**用一个真实的机器人设置一个强化学习任务**

摘要---强化学习是一种很有前途的方法，可以为复杂多样的机器人任务开发难以设计的自适应解决方案。然而，使用真实世界的机器人学习往往是不可靠和困难的，这导致他们在强化学习研究中的应用率很低。由于缺乏设置机器人学习任务的指导方针，这一困难变得更加严重。在这项工作中，我们开发了一个UR5机器人手臂的学习任务，以揭示任务设置的一些关键元素，并研究它们对机器人挑战的贡献。我们发现学习性能对设置非常敏感，因此在设置细节上的疏忽和遗漏会使有效的学习、重现性和公平的比较变得困难。我们的研究提出了一些缓解措施，以帮助未来的实验者避免困难和陷阱。实验结果表明，该系统具有较高的可靠性和可重复性，为基于真实机器人的强化学习研究提供了可能。

I．介绍

尽管最近取得了一些成功，但在寻求通用强化学习(RL)代理时，现实世界中的机器人还没有得到充分利用，目前这主要局限于仿真。这种利用不足主要是由于机器人不可靠和不良学习成绩令人沮丧。尽管最近有几种RL方法在仿真中被证明是非常有效的，但当应用于实际任务时，它们往往会产生较差的性能。这种无效有时归因于现实世界的一些整体方面，包括数据收集速度慢、部分可观测性、噪声传感器、安全和物理设备的脆弱。这种障碍有助于依赖间接方法，例如模拟到现实的转移和集体学习，这有时可以弥补从单一实际经验中学习的失败。

在现实的RL研究中，一个经常被忽视的缺点是缺乏基准学习任务或标准来设置机器人实验。使用模拟机器人进行的实验通常是在基准任务上进行的，使用容易获得的模拟器和标准化的接口，从而减轻了实验人员的许

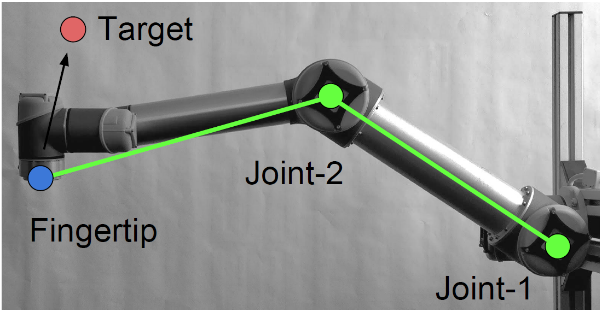


图 1 我们使用UR5机器人手臂来定义基于两个关节(如上所示)和六个关节的强化学习任务。奖励是指尖(蓝色)和随机目标位置(红色)之间的负距离。

多任务设置细节，如动作空间、动作周期时间和系统延迟。另一方面，用真实世界的机器人来设置学习任务远非显而易见。实验人员必须投入大量的工作，在学习主体和机器人之间建立一个特定于设备的传感器运动接口，并确定定义学习任务的环境的所有方面。这样的选择对于有效和可重复的学习性能是至关重要的。不幸的是，RL研究与真实世界机器人合作通常不会描述许多这些细节，更不用说以受控方式研究它们的影响，尽管存在一些值得注意的例外。

在这项工作中，我们的问题是如何建立一个现实世界的机器人任务，使一个现成的标准RL方法的实现可以有效和可靠地执行。我们通过为UR5机器人手臂开发一个Reacher任务来解决这个问题（参见图1），其中一个代理通过试错法学习到机械臂的低水平动作到达任意目标位置。这个任务在模拟中很容易解决，但在现实机器人中却很难。对于学习方法，我们使用TRPO的rllab实现，这是一种流行的学习方法，其超参数具有强大的性能。

在设置UR5 Reacher时，我们描述了设置真实RL任务的步骤和元素，包括数据传输媒介、并发性、计算的顺序和延迟、低级驱动类型以及操作它们的频率。通过探索这些元素的不同变化，我们通过机器人学习来研究它们对困难的个体贡献。我们发现，在动作执行前，时间延迟的可变性和选择动作代表的不平凡性对学习是非常有害的。通过考虑这些影响，我们不仅可以实现与现实世界机器人的有效性能，而且可以实现以高度可靠的方式从零开始学习的重复性，即使重复运行数小时在不同的时间使用不同的物理机器人。

II强化学习任务制定

强化学习（RL）任务由代理和相互交互的环境组成，正式建模为马尔可夫决策过程（MDP）。在MDP中，代理在离散的时间步t = 1,2,3，...与其环境交互，其中在每个步骤t，代理接收环境的状态信息St∈S和标量奖励信号Rt∈R 。代理使用由概率分布π（a | s）def == Pr {At = a | St = s}控制的随机策略π来选择动作At∈A。环境转换到新状态St + 1并使用转移概率分布在下一个时间步骤产生新的奖励Rt + 1：



代理的绩效通常根据其未来的累积奖励来评估，称为回报其中γ∈[0,1]是折扣因子。

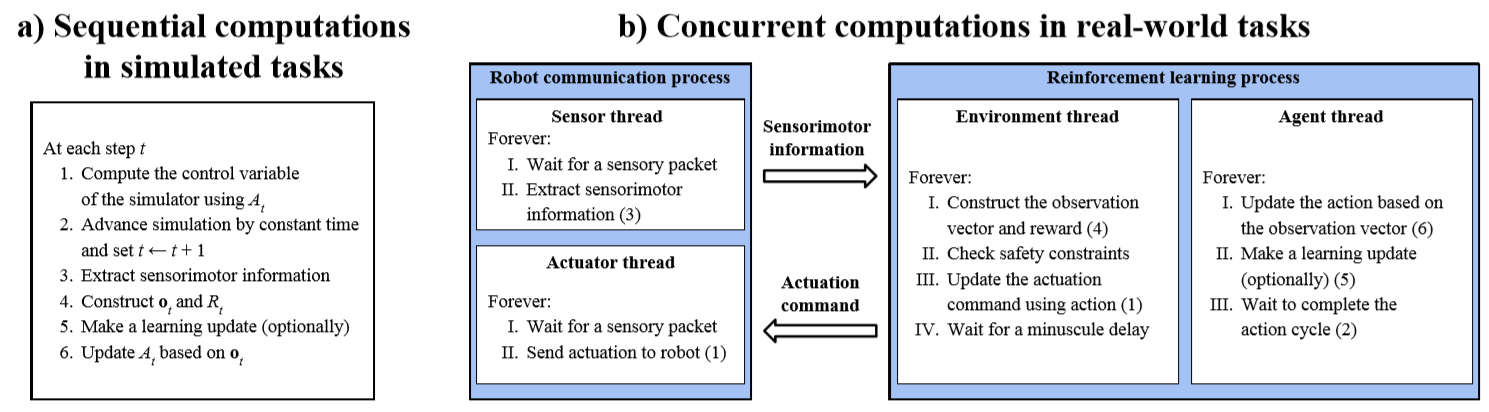
代理的目标通常是找到一个最大化预期回报的策略。这些策略通常是通过估计行为值(如Q-learning)或直接参数化策略并优化策略参数(如TRPO)来学习的。在实践中，代理并不接收环境的全部状态信息，而是通过一个实值观测向量ot对其进行部分观测。在此框架下，RL任务主要由三个要素来描述:观察空间、行动空间和奖励功能。

III．UR5 REACHER任务

在本节中，我们使用UR5机器人设计了一个Reacher任务，我们称之为UR5 Reacher。我们把它设计得类似于OpenAI-Gym Reacher，在这里，一个代理通过对一个模拟的双关节机器人手臂的直接扭矩控制来学习到达任意目标位置。通过使用神经网络非线性地参数化策略，诸如TRPO的策略搜索方法可以在几千个时间步骤中相当好地解决Gym Reacher。基于Gym Reacher设计任务允许我们设定合理的学习时间预期，利用已经做出的选择，并隔离硬件界面设计决策中出现的挑战。下面，我们将介绍UR5机器人的界面和UR5 Reacher任务的详细信息。

UR5是一款轻便、灵活的工业机器人，由通用机器人制造，有六个关节。UR5的低层机器人控制器称为URControl，可以通过TCP/IP连接进行通信来编程。机器人可以在脚本级使用一种称为URScript的编程语言进行控制。建立连接后，我们可以将URScript程序从计算机发送到URControl作为套接字上的字符串。URScript程序在URControl上实时运行，每8毫秒发送一次状态包。URControl中的每个包包含机器人的传感器运动信息，包括角位置、速度、目标加速度和所有关节的电流。该机器人可以通过URScript在同一个8毫秒的时钟上发送低层驱动命令来控制。URScript servoj命令提供位置控制接口，speedj提供速度控制接口。不像Gym Reacher，没有扭矩控制界面。

在UR5 Reacher中，我们从底座驱动第二个和第三个关节。我们还将其扩展到所有六个关节都被激活的任务，我们称之为UR5 Reacher 6D。观测矢量包括关节角度、关节速度、目标与指尖坐标的矢量差。与Gym Reacher不同的是，我们没有包含关节角度的正弦或余弦，也没有目标位置坐标，在不丢失基本信息的情况下，简化和减少了观察空间。我们将前面的操作作为观察向量的一部分，这对于在有延迟的系统中学习是有帮助的。在Gym Reacher中，奖励函数定义为：Rt = -dt -pt-1，其中dt是目标与指尖位置之间的



欧几里德距离，而pt是At的L2范数，用于惩罚大扭矩。我们使用相同的奖励函数，但通过省略惩罚项来简化它。UR5 Reacher由交互的片段组成，每个片段有4秒长，允许充分的探索。UR5的指针在UR5 Reacher中的2维0.7m×0.5m边界内以及UR5 Reacher 6D中的3维0.7m×0.5m×0.4m边界内被限制。每一段在边界内随机选择目标位置，手臂从边界中间开始。除了在边界内约束指尖外，还将机器人约束在关节角边界内，以避免自碰撞。

在仿真中很少研究实际任务的其他几个关键方面，如操作周期时间、连接介质、驱动类型的选择、计算中的并发性和延迟。这些方面是当前工作的重点。

IV实际任务设置的元素

在本节中，我们将描述设置实际学习任务的关键元素、每个元素的不同选择以及我们为基线UR5-Reacher设置所做的选择。

1. 计算的并发性、顺序和延迟

在模拟任务中，同步执行代理和与环境相关的计算是很自然的，这在实际任务中可能并不理想。图2(a)显示

了在一个典型的模拟实验中，在每个情节中依次执行的计算步骤。前四个

(1-4)计算步骤与环境相关，后两个

(5,6)与代理相关。在步骤2中，模拟的世界在时间上离散地前进，而在其余时间内没有变化。这样，模拟任务就可以符合MDP框架，在MDP框架中，观察和执行之间的时间不会提前。

在实际任务中，在每个代理和与环

境相关的计算过程中，时间都在不断前进。因此，代理总是对延迟的感觉运动信息进行操作。错误的同步和计算顺序会进一步放大总延迟，这可能会导致更困难的学习问题和减少响应控制的潜力。因此，设置学习任务的设计目标是管理和最小化延迟。提出了不同的方法来缓解这一问题，例如用操作来增加状态空间或预测操作执行的未来状态。这些方法并没有将延迟最小化，而是从学习代理的角度进行了补偿。这个问题很少讨论的一个方面是，不同的任务计算顺序或组合可能具有不同的总体延迟。

在UR5 Reacher中，我们用Python实现了计算步骤，并将它们分配到两个异步过程中:机器人通信过程和增强学习(RL)过程。它们交换感觉运动信息和驱动命令。图2(b)描述了UR5 Reacher的计算模型，它也可以作为其他实际任务的计算模型。2(b)中的一些计算步骤在与2(a)直接相关时以2(a)中的步骤号结束。机器人通信过程是一个设备驱动程序，它在一个传感器线程中以8ms的周期时间从URControl中收集传感器运动数据，并在一个单独的执行器线程中向URControl发送驱动命令。RL过程包含检查空间边界的环境线程，基于UR5感应电动机包计算观察向量和奖励函数，并基于快速循环中的动作更新致动器线程的致动命令。RL流程中的代理线程定义任务时间步骤并确定操作周期时间。它使用代理的策略传递进行学习更新并计算操作。

对于学习代理，我们使用TRPO的rllab实现。每隔几段，它就会偶尔执行一次代价高昂的学习更新。我们在每段之间安排这些更新，以确保它们不会干扰代理的感觉运动体验的正常过程。因此，TRPO的学习更新发生在代理线程中，但不是每个操作周期或时间步骤。将RL进程拆分为两个线程可以比同时检查操作更新更快地检查安全约束。此外，我们建议在更新操作之后再进行学习更新，这与模拟任务的步骤5和步骤6(图5a)不同，在模拟任务中，它们的计算顺序是相反的。

图5(b)中真实任务的计算模型建议并发性和一定的计算顺序，以避免不必要的系统延迟。例如，将机器人通信过程分成两个线程，允许与物理设备进行异步通信。这有助于在计算操作之后立即分派操作，而不是等待学习更新，这可能会增加从观察到操作的延迟。该计算模型还扩展到包含多个设备的机器人任务，每个设备有一个机器人通信过程，这允许代理快速访问传感器运动信息。

B.数据传输的媒介

很自然地，我们会考虑将一个具有有限机载计算能力的移动机器人与一个计算能力更强的基站通过Wi-Fi或蓝牙配对，而不是通过USB或以太网。Wi-Fi通常会导致流包到达时间的变化。在UR5 Reacher中，机器人通信过程通过TCP/IP连接与URControl通信。我们在基线设置中使用以太网连接，因为通过以太网通信可以更严格地控制系统延迟。不过，我们也测试了使用Wi-Fi连接的效果。图3显示了使用10,000个包测量有线和无线连接的包到达时间的变化。URControl每8毫秒发送一次数据包。对于有线连接，到达间隔时间始终在8ms左右，所有时间

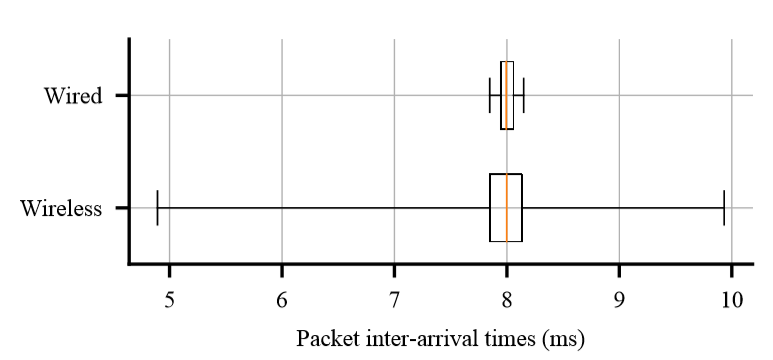


图 3 与UR5一起使用的有线和无线连接的包到达时间。框图显示了不同的百分位数(5,25,50,75,95)。

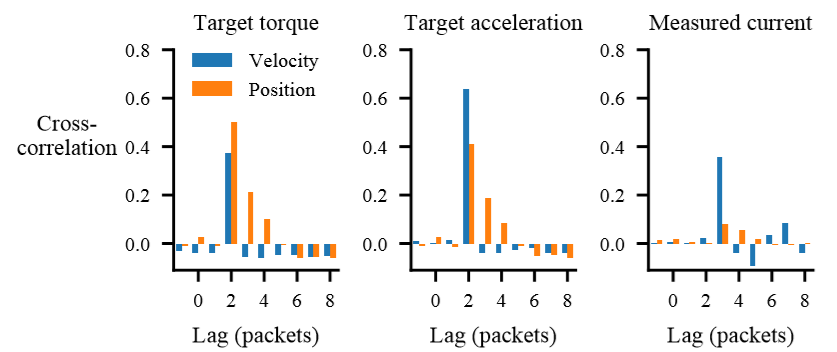
都在[7.8,8.6]ms之间。对于无线连接，有更大的可变性，整个范围在[0.2,127]ms之间变化。

C. 向机器人发送驱动命令的速率

不同的机器人设备以不同的方式工作。一些机器人允许外部计算机直接在它的控制表中写东西。机器人控制器根据控制表控制执行机构，不等待外部计算机的指令。其他一些机器人，如UR5，提供了一个接口，控制器根据外部计算机反复发送的驱动命令控制执行器。我们将这些命令从外部计算机传输到机器人称为机器人驱动。在UR5 Reacher中，我们选择机器人驱动周期时间默认为8ms。

D．动作周期

操作周期时间，也称为时间-步骤持续时间，是由代理的策略进行的两个后续操作更新之间的时间。为特定的任务选择一个周期时间并不明显，而且文献中缺乏对该任务设置元素的指导或调查。较短的周期时间可能包



动作(速度和位置控制)与后续电机信号(目标扭矩、目标加速度和测量电流)之间的关系

图 4 我们比较了基于速度和位置控制的动作，通过显示它们与UR5的三个电机信号的相互关系:目标扭矩(左)、目标加速度(中)和测量电流(右)。

括更好的策略和更好的控制。然而，如果具有太短循环时间的后续观察向量的变化对于代理是不可察觉的，则结果是现有学习方法难以或不可能的学习问题。较长的周期时间可能会限制可能的策略集和控制的精度，但也可能使学习问题更容易。如果循环时间过长，也会降低数据采集的速度，从而影响学习速度。

在我们的并行计算模型中，可以选择不同于机器人动作周期的动作周期时间。当动作周期较长时，执行器线程重复向机器人发送相同的命令，直到基于新动作计算出新的命令为止。这可能对动作周期较长的代理有利。例如，为了快速到达目标，手臂可以通过多次重复类似的机器人动作来获得动力。周期较短的代理必须学会获得这样的动力，而周期较长的代理则可以通过设计免费获得这种动力。我们在图片包含 物体

描述已自动生成基线设置中选择40ms动作周期时间，并比较较短和较长周期时间的效果。

E．动作空间:位置vs速度控制

对于现实世界的任务来说，选择动作空间可能很困难，因为物理机器人控制器的设计通常没有考虑到学习低级控制。一个与机器人近期状态有很强因果关系的控制变量是一个合适的动作选择。在模拟机器人任务中，力矩或加速度通常被选择为动作。我们感兴趣的是使用一个低层驱动命令来控制UR5，使该任务类似于Gym Reacher。UR5允许每8毫秒发送一次命令来控制位置和速度。我们可以直接使用速度控制作为动作，但直接使用位置控制是不可行的。带有直接位置控制的随机生成的初始策略生成角位置命令序列，这些命令会导致剧烈、突然的移动和紧急停止。

我们为基线设置选择了直接速度控制，并与平滑的位置控制形式进行了比较。为了避免机器人在直接速度控制下突然运动，我们只需要将角速度限制在[- 0.3，+0.3]rad/s，引轴加速度之间的角速度限制在1.4 rad/s2的最大值即可。通过位置控制，我们需要应用平滑两次，其中动作成为所需位置的二阶导数的代理;应用平滑一次无法避免突然的运动。我们的平滑技术可以描述为:

这里t是代理时间步长，z是动作矢量，q是测量的关节位置，qdes是作为位置控制命令发送的期望关节位置，y是第一个导数变量。运算符剪切[a，b]之间的值。我们根据角度安全边界将τ设置为动作周期时间ymax = -ymin = 1和（qmin，qmax）。我们根据URScript API为位置控制命令的增益选择默认值。

URControl在移动机器人之前进一步调节位置和速度命令。图4显示了基于由具有8ms动作周期时间的随机代理收集的数据的两种动作类型和三种不同的电动机信号之间的互相关。这些电机信号是目标加速度、目标转矩和被测电流，与UR5中的其他信号相比，与后续电机事件的关系更为密切。在针对目标加速度和扭矩的两个包之后以及在测量电流的三个包之后，两个动作与电动机信号具有最高的相关性。这一观察结果促使我们选择基线周期时间大于8ms。除目标转矩外，速度控制与电机信号的相关性高于位置控制。速度控制的相关关系都集中在一个时移上，而位置控制的相关关系则在多个连续时移上存在，表明两者之间的关系更为间接。

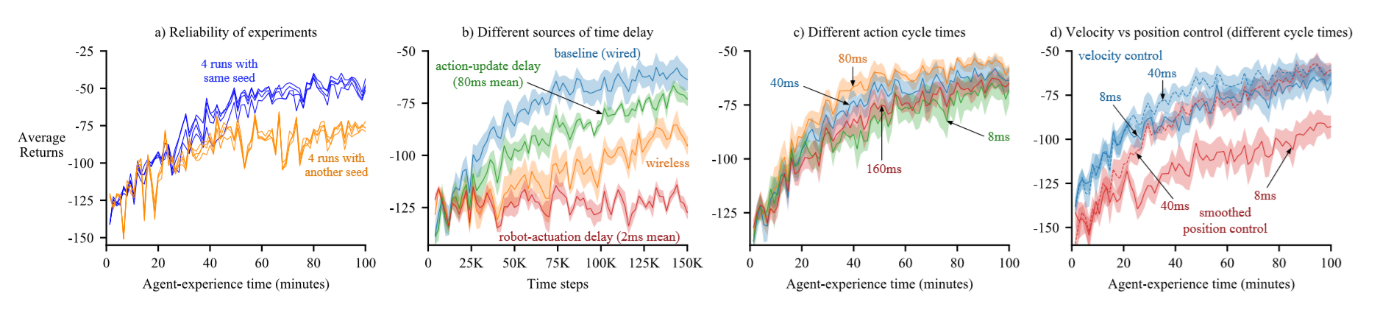


图 5 在不同的UR5 Reacher设置中学习表现。a) 我们的基线设置允许具有有效学习性能的高重复性实验，其中使用相同种子的独立运行提供了类似的学习曲线。b) 使用无线连接或使用不同的人为延迟源对不同程度的学习都是有害的。C) 与我们的基线设置(40ms)相比，使用太短或太长的操作周期时间会导致更差的学习性能。D) 在两种不同的动作周期内，速度控制明显优于平滑位置控制。

V.不同任务设置元素的影响

我们对基线任务设置进行了更改，以研究不同元素的影响。在任务设置的每个变体中，我们使用具有相同超参数的TRPO：折扣因子为0.995，批量大小为20段，步长为0.04，在五种不同的机器人健身任务中具有最佳整体性能。该策略由正态分布定义，其中均值和标准差由神经网络表示。策略网络和批评网络都使用两个隐藏层，每个层包含64个节点。对于每个实验，我们都进行了5次独立的试验，并观察了一段时间后的平均回报。

神经网络因其性能依赖于初始权值而臭名昭著。最近，Henderson等人提醒人们，从实验中得出错误的结论是多么容易。举例说明，如果使用不同的随机化种子集重复实验，相同的算法可能会获得显著不同的性能。因此，两种不同的方法或设置可能看起来明显不同，这仅仅是因为它们之间不同的伪随机数序列产生的随机机会。我

们在设置实验时格外谨慎，以确保在每个任务设置变量中使用相同的五组初始网络和五组目标位置序列。

为了验证我们实验设置的正确性，我们对两个不同的种子重复基线实验4次，如图5(a)所示。每次试验包含150,000个时间步长或100分钟的代理体验时间，以及包括重置在内的约三个小时的实时总时间。随着时间的推移，每个学习曲线显着改善，并且代理获得更高的平均回报，从而产生有效且一致的到达行为，如伴随视频中所示。值得注意的是，对于相同的种子，所有的学习曲线都非常相似，即使它们是通过在不同的日子、不同的物理UR5单元上运行每个试验多个小时而生成的。这证明了UR5的精度，TRPO的稳定性，以及我们实验和任务设置的可靠性。

图5(b)显示了使用无线连接的影响。实线表示平均收益，阴影区域表示标准误差。与有线连接的基线设置相比，无线连接导致了性能的显著下降，这可以归因于传感器运动包到达计算机和URControl的驱动命令的变化和延迟。为了研究它们的影响，我们分别在动作更新和向URControl发送驱动命令时，注入了人工指数随机延迟，粗略地模拟了Wi-Fi传输延迟。动作更新延迟实际上类似于观察延迟，也可能由无效的实现引起。这两种延迟都增加了动作如何影响后续观察的不确定性，从而使学习问题变得困难，而延迟机器人的动作还可能影响机器人的操作。图5(b)显示，平均80ms的随机动作更新延迟导致学习性能显著下降。另一方面，增加一个平均2毫秒的随机机器人驱动延迟会彻底破坏学习。

在图5(c)中，我们展示了选择不同操作周期时间的影响。当循环时间减少到8ms时，性能恶化。另一方面，当增加到80ms时性能有所提高，但当增加到160ms时性能明显下降。在图5(d)中，我们展示了在8ms和40ms两种不同的动作周期中，直接速度控制和平滑位置控制的学习性能。在这两种情况下，平滑位置控制的效果都明显低于速度控制。

最后，我们研究了我们的基线设置是否仍然有效的六关节控制应用于UR5 Reacher 6D。为了适应问题的复

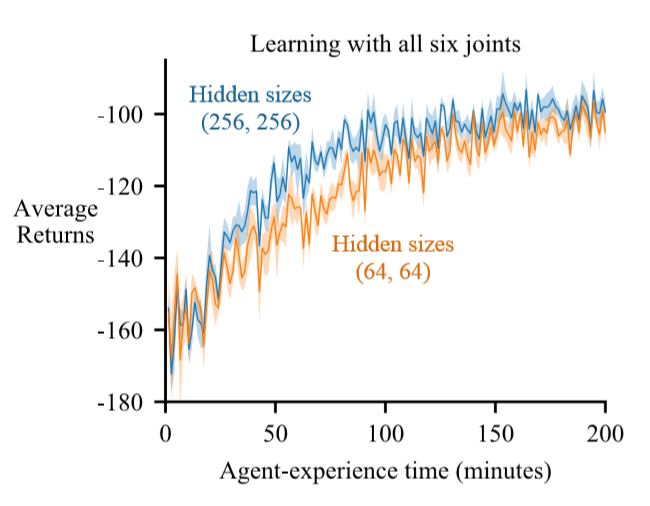


图 6 在UR5 Reacher 6D中学习，所有六个关节都被激活。

杂性，我们探索了具有更大隐层的策略和批评网络。图6显示了200分钟内显著的学习进度，随着时间的推移，学习进度还在不断提高。这段视频展示了跑步更久后的有效行为。

VI．讨论和结论

在这项工作中，我们与UR5机器人一起设计和开发了一个学习任务，以讨论实际任务设置的关键要素。我们的讨论总结在以下假设中，用于实际的机器人学习任务，我们将其作为基线设置的指导：

1. 在不同计算阶段发生的系统延迟通常对学习有害。因此，有线通信优于无线通信。
2. 动作周期时间太短会使学习更加困难。太长的动作周期时间也会影响性能，因为它们会降低精度并导致数据收集速度变慢。
3. 通过与未来观察的更直接的关系，更直接地选择行动应用的动作空间，以便更直接地应用动作。
4. 由于减少了延迟，一些并行计算优于常规模拟任务的顺序计算。

我们通过创建基线设置的变量来研究前三个假设的有效性，我们的研究在很大程度上支持了它们。我们证明了学习性能对某些设置元素非常敏感，特别是系统延迟和操作空间的选择。相比之下，性能对不同的动作周期时间不那么敏感。我们的结果表明，减轻任何来源的延迟可能是有益的，这表明了最后一个假设的前景。我们的研究只是向全面了解实际学习任务迈出了一小步，这需要使用不同的任务，学习方法和超参数集进行更彻底的调查和验证。我们的基线设置允许我们使用现成的RL方法进行高度可靠和可重复的实际实验。尽管围绕机器人的障碍和挫折以及深度RL研究的可重复性，这仍然是对广泛的RL实验和现实世界机器人研究的可行性的有力证明。