"Don't forget to put the milk back!" Dataset for Enabling Embodied Agents to Detect Anomalous Situations

James F. Mullen Jr,[1,2] Prasoon Goyal,[1] Robinson Piramuthu,[1] Michael Johnston,[1]
Dinesh Manocha[2], and Reza Ghanadan[1]

***Abstract*— Home robots intend to make their users lives easier. Our work moves toward more helpful home robots by enabling them to inform their users of dangerous or unsanitary anomalies in the home. Some examples of these anomalies include the user leaving their milk out, forgetting to turn off the stove, or leaving poison accessible to children. To enable home robots with these abilities, we have created a new dataset, which we call SafetyDetect. The SafetyDetect dataset consists of 1000 anomalous home scenes, each of which contains unsafe or unsanitary situations for an agent to detect. Our approach utilizes large language models (LLMs) alongside both a graph representation of the scene which encodes relationships between the objects in the scene. Our key insight is that this connected scene graph and the object relationships it encodes enables the LLM to better reason about the scene — especially as it relates to detecting dangerous or unsanitary situations. Our most promising approach utilizes GPT-4 and pursues a classification technique where object relations from the scene graph are classified as normal, dangerous, unsanitary, or dangerous for children. This method is able to correctly identify over 90% of anomalous scenarios in the SafetyDetect Dataset. Additionally, we conduct real world experiments on a ClearPath TurtleBot where we generate a scene graph from visuals of the real world scene, and run our approach with no modification. This setup resulted in little performance loss. The SafetyDetect Dataset and code will be released to the public upon this papers publication.**

I. Introduction

Detecting anomalies consisting of unsafe and unsanitary conditions in the home is key functionality required for home robots to be useful for users. For instance, if you forget to put your milk back in your fridge, or leave the front door ajar, you would expect your home robot to notify you, or solved the problem itself. Additionally, users with children will materially benefit from a robot that can monitor the environment for their children's safety.

These types of scenarios can present a real danger to people in the home. For example, 31% of home cooking fires are caused by unattended equipment [1], over 42,000 people died from falls sustained at home or at work [2], and accidents, including poisoning and suffocation, are the leading cause of death for children in the United States [3]. If a home robot can monitor the stove to make sure it is properly turned off, police the environment for tripping hazards, and monitor the home for accessible poisons or suffocation hazards, many of these fires, injuries, or deaths can be prevented.

[1]Amazon Science { prasog,robinpir,mjohnstn, ghanadan} @amazon.com
[2]University of Maryland mullenj@umd.edu, dmanocha@umd.edu

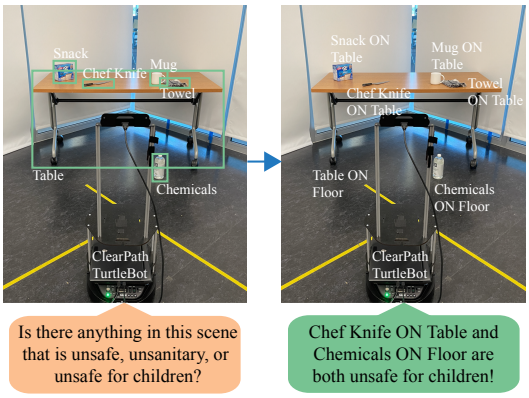

Fig. 1. In this work we aim to enable embodied agent to detect unsafe and unsanitary conditions in the home. For this, we first create a new, unique dataset with unsafe and unsanitary conditions to detect. We then hypothesize that Large Language Models (LLMs) contain the knowledge needed to logically operate on these conditions. Our approach creates LLM prompts that leverage object relationships in the scene from a scene graph (like those in the right image with objects being nodes and relationships being edges), and classifies them. In addition to testing on our dataset, we test in the real world using the ClearPath TurtleBot in scenarios like that shown here.

Building embodied agents that can perform household tasks has seen a lot of recent interest from the robotics and embodied AI communities. Some commonly approached problems include navigation [4], [5], instruction following, and embodied question answering [6], [7], [8]. Each of these tasks defines a precise goal, e.g. navigating to a set location, moving objects to a set location, or answering a question correctly. However, our use case, detecting anomalies in the home, is a poorly formulated problem consisting of an incredibly diverse and disparate set of examples. As such, it is difficult to extensively hard-code scenarios into a robots behavior. One would have to record a rule for where every object can or cannot be kept, and what state it must be in. Additionally, these rules would have to adapt to an unknown and unique set of objects and personal preferences in a household. We hypothesize that recent advances in Large Language Models (LLMs) [9], [10], [11] could enable our use case by providing diverse knowledge on the home that can be leveraged to detect potentially unsafe or unsanitary conditions. However, there are no sources of data that contain environments with these conditions to test potential methods on. Additionally, we show in this work that previous approaches to prompting LLMs are unable to effectively extract the knowledge necessary to detect unsafe and unsanitary conditions.

Main Contributions: We introduce the SafetyDetect dataset to benchmark the ability of embodied AI agents to infer unsafe or unsanitary situations in the house. Figure 1 showcases our task where an agent is spawned randomly in an unknown environment, with an unknown set of potentially unsafe or unsanitary conditions. Without explicit instructions, the agent must discover any potential anomalies and report them to the user.

The SafetyDetect dataset is challenging due to the diverse and unrelated nature of the possible anomalies in the scene. For example, medicine on the ground (poison hazard for children) is hardly connected to moldy produce on the counter (sanitation hazard). Additionally, unlike other anomaly detection tasks commonly found in computer vision literature like [12], [13], [14], [15] in our task agents cannot be trained on a base environment to directly detect changes. This is for two reasons: first, collecting data of a static and perfectly clear home environment is exceptionally challenging for real-world applications, and second, not all changes from the base environment are 'anomalies' as we refer to only those that create dangerous or unsanitary conditions. In essence, agents must not only detect changes, but are required to use some form of knowledge to deduce the danger to the user.

We propose baseline solutions that encapsulate knowledge leveraged from LLMs and demonstrate that this serves as an effective solution for the SafetyDetect dataset. Our key finding is that utilizing a scene graph is a particularly effective way of informing the LLMs of the proper scene context and semantic information for scene understanding and spatial reasoning. Baseline methods that do not use a scene graph perform very poorly on the SafetyDetect dataset.

Our main contributions are as follows:

1) We present the SafetyDetect dataset, a new, first-of-its-kind dataset, built off the VirtualHome simulator[16], aimed at enabling researchers to create embodied agents that can detect unsafe or unsanitary conditions in the home. This dataset, at release, contains 1000 scenarios for users to explore and solve. We additionally provide information about each scenario which informs user preference and how the robot should report a given anomaly to the user.
2) We present an LLM-based method that leverages a scene graph, classification, and chain of thought prompting to perform exceptionally well on a simplified version of this task, with an anomaly detection rate of 96%.
3) We show that use of the scene graph when creating the LLM prompt is important for performance on our SafetyDetect dataset through its ability to provide scene information to the LLM in a concise way.
4) We explore the sim-to-real transfer of our method, tested on the SafetyDetect dataset, and demonstrate its performance with an in-lab demonstration. We do this on a ClearPath TurtleBot by having it create a scene graph before running our method and having it detect unsafe or unsanitary anomalies in a simplified real-world environment.

II. Related Work

A. Similar Datasets or Benchmarks

While to our knowledge no previous work has approached household anomaly detection as a task, there are some alternate tasks that are similar in scope or implementation. Behavior1K [17] is a simulation benchmark where the robots must complete 1000 everyday tasks. [18] and [19] both focus on the idea of cleaning up clutter in the home and placing things in their proper locations. HouseKeep [18] is the most similar to our work as in both tasks, agents are placed randomly into the environment and must find issues, unsafe situations in our case and misplaced objects in theirs. Unique to our work is the specific use case of the detection of unsafe or unsanitary conditions. Approaches on the HouseKeep dataset are built solely around their task and would need significant modification to work on SafetyDetect.

B. Language-Based Embodied AI

Using language to inform robotic agents is a popular task in literature, with work including using generalized grounding graphs [20] for robot manipulation [21], [22] to performing language-guided navigation [4], [5]. Tellex et al. [23] recently presented a useful survey on using language from a robotics perspective.

More recent work tackling this problem by Thomason et al. [6], [7] and Gao et al. [8] has explored the use of human-robot dialogue to gather relevant information for completing tasks. [24] specifically uses dialogue to ascertain anomalies alongside a novel taxonomy of potential anomalies. Similarly, [25] uses Visual Question-Answering (VQA) and a series of yes/no questions associated with a specific list of anomalies to detect said anomalies. Different from these works, we are focused on using natural language derived from a scene graph as a medium for scene understanding. Additionally, our approach does not 'interact' with the robot through dialogue, instead expecting the robot to detect anomalies on their own. However, parsing and utilizing natural language is very relevant in our work, and we are motivated by the techniques developed by these papers.

With the advent of ChatGPT [9], LLaMA [11], FLAN-T5 [26], and other LLMs, the field of Embodied AI worked to leverage them to improve performance on their tasks. Dorbala et al. [27] use language models to inform navigation for object goal navigation. Singh et al. [28] use language models to write code that solves a given task. We differ from these methods in terms of task and prompting technique. Specifically, creating the SafetyDetect dataset to explore the detection of unsafe conditions is substantially different from the prior work which is generally focused on question-answering [29], [30], task-completion [28], [31], or object goal navigation [27]. Additionally, our utilization of the scene graph to provide context to the LLM when prompting is a key differentiator from these methods.

C. Using the Scene Graph

Scene graphs are a common method of representing a scene in computer graphics and 3D modeling where, gen-

Dataset	Goal	Scenarios	Object Categories	Object Models	Scenes
Transport Challenge	Geometric	Inf	50	112	15
Behavior	Predicate	1000	391	1217	15
My House, My Rules	Human Preferences	75	12	12	2
Housekeep	Human Preferences	585	268	1799	14
Ours	**Anomalies**	1000	192	1163	7

TABLE I

COMPARISON OF SAFETYDETECT TO OTHER SIMILAR BENCHMARKS. SAFETYDETECT IS COMPARABLE TO THE OTHER DATASETS IN SCALE WHILE APPROACHING A DIFFERENT END TASK. NOTE THAT WE CAN USE PROCEDURAL GENERATION TO GO BEYOND 1000 SCENARIOS AND 7 SCENES.

erally, nodes of the graph are objects and edges are relationships. For example, nodes could be a 'Table,' 'Floor,' and 'Lamp', and some edges could then be an 'ON' relationship between the 'Lamp' and the 'Table,' and again between the 'Table' and the 'Floor.' Additionally, creating a scene graph from images is a popular problem in the computer vision community [32], [33]. Many simulation platforms for embodied AI are built on top of a scene graph including both Habitat [34] and VirtualHome [16]. This makes a scene graph a relatively easy to access source of information, both in simulation and real world environments, for methods trying to solve embodied AI tasks. However, few works have attempted to leverage the scene graph for scene understanding with LLMs.

In contrast to our approach with *scene* graphs, existing LLM literature uses *knowledge* graphs as a means of finding information to the LLM as context. These methods generally conduct a semantic search of the knowledge graph to find said information [35], [36], [37]. These knowledge graphs are typically different from scene graphs in that they generally encode past experiences or examples as nodes and relationships between those experiences and a ground truth behavior or success/failure as edges. The closest work to ours is SayPlan [38] which in fact uses a scene graph, but only uses it as a knowledge graph and a means of conducting a semantic search on the scene to provide relevant information to the LLM based on the task at hand. In contrast to this, we utilize the scene graph to create strings that encapsulate object relationships to then feed into the LLM alongside a classification approach that does not allow for filtering the graph extensively.

III. ADDING ANOMALIES TO THE HOME: THE SAFETYDETECT DATASET

A. Creating the SafetyDetect Dataset

Task Definition: In SafetyDetect, an embodied agent is tasked with finding any unsafe or unsanitary conditions in the home, and reporting them to the user. Additionally, the SafetyDetect dataset requires the agent to only report conditions that meet the users preferences. For example, if there are no children in the home, the agent should not report a situation that is only dangerous for children. The agent may also be provided a graph representation of the scene, similar to that which can be created by [32] in the real world, with nodes of the graph being the objects that make up the scene, and edges of the graph being relationships like 'ON,' 'INSIDE,' and 'FACING.'

Category	Class Name	Number of Occurances
Safety	Spills	195
	Tripping Hazard	202
	Broken Items	214
	Candle On	190
	Front Door Open	181
	Stove On	207
Sanitation	Refrigerated/Frozen foods out	210
	Expired Produce	199
	Fridge/Freezer Open	184
Safety for Children	Choking Hazard	201
	Sharp Objects	201
	Poison: Cleaning Products	192
	Poison: Medication & Beauty Products	198

TABLE II

OUTLINE OF ALL OF THE CLASSES OF ANOMALIES IN THE DATASET AS WELL AS THE NUMBER OF TIMES THEY OCCUR IN THE DATASET.

Simulator and Scenes: We use VirtualHome [16] as the basis for the SafetyDetect dataset. VirtualHome was chosen for its ease of use, existing support community, and relative simplicity of adding novel objects. VirtualHome contains seven pre-built scenes, but supports procedural generation to create an unlimited number of valid home environments. For the SafetyDetect dataset we rely on the pre-built scenes, but we release code to allow users to procedurally generate new scenes and data samples with unlimited diversity. The pre-built scenes are of single level homes that generally include at least a kitchen, living room, bedroom, and bathroom with some including two rooms of a single type.

Anomalies and Objects: To create SafetyDetect, we first outlined our target set of hazards that fit the overarching categories of unsafe conditions, unsanitary conditions, and conditions which are dangerous for children. The set of hazards, which are the classes of the dataset, is outlined in Table II, alongside the number of occurrences of each in the dataset. This was created primarily by referencing large scale statistical studies of major household dangers for users or their children. For example, [3] outlines the major causes of death for children in the United States. After filtering out dangers to children not relevant to the household environment (auto accidents), and those we did not feel comfortable addressing (firearm related incidents), we chose to cover the most common remaining examples including choking, poison hazards, and sharp objects. Additional individual examples

Fig. 2. A sampling of images from the SafetyDetect dataset showing unsafe conditions. In one of the images, medication and alcohol are on the floor and dangerous for children. In another, a pile of clothes are in the doorway - a tripping hazard for users.

were created through a small scale user study soliciting examples from prospective users. We specifically interviewed 10 users of ages ranging from 24 to 60. These users were simply asked "If you had a home robot who could patrol your house for anything potentially dangerous, unsanitary, or generally a nuisance, what would you want it to detect?" Some users needed to be provided examples to help spur their creativity. The scenarios provided from these users were then filtered into those feasible to implement in the SafetyDetect dataset. This added classes like 'broken items' which adds the detection of items like shattered glasses or mugs that may have been broken by pets.

To embody and visualize these hazard classes in the simulator, we had to determine what objects were necessary to recreate each hazard *and* where those objects could be placed to constitute the hazard while remaining logically plausible. We began by manually noting obvious examples. For example, a candle flame hazard would require a candle object with an active flame, and a tripping hazard must be on the floor. Many objects we required were not present in the VirtualHome simulator. To address this, we added relevant objects from Google Scanned Objects [39], ReplicaCAD [34], Fantastic Breaks [40], and YCB Objects [41] that fit our pre-defined hazard classes. We additionally created new objects and modified similar objects using Blender to cover cases where suitable objects could not be found. For example, we created a spill object and texture, and re-textured various fruits and vegetables to appear as their expired or rotten counterparts. In total, we add over 30 new objects to VirtualHome while leveraging many objects native to the simulator. To pair each object with locations that result in a dangerous or unsanitary condition, we prompted GPT 3.5 to affiliate a specific object with a location given an anomaly class. This step was primarily done to save manual labor as it could be repeated easily and then verified manually to make sure it matched our expectations instead of completing the work by hand. This step resulted in around 10 creative additions, like having milk on the coffee table as if someone made cereal on the couch, that we determined were valid and included in the dataset.

The final assortment of objects and placement locations results in 967 unique anomalies split among the 13 hazard classes. Through combining these anomalies, countless unique scenes can be created.

Generating the Dataset: To generate the final dataset, we randomly sampled 1000 i.i.d scenarios, each of which utilizes one prebuilt scene, contains 0-5 hazards, and shares a set of user preferences. Each hazard consists of the hazard class, a selected object affiliated with said class, and a valid placement of that object to embody the hazard. The final scene graph after placing the object into the scene in VirtualHome is provided. Future users can visualize our scenarios, add more samples with the prebuilt environments, or procedurally generate new scenes and scenarios. Future users are also encouraged to create new user preferences or classifications of anomalies to better reflect changing household needs. For example, a home-robot user could have a child who is allergic to dairy, and as such having milk left accessible could become a danger. Additionally, as children age, the importance of some dangers may change.

Agent: VirtualHome contains an agent for use in exploring the scene. The agent is embodied as a human but contains a RGB camera who's location on the agent and Field of View (FOV) we can alter. This agent can navigate around the scene using the VirtualHome API. This includes commands for moving towards a room or object, forward in the direction of travel, and turning a certain angle.

B. Using the SafetyDetect Dataset

The central challenge of the SafetyDetect dataset is understanding how to detect unsafe or unsanitary conditions in the home and notify the user of their existence. Our goal is to capture a comprehensive but not necessarily all-inclusive set of these conditions so researchers can test their methods before deploying them into a consumer environment.

Episodes: Each episode in the SafetyDetect dataset instantiates one of the seven base scenes of VirtualHome before extending it with 0-5 unsafe or unsanitary conditions. The agent itself is spawned randomly into the environment with no prior knowledge of it and can explore the environment to find potentially unsafe or unsanitary conditions. We call this set of conditions $\mathcal{A}$, where each individual anomaly is denoted with a_i. An agent is then placed randomly into the scene. Next, we define for the agent the relevant context of the scene, specifically the presence of children in the house. The researcher must then create a strategy for the agent that allows it to detect the anomalies in the home, and reproduce $\mathcal{A}$. Our solution is described in the next section.

Evaluations: We evaluate agents/detection schemes for effectiveness and efficiency. All metrics are reported per episode and are then aggregated across multiple episodes to report the averages and standard errors.

1) **Anomaly Success (AS)**: Fraction of all anomalies found in a given episode. This is essentially the true

positive detection rate.

2) **Conditioned Anomaly** Success (CAS): Fraction of anomalies reported correctly minus those reported incorrectly depending on the given context. For example, in a home without children, a knife being left on the counter is not a problem and should not be reported. Reporting in a situation like this could be annoying to the user, hampering the agent's effectiveness.

The Anomaly Success metric is analogous to the Object Success metric in [18] while Conditioned Anomaly success is a novel metric meant to track how well the agent accommodates to user preferences. Additionally, we track the true positive rate for each anomaly individually. Tracking of false positive and false negative results is also conducted automatically but analysis must be conducted manually. Code released with the SafetyDetect dataset allows users to easily glean this information for their own methods.

IV. Finding Anomalies in the Home

For our proposed method we employ the GPT-4 Large Language Model (LLM) [10] to apply knowledge about what scenarios may be unsafe, unsanitary, or dangerous for children in a scene. Specifically, we hypothesize that LLMs contain domain knowledge about what is safe in a home environment, and what might be dangerous, unsanitary, or dangerous for children. To enable performance from the LLM, we provide two key insights: 1) utilizing a scene graph provides the context needed for scene understanding and reasoning, and 2) classification is an effective approach for detecting specific situations in a scene.

Using the Scene Graph. For the scene graph, we assume that other methods exist, or can be created, to generate a scene graph that represents object relations in the scene. Some examples of these methods include [33], and [32]. The VirtualHome [16] simulator provides a scene graph of this type natively. In the scene graph, each object is a node and the relation is an edge. For example, an apple sitting on a kitchen table would have its initial node with a class of apple, and then two edges, on showing it 'ON' the kitchen table, and another showing it 'INSIDE' the kitchen. The use of the scene graph is imperative to our method and what sets it apart from many other approaches which generally provide either a textual description of the environment, or a listing of classes found in the immediate area [27]. Note that in our exploration, we use the native scene graph and in effect assume perfect perception.

For our LLM prompts, we specifically parsed the scene graph for the edge representing the room the object was inside, and the edge for what the object was on top of. This produced object relation strings like 'apple INSIDE kitchen ON counter' or 'medication INSIDE livingroom ON floor.' These strings were in turn fed directly into GPT-4 alongside the rest of the prompt outlining the classification problem and providing some examples to elicit chain-of-thought.

Classifying the Scene Graph. The second key aspect of our proposed method is classifying relationships in the scene graph as a way of differentiating normal and dangerous situations. Specifically, our prompt for GPT-4 asks it to classify object relations as 'normal', 'unsafe', 'unsanitary', or 'unsafe for children'. This classification approach improved performance as similar methods that asked the LLM to directly detect anomalous object relationships would frequently produce false negatives.

Building the Prompt. We also provide the LLM with a series of example anomalies given as an object relation pulled from the scene graph, followed by the correct classification and an explanation. We employ the popular chain-of-thought prompting technique to provide these examples. The prompt we utilize for our method is below, where [CONTENT] is replaced by the object relations for the LLM to classify:

> ***Q:*** *Classify the following object relations in a home into either 'normal', 'unsafe', 'unsanitary', or 'unsafe for children' - ['medication INSIDE livingroom ON floor', 'rawmeat INSIDE kitchen ON counter']*
> ***A:*** *'medication INSIDE livingroom ON floor': 'unsafe for children' as medication is a poison hazard 'rawmeat INSIDE kitchen ON counter': 'unsanitary' as rawmeat requires refrigeration and will go bad if left unrefrigerated.*
> ***Q:*** *Classify the following object relations in a home into either 'normal', 'unsafe', 'unsanitary', or 'unsafe for children' – [CONTENT]*
> ***A:***

This prompt was chosen as it exhibited the best performance of similar prompts while minimizing the size of the prompt to lower API costs. The overall prompt structure is not particularly sensitive to the specific examples utilized with many other iterations producing the same or only marginally worse performance.

Parsing and Evaluating the Response. To parse through the verbose responses, we again prompt a LLM, this time GPT-3.5 Turbo for cost savings with minimal performance loss. We ask the model to simplify the previous response into a one word classification fitting the categories above. For example, a response including "'medication INSIDE livingroom ON floor': 'unsafe for children' as it presents a possible poison hazard," would be filtered into just 'unsafe for children.' This allows us to evaluate the models performance effectively despite the verbose responses provided through chain-of-thought prompting. This parsing method injected a small amount of error into the system, with around 1-2% of responses being improperly parsed in our method. Despite this, we found this method to increase performance over both removing the chain-of-thought and directly outputting a label and directly parsing the response for the label.

V. Experiments and Results

A. Baseline Results

Use of The Scene Graph and Classification. We first tested against the use of the scene graph and classification techniques in the prompts. The results of these tests can

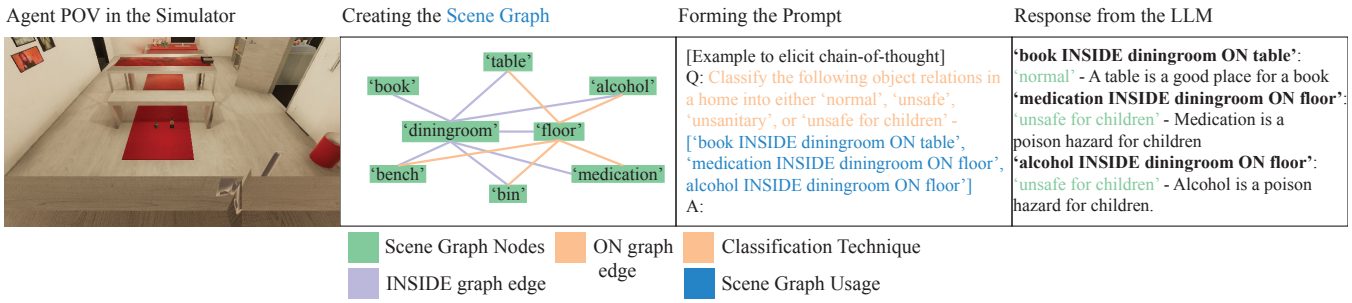

Fig. 3. The flow of our method is depicted here. We first get the scene graph before using it to formulate a prompt which asks the LLM to classify the object relations. The model must then use commonsense reasoning to classify these object relations effectively.

Method	AS ↑	CAS ↑
Object List + Ex, w/ Cla	12.4	5.1
Scene Description + Ex, w/ Cla	21.5	13.6
SG, no Cla	18.8	-
SG + Ex, no Cla	38.8	-
SG & Cla	83.2	80.8
SG + Ex & Cla	94.0	88.8
CoT + SG & Cla (Ours)	**96.0**	**90.5**

TABLE III

COMPARISON OF PERFORMANCE ON SAFETYDETECT. EX REPRESENTS THE INCLUSION OF EXAMPLE, SG REPRESENTS THE USE OF THE SCENE GRAPH, CLA REPRESENTS THE INCLUSION OF CLASSIFICATION. NOTE THAT THE METHODS THAT UTILIZE THE SCENE GRAPH AND CLASSIFICATION ARE SIGNIFICANTLY MORE COMPETITIVE THAN THOSE THAT DO NOT.

Method	AS ↑	CAS ↑
FLAN-T5-Large	81.3	51.2
GPT-3.5-Turbo	94.0	88.8
GPT-4 (Ours)	**96.0**	**90.5**

TABLE IV

COMPARISON OF PERFORMANCE ON SAFETYDETECT FOR DIFFERENT LARGE LANGUAGE MODELS. LLAMA IS NOT INCLUDED AS WE COULD NOT GET IT TO WORK WELL ON THIS TASK.

be seen in Table III. We find that when providing scene information to the LLMs, standard techniques like using object lists or a scene descriptions (first two rows) had difficulty understanding when an anomaly was present. For example, when providing a simple object list, its difficult for the model to know whether an object is in a safe space or not. As such, it tended to err on the side of reporting normalcy. It did relatively well with specific cases like spills or rotten fruit as the class name was sufficient to detect an anomaly. The scene description-based prompts performed better, with a 21.5% detection rate, but it still struggled to pick out dangerous and unsanitary conditions. It performed similarly to the object list method for objects whose class names alone were enough to detect the anomaly and added some performance in other cases like sharp items and refrigeration required items.

When utilizing the scene graph as the basis for communicating the scene context, performance nearly doubles when continuing to provide examples but removing classification. When adding classification back in, performance doubles again to a nearly 4.5× improvement over the scene description-based method for a 94% detection rate. Reformatting the prompt to fit a chain-of-thought scheme further boosted the performance slightly to a 96% detection rate.

Upon further inspection, the logic provided by GPT-4 in response to our method was consistent with real-world commonsense. Anomalies were not only correctly classified, but the logic behind their classification matched that in expert sources and from our sample users. In fact, it found anomalies built into the scene graph that we had not placed there, but were potential hazards. Some scenes were built in VirtualHome through stacking objects on top of each other to create a visually new object, but the scene graph retained this stacked structure and produced illogical relationships. Our method would consistently pick out these scenarios as anomalous. As the underlying structure of the simulator had many of these anomalies, it became difficult to track a false positive rate. Anecdotally, we saw very few false positives in the our method's responses.

Ablation on Models. We also test against the specific use of GPT-4 by deploying the same Chain-of-Thought + Scene Graph & Classification prompt on GPT-3.5-Turbo, FLAN-T5-Large, and LLaMA-2. Our results are shown in Table IV. We find that GPT-3.5 produces a similar level of performance to GPT-4 with FLAN-T5-Large also producing respectable detection rates above 80%. We had difficulty applying LLaMA to this task as it frequently hallucinated new object relations that we did not provide. This made parsing the response increasingly difficult and severely impacted performance. Further experiments with different prompting approachesay be able to improve upon these LLaMA results.

B. Real World Experimentation

The goal of our dataset is to enable new use cases in home robotics. As such, it is important to validate the use of the scene graph in the real world and explore how best to create the scene graph for our purposes. For this experiment, we set

up a ClearPath TurtleBot to navigate the local environment while running a slightly modified version of [32] that builds scene graphs similar to that from VirtualHome for use alongside our CoT + SG & Cla method.

Our real world scene consisted of one room with a small number of objects, including 1-3 anomalies. The TurtleBot captures images from 360° and feeds them into our modified version of [32] that produces an adequate scene graph. We then run our algorithms verbatim on the scene graph. We conduct 20 experiments that covered a subset of eight of our anomaly classes in the SafetyDetect task. We acknowledge that this scenario is a greatly simplified version of a real-world home which may include clutter, obscured objects, additional rooms, and novel objects. These experiments are intended as a proof of concept that the sim2real transfer of our method can work in these simplified scenarios.

We find that performance is similar between the simulator and the real world. Creating the scene graph through [32] also enabled increased performance on some anomalies, such as obstructions, which were not captured well in the scene graph created by the VirtualHome simulator. Some anomalies suffered worse performance as, for example, the scene graph generator we utilized was often unable to detect spills. We believe that many of these issues can be solved by retraining the object detection components of the scene graph builder, or through future scene-graph builders employing VLMs or open-vocabulary models. Overall, our detection rate dropped to 89.1%. Through this experiment, we believe that if provided with a ground truth scene graph, there is little reason why our LLM-based approach would not be able to sustain our simulator-based detection rates of 96% with few false positives. To attempt to compare more directly to our simulated results, we fine-tune [32] directly on the simulator version of SafetyDetect to create an identical scene-graph to that we created in the real world. On the same number of tasks and with the same subset of our dataset, we found that our method utilizing the scene graph from [32] in the simulator performed worse than its real world counterpart with an overall detection rate of 82.1%. Qualitatively evaluating these results reveals that this version of our method primarily missed anomalies due to two main phenomenon: a sim2real gap with the data [32] was initially trained on, resulting in some objects not being perceived properly, and the frontier-based exploration navigation scheme we implemented not producing views of the anomalies that were conducive to them being detected.

Fig. 4. We deploy a ClearPath TurtleBot in the real world. We use existing methods to generate an effective scene graph before utilizing our method to detect anomalies in the scene. In this example, medication and cleaning products are on the table. This is captured in the scene graph and detected by the model as unsafe for children.

VI. Conclusions, Limitations, and Future Work

Home robots intend to make their users lives easier. One way they may do this is by informing users of issues or anomalies in the home. In this work, we moved towards enabling home robots with these abilities by creating a new dataset, built on top of the popular VirtualHome platform, that contains 1000 dangerous or unsanitary scenarios for an agent to detect. We also propose an LLM-based approach that utilizes a scene graph and classification approach. This methods identifies over 90% of anomalous scenarios in our dataset. Additionally, we show that these techniques remain viable in the real world.

Limitations. The methods we proposed are limited by perception, specifically how the scene graph is created. For example, in the scene graph created by the VirtualHome simulator, there was nothing indicating obstructions in the scene. To illustrate this, a relation of 'box INSIDE livingroom ON floor' is okay, but physically the box may be in the doorway creating a tripping hazard. While we assumed perfect perception in creating the scene graph in the simulator, alternate approaches without this assumption could produce weaker results. Similarly, real-world scenes are likely to have significant amounts of clutter, obscured objects, and novel objects for which the perception is likely to be a significant challenge.

Future Work. We leave the topic of how best to explore and perceive the VirtualHome environment and subsequent home environments, as well as how to quantify this exploration, to future work. Additionally, testing in a more complex real world environment with occlusions and clutter would be a valuable extension of this work. Continuation work with multi-modal LLMs like GPT-V could be valuable to both address perception issues and solve the anomaly detection problem in one shot. The next logical step in this work is to get the agent to preemptively solve the anomaly by adding on a task planning and completion task. Part of our reason for using VirtualHome is the relative ease of expanding our dataset to new tasks and research into the necessary computer vision subtasks. For the real world deployment of this work, significant effort needs to be put into optimizing prompts and altering the scheme to lower the costs of querying the LLM. Also valuable would be a further exploration of what additional knowledge about the scene could be encoded into a scene graph to allow for even better detection results in the real-world.

REFERENCES

[1] M. Ahrens, "Home cooking fires." [Online]. Available: https://www.nfpa.org/News-and-Research/Data-research-and-tools/US-Fire-Problem/Home-Cooking-Fires

[2] [Online]. Available: https://www.nsc.org/work-safety/safety-topics/slips-trips-and-falls

[3] J. E. Goldstick, R. M. Cunningham, and P. M. Carter. [Online]. Available: https://www.nejm.org/doi/full/10.1056/NEJMc2201761

[4] Z. Hu, J. Pan, T. Fan, R. Yang, and D. Manocha, "Safe navigation with human instructions in complex scenes," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 753–760, 2019.

[5] V. S. Dorbala, A. Srinivasan, and A. Bera, "Can a robot trust you?: A drl-based approach to trust-driven human-guided navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3538–3545.

[6] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, and R. J. Mooney, "Improving grounded natural language understanding through human-robot dialog," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6934–6941.

[7] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *Conference on Robot Learning*. PMLR, 2020, pp. 394–406.

[8] X. Gao, Q. Gao, R. Gong, K. Lin, G. Thattai, and G. S. Sukhatme, "Dialfred: Dialogue-enabled agents for embodied instruction following," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 049–10 056, 2022.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[10] OpenAI, "Gpt-4 technical report," 2023.

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[12] X. Jiang, G. Xie, J. Wang, Y. Liu, C. Wang, F. Zheng, and Y. Jin, "A survey of visual sensory anomaly detection," 2022.

[13] P. Bergmann., X. Jin., D. Sattlegger., and C. Steger., "The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*, INSTICC. SciTePress, 2022, pp. 202–213.

[14] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6536–6545.

[15] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1975–1981.

[16] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[17] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.

[18] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," in *European Conference on Computer Vision*. Springer, 2022, pp. 355–373.

[19] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023.

[20] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 1507–1514, Aug. 2011. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/7979

[21] J. S. Park, B. Jia, M. Bansal, and D. Manocha, "Efficient generation of motion plans from attribute-based natural language instructions using dynamic constraint mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6964–6971.

[22] R. Paul, J. Arkin, N. Roy, and T. M Howard, "Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators," *Robotics: Science and Systems*, 2016.

[23] 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.

[24] S. M. Lukin, R. Sharma, and M. Bellissimo, "Learning to understand anomalous scenes from human interactions," 2023.

[25] S. M. Lukin and R. Sharma, "Anomaly detection with visual question answering."

[26] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *International Conference on Learning Representations*, 2022.

[27] V. Sashank Dorbala, J. Mullen, James F., and D. Manocha, "Can an Embodied Agent Find Your "Cat-shaped Mug"? LLM-Based Zero-Shot Object Navigation," *arXiv e-prints*, p. arXiv:2303.03480, Mar. 2023.

[28] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," *arXiv preprint arXiv:2209.11302*, 2022.

[29] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[30] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[31] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[32] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 670–685.

[33] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5410–5419.

[34] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," Nov. 2019.

[35] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," 2023.

[36] T. Bratanic, "Knowledge graphs; llms: Multi-hop question answering," Jun 2023. [Online]. Available: https://neo4j.com/developer-blog/knowledge-graphs-llms-multi-hop-question-answering/

[37] J. Zhang, "Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt," *arXiv preprint arXiv:2304.11116*, 2023.

[38] 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 task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[39] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," 2022.

[40] N. Lamb, C. Palmer, B. Molloy, S. Banerjee, and N. K. Banerjee, "Fantastic breaks: A dataset of paired 3d scans of real-world broken objects and their complete counterparts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[41] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-CMU-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, sep 2015.