SSUP model implementation

A. 采样实现

1、在场景内 sample 一个能动的物体，$object \sim Multinomial(\left\{\frac{1}{n_{obj}}||i \in [0, \cdots, n_{obj}]\right\})$

就是等概率随机选一个

2、sample 相对于该物体的 y 轴位置，分布为平均值为 $object_y$ 标准差为 σ_y 的高斯分布，并且被场景的合法最低位置和最高位置截断

3、sample 相对于该物体的 x 轴位置：

计算该物体包围盒的左右边缘 BB_{left} 和 BB_{right}

在 $BB_{left} - \sigma_x$ 到 $BB_{right} + \sigma_x$ 区间均匀 sample 一个值 v

若 $v < BB_{left}$ 或 $v > BB_{right}$，则以包围盒边缘为中心、σ_x 为标准差的正态分布中 sample x 值

否则直接令 $x = v$

Algorithm 1 SSUP model for the Virtual Tools game

```
Sample n_init points from prior π(s) for each tool
Simulate actions to get noisy rewards r̂ using internal model
Initialize policy parameters θ using policy gradient on initial points
while not successful do
    Set acting = False
    With probability ϵ, sample action a from prior
    With probability 1 − ϵ, sample action a from policy
    Estimate noisy reward r̂ from internal model on action a
    if r > T then
        Set acting = True
        Try action a in environment
    else if i ≥ n_iters then
        Set acting = True
        Try best action a* simulated so far which has not yet been tried
    if acting then
        Observe r from environment on action a.
        If successful, exit.
        Simulate r̂ assuming other two tool choices.
        Update policy based on all three estimates and actions.
    else
        Update policy using policy gradient
```

σ_x 和 σ_y 是自由参数，其选择反映了在 y 方向上相对均匀的先验分布，而在 x 方向上则是一个更紧密的分布。这些决策通过图 S1 所示的敏感性分析进行了检验。

模型首先根据此先验为每个工具采样一定数量的初始点 $n_{initial}$，并通过噪声模拟器运行每个动作。策略使用这些带噪声的奖励估计值进行初始化。不以此方式初始化策略会产生不利影响，这一点可以在敏感性分析中看到

B. Implementing simulation

在物理模拟过程中加入噪声模拟人类对碰撞过程的模拟，based on Chipmunk engine

$\phi' \sim warppedNormal(\phi, \sigma_\phi)$，调整碰撞力的方向，环绕正态分布

$e' \sim \mathcal{N}(e, \sigma_e)\ s.t.\ e' > 0$，调整碰撞的弹力大小

对每个 imagined action 做 n_{sims} 次模拟（设定为 4），然后求出 ave reward，用这个 reward 更新 action-outcome 的 policy θ，并且决定是否执行这个 action

C. Implementing updating

使用 sampled actions 得到的 reward 更新 policy π，π 首先选择使用什么 tool，然后对每个 tool，把它放在场景内的某个位置

使用高斯分布的混合模型，每个 tool 对应一个高斯分布，有两个参数对应 tool 的权重（sum 为 1 可以省一个），4 个参数对应高斯分布的 position 和 variance，加起来 2 + 4 * 3 = 12 个参数。

$$\theta \leftarrow \theta + \alpha r \nabla_\theta \log \pi_\theta(a)$$

a 是使用的 action，α 是学习率，r 是 reward

该策略的初始化方式是：每个高斯分布均以屏幕中心为均值，并具有 50 像素的各向同性方差（即高度和宽度的 1/12）。在采取任何实际动作之前，策略会使用从先验分布中采样的 `n_initial` 个动作所得到的带噪声的奖励估计值进行更新。我们发现，这种方法显著提高了稳定性，并且其性能远优于那些相对于场景中物体进行参数化的策略。

使用 epsilon greedy strategy，epsilon 是一个 free parameter，根据 A 节中的过程 sample action

SSUP会持续采样动作，直到它找到一个奖励超过给定阈值T的动作，或者达到最大模拟次数 `n_iters`。如果找到了高奖励的动作，它就在环境中执行该动作。否则，如果达到了内部模拟的最大次数 `n_iters`，则选择它到目前为止模拟过但未在现实世界中尝试过的最佳动作。如果动作成功，模型任务完成；否则，模型会继续恢复模拟。

1、counterfactual updates

在 take action in the env 之后，同时询问 noisy simulator 如果在同一个位置采用另两个 tools 会怎么样，然后用三个 reward 同时更新 policy。可以确定 reward 和 tool 有关还是和 position 有关，同时能稳定 policy gradient

2、reward function definition

直接使用轨迹到目标的距离最小值的归一化值，相对于完全不采用 action 的距离

$$r = 1 - \frac{\min_{t=0,N,obj \in objects} d(obj, goal, action)}{\min_{t=0,N,obj \in objects} d(obj, goal)}$$

优于直接使用最小距离的非归一化的 reward

D、Running on the Virtual Tools game

设定 SSUP 最多只能在 env 中执行十次操作，是人类没解决问题的操作次数的中位数，若十次没找到解则认定为未通过

Physical parameter tuning model details

作为一种替代性学习方案，我们假设人们利用失败动作的观察结果来优化其针对该游戏的内部动力学模型，然后使用这些更新后的模型来选择下一个动作。关键在于，这种替代性学习方案**并不**从一个根据观察更新的策略中采样动作。它总是从以物体为中心的先验分布中采样动作，只有噪声动力学模型能够基于观察结果进行改进和优化。

我们通过一个调整其物理引擎内部与物体动力学相关参数的模型来实现这一学习方案：这些参数包括物体密度 d、摩擦力 f 和弹性 e，我们统称为 Ψ。该模型依赖于与完整 SSUP 框架相同的物理引擎，包括碰撞过程中引入的不确定性（见 B 节）。

在执行每个失败动作后，参数调整模型尝试使用贝叶斯推断来更新其内部参数，以匹配观察到的该动作结果；这是 "analysis-by-synthesis" 的一个实例。模型观察一个失败轨迹的 20 秒，并在时间上均匀分布的 50 个点上采样所有可移动物体的位置。为了近似给定参数化 Φ 下每个观测值的似然，模型对同一动作进行 20 次模拟，并记录在相同时间点上模拟中所有可移动物体的位置。根据这些记录，模型获取每个物体在每个时间点的位置，并参数化一个高斯分布来描述该物体在该时间点应处位置的可能性 $(\mu^t_{obj}, \sigma^t_{obj})$。因此，观察到完整轨迹的似然是每个时间点上各个观测到的物体位置的似然的乘积：

$$P(O|\Phi) = \prod_{obj} \prod_t P(o^t_{obj}|\mu^t_{obj}, \sigma^t_{obj})$$

我们将每个属性的先验参数化为一个高斯分布，其均值以该属性的真实值为中心（密度为1，弹性和摩擦力为0.5），方差为0.025。我们的 proposal function 定义为一组截断高斯分布，其方差为0.02（均值等于当前估计值）。

我们使用上述 likelihood func 和 proposal func，通过Metropolis-Hastings算法进行20次迭代来实现这一推断，使用5条 chains 和5个 burn-in samples。我们发现这足以对每个参数给出可靠的估计。该过程是在概率编程语言Gen中实现的。

每次观察到新的动作时，我们使用先前观察确定的参数来初始化MH过程。通过这种方式，学习（以更精细的先验形式）可以在一个关卡内的一系列动作中持续进行。

Parameter sensitivity analysis

为了确保参数的选择不会不当影响模型性能，我们测试了SSUP模型在其每个参数的不同设置下的表现。我们使用一个称为"累计解曲线间总面积"（Area Between Cumulative Solutions）的指标来衡量模型性能。该指标在每个关卡上的定义为：人类和模型的累计解曲线之间的面积（参见正文中的图6A），并归一化从 0（完美匹配）到1（参与者或模型总是立即解决关卡，而另一方从未解决关卡）之间。该指标将逐次试验的准确率、使用的动作次数以及解决率的变化结合为一个单一指标。

由于SSUP模型包含10个参数，我们无法合理地测试在所有参数选择的全网格上的模型性能；取而代之，我们每次只测试单个维度上的模型性能，即改变一个参数而保持所有其他参数不变。如图S1所示，在SSUP模型所用参数值周围的一个宽范围内，模型性能没有显著差异。这表明，进行更精确但计算上难以实现的参数拟合，最多也只能带来模型拟合度的边际改进。

作为这些参数具有泛化能力的进一步证据，我们在10个验证关卡上使用了完全相同的模型参数化方案，并发现模型同样与人类数据有良好的拟合。

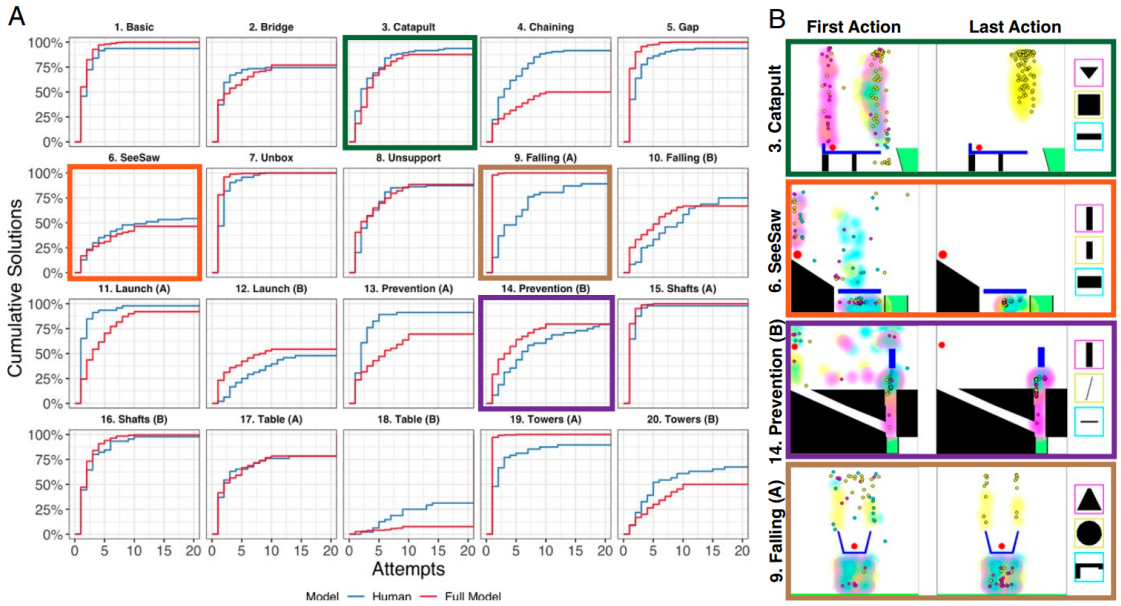

Fig. 6. (*A*) Cumulative solution rate over number of placements for participants vs. the SSUP model. (*B*) Distribution of model actions (background) versus human actions (points) on the first and last attempts of the level for a selection of four levels. The distribution of model actions is estimated based on fitting a kernel density estimate to the actions taken by the model across 250 simulations. Colors indicate the tool used, with the tools and associated colors shown at *Right* of each level. In most levels, the SSUP model captures the evolution of participants' solutions well, including the particular actions chosen; in the few cases that it differs, there is no alternative model that systematically explains these differences.

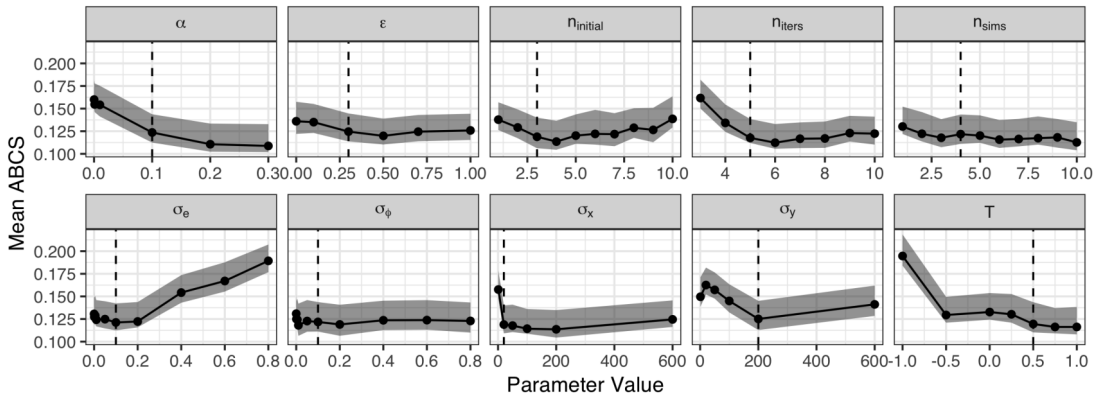

Fig. S1. Mean Area Between Cumulative Solutions (ABCS) values across all trials for different parameter values, keeping all other parameters at the default values. The default values are denoted by the dashed lines. Grey areas indicate 95% bootstrapped confidence intervals. Legend: α: learning rate for policy gradient, ϵ: the probability of sampling an exploratory action from the prior, $n_{initial}$: number of initial samples from the prior used to initialize the policy, n_{iters}: number of internal iterations before action must be taken, n_{sims}: the number of simulations run for each imagined action, σ_e: the noise (in radians) for collision elasticity, σ_ϕ: the noise (in radians) for collision direction, σ_x: the standard deviation of the prior in the x direction, σ_y: the standard deviation of the prior in the y direction, T: -1$\times$ the reward threshold for acting

图 S1. 不同参数值下所有试验的平均累计解曲线间总面积值，所有其他参数保持默认值。虚线表示默认参数值。灰色区域表示 95% 的自举置信区间。

图例说明：

- **α**: 策略梯度的学习率
- **ε**: 从先验分布中采样探索性动作的概率
- **n_initial**: 用于初始化策略时从先验分布中采样的初始样本数量
- **n_iters**: 必须采取动作前的内部迭代次数
- **n_sims**: 为每个想象动作运行的模拟次数
- **σ_e**: 碰撞弹性噪声（单位为弧度）
- **σ_φ**: 碰撞方向噪声（单位为弧度）
- **σ_x**: 先验分布在 x 方向的标准差
- **σ_y**: 先验分布在 y 方向的标准差
- **T**: 采取动作的奖励阈值（值为 -1 × 所示阈值）

A Deep Reinforcement Learning baseline

我们研究了深度强化学习中的一种流行方法——Deep Q Networks（DQN）——是否能够仅从像素输入解决 VTG。 虽然我们预期此类方法需要大量经验才能学习到有用的表征，但我们希望它可能发现一些通用的先验（例如以物体为导向，甚至理解某物应放置在另一物体的上方还是下方），这些先验或许能够迁移到不同类型的关卡中。

A. Random Level generation

为了在第一个实验中比较人类和模型的性能，我们使用了20个手工设计的关卡。然而，深度Q学习需要大量数据进行训练，尽管我们可以通过在给定关卡上执行大量动作来进行训练，但这可能导致对这20个关卡的特定细节产生过拟合。

取而代之，我们基于五个手工设计关卡，从一组五个模板中随机生成关卡。这些模板的设计使得它们包含相同的物体集，并且可以用类似的方式解决，但物体的大小和配置可以变化，这强制引入了解法动作的某种可变性。生成关卡的具体算法因模板而异，但涉及在满足某些几何约束的前提下调整许多物体的大小或位置（例如，每个"桌面"关卡中的'桌面'始终位于斜坡和目标之间，每个弹射器关卡中的球始终放置在弹射器物体上）。各模板的示例场景见图S2A。

每个生成的关卡包含从用于构建20个手工设计关卡工具的可能形状池中随机抽取的三个工具。这些工具可以进一步随机调整大小或以90度角旋转，但受限于它们必须能放入原始工具所在的 90×90 像素区域内。随机生成工具的示例见图S2B。

为了保证每个关卡都有一个合理但非平凡的解法，我们为每个模板提出了一个"解法区域"，该区域与基础关卡的主要解法区域面积相似。例如，basic 模板的解法区域是一个位于关键球上方的矩形区域，而 shafts 模板的解区域则包含直接位于竖井上方的两个区域。接着，我们从该区域随机生成 100 个工具放置位置，并仅筛选出其中能解决关卡的放置位置占比在 3% 到 80% 之间的试验。

我们使用这些模板各自生成了 1,000 个关卡。随后将这些关卡进行划分，为每个模板提供 900 个训练关卡和 100 个验证关卡。

B. Architecture details