Closed Loop Interactive Embodied Reasoning for Robot Manipulation

Michal Nazarczuk[1], Jan Kristof Behrens[2], Karla Stepanova[2], Matej Hoffmann[2] and Krystian Mikolajczyk[1]

***Abstract*— Embodied reasoning systems integrate robotic hardware and cognitive processes to perform complex tasks typically in response to a natural language query about a specific physical environment. This usually involves changing the belief about the scene or physically interacting and changing the scene (*e.g. Sort the objects from lightest to heaviest*). In order to facilitate the development of such systems we introduce a new simulating environment that makes use of MuJoCo physics engine and high-quality renderer Blender to provide realistic visual observations that are also accurate to the physical state of the scene. Together with the simulator we propose a new benchmark composed of 10 classes of multi-step reasoning scenarios that require simultaneous visual and physical measurements. Finally, we develop a new modular Closed Loop Interactive Reasoning (CLIER) approach that takes into account the measurements of non-visual object properties, changes in the scene caused by external disturbances as well as uncertain outcomes of robotic actions. We extensively evaluate our reasoning approach in simulation and in the real world manipulation tasks with a success rate above 76% and 64%, respectively.**

I. Introduction

The research efforts in developing systems capable of high-level perception and reasoning [1], [2], [3], [4], [5], [6] have been increasing in recent years. Many robotics systems include an agent that is required to perform a task with a specified goal given visual observations or instructions in natural language [7], [8], [1], [4], [9]. Such tasks requires a simulation environment for generating data, or on-line agent training, together with an evaluation process. Existing environments [10], [11] differ in terms of physics simulation engines and the quality of visual observations. High computational load of realistic rendering forces a compromise between real-time physics calculation and visual quality of the simulation. Realistic synthetic visual data are crucial for development of robotics systems but the collection of such data is often restricted by a cumbersome setup process and real time robot operations [12]. More powerful simulators, datasets and benchmarks are needed for a closed-loop setup where changes to the world or physical measurements (*e.g.*, weight, elasticity, occlusions, etc.) should be accounted for as directly affecting the next step in action planning.

[1]Michal Nazarczuk and Krystian Mikolajczyk are with Imperial College London `[michal.nazarczuk17, k.mikolajczyk]@imperial.ac.uk`

[2]Jan Kristof Behrens, Karla Stepanova and Matej Hoffmann are with the Czech Technical University in Prague `[jan.kristof.behrens, karla.stepanova]@cvut.cz`,

J.K.S., K.S., and M.H. were supported by the European Union under the project ROBOPROX (reg. no. CZ.02.01.01/00/22_008/0004590). This work originated in the project Interactive Perception-Action-Learning for Modelling Objects (IPALM) (H2020-FET-ERA-NET Cofund-CHIST-ERA III / TAČR EPSILON, No. TH05020001)

In this paper, we emphasise the following contributions[1]:

◦ We introduce MuBlE, a novel modular environment for simulating and executing interactive manipulation tasks characterised by high-quality visual rendering and accurate physics calculation. It employs MuJoCo for physics and Blender for rendering, which we refer to with MuBlE (MuJoCo+Blender Environment). It can generate multimodal data based on scene specification and instruction templates. An example scene and task that can be executed in this environment is shown in Fig. 1. We believe to be the first work focusing on task planning in robotics manipulation while preserving accurate physics modelling between the bodies, in contrast to discrete action spaces such as in ManipulaThor [8] and ALFRED [4].

◦ We propose SHOP-VRB2, a new benchmark for evaluating embodied closed loop reasoning systems for robotic manipulation. It includes 10 types of benchmarking tasks for single- and multi-step tabletop manipulations that require combined reasoning between visual observations (recognising attributes of objects and their relations) and physical measurements (acquiring properties obtainable through interaction only, such as weight or stiffness). SHOP-VRB2 extends [13] with a challenging combination of various modalities (language, vision, and manipulation) for benchmarking Visual Reasoning, VQA (*e.g.* [14]) (language + vision), and Embodied question answering (EQA) (*e.g.* [6], [15]) (language + vision + navigation) approaches.

◦ We propose a closed-loop neuro-symbolic interactive reasoning approach (CLIER), that can plan, evaluate and execute a given task (*e.g.*, *Place the heaviest mug on the box*). The highlight of our approach is that it can plan for perception of invisible object properties (*e.g.*, weight or stiffness) and account for new measurements to adjust the plan. Our reasoning agent is able to recover from a wide range of manipulation failures caused by external disturbances and uncertain outcomes of manipulation actions.

◦ We evaluate our proposed Closed Loop Interactive Embodied Reasoning (CLIER) within MuBlE using the proposed SHOP-VRB2 synthetic benchmark and validate it via comparative experiments between simulated and real-world YCB [16] objects.

In the remainder of this paper, we first review the existing simulators and justify the need for a new environment. We then present the proposed MuBlE and CLIER in Sec. III. Sec. IV introduces our SHOP-VRB2 benchmark. Results and discussion are presented in Sec. V.

[1]The environment, the benchmark and the reasoning approach are released on the project webpage https://michaal94.github.io/CLIER

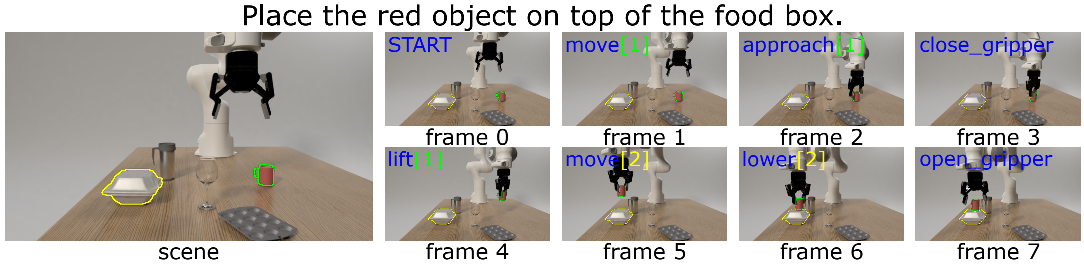

Fig. 1. An example task presenting capabilities of the proposed environment: synthetic scene generation, instruction generation, execution of symbolic actions for manipulation followed by physics calculation and realistic rendering. Symbolic actions with corresponding targets marked in the image. Best viewed zoomed.

II. Related work

Simulation environments for manipulation focused on combined visual and language robotic tasks. State-of-the-art simulators ManipulaTHOR [8] (extension of AI2-THOR [10]), CoppeliaSim [19], or iGibson [20] provide non-photorealistic rendering, lacking accurate shades and reflections. This is in contrast to Blender that can generate photorealistic scenes with procedural object models. ISAAC-Sim [11] is a resource-intensive, closed-source simulator with limited customisation. ThreeDWorld [17] also makes use of a commercial renderer and does not provide the infrastructure for robotics manipulation tasks, including motion planners. Unlike prior works, we pay particular attention to conditional action planning and execution in robotics manipulation while preserving accurate physics modelling between the bodies, and offer both continuous and discrete action spaces, in contrast to discrete action spaces such as in ThreeDWorld [17] and ManipulaThor [8], or pure continuous control of the robot as seen in Robosuite [18]. For example, ManipulaThor [8] considers grasping as a discrete action, while we enable to execute sequences of various custom primitive actions (e.g., approach, gripper closing, lifting, etc.) or continuous control of the robot. From these simulators, only iGibson offers non-kinematic continuous and discrete object states *e.g*. temperature or burned. Our **MuBlE environment** differs from the aforementioned manipulator simulators by a unique and open-source combination of 1) high-quality visual data and realistic physics modelling, which is a key advantage when attempting sim2real transfers (as shown in our experiments), 2) ability to select between operating on continuous action space or usage of primitive actions, and 3) considering both visible and non-visible continuous object states (e.g., weight, or stiffness). We present a comparison between different simulators in Table I.

Benchmarks for language-conditioned manipulation. The advantages of MuBlE environment enabled us to prepare a unique **benchmark for closed-loop embodied reasoning (SHOP-VRB2)** that extends SHOP-VRB [13]. SHOP-VRB is a dataset closely related to the robotic manipulation scenarios considered in this work. It features various objects suitable for robotic manipulation that can be easily obtained. There are many high quality 3D models of various instances of these objects which make them suitable for generating synthetic scenes via Blender rendering that includes procedurally generated materials rather than simply texturing the mesh. We therefore extend it to a dataset and a benchmark SHOP-VRB2 that enables development and testing of systems capable of reasoning on non visual attributes and performing interactive tasks.Compared to other benchmarks, such as ALFRED [4] (based on RoboTHOR), a benchmark for understanding natural language instructions in robotics, VLMbench [21], based on RLbench [22] and CoppeliaSim [19], that adds linguistic commands and realises automatic complex tasks builders, IKEA Furniture Assembly [23], that provides a benchmark for various assembly tasks and CALVIN benchmark [24] offering natural language instructions for long-horizon manipulation tasks, our SHOP-VRB2 benchmark requires long-horizon closed-loop embodied reasoning, that involves manipulation of the world to acquire knowledge about non-visual object properties in order to complete the task or answer a query.

Symbolic approaches gained interest in visual reasoning tasks and several deep learning systems incorporate neuro-symbolic programs. CLEVR-IEP [25] suggested predicting symbolic programs to be executed on disentangled scene representation, and was followed by NS-VQA [26], and NS-CL [27]. V2A [28] introduced a symbolic approach for robotic tasks using only the initial state of the scene, and was evaluated in a real robotic environment [6], however, only open-loop single-step tasks were demonstrated. Symbolic approaches were also considered in Task and Motion Planning systems (TAMP) that were often based on the formal language PDDL [29]. Classic TAMP approaches rely on predefined rules and known dynamic models [30], [31], [32] but learning based methods are increasingly replacing handcrafted heuristics in TAMP methods [33], [34], [35]. PDDLStream [36] extends PDDL to add any sub-symbolic model in a black-box way, which also opens it to the use of learning-based methods. However, it works only on a symbolic level and does not close the loop with the real world. Deep Visual Reasoning [37] directly predicts task plans by considering initial observation only. Regression Planning Networks (RPN) [38] propose a task planning model which performs regression in symbolic space using neural networks. RPN are expanded in [39] by utilising regression planning on scene graph representations to estimate the next subgoal based on the current scene and a given scene graph, *i.e*. solving only subpart of the overall task that our method deals with. The interaction with the real world that

TABLE I

COMPARISON OF THE MUBLE'S CAPABILITIES WITH THE RELATED SIMULATORS FOR THE ROBOTIC EXPERIMENTS. OUR MUBLE CAN BE CHARACTERISED BY HIGH QUALITY RENDERING WITH BLENDER, ACCURATE PHYSICS WITH MUJOCO, INCLUDED VISIBLE AND NON-VISIBLE STATES, VARIOUS ACTION SPACES AVAILABLE, TARGETING TABLETOP MANIPULATION, MODULAR AND EASILY EXTENDABLE.

Simulator	Rendering			Physics		Motion planner	Non-kin. states	Non-visible states	Action space	Speed	Scale	Open-source
	library	material	quality	library	supports							
AI2-THOR [10]	Unity	textures	++	Unity	rigid dyn./animation	✗	✓	✗	Discrete	+	room	✓
ManipulaTHOR [8]	Unity	textures	++	Unity	AI2THOR+manip.	✗	✗	✗	Discrete	+	room	✓
ThreeDWorld [17]	Unity/V-Ray (off.)	Configurable	+++	Unity+FLEX	rigid/particle dyn.	✗	✗	✗	Continuous	++	house	✗
Robosuite [18]	MuJoCo/NVISII	Configurable	++	MuJoCo	rigid/artic. dyn.	✓	✗	✗	Continuous	++	tabletop	✓
ISAACSim [11]	Omniverse RTX	Configurable	++	PhysX	rigid/artic. dyn.	✓	✗	✗	(Disc.)&Cont.	++	house	✗
Habitat 2.0 [9]	Magnum	3D scans/PBR	+	PyBullet	rigid/artic. dyn.	✗	✗	✗	Continuous	+++	house	✓
CoppeliaSim [19]	OpenGL	Gouraoud shad.	+	PyBullet	rigid/artic. dyn.	✓	✗	✗	(Disc.)&Cont.	++	room	✗
iGibson [20]	PyRender/OpenGL	PBR shad.	+	PyBullet	rigid/artic. dyn.	✓	✓	✗	Continuous	++	house	✓
MuBle (ours)	MuJoCo/Blender	procedural	+++	MuJoCo	rigid/artic. dyn.	✓	✓	✓	Disc.& Cont.	+	tabletop	✓

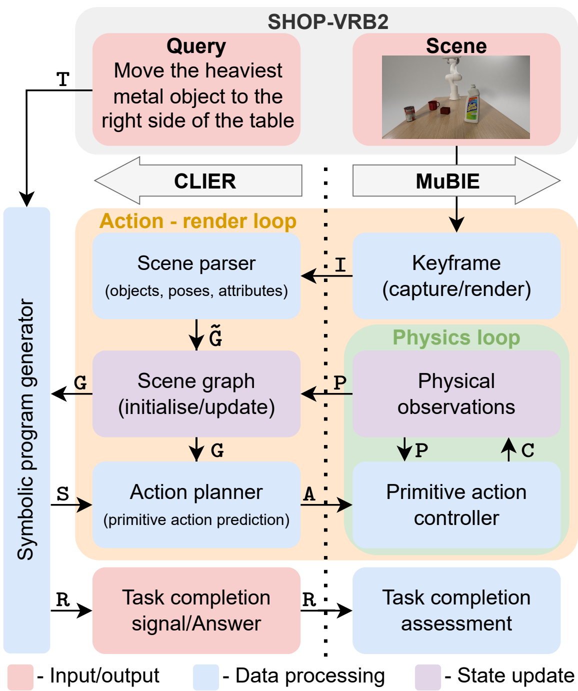

Fig. 2. A diagram of CLIER reasoning and MuBlE environment including interaction between the two, and use of SHOP-VRB2 benchmark. Symbols for transferred data: `T` - text of the query, $\tilde{\texttt{G}}$ - prediction of scene graph elements, `G` - current scene graph, `S` - subgoal (symbolic program requiring physical measurements), `I` - image, `P` - physical observations, `C` - control signal, `A` - primitive action to take, `R` - returned result.

follows a linguistic query (*e.g.measure the weight of the mug*, *find the sofa*) is explored in **embodied question answering approaches** [15], [6], [40]. In [6], language-conditioned visual reasoning is combined with manipulation. Navigation of agents based on vision and language was investigated in VLN [1]. However, both approaches considered tasks that do not react to incoming measurements, *i.e.* reasoning is not conditioned on new observations. Recently, several works utilised large-langauge models (LLMs) to generate action plans for robot manipulation tasks (eg., [41], [42]). While they show impressive versatility and adaptability to novel environments, they suffer from lack of explainability, verifiability, and repeatability. LLMs require careful prompt engineering and filtering of outputs before the robot can execute them. Furthermore, such approaches do not capture the task and trajectory details and do not enable fast-loop reactiveness. These problems have to be resolved to broadly deploy LLMs for embodied reasoning. Using data created with MuBlE environment, we train and evaluate a novel **CLIER reasoning agent** that can act on a natural language query in a real scene. Moreover, we present closed-loop reasoning experiments in simulation and real tabletop scenes. A highlight of our approach is that it plans for perception of invisible object properties, *e.g.* when stacking objects by weight. Moreover, unlike classical methods, our reasoning agent is executed on every key frame, which allows it to recover from a wide range of manipulation failures (e.g., unsuccessful grasping, or falling object during manipulation).

III. MUBLE ENVIRONMENT AND CLIER REASONING

In this section, we present our MuBlE environment for simulating manipulation tasks and reasoning CLIER approach that makes use of this environment. The main modules and data flow are presented in Fig. 2.

MuBlE is built on `robosuite` [18] which is a simulation framework suitable for creating robotic environments inside the physics engine MuJoCo. We equip our environment with high quality rendering powered by Blender. It is designed for a generic tabletop scenario with a single robotic manipulator and a gripper. We use the same set of robots and grippers as `robosuite`. We provide a set of object models in MuJoCo and their counterparts in Blender along with templates to create new items.

We first introduce the scene graph in Sec. III-A and action planning in Sec. III-B that is a part of CLIER. We then present two main components of the environment and reasoning approach: action (render) loop in Sec. III-C, and physics loop in Sec. III-D. Sec. III-E summarises our reasoning pipeline.

A. Scene Graph

A real scene can be captured by a camera or a synthetic one can be rendered by Blender based on a detailed scene description. Scene graph `G` is generated for each key frame `I` starting from the first image of the scene. A keyframe is rendered after a motion corresponding to the primitive action

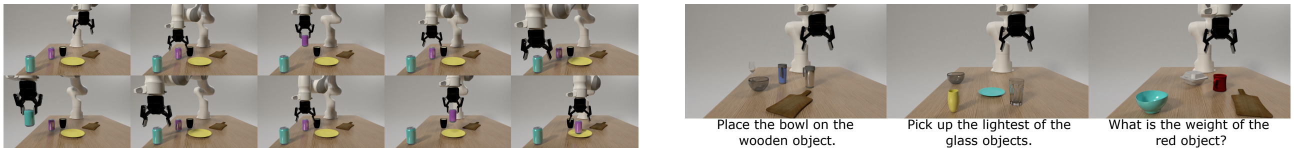

Fig. 3. (Left) Examples of visual observations (selected frames) generated for the actions corresponding to the instruction: *Stack the lightest of metal objects on the yellow object*. Note that metal cans were initially picked up to measure their weight. Further, according to the measurement taken, the heavier of the cans was put down and the lighter one was picked up and stacked on the yellow plate. (Right) Example scenes and corresponding instructions in natural language generated with MuBlE (in the dataset, instructions left to right belong to tasks 7, 3, and 1 in Tab. II).

is finished e.g. moving from object 1 to object 2, approaching grasping pose, closing the gripper, etc. We have chosen the keyframe based approach as it allows us to use Blender for rendering frames, which have sufficient quality for sim2real transfer, and sufficient speed to allow for recovering from failures with the closed loop approach.

To parse the scene into a set of objects with attributes $\tilde{\mathtt{G}}$ we apply an instance segmentation with Mask R-CNN [43], followed by classification of object attributes with ResNet [44] and pose regression with CosyPose [45]. Scene graph is a sequence of feature vectors corresponding to each object in the scene including whether the object is in the gripper or whether the gripper is closed. We use simple geometrical heuristics to create or update the scene graph $\mathtt{G}$ for keyframe $\mathtt{I}$, similarly to [39].

High-quality rendering of every frame can be used for training but it is too slow for real-time experiments. However, many tasks are achievable with low rate visual reasoning and fast feedback loops from other sources of information (*e.g*. force control). Example key frames rendered during execution of a task are presented in Fig. 3 (Left).

B. Symbolic program generator and action planner

As the reasoning method we propose a multi-staged approach inspired by Neural Symbolic VQA approaches [13], [26]. Symbolic program generator is implemented as a Seq2Seq network [46] that given translates the natural language query $\mathtt{T}$ into a sequence of symbolic programs that are evaluated on the scene graph $\mathtt{G}$ for keyframe $\mathtt{I}$, producing a subgoal $\mathtt{S}$ for the task *e.g*. measuring weight. We employ a transformer as the action planner to predict the next primitive action $\mathtt{A}$ and its target, given the current scene graph $\mathtt{G}$ and subgoal $\mathtt{S}$.

To enable planning of various manipulation tasks we design a set of primitive actions $\mathtt{A}$ that can be easily extended in MuBlE:

- `move` – moves the end effector towards a given target (*e.g*. another object or part of the table),
- `approach` – positions the end effector in a grasping position with respect to the target object
- `close_gripper`, `open_gripper` – to grasp or release,
- `lift`, `lower` – lifts or lowers the end effector,
- `weigh` – weighs the object in the gripper,
- `squeeze` – squeezes the object in the gripper to measure its stiffness.

Non visual states, such as weight or stiffness are encoded in the scene graph after measurements, however the transformer is provided only a binary flag indicating whether the property was measured or not and the sorting reasoning is performed once all the measurements are completed.

C. Action (render) loop

Scene graph and action planner make part of the action (renderer) loop in Fig. 2. Primitive action controller implements a control that can execute action $\mathtt{A}$ on target object in MuJoCo physics engine or real robot. Physics loop, discussed in more details in Sec. III-D), calculates physics and collects observations $\mathtt{P}$ (e.g. pose of end effector, physical measurement, force in the gripper, etc.). A control signal is generated in every step of the physics loop, until the path is completed. After the execution, a new key frame is captured in real-world setup or rendered in simulation.

Primitive action execution with obstacle avoidance require the positions and orientations $\mathtt{P}$ of the objects in the scene. Every primitive action, including gripper closure, generates a desired trajectory (position and orientation) $\mathtt{P}$ for the end effector. The action of approaching to the grasp position entails a set of pre-coded grasp sequences. We implement grasping of cylindrical objects (depending on their position and size), grasping by the edge, grasping by the handle etc. We use a simplified model as simulated grasping is often not reliable and inconsistent with real grasping. Whenever grasping pads on all gripper fingers are in contact with the same object, it is considered as grasped. Thereafter, the object's pose is fixed with respect to the end effector. Further, whenever the gripper starts to open, the object is detached from the gripper body with an initial velocity of 0. More details can be found on GitHub page: `https://michaal94.github.io/CLIER`

D. Physics loop

The physics are calculated with the use of MuJoCo engine which makes part of MuBlE and is used by CLIER as shown in Fig. 2. With a user defined timestep, the control signal $\mathtt{C}$ is applied to the manipulator, the corresponding forces are calculated and applied to all objects in the scene. Similarly to `robosuite`, the control signal takes the form of the displacement of the end effector, *i.e*. position and orientation change, along with binary signal for opening and closing of the gripper. By default, the control signal is translated to desired joint forces by the Operational Space Controller [47], which can be customised. A control signal to the environment can be provided by the user directly or can be obtained from the primitive action controller, which calculates the error of current end effector pose and the planned trajectory.

Every step of applying the control $\mathtt{C}$ affects the physics of the scene and generates a set of observations $\mathtt{P}$, which

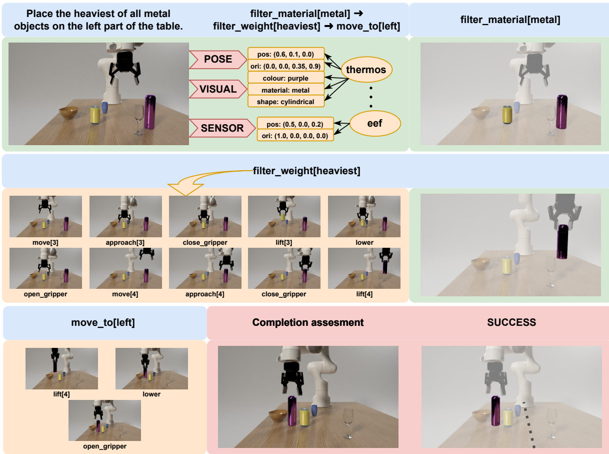

Fig. 4. An example task from the proposed dataset demonstrating the CLIER pipeline. Blue colour denotes execution of symbolic programs, the output targets from programs are shown in the green, yellow denotes executing sequence of primitive actions after having received a trigger from program (blue). Note that actions in yellow result in updates of the scene graph depicted in the top middle.

include: position and orientation of the end effector, joint positions and velocities, status of the gripper closure. Observations of measured non-visual properties, such as stiffness, weight, or elasticity, are also included. Additionally, measurements related to all objects in the scene may be collected, such as positions, orientations, bounding boxes, indication whether the object is currently in the gripper. Any custom sensor can be added to MuBlE as in `robosuite`.

E. Reasoning pipeline

CLIER starts by parsing the visual scene to create or update its semantics and geometry scene graph. Simultaneously, a symbolic program in the CLEVR-IEP [25] format is extracted from a natural language instruction. The program is then executed on the scene graph to select the subgoals that have to be achieved on the scene, *e.g.* measuring the weight of the target when the graph node of the target's weight is empty. Further, action planning regresses the task to a sequence of primitive actions for the current state of the scene. It reevaluates the next step at every pass of the action (render) loop. The reasoning is completed when either the answer is obtained from the program, or the state of the scene matches the target in task completion modules. The components of the pipeline are presented in Fig. 2 and an example task in Fig. 4.

IV. SHOP-VRB2 Dataset

We introduce SHOP-VRB2 dataset created in MuBlE for training and benchmarking. The dataset includes a set of scenes with instructions to perform various tasks (e.g. *Stack metal objects from heaviest to lightest*). Tasks are designed to enforce reasoning simultaneously on the visual observations (recognising attributes of objects and their relations) and continuous physical measurements, taking the feedback loop into account. Every example is accompanied with a ground truth sequence of actions for successful execution, along with visual observations and detailed scene graphs. Some examples can be found in Fig. 3 and 4. We include weight measurement as a representative example of estimating non visual object properties through manipulation due to the ease of repeatability for other researchers but we also demonstrate stiffness measurements. Other properties such as roughness are also possible but require another template for training and an implementation of the control. We first discuss the tools we provide in MuBlE that can be used to generate a scene and natural language instructions Sec. IV-A. We then introduce our SHOP-VRB2 benchmark in Sec. IV-B.

A. Data generation tools

Scene generator is a tool in MuBlE that can procedurally generate data for training models. We provide a template scene that is compatible with the renderer within the framework. The procedural algorithm takes a set of object models, which can be customised, and randomly places them on the tabletop. Desired properties of the objects such as colour, material, and size can also be randomised. We check for collisions between meshes, unlike previous approaches that check only bounding boxes. Afterwards, a maximal desired level of occlusion is ensured. Finally, we render an image of the scene, generating full ground truth, including segmentation masks for all the objects, the robot, and the tabletop, as well as a depth map. Fig. 3 (Right) presents some example scenes generated with MuBlE. Objects in the scene are described in Sec. IV.

Instruction generator is another tool in MuBlE that given the scene, generates natural language instructions or tasks that require reasoning and interaction. We propose a template-based algorithm where we improve upon CLEVR and SHOP-VRB, which are time-consuming and scale poorly when more adjectives are introduced in the templates. This is due to evaluating all possible combinations of describing words, and validating the descriptions with respect to the given scene. Instead, we implement an efficient rejection mechanism. We first generate all combinations of short descriptions for all scene objects. Short descriptions of objects are more natural than in [48], [13]. Next, we evaluate the descriptions using scene constraints *e.g.* the target of the instruction has to satisfy the constraint of being *pickupable* otherwise the description is rejected. The instruction targets are randomly selected from the validated descriptors and used to generate ground truth symbolic program. We automatically produce the ground truth actions by reasoning backwards with predefined pairwise relations *e.g.* lifting an object requires grasping it, grasping requires approaching to the grasping position, etc.). Fig. 3 presents example instructions generated for the given scenes using templates that are described in Sec. IV.

B. Benchmark data

SHOP-VRB2 scenes include 12000 realistically rendered scenes split to train, validation, and test sets: 10000:1000:1000. The scenes contain typical household objects which are easily accessible. Following [13], various

TABLE II

INSTRUCTION TEMPLATES CORRESPONDING TO THE BENCHMARKING TASKS IN THE PROPOSED DATASET.

No.	Instruction Templates
1.	Measure the weight of the `OBJ1`.
2.	What is the weight of all `OBJ1`s?
3.	Pick up the `WS1` of all `OBJ1`s.
4.	Place the `OBJ1` on the `TP1` part of the table.
5.	Remove all `OBJ1`s from the `TP1` part of the table.
6.	Place the `WS1` of all `OBJ1`s on the `TP1` part of the table.
7.	Stack the `OBJ1` on top of the `OBJ2`.
8.	Place the `WS1` of all `OBJ1`s on top of the `OBJ2`.
9.	Stack the `OBJ1` on top of the `OBJ2` on top of the `OBJ3`.
10.	Stack all `OBJ1`s from heaviest to lightest.

instances of objects are included: baking trays, bowls, chopping boards, food containers, glasses, mugs, plates, soda cans, thermoses, and wine glasses. The scenes are generated with 4 to 5 objects per scene (their respective distribution is 47.3% and 52.7%). We make sure the randomization of the materials (plastic, metal, glass, rubber, wood) is realistic. Out of 54317 placed objects, a subset was selected randomly, with 1330 instances of the least common (big chopping board model due to many possible collisions), and 3486 instances of the most common model (bowl – less prone to collisions).
YCB scenes include 30 simulated benchmarking scenes with 9 YCB-Video [49] objects and 3 randomly generated scenes for each of the benchmarking tasks described below. 9 YCB objects were selected so that the objects share various types of visual/physical attributes (cleanser, mustard, mug, bowl, tomato can, Cheez-it, sugar box, meat can, foam brick).
Benchmarking tasks We assign one instruction to each scene to introduce more diversity in visual observations. We designed 10 classes of benchmarking tasks revolving around moving and stacking objects based on their visual (colour, material, shape), and physical properties (weight). Example templates are presented in Tab. II. The task types are closely related to the instruction templates in Tab. II and involve 1) measuring weight of a single object, 2) measuring weight of multiple objects, 3) picking up based on weight, 4) moving single object 5) moving multiple objects 6) moving based on weight 7) stacking objects 8) stacking objects according to weight 9) stacking 3 objects 10) ordering objects according to their weight. `OBJx` refers to a description of an object consisting of a set of visual properties (chosen randomly and validated), *e.g. the red object* presented in Fig. 1. Note that tasks may refer to either one, specific unique object, or a set of objects sharing a certain property *e.g.* template 1 and 2 in Tab. II. Further, `TPx` specifies a part of the table *e.g.* the left and the right part. Finally, `WSx` refers to weight specifier, *i.e.* distinction whether the lightest or the heaviest object from the set is the target of the instruction. The length of the instructions ranges between 5 and 16 words. The resulting sequences contain between 5 (measure the weight of the single object) and 46 (stacking several items according to weight) primitive actions. All instructions are accompanied with symbolic programs in CLEVR-IEP [25] format and task specifications with respect to the scene graph.

TABLE III

(LEFT) SUCCESS RATES FOR CLIER METHOD ON SHOP-VRB2 (VRB, SIM) AND YCB DATASET (SIM/REAL). (RIGHT) EXECUTION OUTCOMES ON SHOP-VRB2, INCL. SUCCESSES (BOLD) AND TYPE OF FAILURES. SEE THE WEBPAGE LINKED IN SEC. I FOR MORE DETAILS AND VIDEO ILLUSTRATION.

Success [%]	VRB	YCB	
Task type	Sim	Sim	Real
Weight single	74.0	66.7	88.9
Weight multi	65.0	100	66.7
Pick up weight	49.0	100	88.9
Move single	76.0	66.7	100
Move multi	47.0	100	44.4
Move weight	23.0	100	100
Stack	56.0	66.7	66.7
Stack weight	31.0	33.3	22.2
Stack three	0.0	66.7	0.0
Order weight	18.0	66.7	100
Overall	43.9	76.7	64.4

Perc[%]	VRB
Exit code	Sim
Correct answer	13.9
Task success	30.0
Task failure	0.1
Execution err	14.4
Loop detected	10.8
Physics err	3.5
Program err	4.5
Recognition err	9.6
Output error	0.6
Scene inconsistent	12.6
Total	100%

Ground truth for visual and physical observations are provided for every scene and its benchmarking instruction. An observation is taken after every primitive action is executed and contains: position and orientation of end effector, gripper status, weight measurement, ground truth scene graph, segmentation masks, and action with its target.

V. BENCHMARKING AND RESULTS

In this section we provide benchmarking results for CLIER. We show results for two datasets: 1) in simulation for SHOP-VRB2 dataset (Sec. V-A), and 2) on a set of 30 benchmarking scenes in simulation and in real-world with YCB objects (Sec. V-B).
Benchmarking metrics are the rate of successful task execution (*e.g.* stacking) and correct question answer (*e.g.* weight query). Additionally, we provide accuracy split into task types presented in Tab. II. Finally, for the inference tool of CLIER we provide a classification of incorrect attempts, including errors of execution, incorrect scene recognition, loop detection, timeout, inconsistency in object tracking.

The latency of various modules is the following. CosyPose inference is 0.8s, attributes recognition: 0.15s, transformer action planner: 0.02s (as run on 2080Ti). The prediction of the next action takes around 1s as measured on the hardware. These are reasonable delays given that these modules are deployed at keyframes.

A. SHOP-VRB2 experiments

Results for CLIER are presented in Table III(Left) which shows success rates for tasks. We observe high success rates for the tasks that include manipulation of single objects (weighing or moving one object), and strong decrease in accuracy for multi-object manipulation (stacking more objects, preceding manipulation with weighting). Further, we identify the most common reasons for failure in Table III(Right) which reports the distribution of error codes from the environment. Execution error (14.4%) accounts for failures in execution of primitive actions which may arise from inaccurate scene description (*e.g.* typically object pose). Scene inconsistency (12.6%) refers to mistakes in tracking objects IDs between frames. Loop detection (10.8%) arises when

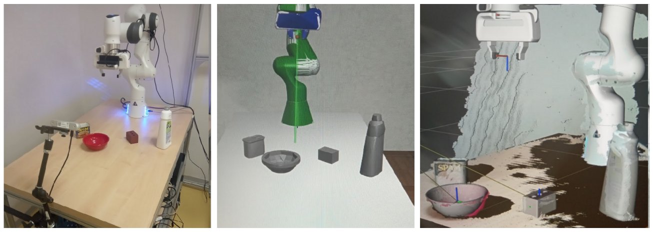

Fig. 5. The real setup with YCB objects (left), corresponding MuJoCo simulation using estimated poses (middle), and RViz visualisation of coloured pointcloud with overlaid grey models detected by CosyPose (right).

primitive action chains are repeated (*e.g.* when approaching object to grasp with misaligned position). Note that this experiment uses simple, ResNet based pose estimation in contrast to more accurate CosyPose [45] that was trained for YCB objects but not for SHOP-VRB2 objects. Finally, we believe that overall accuracy of 43.9% on SHOP-VRB2 indicates that this data forms a challenging benchmark for evaluating visual and interactive reasoning.

B. Real-world experiments

In this section, we report comparative results for real YCB scenes that mirror 30 YCB scenes introduced in Sec. IV-B.

Experimental Setup includes Franka Emika Panda [50] arm with 7 degrees of freedom and a 2-finger parallel gripper. An (extrinsically calibrated) Intel Realsense D455 camera is set to face the robot and capture the front view of the scene (see Fig. 5). To allow CLIER to control the real robot, we developed a thin ZMQ [51] based communication layer that emulates the same interface as for the simulated robot (*i.e.*, motion generation using position-based servoing). Special purpose skills include measuring stiffness via squeezing the object with different forces and measuring deformation or weight via lifting and joint torque differences. Note that the weighing has a relative error of less than $\pm 10\%$ in the range $0 - 200$ g compared to the ground truth weight. The stiffness measurement has a coefficient of variation ranging from $1.6 - 3.4$ %, which is highly repeatable. However, we were not able to obtain ground truth data for stiffness.

Sim2Real Our method uses raw RGB rendered images to train the attribute recognition module (*c.f.* Fig. 2). This is not further tuned between YCB sim and real experiments. The same reasoning model (*c.f.* Sec. III-E) is evaluated in 30 synthetic and real scenes using the same CosyPose [45] on RGB images. The reasoning modules operate on the scene graph thus are agnostic to visual input.

Results We mirror experiments with the real scenes in our simulation environment. To evaluate the contribution of different modules we perform testing along with ablations by replacing them with ground truth (GT) data *i.e.* 1) reasoning success with GT pose and GT attributes 2) pose inference with GT attributes and GT reasoning, 3) pose inference and attribute recognition with GT reasoning, 4) full inference without using GT. The results for all these combinations are in Table IV where the success is the average across 3 runs of each experiment. Note that Seq2seq network in symbolic program generator is not included in the table as it performed

TABLE IV

SUCCESS RATE FOR DIFFERENT SETUPS (INFERENCE VS. GROUND TRUTH) IN SIMULATION AND REAL WORLD. **C** - COSYPOSE, **A** - ATTRIBUTES RECOGNITION, **R** - REASONING, ✓- MODULE USED, ✗- MODULE SUBSTITUTED WITH GROUND TRUTH INFORMATION.

C	A	R	Simulation	Real
✗	✗	✓	86.7%	86.7%
✓	✗	✗	90.0%	80.0%
✓	✓	✗	90.0%	76.7%
✓	✓	✓	76.7%	64.4%

with 100% accuracy, therefor the error for case 1) comes from transformer action planner. We observe a very similar performance between simulation and real which validates the usefulness of high quality vision and physics in MuBlE. The performance for individual tasks with YCB scenes in simulation and real world is reported in Tab. III(Left). Sim and Real differences for YCB can be attributed mostly to CosyPose object detection and its robustness to occlusion.

Typical execution errors Our closed loop reasoning approach is able to recover from various execution errors such as dropping or moving an object (see video). Typical errors during the real world execution are caused by uncertainty in pose estimation. This leads to various cases: 1) grasp succeeds because object's real pose is withing the margin for error. 2) a different grasp is achieved *e.g.* over a rim instead over diameter of an open cylindrical object. 3) the object slips out when closing the gripper. 4) the gripper collides with the object during the approach and causes a robot error (safety stop reflex). The robot can recover from all except case (4) or when the object is pushed out of the workspace in (3). A varied pose of the object in the gripper (i.e., (1) and (2)) occasionally influences subsequent tasks such as stacking or weighing. Detection of occluded objects is also prone to failures. In particular, occlusion from the gripper when grasping is an issue when updating the scene graph. We use the last estimated object pose and leverage the information of the gripper holding 'something' to overcome these pose estimation errors.

Joint limits and singularities. Robot kinematic agnostic servoing of the end-effector can lead to degenerate robot configurations. These situations include 1) joint limits, 2) the borders of the workspace, 3) self-collisions, and 4) alignment of several joint axes. To avoid running into joint limits, we start each experiment in a good standard pose and the robot always rotates to a neutral down-facing end-effector orientation before moving via the smallest rotation to the goal rotation. Self collisions are avoided by the Panda controller and are counted as *robot errors*.

VI. DISCUSSION AND CONCLUSIONS

In this paper, we presented our closed-loop approach for visual and interactive reasoning for robotic manipulation tasks: closed loop interactive embodied reasoning (CLIER). This approach is using our novel MuBlE environment that incorporates MuJoCo physics simulation with high-quality renderer and enables generation of multi-modal demonstration data for robotic manipulation tasks. This fully modular environment enables both data generation and benchmarking of simultaneous reasoning in visual and physical space. CLIER is able to incorporate observations of both visual and physical attributes of the manipulated objects into long-term reasoning. Capturing data from visual and physical measurements in the shared scene graph enables the symbolic reasoning approach and simplifies the implementation of the closed loop approach via keyframe based reasoning. The results from simulated and real-world experiments showed its ability to successfully transfer between the simulated and real environment and to recover from various errors or changes in the scene.

We believe this work provides a new challenge and a benchmark for future systems that aim at bridging the gap between physics simulation for robotic manipulators, realistic visual simulations and execution in real environments.

REFERENCES

[1] H. Huang, *et al.*, "Transferable Representation Learning in Vision-and-Language Navigation," in *Int. Conf. Comput. Vis.*, 2019.

[2] A. Kamath, *et al.*, "MDETR - Modulated Detection for End-to-End Multi-Modal Understanding," in *Int. Conf. Comput. Vis.*, 2021.

[3] W. Kim, *et al.*, "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision," in *Int. Conf. Mach. Learn.*, 2021.

[4] M. Shridhar, *et al.*, "ALFRED A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[5] S. Storks, *et al.*, "Are We There Yet? Learning to Localize in Embodied Instruction Following," in *AAAI Worksh. Reason. Learn. for Human-Machine Dialogs*, 2021.

[6] J. K. Behrens, *et al.*, "Embodied Reasoning for Discovering Object Properties via Manipulation," in *IEEE Int. Conf. Robot. Automat.*, 2021.

[7] M. Deitke, *et al.*, "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform Real Simulation Action Same Object Different Appearance Same Action Different Result," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[8] K. Ehsani, *et al.*, "ManipulaTHOR: A Framework for Visual Object Manipulation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[9] A. Szot, *et al.*, "Habitat 2.0: Training Home Assistants to Rearrange their Habitat," in *Adv. Neural Inform. Process. Syst.*, 2021.

[10] E. Kolve, *et al.*, "AI2-THOR: An Interactive 3D Environment for Visual AI," *CoRR*, 2017.

[11] V. Makoviychuk, *et al.*, "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning," in *Adv. Neural Inform. Process. Syst. D&B*, 2021.

[12] J. Collins, *et al.*, "Benchmarking Simulated Robotic Manipulation through a Real World Dataset," *IEEE Robot. Automat. Let.*, 2020.

[13] M. Nazarczuk *et al.*, "SHOP-VRB: A Visual Reasoning Benchmark for Object Perception," in *IEEE Int. Conf. Robot. Automat.*, 2020.

[14] R. Vedantam, *et al.*, "Probabilistic Neural-symbolic Models for Interpretable Visual Question Answering," in *Int. Conf. Mach. Learn.*, 2019.

[15] A. Das, *et al.*, "Embodied Question Answering," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[16] B. Calli, *et al.*, "The YCB object and Model set: Towards common benchmarks for manipulation research," in *Int. Conf. Adv. Robot.*, 2015.

[17] C. Gan, *et al.*, "ThreeDWorld: A Platform for Interactive Multi-Modal Physical Simulation," in *Adv. Neural Inform. Process. Syst. D&B*, 2021.

[18] Y. Zhu, *et al.*, "robosuite: A modular simulation framework and benchmark for robot learning," *CoRR*, 2020.

[19] E. Rohmer, *et al.*, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[20] C. Li, *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Conference on Robot Learning*, 2022, pp. 455–465.

[21] K. Zheng, *et al.*, "VLMbench: A Compositional Benchmark for Vision-and-Language Manipulation," *Adv. Neural Inform. Process. Syst. D&B*, 2022.

[22] S. James, *et al.*, "RLBench: The robot learning benchmark & learning environment," *IEEE Robot. Automat. Let.*, 2020.

[23] Y. Lee, *et al.*, "IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks," in *IEEE Int. Conf. Robot. Automat.*, 2021.

[24] O. Mees, *et al.*, "CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks," *IEEE Robot. Automat. Let.*, 2022.

[25] J. Johnson, *et al.*, "Inferring and Executing Programs for Visual Reasoning," in *Int. Conf. Comput. Vis.*, 2017.

[26] K. Yi, *et al.*, "Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding," in *Adv. Neural Inform. Process. Syst.*, 2018.

[27] J. Mao, *et al.*, "The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision," in *Int. Conf. Learn. Represent.*, 2019.

[28] M. Nazarczuk *et al.*, "V2A-Vision to Action: Learning robotic arm actions based on vision and language," in *ACCV*, 2020.

[29] M. Fox *et al.*, "PDDL2.1 : An Extension to pddl for Expressing Temporal Planning Domains," *J. Artif. Intell. Research*, 2003.

[30] N. T. Dantam, *et al.*, "Incremental Task and Motion Planning: A Constraint-Based Approach," *Robot. Science Syst.*, 2016.

[31] L. P. Kaelbling *et al.*, "Hierarchical Task and Motion Planning in the Now," in *IEEE Int. Conf. Robot. Automat.*, 2011.

[32] T. Migimatsu *et al.*, "Object-Centric Task and Motion Planning in Dynamic Environments," *IEEE Robot. Automat. Let.*, 2019.

[33] R. Chitnis, *et al.*, "Guided Search for Task and Motion Plans Using Learned Heuristics," in *IEEE Int. Conf. Robot. Automat.*, 2016.

[34] B. Kim, *et al.*, "Learning to guide task and motion planning using score-space representation," *Int. J. Robot. Research*, 2019.

[35] Z. Wang, *et al.*, "Active model learning and diverse action sampling for task and motion planning," in *Int. Conf. Intell. Robot. Sys.*, 2018.

[36] C. R. Garrett, *et al.*, "PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning," *ICAPS*, 2020.

[37] D. Driess, *et al.*, "Deep Visual Reasoning: Learning to Predict Action Sequences for Task and Motion Planning from an Initial Scene Image," *Robot. Science Syst.*, 2020.

[38] D. Xu, *et al.*, "Regression Planning Networks," in *Adv. Neural Inform. Process. Syst.*, 2019.

[39] Y. Zhu, *et al.*, "Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs," in *IEEE Int. Conf. Robot. Automat.*, 2021.

[40] L. Yu, *et al.*, "Multi-Target Embodied Question Answering," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[41] J. Wu, *et al.*, "Tidybot: Personalized robot assistance with large language models," *Int. Conf. Intell. Robot. Sys.*, 2023.

[42] D. Driess, *et al.*, "Palm-e: An embodied multimodal language model," *Int. Conf. Mach. Learn.*, 2023.

[43] K. He, *et al.*, "Mask R-CNN," in *Int. Conf. Comput. Vis.*, 2017.

[44] K. He., *et al.*, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[45] Y. Labbé, *et al.*, "CosyPose: Consistent multi-view multi-object 6D pose estimation," in *ECCV*, 2020.

[46] I. Sutskever, *et al.*, "Sequence to Sequence Learning with Neural Networks," in *Adv. Neural Inform. Process. Syst.*, 2014.

[47] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Robot. Automat. Let.*, 1987.

[48] J. Johnson, *et al.*, "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[49] Y. Xiang, *et al.*, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *Robot. Science Syst.*, 2018.

[50] Franka Emika GmbH, "Franka emika robot's instruction handbook," 2021.

[51] iMatix, "Zero-mq." [Online]. Available: https://zeromq.org