# 使用模仿学习与游戏引擎物理进行角色模拟

Jo˜ao Rodrigues*NOVA LINCS*
*NOVA* 科学与技术学院卡帕里卡，葡萄牙
jfpe.rodrigues@campus.fct.unl.pt

Rui N´obrega*NOVA LINCS*
*NOVA* 科学与技术学院卡帕里卡，葡萄牙 rui.nobrega@fct.unl.pt

arXiv:2301.02123v1 [cs.GR] 5 Jan 2023

摘要——**对于缺乏使用学习算法或创建可视化环境来运行基于智能体仿真经验的人来说，创建能够与 AI 同伴及虚拟环境互动的视觉三维感知角色可能是一项艰巨的任务。本文探索将游戏引擎用作创建和运行图形仿真的工具，并通过 Unity3D 游戏引擎的 ML-Agents 等插件实现三维感知角色。这样便可使用现成算法对智能体进行仿真，并利用游戏引擎的马达来呈现这些智能体。我们探讨如何利用这些工具为游戏创建视觉机器人，并教会它们如何玩游戏，直到它们达到足以在互动游戏中充当真实玩家对手的水平。**

索引词—**交互、游戏、ML-Agents、模仿学习**

## I. 引言

游戏一直使用交互式角色和非玩家角色（NPC），让游戏更有趣、更吸引人，也更容易引发共鸣。随着时间推移，智能体与角色仿真被用于许多不同目的 [1],，从预测大流行病传播等科学研究 [2]，，到休闲娱乐 [3]；游戏中的非玩家角色可以增加互动性，或提高游戏难度。使用这些仿真需要深入了解其所需的各种人工智能、机器学习和深度学习算法。此外，其中许多仿真还需要图形环境，通过渲染技术和物理仿真展示结果。所有这些技术都需要专业知识，而计算机图形开发人员可能并不具备这些知识，尤其是在人工智能和机器学习领域。

智能体仿真还涉及另一个因素，即需要创建具有感知能力的角色。许多基于智能体的建模（ABM）与仿真场景都要求角色能够与环境或同伴互动。为此，需要创建能够感知周围环境的角色，并模拟视觉、触觉、距离等因素。从头开始创建这些感知能力，同样需要高深的编程知识，例如用于视觉和距离感知的光线追踪，或用于触觉的物理。

所有这些需求都给想要创建角色仿真、却缺乏开发上述所有技术所需专业知识的人带来了难题。为了解决因需要构建带有感知角色的图形环境而产生的问题，可以考虑使用游戏引擎来呈现环境并模拟感官。游戏引擎不断发展，旨在简化游戏创作流程；由于它们已具备渲染引擎和物理引擎，能够实现虚拟环境的规则，例如射线投射距离、重力或碰撞，因此似乎是创建和训练角色及环境的理想工具。

除了使用游戏引擎之外，多年来人们还开发了许多工具，通过提供实用功能，简化这些仿真的创建与运行，并促进对不同机器学习和深度学习算法的测试与共享，例如 OpenAI Gym [4] 和 PettingZoo [5]。在考虑使用游戏引擎时，ML-Agents 插件 [3],尤其值得关注，因为它专为与 Unity3D 游戏引擎集成而创建，旨在简化游戏机器人的创建与学习。

最后提到的 ML-Agents 工具包含一项实用功能，可用于教机器人如何玩游戏，即演示记录器。该工具允许记录人类在与机器人相同的环境中进行游戏的步骤，收集机器人在游戏过程中会接收到的观测，并生成演示文件，以便之后用于教机器人如何玩游戏。

本文提出创建一组游戏场景，并利用它们教机器人如何玩游戏。不仅如此，我们还将研究如何利用人类游戏记录来教机器人玩游戏。随后，我们将比较这种方法能否取得与人类水平的游戏表现相近的结果。

为了更好地了解如何利用游戏引擎为基于智能体的仿真提供更出色的图形界面，我们将回顾这些领域中的一些相关研究，以进一步理解智能体仿真、游戏引擎以及可用于连接二者的工具库中的若干概念。随后，我们将阐述本研究背后的思路与概念，以便更好地理解 ML-Agents 的工作原理，以及如何利用它提升学习型机器人的性能。

## II. 相关工作

游戏引擎的主要用途是通过渲染图形、生成音乐和音效来制作游戏，并允许

通过读取输入设备的输入来从外部操控场景。通常，游戏引擎由渲染引擎、动画引擎、物理引擎、人工智能引擎、网络引擎、3D音效引擎和地图编辑器组成 [6]。

虽然顾名思义，游戏引擎的主要用途是制作游戏，但它的功能也支持动画和仿真等更多用途。渲染引擎、动画引擎和物理引擎可以简化动画制作。借助这些引擎，只需选取一个场景进行动画制作，编写预期的运动代码，然后播放结果即可；运动和碰撞 [6]都会遵循物理规律，无需逐帧制作每个动作。与动画制作类似，这些工具也可用于生成仿真。

游戏引擎之所以适用于动画和仿真，是因为它能让用户查看创作结果，而无需编写底层代码来渲染和制作场景动画，因为这些工作由游戏引擎在内部完成 [7]。对于智能体仿真而言，游戏引擎能够操控智能体模型的运动，并且只需向场景中添加游戏对象、调整其属性，就能为智能体创建仿真环境，无需编写大量代码来描述仿真中每个对象的形状、物理属性和运动方式，这一点尤为重要。使用游戏引擎进行仿真的部分相关研究包括 [8]–[10]。

### *A.* 图形化智能体仿真

图形化基于智能体的建模与仿真是一种范式，其中模拟的人类、动物或其他生命体被建模为智能体，它们与同伴及其环境进行交互 [11]。在这些基于智能体的仿真中（有时也称为多智能体仿真），环境发挥着至关重要的作用，因为它会影响所有智能体及其交互，因此必须予以审慎考虑。

为了创建基于智能体的仿真，需要考虑四个要素：智能体集合、交互集合、环境和仿真基础设施 [12]。

自博弈强化学习是指智能体通过探索并与自身对弈来学习。它已成功应用于许多游戏场景；然而，自博弈学习的过程并不稳定，且与一般强化学习相比样本效率更低，尤其是在不完全信息博弈中 [13]。多智能体自动课程已用于解决各种类型的多人游戏，包括双陆棋 [14] 和国际象棋 [15]，等经典离散游戏，以及 Dota 和星际争霸等连续实时游戏。 [13], [16]–[18] 展示了使用视觉场景进行智能体仿真和自博弈强化学习的研究实例。

### *B.* 智能体仿真库

为了在游戏引擎环境中创建感知智能体，关键是找到一个能够仿真其动作的库，而无需编写复杂的算法。幸运的是，如今这类开源库越来越多，只要符合库的规范，就可以将其集成到任何类型的仿真中。

Unity机器学习智能体 [3](ML-Agents) 是一个工具包，可让游戏和仿真环境用于训练智能体。该库专用于 Unity 的 3D游戏引擎，并提供了最先进方法的实现，使游戏开发者和研究人员能够轻松训练智能体，用于二维、三维和 VR/AR游戏。借助 ML-Agents，智能体可以通过所提供的易用 Python API，使用强化学习、模仿学习、神经进化及其他一些方法进行训练。

## III. 仿真与训练工作流

如前所述，目标是获取人类进行游戏的记录，并利用所获得的数据训练机器人。

在本文中，这些记录将称为演示。演示是一种数据结构，用于保存人类每一时刻获得的观测，并将其映射到他们采取的动作，以便机器人能够通过在演示中寻找观测与当前观测相似的动作来决定采取何种动作。

为了更好地了解如何利用演示记录提高机器人的学习速度，首先需要了解如何创建包含学习型机器人的场景，以及 ML-Agents 如何教会它们。对于拥有游戏开发经验的人来说，创建场景相当简单。第一步是构想所需的交互场景，然后在 Unity3D 中构建该场景，放置不同的游戏对象和用于控制所需交互的脚本。

之后，每个机器人都会连接到 ML-Agents 接口，其行为会考虑到动作
它接收来自系统的信息，以及与之交互的环境所提供的观测。每个机器人都会发送一组变量和值，构成一次观测，并提供机器人到世界的光线追踪历史记录。

在 Unity3D 外部，ML-Agents 库通过配置文件运行，该文件定义了运行标识、标志和具体的训练算法。训练算法可分为两类：强化学习算法和模仿算法。对于强化学习算法，ML-

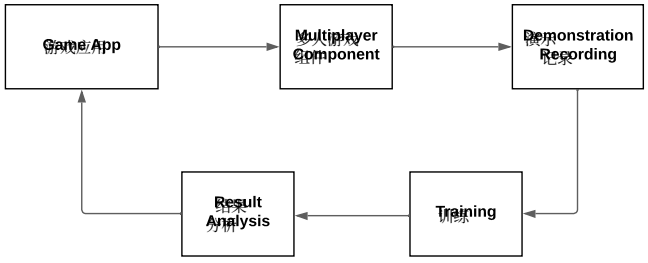


图 1. 演示记录流程图

Agents 提供了适用于单智能体场景的近端策略优化（PPO）和软演员-评论家（SAC），适用于多智能体合作/竞争场景的多智能体事后信用分配（MA-POCA），以及可用于这两种情况的自博弈。

对于模仿算法，ML-Agents 库[1] 提供行为克隆和生成对抗模仿学习（GAIL），利用示范记录进行模仿学习。这些算法可以结合使用，以更好地强化模仿学习；也可以与强化学习算法配合使用，让机器人在演示内容的基础上继续学习，而不只是将其作为学习的起点。

了解了如何利用 ML-Agents 在 Unity3D 中创建能够自主学习的机器人之后，我们可以进一步探讨示范记录如何帮助机器人学习。首先，来看看这些示范记录究竟是什么。在 Unity 中使用 ML-Agents 软件包时，可以为智能体附加一个演示记录器。附加后，该记录器会保存一种数据结构，其中记录了人类玩家执行的所有动作，以及每个动作时刻作为观测提供的所有变量和光线投射命中信息。这就构成了用于模仿学习的演示，因为智能体随后可以利用这些数据结构来决定采取什么动作：将自己的观测与演示文件中的观测进行比较，找出最相似的一项，并执行演示中与该观测对应的动作。这意味着，为了生成可用于学习的示范，我们可以让人类用户游玩这些场景并完成任务，为机器人提供参考。这之所以可行，是因为 ML-Agents接口支持启发式函数，可用于为机器人创建人工控制。

这一方法的思路是，对于那些机器人需要很长时间才能学会、且配置文件更难调优的复杂场景，我们可以利用这些演示来加速这一过程，更快地创建出可用的机器人。为此，我们首先要创建一款既适合智能体也适合用户游玩的游戏。我们还需要定义智能体获得的观测和奖励。如果这款游戏涉及多个智能体同时参与，就必须支持多人游戏，这样多个用户也可以同时游玩。完成以上工作后，我们就可以让用户游玩游戏并记录演示。用户越多、游玩时间越长越好。收集完所有演示后，就可以开始训练机器人。随后可以分析机器人的最终表现；如果仍未达到预期，就可以重复这一过程，收集更多演示，也可以让当前机器人作为用户的对手。图1展示了这一过程的流程图。

[1]ML-Agents，https://github.com/Unity-Technologies/ML-Agents/blob/main/docs/Training-ML-Agents.md

为了测试使用演示记录提高机器人学习速度这一想法，我们决定在如图2所示的3对3玩家竞技环境中进行夺旗游戏。选择这款游戏，是因为我们希望它兼具竞争性和协作性，并具有适度的复杂度，使性能水平的差异易于区分，同时又不会难到让机器人和人类难以学会玩法。

在这款游戏中，两个队伍（蓝队和白队）相互竞争，试图夺取敌方旗帜，并将其一路带回己方场地。每次夺旗并将旗帜带回己方场地的过程，在训练中都算作一个回合。此外，场地上分布着一些墙壁，作为玩家的障碍物；还有一些球，玩家可以接住并投掷，以尝试击中其他玩家。如果玩家在持旗时被击中，就会掉落旗帜并眩晕3秒。旗帜掉落后，两个队伍都可以尝试接住它：如果是敌方旗帜，就将其夺回；如果是己方旗帜，则使其返回出生点。图2展示了这款游戏的示意图。该游戏支持多名用户同时游玩，因此需要为其实现多人游戏组件。这样，人类用户便能通过与其他人类对战，在竞技水平下进行游戏并录制演示。

## IV. 初步结果

在这次初步实验中，人类用户进行了 3 场游戏，以录制演示。在这些游戏场次中，共获得了 10 个演示。录制环节结束后，使用不同数量的示范对机器人进行了多次训练。

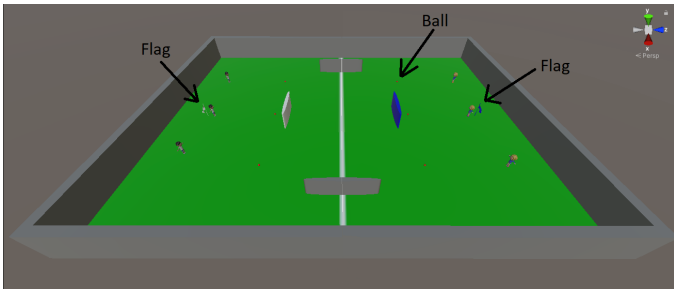


图2. 夺旗场景：玩家需要夺取敌方旗帜，并将其带回己方场地

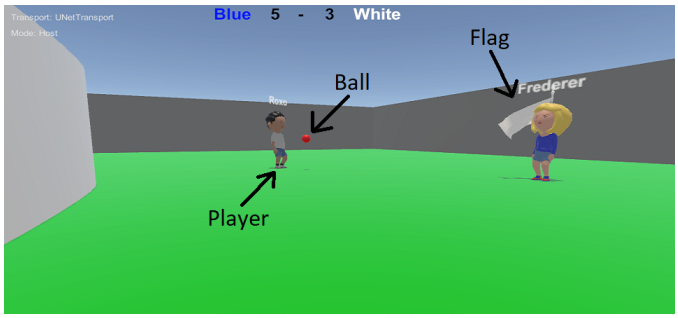


图3. 游戏画面截图，显示所有网络对象

从这些初步结果可以看出，使用的演示越多，机器人的表现越好。每回合游戏所需的时间从使用 1 个示范训练的机器人平均 517 秒，降至使用 10 个示范时的 165 秒。平局（1000 秒后仍无队伍获胜）的数量也随着示范数量增加而减少：使用 1 个示范时，20% 的游戏以平局告终；使用 10 个示范时，这一比例仅为 7%。

在录制环节中，我们注意到，位于场地左侧的队伍（蓝队）在所有进行过的回合中总体胜率为 68.9%，并且在三个游戏场次中赢得的回合数都更多。这种优势也体现在机器人身上，而且随着用于训练的演示增多，优势进一步扩大。仅使用 1 个演示训练机器人时，蓝队赢得了 57 % 的游戏，略占优势；而使用 10 个演示训练机器人时，蓝队的胜率达到 81%，占据了巨大优势，甚至超过了人类蓝队的优势。

回顾每个回合所用的时间，我们分析了各队在获胜回合中结束游戏的速度。出乎意料的是，白队获胜时平均只用 50 秒就能结束游戏，比蓝队快 50 秒；蓝队获胜时平均耗时为 100 秒。对此的一种初步解释是，白队只有在理想情况下才能获胜，而蓝队即使游戏拖得稍久，也能找到取胜的方法。

## V. 结论

本文介绍了一种方法，可在 Unity3D 中创建使用模仿算法的游戏引擎自学习机器人。该技术借助前文所述的 ML-Agents 插件实现。随后，我们重点探讨了这一假设：利用示范记录进行模仿学习，可以有效地为环境复杂的游戏创建机器人。为此，我们创建了一款 3对3 的夺旗游戏，并让多名人类用户同时录制演示。

根据所得结果，我们得出结论：模仿学习可以成为训练适用于游戏的机器人的有效方法，但仍有必要开展进一步研究。首先，需要录制更多演示，以便在拥有大量演示样本时分析机器人的水平，并通过人类与机器人之间的游戏对局，将其与人类水平进行比较。随着样本量增加，若采用更加均衡的结果集，观察结果也会很有意思：届时两支队伍的胜率是否相近，抑或其中一支队伍仍会始终占据优势。最后，还应将这种教学方法拓展到其他游戏中，考察训练出的机器人在复杂程度各异的游戏中的表现和水平，并了解哪些变量有助于或妨碍机器人通过模仿进行学习。


## 致谢

本研究得到 NOVA.ID.FCT/NOVA LINCS (UIDB/04516/2020) 的支持


## 参考文献


[1] C. Macal 和 M. North，“基于智能体的建模与仿真”，2009年12月。[2] K. M. Khalil、M. Abdel-Aziz、T. T. Nazmy 和 A.-B. M. Salem，《埃及大流行性流感的基于智能体建模》。柏林，海德堡：Springer Berlin Heidelberg，2012年，第205–218页。 [在线]。获取地址：https://doi.org/10 1007 978 3 642 25755 1 11 。/ - - - - A. Juliani、V.-P. [3] Berges、E. Teng、A. Cohen、J. Harper、C. Elion、C. Goy、Y. Gao、H. Henry、M. Mattar 和 D. Lange，“Unity：智能体通用平台”，2018年9月。 [在线]。获取地址：http://arxiv.org/abs/1809 02627 。G. Brockman、V. Cheung、[4] L. Pettersson、J. Schneider、J. Schulman、J. Tang 和 W. Zaremba，“OpenAI Gym”，2016年6月。 [在线]。获取地址：http://arxiv.org/abs/1606 01540 。J. K. Terry、B. Black、N. Grammel、M. [5] Jayakumar、A. Hari、R. Sullivan、L. Santos、R. Perez、C. Horsch、C. Dieffendahl、N. L. Williams、Y. Lokesh 和 P. Ravi，“PettingZoo：用于多智能体强化学习的 Gym”，第35届神经信息处理系统大会，9 2020 [ ] 2009 14471 。在线。获取地址：[6] http://arxiv.org/abs/。S. S. Noh、S. D. Hong 和 J. W. Park，“利用游戏引擎技术制作三维娱乐内容”，载于第16届人工现实与远程存在国际会议研讨会（ICAT’06），2006年，第246–251页。[7] A. Bak 和 M. Wojciechowska，“在动画制作流程中使用游戏引擎”，第209–220页，2020年。[8] C. Bartneck、M. Soucy、K. Fleuret 和 E. B. Sandoval，“机器人引擎——让 Unity 3D游戏引擎用于人机交互”，第24届IEEE机器人与人类交互通信国际研讨会，2015年。 [在线]。获取地址：https://www.openrobots.org/ wiki/morse/[9]M. Pasternak、N. Kahani、M. Bagherzadeh、J. Dingel 和 J. R. Cordy，“SimGen：一种用于在 Unity 游戏引擎上生成嵌入式系统仿真与可视化的工具”，载于 MODELS 2018，丛书 MODELS ’18。美国纽约：计算机协会，2018年，第42–46页。 [在线]。获取地址：https://doi.org/10 1145 3270112 3270135 。/。A. Carlos、A. Mol、C. Alexandre、[10] F. Jorge 和 P. M. Couto，“利用游戏引擎进行疏散规划中的虚拟现实仿真”，IEEE计算机学会，2008年。 [在线]。获取地址：http://udn。[11] F. Kl¨ugl 和 A. L. C. Bazzan，“基于智能体的建模与仿真”，《AI Magazine》，第33卷，第3期，第29页，2012年9月。 [在线]。获取地址：https://ojs.aaai.org/index.php/aimagazine/article/view/2425
[12] ¨ F. Klugl、M. Fehler 和 R. Herrler，“关于环境在多智能体仿真中的作用”，第3374卷。Springer Verlag，2005年，第127–149页。[13] G. Brockman、V. Cheung、L. Pettersson、J. Schneider、J. Schulman、J. Tang 和 W. Zaremba，“OpenAI Gym”，2016年。 [在线]。获取地址：https://arxiv.org/abs/1606 01540 。G. [14] Tesauro，“时序差分学习与 TD-Gammon”，《Commun. ACM》，第38卷，第3期，第58–68页，1995年3月。 [在线]。获取地址：https://doi.org/10 1145 203330 203343 。/。D. Silver、T. Hubert、J. [15] Schrittwieser、I. Antonoglou、M. Lai、A. Guez、M. Lanctot、L. Sifre、D. Kumaran、T. Graepel、T. P. Lillicrap、K. Simonyan 和 D. Hassabis，“通过通用强化学习算法自博弈精通国际象棋和将棋”，CoRR，第abs/1712.01815卷，2017年。 [在线]。获取地址：http://arxiv.org/abs/1712. 01815[16] J. Yang、A. Nakhaei、D. Isele、K. Fujimura 和 H. Zha，“CM3：协作式多目标多阶段多智能体强化学习”，ICLR 2020，2020年。[17] OpenAI，“OpenAI Five”，https://blog.openai.com/openai-five/，2018年。[18] O. Vinyals、I. Babuschkin、J. Chung、M. Mathieu、M. Jaderberg 和 W. Czarnecki，“AlphaStar：精通即时战略游戏《星际争霸II》。” [在线]。获取地址：https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii