Supplementary Material for "Verifiably Following Complex Robot Instructions with Foundation Models"

CONTENTS

A1. APPENDIX SUMMARY

These sections presents additional details on our approach for leveraging foundation models and temporal logics to verifiably follow expressive natural language instructions with complex spatiotemporal constraints without prebuilt semantic maps. We encourage readers to visit our website robotlimp.github.io for project summary and demonstration videos.

A2. EXTENDED RELATED WORKS

A. Foundation Models in Robotics

Grounding language referents to entities and actions in the world [1, 2, 3, 4] is challenging in part due to the fact that complex perceptual and behavioral meaning can be constructed from the composition of a wide-range of open-vocabulary components[5, 6, 7, 8, 2]. To address this problem, foundation models have recently garnered interest as an approach for generating perceptual representations that are aligned with language [9, 10, 11, 12, 13, 14]. Because there are an ever-expanding number of ways foundation models are being leveraged for instruction following in robotics (e.g: generating plans [14], code [15], etc.), we focus our review on the most related approaches in two relevant application areas : 1) generating natural language queryable scene representations and 2) generating robot plans for following natural language instruction[16, 17].

Visual scene understanding: The most similar approach for visual scene understanding to ours is NLMap [18], a scene representation that affords grounding open-vocabulary language queries to spatial locations via pre-trained visual language models. Given a sequence of calibrated RGB-D camera images and pre-trained visual-language models, NLMap supports language-queries by 1) segmenting out the most likely objects in the 2D RGB images based on the language queries, and 2) estimating the most likely 3D positions via back-projection of the 2D segmentation masks using the depth data and camera pose. While NLMap is suitable for handling complex descriptions of individual objects (e.g: "green plush toy"), it is fundamentally unable to handle instructions involving complex constraints between multiple objects since it has no way to account for object-object relationships (e.g: "the green plush toy that is between the toy box and door"). LIMP handles these more complicated language instructions by using a novel spatial grounding module to easily incorporate a wide-variety of complex spatial relationships between objects. In addition, our scene representation is compatible with both LLM planners as well as TAMP solvers, whereas NLMap is only compatible with LLM planners.

While NLMap is the most relevant approach to ours, there are other approaches for visual scene understanding and task planning with foundation models which are worth highlighting. VoxPoser [19] leverages the abilities of LLMs to identify affordances and write code for manipulation tasks, along with VLMs complementary abilities to identify open-vocabulary entities in the environment. SayPlan [20] integrates 3D scene graphs with LLM-based planners to bridge the gap between complex, heirarchial scene representations and scalable task planning with open-ended task specifications. Generalizable Feature Fields (GeFF) [21] use an implicit scene representation to support open-world manipulation and navigation via an extension of Neural Radiance Fields (NeRFs) and feature distillation in NeRFs. OK-Robot [22] adopts a system-first approach to solving structured mobile pick-and-place tasks with foundation models by offering an integrated solution to object detection, mapping, navigation and grasp generation. While these methods are related, none of them have all the features of LIMP: 1) Explicit support for both LLM-based planning and off-the-shelf task and motion plan-

ning approaches, 2) Verifiable representations for following complex natural language instructions in mobile manipulation domains that involve object-object relationships, and 3) The ability to dynamically generate task-relevant state abstractions (semantic maps) for individual instructions.

Language instruction for robots: Our approach to handling complex natural language instructions involves translating the command into a temporal logic expression. This problem framing allows us to leverage state-of-the-art techniques from machine translation, such as instruction-tuned large language models. Most similar to our approach in this regard is [16], which uses a multi-stage LLM-based approach and finetuning to perform entity-extraction and replacement to translate natural language instructions into temporal logic expressions. However, [16] relies on a prebuilt semantic map that grounds expression symbols, limiting the scope of instructions it can operate since landmarks are predetermined. Instead, our approach interfaces with a novel scene representation that supports open-vocabulary language and generates the relevant landmarks based on the open-vocabulary instruction. Additionally, the symbols in our temporal logic translation correspond to parameterized task relevant robot skills as opposed to propositions of referent entities extracted from instructions.

B. Planning Models in Robotics

Semantic Maps: Semantic maps [23] are a class of scene representations that capture semantic (and typically geometric) information about the world, and can be used in cojunction with planners to generate certain types of complex robot behavior like collision-free navigation with spatial constraints [24, 25]. However, leveraging semantic maps for task planning with mobile manipulators has been challenging since the modeling information needed may highly depend on the robot's particular skills and embodiment. [26] recently proposed Action-Oriented Semantic Maps (AOSMs), which are a class of semantic maps that include additional models of the regions of space where the robot can execute manipulation skills (represented as symbols). [26] demonstrated that AOSMs can be used as a state representation that supports TAMP solvers in mobile manipulation domains. Our scene representation is similar to an AOSM since it captures spatial information about semantic regions of interest, and is compatible with TAMP solvers, but largely differs in that AOSMs require learning via online interaction with the scene. Instead, our approach leverages foundation models and requires no online learning. Also, once an AOSM is generated for a scene, there is only a closed-set of goals that can be planned for, whereas our approach can handle open-vocabulary task specifications.

Task and Motion Planning: Task and Motion planning approaches are hierarchical planning methods that involve high-level task planning (with a discrete state space) [27] and low-level motion planning (with a continuous state space) [28]. The low-level motion planning problem involves generating paths to goal sets through continuous spaces (e.g: configuration space, cartesian space) with constraints on infeasible regions. When the constraints and dynamics can change, it is referred to as multi-modal motion planning, which naturally induces a high-level planning problem that involves choosing which sequence of modes to plan through, and a low-level planning problem that involves moving through a particular mode. Finding high-level plan skeletons and satisfying low-level assignment values for parameters to achieve goals is a challenging bi-level planning problem[28]. LIMP contains sufficient information to produce a problem and domain description augmented with geometric information for bi-level TAMP solvers like [29, 30].

A3. Language Instruction Module

We implement a two-stage prompting strategy in our language instruction module to translate natural language instructions into LTL specifications. The first stage translates a given instruction into a conventional LTL formula, where propositions refer to open-vocabulary objects. For any given instruction, we dynamically generate K in-context translation examples from a standard dataset [31] of natural language and LTL pairs, based on cosine similarity with the given instruction. Here is the exact text prompt used:

```
You are a LLM that understands operators involved with
    Linear Temporal Logic (LTL), such as F, G, U, &, |, ~ ,
     etc. Your goal is translate language input to LTL
    output.
Input:<generated_example_instruction>
Output:<generated_example_LTL>
...
Input:<given_instruction>
Output:
```

Listing 1: Base prompt used to obtain a conventional LTL formula from a natural language query

The second stage takes the given instruction and the LTL response from the first stage as input to generate a new LTL formula with predicate functions that correspond to parameterized robot skills. Skill parameters are instruction referent objects expressed in our novel Composable Referent Descriptor (CRD) syntax. CRDs enable referent disambiguation by chaining comparators that encode descriptive spatial information. We define eight spatial comparators and provide their descriptions as part of the second stage prompt. We find that LLMs conditioned on this information and a few examples are able translate arbitrarily complex instructions with appropriate comparator choices. Here is the exact prompt used:

```
You are an LLM for robot planning that understands
    operators involved with Linear Temporal Logic (LTL),
    such as F, G, U, &, |, ~ , etc. You have a finite set
    of robot predicates and spatial predicates, given a
    language instruction and an LTL formula that represents
     the given instruction, your goal is to translate the
    ltl formula into one that uses appropriate composition
    of robot and spatial predicates in place of
    propositions with relevant details from original
    instruction as arguments.
Robot predicate set (near,pick,release).
Usage:
near[referent_1]:returns true if the desired spatial
    relationship is for robot to be near referent_1.
pick[referent_1]:can only execute picking skill on
    referent_1 and return True when near[referent_1].
release[referent_1,referent_2]:can only execute release
    skill on referent_1 and return True when near[
    referent_2].
```

```
Spatial predicate set (isbetween,isabove,isbelow,isleftof,
    isrightof,isnextto,isinfrontof,isbehind).
Usage:
referent_1::isbetween(referent_2,referent_3):returns true
    if referent_1 is between referent_2 and referent_3.
referent_1::isabove(referent_2):returns True if referent_1
    is above referent_2.
referent_1::isbelow(referent_2):returns True if referent_1
    is below referent_2.
referent_1::isleftof(referent_2):returns True if referent_1
     is left of referent_2.
referent_1::isrightof(referent_2):returns True if
    referent_1 is right of referent_2.
referent_1::isnextto(referent_2):returns True if referent_1
     is close to referent_2.
referent_1::isinfrontof(referent_2):returns True if
    referent_1 is in front of referent_2.
referent_1::isbehind(referent_2):returns True if referent_1
     is behind referent_2.
Rules:
Strictly only use the finite set of robot and spatial
    predicates!
Strictly stick to the usage format!
Compose spatial predicates where necessary!
You are allowed to modify the structure of Input_ltl for
    the final Output if it does not match the intended
    Input_instruction!
You should strictly only stick to mentioned objects,
    however you are allowed to propose and include
    plausible objects if and only if not mentioned in
    instruction but required based on context of
    instruction!
Pay attention to instructions that require performing
    certain actions multiple times in generating and
    sequencing the predicates for the final Output formula!
Example:
Input_instruction: Go to the orange building but before
    that pass by the coffee shop, then go to the parking
    sign.
Input_ltl: F (coffee_shop & F (orange_building & F
    parking_sign ) )
Output: F ( near[coffee_shop] & F ( near[orange_building] &
     F near[parking_sign] ))
Input_instruction: Go to the blue sofa then the laptop,
    after that bring me the brown bag between the
    television and the kettle on the left of the green seat
    , I am standing by the sink.
Input_ltl: F ( blue_sofa & F ( laptop & F ( brown_bag & F (
     sink ) ) ) )
Output: F ( near[blue_sofa] & F ( near[laptop] & F ( near[
    brown_bag::isbetween(television,kettle::isleftof(
    green_seat))] & F (pick[brown_bag::isbetween(television
    ,kettle::isleftof(green_seat))] & F ( near[sink] & F (
    release[brown_bag,sink] ) ) ) ) ) )
Input_instruction: Hey need you to pass by chair between
    the sofa and bag, pick up the bag and go to the orange
    napkin on the right of the sofa.
Input_ltl: F ( chair & F ( bag & F ( orange_napkin ) ) )
Output: F ( near[chair::isbetween(sofa,bag)] & F ( near[bag
    ] & F ( pick[bag] & F ( near[orange_napkin::isrightof(
    sofa)] ) ) ) )
Input_instruction:  Go to the chair between the green
    laptop and the yellow box underneath the play toy
Input_ltl: F ( green_laptop & F ( yellow_box & F ( play_toy
     & F ( chair ) ) ) )
Output: F ( near[chair::isbetween(green_laptop,yellow_box::
    isbelow(play_toy))] )
Input_instruction: Check the table behind the fridge and
    bring two beers to the couch one after the other
Input_ltl: F ( check_table & F ( bring_beer1 ) & F (
    bring_beer2 ) & F ( couch ) )
Output: F ( near[table::isbehind(fridge)] & F ( pick[beer]
    & F ( near[couch] & F ( release[beer,couch] & F ( near[
    table::isbehind(fridge)] & F ( pick[beer] & F ( near[
    couch] & F ( release[beer,couch] ))))))))
Input_instruction: <given_instruction>
Input_ltl: <stage1_ltl_response>
Output:
```

Listing 2: Second stage prompt to output our LTL syntax with CRD parameterized robot skills

A. *Interactive Symbol Verification*

Verifying sampled LTL formulas is essential, as such we implement an interactive dialog system that presents users with extracted referent composible referent descriptors (CRDs) in sampled formulas as well as the implied task structure—encoded in the sequence of state-machine transition expressions that must hold to progressively solve the task. We translate the task structure into English statements via a simple deterministic strategy that replaces logical connectives and skill predicates from the formula with equivalent English phrases. Users can verify a formula as correct or provide corrective statements which are used to reprompt the LLM to obtain new formulas. Here is the exact text prompt used for reprompting.

```
There was a mistake with your output LTL formula: Error
    with <verification_type>! Consider the clarification
    feedback and regenerate the correct output for the
    Input_instruction. Make sure to adhere to all rules and
     instructions in your original prompt!
previous_output:<last_response>
error_clarification: <given_error_clarification>
correct_output:
```

Listing 3: Corrective reprompting prompt used to obtain new LTL formulas

As an illustration, the instruction *"Bring the green plush toy to the whiteboard in front of it"* yields the interactive Referent and Task Structure Verification dialog below:

```
**************************
Instruction Following
**************************
Input_instruction: "Bring the green plush toy to the
    whiteboard in front of it"
Sampled LTL formula: F(A & F(B & F(C & FD)))
    A: near[green_plush_toy]
    B: pick[green_plush_toy]
    C: near[whiteboard::isinfrontof(green_plush_toy)]
    D: release[green_plush_toy, whiteboard::isinfrontof(
    green_plush_toy)]

**************************
Referent Verification
**************************
I extracted this list of relevant objects based on your
    instruction:
    * whiteboard::isinfrontof(green_plush_toy)
    * green_plush_toy
Does this match your intention? (y/n)

****************************
Task Structure Verification
****************************
Based on my understanding here is the sequence of subgoal
    objectives needed to satisfy the task:
Subgoal_1:
    Logical Expression: A&!B
    English translation: I should be near the [
    green_plush_toy] and not have picked up the [
    green_plush_toy]
Subgoal_2:
    Logical Expression: B&!C
    English translation: I should have picked up the [
    green_plush_toy] and not be near the [whiteboard::
    isinfrontof(green_plush_toy)]
Subgoal_3:
    Logical Expression: C&!D
    English translation: I should be near the [whiteboard::
    isinfrontof(green_plush_toy)] and not have released the
     [green_plush_toy] at the [whiteboard::isinfrontof(
    green_plush_toy)]
Subgoal_4:
    Logical Expression: D
```

```
    English translation: I should have released the [
     green_plush_toy] at the [whiteboard::isinfrontof(
     green_plush_toy)]
Does this match your intention? (y/n)
```

Listing 4: Interactive referent and task structure verification dialog.

A4. Spatial Grounding Module

The spatial grounding module detects and localizes specific instances of objects referenced in a given instruction by first detecting, segmenting and back-projecting all referent occurances and then filtering based on the descriptive spatial details captured by each referent's composable referent descriptor (CRD). We use the Owl-Vit model [32] to detect bounding boxes of open-vocabulary referents and SAM [33] to generate masks from detected bounding boxes. To illustrate referent filtering via spatial information, consider an example scenario where the goal is to resolve the composable referent descriptor below:

$$\text{whiteboard} :: isinfrontof(\text{green\_plush\_toy}) \tag{A.1}$$

Let $W = \{w_1, w_2, \ldots, w_n\}$ and $G = \{g_1, g_2, \ldots, g_m\}$ represent the set of representative 3D positions of detected whiteboards and green_plush_toys respectively. The cartesian product of these sets enumerates all possible pairs (w, g) for comparison.

$$W \times G = \{w, g) \mid w \in W, g \in G\} \tag{A.2}$$

The 'isinfrontof(w, g)' comparator is applied to each pair, yielding a subset S that contains only those 'whiteboards' that satisfy the 'isinfronto' condition with at least one 'green_plush_toy'.

$$S = \{w \in W \mid \exists g \in G \text{ such that isinfrontof}(w, g) \text{ is true}\} \tag{A.3}$$

A. 3D Spatial Comparators

Our 3D spatial comparators enable Relative Frame of Reference (FoR) spatial reasoning between referents, based on their backprojected 3D positions. Threshold values in the spatial comparators give users the ability to specify the sensitivity or resolution at which spatial relationships are resolved, we keep all threshold values fixed across all experiments. Below is a description of each spatial comparator.

```
1. isbetween(referent_1_pos, referent_2_pos, referent_3_pos
     , threshold): Returns true if referent_1 is within '
     threshold' distance from the line segment connecting
     referent_2 to referent_3, ensuring it lies in the
     directional path between them without extending beyond.
2. isabove(referent_1_pos, referent_2_pos, threshold):
     Returns true if the z-coordinate of referent_1 exceeds
     that of referent_2 by at least 'threshold'.
3. isbelow(referent_1_pos, referent_2_pos, threshold):
     Returns true if the z-coordinate of referent_1 is less
     than that of referent_2 by more than 'threshold'.
4. isleftof(referent_1_pos, referent_2_pos, threshold):
     Returns true if the y-coordinate of referent_1 exceeds
     that of referent_2 by at least 'threshold', indicating
     referent_1 is to the left of referent_2.
5. isrightof(referent_1_pos, referent_2_pos, threshold):
     Returns true if the y-coordinate of referent_1 is less
     than that of referent_2 by more than 'threshold',
     indicating referent_1 is to the right of referent_2.
6. isnextto(referent_1_pos, referent_2_pos, threshold):
     Returns true if the Euclidean distance between
     referent_1 and referent_2 is less than 'threshold',
     indicating they are next to each other.
7. isinfrontof(referent_1_pos, referent_2_pos, threshold):
     Returns true if the x-coordinate of referent_1 is less
     than that of referent_2 by more than 'threshold',
     indicating referent_1 is in front of referent_2.
8. isbehind(referent_1_pos, referent_2_pos, threshold):
     Returns true if the x-coordinate of referent_1 exceeds
     that of referent_2 by at least 'threshold', indicating
     referent_1 is behind referent_2.
```

Listing 5: Implementation description of 3D spatial comparators

A5. Task and Motion Planning Module

We present pseudocode for our Progressive Motion Planner (Alg.1) and our algorithm for generating Task Progression Semantic Maps (Alg.2). Alg.2 generates a TPSM $\mathcal{M}_{\text{tpsm}}$ by integrating an environment map ($\mathcal{M}$) and a referent semantic map ($\mathcal{M}_{\text{rsm}}$) given a logical transition expression ($\mathcal{T}$), a desired automaton state ($\mathcal{S}'$), and a nearness threshold (θ). The algorithm first initializes $\mathcal{M}_{\text{tpsm}}$ with a copy of $\mathcal{M}$ and extracts relevant instruction predicates from $\mathcal{T}$. For each predicate (parameterized skill), the algorithm identifies satisfying referent positions in $\mathcal{M}_{\text{rsm}}$, generates a spherical grid of surrounding points within a radius θ, and assesses how these points affect the progression of the task automaton towards $\mathcal{S}'$. These points demarcate regions of interest, and are assigned a value of *1* if they cause the automaton to transition to the desired state, *-1* if they lead to a different automaton state or violate the automaton, and *0* if they do not affect the automaton. The points are then integrated into $\mathcal{M}_{\text{tpsm}}$, yielding a semantic map that identifies goal and constraint violating regions.

A6. Robot Skills

We define three predicate functions: **near**, **pick** and **release** for the navigation, picking and placing skills required for multi-object goal navigation and mobile pick-and-place. As highlighted in the main paper, we formalize navigation as continuous path planning problems and manipulation as object parameterized options. We discuss navigation at length in the paper, so here we focus on the pick and place manipulation skills.

Pick Skill: Once the robot has executed the near skill and is at the object to be manipulated, we take a picture of the current environment to detect the object using the Owl-Vit model. The robot is guaranteed to be facing the object as the computed path plan uses the backprojected object 3D position to compute yaw angles for the robot. After detecting the object in the picture, we obtain a segmentation mask from detected boundary box using the Segment Anything model, and compute the center pixel of this mask. We feed this center pixel to the Boston dynamics grasping API to compute a motion plan to grasp the object.

Release Skill: We implement a simple routine for the release skill which takes two parameters: the object to be placed and the place receptacle. Once a navigation skill gets the robot to

Algorithm 1 Progressive Motion Planning Algorithm

procedure PMP($X_{start}, \varphi, \mathcal{M}, \mathcal{M}_{rsm}, \theta$)
Input:
X_{start}: Start position in the environment.
φ: CRD syntax LTL formula specifying task objectives.
$\mathcal{M}$: Environment map.
$\mathcal{M}_{rsm}$: Referent semantic map.
θ: Nearness threshold.
Output:
Π: Generated task and motion plan.
  $\mathcal{A} \leftarrow$ ConstructAutomaton(φ)
  path $\leftarrow$ SelectAutomatonPath($\mathcal{A}$) ▷ Task plan
  while Π.status is active **do**
    while $\mathcal{A}$.state != path.acceptingState **do**
      $\mathcal{S}, \mathcal{T}, \mathcal{S}' \leftarrow \mathcal{A}$.GetTransition(path.currentStep)
      objective $\leftarrow$ NextObjectiveType($\mathcal{T}$)
      if objective = "skill" **then**
        Π.UpdateWithSkill($\mathcal{T}$)
        $\mathcal{A}$.UpdateAutomatonState($\mathcal{S}'$)
      else if objective = "navigation" **then**
        $\mathcal{M}_{params} \leftarrow \mathcal{M}, \mathcal{M}_{rsm}, \mathcal{T}, \mathcal{A}, \mathcal{S}', \theta$
        $\mathcal{M}_{tpsm} \leftarrow$ GENERATETPSM($\mathcal{M}_{params}$)
        $\mathcal{O} \leftarrow$ GenerateObstacleMap($\mathcal{M}_{tpsm}$)
        plan $\leftarrow$ FMT*($X_{start}, \mathcal{O}$) ▷ Path plan
        if plan.exists **then**
          Π.UpdateWithPlan(plan)
          $X_{start} \leftarrow$ plan.endPosition
          $\mathcal{A}$.UpdateAutomatonState($\mathcal{S}'$)
        else
          $\Pi, \mathcal{A}$, path $\leftarrow$ Backtrack($\Pi, \mathcal{A}$, path)
        end if
      end if
    end while
  end while
  return Π
end procedure

Algorithm 2 Task Progression Semantic Mapping Algorithm

procedure GENERATETPSM($\mathcal{M}, \mathcal{M}_{rsm}, \mathcal{T}, \mathcal{A}, \mathcal{S}', \theta$)
Input:
$\mathcal{M}$: Environment map.
$\mathcal{M}_{rsm}$: Referent semantic map.
$\mathcal{T}$: Automaton transition expression.
$\mathcal{A}$: Task Automaton.
$\mathcal{S}'$: Desired State.
θ: Nearness threshold.
Output:
$\mathcal{M}_{tpsm}$: Task Progression Semantic Map.
  $\mathcal{M}_{tpsm} \leftarrow$ Copy($\mathcal{M}$)
  $\mathcal{P} \leftarrow$ ExtractRelevantPredicates($\mathcal{T}$)
  for p in $\mathcal{P}$ **do**
    $\mathcal{R} \leftarrow$ QueryPositions($\mathcal{M}_{rsm}, p$)
    for r in $\mathcal{R}$ **do**
      $G \leftarrow \{g \mid g = r + \delta, \|\delta\| \leq \theta\}$ ▷ spherical grid of surrounding points
      for g in G **do**
        $\mathcal{Q} \leftarrow$ TruePredicatesAt($g, \mathcal{M}_{rsm}, \theta$)
        $\mathcal{S}_{next} \leftarrow$ ProgressAutomaton($\mathcal{A}, \mathcal{Q}$)
        if $\mathcal{S}_{next} = \mathcal{S}'$ **then**
          $g.value \leftarrow 1$ ▷ Goal value
        else if IsUndesired($\mathcal{S}_{next}$) **then**
          $g.value \leftarrow -1$ ▷ Avoidance value
        else
          g.value $\leftarrow 0$
        end if
      end for
      AddPoints($\mathcal{M}_{tpsm}, G$)
    end for
  end for
  return $\mathcal{M}_{tpsm}$
end procedure

the place receptacle, the robot gently moves its arm up or down to release the grasped object, based on the place receptable 3D position. Future work will implement more complex semantic placement strategies to better leverage LIMP's awareness and spatial grounding of instruction specific place receptacles. Kindly, visit our website to see demonstrations of these skills.

A7. Evaluation and Baseline Details

All computation including planning, loading and running pretrained visual language models was done on a single computer equipped with one NVIDIA GeForce RTX 3090 GPU. We leverage GPT-4-0613 as the underlying LLM for our instruction understanding module and all our baselines. In all experiments we set the LLM temperature to 0, however since deterministic greedy token decoding is not guaranteed with GPT4, we perform multiple (3) queries for each instruction and evaluate on the most recurring response (mode response).

We compare LIMP with baseline implementations of NLMap-Saycan [18] and Code-as-policies [34]. Both baselines use the same GPT-4 LLM, prompting structure, and in-context learning examples as our language understanding module. We integrate our composible referent descriptor syntax, spatial grounding module and low-level robot control into these baselines as APIs. This enables baselines to execute plans by querying relevant object positions, using our FMT* path planner to find paths to said positions and executing manipulation options.

A. NLMap-Saycan Implementation Prompt

```
You are an LLM for robot planning that understands logical
    operators such as &, |, ~ , etc. You have a finite set
    of robot predicates and spatial predicates, given a
    language instruction, your goal is to generate a
    sequence of actions that uses appropriate composition
    of robot and spatial predicates with relevant details
    from the instruction as arguments.
Robot predicate set (near,pick,release).
Usage:
near[referent_1]:returns true if the desired spatial
    relationship is for robot to be near referent_1.
```

```
pick[referent_1]:can only execute picking skill on referent_1 and return True when near[referent_1].
release[referent_1,referent_2]:can only execute release skill on referent_1 and return True when near[referent_2].
Spatial predicate set (isbetween,isabove,isbelow,isleftof,isrightof,isnextto,isinfrontof,isbehind).
Usage:
referent_1::isbetween(referent_2,referent_3):returns true if referent_1 is between referent_2 and referent_3.
referent_1::isabove(referent_2):returns True if referent_1 is above referent_2.
referent_1::isbelow(referent_2):returns True if referent_1 is below referent_2.
referent_1::isleftof(referent_2):returns True if referent_1 is left of referent_2.
referent_1::isrightof(referent_2):returns True if referent_1 is right of referent_2.
referent_1::isnextto(referent_2):returns True if referent_1 is close to referent_2.
referent_1::isinfrontof(referent_2):returns True if referent_1 is in front of referent_2.
referent_1::isbehind(referent_2):returns True if referent_1 is behind referent_2.
Rules:
Strictly only use the finite set of robot and spatial predicates!
Strictly stick to the usage format!
Compose spatial predicates where necessary!
You should strictly stick to mentioned objects, however you are allowed to propose and include plausible objects if and only if not mentioned in instruction but required based on context of instruction!
Pay attention to instructions that require performing certain actions multiple times in generating and sequencing the predicates for the final Output!
Example:
Input_instruction: Go to the orange building but before that pass by the coffee shop, then go to the parking sign.
Output:
1. near[coffee_shop]
2. near[orange_building]
3. near[parking_sign]
Input_instruction: Go to the blue sofa then the laptop, after that bring me the brown bag between the television and the kettle on the left of the green seat, I am standing by the sink.
Output:
1. near[blue_sofa]
2. near[laptop]
3. near[brown_bag::isbetween(television,kettle::isleftof(green_seat))]
4. pick[brown_bag::isbetween(television,kettle::isleftof(green_seat))]
5. near[sink]
6. release[brown_bag,sink]
Input_instruction: Hey need you to pass by chair between the sofa and bag, pick up the bag and go to the orange napkin on the right of the sofa.
Output:
1. near[chair::isbetween(sofa,bag)]
2. near[bag]
3. pick[bag]
4. near[orange_napkin::isrightof(sofa)]
Input_instruction:  Go to the chair between the green laptop and the yellow box underneath the play toy
Output:
1. near[chair::isbetween(green_laptop,yellow_box::isbelow(play_toy))]
Input_instruction: Check the table behind the fridge and bring two beers to the couch one after the other
Output:
1. near[table::isbehind(fridge)]
2. pick[beer]
3. near[couch]
4. release[beer,couch]
5. near[table::isbehind(fridge)]
6. pick[beer]
7. near[couch]
8. release[beer,couch]
Input_instruction: <given_instruction>
Output:
```

Listing 6: Exact prompt to implement NlMap-Saycan LLM planner

B. *Code-as-Policies Implementation Prompt*

```
##Python robot planning script
from robotactions import near, pick, release
spatial_relationships = [
"isbetween", #referent_1::isbetween(referent_2,referent_3):returns true if referent_1 is between referent_2 and referent_3.
"isabove",   #referent_1::isabove(referent_2):returns True if referent_1 is above referent_2.
"isbelow",   #referent_1::isbelow(referent_2):returns True if referent_1 is below referent_2.
"isleftof",  #referent_1::isleftof(referent_2):returns True if referent_1 is left of referent_2.
"isrightof", #referent_1::isrightof(referent_2):returns True if referent_1 is right of referent_2.
"isnextto",  #referent_1::isnextto(referent_2):returns True if referent_1 is close to referent_2.
"isinfrontof", #referent_1::isinfrontof(referent_2):returns True if referent_1 is in front of referent_2.
"isbehind"  #referent_1::isbehind(referent_2):returns True if referent_1 is behind referent_2.]
##Rules:
##Strictly only use the finite set of robot and spatial predicates!
##Strictly stick to the usage format!
##Compose spatial predicates where necessary!
##You should strictly stick to mentioned objects, however you are allowed to propose and include plausible objects if and only if not mentioned in instruction but required based on context of instruction!
##Pay attention to instructions that require performing certain actions multiple times in generating and sequencing the predicates for the final Output!
# Go to the orange building but before that pass by the coffee shop, then go to the parking sign.
ordered_navigation_goal_referents = ["coffee_shop", "orange_building", "parking_sign"]
for referent in ordered_navigation_goal_referents:
    near(referent)
# Go to the blue sofa then the laptop, after that bring me the brown bag between the television and the kettle on the left of the green seat, I am standing by the sink.
ordered_navigation_goal_referents = ["blue_sofa", "laptop", "brown_bag::isbetween(television,kettle::isleftof(green_seat))", "sink"]
referents_to_pick= ["brown_bag::isbetween(television,kettle::isleftof(green_seat))"]
release_location_referents = ["sink"]
picked_item = None
for referent in ordered_navigation_goal_referents:
    near(referent)
    if referent in referents_to_pick:
        pick(referent)
        picked_item = referent
    if referent in release_location_referents:
        release(picked_item, referent)
#Hey need you to pass by chair between the sofa and bag, pick up the bag and go to the orange napkin on the right of the sofa.
ordered_navigation_goal_referents = ["chair::isbetween(sofa,bag)", "bag", "orange_napkin::isrightof(sofa)"]
referents_to_pick= ["bag"]
picked_item = None
for referent in ordered_navigation_goal_referents:
    near(referent)
    if referent in referents_to_pick:
        pick(referent)
        picked_item = referent
#Go to the chair between the green laptop and the yellow box underneath the play toy
near("chair::isbetween(green_laptop,yellow_box::isbelow(play_toy))")
#Check the table behind the fridge and bring two beers to the couch one after the other
for i in range(2):
    near("table::isbehind(fridge)")
    pick("beer")
```

```
    near("couch")
    release("beer", "couch")

#<given_instruction>
```

Listing 7: Exact prompt to implement Code-as-policies planner

C. Instruction set

We perform a large-scale evaluation on 150 instructions across five real-world environments. The taskset includes 24 tasks with fine-grained object descriptions (NLMD) from [18], 25 tasks with complex language (NLMC) from [18], 25 tasks with simple structured phrasing (OKRB) from [35], 37 tasks with complex temporal structures (CT) from [16], and an additional 39 tasks with descriptive spatial constraints and temporal structures (CST).

```
1: put the red can in the trash bin
2: put the brown multigrain chip bag in the woven basket
3: find the succulent plant
4: pick up the up side down mug
5: put the apple on the macbook with yellow stickers
6: use the dyson vacuum cleaner"
7: bring the kosher salt to the kitchen counter
8: put the used towels in washing machine
9: move the used mug to the dish washer
10: place the pickled cucumbers on the shelf
11: find my mug with the shape of a donut
12: put the almonds in the almond jar
13: fill the zisha tea pot with a coke from the cabinet
14: take the slippery floor sign with you
15: take the slippers that have holes on them to the shoe
    rack
16: find the mug on the mini fridge
17: bring the mint flavor gum to the small table
18: find some n95 masks
19: grab the banana with most black spots
20: fill the empty bottle with lemon juice
21: throw away the apple that's about to rot
22: throw away the rotting banana
23: take the box of organic blueberries out of the fridge
24: give a can of diet coke to the toy cat
```

Listing 8: Nlmap Detailed Object Tasks (NLMD)

```
1: I opened a pepsi earlier, bring an open can to the
    orange table
2: I spilled my coke, can you put a replacement on the
    kitchen counter
3: I spilled some coke on the television, go and bring
    something to clean it up
4: I accidentally dropped that jalapeno chips after eating
    it. Would you mind throwing it away
5: I like fruits, can you put something I would like on the
     yellow sofa for me
6: There is a green counter, a yellow counter, and a table.
     visit all the locations
7: There is a green counter, a trash can, and a table.
    visit all the locations
8: Redbull is my faviorite drink, can you put one on the
    desk please
9: Would you bring a coke can to the door for me
10: Please, move the pepsi to the red counter
11: Can you move the coke can to the orange counter
12: Would you throw away the bag of chips for me
13: Put an energy bar and water bottle on the table
14: Bring a lime soda and a bag of chips to the sofa
15: Can you throw away the apple and bring a coke to the
    bed
16: Bring a 7up can and a tea to the office desk
17: Move the multigrain chips to the table and an apple to
    the yellow counter
18: Move the lime soda, the sponge, and the water bottle to
     the table
19: Bring two sodas to the table
20: Move three cokes to the trash can
21: Throw away two cokes from the counter
22: Bring two different sodas to the cabinet, there is a
    coke, pepsi, soup, tea and 7up in the fridge
23: Bring an apple, a coke, and water bottle to the sofa
24: I spilled my coke on the table, throw it away and then
    bring something to help clean
25: I just worked out, can you bring me a drink and a snack
     to recover, i am on the sof
```

Listing 9: Nlmap Complex Language Understanding Tasks (NLMC)

```
1: Move the Takis on the desk to the nightstand
2: Move the soda can to the box
3: Move the purple shampoo to the red bag
4: Move the white meds box to the trash bin
5: Move the power adapter to the chair
6: Move the blue gloves to the sink
7: Move the McDonalds paper bag to the stove
8: Move the herbal tea can to the box
9: Move the cooking oil bottle to the marble surface
10: Move the milk bottle to the chair
11: Move the purple shampoo to the white rack
12: Move the purple lightbulb box to the sofa chair
13: Pick up purple medicine, drop it on chair
14: Pick up eyeglass case, drop it on chair
15: Pick up grey rag, drop it in sink
16: Pick up golden can rag, drop it on table
17: Pick up red navaratna oil, drop it on table
18: Pick up purple shampoo, drop it on green rack
19: Pick up taki chips, drop it on countertop
20: Pick up bandage box, drop it in dustbin
21: Pick up white aerosol, drop it in trash can
22: Pick up peanut butter, drop it on countertop
23: Pick up blue gloves, drop it in sink
24: Pick up brown box, drop it on chair
25: Pick up axe body spray, drop it on shel
```

Listing 10: Ok-Robot Tasks (OKRB)

```
1: go to brown bookshelf, metal desk, wooden desk, kitchen
    counter, and the blue couch in any order
2: move to grey door, then bookrack, then go to the brown
    desk, then counter, then white desk
3: visit brown wooden desk but only after bookshelf
4: go from brown bookshelf to white metal desk and only
    visit each landmark one time
5: go to brown wooden desk exactly once and do not visit
    brown desk before bookshelf
6: go to brown desk only after visiting bookshelf, in
    addition go to brown desk only after visiting white
    desk
7: visit the blue IKEA couch, in addition never go to the
    big steel door
8: visit white kitchen counter then go to brown desk, in
    addition never visit white table
9: go to the grey door, and only then go to the bookshelf,
    in addition always avoid the table
10: go to kitchen counter then wooden desk, in addition
    after going to counter, you must avoid white table
11: Go to bookshelf, alternatively go to metal desk
12: Go to counter, alternatively go to metal desk
13: Go to the counter, but never visit the counter
14: do not go to the wooden desk until bookshelf, and do
    not go to bookshelf until wooden desk
15: go to brown desk exactly once, in addition go to brown
    desk at least twice
16: move to couch exactly twice, in addition pass by
    counter at most once
17: navigate to the counter then the brown desk, in
    addition after going to the counter, you must avoid
    doorway
18: visit counter at least six times
19: either go to bookshelf then the brown desk, or go to
    couch
20: navigate to the wooden door, the glass door and the
    table, kitchen counter, and the blue couch in any order
21: go to the painting, then find the kitchen table, then
    front desk, then staircase
22: navigate to classroom but do not visit classroom before
     the white table
23: only visit classroom once, and do not visit classroom
    until you visit elevator first
```

```
24: Go to the staircase, front desk and the white table in that exact order. You are not permitted to revisit any of these locations
25: go to the front desk then the yellow office door, in addition do not visit the glass door
26: go to the stairs then the front desk, in addition avoid purple elevator
27: move to elevator then front desk, in addition avoid staircase
28: go to front desk then the cabinet, always avoid the elevator
29: Go to elevator, alternatively go to staircase
30: Visit the elevator exactly once, in addition visit the front desk on at least 2 separate occasions
31: Go to the office, in addition avoid visiting the elevator and the classroom
32: Visit the front desk, in addition you are not permitted to visit elevator and staircase
33: Visit the purple door elevator, then go to the front desk and then go to the kitchen table, in addition you can never go to the elevator once you have seen the front desk
34: Visit the front desk then the sofa then the white table , in addition if you visit the sofa you must avoid the television after that
35: Go to the glass door, but never visit the glass door
36: do not go to the white table until classroom, and do not go to the classroom until white table
37: find the office, in addition avoid visiting the front desk and the classroom and the table
```

Listing 11: Lang2LTL Complex temporal Tasks (CT)

```
1: Go to the red sofa but dont pass by any door then go to the computer, the sofa is behind the pillar and on the right side
2: Find the door with posters in front of the yellow pillar then go stand by the tree but avoid any sofa
3: There is a cabinet, a television and a tree in this room . I want you to go to the last item I mentioned then pass by the second item but dont go close to the first when you are going to thsecond item.",
4: Go to the trash bin, after that go to the large television. Actually can you go to the laptop on the table before doing all that?
5: Try to go to the tree without going near the trash bin.
6: Find the yellow trash bin and go to it, then go by the white table. But before doing the first thing visit where the fridge is then after the last thing, stop by the red sofa but avoid theooden door with posters",
7: Go to the television but avoid the tree.
8: Go to any trash bin but not the yellow one
9: Visit one of the televisions in this room then go to the fridge to the left of the cabinet
10: I need you to first go to the table between the pillar and the door with posters, then go to the tree. You know what, ignore the last thing I asked you to go to, after the first thing go tthe computer next to the green box instead.
11: Go to the red sofa then the blue one next to the cabinet but first pass by the book shelf
12: Go to water filter, dont pass by any computer to the left of the red sofa
13: Visit these things in the following order the microwave , then the black chair infront of the yellow robot then the fridge, but before doing any of this go to the robot
14: Go to the cabinet between the blue sofa and the brown box.
15: I want you to go the cabinet but dont go near any black chair. Hold on, actually can you pass by the whiteboard before doing all that?
16: I need you to go to the red sofa but dont go anywhere near the sink
17: You can either go as close as you can to the sink or the plant pot but dont go near the yellow robot.
18: Go to the green curtain after that the large television but try not to go near the red sofa
19: Go to the sink then the red sofa, then the guitar. After that I want you to return to the first thing you visited.
20: I need you to go to the sink then any computer on a table but before you start that can you first go to the bookshelf, when you are done with all that, return to brown box
21: There are a couple of things in this room, a coffee machine, a robot, a computer and a plant pot. Visit the second item I mentioned then the first, after that visit the brown box, then go  the items you haven't visited yet.
22: Avoid any cabinet but go get the mug on the blue sofa and take it to red one
23: Please get me my plant pot from between the sofa and the sink, bring it to the white cabinet
24: Can you pick up my book from the left side of the television and bring it to me? I am sitting on the couch which is to the right of the door with posters. Make sure to never go near the trhbin while doing all this, it on the yellow table.
25: Bring the green plush toy to the whiteboard infront of it, watchout for the robot infront of the toy
26: Go get my yellow bag and bring it to the table between the yellow pillar and the wooden door with posters next to the whiteboard
27: Find the bottle that is on the table to the left of the computer and bring it to the wardrobe that is next to the glass door
28: Go to the cabinet between the blue sofa and the yellow robot or the cabinet to the left of the blue sofa.
29: Go take the green toy that is next to the sofa under the poster and bring it to the bookshelf
30: I have a white cabinet, a green toy, a bookshelf and a red chair around here somewhere. Take the second item I mentioned from between the first item and the third. Bring it the cabinet butvoid the last item at all costs .
31: Hey I am standing by the whiteboard infront of the bookshelf, can you bring me the mug from the table to the left of the fridge?
32: Never pass by a robot, but i need you to bring the bag on the table to the cabinet
33: Take a soda can to the fridge, you can find one on the table to the left of the blue sofa
34: Hey, can you pick up my bag for me? It is under the table infront of the glass door, bring it to the cabinet. Actually you know what, forget what i asked for and bring the green toy instea its at the same place",
35: Go to the sink and take the mug beside the coffee maker , drop it off at the red sofa
36: Go to the sofa behind the chair next to the orange counter in front of the pillar
37: I want you to go visit the chair behind the sofa then the green bag, do that 3 times but during the second time avoid the fridge
38: Go bring the plush toy between the sofa and the television to the couch, my cat is on the brown table so please dont pass near the table when you are returning with the toy
39: I think I left my wallet on the kitchen counter, go get it i will meet you at the bed. There is a lantern near the bed, make sure you dont hit it
```

Listing 12: Complex Spatiotemporal Tasks (CST)

REFERENCES

[1] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[2] V. Blukis, R. A. Knepper, and Y. Artzi, "Few-shot object grounding and mapping for natural language robot instruction following," *arXiv preprint arXiv:2011.07384*, 2020.

[3] R. Patel, R. Pavlick, and S. Tellex, "Learning to ground language to temporal logical form," in *NAACL*, 2019.

[4] C. Wang, C. Ross, Y.-L. Kuo, B. Katz, and A. Barbu, "Learning a natural-language to ltl executable semantic

parser for grounded robotics," in *Conference on Robot Learning*, pp. 1706–1718, PMLR, 2021.

[5] K. Zheng, D. Bayazit, R. Mathew, E. Pavlick, and S. Tellex, "Spatial language understanding for object search in partially observed city-scale environments," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pp. 315–322, IEEE, 2021.

[6] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex, "Grounding language to landmarks in arbitrary outdoor environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 208–215, IEEE, 2020.

[7] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: Interactively translating unstructured natural language to temporal logics with large language models," *arXiv preprint arXiv:2303.04864*, 2023.

[8] X. Wang, W. Wang, J. Shao, and Y. Yang, "Lana: A language-capable navigator for instruction following and generation," *arXiv preprint arXiv:2303.08409*, 2023.

[9] S.-M. Park and Y.-G. Kim, "Visual language navigation: A survey and open challenges," *Artificial Intelligence Review*, vol. 56, no. 1, pp. 365–427, 2023.

[10] D. Shah, B. Osiński, S. Levine, *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning*, pp. 492–504, PMLR, 2023.

[11] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," *arXiv preprint arXiv:2209.09874*, 2022.

[12] B. Yu, H. Kasaei, and M. Cao, "L3mvn: Leveraging large language models for visual target navigation," *arXiv preprint arXiv:2304.05501*, 2023.

[13] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," *arXiv preprint arXiv:2212.04088*, 2022.

[14] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," *arXiv preprint arXiv:2210.05714*, 2022.

[15] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022.

[16] J. X. Liu, Z. Yang, I. Idrees, S. Liang, B. Schornstein, S. Tellex, and A. Shah, "Lang2ltl: Translating natural language commands to temporal robot task specification," *arXiv preprint arXiv:2302.11649*, 2023.

[17] E. Hsiung, H. Mehta, J. Chu, X. Liu, R. Patel, S. Tellex, and G. Konidaris, "Generalizing to new domains by mapping natural language to lifted ltl," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3624–3630, IEEE, 2022.

[18] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11509–11522, IEEE, 2023.

[19] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[20] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," in *7th Annual Conference on Robot Learning*, 2023.

[21] R.-Z. Qiu, Y. Hu, G. Yang, Y. Song, Y. Fu, J. Ye, J. Mu, R. Yang, N. Atanasov, S. Scherer, *et al.*, "Learning generalizable feature fields for mobile manipulation," *arXiv preprint arXiv:2403.07563*, 2024.

[22] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Ok-robot: What really matters in integrating open-knowledge models for robotics," *arXiv preprint arXiv:2401.12202*, 2024.

[23] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[24] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic information for robot navigation: A survey," *Applied Sciences*, vol. 10, no. 2, p. 497, 2020.

[25] A. Pronobis, *Semantic mapping with mobile robots*. PhD thesis, KTH Royal Institute of Technology, 2011.

[26] E. Rosen, S. James, S. Orozco, V. Gupta, M. Merlin, S. Tellex, and G. Konidaris, "Synthesizing navigation abstractions for planning with portable manipulation skills," in *Conference on Robot Learning*, pp. 2278–2287, PMLR, 2023.

[27] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[28] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[29] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and black-box samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, pp. 440–448, 2020.

[30] R. Holladay, T. Lozano-Pérez, and A. Rodriguez, "Planning for multi-stage forceful manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6556–6562, IEEE, 2021.

[31] J. Pan, G. Chou, and D. Berenson, "Data-efficient learning of natural language to linear temporal logic translators for robot task specification," *arXiv preprint arXiv:2303.08006*, 2023.

[32] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, *et al.*, "Simple open-

vocabulary object detection with vision transformers. arxiv 2022," *arXiv preprint arXiv:2205.06230*.

[33] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[34] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, IEEE, 2023.

[35] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Ok-robot: What really matters in integrating open-knowledge models for robotics," *arXiv preprint arXiv:2401.12202*, 2024.