RoboGolf: Mastering Real-World Minigolf with a Reflective Multi-Modality Vision-Language Model

Hantao Zhou [*1] **Tianying Ji** [*2] **Lukas Sommerhalder** [1] **Michael Görner** [1] **Norman Hendrich** [1] **Fuchun Sun** [2] **Jianwei Zhang** [1] **Huazhe Xu** [2 3]

Abstract

Minigolf is an exemplary real-world game for examining embodied intelligence, requiring challenging spatial and kinodynamic understanding to putt the ball. Additionally, reflective reasoning is required if the feasibility of a challenge is not ensured. We introduce **RoboGolf**, a VLM-based framework that combines dual-camera perception with closed-loop action refinement, augmented by a reflective equilibrium loop. The core of both loops is powered by finetuned VLMs. We analyze the capabilities of the framework in an offline inference setting, relying on an extensive set of recorded trajectories. Exemplary demonstrations of the analyzed problem domain are available at https://robogolfvlm.github.io/.

1. Introduction

Recent advancements in large foundation models (Yang et al., 2023b; Ahn et al., 2024; Maatouk et al., 2023) offer promising avenues for augmenting robot intelligence, particularly through the integration of large language models (LLMs) and vision-language models (VLMs). Pioneering projects in controlled robot manipulation (Ahn et al., 2022; Liang et al., 2023; Hu et al., 2023; Huang et al., 2023) demonstrate impressive behavior. A present limitation of these systems is their assumption that the tasks they are asked to fulfill are always feasible without adjustments.

To effectively handle complex, open-ended scenarios, what more do robots need beyond spatial and kinodynamic understanding? We consider higher-level reflective reasoning mechanisms inspired by the philosophical "reflective equilibrium" method (Daniels, 1979; Cath, 2016). Specifically, intelligent robots should not only understand tasks and scenarios and then design possible solutions, but also proactively suggest adjustments to make tasks feasible if they exceed the robot's current capabilities. This ability to proactively propose practical modifications to tasks can avoid repeated trial and error under infeasible tasks.

Minigolf stands out as an exemplary setting for demonstrating advanced intelligence. Its highly variable course settings, with various endpoints and obstacles, create numerous combinations and possibilities. The robot is expected to identify feasible routes or modify infeasible ones to become solvable. Additionally, the kinodynamic interactions with endpoints and obstacles are complex; an extreme case involves one ball pushing a second ball into the target.

We propose **RoboGolf**, a framework that integrates dual-camera perception, an inner inference loop refining action parameters between attempts, and an outer loop, inspired by the method of reflective equilibrium, to identify impossible golf courses and propose feasible modifications. The core of both loops is realized through fine-tuned VLMs.

2. Related Works

Kinodynamic understanding. Understanding kinodynamics has been a longstanding challenge in robotics. Various approaches have been explored, including trajectory prediction (Westny et al., 2024; Fragkiadaki et al., 2015) and physical reasoning via training specific deep neural networks (Girdhar et al., 2020; Bhattacharyya et al., 2016; Battaglia et al., 2018; Groth et al., 2018). However, these methods often require extensive ground-truth data, limiting their real-world application without a simulator. In contrast, exploring the kinodynamic capacity of VLMs through finetuning on small data regimes presents an appealing alternative. Our work demonstrates this by fine-tuning VLMs on a small scale of auto-labeled data.

LLMs and VLMs for robotics. LLMs (Zeng et al., 2023; Ahn et al., 2022) and VLMs (Huang et al., 2023; Hu et al., 2023; Durante et al., 2024; Brohan et al., 2023; Liu et al., 2024a) have shown great potential in advancing robotic scenario understanding from textual or visual input and enhancing decision-making capabilities. However, most previous

[*]Equal contribution [1]Universität Hamburg [2]Tsinghua University [3]Shanghai Qi Zhi Institute. Correspondence to: Tianying Ji <jity20@mails.tsinghua.edu.cn>.

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

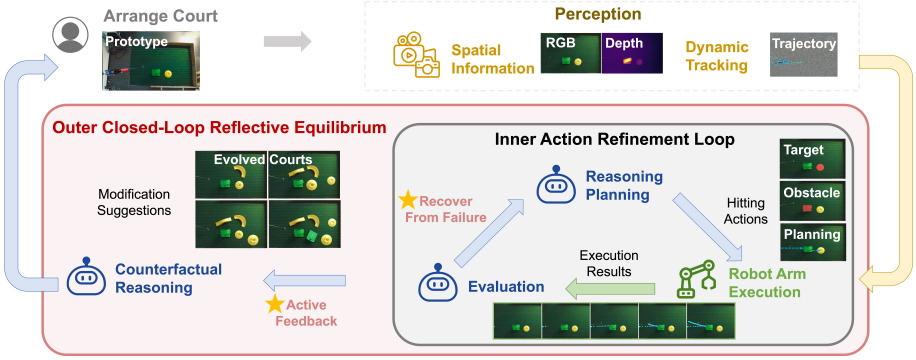


Figure 1. **Conceptual Overview.** Our system integrates dual-camera scene perception, an inner action refinement loop predicting hit parameters, and an outer reasoning loop assessing course feasibility. An RGB-D camera captures the arranged spatial scene and ball trajectories are tracked using an event camera. The inner loop derives hitting parameters, adjusting them through evaluation of failed attempts. The outer reflective loop uses counterfactual reasoning to suggest course modifications in unsolvable scenarios.

work allow the robot to attempt tasks only once without addressing failures. Recently, interest has emerged in devising VLMs with closed-loop control for robotics tasks (Zhi et al., 2024), enabling robots to recover from failures. Despite this progress, these robots still passively complete tasks without questioning or modifying impractical scenarios caused by human errors.

3. Method and Paradigm

Our system, conceptualized in Figure 1, encompasses three major components: scene perception, action execution with closed-loop refinement, and an outer reflective reasoning loop to resolve impractical courses.

To facilitate brief episodic trials, we limit each test to commence from a consistent initial position, involving a single motion action can be parameterised. This parameter choice underpins our method for parameter inference via the VLM.

3.1. Perception

Accurate perception of the court and golf ball movement is essential for correct parameter inference. Spatial information—necessary to estimate feasible trajectories to goals—is captured using an Azure Kinect RGB-D camera. However, the RGB-D camera suffers from motion-blur and is inadequate for capturing the rapid movements of a golf ball. To improve overall perception quality, we therefore implement a dual-camera setup with an additional Davis345 event camera. While the event camera is optimized for fast motion, it often produces high noise levels (Rebecq et al., 2019; Gallego et al., 2020). We then employ RGB-D videos as prompts for SAM (Kirillov et al., 2023) to reduce noise in event-camera videos and to distinguish the trajectories of the golf club and golf balls. Figure 2 illustrates the workflow of the perception module.

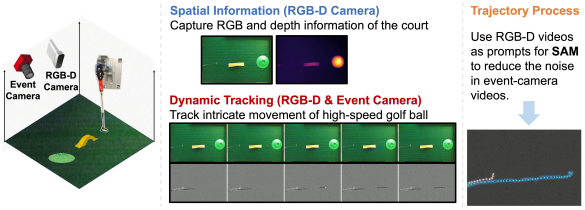


Figure 2. **Perception module.** RoboGolf employs a dual-camera setup. Spatial Information: Use an RGB-D camera to capture details of the minigolf course. Dynamic Tracking: Use the RGB-D camera and event camera to record the movement of the high-speed golf ball. Trajectory Process: To obtain an accurate trajectory, RoboGolf leverages both cameras, using RGB-D videos as prompts to refine data from the event camera.

3.2. Inner Loop Action Refinement

The inner action refinement loop involves the derivation of action parameters and their episode execution on the robot arm.

Reasoning and planning involves mapping a route to the goal $\mathbf{g} \in \mathbb{R}^2$ and deriving parameters to execute it. Specifically, the system reads the user's input $\mathcal{S}$ and uses LISA (Lai et al., 2023; Yang et al., 2023a) to identify the endpoint $\mathbf{g}$ from the court observation $\mathcal{OV}$: $\mathbf{g} = \mathcal{F}_{VLM}(\mathcal{S}; \mathcal{OV})$. Leveraging LISA, we thus identify a suitable endpoint in the visual field $\mathcal{V} \subseteq \mathbb{R}^2$. All other recognized entities become obstacles $\mathcal{O} \subseteq \mathbb{R}^2$. The model maps these to a layout and finds a route $\mathbf{r}$ from start $\mathbf{s}$ to endpoint $\mathbf{g}$: $\mathbf{r} = \mathcal{F}_{VL}(\mathbf{s}, \mathbf{g}, \mathcal{O}; \mathcal{OV})$. Finally, it estimates hitting speed v and angle θ considering obstacles and distance to $\mathbf{g}$: $v = \mathcal{F}_v(\mathbf{r}, \mathcal{O}, \mathbf{g}; \mathcal{OV})$ and $\theta = \mathcal{F}_\theta(\mathbf{r}, \mathcal{O}, \mathbf{g}; \mathcal{OV})$.

Failure analysis through evaluation involves assessing execution results and refining parameters accordingly. Specifically, the system evaluates outcomes using $\mathbf{e} = \mathcal{F}_e(\mathbf{r}, v, \theta, \mathbf{g}; \mathcal{OV})$, where $\mathbf{e}$ represents the evaluation result. If the ball misses the endpoint, it identifies failure reasons such as speed or angle deviations: $\mathcal{D}_v = \mathcal{F}_{VLM}(\mathbf{e}, \mathcal{S}; \mathcal{OV})$

and $\mathcal{D}_\theta = \mathcal{F}_{VLM}(\mathbf{e}, \mathcal{S}; \mathcal{OV})$. These descriptions specify parameter deviations from optimal values. Using these evaluations, the model refines hitting parameters with $\mathbf{p}' = \mathcal{F}_r(\mathbf{p}, \mathbf{e}, \mathcal{D}_v, \mathcal{D}_\theta; \mathcal{OV})$. This iterative process improves the system's precision and accuracy based on historical data.

Spatial and kinodynamic understanding VLM. To enhance the specific capabilities of VLM through fine-tuning, we collected 500 video sequences of a robot hitting minigolf balls on various courses. The hitting parameters were randomly generated around successful hit metrics. We employed a combined approach to semi-automatically label the data using GPT-4V (OpenAI, 2023) and Claude3 (Anthropic, 2024). Subsequently, we fine-tuned LLaVA (Liu et al., 2023b; 2024b; 2023a; Li et al., 2024) with the auto-labeled data, which included question-answer pairs for each image. Our finetuned VLM shines in spatial and kinodynamic understanding for the minigolf game.

3.3. Outer closed-loop reflective equilibrium

The outer reflective equilibrium module acts as the higher-level reasoning loop beyond the episodic rollout. Specifically, counterfactual reasoning assesses the feasibility of the course based on the evaluation of previous rollouts. Upon identifying a task as infeasible, it actively suggests physical modifications to the scenario. After a human operator rearranges the setup as suggested, the next iteration commences.

In Figure 10a, to hit the golf ball into the yellow round target, it needs to traverse a roller coaster obstacle. The action reasoning determines that a very strong force is required. After applying the maximum force in the correct direction, the execution results reveal that the ball cannot pass through the roller coaster and instead rolls backward. The counterfactual reasoning component identifies the endpoint as infeasible and actively suggests adding a redirecting surface, such as a yellow curved obstacle.

Counterfactual reasoning VLM. To obtain the counterfactual reasoning capability of VLM, we collect images of 500 different impossible courts to finetune the LLaVA. We autolabel the data using the rule-based method combined with the GPT-4V. The types of impossibilities are varied, including unreachable goals due to limitation of the robot arm, obstacles standing in the way, or slightly misadjusted obstacle angles. For more details refer to Appendix B.2.1.

4. Experimental Results

This study mainly targets the VLM inference framework. But even with comparably precise actuation through the robot arm, the episode dynamics upon hitting a ball remain a stochastic process. To circumvent this additional challenge for this study, we restrict our methods and qualitative evaluation to *offline dataset analysis*. We utilize the before-

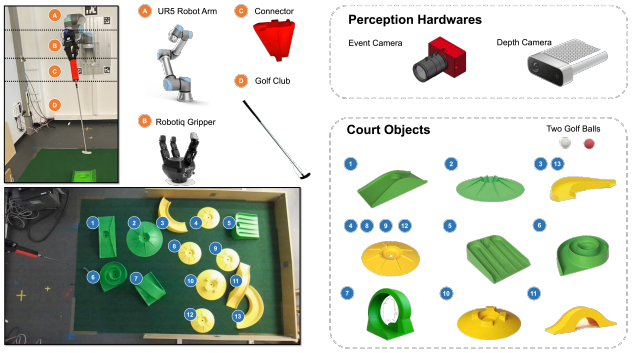


Figure 3. **RoboGolf hardwares and setup.**

mentioned 500 video sequences of classified episodes to approximate the parameter landscape of 25 court arrangements around a successful trial.

After each action prediction, we project the selected parameters onto the closest sample in the dataset and assume it as execution result.

For the study paradigm, we adopt the hardware and course setup depicted in Figure 3. A UR5 arm with Robotiq 3-finger gripper holds a minigolf club fixed in a 3D-printed adapter. We include 9 distinct court obstacles commonly found in minigolf and two differently-colored golf balls, supporting a diverse set of courses to explore.

4.1. Action refinement on exemplary courts

The first condition involves repeated episode rollouts, using the inner-loop action refinement to optimize hitting parameters and putt the ball. The main challenge lies in the required spatial and kinodynamic understanding to plan a feasible ball route and generate the necessary hitting parameters. We have crafted multiple levels of challenges to test the system's capabilities. In the figures below, the dotted lines show a successful trajectory, while the marks below indicate the success or failure of the system to identify the correct hitting parameters in each iteration.

Simple courts. Initially, we evaluate single goals and apply our inner closed-loop control to iteratively improve the hitting parameters, directing the ball to the goal. Generally, the system provides reasonable initial guesses Our RoboGolf system demonstrates robustness in handling these variations, as shown in Figure 4.

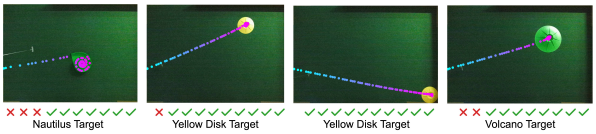


Figure 4. **Performance in simple courts.**

Medium courts. For the medium courts, we consider challenges involving two obstacles. The system successfully reasons about various challenges, including endpoint posi-

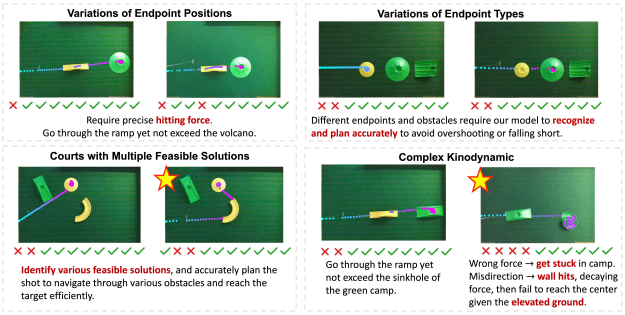


Figure 5. **Performance in medium courts.**

tions, endpoint types, multiple feasible solutions, and complex kinodynamic, as shown in Figure 5. The model must recognize different endpoints and obstacles and adjusting hitting parameters to avoid errors. It identifies feasible solutions and efficiently plans shots to navigate the obstacles. Additionally, it handles complex scenarios, such as circumventing wrong goals or misdirecting the ball into walls, by adjusting to elevated or uneven terrain.

Complex courts. This condition features overly full designs with numerous redundant obstacles strategically placed to confound players and multiple feasible routes leading to different endpoints. Figure 6 highlights the systems reasoning capabilities in these scenarios.

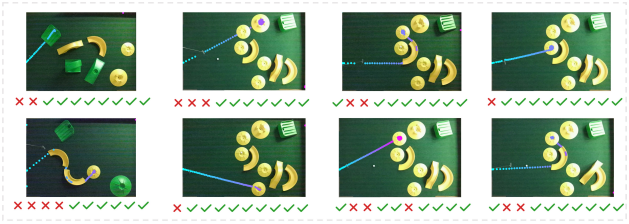


Figure 6. **Performance in complex minigolf course layouts.**

Billiard challenge. To challenge the framework beyond single-ball trajectories, we design a court with multi-object interaction, illustrated in Figure 7. The goal is to hit the red ball from the starting point to bump the white ball, which is initially positioned at a different location, into the yellow round endpoint. This task significantly challenges the kinodynamic understanding of the VLM and the system can infer correct hitting parameters within 6 attempts.

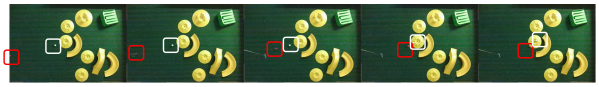


Figure 7. **Key frames of bilateral golf balls impact challenge.**

4.2. Remediate the impossible via active modifications

The key challenge in actively modifying the golf course to make impossible tasks feasible lies in first determining that a task is indeed impossible and then proposing reasonable modifications. The experiments demonstrate that RoboGolf could identify infeasible tasks and modify the court to achieve successful outcomes.

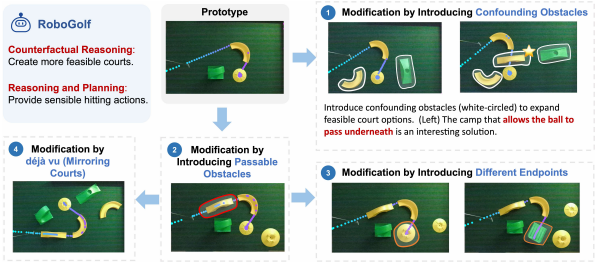


Figure 8. **Active golf court design.**

Impossibles courts. Determining whether a minigolf challenge is impossible involves counterfactual reasoning. Utilizing counterfactual reasoning, the reflective VLM can reveal scenarios where no viable route exists from the starting point to the desired endpoint, as indicated by predicted obstacle maps. In this case, the task is deemed impossible. A more challenging form of impossibility emerges in dynamic limitations. An illustrative example can be found in Figure 10a. Initially, the system may perceive the task as feasible, envisioning the golf ball traversing the roller coaster. However, despite attempting to strike with maximum velocity, the ball does not overcome the obstacle and rolls back. Consequently, RoboGolf concludes the task to be impossible due to hardware constraints.

Active modifications. RoboGolf can actively modify courts to either remediate the impossible or realize new golf court designs, resulting in more feasible court variants. To convert an impossible task into a feasible one, RoboGolf recommends incorporating a redirecting surface, as shown in Figure 10a. This alteration effectively changes the trajectory, making the task achievable. Furthermore, the system can enhance existing feasible setups to create evolved court variants, as depicted in Figures 8 and 10b. RoboGolf devises several evolved courts by adding confounding obstacles, varying endpoints, passable obstacles, and mirroring courts. These adjustments not only diversify the experimental conditions but also offer deeper insights into the system's adaptability and performance under different scenarios.

5. Conclusion

RoboGolf explores the potential of multi-modal VLMs in enhancing robotic autonomy. It exhibits a sophisticated understanding of complex minigolf courts and showcases the use of reflective equilibrium reasoning to assess feasibility and proactively propose court adjustments where needed.

Our analysis has shown promising results, and we target future work along the paradigm. Prominently, we aim to include uncertainty in action execution and deploy the system in online settings. A second promising extension targets the autonomous obstacle rearrangement through suggestions from the outer reflective loop.

Acknowledgement

This work was supported by the DFG Transregional Research Centre CML, TRR-169. We would like to express our sincere gratitude to Tom Sanitz for the shared initial conceptualization of the robotic paradigm.

References

Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R., and Yin, W. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Anthropic. Introducing the next generation of claude, March 2024. URL https://www.anthropic.com/news/claude-3-family. Accessed: 2024-05-29.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bhattacharyya, A., Malinowski, M., Schiele, B., and Fritz, M. Long-term image boundary extrapolation. *CoRR*, 2016.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Cath, Y. Reflective equilibrium. *The Oxford handbook of philosophical methodology*, 1, 2016.

Daniels, N. Wide reflective equilibrium and theory acceptance in ethics. *The journal of philosophy*, 76(5):256–282, 1979.

Durante, Z., Sarkar, B., Gong, R., Taori, R., Noda, Y., Tang, P., Adeli, E., Lakshmikanth, S. K., Schulman, K., Milstein, A., et al. An interactive agent foundation model. *arXiv preprint arXiv:2402.05929*, 2024.

Fragkiadaki, K., Agrawal, P., Levine, S., and Malik, J. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

Girdhar, R., Gustafson, L., Adcock, A., and van der Maaten, L. Forward prediction for physical reasoning. *arXiv preprint arXiv:2006.10734*, 2020.

Groth, O., Fuchs, F. B., Posner, I., and Vedaldi, A. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the european conference on computer vision (eccv)*, pp. 702–717, 2018.

Hu, Y., Lin, F., Zhang, T., Yi, L., and Gao, Y. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

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. In *Conference on Robot Learning*, pp. 540–562. PMLR, 2023.

iniVation AG. Davis346. PDF Document, August 2019. URL https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf.

iniVation AG. dv-processing Documentation, rel_1_7. Online Documentation, April 2022. URL https://dv-processing.inivation.com/rel_1_7/index.html.

iniVation AG. DV-GUI Documentation. Web Page, April 2024. URL https://docs.inivation.com/software/dv/gui/index.html.

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.

Lai, X., Tian, Z., Chen, Y., Li, Y., Yuan, Y., Liu, S., and Jia, J. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.

Li, B., Zhang, K., Zhang, H., Guo, D., Zhang, R., Li, F., Zhang, Y., Liu, Z., and Li, C. Llava-next: Stronger llms supercharge multimodal capabilities in the wild, May 2024. URL https://llava-vl.github.io/blog/2024-05-10-llava-next-stronger-llms/.

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 *2023 IEEE*

International Conference on Robotics and Automation (ICRA), pp. 9493–9500. IEEE, 2023.

Liu, F., Fang, K., Abbeel, P., and Levine, S. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024a.

Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning, 2023a.

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

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

Maatouk, A., Piovesan, N., Ayed, F., De Domenico, A., and Debbah, M. Large language models for telecom: Forthcoming impact on the industry. *arXiv preprint arXiv:2308.06013*, 2023.

Microsoft. Azure kinect dk. Microsoft Store, 2024. URL https://www.microsoft.com/en-us/d/azure-kinect-dk/8pp5vxmd9nhq?activetab=pivot:overviewtab. Accessed on: 2024-05-29.

OpenAI. GPT-V System Card. OpenAI Website, 2023. URL https://cdn.openai.com/papers/GPTV_System_Card.pdf.

Rebecq, H., Ranftl, R., Koltun, V., and Scaramuzza, D. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1964–1980, 2019.

Westny, T., Olofsson, B., and Frisk, E. Diffusion-based environment-aware trajectory prediction. *arXiv preprint arXiv:2403.11643*, 2024.

Yang, S., Qu, T., Lai, X., Tian, Z., Peng, B., Liu, S., and Jia, J. An improved baseline for reasoning segmentation with large language model. *arXiv preprint arXiv:2312.17240*, 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.

Zeng, F., Gan, W., Wang, Y., Liu, N., and Yu, P. S. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023.

Zhi, P., Zhang, Z., Han, M., Zhang, Z., Li, Z., Jiao, Z., Jia, B., and Huang, S. Closed-loop open-vocabulary mobile manipulation with gpt-4v. *arXiv preprint arXiv:2404.10220*, 2024.

A. Illustrations of RoboGolf Execution

A.1. Illustration of inner closed-loop action refinement.

Inner closed-loop action refinement involves reasoning and planning, executing hitting actions with a real-world robot arm, and recovering from failures through evaluation. As shown in Figure 9, the process begins with initial planning, calculating the required hitting force and angle based on the endpoint's distance and obstacle positions. If the initial attempt fails, the system iteratively adjusts the hitting parameters—such as force and angle—based on trajectory evaluations until the golf ball successfully reaches the target.

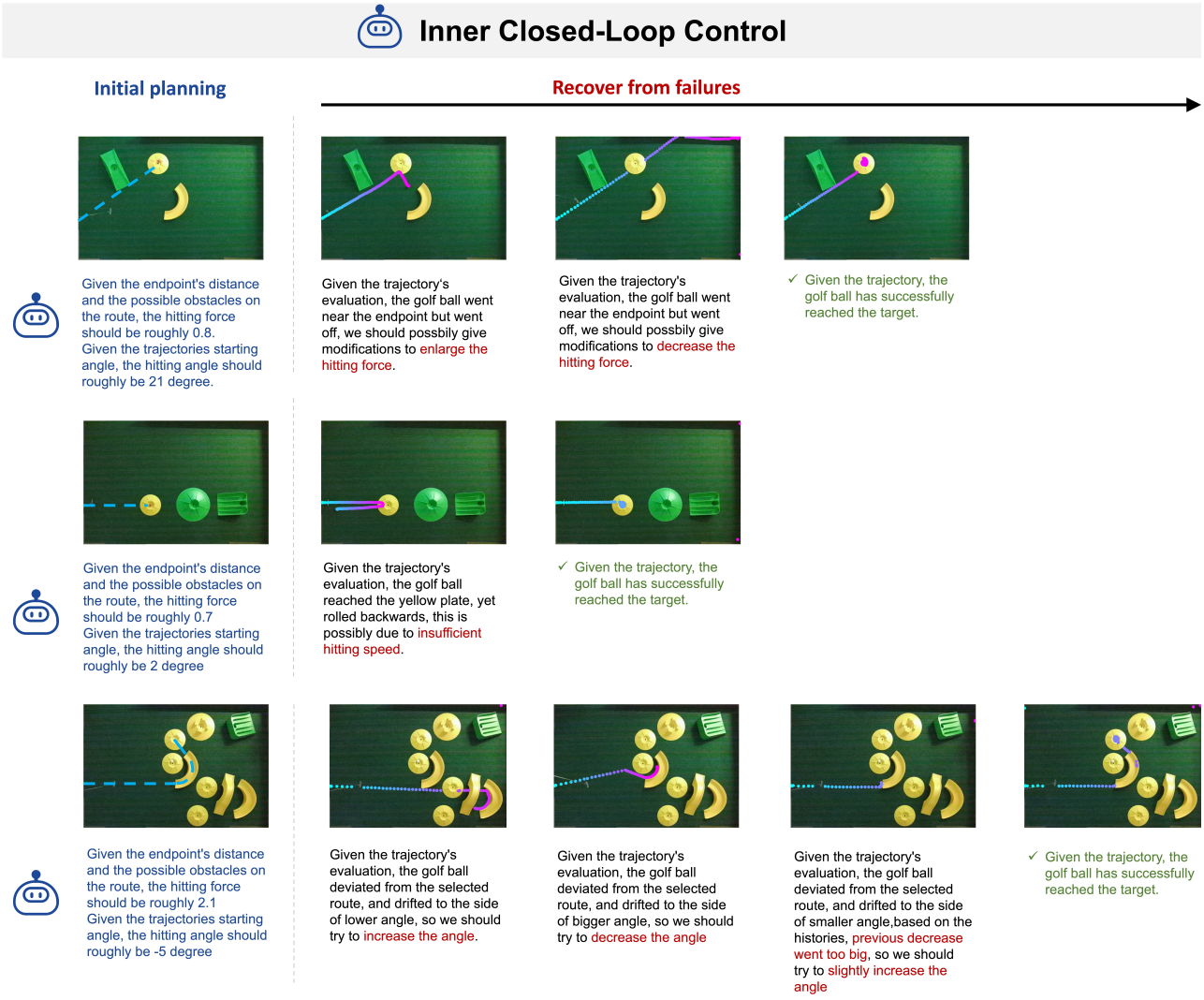
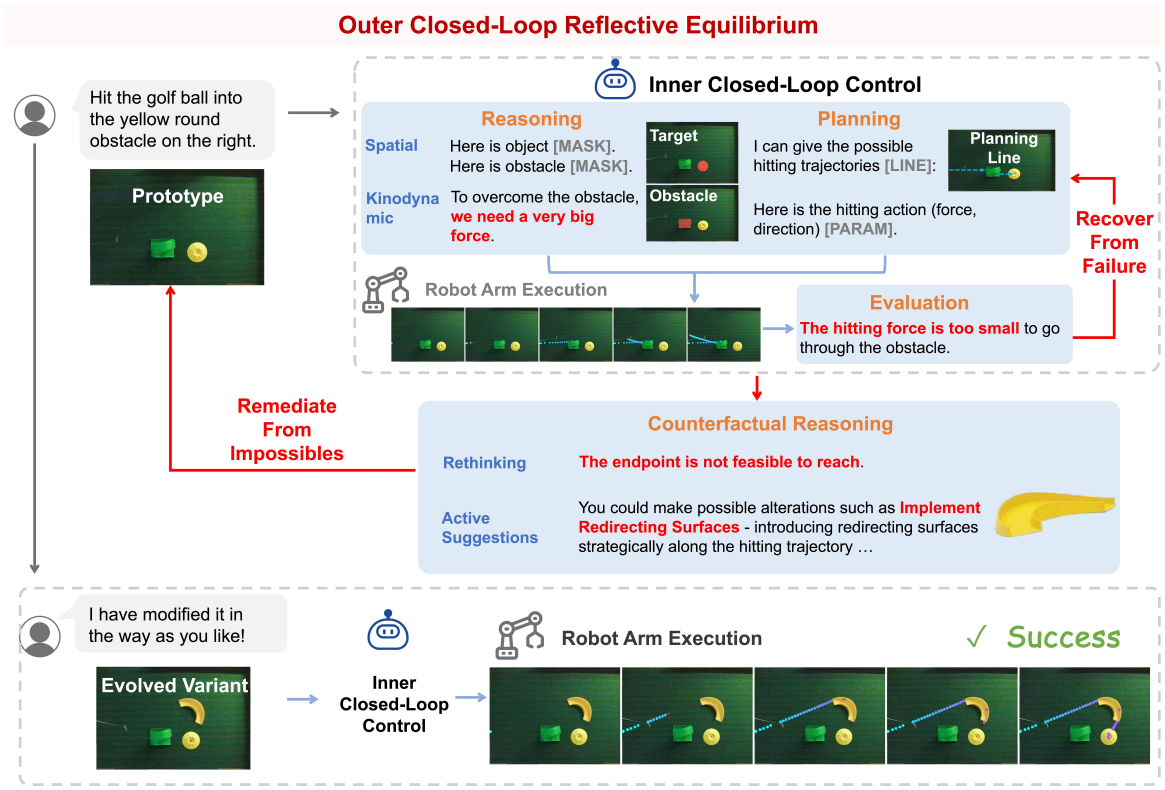


Figure 9. **Illustration of the inner loop action refinement of RoboGolf.**

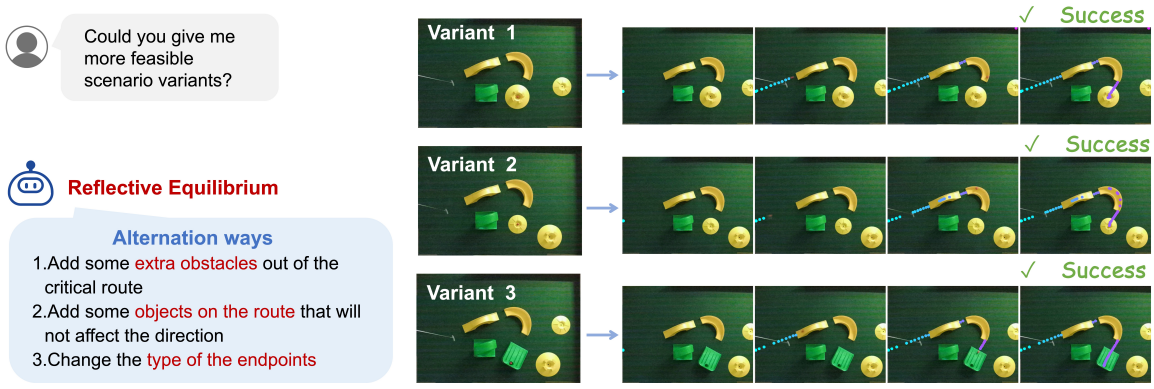
A.2. Illustration of outer closed-loop reflective equilibrium.

In Figure 10a, RoboGolf attempts to hit the golf ball into a yellow round target, which requires traversing a roller coaster obstacle. Initially, the inner loop requires a very strong force. Despite applying the maximum force in the correct direction, the execution results reveal that the ball cannot pass through the roller coaster and instead rolls backward. RoboGolf's counterfactual reasoning identifies the endpoint as infeasible and actively suggests adding a redirecting surface, such as a yellow curved obstacle.

RoboGolf can enhance existing feasible setups to create evolved court variants. As depicted in Figure 10b, RoboGolf devises several evolved courts by adding confounding obstacles, varying endpoints, and passable obstacles. These adjustments not only diversify the experimental conditions but also offer deeper insights into the system's adaptability and performance under different scenarios.



(a) **Illustration of outer closed-loop reflective equilibrium.**



(b) **Illustration of the active modification capability of RoboGolf.**

Figure 10. **Illustration of outer closed-loop reflective equilibrium and the counterfactual reasoning ability of RoboGolf.**

B. Implementation Details

B.1. Perception

B.1.1. Event camera

Calibration of the Event Camera For intrinsic parameters, the distortion of the event camera is relatively significant, we use the calibration method provided by the DV GUI (iniVation AG, 2024). We use a metal-based checkerboard to calibrate the intrinsic parameters and save the intrinsic calibration parameters for later usage.

For the extrinsic parameters, we selected the event camera model with RGB+event information to align the event information with the RGB-D camera.

Recordings We use the DV GUI application to record information. As demonstrated in Figure 11 the event information first undergoes noise filtering and then proceeds to the undistortion module, utilizing the parameter file from the calibration. Next, we use the streaming module to stream the event information and RGB data. Subsequently, we employ the DV-processing (iniVation AG, 2022) Python code to write a server that records both the event and RGB information.

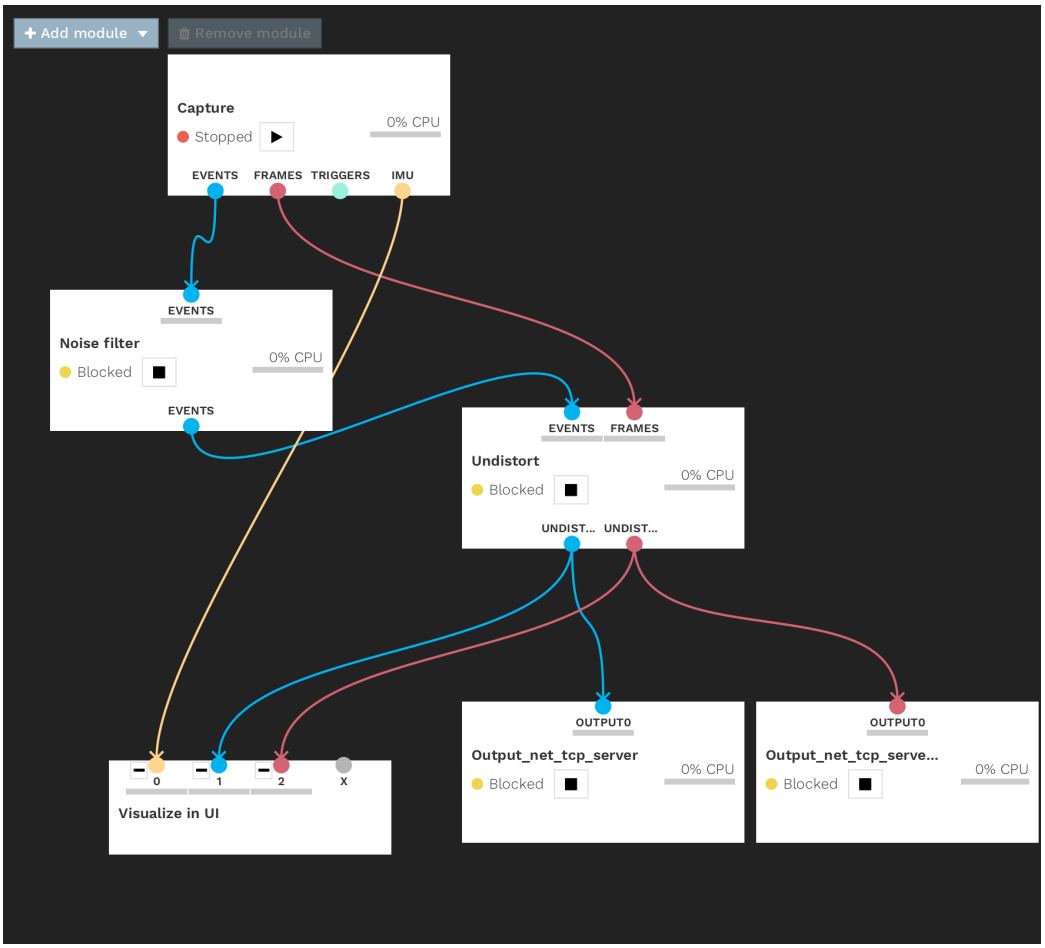


Figure 11. **Recorder flow for event and RGB information.**

B.1.2. RGB-D camera

For the selection of the RGB camera, we adopted the camera Moto from Microsoft Azure Kinnect, it is a structure light-based decks camera with RGB information.

Before the actual application of the RGB camera, we first calibrate the intrinsic parameters to avoid this distortion and

then we also calibrate the relationship between the depth information and the RGB information in order to match them to a well-aligned cartesian space.

The recording process is conducted using the ROS recording system due to the nature of the data generated by the RGBD camera. It cannot process and store all the data in real-time causing certain frames to be lost therefore we have to adopt the ROS system to record the road, and after the recording, we can then process the data to store them as videos.

B.2. Technical details

B.2.1. VLM FINETUNE

Auto label data The preparation of data is crucial for the training of vision language models, serving two distinct purposes that necessitate a separate labeling mechanism, albeit with a shared processing mechanism.

The auto-labeling process comprises several stages:

1. **Stage 1:** We employ rule-based traditional methods to detect the rotation and direction of obstacles. Specifically, we adopt Structure-Aware Masks (SAM) to generate masks for objects on the ground and subsequently utilize rule-based methods to match with the original model, thereby obtaining a rotational angle.
2. **Stage 2:** We track the ball across the court and segment and extract critical frames where the ball interacts with obstacles on its path.
3. **Stage 3:** Finally, we generate data with questions and answers based on the previous information. This stage marks the separation of the dataset dedication for different purposes. The exact measures for generating data are as follows:

kinodynamic understanding data To improve the kinodynamic understanding aspect of the VLM, we evaluated the current comprehension abilities of leading motors. Given that large multimodality models and vision-language models tend to generate unsatisfactory answers and potential hallucinations when asked vague questions, we are dividing the labor required for this fine-tuning process.

We prepared several types of questions to label the data for training the visual language model. The first type of question addresses the number of obstacles on the court and their positional relationships.

The second type of question is that the evaluation of whether the ball hits the goal successfully or not.

The third type of question, which is similar to the second, is concerned with the reasons behind the ball's deviation from its endpoint. This requires us to furnish the auto-labeling vision language models with extensive information about the golf ball's trajectory and speed circumstances.

reflective equilibrium To prepare the data of finetuning the visual language models to make it capable of providing reflective equilibrium by counterfactual reasoning, we have to label the data with questions focus more on the relationship between the obstacles. Collected scenarios are combined with an evaluation of the hitting histories constituting pairs of providing hitting histories and questions and a judgment of impossible.

B.2.2. INSTRUCTIONS

Endpoint Intention Sementations Firstly the framework provides with a top-down RGB image of the court with the human's instruction describing which endpoint to select then firstly the prompts and the RGB image are provided to the vision language model LISA, to recognize the masks and the spatial relationship with the others. We also use the finetuned VLM to provide some basic comments on the situation.

Detailed Prompt Example for Vision-Based Miniature Golf Endpoint Detection

Purpose: This prompt is designed to enable a vision language model to accurately identify and segment the desired endpoint in a miniature golf setup using a provided top-down RGB image and user-defined language instructions. This process supports the robotic arm in executing a precise, single-shot attempt.

Input Requests

- **Image of the Miniature Golf Court:**
  - *Description:* A top-down RGB image of the 2-meter by 3-meter golf court featuring various plastic obstacles and designated endpoints.
- **User Instruction:**
  - *Description:* Textual instructions specifying the target endpoint for the golf ball. Example: "Identify the endpoint near the large blue obstacle on the right side."

Output Response

- **Format:** Image with segmentation mask overlay on the desired endpoint:

  Example of expected output: A modified version of the input image highlights the segmented endpoint with a distinct color overlay, distinguishing it from other elements in the scene.

Rule Definitions and Reasoning Steps

1. **Image Preprocessing:**
   - Perform necessary adjustments to enhance image quality for better feature recognition (e.g., brightness, contrast).
2. **Endpoint Detection:**
   - Apply deep learning techniques to interpret and segment the area described in the user's instructions.
   - Utilize natural language processing to correlate textual descriptions with visual features in the image.
3. **Segmentation and Annotation:**
   - Use semantic segmentation algorithms to isolate and highlight the designated endpoint.
   - Annotate the segmented area clearly to guide the robotic arm in targeting.
4. **Output Validation:**
   - Ensure the accuracy of the segmentation by cross-referencing the visual output with the user's original instructions.

Route Planner Design In this study, we address the challenge of developing a robotic golf ball playing assistant that can intelligently plan a path across a golf course by avoiding obstacles and optimizing trajectory. To this end, we propose a structured approach for path planning, which includes both high-level strategic planning and detailed motion planning. Our system takes as input a detailed description of the golf course and a top-down annotated image showing key obstacles and target locations. Based on this input, the system outputs a plan in the form of a JSON dictionary, which details waypoints, motion directions, and specific actions at each stage of the route. Key components of our solution include 1) the identification of critical key points where the golf ball should pass or change direction, 2) the determination of optimal waypoints that guide the ball along the safest and most efficient trajectory, and 3) the visualization of this route on the annotated map to verify and adjust the plan as needed. Each step is designed to ensure that the robot can effectively parse natural language instructions and translate them into a practical and executable motion plan, thus enhancing the robot's capabilities in outdoor sports environments.

Detailed Prompt Example for Robotic Golf Ball Playing Assistant

Purpose: This prompt is designed to guide the robotic assistant in selecting an optimal route for a golf ball on a

simulated golf course, given a top-down view image of the court with annotated assist lines and obstacle detections, to achieve a successful shot in a single attempt.

Input Requests

- **Task Information:**
  - *Instruction:* **"Select the best route for the golf ball to reach the hole, avoiding all obstacles and minimizing the stroke count, given the provided top-down view image of the court."**
  - *Desired Endpoint:* The hole location on the golf course.
- **Image of the Court:**
  - *Description:* Annotated top-down view image of the golf course, including:
    - *Assist Lines:* Possible pathways for each obstacle detected.
    - *Obstacle Detections:* Locations of sand bunkers, water hazards, and other obstacles.
    - *Desired Points:* Annotated points of interest, such as the hole location.

Output Response

- **Format:** JSON dictionary with strategic planning details:

```
{
  "route": [
    {"keypoint": "Selected striking point 1",
     "angle": "20 degrees to the north"},
    {"keypoint": "Selected striking point 2",
     "angle": "15 degrees to the east"},
     ...
    {"keypoint": "Hole", "angle": "0 degrees"}
  ],
  "stroke_count": 1,
  "total_distance": 50.0
}
```

Rule Definitions and Reasoning Steps

1. **Keypoint Selection:**
   - Identify and mark feasible hitting points considering obstacle locations, estimated ball trajectory, and assist lines.
   - Evaluate the feasibility of each key point based on the robotic arm's capabilities and the golf ball's dynamics.
2. **Route Planning:**
   - Define a sequence of key points to form a route that minimizes the stroke count and total distance traveled.
   - Optimize the route by adjusting the key points and angles to ensure a successful shot.
3. **Endpoint Verification:**
   - Verify that the selected route reaches the desired endpoint (the hole).
   - Adjust the route if necessary to ensure a successful shot.

Parameter Generation

Detailed Prompt Example for Robotic Minigolf Playing Assistant

Purpose: This prompt is designed to guide the robotic assistant in estimating the hitting parameters for a mini golf shot, given a selected route and image of the court, to achieve a successful shot in a single attempt.

Input Requests

- **Route Information:**
  - *Route JSON:* JSON dictionary with the selected route, including key points and angles:

```
{
    "route": [
        {"keypoint": "Selected striking point 1",
            "angle": "-20 degrees"},
        {"keypoint": "Selected striking point 2",
            "angle": "15 degrees"},
        ...
        {"keypoint": "Hole", "angle": "0 degrees"}
    ],
    "stroke_count": 1,
    "total_distance": 50.0
}
```

- **Image of the Court:**
  - *Description:* Top-down view image of the mini golf court, including obstacles and endpoints.

Output Response

- **Format:** JSON dictionary with hitting parameters:

```
{
    "hitting_angle": 25.0,
    "hitting_speed": 0.7,
    "confidence_score": 0.9
}
```

Rule Definitions and Reasoning Steps

1. **Hitting Parameter Estimation:**
   - Utilize the provided route and image to estimate the hitting angle and speed required to reach the hole.
   - Consider the robotic arm's capabilities and the golf ball's dynamics in the estimation process.
2. **Confidence Score Calculation:**
   - Calculate a confidence score based on the estimated hitting parameters and the selected route.
   - Adjust the hitting parameters if necessary to ensure a successful shot.

Hitting trajectory evaluation

Detailed Prompt Example for Robotic Minigolf Playing Assistant

Purpose: This prompt is designed to guide the robotic assistant in evaluating the hitting trajectory and parameters of a mini golf shot, given the recorded trajectory and original route estimation, to identify areas for improvement and provide suggestions for modification.

Input Requests

- **Trajectory Information:**
  - *Recorded Trajectory:* The actual trajectory of the golf ball during the previous execution.
  - *Original Route Estimation:* The originally expected route estimated by connecting the key points and trajectory histories.
  - *Hitting Parameters:* The hitting speed and angle used in the previous execution.

Output Response

- **Format:** JSON dictionary with evaluation and suggestion details:

```
{
    "endpoint_reached": true/false,
    "deviation_reason": "insufficient speed" / "incorrect angle" /...,
    "modification_suggestion": {
        "parameter": "hitting speed" / "hitting angle",
        "direction": "increase" / "decrease",
        "amount": 0.1
    }
}
```

Rule Definitions and Reasoning Steps

1. **Endpoint Verification:**
   - Check if the golf ball stopped in the desired endpoint (under the red mask).
   - If yes, no further evaluation is needed.
2. **Deviation Analysis:**
   - Identify the reason for deviation from the endpoint (e.g., insufficient speed, incorrect angle).
   - Evaluate the deviation from several perspectives (e.g., speed, angle, trajectory shape).
3. **Modification Suggestion:**
   - Based on the deviation reason, suggest modifying the hitting speed or angle in a specific direction and amount.
   - Provide a detailed explanation for the suggestion.

Feasibility Evaluations

Detailed Prompt Example for Robotic Minigolf Playing Assistant

Purpose: This prompt is designed to guide the robotic assistant in evaluating the feasibility of a mini golf court scenario, given the image with assistive lines and obstacle masks, as well as the evaluation of previous hitting attempts, to determine whether the court is feasible and provide reasons for impossibility.

Input Requests

- **Court Information:**
  - *Image with Assistive Lines:* Top-down view image of the mini golf court with annotated assistive lines.
  - *Obstacle Masks:* Masks of the plastic obstacles on the court.
- **Previous Attempt Evaluation:**
  - *Hitting Attempt Results:* Evaluation of previous hitting attempts, including success/failure and reasons for failure.

Output Response

- **Format:** JSON dictionary with feasibility decision and reason:

```
{
    "feasibility": true/false,
    "reason": "insufficient space" / "obstacle blocking" /...
}
```

Rule Definitions and Reasoning Steps

1. **Court Analysis:**
   - Analyze the court image with assistive lines and obstacle masks to identify potential obstacles and challenges.
   - Evaluate the feasibility of the court based on the robotic arm's capabilities and the golf ball's dynamics.
2. **Previous Attempt Consideration:**
   - Consider the evaluation of previous hitting attempts to identify patterns and reasons for failure.
   - Use this information to inform the feasibility decision.
3. **Feasibility Decision:**
   - Based on the court analysis and previous attempt consideration, make a feasibility decision (true/false).
   - Provide a reason for the decision, if the court is deemed infeasible.

Modification Suggestions

Detailed Prompt Example for Robotic Minigolf Playing Assistant

Purpose: This prompt is designed to guide the robotic assistant in suggesting modifications to the mini golf course or golf ball selection to achieve reflective equilibrium, making the task possible, given the evaluation results and possible modification types.

Input Requests

- **Evaluation Results:**
  - *Visibility Value:* The current visibility value of the golf ball on the court.
  - *Modification Types:* The types of modifications that can be applied to the current status, including:
    - Removing obstacles
    - Setting obstacles

    - Watering angles and positions of existing obstacles
    - Changing the endpoint
    - Adding new obstacles

- **Obstacle Information:**
  - *Descriptions and Names:* Descriptions and names of the obstacles that can be added, including their properties and effects on the game.

Output Response

- **Format:** JSON dictionary with exact execution instructions:

```
{
    "modification_instructions": [
        {"type": "remove", "obstacle": "obstacle 1"},
        {"type": "add", "obstacle": "obstacle 2", "position": "top-left",
            "angle": "30 degrees"},
        {"type": "change", "endpoint": "new endpoint location"}
    ]
}
```

Rule Definitions and Reasoning Steps

1. **Modification Analysis:**
   - Analyze the evaluation results and possible modification types to identify the most effective modifications.
   - Evaluate the feasibility of each modification based on the robotic arm's capabilities and the golf ball's dynamics.
2. **Obstacle Selection:**
   - Select the most suitable obstacles to add or remove based on their properties and effects on the game.
   - Determine the optimal positions and angles for the added obstacles.
3. **Endpoint Adjustment:**
   - Determine if the endpoint needs to be changed to achieve reflective equilibrium.
   - Calculate the new endpoint location if necessary.
4. **Instruction Generation:**
   - Generate exact execution instructions based on the modification analysis, obstacle selection, and endpoint adjustment.
   - Provide a detailed explanation for the suggested modifications.

B.2.3. Robot Trajectory Generation

We compute the robot trajectory to hit the ball with the head of the club based on two input parameters for hitting angle and hitting velocity and assume a fixed Cartesian position of the ball. Initially, the system computes a joint configuration through inverse kinematics to reach the golf ball position with the head of the minigolf club at the correct angle. To achieve a target Cartesian velocity of the club at the ball location, we consider a second inverse kinematics solution assumed to be reached 100ms before the ball location, thus fixing the Cartesian position to the ball location given the target velocity. The resulting two waypoints define the plateau of a trapezoidal velocity profile in joint space and the corresponding ramp

up/ramp down phases are computed with pre-determined accelerations. Based on the kinematic design of our setup and the dynamics limits for the robot joints, the ball position always constitutes the lowest point of the computed trajectory. All motions are kinematically validated through the MoveIt framework before execution.

C. Experimental Setups

C.1. Robot Hardware

For the robot, we adopted the UR5 robot arm, mounted on the wall. It is a 6 DoF robot arm with reasonable accuracy. The gripper we employed the 3-Finger Adaptive Robot Gripper from Robotiq, is renowned for its high versatility and adaptability across varied robotic applications. This advanced gripper is designed to efficiently handle objects of almost any shape, thanks to its unique ability to adjust grip based on the object's form. Equipped with multiple gripping modes and precise control over each finger, the gripper integrates seamlessly with systems like Universal Robots, making it a prime choice for intricate tasks in modern automated environments.

The golf club's holder is 3D-printed, constituted by two parts. The first part is the holder that fits the 3-finger gripper, the second part is the gripper for the golf club. The two parts are printed using PLA material and glued together.

C.2. Minigolf settings

The minigolf setting, sourced as a complete set from the manufacturer, is designed to provide everything needed for playing minigolf. This includes the convenience of having all necessary equipment like clubs, balls, scorecards, and a variety of obstacles. These components are engineered for durability, being break-resistant, UV-protected, and weatherproof, making them suitable for both indoor and outdoor use. The modular design allows for the creation of customized courses, maximizing space and enhancing gameplay flexibility, ideal for family entertainment and varied play environments.

The minigolf set comes with different types of balls for selection, in different colours and textures. By testing, we figured that the red-colored ball with a relatively soft texture provides the most stable performance with different obstacles and the most notable optical character in the recordings.

For the ground, we select an artificial grass carpet to better simulate the real golf court. we set the course to be the size of 2 meters by 3 meters.

C.3. Hardware for recording

We adopt the davis 346 (iniVation AG, 2019) for the event info recordings The DAVIS 346 is a high-performance event camera featuring a 346 x 260-pixel dual-mode sensor, combining Dynamic Vision Sensor (DVS) technology and Active Pixel Sensor (APS) capabilities. This model boasts remarkable specs such as a 120 dB DVS dynamic range and minimal latency of around 20 microseconds, suited for precise imaging tasks. Its robust design includes an anodized aluminum case and a CS-mount lens system, making it adaptable for various optical needs. Additionally, the camera is equipped for multi-camera synchronization and supports power and data transfer via a USB 3.0 connection, ensuring versatility in deployment for advanced visual applications.

We adopt the Azure Kinect (Microsoft, 2024) as the RGB-D camera. The Azure Kinect DK represents a significant advancement in sensor technology, integrating a sophisticated depth camera system and a high-resolution color camera into a single device. This tool is designed for developers and commercial applications, featuring a robust set of capabilities including multiple operating modes for depth perception and high-resolution video capture. Notably, the device is housed in a durable case, supports extensive environmental operating ranges, and offers advanced synchronization features. This makes the Azure Kinect DK a versatile and powerful tool for a variety of technological applications.