STREAM: Embodied Reasoning through Code Generation

Daniil Cherniavskii [1] **Phillip Lippe** [1] **Andrii Zadaianchuk** [1] **Efstratios Gavves** [1]

Abstract

Recent advancements in the reasoning and code generation capabilities of Large Language Models (LLMs) have enhanced planning for Embodied AI tasks. However, efficiently informing the agent about the environment remains a challenge. Inspired by modular visual reasoning, we propose a novel approach that uses code generation to ground the planner in the environmental context and reason about past experiences. Our framework allows the LLM to aggregate information from relevant observations via API calls to image understanding models, including flexible VLMs. We evaluate our approach using Embodied Question Answering (EQA) and develop a synthetic data collection procedure using simulator ground truth states. Our framework shows notable improvements over baseline methods.

1. Introduction

With Foundation Models conquering traditional downstream tasks such as image generation (Saharia et al., 2022), object recognition (Minderer et al., 2022; Kuo et al., 2022), segmentation (Kirillov et al., 2023), video understanding (Arnab et al., 2021), object tracking (Yang et al., 2023a), and image captioning (Yu et al., 2022), embodied reasoning is becoming a sought-after target (Huang et al., 2022b). Embodied reasoning presents unique challenges over image- and video-based reasoning, including multimodality, spatiotemporal structure, and action-conditioned data collection difficulties. Notably, it requires multistep (Dasgupta et al., 2022; Yu et al., 2019), modular, and interpretable reasoning to handle partial observability, general applicability, and effective communication with humans and AI agents.

To achieve this, literature (Surís et al., 2023; Roziere et al., 2023; Li et al., 2022) has focused on AI systems that combine pre-trained modules like LLMs (Achiam et al., 2023; Team et al., 2024; Brown et al., 2020; Touvron et al., 2023) or VLMs (Wang et al., 2023; Liu et al., 2023; Bai et al., 2023). While template-based program generation offers flexibility, it lacks generalization to arbitrary queries and open worlds (Yu et al., 2019). LLMs, with their generalization and in-context learning capabilities, are promising but struggle with modularity due to the combinatorial growth of sample space. We propose using code generation as an intermediate module to attain multistep, modular, and interpretable embodied reasoning, inspired by recent progress in visual question answering (Surís et al., 2023; Roziere et al., 2023; Li et al., 2022). Our system integrates perception, memory, and code generation LLM modules with online Python code execution. Unlike ProgPrompt (Singh et al., 2023), which focuses on planning, we focus on its prerequisite – embodied reasoning. Our contributions include presenting STREAM, a method for grounded reasoning from embodied experience without training, proposing an efficient data collection procedure for synthetic question-answer pairs, and evaluating our method on a new benchmark called QuEST, demonstrating its superiority against strong baselines.

2. Related work

Episodic Memory Question Answering. Visual Question Answering (VQA) (Antol et al., 2015; Goyal et al., 2017) proposes a flexible framework for evaluating the understanding of natural images but is often limited to single-image contexts. To address this limitation, VQA systems have been enhanced to handle more complex inputs like videos (Zhong et al., 2022; Choudhury et al., 2023; Liang et al., 2024), 3D scenes (Ma et al., 2022; Fu et al., 2024; Chen et al., 2022), and full environments (Das et al., 2018). OpenEQA (Majumdar et al., 2024) introduces a practical setting for Episodic Memory Embodied QA (EM-EQA), where the agent can passively process previously observed visual history to answer questions about the environment. Through the costly human labeling process, the authors managed to collect 1636 questions. As the baselines, they use multimodal foundation models like GPT-4V (Yang et al., 2023b) to process entire visual histories or caption each frame and aggregate these captions with an LLM. However, processing long videos or reconstructing 3D scenes can be prohibitively expensive or inefficient. Our work focuses on a nuanced

[1]University of Amsterdam. Correspondence to: Daniil Cherniavskii <d.cherniavskii@uva.nl>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

approach where the agent actively reasons about processing the observation history to answer questions, grounding answers in the visual history effectively.

Code for Visual Reasoning and Embodied AI. Answering complex questions requires combining basic skills, which can benefit from a reasoning module that integrates these skills. Modular VQA techniques like Neural Modular Networks (NMNs) (Andreas et al., 2016), NS-VQA (Yi et al., 2018), and Probabilistic Neural-symbolic Models (Vedantam et al., 2019) have integrated learned modules but face challenges in open-ended settings. Leveraging LLMs like GPT-4 (Achiam et al., 2023) for code generation, specialized in generating programs, has shown promise (Subramanian et al., 2023; Gupta & Kembhavi, 2022b; Surís et al., 2023; Choudhury et al., 2023; Ge et al., 2023; Liang et al., 2024). ViperGPT (Surís et al., 2023) and ProViQ (Choudhury et al., 2023) deliver zero-shot results on VideoQA datasets like NeXT-QA (Xiao et al., 2021) but are limited in temporal and situated reasoning. We propose extending the modular vision framework to efficiently process the temporal history of visual observations and agents' actions in environments like AI2-THOR (Kolve et al., 2017). This includes tools for flexible scene understanding, such as depth models for exact locations and VLMs (Wang et al., 2023) for general visual questions.

In the active robotic setting, code generation with LLMs has been useful (Singh et al., 2023; Liang et al., 2022; Huang et al., 2023). ProgPrompt (Singh et al., 2023) generates robot task plans as code, while CodeAsPolicies (Liang et al., 2022) translates natural language commands into robot policy code. VoxPoser (Huang et al., 2023) uses LLMs to generate code interacting with VLMs to produce sequences of 3D affordances. These approaches are promising for tabletop manipulation but struggle in partially observable environments where the episode history is crucial for plan generation. Our work focuses on processing visual histories and envisions combining this with code generation for future planning tasks.

3. QuEST benchmark

In creating a benchmark to evaluate complex, multistep, and grounded spatiotemporal embodied reasoning, we have a few requirements. First, the answers must be grounded, ensuring justification for further planning. Second, we aim to maintain low annotation costs while keeping the data realistic. Third, the partially observable nature of embodied reasoning must be reflected in queries, requiring tasks that involve multiple locations, vantage points, and times. This forms the basis of our benchmark, QuEST (**Que**stioning in **E**mbodied **S**imulated **T**asks).

In the most general and realistic setting for embodied reasoning, we can only assume that the agent knows the world via its own experiences, that is, without assuming complete 3D maps (He et al., 2024), prescribed lists of objects present (Singh et al., 2023), or constrained lists of possible reasoning tasks (Huang et al., 2022a). We formulate our embodied agent data setting as a collection of n *experience trajectory* tuples $\left\{\left(s_t, a_t\right)_{t=1,\ldots,T}\right\}_{i=1,\ldots,n}$ in arbitrary recorded times t, comprising agent states and observations $s_t = \left(l_t, p_t, x_t\right)$ of locations l, camera pose parameters p, RGB-D image observations x_t, as well as actions a_t, taken in the next moment. Inspired by (Das et al., 2018) we evaluate embodied reasoning with grounded embodied question answering, that is, requiring that given a novel query prompt q, the agent not only answers correctly in a required structured format but also provides an accurate grounding for the objects in answer, i.e. its coordinates.

To generate such question-answer pairs, we use ALFRED (Shridhar et al., 2020), a dataset of household tasks in the AI2-THOR simulator (Kolve et al., 2017), where agents complete tasks through a series of actions. We use the FiLM agent (Min et al., 2021) with ground truth depth estimation and pre-trained semantic segmentation models for RGB inputs, and along with embodied agent data, we record the ground truth states of the simulator, i.e. objects locations, states, and bounding boxes for each observation. The latest is optional and could be switched to any object recognition model. To maintain a realistic data generation process where the agent may perform random exploration, we sample 1,529 experience trajectories using the unseen test part of ALFRED, with the agent navigating and interacting with the environment. Utilizing the knowledge of the ground truth environment states in these trajectories, we generate questions based on 10 hand-coded templates. Each of the template questions tests the spatiotemporal understanding of the model, from simple ones (e.g. *"Have you seen the TV?"*, *"What objects do you see now?"*) to a more complex (e.g. *"Did you open the drawer before or after you picked up the watch?"*, *"What is the state of the cabinet? Open or closed?"*). The data collection process is demonstrated in Figure 4.

For evaluation, as the answers to the questions are structured, we use the Jaccard score for list-based answers (e.g. "Q: What objects are on the table? A: ['credit card(credit_card_id)']") and accuracy for specific responses (e.g. yes/no questions). We also assess the executability of generated solutions, ensuring the generated code runs correctly and the answer satisfies the requested format. Our dataset has around 36,000 questions, with a roughly balanced distribution across categories. For a full description of data statistics and examples of generated questions, please see Appendix A.1.

4. STREAM

We propose STREAM (**S**patio-**T**emporal **R**easoning, **E**mbodied **A**ction, and **M**emory), an AI system designed for spatiotemporal embodied reasoning. Our method addresses the need for AI systems capable of complex problem-solving by integrating multiple pre-trained modules. STREAM comprises three core modules: Perception, Memory, and Reasoning. Inspired by the code generation capabilities of ViperGPT (Surís et al., 2023) and VISPROG (Gupta & Kembhavi, 2022a), we make the Reasoning module the central component, uniting the system by coordinating the interactions between the Perception and Memory modules through the Reasoning API. This integration allows the system to process and interpret data effectively, facilitating advanced spatiotemporal embodied reasoning in dynamic environments. The complete pipeline can be found in Figure 5.

Reasoning API. The Reasoning API is the backbone of STREAM, constructed around three primary classes: `SceneObjectAPI`, `ActionAPI`, and `RGBDImageAPI`. The `SceneObjectAPI` class manages object instances, providing essential attributes such as object type, location, and bounding boxes, which are tied to specific images. The `RGBDImageAPI` class captures the agent's experiences over time, It consists of an RGB-D frame, camera and agent positions, and a timestep of observation. The `ActionAPI` class details the actions performed by the agent, including action types and targeted objects. Additionally, the API includes useful functions like `get_all_objects_images()` and `get_relative_position()` to facilitate modular and reusable code generation. These classes and functions enable the system to capture temporal, spatial, and visual properties of the scene in a structured way, ensuring robust and interpretable reasoning.

Reasoning module. The reasoning module of STREAM employs the Gemma 7B code generation LLM (Team et al., 2024), which is adept at producing high-quality code. This module leverages the in-context learning abilities of the LLMs (Brown et al., 2020), allowing it to generate code based on provided examples and prompts. In our experiments, we conducted ablation studies to optimize the model's performance, exploring various quantization strategies, model sizes, and design choices to balance cost and efficiency. The reasoning module's ability to generate modular and interpretable code is crucial for enabling the system to perform complex tasks that require spatiotemporal reasoning and coordination between different AI modules.

Perception module. The perception module in STREAM utilizes a vision-language model (VLM) to handle queries about the visual properties of the scene. Among three VLMs: Qwen-VL (Bai et al., 2023), LLaVa-1.6 (Liu et al., 2024), CogVLM (Wang et al., 2023), we selected CogVLM based on its grounding abilities and its superior performance in object recognition and localization tasks on our dataset subsample of 5000 images, achieving a 0.75 mIoU score. The perception module allows the system to verify and extract visual and spatial properties of objects through functions like `verify_property()` and `get_object_state()`. By efficiently querying visual information, the perception module supports the reasoning module in making informed decisions about the environment, enhancing the system's overall capability to process and interpret complex scenes.

Memory module. The memory module of STREAM consists of instances from the API classes, enabling the system to store and access past experiences. Preprocessing steps include depth estimation, object recognition, and linking to calculate 3D positions and assign unique IDs to objects across different images. In the simulator, we use ground truth states to recognize objects and link them, but for real-world applications, methods using CLIP (Radford et al., 2021) and spatial comparison can be employed. This module's robust preprocessing and data management ensures that the system can track objects over time and space, facilitating complex queries and enabling detailed spatiotemporal reasoning.

5. Experiments

5.1. Baselines

We implement four baselines with similar parameter complexity to evaluate our model's performance. The first baseline is VideoLLaVA (Lin et al., 2023), a model for video understanding that processes the entire sequence of images as input and directly generates the answer. This comparison assesses whether our structured reasoning process, decoupled from raw data, captures and utilizes temporal, spatial, and visual information more effectively than a model where reasoning and data processing are entangled. While VideoLLAVA can generate structured outputs, it cannot ground its answers and explain the process behind its solutions. The second baseline, the Textual Embodied Large Language Model (TE-LLM), uses the same LLM (Gemma-7B) but provides the entire embodied experience in plain text format. Due to the LLM's token limit, we subsample observations, ensuring frames with non-trivial actions are included and each observed object is represented at least once. The admissible length of the input in this case is 30 observations on average. The model is prompted with the list of observed objects and their locations, the robot's position, and the next action. This baseline evaluates whether dynamically

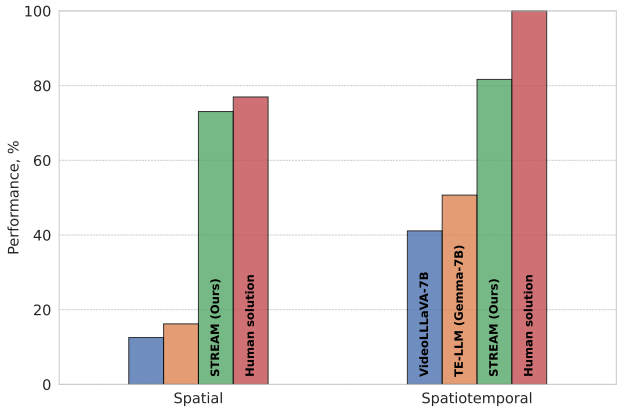

Figure 1. Average performance for spatial and spatiotemporal groups of questions across different models.

generating and executing functions offer more effective use of the LLM's reasoning capabilities compared to a direct text-based approach.

The third baseline involves applying CogVLM throughout the experience trajectories to obtain dense captions for all relevant frames. These captions, along with recognized objects and their respective locations and timesteps, are then added to the Gemma prompt for final reasoning. This approach, however, proved computationally heavy, and full comparisons were not possible, so we compared this baseline to a smaller subset of the benchmark in our ablation studies. The full prompts for each model, as well as the Table 4 of baselines functional comparison, are detailed in the Appendix A.2.

Additionally, we report the results of a hand-coded human solution as a "soft upper bound" baseline, where a human expert writes near-ideal code using our developed API.

5.2. Results

We present average performance results in Figure 1. For a complete executability and performance evaluation of every question type, please refer to Table 5. STREAM generates executable Python programs with more than 85% executability for 6 from 10 question types. VideoLLAVA has near perfect "executability", however, TE-LLM returns the correct format for only 44.5% of QuEST questions because it can include information up to 30 frames before maxing out the token limit. Note that VideoLLaVA and TE-LLM inherently cannot process certain question types, see Table 4.

In terms of accuracy, STREAM has a clear edge over the baselines, surpassing VideoLLaVA by 15-90% and TE-LLM by 12-95%. VideoLLaVA struggles because it is forced to always generate output directly and without multi-step reasoning. This is hard for more complex questions like *"What objects did you see on the kitchen table?"*, where VideoLLaVA would have to process all frames directly, reaching only 10.8% accuracy. TE-LLM copes better with multi-step reasoning as it reduces complex images to language tokens that the subsequent LLM can reason about. Overall, however, having code generation as an intermediate reasoning module is important. Another natural conclusion from this observation is that even though LLMs may have tremendous generalization capabilities, the way we use the same LLM — *e.g.* for intermediate code rather than natural language generation — can have a big impact on the final performance.

Table 1. Ablating different LLMs for reasoning

Model	Avg. Acc.	Avg. Exec. %
Gemma-2B	4.6	2.26
TE-LLM + CogVLM	34.2	32.84
Gemma-7B 4bit	67.1	76.48
Gemma-7B	71.1	77.32

Ablations. In Table 1 we compare different LLMs for our reasoning module, specifically Gemma2B, Gemma7B with weights quantized to 4 bits, and CogVLM, on a subset of 100 experience trajectories. Further, in Table 6 in the Appendix, we provide the detailed per-category accuracies. We observe that the Gemma 2B is not sufficient for reliable reasoning, while the quantized Gemma 7B approaches closely the unquantized version for almost all question categories while being four times smaller in memory.

6. Conclusion

This paper proposes a novel framework that leverages the advanced capabilities of LLMs in code generation to improve Embodied QA based on agents' experiences. Our modular STREAM method dynamically generates simple Python programs to process historical information using image understanding models (e.g., VLMs) in a structured way. To evaluate such STREAM, we created a large and synthetic EQA dataset named QuEST. It tests the ability to reason about spatial and spatiotemporal questions and ground the answers in the history of the agent's observations. We show that the intermediate reasoning module is necessary not only for interpretable multistep spatiotemporal reasoning but also for allowing for effective and efficient Embodied QA.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.

Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., and Zhou, J. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. 2023.

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

Chen, W., Hu, S., Talak, R., and Carlone, L. Leveraging large language models for robot 3d scene understanding. *arXiv preprint arXiv:2209.05629*, 2022.

Choudhury, R., Niinuma, K., Kitani, K. M., and Jeni, L. A. Zero-shot video question answering with procedural programs. *arXiv preprint arXiv:2312.00937*, 2023.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–10, 2018.

Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., and Fergus, R. Collaborating with language models for embodied reasoning. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Fu, R., Liu, J., Chen, X., Nie, Y., and Xiong, W. Scene-llm: Extending language model for 3d visual understanding and reasoning. *arXiv preprint arXiv:2403.11401*, 2024.

Ge, J., Subramanian, S., Shi, B., Herzig, R., and Darrell, T. Recursive visual programming. *arXiv preprint arXiv:2312.02249*, 2023.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Gupta, T. and Kembhavi, A. Visual programming: Compositional visual reasoning without training. *ArXiv*, abs/2211.11559, 2022a.

Gupta, T. and Kembhavi, A. Visual programming: Compositional visual reasoning without training. 2023 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, 2022b.

He, Q., Lin, K., Chen, S., Hu, A., and Jin, Q. Think-program-rectify: 3d situated reasoning with large language models. *arXiv preprint arXiv:2404.14705*, 2024.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022a.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *arXiv preprint arXiv:2207.05608*, 2022b.

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

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

Kuo, W., Cui, Y., Gu, X., Piergiovanni, A., and Angelova, A. F-vlm: Open-vocabulary object detection upon frozen vision and language models. *arXiv preprint arXiv:2209.15639*, 2022.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

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

Liang, L., Sun, G., Qiu, J., and Zhang, L. Neural-symbolic videoqa: Learning compositional spatio-temporal reasoning for real-world video question answering. *arXiv preprint arXiv:2404.04007*, 2024.

Lin, B., Zhu, B., Ye, Y., Ning, M., Jin, P., and Yuan, L. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning, 2023.

Liu, H., Li, C., Li, Y., Li, B., Zhang, Y., Shen, S., and Lee, Y. J. Llava-1.6: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-1-6/.

Ma, X., Yong, S., Zheng, Z., Li, Q., Liang, Y., Zhu, S.-C., and Huang, S. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*, 2022.

Majumdar, A., Ajay, A., Zhang, X., Putta, P., Yenamandra, S., Henaff, M., Silwal, S., Mcvay, P., Maksymets, O., Arnaud, S., Yadav, K., Li, Q., Newman, B., Sharma, M., Berges, V., Zhang, S., Agrawal, P., Bisk, Y., Batra, D., Kalakrishnan, M., Meier, F., Paxton, C., Sax, S., and Rajeswaran, A. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Min, S. Y., Chaplot, D. S., Ravikumar, P. K., Bisk, Y., and Salakhutdinov, R. Film: Following instructions in language with modular methods. In *International Conference on Learning Representations*, 2021.

Minderer, M., Gritsenko, A., Stone, A., Neumann, M., Weissenborn, D., Dosovitskiy, A., Mahendran, A., Arnab, A., Dehghani, M., Shen, Z., et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pp. 728–755. Springer, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530. IEEE, 2023.

Subramanian, S., Narasimhan, M., Khangaonkar, K., Yang, K., Nagrani, A., Schmid, C., Zeng, A., Darrell, T., and Klein, D. Modular visual question answering via code generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 747–761, 2023.

Surís, D., Menon, S., and Vondrick, C. Vipergpt: Visual inference via python execution for reasoning. *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2023.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vedantam, R., Desai, K., Lee, S., Rohrbach, M., Batra, D., and Parikh, D. Probabilistic neural symbolic models for interpretable visual question answering. In *International Conference on Machine Learning*, pp. 6428–6437. PMLR, 2019.

Wang, W., Lv, Q., Yu, W., Hong, W., Qi, J., Wang, Y., Ji, J., Yang, Z., Zhao, L., Song, X., et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

Xiao, J., Shang, X., Yao, A., and Chua, T.-S. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on*

computer vision and pattern recognition, pp. 9777–9786, 2021.

Yang, J., Gao, M., Li, Z., Gao, S., Wang, F., and Zheng, F. Track anything: Segment anything meets videos, 2023a.

Yang, Z., Li, L., Lin, K., Wang, J., Lin, C.-C., Liu, Z., and Wang, L. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9 (1):1, 2023b.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

Yu, L., Chen, X., Gkioxari, G., Bansal, M., Berg, T. L., and Batra, D. Multi-target embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6309–6318, 2019.

Zhong, Y., Xiao, J., Ji, W., Li, Y., Deng, W., and Chua, T.-S. Video question answering: Datasets, algorithms and challenges. *arXiv preprint arXiv:2203.01225*, 2022.

A. Appendix

A.1. QuEST data collection and statistics

We generate a total of 1,529 trajectories with a limit on the number of frames of 1,000. The median length of the trajectory is 261. For each trajectory, we randomly sample 3 questions from each of the 10 pre-defined question categories at random timesteps, totaling to around 36k questions. The distribution of the number of different question categories is roughly balanced, see Figure 2 in the appendix for the precise distribution. For trajectory length distribution, please refer to Figure 3.

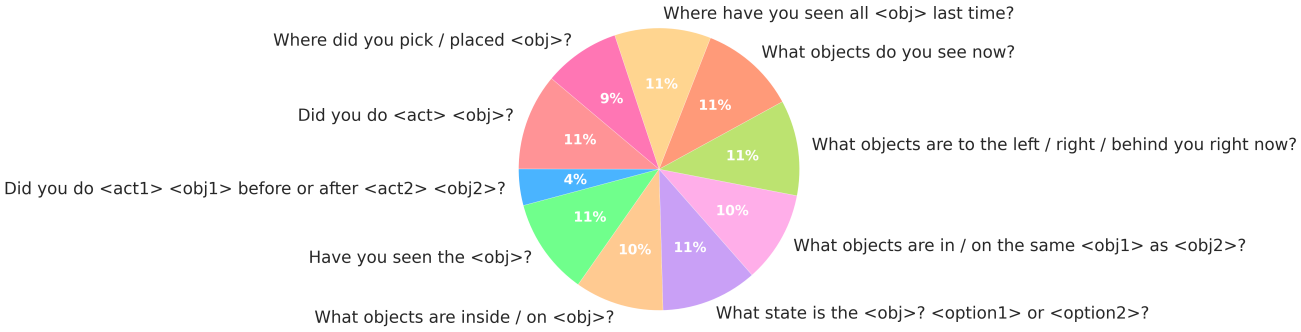

Figure 2. The distribution of question categories in the dataset.

Table 2. Examples of questions.

Question	Example
did_you_do_a_before_b	Did you pick up the statute before or after you put down the watch?
what_is_inside_on	What is on the coffee table?
did_you_do	Did you open the drawer?
have_seen	Have you seen the book?
what_objects_in_same	What objects are on the same shelf as the vase?
what_objects_see_now	What objects do you see now?
where_put_pick_obj	Where did you pick up the plate?
what_objects_lbr	What objects are to the left of you know?
where_have_seen_last	Where have you seen the fork last?
what_object_state	What is the state of the microwave? Is it on or off?

A.2. Model, baselines and results

Listing 1. The template code for the proposed Reasoning API.

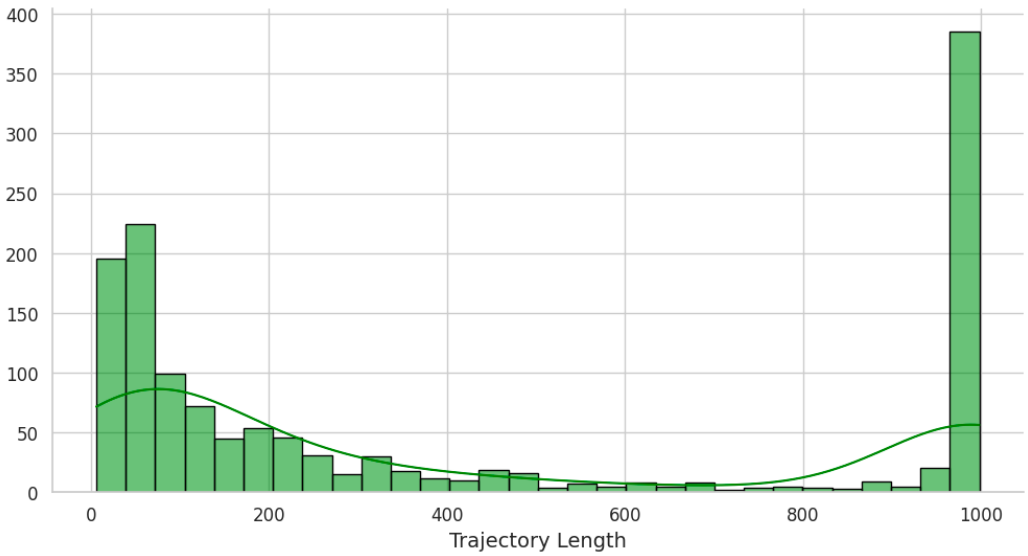

Figure 3. The distribution of trajectory lengths in the dataset.

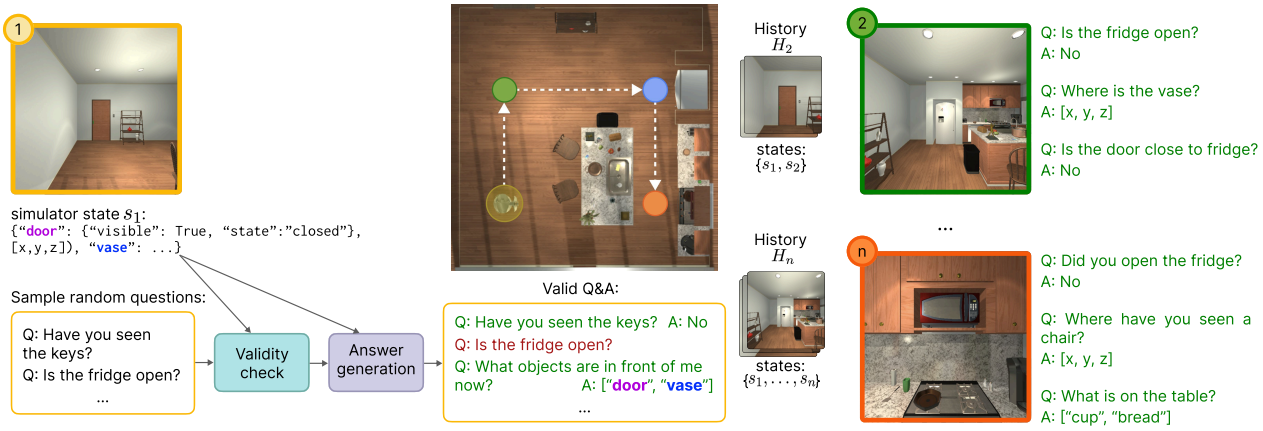

Figure 4. The scheme describing the process of data collection. We start by sampling random questions from our question types. After this, we pass history simulator states together with questions to a validity check that filters questions that can not be answered from current history. For example, if the history contains no fridge objects, then the question about the fridge state, such as *"Is the fridge open?"* is not valid. Finally, we generate answers to valid questions using the history of simulator states.

Table 3. Dataset statistics. RCM stands for Random Classifier Metric.

Question	Metric	# samples	Median timestep	RCM
did_you_do_a_before_b	Accuracy	1512	113.0	50.0
what_is_inside_on	Jaccard	3708	112.0	2.0
did_you_do	Accuracy	4020	50.0	50.0
have_seen	Accuracy	4020	84.0	50.0
what_objects_in_same	Jaccard	3798	90.0	1.6
what_objects_see_now	Jaccard	4020	97.0	4.2
where_put_pick_obj	Accuracy	3183	74.0	50.0
what_objects_lbr	Jaccard	3972	93.0	3.6
where_have_seen_last	Jaccard	4020	71.0	1.6
what_object_state	Accuracy	4011	71.0	50.0

Table 4. Comparison of different approaches based on their ability to provide grounded answers, interpretable solutions, and structured output.

Approach	**Grounded answers**	**Interpretable solutions**	**Structured output**
VideoLLaVA-7B	×	×	✓
TE-LLM (Gemma-7B)	✓	×	✓
TE-LLM (Gemma-7B) + CogVLM	✓	×	✓
STREAM (Ours)	✓	✓	✓

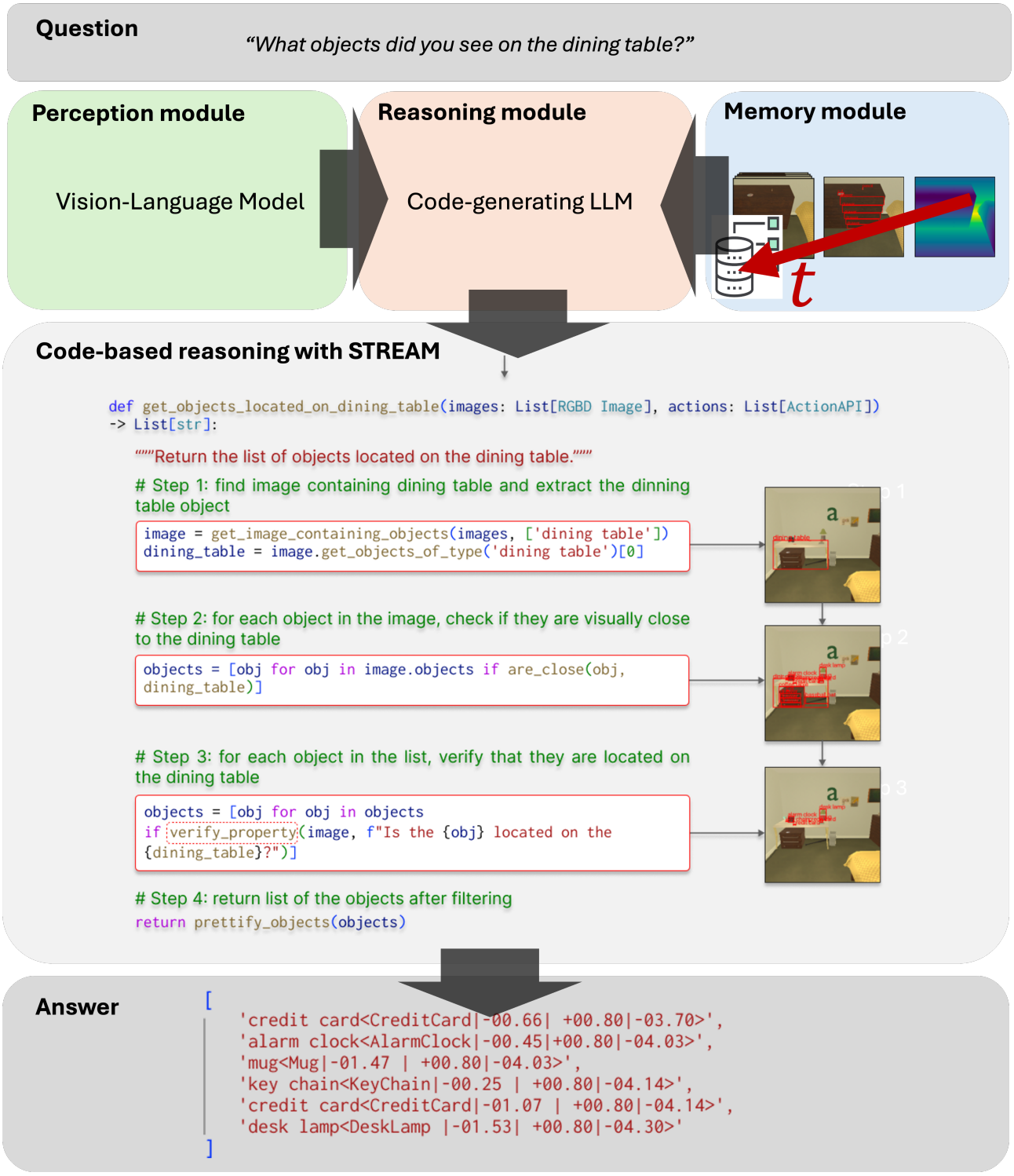

Figure 5. The STREAM pipeline. The code generating LLM is prompted with a description of the API, a few examples and a question, and produces a general data-independent code solution, which employs other models like a Vision Language Model (VLM). The input to the generated code function is an agent's history, which consists of a sequence of actions and image observations, preprocessed with depth estimation and object recognition models. The code solution is then executed to produce the answer. Here, the execution traces are presented.

You are a household robot. You are provided with a sequence of observations, actions, and robot positions below. Please, answer the question after this sequence.

1. Caption: {caption}. Observed objects: [{obj1.obj_type}¡{obj1.id}¿, {obj2.obj_type}¡{obj2.id}¿, ...]; Robot position: {'x':0.0, 'y':0.0, 'z':0.0}; Action: {action}.

...

...

...

QUESTION: {question}. Please, return the output in the following format: {output_format}. Example: {example_of_output_format}.

Figure 6. The template of the prompt for TE-LLM (+ CogVLM) baseline.

You are seeing a first person video of the robot performing household tasks. Please, answer the following question as if you are the robot.
QUESTION: {question}. Please, return the output in the following format: {output_format}. Example: {example_of_output_format}.

Figure 7. The template of the prompt for VideoLLaVA baseline.

Table 5. Comparing with VideoLLaVA-7B and TE-LLM baselines on QuEST. Blue indicates questions that require spatiotemporal reasoning, Green indicates questions that require spatial only reasoning. See Table 3 for the corresponding evaluation metrics for each question type.

	did_you_do		did_you_do a_before_b		have_seen		where_have seen_last		what_objects see_now		Average (spatiotemporal)	
Model	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.
VideoLLaVA-7B	100.0	37.7	100.0	51.4	100.0	64.5	-	-	100.0	10.8	100.0	41.1
TE-LLM (Gemma-7B)	31.7	70.8	51.0	49.6	57.8	**94.5**	44.1	35.1	7.9	3.4	38.5	50.7
STREAM (Ours)	89.8	**94.2**	17.6	**72.4**	91.7	**94.6**	91.5	**47.4**	100.0	**99.7**	78.1	**81.7**
Human solution	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

	what_objects in_same		what_objects lbr		what_is inside_on		what_object state		where_put pick_obj		Average (spatial)	
Model	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.	Exec.	Perf.
VideoLLaVA-7B	100.0	5.4	100.0	9.1	100.0	7.4	100.0	28.3	-	-	100.0	12.6
TE-LLM (Gemma-7B)	80.1	4.1	-	-	37.5	12.1	-	-	45.8	32.4	54.5	16.2
STREAM (Ours)	77.3	**40.7**	100.0	**99.9**	99.4	**64.7**	64.5	**64.1**	64.7	**95.9**	81.2	**73.1**
Human solution	100.0	51.3	100.0	100.0	100.0	66.1	100.0	67.4	100.0	100.0	100.0	77.0

Table 6. Ablation study results with different LLM models for the reasoning module.

Question	Model	Exec.	Metric
did_you_do	STREAM (Gemma-2B)	1.4	42.86
	TE-LLM (Gemma-7B) + CogVLM	32.8	79.03
	STREAM (Gemma-7B 4bit)	68.6	89.08
	STREAM (Gemma-7B)	89.8	**94.17**
did_you_do_a_before_b	STREAM (Gemma-2B)	1.1	0.0
	TE-LLM (Gemma-7B) + CogVLM	55.1	46.51
	STREAM (Gemma-7B 4bit)	24.7	39.13
	STREAM (Gemma-7B)	17.6	**72.41**
have_seen	STREAM (Gemma-2B)	0.0	0.0
	TE-LLM (Gemma-7B) + CogVLM	59.8	90.27
	STREAM (Gemma-7B 4bit)	85.3	98.71
	STREAM (Gemma-7B)	91.7	**94.59**
what_is_inside_on	STREAM (Gemma-2B)	0.0	0.0
	TE-LLM (Gemma-7B) + CogVLM	43.9	11.4
	STREAM (Gemma-7B 4bit)	87.0	61.61
	STREAM (Gemma-7B)	99.4	**64.74**
what_object_state	STREAM (Gemma-2B)	0.0	0.0
	TE-LLM (Gemma-7B) + CogVLM	6.9	0.0
	STREAM (Gemma-7B 4bit)	59.2	60.07
	STREAM (Gemma-7B)	64.5	**64.1**
what_objects_in_same	STREAM (Gemma-2B)	0.0	0.0
	TE-LLM (Gemma-7B) + CogVLM	87.4	5.73
	STREAM (Gemma-7B 4bit)	84.0	38.27
	STREAM (Gemma-7B)	77.3	**40.67**
what_objects_lbr	STREAM (Gemma-2B)	19.5	7.53
	TE-LLM (Gemma-7B) + CogVLM	0.0	nan
	STREAM (Gemma-7B 4bit)	100.0	99.81
	STREAM (Gemma-7B)	100.0	**99.91**
what_objects_see_now	STREAM (Gemma-2B)	0.0	0.0
	TE-LLM (Gemma-7B) + CogVLM	3.2	0.0
	STREAM (Gemma-7B 4bit)	100.0	99.75
	STREAM (Gemma-7B)	100.0	**99.72**
where_have_seen_last	STREAM (Gemma-2B)	2.1	0.0
	TE-LLM (Gemma-7B) + CogVLM	40.2	25.83
	STREAM (Gemma-7B 4bit)	91.4	40.83
	STREAM (Gemma-7B)	91.5	**47.41**
where_put_pick_obj	STREAM (Gemma-2B)	0.8	0.0
	TE-LLM (Gemma-7B) + CogVLM	32.1	54.72
	STREAM (Gemma-7B 4bit)	56.2	96.08
	STREAM (Gemma-7B)	64.7	**95.92**