步骤规划器：构建跨层次子目标树作为具身长时段任务规划器

Tianxing Zhou[1,2], Zhirui Wang[1†], Haojia Ao[1†], Guangyan Chen[1], Boyang Xing[3], Jingwen Cheng[3], Yi Yang[1], Yufeng Yue[1*]

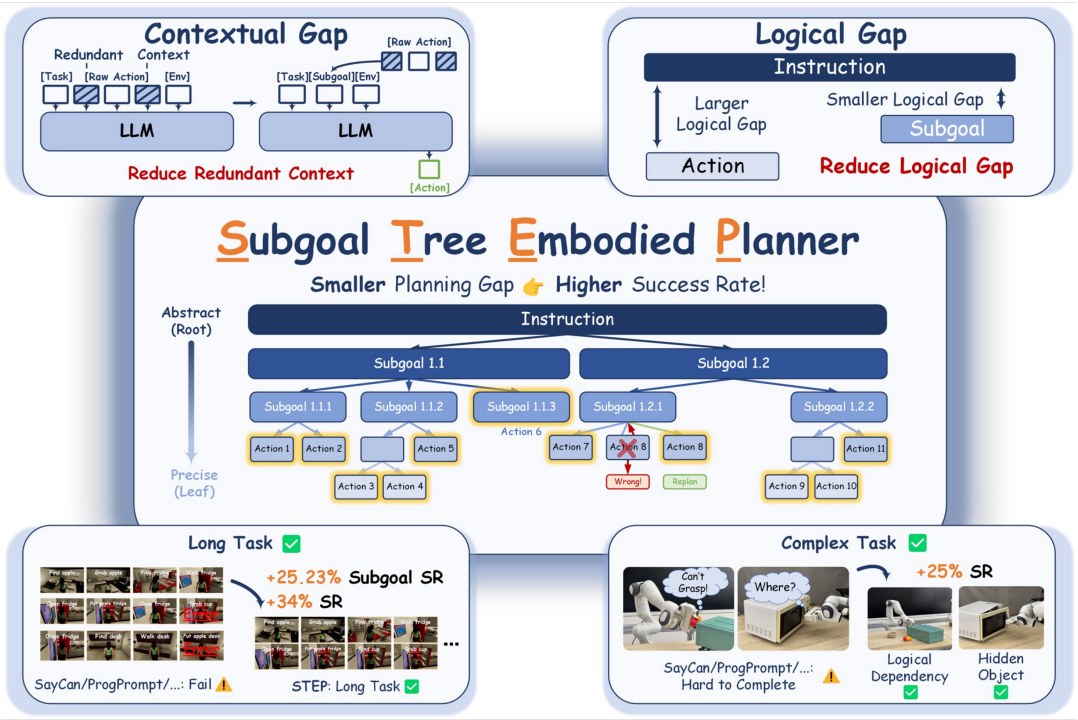

Fig. 1. **我们 STEP 的说明**。通过构建一个分层树结构，将任务从粗粒度分解到细粒度，STEP 在规划过程中有效地弥合了逻辑和上下文的差距，在长周期和复杂任务中表现出优于现有方法的性能。

***Abstract*— 执行长期任务规划的能力对于在现实世界环境中部署机器人至关重要。然而，由于大型语言模型（LLMs）在处理长期具身任务时的推理能力有限，直接使用它们作为动作序列生成器通常会导致成功率较低。在 STEP 框架中，我们通过一对闭环模型构建了一个子目标树：一个子目标分解模型和一个叶节点终止模型。在此框架内，我们开发了一种从粗到细分辨率层次结构的树形结构。子目标分解模型利用基础 LLM 将复杂的目标分解为可管理的子目标，从而扩展了子目标树。叶节点终止模型基于环境状态提供实时反馈，确定何时终止树的扩展，并确保每个叶节点可以直接转换为基本动作。在 VirtualHome WAH-NL 基准测试和实际机器人上进行的实验表明，STEP 实现了长期具身任务完成的成功率分别高达 34%（WAH-NL）和 25%（真实机器人），优于现有最优方法。**

*This work was supported by National Key RD Program of China (Grant No.2024YFB4708900).

* Corresponding author: Yufeng Yue (yueyufeng@bit.edu.cn)

† Equal Contribution

[1] School of Automation, Beijing Institute of Technology, Beijing, 100081, China

[2] Beijing Zhongguancun Academy, Beijing, 100094, China

[3] Humanoid Robot (Shanghai) Co., Ltd., Shanghai, 200093, China

I. 介绍

在现实世界中，大多数任务本质上是长期的，例如整理房间、进行实验或组装设备，这要求机器人具有稳定执行延长动作序列的能力。近年来，由大语言模型赋能的机器人展现了卓越的能力，使它们能够完成许多任务 [1][2][3][4] 。然而，当面对长期的任务，特别是那些涉及复杂场景时，大语言模型规划者往往会产生次优结果。

为了将大语言模型应用于长期任务，之前的研究提出了诸如分步方法如 FLTRNN[5]、LLM-State[6] 等。这些方法通过在规划过程中整合任务分解和内存

管理有效地提高了复杂长期任务中大语言模型的准确性和成功率。然而，这些方法主要依赖大语言模型进行直接推理，因此受到这些模型固有推理限制的约束，导致随着任务复杂性的增加，性能显著下降。

为了解决上述长期规划场景中的性能退化问题，最近的研究受到思想之树（ToT）框架 [7] 的启发，在具身规划中应用了多种采样和计划选择策略，包括 Tree-Planner[8] 和 RAP[9]。这些方法通过在每个决策节点上采样各种计划并选择最优路径来构建树状结构。尽管此类方法在长期任务中展示了更高的成功率和增强的灵活性，但它们从根本上仍然试图在一个实例中生成全面的计划。即使通过从 LLMs 产生的多代结果中选择最优结果来进行优化过程，这一限制依然存在，因为 LLMs 在每个推理步骤中仍直接应用于长序列推理任务。

尽管取得了进展，但将长期任务可靠地转化为一系列动作仍然是一项艰巨的挑战。在长期任务中，性能故障主要源于两个因素：(1) **上下文间隙**：长任务序列固有地向输入上下文引入冗余信息，这一现象在长期任务中尤为明显。随着上下文中冗余信息的比例增加，它会负面影响推理能力。更为明显的上下文差距削弱了规划者理解关键任务相关信息的能力，从而损害其推断性能。(2) **逻辑缺口**：在任务规划情境中，扩展序列通常对应着越来越抽象的语言指令。因此，从抽象指令到具体可执行步骤的推断过程变得更加具有挑战性。短期任务如“拿起一个物体”可以轻松与具体的动作如“抓取”关联起来，而更为复杂的任务如“清理桌子”则相对难以直接连接到原始行动上。更为明显的逻辑差距与较低的推理成功率相关。

根据前面的分析，减少逻辑和上下文差距对于提高机器人在长时任务规划中的表现至关重要。为了实现这一目标，我们提出了 **S**ubgoal**T**ree**E**mbodied**P**lanner（**步骤**）。与 ToT 方法中独立构建每个分支的时间动作序列的树结构不同，STEP 构造的树在每个层级都包含从前一层派生出的子目标，通过逻辑分解系统地建立这棵树，叶节点作为动作序列。这种方法通过子目标树连接指令和行动，减少了它们之间的逻辑差距。此外，由于树结构具有从粗到细分解的特点，使用父节点而不是整个任务作为规划器的输入上下文可以减少冗余上下文。这使得规划器能够专注于每一步的分解过程，从而缩小了上下文差距。

为了构建子目标树结构，我们开发了一个闭环框架，其中包括：(1) 一个**子目标分解模型**，它递归地将复杂任务分解为精确的、离散的子任务，并以层次树结构组织这些子任务。(2) 一个**叶节点终止模型**，评估每个分解节点是否满足任务要求并可以直接映射到特定动作，从而确定何时终止子目标树的扩展。

这种方法通过使规划者专注于当前子目标的分解或终止，有效地减少了规划过程中的逻辑和语境差距。

我们的主要贡献有三个方面：

(I) 我们提出了框架**步骤**，该框架利用基础大语言模型在具身任务中进行长时推理，从而缓解了上下文和逻辑上的差距。这种方法使大语言模型能够更高效地对具身任务进行长链推理。

（二）我们提出一对闭环**子目标分解模型**和**叶节点终止模型**来实现子目标树的扩展和终止，便于动态构建子目标树。

（三）广泛的实验表明，在 VirtualHome 和真实机器人系统中的四种不同操作任务中，SOTA 方法的成功率分别达到了最高 34%（WAH-NL）和 25%（真实机器人），超过了之前的方法。

II. 相关工作

A. LLMs 作为具身规划器

经过大量多模态数据集训练的大语言模型，拥有广泛的关于现实环境的高级先验知识。随着模仿学习的发展，机器人现在能够更有效地执行短任务 [10][11][12] 。先前的研究已经证明了将大语言模型作为具身规划者来实现长时域任务的有效性，通过调用相关的技能库和组合技能函数 [13][14] 。这种方法有效利用了模型的先验知识，将人类自然语言指令转化为可执行的机器人动作序列。然而，由于计划序列与现实世界约束或可能性之间的不匹配，此方法经常会遇到挑战，导致任务成功率降低。

为了解决这些挑战，研究人员采用了强化学习来评估计划 [15]，而其他人则采纳了基于大语言模型的自反馈方法，使他们能够反思自己计划 [16][17] 的正确性。此外，可以引入外部环境反馈来评估任务执行后的场景 [18][19]。强化学习方法可以根据价值函数有效确定规划的准确性，但需要大量的数据才能获得用于行动评估的强大网络。外部反馈方法通过直接纳入环

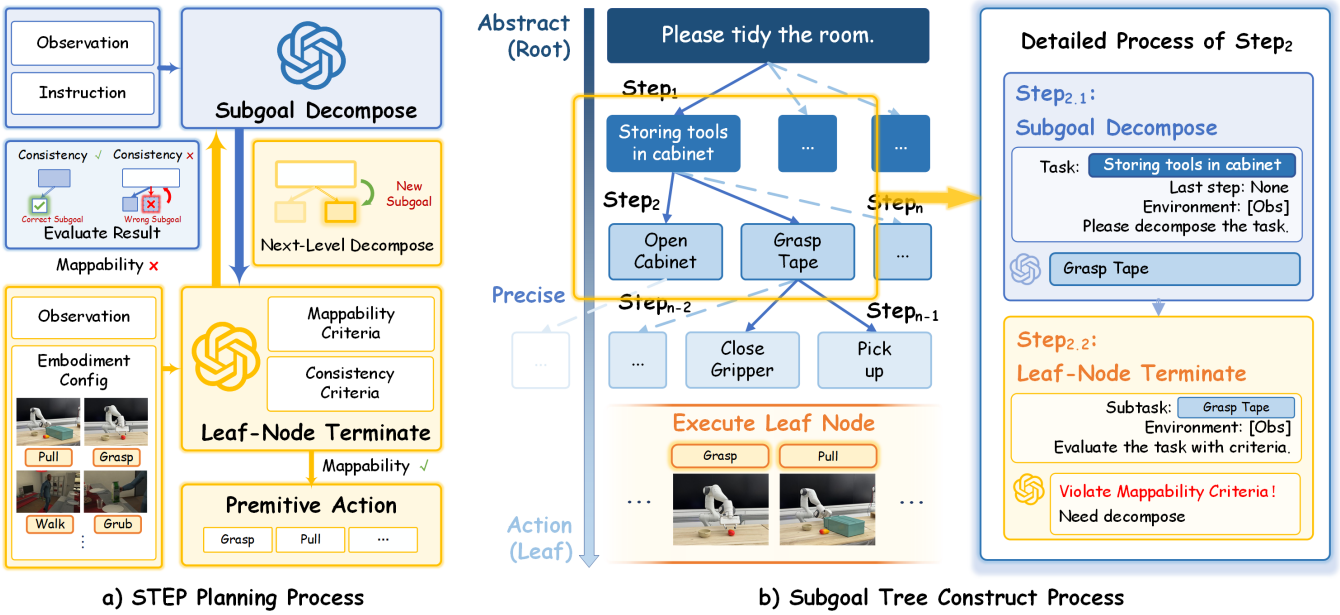

Fig. 2. **STEP 框架。a)** 步骤通过子目标分解和叶节点终止的闭环过程构建一个子目标树。在子目标分解模型中，大语言模型将复杂任务分解为子任务。然后每个子目标根据可映射性和一致性标准进行评估，以确定是否应进一步分解、重新规划或直接执行。**b)** 步骤逐步细化子目标为更精确的动作，并通过迭代的子目标分解和叶节点终止（右侧）来确定每个新生成的子目标（左侧）的下一个可执行动作。这个递归过程生成一个完整的子目标树，如使用从第 2 步开始的进程作为示例所示。

境反馈，使大语言模型能够直观理解执行结果，从而减少内部推理过程 [20]。然而，其实时有效性有限，因为评估只能在执行后进行，限制了它的应用范围。

B. 长链推理与大语言模型

大语言模型在涉及规划和推理的任务中展现出了熟练的能力。在这方面，研究人员已经开始调查大语言模型用于长链条规划的潜力。仅依赖基础大语言模型的研究显示了任务完成上的不足表现。直接将 GPT-4 turbo 用作 Travel Planner 基准中的规划器，导致最终通过率较低 [21]。为了解决这个问题，一些研究者采用重新排序技术 [22] 和迭代校正 [16][23] 来缓解大语言模型所面临的内在困境。然而，这两种方法并没有从根本上解决作为规划器的大语言模型的固有难题。

同时，一些研究人员通过将自然语言任务转化为抽象代码生成 [24][25] 来利用 LLMs 的编码能力。这些方法使 LLMs 能够有效承担长期规划任务，将其转换为更适合处理的推理任务。然而，基于树的方法需要 LLMs 参与常识获取任务，而基于代码或其他逻辑表达式转化方法由于其预定义的基本单元和函数，在开放环境中限制了规划应用。因此，这两种方法在适应复杂环境和任务方面都面临困难，这限制了规划器的应用范围。

III. 方法

在这项研究中，我们提出 STEP 作为一个框架，使具身代理能够将高层次任务分解为一系列基本动作，以应对现实世界中的长期任务。在以下各节中，我们首先分析了具身任务规划中存在的规划差距，并介绍了**子目标树**架构（第 III-A 节）。随后，我们提出了**子目标分解模型**，该方法系统地将高层次任务分解为可执行子任务，减少了逻辑和上下文差距（第 III-B 节）。然后，我们详细介绍了通过开发**叶节点终止模型**来评估子目标树叶节点的方法（第 III-C 节）。通过迭代优化这个任务分解框架，我们的目标是将抽象、复杂的指令转化为具体的可执行动作序列，从而使机器人系统能够有效地适应并完成现实世界环境中的多样化任务。

A. 子目标树

预备知识。在典型的大型语言模型规划器中，规划器被定义为部分可观察的马尔可夫决策过程（POMDP），表示为 $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T} \rangle$，其中 $\mathcal{S}$ 表示状态集，$\mathcal{O}$ 表示观测集，$\mathcal{A}$ 表示动作集，$\mathcal{T}$ 表示转移模型。在实际推理中，生成动作序列的过程被描述为：

$$\pi_\phi(a_t | g, h_t, o_t). \tag{1}$$

此过程利用大语言模型的推理能力，根据目标 g、历史动作 $h_t = \{a_0, \cdots, a_{t-1}\}$ 和当前观察状态 o 生成当前状态的动作 a_t。

然而，该方法论面临一个根本的理论挑战：高级指令与相应的低级可执行动作之间存在显著的**上下文**和**逻辑缺口**。

上下文间隙。从语境角度来看，基础模型在处理和生成长上下文时表现出次优的一致性，这归因于它们基于注意力的架构。这种限制源于长时间输入上下文 [26] 所导致的注意力减少，从而影响内部一致性。同样，在具身化的情境中，延长的任务序列会导致更多冗余信息的输入，因为大语言模型必须处理人类指令和已执行动作的完整历史以生成当前行动。这导致了大量无关上下文被输入，引起了注意力减少，进而导致长期任务规划成功率降低。

逻辑缺口。从逻辑角度来看，基础模型在学习阶段并未经过专门训练将抽象知识转化为具体行动，因此下一标记预测方法遇到了显著的限制 [27]。这种限制阻碍了它们有效地将抽象自然语言指令与特定可执行步骤结合的能力，导致规划复杂实体任务时表现不佳。因此，高层次指令和所需详细动作之间的巨大差距导致在生成长时段任务的全面计划时成功率降低。

子目标树。为了解决上述问题，我们提出了子目标树，这是一种层次结构，系统地从粗粒度逐步细化到细粒度，有效减少了规划过程中逻辑和上下文的差距。在子目标树中，人类指令作为根节点，在规划过程中逐渐分解成多个子节点。每个子节点代表其父节点的一个子目标，通过所有子节点的累积效应可以实现每个父节点的效果。这个分解过程递归进行，直到每个叶节点对应一个可执行的操作。此时，叶节点的序列表示要执行的具体操作，所有叶节点的累积效果与根节点的预期效果一致，从而确保了分解的可靠性。

通过缩小层级之间的逻辑差距，该方法增强了每个推理过程的合理性和连贯性。通过实施单层推理工作流，规划者利用来自其父节点的上下文输出和当前层次内的顺序推理历史，从而充分利用父任务内在包含的高层次使命目标，导致逻辑差距降低。

因此，分层分解框架确保了来自子目标树不同分支的历史行动系统地被排除在当前规划过程的输入上下文之外。这种架构约束有效地将所需的任务分解转换为精炼且与情境相关的必要输入信息表示。通过消除冗余的上下文输入，这种方法使 LLM 规划器能够更精确地专注于特定的分解任务，从而产生明显更优的规划结果。

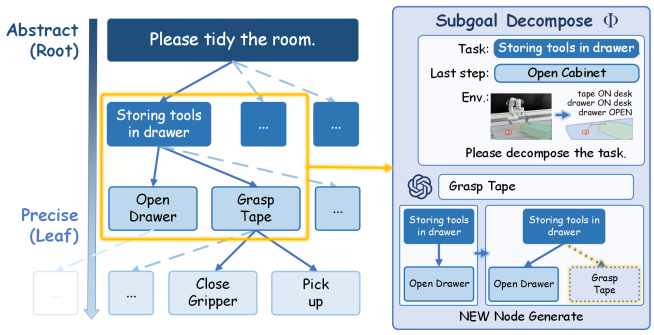

Fig. 3. **子目标分解模型的框架**。将“将工具存入抽屉”作为分解的例子，子目标分解模型可以直接从父节点和上一步推断出要生成的下一个子目标，即抓取胶带。

B. 子目标分解模型

我们将分解模型公式化为一个 POMDP。在一个 POMDP 设置中，观察值 o_t 代表潜在状态 s_t 的一个子集。代替方程 (1) 其中 g 表示自然语言中的目标，h_t 作为行动历史，o_t 作为观察结果，子目标分解的过程可以描述为：

$$\pi_{tree}(\Phi^n(\{b_i\}|_{i=0}^n)|\Phi^{n-1}(\{b_i\}|_{i=0}^{n-1}), \Phi^n(\{b_n\}|_{i=0}^{n-1}, b_{i-1}), o_t). \tag{2}$$

其中 Φ 表示目标分解算子，b_i 表示第 $(i-1)$ 层子目标的第 b_i 个分支 (b_i= 0, 1, 2...)，$\Phi^n(\{b_i\}|_{i=0}^n)$ 表示分支 $\{b_0,b_1,...,b_{n-1}\}$ 的第 b_n 个子目标。生成的子目标由父节点、左侧节点和当前环境观察决定。因此，我们可以推断出：

$$\Phi^n(\{b_i\}|_{i=0}^n) = \Phi(\Phi^{n-1}(\{b_i\}|_{i=0}^{n-1}), b_n). \tag{3}$$

显示了基本的子目标分解过程。

具体过程如图 (3) 所示。在每一步中，规划器获得当前观测 o_t、较高层级的规划结果 $\Phi^{n-1}(\{b_i\}|_{i=0}^{n-1})$ 以及同一层级的上一步 $\Phi^n(\{b_n\}|_{i=0}^{n-1}, b_{i-1})$。

C. 叶节点终止模型

为了便于终止子目标树分解过程，我们设计了一个与实现相关的叶节点终止模型。该模型旨在当通过分解达到的细节程度能够精确地将生成的组件映射到一个基本操作时停止进一步细化。

在叶节点终止模型中，由子目标树生成的当前子任务、以及更高层次的任务要求、同一层级的先前子任

务和具身配置被输入到 LLM。然后，LLM 使用上述评估过程对来自子目标树的每个子任务输出进行评估。

因此，我们认为与体现相关的叶节点终止模型必须具备两个基本功能，其中包括：

可映射性标准. 评估任务是否可以映射为显式的基本动作。标准如下所示：

$$\forall a_i \neq a_j, f(\Phi_T(i)) \to a_i, f(\Phi_T(j)) \to a_j, \\ \Phi_T(i) \neq \Phi_T(j). \tag{4}$$

2. **一致性标准**. 确定当前子目标是否满足实现的适用性，与前一个子目标的要求一致，并且可以在当前环境中执行。

功能性. 子任务与具身能力的兼容性评估 ，表示 $\mathcal{A}(\mathcal{E})$ 为实现 $\mathcal{E}$ 的功能，$\mathcal{A}$ 表示子目标树的叶节点作为动作 $\mathcal{A} =< a_0 \wedge a_1 \cdots \wedge a_n >$，$\Phi_T$ 作为叶节点。标准如下所示：

$$\forall a_i \in \mathcal{A}, f(\Phi_T(i)) \to a_t, \\ a_i \in \mathcal{A}(\mathcal{E}). \tag{5}$$

例如，在某些情况下，大型语言模型可能会忽略当前实现的能力限制，导致规划不可行的任务。一个具体的例子就是一个单臂在已经持有物体的情况下被指示用其末端执行器执行额外任务。因此，一个好的叶节点终止模型应该包含评估提议计划可行性的功能，包括当前实现执行该计划的能力以及该计划在现有环境中的可行性。

任务一致性。在当前环境约束下评估子任务的可执行性，确定是否可以无需进一步分解即可继续实施。标准显示如下：

$$\forall \Phi^n(\{b_i\}|_{i=0}^n) \in T, \\ \Phi^n(\{b_i\}|_{i=0}^n) = \Phi(\Phi^{n-1}(\{b_i\}|_{i=0}^{n-1}), b_n). \tag{6}$$

环境. 当前环境约束下子任务可执行性的评估，确定是否可以在不进一步分解的情况下继续实施，表示 $\mathcal{A}(\mathcal{S})$ 为状态约束，T_t，将 T 的叶节点表示为轨迹 $T_t =< a_0 \wedge a_1 \cdots \wedge a_n >$。标准如下所示：

$$\forall a_i \in \mathcal{A}, f(\Phi_T(i)) \to a_t, \\ a_i \in \mathcal{A}(\mathcal{S}). \tag{7}$$

标准的逻辑结构如图 (4) 所示。

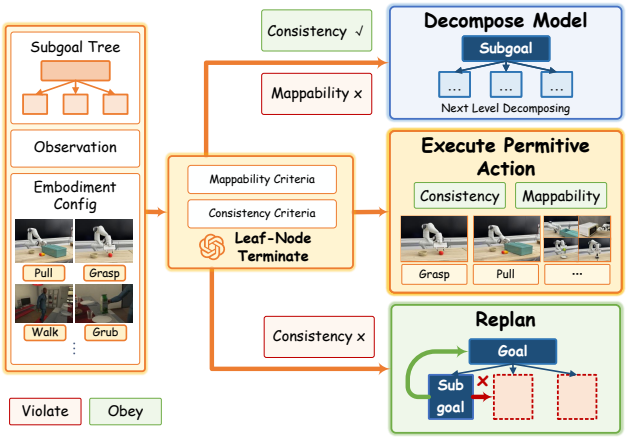

Fig. 4. **叶节点终止模型框架**。可映射性标准和一致性标准用于确定当前节点应执行原始动作、执行下一个分解，还是重新规划。

- 如果一个任务满足两个标准，则将其映射到宏操作，并返回成功状态。
- 如果一项任务不符合可映射性标准但满足一致性标准，则返回进一步细化的指示。
- 如果一个任务未能满足一致性标准，则返回规划错误，并重新启动上一轮的子目标规划。

总之，子目标树构建过程在算法 (1) 中进行了描述。

IV. 实验

A. 虚拟家庭模拟器的评估

环境。我们在 VirtualHome 环境中进行了实验 [28]，这是一个为复杂的家庭任务设计的模拟平台。在实验中，我们使用了 WAH-NL 基准测试 [29]，该基准测试旨在将自然语言指令转换成现实生活场景中的动作序列。这个基准测试对机器人任务规划提出了挑战，原因包括每个任务步骤较长、场景内物体丰富以及物体之间的逻辑依赖（例如，获取柜子内的物品必须先打开柜子）。

数据集。我们在 WAH-NL 基准的包含 350 个标记机器人数据集的注释数据集中进行了测试，该数据集包含 100 条记录。每个数据集至少包括一个自然语言指令、初始场景状态以及至少一个子目标完成标准。如果所有子目标标准均得到满足，则认为整体任务目标已完成。评估后，我们获得了子目标成功率（各子目标的成功率）和总体任务成功率。

基线。作为实验，我们使用了 WAH-NL 基准来比较 Saycan[15]、LoTa-Bench[29] 和 ProgPrompt[30] 方法，并分别分析了这三种方法的失败模式。基于

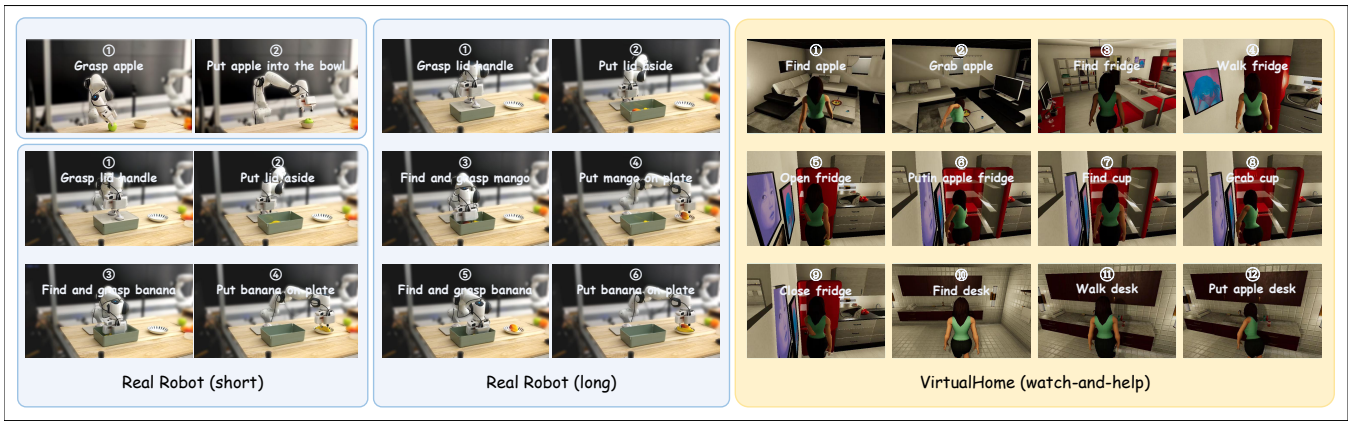

Fig. 5. **规划结果**. 该实验在 Virtual Home 模拟器和真实机器人上进行了测试，展示了 STEP 在复杂环境和长期任务中的规划性能。

Algorithm 1 子目标树构造算法

Input: 指令 *goal*, 观察 *obs*

Output: 子目标树 T

子目标 =*goal*
树添加节点 (子目标)　　▷ *Initialize subgoal tree*
while evaluate(Tree.allLeafNode) is not mappable **do**
　子目标 = 分解 (子目标,*obs*)
　　▷ *Decompose current subgoal*
　if evaluate(subgoal) is mappable **then**
　　执行 (子目标. 动作)
　　▷ *Execute after successfully decomposed*
　　if allNeighbourComplete(subgoal) **then**
　　子目标 = 子目标. 父项. 下一个
　　▷ *Return to parent subgoal after complete*
　　else
　　子目标 = 子目标. 下一个
　　▷ *Move on to next subgoal*
　　end if
　else
　　子目标 = 子目标. 父项
　　▷ *Replan parent node when error occurs*
　end if
end while

EAI[31]，我们将它们分类为语法错误、目标满足误差（包括缺失状态、缺失关系、缺失目标动作）和轨迹运行时错误（包括错误顺序、功能错误、多余/缺少步骤）。

主要结果。根据表 (I) 中的结果，可以获得 STEP 的几个优势：

TABLE I

不同方法在 WAH-NL 基准上的性能。

方法	序列。↑	SSR。↑
SayCan[15]	1%	2.07%
ProgPrompt[30]	3%	18.69%
LoTa-Bench[29]	6%	36.79%
步骤 (我们的)	**40%**	**62.02%**

(i) STEP 在 WAH-NL 中的表现优于基线系统，成功率提高了 34%，子目标成功率提高了 25.23%。LoTa-Bench 和 SayCan 的实证性能表现出与 [29] 中报告的结果相似的成果。

(ii) 实验表明，在 WAH-NL 基准测试中，成功率的提升比例远高于子目标成功率的提升比例，这说明 STEP 对时间序列较长的任务有更大的影响，表明基于子目标树的任务分解方法能够更好地应对长时任务。

结果分析。通过对表 (II) 中各种基线失败结果的分析。STEP 在所有级别上都优于其他所有基线，在目标满足误差和轨迹运行时错误方面表现更佳。STEP 的主要错误类型被归类为“额外/缺少步骤”，这是由于基于大语言模型的叶节点终止模型的局限性造成的。这种限制源于叶节点终止模型依赖于大语言模型，尽管它们具有强大的能力，但基础模型仍然表现出相对有限的环境理解力，在某些场景中可能导致误判。这构成了 STEP 框架进一步性能改进的主要瓶颈。

此外，STEP(15%) 和基线方法 (平均 16.3%) 之间的语法错误比率没有显著差异。这可以归因于这两种方法使用相同的主干网络，主要的语法错误来源来自

TABLE II

不同方法在 WAH-NL 基准上的误差类型分析。

方法	目标满足误差			轨迹运行时错误		
	缺失状态	缺失的关系	缺失目标行动	错误顺序	额外/缺失步骤	功效误差
Saycan[15]	5%	7%	9%	9%	32%	15%
Prog-Prompt[30]	2%	6%	10%	11%	36%	12%
LoTa-Bench[29]	3%	3%	9%	28%	36%	13%
步骤（我们的）	**1%**	**2%**	**2%**	**2%**	**27%**	**3%**

大语言模型的幻觉。因此，不同的方案表现出相似的语法错误比率。

B. 真实机器人评估

对于底层规划器，我们使用了 RoboScript API[32] 并将其部署在实际机器人上。我们将任务分为四个等级。

- “短简单”类别包含少于 5 个步骤且所有对象均可见。
- “简短复杂”类别包含少于 5 个步骤，但包括隐藏的物体（例如，抽屉里的物品）。
- 长简类别包含 5 到 8 个步骤，并且所有对象都是可见的。
- 长复杂类别包含 5 到 8 个步骤，并包括隐藏对象。

每个组包含 2 个任务，每个任务使用来自 IV-A 部分的基线进行 5 次测试。

环境设置。在实际机器人实验中，我们选择了 Franka Emika Panda，并使用了 OpenAI 的 GPT-4o 作为大语言模型的核心。对于底层规划器，我们采用了 RoboScript[32] 的 API，负责低级运动规划和控制，并成功将这个集成系统部署到了实际机器人上。

主要结果。如表 (III) 所示，在真实机器人实验中，我们观察到在简单任务中，STEP 的性能与不同基线相似。

在更复杂的任务中，随着任务长度的增加，STEP 表现出优越的性能。

随着任务复杂度的增加（需要识别未见过的物体），STEP 的性能优势变得更加明显。

即使使用相同的 LLM 骨干网络，STEP 在需要更强推理能力的任务中表现更好。

TABLE III

不同方法在真实机器人上的成功率。

方法	短简单	短复数	长简单	长复数
Saycan[15]	8/10	6/10	4/10	1/10
Prog-Prompt[30]	9/10	8/10	3/10	1/10
RoboScript[32]	9/10	8/10	5/10	1/10
LoTa-Bench[29]	10/10	8/10	6/10	1/10
步骤 (我们的)	**10/10**	**10/10**	**9/10**	**6/10**

C. 消融研究

为了评估 STEP 的不同组件对规划成功率的贡献，我们设计了以下消融实验：

无树结构：在这种情况下，我们通过用原始父节点和之前执行的所有叶节点替换父节点来修改每个任务的输入信息。此修改消除了树结构对当前任务的影响，同时保持了类似的逻辑差距。由此产生的冗余信息使我们可以具体研究树结构如何通过减少上下文差距来提高成功率。

(2) **无子目标树：**在这种情况下，我们通过用根节点替换父节点并替换所有先前执行的叶节点来修改每个任务的输入信息。这种方法消除了子目标树结构，去掉了 STEP 方法中通过多步推理实现的任务压缩效果和推理难度降低。通过保持类似于“-w/o tree”组的上下文差距，这种条件使我们能够研究子目标任务压缩对任务规划效果的具体影响。

评估上述方法在第 IV-A 节中。

结果分析。表 (IV) 中所示的评估结果表明，与完整的 STEP 实现相比，消融子目标和树结构的 STEP 的成功率降低了。

如图 (6) 所示，所有方法的成功率随着任务长度的增加而下降。

TABLE IV

不同方法在 WAH-NL 基准上的性能。

方法	序列号。	SSR.
我们的全部	**40%**	**62.02%**
-w/o tree structure	8%	34.12%
-w/o subgoal tree	9%	27.30%

如观察到的，”-w/o 树结构” 配置的表现劣于” 完整 STEP” 方法，而”-w/o 子目标树” 配置的表现甚至比”-w/o 树结构” 更差。此证据证明了子目标树结构在减少逻辑和上下文差距方面的能力共同促成了 STEP 的有效性提升。

值得注意的是，具有去除子目标树的方法表现出更高的下降率，这归因于子目标树减少抽象指令与行动之间更大逻辑差距的原则，在更复杂的任务中尤为如此。因此，随着任务变长，逻辑差距对性能的影响变得越来越明显，证明了子目标树在规划过程中有效地缩小了逻辑差距。

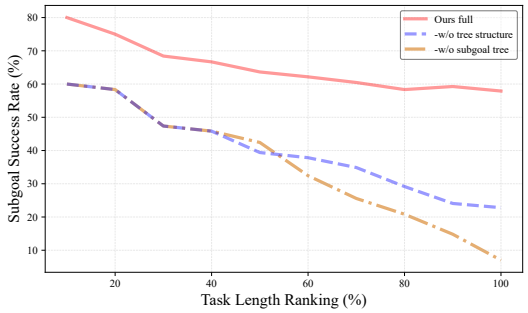

Fig. 6. **任务长度和子目标成功率在不同基线下的变化**。在 WAH 基准测试中，我们将任务按长度分为 10%的区间，从短到长，并计算每个区间内任务的子目标成功率。

V. 结论

本文中，我们介绍了 STEP，一种使 LLMs 适应长期范围的具身规划任务的一般方法。通过构建子目标树，STEP 减少了具身规划任务中的上下文和逻辑差距，使 LLMs 能够有效地作为机器人规划器处理复杂的具身任务。首先，STEP 使用子目标分解模型分层构建子目标树来分解任务。然后，它采用叶节点终止模型确定何时终止每个分支的扩展。完成分支后，执行相应的动作。在模拟器和真实机器人中进行的实验表明，在具身任务中，特别是在长期范围场景和复杂环境中，STEP 显著优于基线方法。总结来说，STEP 扩展了 LLMs 在机器人具身任务中的适用性，使机器人能够更好地在开放世界环境中执行长期的人类指令。

致谢

我们衷心感谢北京理工大学的袁益君先生在实验过程中提供的帮助。

References

[1] Mu Yao, Zhang Qinglong, Hu Mengkang, and et. al. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Chen Guanyan, Wang Meiling, Cui Te, and et. al. Vlmimic: Vision language models are visual imitation learner for fine-grained actions. *arXiv preprint arXiv:2410.20927*, 2024.

[3] Wake Naoki, Kanehira Atsushi, Sasabuchi Kazuhiro, and et. al. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *IEEE Robotics and Automation Letters*, 2024.

[4] Guangyan Chen, Meiling Wang, Te Cui, Yao Mu, Haoyang Lu, Zicai Peng, Mengxiao Hu, Tianxing Zhou, Mengyin Fu, Yi Yang, and Yufeng Yue. Fmimic: Foundation models are fine-grained action learners from human videos. *arXiv preprint*, 2025.

[5] Sermanet Pierre, Ding Tianli, Zhao Jeffrey, and et. al. Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 645–652. IEEE, 2024.

[6] S P Sharan, Zhao Ruihan, topcu ufuk, and et. al. Plan diffuser: Grounding llm planners with diffusion models for robotic manipulation. In *Bridging the Gap between Cognitive Science and Robot Learning in the Real World: Progresses and New Directions*, 2024.

[7] Yao Shunyu, Yu Dian, Zhao Jeffrey, and et. al. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[8] Yao Shunyu, Yu Dian, Zhao Jeffrey, and et. al. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Hao Shibo, Gu Yi, Ma Haodi, and et. al. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[10] Guangyan Chen, Meiling Wang, Te Cui, Luojie Yang, Qi Shao, Lin Zhao, Tianle Zhang, Yihang Li, Yi Yang, and Yufeng Yue. Unifying latent action and latent state pre-training for policy learning from videos. *arXiv preprint*, 2025.

[11] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. *pi*_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[12] Guangyan Chen, Te Cui, Meiling Wang, Chengcai Yang, Mengxiao Hu, Haoyang Lu, Yao Mu, Zicai Peng, Tianxing Zhou, Xinran Jiang, Yi Yang, and Yufeng Yue. Graphmimic: Graph-to-graphs generative modeling from videos for policy learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[13] Vemprala Sai, Bonatti Rogerio, Bucker Arthur, and et. al. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, 2024.

[14] Yao Shunyu, Zhao Jeffrey, Yu Dian, and et. al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[15] Ahn Michael, Brohan Anthony, Brown Noah, and et. al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[16] Shinn Noah, Cassano Federico, Berman Edward, and et. al. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[17] Lin Bill, Yuchen, Fu Yicheng, Yang Karina, and et. al. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Huang Wenlong, Xia Fei, Xiao Ted, and et. al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[19] Cui Te, Chen Guangyan, Zhou Tianxing, and et. al. Human demonstrations are generalizable knowledge for robots. *arXiv preprint arXiv:2312.02419*, 2023.

[20] Vineet Bhat, Ali Umut Kaypak, Prashanth Krishnamurthy, and et. al. Grounding llms for robot task planning using closed-loop state feedback. *arXiv preprint arXiv:2402.08546*, 2024.

[21] Achiam Josh, Adler Steven, Agarwal Sandhini, and et. al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[22] Wang Xuezhi, Wei Jason, Schuurmans Dale, and et. al. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[23] Madaan Aman, Tandon Niket, Gupta Prakhar, and et. al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Chaffin Antoine, Claveau Vincent, Kijak Ewa, and et. al. Ppl-mcts: Constrained textual generation through discriminator-guided mcts decoding. *arXiv preprint arXiv:2109.13582*, 2021.

[25] Gu Yu, Deng Xiang, Su Yu, and et. al. Don't generate, discriminate: A proposal for grounding language models to real-world environments. *arXiv preprint arXiv:2212.09736*, 2022.

[26] Freda Shi, Xinyun Chen, Kanishka Misra, and et. al. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[27] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

[28] Puig Xavier, Ra Kevin, Boben Marko, and et. al. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018.

[29] Choi Jae-Woo, Yoon Youngwoo, Ong Hyobin, and et. al. Lota-bench: Benchmarking language-oriented task planners for embodied agents. *arXiv preprint arXiv:2402.08178*, 2024.

[30] Singh Ishika, Blukis Valts, Mousavian Arsalan, and et. al. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[31] Li Manling, Zhao Shiyu, Wang Qineng, and et. al. Embodied agent

interface: Benchmarking llms for embodied decision making. *arXiv preprint arXiv:2410.07166*, 2024.

[32] Chen Junting, Mu Yao, Yu Qiaojun, and et. al. Roboscript: Code generation for free-form manipulation tasks across real and simulation. *arXiv preprint arXiv:2402.14623*, 2024.