6

Natural Language Conditioned Planning of Complex Robotics Tasks

Toms Eduards Zinars, Oskars Vismanis, Peteris Racinskis, Janis Arents, and Modris Greitans

Institute of Electronics and Computer Science, Latvia

Abstract

As natural language processing advances in the field of robotics, enabling seamless human-robot interaction, it becomes imperative to identify the most effective approach for conditioning complex robotics tasks using natural language commands. This article reviews various state-of-the-art methods for natural language-conditioned planning, with a particular focus on mobile manipulation. The authors explore and review different architectures and techniques to comprehend, interpret, and execute natural language commands. Challenges are identified along the way, and conceptual architecture is proposed to tackle them in an efficient manner.

Keywords: natural language processing, mobile manipulation, action primitives, edge AI.

6.1 Introduction

Everyday interactions between people are usually performed in a very casual manner through natural language, or NL, as it is an comfortable way of communication. This has been carried over to our appliances, such as phones and cars, with the use of voice commands. It logically follows that robotic assistants, be they industrial or service, which operate in an environment with people of various machinery handling skill levels, would benefit from a user interface utilizing Natural Language Processing (NLP) techniques to process unstructured input. As the environments in which these systems get deployed are diverse and varied, a general understanding of language and its relation to objects is crucial to a rational implementation.

Large Language Models (LLM) [1][2] have shown to be very capable of tackling the task of inferring NL inputs for commanding a robotic agent. They can be made to operate with multi-modal information [3], but they often rely on cloud service models that require immense computing resource [4][5][6]. The possibility exists to use smaller models that can be run locally [7][2]. Robotic systems are typically specialized to operate under certain conditions, and a trained technician must perform any new adaptations. However, as real-life environments are usually unique in their layout and the objects they contain, it can be hard to predict what the robotic agent needs to know. In section 1.2, we do a quick overview of NL processing, LLM and multi-modal embeddings and their recent implementations and uses in planning for robotic systems.

To function in such environments, a more general approach for representing robot tasks can be used, as, fundamentally, a robot system can be thought of as a handful of basic operations, but they are usually very task-dependant. One such approach is using action primitives. They provide the high-level planner with an abstract, symbolic representation of available actions so a task plan can be made. Another role they fulfill is low-level planning, a set of functions that can be adapted to specific scenarios and reused for different tasks [8]. Details about action primitives, their synthesis and implementations are described in section 1.3.

The primary type of robot control we address with the approach outlined in this chapter is mobile manipulation, which can be described as the joint control of a mobile base and a manipulator arm. The overall workflow of such a system can be generalized to receiving, planning and executing a task. In our conceptual architecture, the function of receiving a task is done with NL commands. For this, a two-stage technique is proposed. A high-level LLM-based algorithm for mapping NL commands into a constrained space of action primitives, and a library of action primitives at the low level, which is further elaborated on in section 1.5. This chapter also includes section 1.4, where we provide a brief overview of identified challenges and issues regarding both the practical use of NLP and the implementation of Action Primitives, while section 1.6 concludes the chapter with our views on future developments.

6.2 Natural Language Processing for Robotics

Natural language is vast and vague, so much so that people often have problems understanding each other. An NL command can consist of a request that only makes sense in the context of the situation and the environment. It is not enough for a robotic system to just deduce what is being asked of it but also to be able to ground this information in the environment it finds itself in and act upon through a concise and safe plan of actions.

The current state-of-the-art performance in NLP can be found in language models based on transformer architecture [9], specifically the LLMs. Starting at 10 billion, typically having 100-300 and with a couple exceeding 1 trillion, these LLMs have demonstrated impressive performance across various language-related tasks [1][10].

6.2.1 Large language models

Large Language Models are powerful natural language interpreters [11]. An important quality they share with the smaller models is multitask learning – the ability for one model to become relatively proficient at several different tasks but masters of none [12]. However, as their size grows and performance increases [13], these models begin to overtake previous-generation specialist fine tuned models [10][11].

Training large language models from scratch is a very expensive process in terms of energy, computing and time, typically requiring massive clusters of dedicated high-end hardware that train them non-stop for several weeks [2]. As this is only feasible for large tech corporations such as Google, Meta, OpenAI, Amazon and Huawei [2], a pre-trained model can be specialized for a downstream task through fine-tuning - using a specialized dataset to introduce into the model specific knowledge or teach it new operations all together [14], making them more accessible for specific tasks.

When the parameter count reaches into the tens and hundreds of billions, the language models begin to exhibit new qualities that are not present in their smaller counterparts, referred to as emergent abilities [1][15], three prominent ones, as highlighted in [1], are:

- In-context learning [11]- the ability to perform tasks not part of the training corpus, based on an instruction and several input-output examples(few-shot) provided in the prompt;
- The instruction following [14] relies on fine-tuning a model using a dataset containing natural language instructions, which results in

improved zero-shot (no example given) prompting performance for unseen tasks;
- Step-by-step [16] improves the model's complex reasoning by leveraging chain of thought prompting by adding step-by-step instructions to the examples for a few-shot task prompt.

As fine-tuning takes computing resources and time, many pre-trained models are fully capable of being used out of the box [11]. By carefully structuring the input prompts, it is often possible to condition the model to provide the desired output for the task at hand [17], an approach known as prompt engineering [1].

As a way to improve the fine-tuning process, a lot of work has been done in developing various parameter-efficient fine-tuning methods (PEFTs) [18] that consider fine-tuning only parts of the overall model. Instead of having several different fine-tuned copies of the same LLM weights, using methods such as Low-Rank Adapters (LoRA) [19] one can have a single instance of an LLM and then simply apply the corresponding fine-tuned adaptation, saving on space and compute resource.

Running the LLM inference process, even on the smaller models, requires capable hardware, primarily GPUs with sufficient VRAM [7]. The model's parameters are typically stored in a 16-bit float format (FP16), translating to roughly 2GBs for every 1 billion parameters. A remedy for this issue is model quantization, a method where the parameters of the model are converted into smaller 8-, 4-, 3- and even 2-bit formats [20], which (plural) can reduce the required VRAM down from 14GB to roughly 4 GBs for a 7B model (7B representing 7 billion parameters) when using 4-bit quantization, with marginal loss to performance [7]. The requirements can be further reduced by using methods that share inference between CPU and GPUs [21], which provides perspectives for application in edge AI.

As language models have been trained on general data such as textual information sourced from books or the internet and/or coding languages [1][2][22], they gain a broad internal knowledge base that can be leveraged to create a human-machine interface capable of decoding obscure NL requests into actions understandable to a robotic agent system [4][17].

6.2.2 Multi-modal embeddings

The recent progress in autoregressive and sequence-to-sequence NLP processing with LLMs has enabled a number of related advances. In particular, the abstract vector nature of the tokens being processed by transformers has been exploited to create mappings between radically different data modalities. CLIP [3], short for Contrastive Language-Image Pre-training, is a notable example. It jointly trains an image classifier and text encoder on image-caption pairs. Each model outputs a vector in the same latent space. The cosine similarity of vectors corresponding to matching image-caption pairs is maximized, while that between all others is minimized. The result is a pair of models capable of mapping the greatly dissimilar image and text input spaces to a common latent "concept" space.

CLIP and similar systems have since been commonly described as vision-language models (VLMs). Subsequent work, such as *LSeg* [23] and *ConceptFusion* [24], has been done to extend the vision model in a VLM to produce segmentation maps – embeddings for each image in a pixel. These have subsequently found use in robotics, particularly in creating maps amenable to natural language queries. For example, in [25], a 2-dimensional grid map is constructed using LSeg and depth imagery, which can then be used to find navigation goals using text prompts. ConceptFusion [24] expands upon the mapping problem, producing 3-dimensional embedding-tagged point clouds. Some approaches do away with explicit maps entirely, instead using a Neural Radiance Field to predict the embedding associated with any point in the environment directly [26].

The ability of transformer models to map between and autoregressively generate sequences of arbitrary vectors has been directly exploited for robot control in works such as [27], where robot actions are predicted directly from text prompts and images of the scene in which the robot should operate. The inputs need not be limited to a single type of embedding — in [28] and [29], a large transformer is trained to operate on input sequences containing multiple types of embeddings — such as VLM tokens, robot state encodings, scene representations and past actions — with PaLM-E in [29] being directly based on a pre-trained LLM.

6.2.3 Recent implementations of high-level planning for mobile manipulation

The practical implementation of language models for use in high-level planning of mobile manipulator systems has taken various approaches. Some approaches use the language models to extract language features that are further passed into more specialized modules for processing [30][31], others use the language models as active elements of the planning process [5][6][32], and others yet use the models for low-level planning [33]. Some models perform their own mapping of the environment through computer vision [30][34], but it would seem that map integration is an underutilized solution, though some works are exploring combining embeddings from the language model with embeddings stored in a semantic map [6].

Many of the highlighted works rely on prompting and using pre-trained models [4][5][6][32][33][35]. The importance of proper prompting technique is explored in [19], which presents a method for selecting and formatting prompts to elicit outputs usable in robotic systems. They define a starting prompt that describes the role the LLM is supposed to play and condition it to respond only when directly prompted to by a specific keyword. That is followed up with a sequence of instructions, explanations and templates that describe the desired output format and contents. The prompt is finished up by providing several examples of how the output should look. The ability to provide NL feedback to improve and correct mistakes during inference is also showcased.

While not an example of a natural language command, ProgPrompt [32] takes an input prompt of Python code containing imports of action primitives, a list of available objects, example tasks and the start of the desired operation. The LLM then returns a generated plan in Python that uses assertions to ensure a feedback loop once the agent encounters variables in the environment and can successfully execute the appropriate action.

Lang2LTL [6] utilizes a modular system where LLMs are used to perform several subtasks in the interest of generating a grounded relation to objects and places that the system stores in a database. One module is tasked with extracting place names from the request prompt. These extracted names are then compared to the database objects through embedding cosine similarity, and then generalizing the input request with substitutions and passing it through a fine-tuned LLM symbolic translator that generates the LTL formula, finished by inserting the found database objects in their respective substitution locations.

Text2Motion [5] utilizes an LLM model that performs task planning in conjunction with geometric feasibility planning that evaluates if the plan generated by the LLM is valid or not. They evaluated planning the whole sequence of actions and then validating it, planning and validating each individual step of the sequence and a hybrid system that tries creating a full plan, falls back to individual step in case of a failure, then tries finishing the plan fully again. For implementation, they rely on OpenAI's GPT series [36] and execute their system through prompt engineering.

Language to Rewards [33] employs a two-stage LLM setup in the form of a Reward Translator, where the first LLM (Motion descriptor) is used to translate the input sentence into a structured natural language instruction. The second LLM (Reward Coder) then generates a usable code in the form of reward functions that can be passed directly to a low-level motion controller, skipping the use of action primitives altogether. Both LLMs are conditioned by leveraging in-context learning by prompts. The first one contains templates to use when creating the task description, while the second is prompted with a general program description. Both prompts contain a list of instructions to guide inference to the desired result. While this approach doesn't perform long-horizon tasks, it does showcase the possibility of using LLMs for low-level planning to some degree.

Say-Can [3] explored the issue of grounding an LLM planning system in the real world, as without any feedback elements or information about the current environment, the language model can propose logical but contextually impossible solutions such as suggesting using a vacuum cleaner when one isn't available. They achieved this by using a two-part system - the LLM provides probabilities for action relevance to the given task, while a *value function* provides probabilities of how likely it is to succeed in doing specific actions. The multiplication of these two values is chosen as the action for the plan to perform.

Inner Monologue [35] explores using feedback mechanisms to improve task completion. By being able to receive information from the environment in the form of language input, such as sensory data about detected objects or whether the planned action was successful or not, the agent can attempt to perform the action again or replan, whereas without such feedback the agent would fail the task outright. When the agent is met with an ambiguous situation, such as a request for "a drink", by asking the user for clarification, it can form a dialogue that helps execute the task more successfully. They also question if the answering could be done by another LLM as well.

There are doubts by some if LLMs are reliable enough to be used for planning operations [31][37] but do recognize their utility as language interpreters. One such implementation is proposed in [31] with LLM+P, a language model coupled with a classical planner. A classical planner provides proven ability in task solving, while the LLM can provide its understanding of language to be able to interpret a large amount of tasks and then translate them to a structured planning language such as PDDL [47].

Language is only one part of a system meant to operate in an environment. The ALFRED benchmark was introduced in 2020 as a way to test agent systems that use both natural language instructions and ego-centric vision [38]. While not the first, it did combine several functions to create a benchmark that tests proposed systems in a non-reversible, partially observable environment. Models are tasked with solving basic household tasks within a limited number of actions, which typically involve moving and modifying objects using tools or special locations, requiring a specific sequence of actions to execute. Many models also build a map representation of the environment [30][34]. The baseline model relied on an LSTM (long-short-term memory, predecessor to the transformer architecture) based language model that managed to achieve only a 0.4% success rate in the unseen tests [38]. Later attempts would implement PLMs such as BERT [39] and improve the success rate to 50% [34]. At the time of writing, the best-performing models that have available materials are Prompter [30] and CAPEAM [34], both utilizing BERT for their language processing. BERT is an older language model (from late 2018) with sub-1 billion parameters, far from state-of-the-art in language models, making a direct comparison hard as LLM-based systems seem to rely on their own evaluation methods [3].

Prompter [30] utilizes its language model in a semantic search module, using natural relations between objects (apples found in kitchens, toothbrushes in bathrooms), speeding up the search. The benefits of using a language model for such a role is leveraging its inherent knowledge of language to determine word relations, whereas previous methods relied on using additional training. For its vision substream, based on [39], the agent creates a 2D top-down semantic map from images it received through its ego-centric vision, which is a 2D RGB image that is processed into a depth map and segmented to create masks.

CAPEAM [34] uses a fine-tuned BERT implementation for predicting what predefined role fits what words from the input sentence. Context Aware Planning module uses a sub-goal planning element that first finds a general template to use based on input requests and then a so-called "detailed planner" to insert the contextual information into predefined places. Environment Aware Memory is responsible for vision operations, utilizing memorization for object locations as well as saving previous segmentation masks, as it was found it helps to identify objects that become obscured through later actions.

6.3 Action Primitives for Mobile Manipulation

In [40], the authors define the concept of *situated robotics*, which describes robotic systems in complex, dynamic or, in other words, *situated* environments. The amount of an environment's situatedness directly affects the complexity of the robot control system and its need to adapt to new situations, which shows that the more complex the environment, the more complex the control system needs to be. Derived from different action definitions, different approaches to robot control exist, such as reactive control, deliberative control, hybrid control (a combination of the first two), and behaviour-based control [40]. Reactive Controls can be referred to as an IF...THEN rule interpreter while Deliberative control - as functioning in a higher level. Behaviour-based control, on the other hand, functions a bit differently; at its core is the concept of behaviours, which are functionalities varying in complexity that get activated depending on their predefined inputs, which can be sensory data or other behaviours, and they output control commands for actuators or other behaviours. Where the other control type's lack in either computation efficiency or complexity, the behaviour-based control can manage a combination of the two, that is easier to engineer and upgrade than hybrid control [40]. From the concept of behaviours, abstract or primitive behaviour can be derived, which is a more general function made to be reusable in different scenarios. These are often referred to in the literature as the **action primitives** (aka manipulation primitives, task primitives, skills etc.).

To plan and execute tasks in situated environments, some form of Task and Motion Planning (TAMP) is usually required, as seen in [4][41][42][43], and for that, it is best to have representations of the environment and available robot actions. As mentioned before, dynamic environments can have many different actions fulfilled in them and engineering all of them can be a time-consuming process [if even possible]. A better approach might be to use said **action primitives**, which would be task-specific only in the execution phase, depending on sensory inputs. This representation of behaviours allows for a more general form of activation conditions as these are usually atomic functions that do only one thing, but not in a way feedback control would be managed (for example, move by a certain angle) [8][44][40]. The granularity of primitives depends on the usage, but in robotics, it is usually a control command to make a robot move. As action primitives are usually computationally light, one system can be made to work for different tasks.

Primitives can be divided by their usage, the simplest form being the primitive itself. After primitives come actions, and after that - activities [44].

Taking as an example a robot doing picking, that could be considered an action, which would be made up of multiple primitives, in this case, moving to the object and closing the gripper. The case of putting multiple actions in a sequence, such as picking and placing, could be referred to as an activity of moving an object. This way of referencing actions is especially useful for task planning, as the symbolic representation of a task does not always include all the steps needed to complete it [41]. If, for example, the task of picking up an object and placing it somewhere is given, the task planner does not need to think about the specific primitives needed, such as moving the arm to the specific spot and closing the gripper, it just needs to make a sequence of actions, that fulfil the task. The primitives are then left for the motion planner to check geometrically if the task plan is feasible. Many systems would then use a so-called action library or a set of skills [41][42][45] that can be used by the task planner to know what the system is capable of and use it when making plans.

Symbolically, actions and the action-state relations can be predefined using task planning languages such as STRIPS [46] or its successor, PDDL [47], or it can be done during the planning process using Large Language Model (LLM) prompt engineering [48]. Regarding the environment, the representation can be about locations, objects, etc. Actions refer to what tools the robot has at its disposal, in other words, what it can do to accomplish its tasks.

6.3.1 Methods for creating primitives

There can be many kinds of primitives made for specific applications, which means that creating them is a process of its own, and there are different ways of synthesis. For example, Jeon et al. [41] create a service robot application, which utilizes an action library in which one of the actions is *hold_object*. This action can be decomposed into two action primitives – *approach_object* and *close_hand*. For task planning purposes, actions, action primitives and their interrelations are represented using the PDDL language. This allows them to be used with PDDL-supported planners. In addition, it predefines the action's preconditions, effects, requirements, etc. This method deeply relies on the engineer's capabilities and understanding of the tasks they're making the primitives for. Also, in the case of wanting to add new functionality, it can be a long process, depending on the complexity of the task.

Action primitives can also be extracted from human motion via imitation learning (IL) [49]. In [50], manually segmented human motion capture data is used with a spatio-temporal non-linear dimension reduction technique to cluster similar segments of motion into generalized primitives. Similarly, in [43], imitation learning is used, in terms of behavioural cloning (BC), to learn action primitives, but their combinations into actions are then learned with reinforcement learning (RL). With these methods, the system is able to do the task of pouring cereal into a bowl. In [51], however, the stereotypical motions of a human picking up a cup are recorded and used as a basis for actions, though in the form of dynamic movement primitives (DMP).

A different approach to creating primitives is only making them when needed. Gizzi et al. [42] look at using action primitives as a way of creative problem-solving. They use the definition of a MacGyver problem [52], which describes an environment that has everything necessary for successful task execution, and the robot has all the tools it needs. Still, the specific approach it must take is unknown. The system begins with a set of predefined actions and is tasked to do an indirect task, such as reaching an obstructed object or location. Whenever met with a situation where the robot cannot fulfil its task with the actions it knows, it starts generating combinations of available objects and interactions with them using the available actions. The environment and actions are described using PDDL. In this case, the actions available are *obtain_object* and *press_button*, and the robot is tasked with reaching an object on the other side of a wall. There are also buttons present. When the initial *obtain_object* fails, it starts generating different combinations of actions and tries executing the feasible ones one by one until it manages to hold down one of the buttons to move the wall.

In the case of service robots or any robots that might work in environments simultaneously with humans, there is also a positive effect when creating the primitives to function similarly to human motion, as humans can better understand them from the point of predictability. This approach also allows for easier training using imitation learning [51].

6.3.2 Action primitive implementations

In relation to mobile manipulation or manipulation in general, the action primitive has been used consistently in systems that do more general or environment-adaptive tasks. In [42], action primitives are used as a basis for finding solutions to given problems in the sense of finding new primitives that would be applicable to the situation.

In [53], such action primitives as *pulling*, *pushing*, *grasping* and *pivoting* are included. A dual-arm robot ABB YuMi with custom tactile sensors is used. With the predefined action library and tactile feedback combination, the system is able to do dexterous manipulation based on robot/object interaction plans. This same setup is used in [54] to manipulate rigid objects based on pointcloud data using long-horizon planning. The planning problem is defined as an action primitive sequencing, where the symbolic representation of actions as action primitives allows for the planner to set aside the reasoning about robot-object dynamics.

For service robots, action primitives are used to create and execute plans for such tasks as pouring cereal [43] or juice [41]. In [43], basic action primitives are learned and then combined into such actions as *pick*, *bowl* (picking an object and placing it in a bowl) and *breakfast* (moving objects in a pouring manner directed toward the bowl). For similar tasks, [41] uses such primitives as *approach_object, close_hand, and move_arm*. These are then used in plans to accomplish tasks like moving an object out of the way and then moving a package in a pouring motion.

6.4 Identified Challenges

Robotic agents designed to work alongside people are under great scrutiny, as such systems must be, first and foremost, safe, adequately efficient and easy to use. Such systems have to be robust, with no room for ambiguity. Yet, the datasets that language models are trained on come with biases that the model inherits and these biases can affect the inference process and result in seemingly random erroneous outputs [1][37] or repetitive cycles [5]. There is also a general risk of hallucination from LLM providing absurd plans to the robotic agent system or failing to generate anything at all.

Many promising implementations rely on using the closed-source GPT series for their research [17][5][6][33]. While these models are state-of-the-art and the main target for other models to beat [22], a real implementation for any mobile agent lacks practical autonomy if it must have a constant connection to a cloud service. If it is true that certain traits of language models are limited to the larger models and begin appearing only at 100B+ parameters [1][15], even with quantization, these larger models would require powerful GPUs that are impractical to be placed onto a physical agent to ensure autonomous operation, limiting the system to require a local network where a remote resource provides inference to the system. This impedes taking advantage of the LLMs emergent abilities for edge AI applications.

One of the main focuses of NL commanded mobile manipulation systems is higher autonomy, but they often still lack in their abilities. For example, a clear bottleneck of such systems is the range and capabilities of the skills they possess as [4], even if the NL command can be translated to the system correctly, it cannot execute something it does not know how.

Another challenge, directly involving manipulation, can be addressed in the form of object recognition and grasping. A real environment usually has many different types of objects, and a robot operating in such an environment would be expected to be able to manipulate any of them. Modern approaches manage such a problem using 3D model databases from the internet [51] or pre-trained vision-language models [55] that can recognize previously unseen objects. Grasping them is then dependent on grasp point recognition [56][57] and the quality of active primitives used.

6.5 Conceptual Architecture

Based on what has been covered, we have considered various implementations for robotics systems with natural language commands. There is no one definitive architecture (as can be seen by examples in chapter 6.2.3), given differing sizes, scopes and environments in which such agents are expected to operate. The underlying principles we believe such a system would need to have are the need to process the language input, a way to locate itself and objects in the environment, plan its actions and finally execute the plans at the physical level, with a feedback system to account for changing factors in the environment or the user's request.

In Figure 6.1 we present a simplified interpretation of how such a system could be structured in two distinct main modules or levels – the language processing module and the execution module. Other works have also approached the problem with some type of two-part design [4][34]. What may seem missing from this schematic is a dedicated planning module, but here it is understood that high-level or task planning is done by the NLP module, whereas low-level planning or motion planning is done in the execution module.

At the core of the NLP module would be an LLM that could be run locally, at least on a cluster, but which one fits this use case needs to be investigated. The module's task is to interpret the user's input request in a way that the rest of the system may act upon it. For successful communication between the levels, the execution module provides the NLP module with a description of what it can do and how the language module output needs to be formatted, allowing for new capabilities to be introduced to and utilized by the system

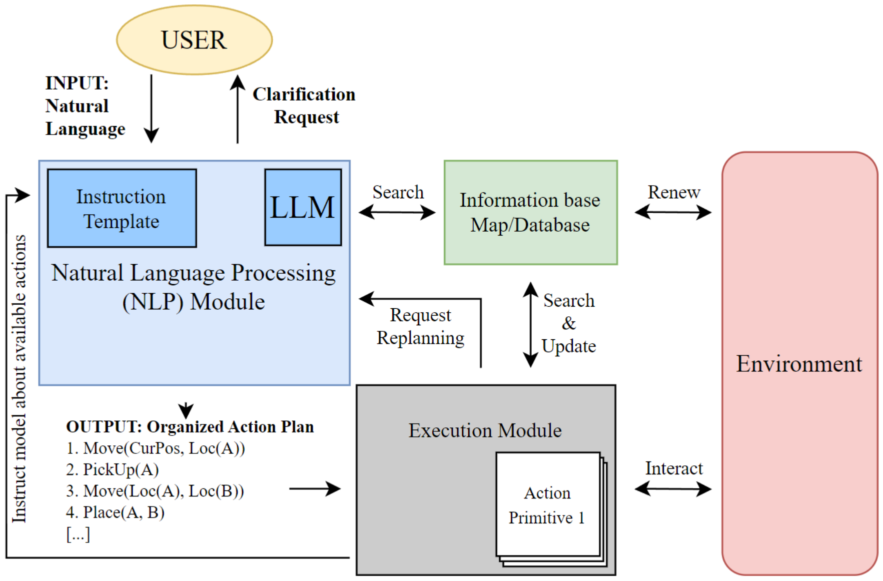

Figure 6.1 Proposed Mobile Manipulator Control System

that were not originally accounted for. An interface with a map (or database) grounds the language module in its current environment and permits the NLP module access to task-relevant information. It would be preferable as well for the NLP module to be able to take received feedback from the execution module and present it to the user for clarification.

The execution module consists of an action library, which is a set of action primitives the agent can execute to accomplish tasks. The module is responsible for physical execution, so motion planning is also done here. Whenever a task plan is received, this module checks its geometrical feasibility and, in case of a failure, requests for a replanning. Once a feasible plan is made, the module executes the task sequence.

The main benefits of such a layout is the idea that the information about the environment is contained within the semantic map, while the LLM possesses general linguistic knowledge. As action primitives should be the same regardless of where it is, by combining information extracted from the input with information available on the map, the system is not hard-coded to a particular place. By relying on a smaller and locally run LLM instance, the hope is to ensure the ability of the system to operate successfully in an edge AI use case.

6.6 Conclusions and Outlook

Natural language application in robotics is an ever more relevant field of research and development. The rise of LLMs has made applying general language understanding to computer systems seem deceptively trivial, but there is still much ambiguity to overcome. We hope to see the field develop in both directions - more research done on larger models to see how much they are capable of, as well as more development to bring these high-level abilities down to smaller and smaller model sizes to enable true edge AI applications.

When working in complex environments, action primitives can be used as a powerful tool to generalize actions available to a robotic system. This can be useful both with task and motion planning as they allow for these two processes to be less intertwined without affecting their effectiveness. There are different approaches to creating action primitives, and the future seems to be headed towards automated synthesis with different machine-learning techniques. This chapter proposes a conceptual architecture for NL-commanded mobile manipulation, consisting of an NLP module for command interpreting and high-level planning and an execution module that utilizes action primitives for low-level planning and execution.

Acknowledgements

This research was conducted as part of the EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, which has received funding from KDT JU under grant agreement No 101097300. The KDT JU receives support from the European Union's Horizon Europe research and innovation program and Austria, Belgium, France, Greece, Italy, Latvia, Luxembourg, Netherlands, and Norway.

References

[1] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J., & Wen, J. (2023). A Survey of Large Language Models. *ArXiv, abs/2303.18223.*

[2] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., & Mian, A. S. (2023). A Comprehensive Overview of Large Language Models. *ArXiv, abs/2307.06435.*

[3] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision." International Conference on Machine Learning (2021).

[4] Ahn M., Brohan A., Brown N., Chebotar Y., Cortes O., David B., Finn C., Gopalakrishnan K., Hausman K., Herzog A., Ho D., Hsu J., Ibarz J., Ichter B., Irpan A., Jang E., Ruano R. J., Jeffrey K., Jesmonth S., Joshi N. J., Julian R. C., Kalashnikov D., Kuang Y., Lee K.-H., Levine S., Lu Y., Luu L., Parada C., Pastor P., Quiambao J., Rao K., Rettinghouse J., Reyes D. M., Sermanet P., Sievers N., Tan C., Toshev A., Vanhoucke V., Xia F., Xiao T., Xu P., Xu S., Yan M. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. Conference on Robot Learning, 2022.

[5] Lin, K., Agia, C., Migimatsu, T., Pavone, M., & Bohg, J. (2023). Text2Motion: From Natural Language Instructions to Feasible Plans. *ArXiv, abs/2303.12153*.

[6] Liu, J., Yang, Z., Idrees, I., Liang, S., Schornstein, B., Tellex, S., & Shah, A. (2023). Lang2LTL: Translating Natural Language Commands to Temporal Robot Task Specification. *ArXiv, abs/2302.11649*.

[7] Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *ArXiv, abs/2210.17323*.

[8] Schaal S., Ijspeert A., Billard A. Computational approaches to motor learning by imitation. In: Philosophical Transactions of the Royal Society B: Biological Sciences, 2003, 358(1431), 537–547.

[9] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. NIPS.

[10] OpenAI (2023). GPT-4 Technical Report. *ArXiv, abs/2303.08774*.

[11] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language Models are Few-Shot Learners. ArXiv, abs/2005.14165.

[12] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

[13] Kaplan, J., McCHuang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P.R., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., & Ichter, B. (2022). Inner Monologue: Embodied Reasoning through Planning with Language Models. Conference on Robot Learning.andlish, S., Henighan, T. J., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models. ArXiv, abs/2001.08361.

[14] Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2021). Finetuned Language Models Are Zero-Shot Learners. ArXiv, abs/2109.01652.

[15] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent Abilities of Large Language Models. Trans. Mach. Learn. Res., 2022.

[16] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Xia, F., Le, Q., & Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. ArXiv, abs/2201.11903.

[17] Wake, N., Kanehira, A., Sasabuchi, K., Takamatsu, J., & Ikeuchi, K. (2023). ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application. ArXiv, abs/2304.03893.

[18] Lialin, V., Deshpande, V., & Rumshisky, A. (2023). Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. ArXiv, abs/2303.15647.

[19] Hu, J. E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. ArXiv, abs/2106.09685.

[20] Chee, J., Cai, Y., Kuleshov, V., & Sa, C.D. (2023). QuIP: 2-Bit Quantization of Large Language Models With Guarantees. ArXiv, abs/2307.13304.

[21] Gerganov, G. Port of Facebook's LLaMA model in C/C++. Available at: https://github.com/ggerganov/llama.cpp [Accessed August 23, 2023]

[22] Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M. P., Ferrer, C.C., Grattafiori, A., Xiong, W., D'efossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., & Synnaeve, G. (2023). Code Llama: Open Foundation Models for Code.

[23] Li, Boyi, Kilian Q. Weinberger, Serge J. Belongie, Vladlen Koltun and René Ranftl. "Language-driven Semantic Segmentation." ArXiv abs/2201.03546 (2022).

[24] Jatavallabhula, Krishna Murthy, Ali Kuwajerwala, Qiao Gu, Mohd. Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Varma Keetha, Ayush Kumar Tewari, Joshua B. Tenenbaum, Celso M. de Melo, M. Krishna, Liam Paull, Florian Shkurti and Antonio Torralba. "ConceptFusion: Open-set Multi-modal 3D Mapping." ArXiv abs/2302.07241 (2023).

[25] Huang, Chen, Oier Mees, Andy Zeng and Wolfram Burgard. "Visual Language Maps for Robot Navigation." 2023 IEEE International Conference on Robotics and Automation (ICRA) (2022): 10608-10615.

[26] Shafiullah, Nur Muhammad (Mahi), Chris Paxton, Lerrel Pinto, Soumith Chintala and Arthur Szlam. "CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory." ArXiv abs/2210.05663 (2022).

[27] Brohan, Anthony, Noah Brown, Justice Carbajal, Yevgen Chebotar, Krzysztof Choromanski, Tianli Ding, Danny Driess, Chelsea Finn, Peter R. Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil J. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Sergey Levine, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Pierre Sermanet, Jaspiar Singh, Anika Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Ho Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Ted Xiao, Tianhe Yu and Brianna Zitkovich. "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control." ArXiv abs/2307.15818 (2023).

[28] Reed, Scott, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley D. Edwards, Nicolas Manfred Otto Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar and Nando de Freitas. "A Generalist Agent." Trans. Mach. Learn. Res. 2022 (2022).

[29] Driess, Danny, F. Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Ho Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke,

Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch and Peter R. Florence. "PaLM-E: An Embodied Multi-modal Language Model." International Conference on Machine Learning (2023).

[30] Inoue, Y., & Ohashi, H. (2022). Prompter: Utilizing Large Language Model Prompting for a Data Efficient Embodied Instruction Following. ArXiv, abs/2211.03267.

[31] Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023). LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. ArXiv, abs/2304.11477.

[32] Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2022). ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. 2023 IEEE International Conference on Robotics and Automation (ICRA), 11523-11530.

[33] Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K., Arenas, M. G., Chiang, H.L., Erez, T., Hasenclever, L., Humplik, J., Ichter, B., Xiao, T., Xu, P., Zeng, A., Zhang, T., Heess, N.M., Sadigh, D., Tan, J., Tassa, Y., & Xia, F. (2023). Language to Rewards for Robotic Skill Synthesis. ArXiv, abs/2306.08647.

[34] Kim, B., Kim, J., Kim, Y., Min, C., & Choi, J. (2023). Context-Aware Planning and Environment-Aware Memory for Instruction Following Embodied Agents. ArXiv, abs/2308.07241.

[35] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P.R., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., & Ichter, B. (2022). Inner Monologue: Embodied Reasoning through Planning with Language Models. CoArialArialnference on Robot Learning.

[36] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L.E., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R.J. (2022). Training language models to follow instructions with human feedback. ArXiv, abs/2203.02155.

[37] Xie, Y., Yu, C., Zhu, T., Bai, J., Gong, Z., & Soh, H. (2023). Translating Natural Language to Planning Goals with Large-Language Models. ArXiv, abs/2302.05128.

[38] Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., & Fox, D. (2019). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. 2020 IEEE/CVF

Conference on Computer Vision and Pattern Recognition (CVPR), 10737-10746.

[39] Min, S., Chaplot, D.S., Ravikumar, P., Bisk, Y., & Salakhutdinov, R. (2021). FILM: Following Instructions in Language with Modular Methods. ArXiv, abs/2110.07342.

[40] Siciliano B., Khatib O. Handbook of Robotics. 2. izd. Berlin, Heidelberg: Springer-Verlag, 2016. 2304lpp. ISBN 978-3-319-32550-7.

[41] Jeon J., Jung H., Yumbla F., Luong T. A., Moon H. Primitive Action Based Combined Task and Motion Planning for the Service Robot. In: Frontiers in Robotics and AI, 2022, 9.

[42] Gizzi E., Castro M. G., Sinapov J. Creative Problem Solving by Robots Using Action Primitive Discovery. In: 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2019, 228-233.

[43] Strudel R., Pashevich A., Kalevatykh I., Laptev I., Sivic J., Schmid C. Learning to combine primitive skills: A step towards versatile robotic manipulation. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2019, 4637-4643.

[44] Moeslund T. B., Hilton A., Krüger V. A survey of advances in vision-based human motion capture and analysis. In: Computer Vision and Image Understanding, 2006, 104(2-3), 90-126.

[45] Simeonov A., Du Y., Kim B., Hogan F. R., Tenenbaum J. B., Agrawal P., Rodriguez A. A Long Horizon Planning Framework for Manipulating Rigid Pointcloud Objects. In: *Conference on Robot Learning*, 2020.

[46] Fikes R., Nilsson N. J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 1971, 2, 189-208.

[47] McDermott D., Ghallab M., Howe A. E., Knoblock C. A., Ram A., Veloso M. M., Weld D. S., Wilkins D. E. PDDL-the planning domain definition language, 1998.

[48] Lin, K., Agia, C., Migimatsu, T., Pavone, M., Bohg, J. Text2Motion: From Natural Language Instructions to Feasible Plans. In: ArXiv, 2023.

[49] Racinskis P, Arents J, Greitans M. A Motion Capture and Imitation Learning Based Approach to Robot Control. Applied Sciences. 2022; 12(14), 7186.

[50] Jenkins O. C., Matarić M. J. Deriving action and behavior primitives from human motion data. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, 3, 2551-2556.

[51] Beetz M., Stulp F., Esden-Tempski P., Fedrizzi A., Klank U., Kresse I., Maldonado A., Ruiz F. Generality and legibility in mobile manipulation: Learning skills for routine tasks. In: Autonomous Robots, 2010, 28(1), 21-44.

[52] Sarathy V., Scheutz M. MacGyver problems: Ai challenges for testing resourcefulness and creativity. In: Advances in Cognitive Systems, 2018, 6, 31–44.

[53] Hogan F. R., Ballester J., Dong S., Rodriguez A. Tactile Dexterity: Manipulation Primitives with Tactile Feedback. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, 8863-8869.

[54] Simeonov A., Du Y., Kim B., Hogan F. R., Tenenbaum J. B., Agrawal P., Rodriguez A. A Long Horizon Planning Framework for Manipulating Rigid Pointcloud Objects. From: Conference on Robot Learning, 2020.

[55] Stone A., Xiao T., Lu Y., Gopalakrishnan K., Lee K., Vuong Q.H., Wohlhart P., Zitkovich B., Xia F., Finn C., Hausman K. Open-World Object Manipulation using Pre-trained Vision-Language Models. In: ArXiv, abs/2303.00905, 2023.

[56] Ugalde F. R. Compact Models of Objects for Skilled Manipulation, 2015.

[57] Mahler J., Matl M., Satish V., Danielczuk M., DeRose B., McKinley S., Goldberg K. Learning ambidextrous robot grasping policies. In: Science Robotics, 2019, 4.