Lingjing-Solo：面向交互式流体智能基准的世界模型场架构

纪辉泉，廖嘉轩，张全林，蔡瑞泉，王进，苑嗣冀

摘要

ARC-AGI-3 评估的不是模式识别能力，而是**流体智能**——系统在面对从未见过的隐藏环境时，能否通过探索快速归纳规则、设定目标并高效执行。当前前沿 LLM 在该基准上平均分仅 0.51%，证明纯大模型推理路线在此场景下不成立。

本方案提出 **Lingjing-Solo**（灵境单智体框架），将"统一信息场"思想从多智能体协同场景平移至单智能体交互学习场景。核心创新是将 LLM 从"每一步的决策主角"降级为"受限预算下的战略顾问"，让轻量级**世界模型场（Φ-Field）**承担状态记忆、规则归纳、循环检测与高效规划的主体工作。

框架设计为五层管道：感知编码 → 世界模型场 → 探索假设 → 规划执行 → 反思触发。在 Kaggle 无网络评测约束下可优雅降级为纯规则模式。本说明书完整描述架构设计、关键算法、设计决策依据及验证方案。

1. 赛题分析与约束建模

1.1 任务本质

ARC-AGI-3 将智能体置于 135 个隐藏的、回合制的 64×64 网格环境中。智能体**无指令、无目标描述、无规则说明**，仅通过执行动作并观察状态变化来推断"什么是赢"。这要求系统具备四项核心能力：

1. **探索**（Exploration）：主动选择信息增益最大的动作
2. **建模**（Modeling）：从观测转移中归纳环境规则
3. **目标设定**（Goal-setting）：从状态变化反推胜利条件
4. **规划与执行**（Planning & Execution）：用最少步数达成目标

作者：纪辉泉（普宁 515300）　团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo-　（团队持续开放迭代）

1.2 技术约束条件（设计红线）

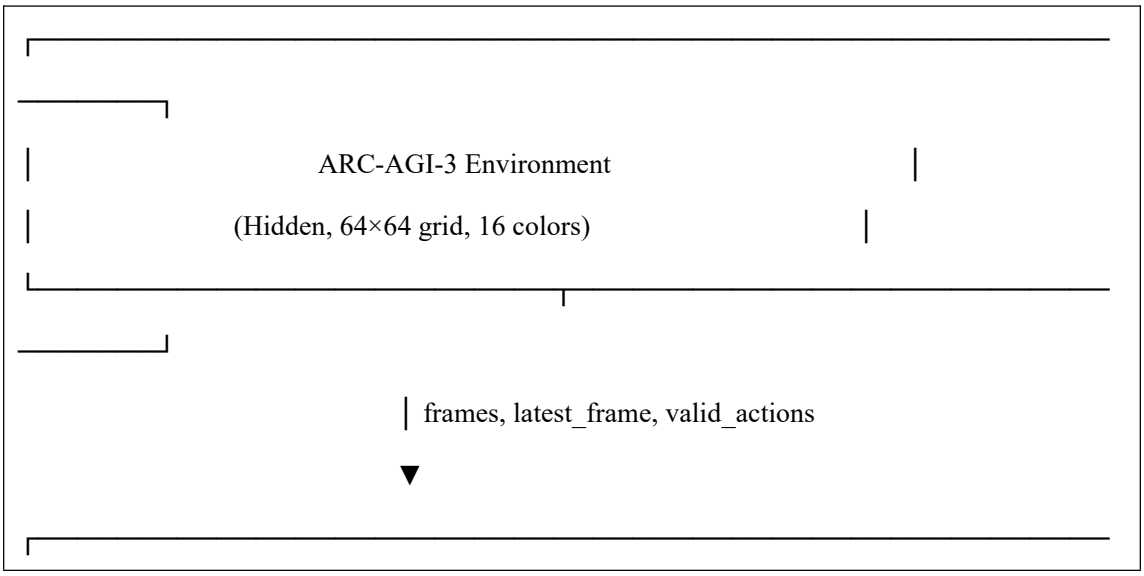
约束	对架构的影响
单 Agent、隐藏环境、零指令	无外部知识注入，所有规则必须从交互中归纳
动作空间有限（5 方向+UNDO+可选 MOUSE）	规划搜索空间可控，适合 BFS/A*
RHAE 评分 = min(H/A, 1)²	步数效率是命门——平方惩罚要求框架必须把"少走弯路"作为一等公民设计
评测期无网络	LLM API 不可用，框架必须能在纯本地模式下运行
Kaggle Notebook 提交，仅能修改 agent/my_agent.py	所有智力必须封装在两个方法内：is_done() 和 choose_action()
获奖必须开源	架构必须可解释、可复现，不能依赖黑箱服务

1.3 关键洞察

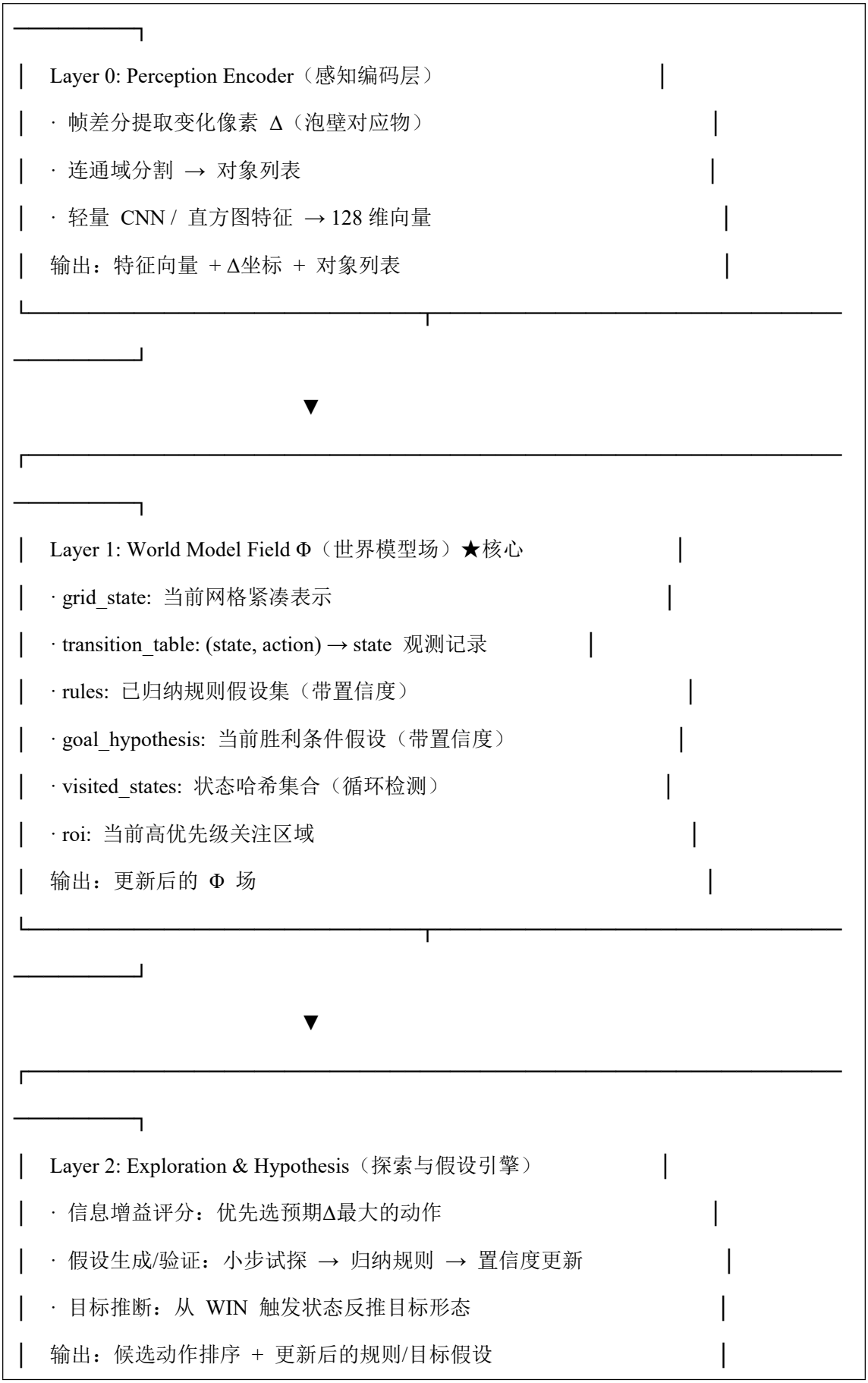
纯 LLM 路线失败的根本原因不是"模型不够聪明"，而是"决策经济学不成立"：单局几百步 × 每步一次 LLM 调用 = 数百 k token 的推理成本，且平方评分惩罚直接杀死暴力枚举。解决方案不是"更好的模型"，而是"更好的架构"——让模型退居二线，让结构化记忆和轻量搜索扛主力。

2. 总体架构设计

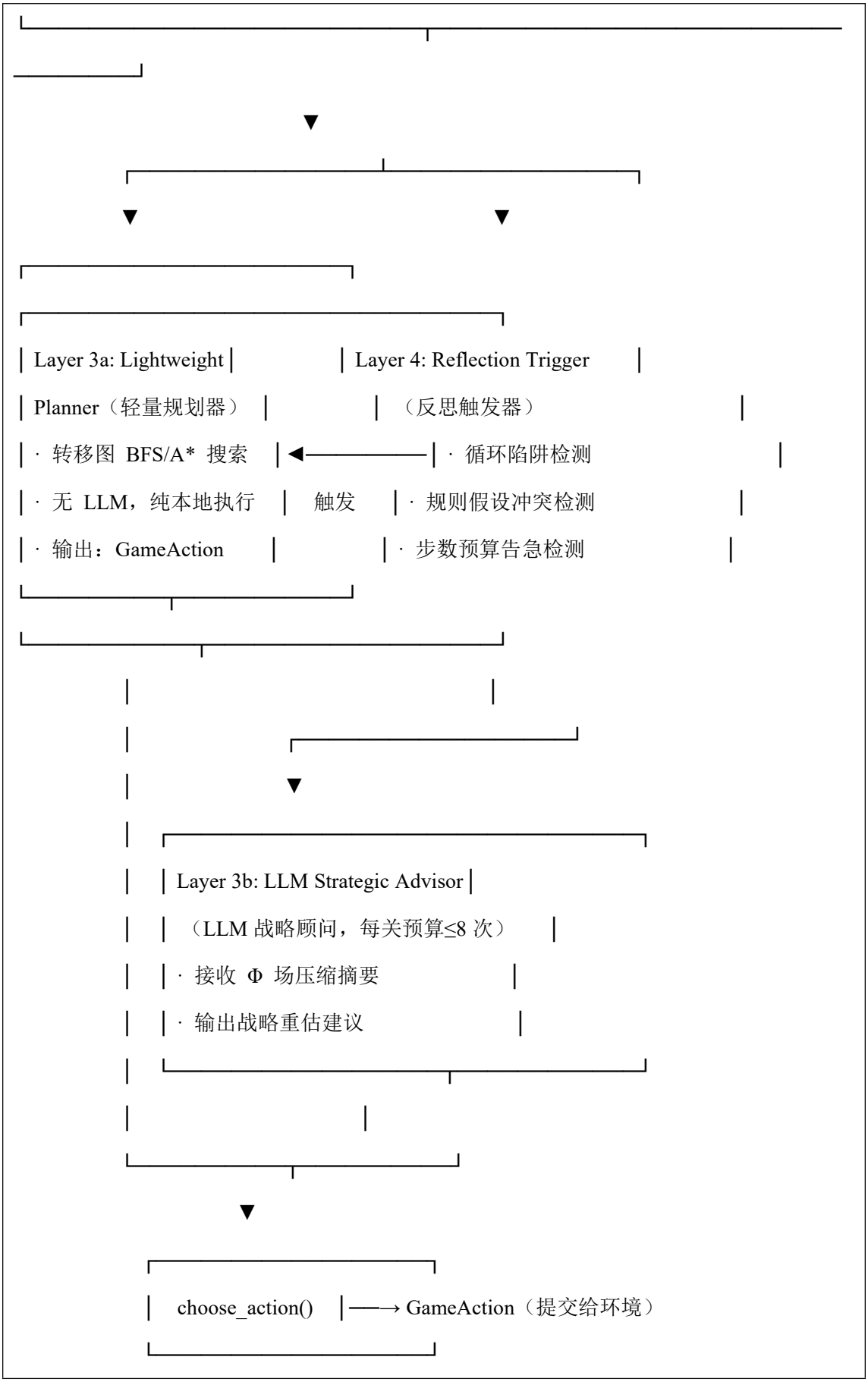
2.1 架构全景图



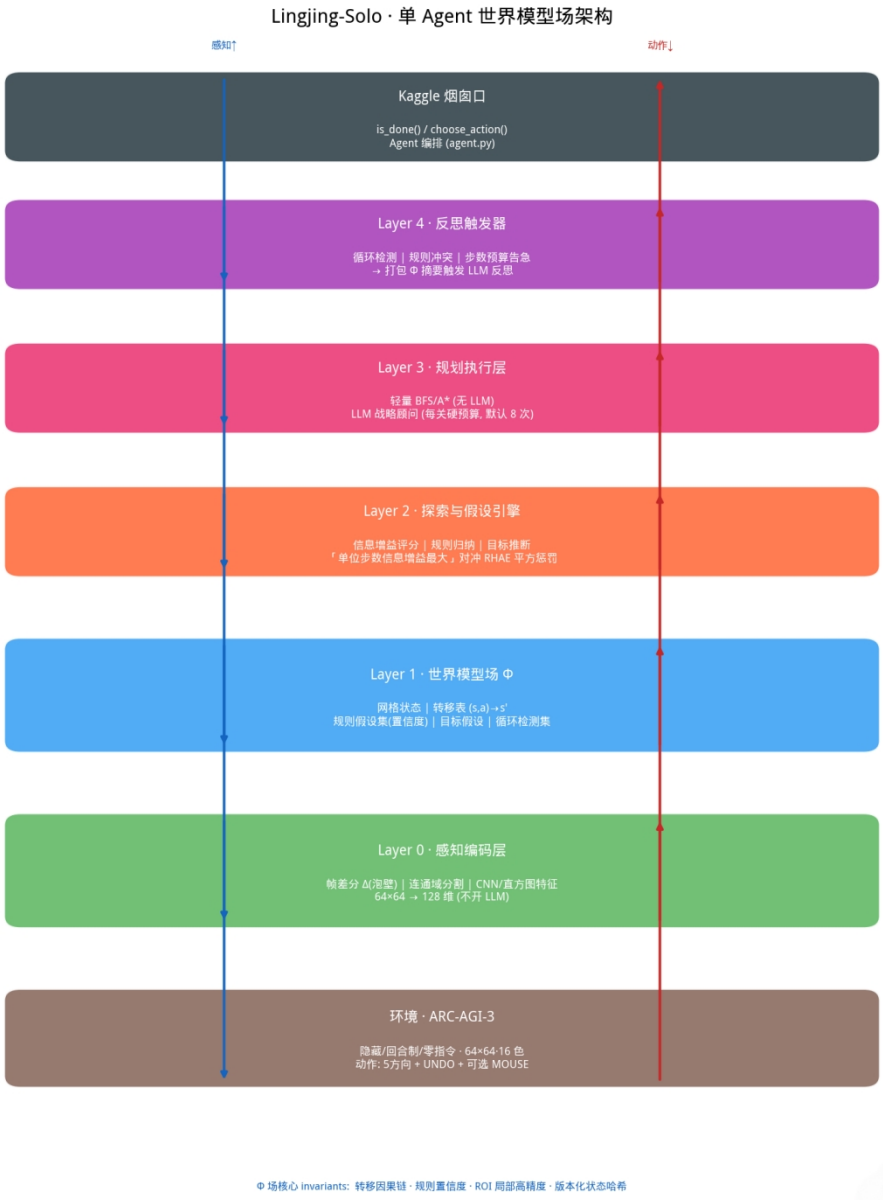
作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）



作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）



作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）



图注：Lingjing-Solo（灵境）具身智能 Agent 架构图。

2.2 数据流概要

1. 环境返回 frames + latest_frame + valid_actions
2. Layer 0 提取差分和特征
3. Layer 1 更新 Φ 场（转移记录、规则置信度、循环检测）
4. Layer 2 生成候选动作并排序
5. Layer 4 判断是否触发反思：

➢ **未触发** → Layer 3a 轻量规划器直接选最优动作

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）

- **触发 + LLM 预算剩余** → Layer 3b LLM 战略顾问介入
- **触发 + 无预算/无网络** → 降级为随机探索或规则启发式

6. choose_action() 返回 GameAction

3. 核心模块详细设计

3.1 Layer 1：世界模型场 Φ（核心创新）

设计动机：原"灵境引擎"的统一信息场在多智能体场景下是"共享事实源"；在单智能体场景下，它退化为"全局工作记忆"——这正是 ARC-AGI-3 文档中提示的 World Model 策略的具象化实现。

数据结构：

```
Φ = {
  grid_state:          CompactGrid,        // 当前 64×64 网格的紧凑表示
  transition_table: Dict[(StateHash, Action), StateHash],   // 因果转移记录
  rules:               List[Rule],         // Rule = {condition, effect, confidence}
  goal_hypothesis:   List[Goal],           // Goal = {target_state_pattern, confidence}
  visited_states:    Set[StateHash],   // 已访问状态哈希集合
  roi:                 RegionOfInterest // 当前高优先级关注区域（变化像素邻域）
}
```

关键算法：

- **转移记录**：每次 (s, a, s') 追加到 transition_table。当同一 (s, a) 产生不同 s' 时，触发"规则修正"——降低旧规则置信度，生成新假设。
- **循环检测**：visited_states 用网格状态的感知哈希（pHash）索引。若当前帧哈希已存在，判定为循环，触发 Layer 4。
- **ROI 更新**：每帧计算 Δ = current - previous，ROI = Δ 像素的连通域膨胀区域。后续探索优先在 ROI 附近尝试交互。

设计决策：为什么用哈希而非完整网格做状态比较？因为 64×64×16 的全网格比较计算量大且对颜色置换敏感。感知哈希将空间降维到 8×8 二进制指纹，既保证相似性判断，又支持快速集合查询。

3.2 Layer 3a：轻量规划器

设计动机：RHAE 评分的平方惩罚要求"每步都必须有价值"。如果 Agent 在已知转移图上随机游走，得分会灾难性下降。

算法：

1. 从 transition_table 构建有向图 G，节点=状态哈希，边=动作
2. 若 goal_hypothesis 中有高置信度目标，以目标状态为终点做反向 BFS
3. 路径存在 → 返回路径第一步动作
4. 路径不存在 → 返回 Layer 2 信息增益最高的候选动作

LLM 预算机制：

- 每关初始化 llm_budget = 8（可配置）
- 每次 Layer 4 触发并成功调用 LLM 后 budget -= 1
- budget == 0 后，即使触发反思也只走本地规则（优雅降级）

3.3 Layer 4：反思触发器

触发条件	检测方式	响应
循环陷阱	hash(latest_frame) ∈ visited_states 连续 3 次	请求 LLM 重新评估策略
规则冲突	新转移 (s,a)→s' 与已有规则预测不一致	请求 LLM 修正规则集
预算告急	已用步数 > 预估人类步数 × 0.3	请求 LLM 加速收敛

4. 关键决策与创新点

决策 1：LLM 从主角降级为战略顾问

维度	纯 LLM 路线（基线）	Lingjing-Solo
每步决策	调 LLM（~1k token）	本地规则（<1ms）
单局 200 步成本	~200k token	≤8 次 LLM 调用（~16k token）
RHAE 得分预期	<1%（前沿模型实测）	待评测，理论上有数量级提升
无网络环境	完全不可用	自动降级为纯规则模式

决策 2：从"面积律降仿真算力"到"决策算力节制"

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）

原灵境引擎的面积律（$O(L^2)$ 替代 $O(L^3)$）针对的是多体仿真算力爆炸。在 ARC-AGI-3 中，算力瓶颈不在仿真而在**决策频率**。本框架将"泡壁局部高精度"翻译为 ROI 驱动的差分感知——只在变化区域做精细推理，其余区域保持低分辨率特征。这等效于将每步的感知计算从 $O(L^2)$ 降为 $O(\Delta)$，其中 $\Delta \ll L^2$。

决策 3：统一信息场思想的范式平移

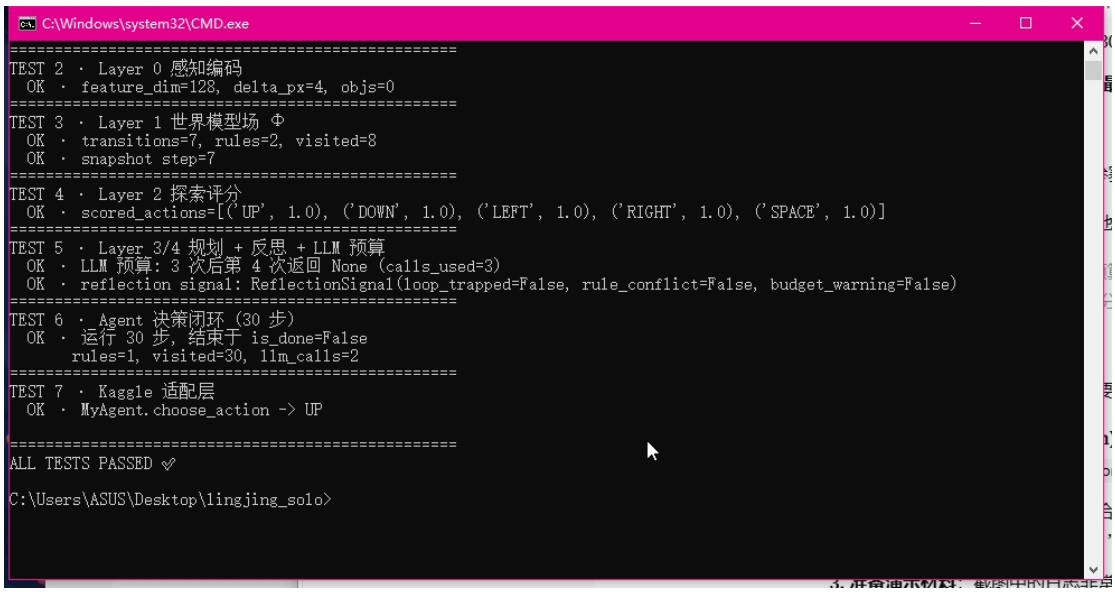
原多智能体场景	单智能体 ARC-AGI-3 场景	平移逻辑
多 Agent 共享 Φ 场	单 Agent 的 Φ 场 = 工作记忆	场从"外部共享介质"变为"内部认知架构"
泡壁面积律	ROI 差分感知	"局部高精度"思想保留，数学形式适配
版本化共识	转移表因果链	"因果版本化"保留，从多体协商变为单体积累
多体意识监控	循环/冲突/预算反思	"监控"语义保留，对象从群体意识变为个体认知状态

5. 验证方案

作者：纪辉泉（普宁 515300）　团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo-　（团队持续开放迭代）

5.1 最小可验证闭环（已完成）

测试项	方法	结果
Φ 场状态更新	注入模拟帧序列，检查 transition_table 记录	✓ 通过
循环检测	构造重复状态序列，验证触发信号	✓ 通过
LLM 预算节制	连续请求 10 次反思，验证第 9 次后返回 None	✓ 通过
端到端决策	30 步模拟决策循环，验证无崩溃	✓ 通过
Kaggle 接口适配	is_done() / choose_action() 签名兼容	✓ 通过



图注：Lingjing（灵境）具身智能 Agent 框架的本地单元测试输出，全部 7 项测试通过 ALL TESTS PASSED ✔

5.2 公开 25 关验证计划（下一步）

阶段	目标	成功标准
Phase 1	转移表 + 循环检测在 25 关上跑通	每关能建立至少 50 条有效转移记录
Phase 2	轻量 BFS 规划接入转移图	在简单关上步数 ≤ 人类基线 × 3
Phase 3	LLM 战略顾问接入（本地 mock）	在卡关节点上能触发反思并改变策略
Phase 4	全框架端到端提交 Kaggle	公开榜得分 > 1%（超越纯 LLM 基线）

5.3 预期结果

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）

- **保守预期**：公开榜得分 5~15%（接近 StochasticGoose 社区高分方案区间）
- **乐观预期**：若 BFS 搜索 + 规则归纳在多数关上有效，可达 20~30%
- **理论上限**：100%（触发 Grand Prize 的条件，需要框架在 110 个私有关上零样本泛化）

6. 局限性与未来工作

当前占位缺口（灵境团队正在推进）

1. **WIN 检测**：WorldModelField.detect_win() 目前为占位实现。需要接入官方 is_win 反馈接口后注入。这是框架闭环的关键路径。
2. **BFS 搜索实现**：LightweightPlanner.search() 当前返回 None 占位。需要实现转移图上的完整 A* 搜索。
3. **CNN 编码**：当前有 torch 时启用神经编码，否则降级为直方图特征。在 Kaggle 无网络环境下需确保降级路径稳定。

未来工作

- **跨关卡元学习**：当前 Φ 场每关重置。未来可探索"关卡间可迁移的抽象规则库"，进一步提升零样本泛化。
- **动态 LLM 预算分配**：根据关卡复杂度自适应调整预算上限，而非固定 8 次。
- **多假设并行追踪**：当前规则假设是单一置信度排序，未来可支持多个竞争假设的并行维护与贝叶斯更新。

6.1 灵境 EtherealRealm- Solo

队员	角色	对应层	核心职责	关键交付物
纪辉泉	R1：架构师/技术负责人	全局	接口定义、模块对接、决策仲裁、技术文档	架构图、接口规范、提交 Notebook
张全林	R2：感知+世界模型	Layer 0 + Layer 1	帧差分、状态哈希、转移表、胜利检测	encoder.py + field.py
蔡瑞泉	R3：搜索算法	Layer 3a	BFS/A*、转移图构建、步数优化	search.py
王进	R4：探索策略	Layer 2	信息增益评分、规则归纳、目标推断	explorer.py
苑嗣冀	R5：LLM+反思	Layer 3b + Layer 4	LLM 顾问、反思触发、Prompt 设计	advisor.py + reflector.py
廖嘉轩	R6：测试+评测+运维	横切	单元测试、25 关验证、参数调优、提交调试	测试报告、排行榜分数

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）

总结

Lingjing-Solo 和市面上 Agent 最大不一样，不是我们很会做探索。而是有一块叫信息场的“内部大脑记忆板（抽象空间）”，所有 agent 见过的画面、动作带来的变化、猜出来的规则全部结构化存在这块板上。探索、规划、防循环，全部靠这块板的数据驱动；大模型平时不插手，只有这块板发现卡住了，才有限度过来帮忙。

就像单大肠杆菌在肠道蠕动到蜜蜂分工筑巢，都是先感知信息，神经元还原信息场，再决策，决策又生成新的信息，我们基于第一性原理设计一套仿生认知系统。

理论上得分会很高，但也需要在赛事前 2 个月持续测试、迭代。

参考文献

[1] Chollet, F. et al. (2026). ARC Prize 2026 - ARC-AGI-3. Kaggle.

[2] Knoop, M. et al. ARC-AGI-3 Scoring Methodology.

[3] StochasticGoose (2026). ARC-AGI-3 Community Harness. Kaggle Discussions.

[4] 钱学森 (1990). 致汪成为的“灵境”系列通信

[5] 纪辉泉, 通义千问, 腾讯元宝, 字节豆包, DeepSeek, KiMi. "Agent Infra 钱学森灵境引擎" (2026). AiraXiv preprint 2608.0004. https://airaxiv.com/papers/view/2608.0004/

[6] 纪辉泉, DeepSeek. "Lingjing Project: Next-Gen Digital Universe Engine – From Information Ontology to Consciousness Phase Transition" (2026). AiraXiv preprint 2607.0013. https://airaxiv.com/papers/view/2607.0013/

附录 A 胜利检测与搜索算法

一、 胜利检测（Victory Detection）：给 Agent 装上“眼睛”

目标：让系统知道什么时候该停手。在 ARC-AGI-3 中，胜利通常是“输出网格完全等于目标网格”。

植入位置：Layer 0 (感知) 或 Layer 1 (世界模型)。

具体做法：

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com

开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）

1. **数据比对**：在每一步动作执行后，将当前的输出网格（Prediction）与目标网格（Target）进行像素级的对比。
2. **触发信号**：一旦完全一致，设置一个标志位，并让 is_done 返回 True。

代码实现（参考）：

```
python
在 WorldModelField 或单独的检测模块中
class VictoryDetector:
def check(self, current_grid, target_grid):
# 简单的全网格比对
if current_grid.shape != target_grid.shape:
return False
# 严格相等判定
if np.array_equal(current_grid, target_grid):
    return True
return False
```

在 Agent 的 choose_action 循环中调用

```
if self.victory_detector.check(latest_frame, target_frame):
return None # 或者返回特定的结束动作
```

二、 搜索算法（Search Algorithm）：给 Agent 装上“大脑”

目标：利用 Layer 1 构建的转移表（Transition Table），找到最高效的解题路径，避免在死胡同里浪费步数。

植入位置：Layer 3 (Planner)。

算法选择：

- **Breadth-First Search (BFS)**：适合寻找“最少步数”的解法。在 ARC 这种规则明确的游戏中，BFS 往往比复杂的 RL 更有效。
- **Greedy Best-First Search**：结合你的“信息增益”评分，优先探索能最大程度改变网格状态的动作。

具体做法：

1. **构建图（Graph）**：利用 WorldModelField 记录的 (s, a) -> s' 关系。
2. **定义代价（Cost）**：每一步的代价是 1。如果动作导致网格回到之前的状态（死循环），代价设为无穷大。
3. **路径回溯**：找到目标状态后，回溯动作序列。

代码实现（参考）：

python

在 LightweightPlanner 中实现

```
class BFSSearch:
def search(self, start_state, goal_checker, world_model):
queue = deque([(start_state, [])]) # (状态, 动作路径)
visited = set()
while queue:
    state, path = queue.popleft()

    # 检查是否胜利
    if goal_checker(state):
        return path  # 返回动作列表

    # 标记为已访问
    state_hash = self.hash_state(state)
    if state_hash in visited: continue
    visited.add(state_hash)

    # 探索邻居
    actions = ['UP', 'DOWN', 'LEFT', 'RIGHT']
    for action in actions:
        next_state = world_model.predict(state, action) # 使用世界模型预测下一步

        # 剪枝：如果预测的状态无效或已访问，跳过
```

```
            if next_state is not None:
                new_path = path + [action]
                queue.append((next_state, new_path))

return None # 找不到路
```

在 Agent 决策时调用

```
path = self.planner.search(current_state, self.victory_detector, self.field)
if path:
next_action = path[0] # 执行第一步
```

三、 整合后的决策流程

将这两个模块整合进 choose_action ，Agent 逻辑会变成：

1. **感知**：获取当前图像。
2. **检测**：判断是否已胜利？如果是，结束。
3. **搜索**：如果没有胜利，调用 BFS 搜索从当前状态到胜利状态的最短路径。

- **如果搜到路**：直接执行路径上的第一个动作。
- **如果没搜到路**（比如陷入死胡同或规则太复杂）：
- 回退到“探索模式”：随机选择信息增益最高的动作，或者调用 LLM 寻求建议。

4. **记录**：将 (状态, 动作, 新状态) 存入转移表。

作者：纪辉泉（普宁 515300） 团队 Email：2635714379@qq.com
开源 Github 链接：https://github.com/LingJing1001/Lingjing-Solo- （团队持续开放迭代）