文章编号:1671-4598(2025)05-0255-07 DOI:10.16526/j.cnki.11-4762/tp.2025.05.030 中图分类号:TP39 文献标识码:A

基于 AIGC 的机械臂抓取动作序列自动生成研究

王 彬,马兴录,陈彦泽
(青岛科技大学 信息科学技术学院,山东 青岛 266061)

摘要:目前的抓取规划方法往往受限于预定义的规则或手工制定的动作序列,限制了其适用性和灵活性,为了扩展机械臂抓取的适用空间,提高机械臂自主抓取的智能化水平,探索了一种基于 AIGC 的机械臂抓取动作序列的自动生成方法;该方法以开源中文大语言模型 ChatGLM2 为核心,利用大语言模型的生成能力,自动生成机械臂抓取动作序列并生成可执行的程序;为验证提出的方法,选择了 UR5 机械臂作为实验对象,并在 Coppelisim 仿真环境中进行了大量实验,通过验证生成的抓取动作序列程序能否成功完成抓取任务,评估了所提出方法的有效性;实验结果表明,基于 AIGC 的自动生成方法能够有效地生成符合实验目标的机械臂抓取动作序列以及可执行的程序。

关键词:ChatGLM2;机械臂抓取;虚拟仿真;动作序列;自动生成

Research on Automatic Generation of Robotic Arm Grabbing Action Sequence Based on AIGC

WANG Bin, MA Xinglu, CHEN Yanze
(School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China)

Abstract: Current grabbing planning methods are often constrained by pre-defined rules or manually specified action sequences, which limits their applicability and flexibility. In order to expand the applicable range of grasping robotic arms and improve the intelligence level grasped autonomously by robotic arms, an AI generated content (AIGC) based automatic generation method for robotic arm grabbing action sequences is presented. Taking the open-source Chinese large language model ChatGLM2 as the center, this method adopts generation capabilities of large language models to automatically generate the sequences of robotic arm movements and create executable programs. To verify the proposed method, the UR5 robotic arm was selected as the experimental subject, and a large number of experiments were conducted in the Coppelisim simulation environment. By verifying whether the generated sequence program can successfully complete the grasping task, the effectiveness of the proposed method was evaluated. Experimental results show that the automatic generation method based on AIGC can effectively generate the robotic arm grasping action sequence and executable program and meet the experimental goal.

Keywords: ChatGLM2; robotic arm grabbing; virtual simulation; action sequence; automatic generation

0 引言

机械臂作为从工业制造到仓储等应用机器人的关键组件之一,其效率高以及高精度等特性可以在自动化以及协作方面发挥重要作用,已经被广泛应用;然而,传统的机械臂抓取动作都是通过提前设置好的路径或者预定义的规则[1],一旦任务发生改变,就需要重新进行路径规划以及程序的改写、检查部署等步骤,且无法完成交互[2],这限制了其适应性和灵活性。

AIGC (AI-Generated Content) 指使用 AI 技术自动生成的内容,近几年来,随着大语言模型的提出以及 ChatGPT 等大语言模型的迅速崛起,为人工智能领域的发展开创了新的纪元,作为 AIGC 最主要技术基础之一的大语言模型是在大量的文本数据集上训练得来,在文本生成、语言理解、决策、规划和推理等方面有不俗

收稿日期:2024-02-29; **修回日期**:2024-04-03。
基金项目:国家自然科学基金(62201314);山东省自然科学基金(ZR2020QF007)。
作者简介:王 彬(1996-),男,硕士。
通讯作者:马兴录(1970-),男,硕士,副教授。
引用格式:王 彬,马兴录,陈彦泽.基于 AIGC 的机械臂抓取动作序列自动生成研究[J].计算机测量与控制,2025,33(5):255-261.

的表现，而且还可应用于代码生成、程序分析等领域，其强大的能力以及应用前景使得国内外无数学者纷纷投入其中。

自从谷歌发布了 SayCan 以来，使用大语言模型来驱使机器人成为了当前的研究热点[3]，国内外对其研究包括但不限于交互、知识、内存和记忆、推理与规划、转移性和泛化、多模态输入以及工具的使用等方面，其中也涌现出了很多优秀的成果。

但是现有的研究方法基本无法适用于长期的任务执行[4]，利用大语言模型生成的代码仅仅局限于下一步的动作[5]，当需要多个步骤才能完成任务时，很难一次性生成整个执行程序，因此，本文的旨在研究如何运用大型语言模型的生成能力来实现长期任务规划，并自动生成能准确完成任务的完整控制程序。这样，用户只需直观地表达出需求，大语言模型即可根据用户传递的自然语言来识别出用户目的，做出相应的回应及决策，简便，高效，大大降低了使用门槛[6]。

1 系统整体设计

1.1 系统运作流程

系统的整体设计流程图如图 1 所示，由用户通过自然语言输入描述抓取任务的要求和目标，大语言模型对用户输入的自然语言进行分析和理解，进而判断出用户的目的和任务要求，然后将整个任务分解成多个子任务，在对任务进行分解之后，大模型根据具体的子任务将子任务映射到对应的函数上，函数包括了定位、移动、抓取等功能，在这一过程中，大语言模型不会直接生成代码，而是继续进入检查环节，用来重新审查任务要求以及生成序列的合理性，如果有子任务无法匹配到对应的函数，则会将子任务进一步分解去匹配函数库，当给出的函数库足够全面时，子任务都会得到相对应的函数，然后实现代码的生成；根据任务的先后顺序和依赖关系，将子任务按照合适的顺序连接起来，形成完整的抓取动作序列；之后继续使用大模型对生成的代码进行反思和检查，以改进代码，确保符合用户的要求，提高代码的可执行性；最后将生成的代码在虚拟仿真环境中进行验证，观察生成的抓取动作序列是否满足预期的任务要求。在这个过程中，也可以将大模型犯下的错误记录下来，在同一轮的实验中，将错误地方以提示的方式输入到大模型，避免大模型出现相同的错误[7]。

1.2 系统组成

系统的主要组成部分是大语言模型，本文选用的是清华大学开源的大语言模型 ChatGLM2-6B，ChatGLM2-6B 是中英文开对话模型 ChatGLM-6B 的迭代版本，不仅保留了初代模型对话流畅，部署简单等特性之外，相较于初代版本，具有更强大的性能，支持更长的上下文，还有更高效的推理、更低的显存占用以及开放

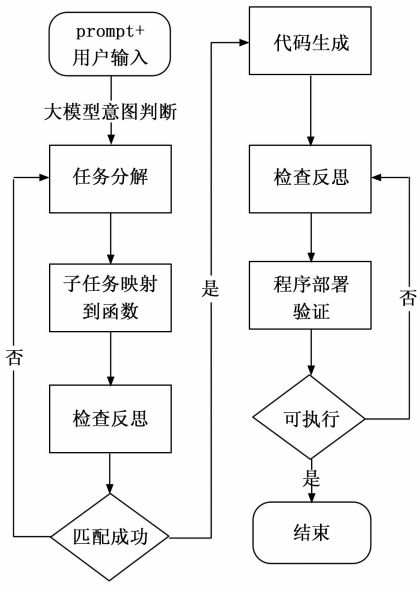

图 1 系统流程图

的协议，不仅对学术研究完全开放，更允许登记后免费商业使用[8]。

本文搭载平台是 13 700 k＋4 080，系统是 Ubuntu 20.04 LTS，在不进行量化的情况下进行推理和微调。

仿真实验平台选用的是 CoppeliaSim Edu 4.3 版本，机械臂选用 UR5，机械爪选用 RG2，使用 Python 代码通过 sim.py 函数库与 CoppeliaSim 软件进行通讯来控制机械臂的动作。

2 研究方法

本文提出了一种机械臂抓取动作序列自动生成框架，由用户语音或文字输入自然语言，大语言模型对用户的输入的自然语言进行意图的判断，理解用户所输入自然语言的含义，任务分解部分负责调用大语言模型对理解的任务指令进行分解，并将分解得到的动作序列与函数库中的函数进行匹配，以生成执行程序的主体部分，任务反思部分负责将生成的动作序列进行检查，确保分解出的动作序列可以实现任务要求，并对生成的代码部分进行检查，以提高代码的可执行性。

2.1 模型的搭建与微调

根据官方教程下载模型文件，安装依赖之后，本地部署了 ChatGLM2-6B 的模型，可以网页端调用也可以本地终端直接调用，网页端的回复速度略快于本地终端；实际测试下来，因为只有 6B 的通用参数，效果与 ChatGPT 相比还有一定的差距，通常提高模型性能的途径就是增加参数和微调[9]，所以针对具体的实验场景以及要实现的目标准备了微调数据集，实验数据集包括 900 多条数据，包括对用户意图的理解，目的提取、任务的分解及生成代码的示例等。微调数据集如图 2 所示；采用 LoRA 方式进行轻量级微调[10]。

```
{
    "instruction": "请将黄色圆柱体移动到绿色桌子上",
    "input": "",
    "output": "将黄色圆柱体移动到绿色桌子上，可以分解成以下动作序列：1.找到圆柱体和桌子的位置坐标2.移动机械臂到圆柱体上方3.执行
  },
{
    "instruction": "请将红色长方体放在蓝色长方体旁边",
    "input": "",
    "output": "将红色长方体放在蓝色长方体旁边，可以分解成以下动作序列：1.找到红色和蓝色长方体的位置坐标2.移动机械臂到红色长方体
  },

{
    "instruction": "请将橙色圆球放在紫色立方体上方",
    "input": "",
    "output": "将橙色圆球放在紫色立方体上方，可以分解成以下动作序列：1.找到橙色圆球和紫色立方体的位置坐标2.移动机械臂到橙色圆球
  },
{
    "instruction": "请在蓝色平台上放置一个红色球体，然后将绿色圆柱体放在红色球体上方",
    "input": "",
    "output": "将蓝色平台上放置一个红色球体的动作序列：1.找到蓝色平台的位置坐标2.移动机械臂到平台上方3.执行放置动作4.找到红色球
  },
```


图 2　数据集示例

微调前后效果对比如图 3、图 4 所示，假设用户指令是：将红色的方块放到绿色的篮子里，与 Prompt 一起组成用户输入。

用户输入：现在场景中有一台机械臂，还有各种不同颜色的小方块以及各种颜色的篮子，将所有的红色的方块放到绿色的篮子里

chatglm：将用户输入的自然语言分解成具体的动作序列可以分为以下几个步骤：
1. 识别语义：将用户输入的自然语言转换为语义动作。例如，“将所有的红色的方块放到绿色的篮子里”可以识别为“/将/所有的/红/方/块/放到/绿/的/篮/子里”。
2. 解析动作：将语义动作进一步解析成具体的动作序列。例如，识别出“将”和“放到”是两个独立的动作，而“红”和“方”则是两个新的语义动作。可以将这些语义动作组合成一个动作序列，例如“/将所有的红方块放到绿色篮子里”。
需要注意的是，由于机械臂和篮子可能不在机械臂的抓取范围内，或者由于其他原因无法执行操作，因此需要对生成的代码进行验证和调整。
在实际应用中，为了更好地处理自然语言理解和运动规划的问题，通常需要使用深度学习技术，例如循环神经网络（RNN）和卷积神经网络（CNN），来生成可以执行的动作序列。

图 3　微调前

用户输入：现在场景中有一台机械臂，还有各种不同颜色的小方块以及各种颜色的篮子，将所有的红色的方块放到绿色的篮子里

chatglm：将所有的红色的方块放到绿色的篮子里，可以分解成以下动作序列：
1. 识别红色方块
2. 找到方块位置
3. 移动机械臂到方块位置
4. 执行抓取
5. 识别绿色篮子
6. 找到绿色篮子位置
7. 移动机械臂到绿色篮子位置
8. 放下篮子。

图 4　微调后

在经过微调以及合适的 Prompt 之后，模型可以达到预期使用效果。

2.2　提示（Prompt）机制

在研究如何让机械臂做正确的事情方面，Prompt Engineering 绝对是一个很好的解决方案；如果把大语言模型当做是帮助用户完成任务的工具，Prompt 则是给这个工具的前提条件，它告诉大模型应该如何去做，以及达到什么样的程度；Prompt 通常只是一句话，一段文字等等形式，因为对大模型理解用户的自然语言起到帮助作用，所以后来被称为 Prompt[11]；到目前，有关 Prompt Engineering 已经衍生出很多种类，而 Few-shot Prompt 在使用中最为广泛，由于大模型受到参数量以及数据集的知识库时间的限制，所以往往很多时候大模型并不能很好的回答用户的问题，而 Few-shot Prompt 旨在给大模型的输入中加入少量的示例，相当于给予大模型一些额外的知识，所以大模型在输出上会表现更好[12]。

在 Prompt 中，除了需要包含用户需要大语言模型做什么之外，如何进行额外的描述来引导大语言模型的行为是判断优质 Prompt 的关键。

在面对机械臂的抓取动作序列生成中，建立一套专用的提示方法可以帮助大语言模型理解任务，理解场景，能够更好地帮助用户完成任务[13]；在对应到机械臂时，应该有具体的底层控制以及封装好的函数（例如控制夹爪开关的函数，移动的函数，调用各种外部传感器以及运动学函数等）可以直接调用来达到控制机械臂的效果，然后通过 Prompt 将这些函数以及规则（包括需要生成什么样的代码，生成代码的哪一部分，需不需要有多余的解释或者使用循环嵌套等规则）告诉大语言模型，大语言模型根据规则生成可部署的程序代码，可以由用户或者设计好的程序来自动部署程序控制机械臂完成任务[14]，实验表明，在经过合适的 Prompt 之后，大语言模型生成的代码可以满足预期的目标。

2.3　任务分解

在一开始的实验测试中，实验的最先测试是直接让大模型生成测试代码，发现在一些简单的任务上大模型生成的代码可以应用在机械臂上实现任务目标，但是当目标稍微复杂一些，尤其是涉及到连续任务的时候大模型就会表现出“胡言乱语”的现象，即便是给出的描述已经足够详细，但是由于提示给出的函数只是一些基础的函数，所有的复杂任务控制都是由基础函数一步步拼接而来，而用户所描述的任务几乎只包含了开始和结束，在不告诉大语言模型如何做的情况下，几乎不可能生成有效的代码来完成任务。

为了解决大语言模型“胡言乱语”的问题，之前的研究中像思维链或者 LtM 考虑通过给大模型一个以人

类思维思考的例子来让大语言模型进行类比学习[15]，或者将一个问题分解成子问题逐步解决，还有使用程序辅助方式来帮助大模型更好地理解问题[16]，并生成更准确和有逻辑的回答。但是他们对问题的描述都是具体的，而且只是人为的赋予了大语言模型隐式推理的能力，对一些理论逻辑性强的问题有较强的提升效果，但仍然存在事实幻想[17]，给出的推理路径仍然会存在错误；相对而言，ReAct 框架中提出的推理加行动更适合一些自然的场景，简单来说就是让大语言模型将“心里话”说出来，向人一样去思考，并把思考得到的结果转化为行动，以达到最终的目的[18]。

而本文的实验更偏向于自然语言，旨在自然语言的理解[19]，大模型在生成回答之前需要先判断用户的意图，然后才能做出正确的选择，例如，“将红色的方块放到绿色的篮子里”，在这简单的一句话中，大模型首先需要自行判断用户这段话的意图所指，然后才能有接下来的行动，在这段话中，大模型应该理解的用户的意图是红色的方块，需要完成的指令是将红色的方块放到绿色的篮子里，之后开始完成用户指令。

本文将解决问题的能力完全赋予给大模型，在微调的数据集里加入的是如何将用户意图转化成不同子任务的示例来供大模型学习，这样大模型就可以集分解与处理为一体，在将用户意图分解为子任务的同时对子任务进行处理，构建任务树，将子任务分解的结果映射到函数库，不需要将子任务的结果写入 Prompt 再次发送给大模型来一步步的推理，节省了大量的时间的同时也减轻了用户的工作量。

在上述的例子中“将红色的方块放到绿色的篮子里”，在大模型识别出用户的意图及任务指令之后，将对任务（将红色的方块放到绿色的篮子里）进行分解，根据微调中的分解示例以及给定的 Prompt，大模型将任务分解成以下几个子任务，首先第一步是找到红色方块的位置，第二步是将机械臂移动到红色方块的位置并执行抓取的动作，第三步是找到绿色篮子的位置，第四步是将机械臂抓取的红色篮子移动到放置位置并执行放置操作；根据 Prompt 中给定的代码函数，为每个子任务选择合适的函数来构造相应的代码，当然并不一定每个分解出的子任务都恰好有对应的函数，对于没有找到直接对应函数的子任务，大语言模型会根据任务分解框架对子任务继续进行分解，直到所有子任务都有对应的来自函数的映射为止，最后将所有代码连接在一起形成整个抓取动作序列的代码程序整体。任务分解示意图如图 5 所示。

2.4 任务反思机制

受限于大模型的能力以及任务的复杂度，生成代码程序的可执行率依旧达不到预期目标，实验在相同场景

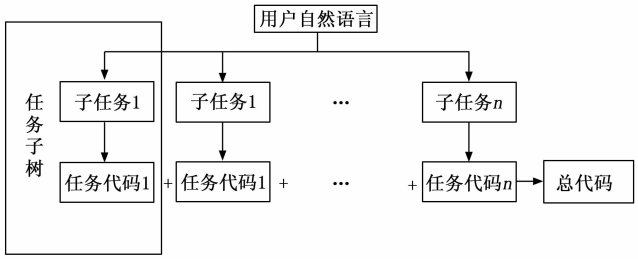

图 5 分解示意图

（场景中包含机械臂，不同形状颜色的物体以及不同颜色的框子）下的不同任务中，实验对不同复杂程度的任务各进行了 10 次抓取实验，发现当所生成的动作序列小于 4 个的时候实验成功率可以达到 90%，当任务的复杂度提升，实验的成功率会跟着降低，当生成序列超过 8 个的时候，成功率下降到 30%，还会犯编写代码时的逻辑错误。

为此，提出了针对机械臂抓取动作生成的任务反思机制，反思机制分为两部分，一部分是子任务与对应的函数的映射关系的反思，另一部分是大语言模型生成程序的反思，本文的反思机制不是根据上下文的记忆进行打分由高到低排列[20]，而是根据大语言模型生成的内容结合用户的输入重新对答案进行反思检查。

对于用户输入的自然语言指令而言，复杂程度由用户自己决定，有一些简单的命令可以直接找到对应的函数，调用即可直接出结果，而有一些则需要复杂的步骤才能得出结果，在进行任务分解的时候，本文尝试构建一套任务－函数－反思机制，在接收到 Prompt 与用户的指令共同组成的输入时，对用户意图进行判断，将用户指令与给定的函数库进行匹配，若能直接找到对应的函数，则直接调用函数完成程序，事实上这是极少数的情况，若找不到，则将用户指令进行分解，尝试按照人类的思考方式，在明确任务的目标之后，将任务按照顺序进行分解，得到若干子任务去匹配函数库，若子函数得不到能够匹配的函数，按照任务分解规则将子任务分解再去匹配函数库，如此反复，当提示中给出的函数库足够全面，子任务都会得到相对应的函数去组成程序，最后将程序应用在机械臂上。

大语言模型受到参数量的限制，在生成的代码上不能做到完全正确[21]，很多时候面对一些复杂的任务不能很好的生成有效的代码，甚至有很多逻辑上的错误，很难一次性就给出正确的答案，而如果每次都需要人为的去挑错误又是一件费时费力的事情，所以本文同样是利用大语言模型，将大语言模型生成的程序代码结合 Prompt 再次发送给大语言模型，让大语言模型自身来完成代码的纠正，实验表明，将大语言模型生成的代码重新发送给大语言模型纠错确实可以有效的纠正一些错

误，但不适合反复的将调整后的代码发给大语言模型，因为大语言模型一旦训练完成，本质上只是一个知识量庞大的数据库，用户输入的每一句话都相当于一个 Prompt，这些内容对大语言模型具有引导作用，反复的将代码输入给大模型可能会使大模型产生怀疑，导致将原本正确的答案变成错误的答案。

3　实验测试与结果分析

3.1　实验测试

本实验在仿真环境下进行测试，测试软件为 CoppeliaSim EDU 版本，CoppeliaSim 是一款非常好用的机器人仿真软件，自带了很多用于机械臂仿真的函数库，在实验中可以充分利用这一点，而且生成的程序易于部署和验证。机械臂选用的是软件自带的 UR5 机械臂，末端安装上 RG2 夹爪，本文实验场景由简易到复杂，一步步来验证大语言模型的生成能力。

实验过程：以“将红色的方块放到绿色的篮子里”为例，本文首先准备实验数据，根据实验要做的抓取测试以及 CoppeliaSim 中 UR5 机械臂的特性，构建实验所用的场景，场景中包括 UR5 机械臂，RG2 夹爪，一个红色小方块以及一个绿色的篮子；实验所用到的函数有识别，移动，抓取等，由于软件自定义的函数参数量太多且多为全局变量，为了更加方便的调用，除了保留了一部分软件自定义的函数之外，利用现有函数自定义了一些函数，功能相同但减少参数量供大语言模型调用。

在实验开始时，首先构建 Prompt，包括所用到的函数以及一些约束等等，再与用户输入共同组成输入，在这个例子中，给出的输入是：现在有一个 UR5 机械臂，末端有 RG2 夹爪，场景中有一个红色的方块(block)，一个绿色的篮子（green _ basket），可供使用的函数有：recognition _ object（objects）＃识别目标位置；move _ to _ pose（objects）＃移动到目标位置；openrg2（）＃打开夹爪；closerg2（）＃关闭夹爪；将红色的方块放到绿色的篮子里。

之后，大语言模型会根据任务分解原则，去匹配能实现功能的函数，如果没有，将任务分解成子任务以匹配给出的函数，为了简化一开始的实验难度，这里所使用的函数是本文根据需要自定义的函数，每个函数里或多或少用到多个基础函数，否则分解过程将更加繁琐；大语言模型这里给出结果是：1. 识别红色方块位置：recognition _ object（objects）；2. 移动机械臂到方块位置：move _ to _ pose（object _ position）；3. 执行抓取：closerg2（）；4. 识别绿色篮子：recognition _ object（objects）；5. 移动机械臂到绿色篮子位置：move _ to _ pose（object _ position）；6. 放下木块：openrg2（）；生成程序如图 6 所示。

```
# 识别红色方块位置
block.position = recognition_object(block)

# 移动机械臂到红色方块位置
move_to_pose(block.position)

# 执行抓取操作
closerg2()

# 识别绿色篮子位置
green_basket_position = recognition_object(green_
basket)

# 移动机械臂到绿色篮子位置
move_to_pose(green_basket_position)

# 放下木块
openrg2()

# 重新识别物体，更新场景信息
objects = recognition_object(objects)
```

图 6　生成程序示例

之后连带着开始的 Prompt 以及生成之后的代码重新作为大语言模型的输入，末尾让大语言模型对刚才的输入进行检查判断，正确则给出正确，不正确则指出问题所在并给出改正后的答案；在这个示例中，由于涉及到的函数较少，逻辑较为简单，所以大语言模型的回复是：这段代码的整体逻辑是正确的，可以实现将红色方块放到绿色篮子中的任务。然而，在代码中存在一些问题和改进的空间：1）函数调用：代码中的函数调用缺少对象参数，应该将 objects 作为参数传递给函数。2）夹爪操作：你可能需要对函数添加对象。这是一个积极的好现象，大语言模型不仅会对程序的逻辑进行思考，而且还会提出质疑帮助用户解决一些潜在的问题。

最后将生成的代码部署到机械臂上，可以看到机械臂完成了这个简单的抓取任务，实验图如图 7～8 所示。

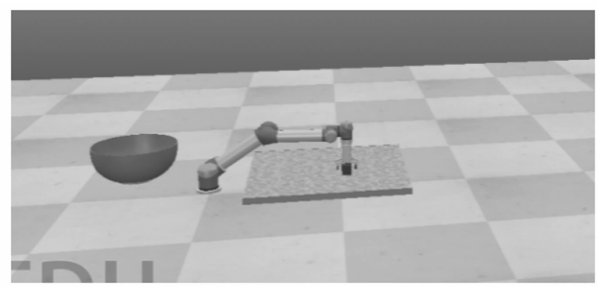

图 7　机械臂抓取方块

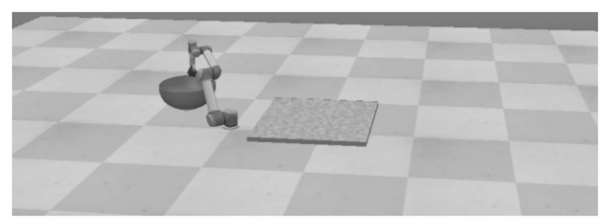

图 8　机械臂抓取方块

在之后的测试中，对实验的布置进行修改，增加了更多种类颜色的物体，不同的抓取策略，以及一些基础

函数，不仅仅是抓取、移动，还有传感器的调用，机械臂运动学函数，避障，路径规划等，逐步模拟真实场景下的抓取。

3.2 结果分析

经过多轮的测试下来发现，在不对大语言模型进行提示约束的情况下，大语言模型生成的结果与预期相差太远，在经过 Prompt 之后，大语言模型开始展现生成能力，对于一些简单的场景，简单的任务，大语言模型能够出色的完成实验的要求，生成的代码可以部署在机械臂上使用，对于一些较为复杂的场景及任务而言，受到生成序列的长度的影响，准确率开始下降，此时对任务进行分解，并尽可能的向提供的函数上靠近，结合人的思维，对函数库中的函数进行充分的分析，例如后面加大了实验的难度，实验场景如图 9 所示，任务是将不同颜色方块以及圆柱放到颜色相同的碗里面，最终得到的生成动作序列是和用户所提供的函数一一对应的，在这一次的生成序列中，其中第 6 个子任务移动到目标碗的上方中，又分为了获取目标碗的坐标-getposition (objectname)，运动学逆解机械臂关节角度-UR5ik (tarpos)，移动到目标位置-movetopos (UR5 _ angle) 等 3 个对应的代码，最终将这些生成的动作序列连接起来，就构成了所需要的生成序列。

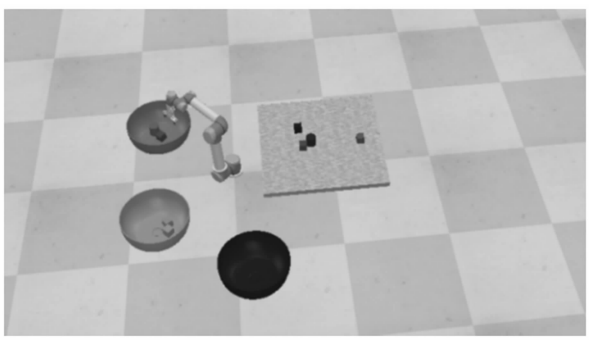

图 9 抓取实验图

对于生成的代码而言，如果每一次生成代码都需要用户自己去检查费时费力，此时将大语言模型生成的代码重新作为输入反馈给大语言模型，由大语言模型重新对任务要求进行审查，在明确任务要求的前提下，对代码进行反思检查，在上述的案例中发现第一次生成的代码中直接使用了碗的颜色而没有经过识别，这一点是在大语言模型的检查下发现的，其描述如下：上述代码可能存在一些问题，因为它在循环中使用了相同的移动到碗上方的位置的步骤 (bowl _ pos)，而没有根据不同的碗来调整目标位置。这可能导致无法正确将物体放置到目标碗中。于是在改进之后添加了遍历 3 个碗执行颜色判断的代码来确保实验中小方块或者小圆柱的颜色与碗的颜色保持一致。在经过对任务分解以及对任务的反思以及代码的反思之后，将大语言模型生成的结果部署在机械臂上，可以完成对任务的要求。

之后实验又分别对不同场景以及不同的任务分别进行了 10 次的实验，受限于实验场景以及大语言模型的参数，本文的生成序列长度没有超过 13，在生成动作序列 n 小于 4 的情况下，实验完成的成功率为 100%，生成动作序列在 4～8 之间的平均成功率为 86%，当生成动作序列大于 8 时，实验的平均成功率为 66%。生成序列长度与平均成功率如表 1 所示。

表 1 不同长度序列实验成功率

序列数量	$n<4$	$4<=n<=8$	$n>8$
平均成功率/%	100	86	66

实验中同样发现，在一些复杂的任务中，当使用的函数数量过多、参数量过多，且包含大量基础函数的情况下，大语言模型生成的结果就很不理想，而且在编写 Prompt 时耗费的时间也更多，因此把一些常用的功能用基础的函数封装成新的函数不仅可以减少用户编写 Prompt 的工作量，而且条理会更加的清晰，也更方便大语言模型的理解，对于特定的领域下的任务而言，任务基本上含有高度的相似性，所以用户自定义函数可以适用于本领域下的多个任务需求，在条件允许的情况下，用户在初始时封装一定功能的函数可以大大的提高大语言模型生成的准确率，而且后期不需要太多的调整。

4 结束语

在传统机器人技术中，深度学习模型是在为特定任务量身定制的小型数据集上进行训练的，这限制了它们在各种应用中的适应性；而大语言模型具有很强的泛化能力，在一定情况下甚至具有为训练数据中不存在的问题找到零样本解决方案的紧急能力；在机器人领域中，机器人需要这样一个强力的大脑来进行判断和决策，以解决可能遇到的各种问题，然而，如何让大语言模型生成可用的信息来帮助用户解放双手仍然是一个具有挑战性的难题。因此，本文提出了一种基于大语言模型的机械臂抓取动作序列的自动生成方法，通过将任务分解成子任务，与用户定义的函数库中的函数进行匹配，然后大语言模型根据生成的动作序列与函数之间的连接，结合任务特性构造出程序主体部分，同时，引入了反思机制，包括对任务分解部分的反思，是否可以找到相对应的函数来完成这部分任务，对生成代码的反思，同样借用大语言模型的能力来对代码的整体逻辑以及错误进行检查优化，以提高可靠性和准确率，实验表明了这种反思机制的有效性，使最终得以部署的代码更加符合预期效果。

通过实验结果显示，本文提出的方法在机械臂控制领域具有一定的应用价值，能够减少开发人员的工作量；但是，仍然还是有一些局限性以及可以改进的空间，第一点局限性是指承载大语言模型需要较高的硬件支持，小型的嵌入式设备无法承载；第二点是生成内容的质量一方面受到 Prompt 的影响，另一方面还受限于大语言模型参数量以及质量的影响，在解决长期规划任务的能力较弱，因此除了一开始使用数据集对模型进行微调之外，将实验过程中的错误收集起来做成数据集再对大语言模型进行微调或许是一个好的办法。

在未来，仍然可以继续探索更高级的任务分解和子任务匹配方法，以实现更复杂和多样化的机械臂任务的自动生成；还可以与深度强化学习等 AI 技术相结合，来实现更智能化的自我学习以及判别；而且，随着人工智能技术的发展，可以选择性能更加优秀的模型以及融合多种模型，实现 AIGC 的融合应用。此外，需要一个将生成代码与机械臂控制系统相结合的执行框架，用以自主在仿真环境或者真实场景中验证方法的可行性和实用性。

参考文献：

[1] TAMIZI M G, YAGHOUBI M, NAJJARAN H. A review of recent trend in motion planning of industrial robots [J]. International Journal of Intelligent Robotics and Applications, 2023, 7 (2): 253 - 274.

[2] HASAN M, OZEL C, POTTER S, et al. SAPIEN: affective virtual agents powered by large language models [C] //2023 11th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW). IEEE, 2023: 1 - 3.

[3] BROHAN A, CHEBOTAR Y, FINN C, et al. Do as i can, not as i say: Grounding language in robotic affordances [C] //Conference on Robot Learning. PMLR, 2023: 287 - 318.

[4] WU Y, JIANG A Q, LI W, et al. Autoformalization with large language models [J]. Advances in Neural Information Processing Systems, 2022, 35: 32353 - 32368.

[5] LIN K, AGIA C, MIGIMATSU T, et al. Text2motion: From natural language instructions to feasible plans [J]. Autonomous Robots, 2023, 47 (8): 1345 - 1365.

[6] SHEN L, SHEN E, LUO Y, et al. Towards natural language interfaces for data visualization: A survey [J]. IEEE transactions on visualization and computer graphics, 2022, 29 (6): 3121 - 3144.

[7] LIU Z, BAHETY A, SONG S. REFLECT: summarizing robot experiences for failure explanation and correction [C] //Conference on Robot Learning. PMLR, 2023: 3468 - 3484.

[8] DU Z, QIAN Y, LIU X, et al. GLM: general language model pretraining with autoregressive blank infilling [C] //Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 2022: 320 - 335.

[9] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners [J]. Advances in neural information processing systems, 2020, 33: 1877 - 1901.

[10] HUE J, SHEN Y, WALLIS P, et al. LoRA: Low-Rank adaptation of large language models [DB/OL]. [2021-10-16]. https: //doi.org/10.48550/arXiv.2106.09685.

[11] LIU P, YUAN W, FU J, et al. Pre-train, Prompt, and predict: A systematic survey of Prompting methods in natural language processing [J]. ACM Computing Surveys, 2023, 55 (9): 1 - 35.

[12] GU Y, HAN X, LIU Z, et al. PPT: Pre-trained prompt tuning for Few-shot learning [C] //Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022: 8410 - 8423.

[13] VEMPRALA S, BONATTI R, BUCKER A, et al. ChatGPT for robotics: Design principles and model abilities [DB/OL]. [2023-1-19]. https: //doi.org/10.48550/arXiv.2306.17582.

[14] LI Y, CHOI D, CHUANGJ, et al. Competition-level code generation with alphacode [J]. Science, 2022, 378 (6624): 1092 - 1097.

[15] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought Prompting elicits reasoning in large language models [J]. Advances in Neural Information Processing Systems, 2022, 35: 24824 - 248371.

[16] GAO L, MADAAN A, ZHOU S, et al. Pal: Program-aided language models [C] //International Conference on Machine Learning. PMLR, 2023: 10764 - 10799.

[17] ZHAO W X, ZHOU K, LI J, et al. A survey of large language models [DB/OL]. [2023-11-27]. https://doi.org/10.48550/arXiv.2303.18223.

[18] YAO S, ZHAO J, YU D, et al. ReAct: synergizing reasoning and acting in language models [DB/OL]. [2023-3-10]. https: //doi.org/10.48550/arXiv.2210.03629.

[19] BRANDIZZI N. Toward more Human-Like AI communication: A review of emergent communication research [J]. IEEE Access, 2023, 11: 142317 - 142340.

[20] PARK J S, O'BRIEN J, CAI C J, et al. Generative agents: Interactive simulacra of human behavior [C] //Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023: 1 - 22.

[21] OUYANG L, WU J, JIANG X, et al. Training language models to follow instructions with human feedback [J]. Advances in Neural Information Processing Systems, 2022, 35: 27730 - 27744.