件优化后，仿真加速效果越好。消息处理异步化则与仿真模型挂载传感器及 gzclient 开启数量有关，传感器挂载数量越多，gzclient 开启数量越多，则利用消息处理异步化优化后，仿真加速效果越好。经过优化后，Gazebo 所支持的实时仿真规模有了很大的提升。

结束语 本文提出了一种基于异步策略的 Gazebo 仿真循环优化重构方案，并基于 rosflight 软件包进行了仿真测试实验，验证了方案的有效性。经过异步策略优化重构后，其有效避免了 Gazebo 仿真核心外代码对仿真核心物理求解的同步阻塞，Gazebo 仿真性能及其所支持的实时仿真机器人规模均得到了很大的提升。实验过程中发现在异步优化后，Gazebo 仿真核心部分——物理求解成为了大规模仿真瓶颈。为了实现物理求解的高效运行，主要解决办法是寻找更加高效的迭代求解算法或对该部分进行并行。虽然，目前已有物理求解方面的并行优化研究成果，然而，这些并行优化对 Gazebo 仿真加速效果有限，仍然不能有效满足日益增长的 Gazebo 大规模高精度实时仿真的需求。因此，下一步工作将聚焦于如何有效加速 Gazebo 仿真核心部分——物理解算之上。

参 考 文 献

[1] MEYER J, SENDOBRY A, KOHLBRECHER S. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo[J]. Lecture Notes in Computer Science, 2012: 400-411.

[2] ROBERTS D J, WOLFF R, OTTO O. Constructing a Gazebo: Supporting Team Work in a Tightly Coupled, Distributed Task in Virtual Reality[J]. Presence, 2003, 12(6): 644-657.

[3] FOLGADO E, RINCÓNM J R, ÁLVAREZ R. A Multi-robot Surveillance System Simulated in Gazebo[C]// International Work-Conference on the Interplay Between Natural and Artificial Computation. 1970.

[4] FURRER F, BURRI M, ACHTELIK M. RotorS-A Modular Gazebo MAV Simulator Framework[M]. Springer International Publishing, 2016.

[5] HSIEH M A, KUMAR V, CHAIMOWICZ L. Decentralized controllers for shape generation with robotic swarms[J]. Robotica, 2008, 26(5): 691-701.

[6] FAIGL J, KRAJNIK T, CHUDOBA J, et al. Low-cost embedded system for relative localization in robotic swarms[C]// IEEE International Conference on Robotics & Automation. IEEE, 2013.

[7] BACHRACH J, BEAL J, MCLURKIN J. Composable continuous-space programs for robotic swarms[J]. Neural Computing & Applications, 2010, 19(6): 825-847.

[8] KOENIG N, HOWARD A. Design and use paradigms for Gazebo, an open-source multi-robot simulator[C]// 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004(IROS 2004). IEEE, 2004.

[9] LU Y. A framework for problem standardization and algorithm comparison in multibody system[C]// International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. 2014.

[10] DRUMWRIGHT E, SHELLD . An evaluation of methods for modeling contact in multibody simulation[C]// IEEE Int. Conf. on Robotics and Automation, 2011: 1695-1701.

[11] SHAMSHIRI R R, HAMEED I A, PITONAKOVA L, et al. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison[J]. Int. J. Agric. & Biol. Eng., 2018; 11(4): 15-31.

[12] TODOROV E, EREZ T, TASSA Y. Mujoco: A physics engine for modelbased control[C]// IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2012: 5026-5033.

[13] TODOROV E. A convex, smooth and invertible contact model for trajectory optimization[C]// IEEE Int. Conf. on Robotics and Automation. 2011: 1071-1076.

[14] ROGLIATO B, DAM A T, PAOLI L, et al. Numerical simulation of finite dimensional multibody nonsmooth mechanicalsystems[J]. Applied Mechanics Reviews, 2002, 55: 107-150.

[15] JIA Y B. Three-dimensional impact: energy-based modeling of tangential compliance[J]. Int. J. Robotic Research, 2013, 32(1): 56-83.

[16] DURIEZ C. Control of elastic soft robots based on real-time finite element method[C]// IEEE International Conference on Robotics and Automation. IEEE, 2013: 3982-3987.

[17] DURIEZ C, DUBOIS F, KHEDDAR A, et al. Realistic haptic rendering of interacting deformable objects in virtual environments[J]. IEEE Transactions on Visualization and Computer Graphics, 2006, 12(1): 36-47.

[18] JACKSON J, ELLINGSON G, MCLAIN T. ROSflight: A lightweight, inexpensive MAV research and development tool[C]// 2016 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2016.

[19] FEATHERSTONE R. Rigid Body Dynamics Algorithms[M]. Springer, New York, 2008.

[20] HOLLARS M, ROSENTHAL D, SHERMAN M. SD/FAST user's manual[J]. Bioinformatics, 1991, 20(2): 3258-3260.

[21] CORKE P. A robotics toolbox for MATLAB[J]. Robotics & Automation Magazine, IEEE, 1996, 3(1): 24-32.

ZENG Lei, born in 1989, master. His main research interests include computer simulation and high performance computing.

LI Hao, born in 1990, Ph. D, assistant researcher. His main research interests include parallel computing, robotics and computer simulation.