Project Group/Seminar/Lab WT/ST [Year] Title

Student Name
Matriculation No: 0000000

Supervisor: Supervisor Name

January 18, 2026

Abstract

To fully leverage mobile manipulation robots, agents must autonomously execute long-horizon tasks in large, unexplored environments [1]. This paper addresses the challenge of interactive object search, where robots must navigate, explore, and manipulate objects (e.g., opening doors and drawers) to find targets in partially observable settings [2]. We propose MoMa-LLM, a novel approach that grounds Large Language Models (LLMs) within structured representations derived from open-vocabulary scene graphs, which are dynamically updated during exploration [3]. By tightly interleaving these representations with an object-centric action space, the system achieves zero-shot, open-vocabulary reasoning [4]. To rigorously benchmark performance, we introduce a novel evaluation paradigm utilizing full efficiency curves and the Area Under the Curve for Efficiency (AUC-E) metric [5]. Extensive experiments in simulation and the real world demonstrate that MoMa-LLM substantially improves search efficiency compared to state-of-the-art baselines like HIMOS and ESC-Interactive [6].

1 Introduction

1.1 Motivation and Challenge

Interactive embodied AI tasks in large, human-centered environments require robots to reason over long horizons and interact with a multitude of objects [7]. In many real-world scenarios, these environments are a priori unknown or continuously rearranged, making autonomous operation significantly more difficult [8]. Specifically, the task of interactive object search presents unique challenges because objects are not always openly visible; they may be stored inside receptacles like cabinets or drawers, or located behind closed doors [9]. Consequently, an agent cannot rely on directional reasoning alone but must actively manipulate the environment—opening doors to navigate and searching through containers—to succeed [10].

1.2 Limitations of Existing LLM Methods

Recent advancements have shown the potential of Large Language Models (LLMs) for generating high-level robotic plans [11]. However, existing methods such as SayCan or SayPlan are insufficient for interactive search in unexplored settings for several reasons:

- **Assumption of Full Observability:** Many prior works focus on fully observed environments, such as tabletop manipulation or pre-explored scenes [12].
- **Scalability and Hallucination:** In large, partially observable scenes with numerous objects, simply providing an LLM with raw observations or lists of objects increases the risk of generating impractical sequences or hallucinations [13]. Simple prompting strategies or raw JSON inputs of full scene graphs have proven insufficient for complex reasoning in these contexts [14].
- **Limited Scope:** Methods often restrict tasks to single rooms or rely on non-interactive navigation, failing to address the complexities of multi-room exploration and physical interaction [15].

1.3 Proposed Solution

To address these limitations, the authors propose MoMa-LLM, a method that grounds LLMs in dynamically built scene graphs [16]. This approach utilizes a scene understanding module that constructs open-vocabulary scene graphs from dense maps and Voronoi graphs as the robot explores [17]. By extracting structured and compact textual representations from these dynamic graphs, the system enables pre-trained LLMs to plan efficiently in partially observable environments [18]. This allows the robot to perform zero-shot, open-vocabulary reasoning, extending readily to a spectrum of complex mobile manipulation tasks [19].

2 Related Work

Here is the summary of the related work categorized by the gaps MoMa-LLM addresses:

2.1 3D Scene Graphs

- **Existing Approaches:** 3D scene graphs abstract dense maps into hierarchical, object-centric representations, with works like Hydra focusing on representing dynamically changing scenes [1]. Other approaches, such as ConceptGraphs and VoroNav, investigate zero-shot perception inputs for task planning [2].
- **MoMa-LLM Differentiation:** While previous works utilize scene graphs for reasoning, they have not realized object navigation using both dynamic and interactive scene graphs [3]. MoMa-LLM constructs open-vocabulary scene graphs dynamically from dense maps and Voronoi graphs, tightly interleaving them with an object-centric action space as the environment is explored [4].

2.2 LLMs for Planning

- **Existing Approaches:** Models like LLM-Planner use information retrieval for planning, while SayPlan focuses on identifying subgraphs within large, known scene graphs [5]. Most efforts focus on fully observed environments (e.g., table-top manipulation) or a priori explored scenes [6].
- **MoMa-LLM Differentiation:** Current methods struggle to generate grounded, executable plans for partially observed or unexplored environments [7]. MoMa-LLM addresses this by grounding LLMs in dynamically built graphs, enabling reasoning over long horizons in fully unexplored scenes where simple prompting strategies are insufficient [8].

2.3 Object Search

- **Existing Approaches:** Classical methods include frontier exploration and reinforcement learning, while recent semantic methods like ESC use language models to score frontiers based on object co-occurrences [9]. SayNav utilizes scene graphs but relies on non-interactive search and hardcoded heuristics for navigation [10].
- **MoMa-LLM Differentiation:** Unlike methods that rely on pairwise scoring or directional reasoning, MoMa-LLM treats search as a planning problem where the full scene is encoded jointly [11]. It specifically fills the gap of interactive search, allowing the agent to reason about manipulating the environment (e.g., opening doors or receptacles) to find objects that are not freely accessible [12].

2.4 Structured Knowledge Representation in MoMa-LLM

To effectively use Large Language Models (LLMs) for robotic planning, MoMa-LLM converts complex 3D scene graphs into a structured textual format. This process bridges the gap between the robot's geometric understanding and the LLM's reasoning capabilities, focusing on three properties: grounding (adherence to physical reality), specificity (avoiding irrelevant context), and open-set reasoning (handling unknown semantics) [1].

The representation is composed of four key elements:

2.4.1 Scene Structure Encoding

Rather than feeding raw data (like JSON) to the LLM, the system extracts a structured list of rooms and objects designed for compact text encoding [2].

- **Distance Abstraction:** Exact metric distances are binned and mapped to natural language adjectives (e.g., "near", "far") to facilitate easier reasoning for the LLM [3].
- **Object States:** The state of an object is encoded directly into its name (e.g., "opened door" or "closed fridge") to provide immediate context on interactability [4].
- **Summarization:** To keep the context concise, matching nodes are summarized with counters, and open doors that do not provide new connectivity are filtered out [5].

2.4.2 Handling Partial Observability

Since the environment is initially unknown, the representation explicitly encodes "frontiers" (boundaries between explored and unexplored space) to enable exploration-exploitation reasoning [6].

- **Frontier Types:** The system differentiates between two types of unknown space:
  - "Unexplored Area": Space within a room (e.g., occluded space behind furniture) [7].
  - Leading Out: Frontiers that lead to entirely new areas or rooms, which are listed separately [8].
- **Affordances:** Instead of explicitly listing what every object can do, the system relies on the LLM to infer affordances (e.g., that a fridge can be opened) from the object descriptions and states [9].

2.4.3 Dynamic History Realignment

To maintain a valid history in a constantly changing map (where a "living room" might later be reclassified as a "kitchen"), the system uses a novel realignment mechanism [10].

- **Position Tracking:** The system tracks the physical interaction positions of previous actions [11].
- **Retrospective Updates:** When constructing the prompt, previous actions are re-mapped to the current scene graph. For example, if a room was previously called "living room" but is now classified as "kitchen," the history will display `explore(kitchen)` instead of the original `explore(living room)` [12].

2.4.4 Re-trial and Feedback

To handle failures without overwhelming the context window, the system provides simplified feedback.

- **State Feedback:** The LLM receives simple status updates like "success", "failure", or "invalid argument" [13].
- **Replanning:** If a command is syntactically invalid or infeasible, the LLM is prompted to try again immediately. If a physical action fails during execution, the scene is re-encoded with the new state, and the LLM makes a fresh decision [14].

3 Paper Summary

3.1 Problem Statement

The authors address the challenge of embodied reasoning where a robotic agent must locate specific objects within large, unexplored, and human-centric environments [1]. To succeed, the agent must simultaneously perceive the environment to build a representation and reason about exploration and interaction (e.g., opening doors or drawers) to complete the task [2].

3.1.1 Formalization

The problem is formalized as a Partially Observable Markov Decision Process (POMDP), denoted as a tuple $\mathcal{M} = (S, A, O, T(s'|s, a), P(o|s), r(s, a))$ [3]. The components are defined as follows:

- **Goal (g):** A natural language description of the task the agent must complete [4].
- **State Space (S) & Action Space (A):** The underlying states and available actions for the agent [5].
- **Observation Space (O):** The agent's current observation o, consisting of a posed RGB-D frame I_t [6].
- **Transition Probability (T):** Defined as $T(s'|s, a)$, representing the probability of moving from state s to s' given action a [7].

- **Observation Probability (P):** Defined as $P(o|s)$, representing the likelihood of receiving observation o in state s [8].
- **Reward Function (r):** Defined as $r(s,a)$, providing feedback on the action taken [9].

3.1.2 Task: Semantic Interactive Object Search

The paper introduces the task of Semantic Interactive Object Search. In this scenario:

- **Objective:** The agent must find a target object category (e.g., a specific food item or tool) within an indoor environment [10].
- **Constraints:** The environment is initially unexplored, and objects may not be openly visible [11].
- **Interaction:** The task requires physical manipulation, such as opening doors to navigate through rooms and searching inside receptacles like cabinets and drawers to reveal hidden objects [12].
- **Success Condition:** The task is considered successful only if the agent observes an instance of the target category and explicitly executes the `done()` command [13].

3.1.3 Extension of Schmalstieg et al. [7]

This work builds directly upon the "interactive search task" introduced by Schmalstieg et al. (referenced as [7] in the source text) [14]. The authors extend this baseline in several critical ways to increase complexity and realism:

- **Semantic vs. Random Placement:** While Schmalstieg et al. focused on random target placements, this work introduces a semantic variation [15]. The environment maintains realistic semantic co-occurrences (e.g., specific objects appearing near related objects), allowing the agent to use visible objects as cues to locate the target [16].
- **Scale and Complexity:** The task is expanded to include a much larger number of objects and receptacles compared to the restricted set used in the previous work [17].
- **Object Relations:** It incorporates a prior distribution of realistic room-object and object-object relations, meaning the presence of certain furniture or items informs the probability of the target's location [18].

3.2 Approach: MoMa-LLM

To enable high-level reasoning, we construct a hierarchical scene graph $\mathcal{G}_S$ that abstracts the environment into room and object-level entities, grounded by a fine-grained navigational graph. The construction process consists of three stages: dynamic mapping, topological graph generation, and semantic classification.

3.2.1 Dynamic RGB-D Mapping

The foundation of the scene representation is a semantic 3D map built from the robot's onboard perception.

- **Voxel Grid Construction:** The agent processes a stream of posed RGB-D frames $\{I_0, ..., I_t\}$ containing semantic information [1]. The point clouds from these frames are transformed into a global coordinate frame and arranged into a dynamic 3D voxel grid $\mathcal{M}_t$ [2]. This grid is updated continuously as the agent explores new areas or observes dynamic changes [3].
- **Occupancy Map:** To facilitate navigation, we derive a two-dimensional bird's-eye-view (BEV) occupancy map $\mathcal{B}_t$ from the 3D grid [4]. This is achieved by analyzing "stixels" (vertical columns of voxels) in $\mathcal{M}_t$; we identify the highest occupied entry per stixel to infer obstacle positions, walls, and explored free space $\mathcal{F}_t$ [5].

3.2.2 Voronoi Graph (GVD)

We abstract the dense metric map into a sparse topological graph $\mathcal{G}_{\mathcal{V}}$ to support efficient navigation and spatial reasoning.

- **Computation:** We first inflate the obstacles in the BEV map $\mathcal{B}_t$ using a Euclidean Signed Distance Field (ESDF) to ensure robustness [6]. Based on this inflated map, we compute a Generalized Voronoi Diagram (GVD) [7]. The GVD consists of a set of points (nodes) that have equal clearance to the closest obstacles [8]. We filter this diagram by excluding nodes that are in the immediate vicinity of obstacles or outside the explored bounds $\mathcal{B}_t$ [9].
- **Navigation Utility:** The resulting nodes and edges form the navigational Voronoi graph $\mathcal{G}_{\mathcal{V}} = (\mathcal{V}, \mathcal{E})$ [10]. We extract the largest connected component to serve as the robot-centric navigation graph, sparsifying it to reduce the node count [11]. This graph allows the robot to navigate robustly by maximizing clearance from obstacles.

3.2.3 Scene Graph & Room Classification

The final layer, the 3D Scene Graph $\mathcal{G}_S$, clusters the Voronoi graph into semantic regions (rooms) and assigns objects to them.

- **Room Clustering (Door-Based Separation):** Unlike previous methods that rely on geometric constrictions (which fail in open layouts), we separate the global Voronoi graph $\mathcal{G}_{\mathcal{V}}$ into distinct room regions $\mathcal{G}_{\mathcal{V}}^R$ based on detected doors [12].
  - We model the probability distribution of observed door positions using a mixture of Gaussians: $\rho_{\mathcal{N}}(x, H) = \frac{1}{N_D} \sum_{i=1}^{N_D} K_H(x - x_i)$, where x_i are door coordinates and H is the bandwidth matrix [13].

    - Edges and nodes of the Voronoi graph that fall into high-probability door regions are removed, effectively cutting the graph into disjoint components representing distinct rooms [14].
    - Connectivity between these rooms is re-established by calculating shortest paths between the disjoint components; if a path traverses exactly two components, they are marked as neighbors [15].

- **Object Assignment:** Objects are assigned to rooms by minimizing a weighted travel distance. For an object o, we identify the Voronoi node n_o that minimizes the path length to the viewpoint v_p from which the object was seen, weighted by the Euclidean distance to the object (Eq. 2 in the paper) [16]. This prevents erroneous assignments through walls [17].

- **LLM-Based Room Classification:** Once rooms are clustered and objects are assigned, we employ an LLM for open-set classification [18].
    - As shown in Fig. 3, the LLM is provided with a list of object categories contained within each room cluster (e.g., "armchairs, carpet, table-lamp" for room-0) [19].
    - The LLM analyzes these object compositions to infer the semantic room type (e.g., classifying room-0 as a "living room" and room-1 as a "bedroom") [20].
    - This classification is performed dynamically at each high-level policy step as the scene evolves [21].

3.2.4 Grounded High-Level Planning

The core contribution of MoMa-LLM lies in its ability to ground a Large Language Model (LLM) within a dynamically evolving scene graph. Rather than feeding raw data or unstructured lists to the model, the system extracts structured knowledge to bridge the gap between the robot's perception and the LLM's reasoning capabilities [1]. This process ensures the planning is grounded in physical reality, specific enough to avoid hallucinations, and open-set to handle unknown environments [2].

The Prompt Design The text-based scene representation is wrapped in a carefully engineered prompt structure designed to elicit logical planning from the LLM. As illustrated in Fig. 4, the prompt consists of several distinct components [8]:

- **System & Task:** Clearly defines the agent's role (e.g., "You are a robot in an unexplored house") and the specific goal (e.g., "Find a stove") [9].

- **Skill API:** explicitly lists the available high-level function calls the robot can execute. These include `Maps(room, object)`, `go_to_and_open(room, object)`, `close(room, object)`, `explore(room)`, and `done()` [10].

- **Scene Context:** The structured knowledge extracted from the scene graph (as described above) is inserted here, detailing the current room, observed objects, and unexplored frontiers [11].

- **Analysis & Reasoning (Chain of Thought):** The prompt enforces a "Chain-of-Thought" process. It requires the LLM to first output an Analysis of where the target might be found and the necessary actions, followed by Reasoning to justify the specific next step, before finally generating the Command (function call) [12].

Handling History with Dynamic Re-alignment A major challenge in dynamic scene graph planning is that the environment representation changes over time—rooms may be reclassified, or boundaries may shift as new areas are revealed [13]. This makes a static history of previous actions invalid or confusing. MoMa-LLM addresses this via Dynamic Re-alignment:

- **Tracking Positions:** Instead of just memorizing the text of previous commands, the system tracks the physical interaction positions of past actions [14].
- **Re-mapping:** Before each new query to the LLM, the system re-aligns these past positions to the current state of the scene graph. It matches the old positions to the currently valid Voronoi nodes and room labels [15].
- **Example:** If the robot previously executed `explore(living room)`, but identifying a fridge later causes that area to be reclassified as a kitchen, the history presented to the LLM in the next step will automatically update to `explore(kitchen)` [16].

This ensures the LLM always acts on a consistent, up-to-date view of the world, preventing logical inconsistencies caused by the evolving map.

4 Experiments

4.1 Experimental Setup

The authors evaluated MoMa-LLM in both extensive simulation environments and a physical real-world deployment.

- **Simulation:** Experiments were conducted using the iGibson simulator, utilizing 15 interactive scenes based on real-world scans [1]. The agent controlled a Fetch robot to navigate and interact with these environments [2].
- **Real-World:** The approach was transferred to a Toyota HSR robot equipped with an RGB-D camera and LiDAR [3]. The testing environment was an apartment comprising four rooms (kitchen, living room, hallway, and bathroom) with 54 object categories [4].

4.2 Evaluation Metrics

The evaluation employed standard metrics such as Success Rate (SR) and Success weighted by Path Length (SPL) [5]. However, the authors strongly emphasized a novel metric designed to address limitations in existing evaluation paradigms:

- **Area Under the Efficiency Curve (AUC-E):** The authors argue that standard metrics rely on arbitrary time budgets (e.g., a maximum number of steps), which fail to distinguish between agents that search thoroughly but slowly versus those that are fast but less comprehensive [6].
- **Rationale:** To resolve this, they calculate an efficiency curve plotting success rates against varying time budgets [7]. The AUC-E distills this curve into a single number, providing a robust measure of search efficiency that is independent of arbitrary cutoffs [8].

4.3 Baselines

MoMa-LLM was compared against a diverse set of baselines ranging from heuristics to learning-based methods:

- **Random & Greedy:** Heuristic baselines that select actions uniformly or based on distance [9].
- **ESC-Interactive:** An extension of the "Exploration with Soft Commonsense Constraints" approach, which scores frontiers based on object co-occurrences [10].
- **HIMOS:** A hierarchical reinforcement learning approach adapted for interactive search [11].
- **Unstructured LLM:** A baseline using the same LLM but provided with a raw JSON scene graph rather than the structured, language-grounded representation proposed by the authors [12].

4.4 Results

4.4.1 Simulation Results (Table I)

MoMa-LLM demonstrated superior performance across all metrics, proving to be the most efficient and effective method.

- **Top Performance:** MoMa-LLM achieved the highest Success Rate (97.7%), SPL (63.6), and AUC-E (87.2) [13].
- **Comparison:** It significantly outperformed the Unstructured LLM (SR 86.3%, AUC-E 77.6%), highlighting that simply feeding a scene graph to an LLM is insufficient without structured grounding [14]. The Unstructured LLM also generated significantly more invalid actions (0.41 vs 0.19) [15].
- **Efficiency:** While HIMOS achieved a high success rate (93.7%), it lacked efficiency (SPL 48.5) compared to MoMa-LLM [16]. ESC-Interactive performed well but struggled to optimize long-horizon sequences compared to the planning capabilities of the LLM [17].

4.4.2 Real-World Results (Table III)

The system successfully transferred to the physical robot with consistent results.

- **Success:** Both MoMa-LLM and the strongest baseline, ESC, achieved an 80% success rate (8/10 episodes) [18].
- **Efficiency Gains:** MoMa-LLM was drastically more efficient, traveling nearly half the distance of ESC (17.9m vs. 33.9m) and requiring fewer object interactions (2.2 vs. 3.5) to locate the target [19].
- **Conclusion:** The results confirm that MoMa-LLM's structured knowledge representation enables more target-driven behavior, reducing unnecessary exploration and interaction compared to co-occurrence or heuristic methods [20].

5 Discussion

The proposed MoMa-LLM framework presents a significant advancement in semantic interactive object search by grounding Large Language Models (LLMs) in dynamic scene graphs. While the approach demonstrates impressive zero-shot capabilities and efficient search strategies, a critical analysis reveals several limitations regarding its reliance on ideal perception and the computational overhead of its reasoning modules.

5.1 Strengths

- **Robust Open-Vocabulary Reasoning:** A primary strength of the system is its ability to perform zero-shot, open-vocabulary reasoning, allowing it to adapt to novel semantic categories without retraining [1]. This capability enables the system to outperform state-of-the-art baselines like HIMOS and ESC-Interactive by effectively leveraging the "accumulated knowledge about the human world" contained within pre-trained LLMs [2].
- **Real-World Transferability:** The system demonstrates strong potential for zero-shot transfer from simulation to the real world. By separating high-level reasoning from low-level execution, the authors achieved an 80% success rate in real-world experiments using a Toyota HSR robot, despite the scene layout and sub-policies differing significantly from the simulation environment [3].
- **Resilience to Segmentation Errors:** The policy exhibits robustness against imperfect room segmentation. Although the door-based room separation algorithm tends to under-segment areas—particularly in "open room concepts" like combined kitchen-living rooms—the agent's reasoning capability allows it to function effectively even when objects from multiple functional areas are clustered into a single room node [4].

5.2 Weaknesses and Limitations

- **Reliance on Ground Truth Perception:** A significant limitation of the current implementation is its dependence on "ground truth perception for semantic masks, depth, localization and handle detection" [5]. In real-world deployment, sensor noise and occlusion would likely degrade the accuracy of the dynamic scene graph, potentially leading to planning failures that are not accounted for in the simulation results. The authors acknowledge that extracting meaningful feedback for failure analysis in real-world robots remains an open problem [6].

- **Difficulties with Open-Room Layouts:** The scene graph construction relies on a door-based separation method, which struggles with open floor plans. The evaluation shows that this method is "prone to open room concepts," leading to under-segmentation where distinct functional areas are merged [7]. While the policy is robust to this, the segmentation accuracy itself drops significantly; for example, in simulation, the average room category accuracy was only 27.6% due to these layout ambiguities [8].

- **Computational Latency:** The system incurs a high computational cost, largely driven by the LLM reasoning steps. An analysis of runtime components reveals that "LLM queries for high-level reasoning take up the majority" of the time in simulation [9]. Furthermore, the current implementation fully recomputes the scene graph at every time step rather than performing incremental updates, which is inefficient for long-horizon tasks [10].

- **Limited Feedback Modality:** The feedback mechanism provided to the LLM is textually sparse (e.g., "success" or "failure") [11]. The system lacks detailed visual feedback, which prevents the planner from understanding why a specific action failed (e.g., a gripper slipping off a handle vs. a locked door), thereby limiting its ability to recover from complex physical failures [12].

5.3 Future Work

To address these limitations, the authors propose several avenues for future research:

- **Integration of Vision-Language Models (VLMs):** The authors suggest that replacing the current adjective-based distance encodings with "vision-language models is very promising" [13]. This would allow for a more direct incorporation of dense spatial and geometric information into the reasoning process.

- **Relaxing Perception Assumptions:** Future iterations aim to move away from ground-truth reliance by "fully constructing scene graphs from noisy sensor inputs," potentially utilizing open-vocabulary representations directly from raw sensor data [14].

- **Holistic Room Clustering:** To better handle non-standard layouts, the authors point toward "more holistic approaches to incorporate spatial and semantic details in room clustering" rather than relying solely on geometric constrictions like doors [15].
- **Enhanced Visual Feedback:** Improving robustness will require incorporating "more detailed visual feedback for the identification of object states and failure reasons," allowing the agent to distinguish between different types of interaction failures [16].

6 Conclusion

In this work, we presented MoMa-LLM, a novel and unified framework designed to ground Large Language Models (LLMs) within dynamically constructed, open-vocabulary scene graphs. By tightly interleaving high-level semantic reasoning with object-centric low-level policies, our approach allows mobile manipulation robots to effectively explore, navigate, and interact with large, initially unknown environments. We demonstrated that extracting structured knowledge—incorporating spatial constraints and exploration history—is critical for enabling LLMs to generate feasible and efficient plans in complex real-world scenarios.

Through extensive evaluation in both simulation and the real world, we showed that MoMa-LLM significantly outperforms conventional heuristics, reinforcement learning methods, and unstructured LLM baselines in terms of search efficiency and decision-making capabilities. Furthermore, our results highlight the framework's flexibility, proving it is readily extendable to abstract "fuzzy" search tasks and general household activities beyond simple object retrieval.

7 LaTeX Example Usage

This section contains general example about how to use LaTeX. You can find examples for floating environments, figures, equations, algorithms, and tables.

7.1 Equations

Use equations as shown below.

$$a^2 + b^2 = c^2 \tag{1}$$

Use Refs as this: The Eq. (1) is the Pythagorean theorem.

You can use section refs the same way: This belongs to Sec. **??**. Use at least two structural elements at the same level.

7.2 Figures

Use figures as shown in Fig. 1.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Figure 1: This figure is taken from [1].

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

7.3 Tables

Use tables as shown below.

Table 1: Average runtime.

First	Second	Third	Fourth
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.4 Algorithms

Use tables as shown below.

Algorithm 1: Querying placement pose while object is visible in scene

```
Input  : Object identifier $objectID$
Output: Placement pose $placePose$.
$n \leftarrow 0$
while $objectIsVisible = True$ do
    $placePose \leftarrow getPlacePose(objectID)$
    $n \leftarrow n + 1$
    return $placePose$
```

References

[1] Wikipedia, “Schnabeltier,” http://de.wikipedia.org/wiki/Schnabeltier, accessed on 7/10/2012.