Autonomous Minigolf Mastery

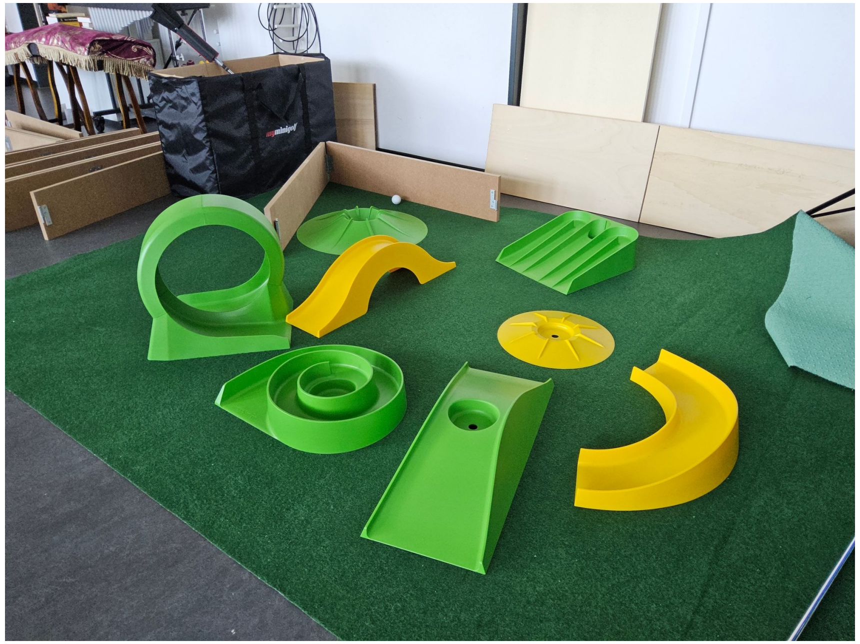
Enabling Real-World Robot Arms
through a Multi-Modality Vision-Language Model Framework

Hantao Zhou

5/7/24

Task Definition

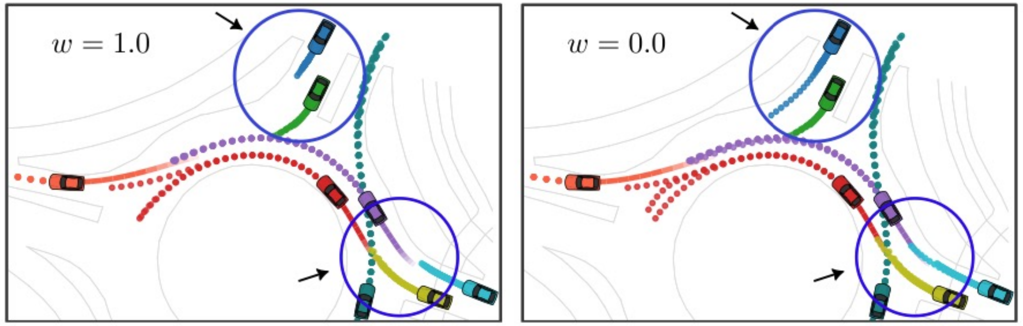
- **Mastering Minigolf in Various Course with a Robot Arm**
  - Learn to how to successfully hit the goal with the ball without human intervention
  - Adaptable to complex courts with different obstacles



Enable robot to integrate visual data with planning and adapt based on outcomes.

Propose novel framework that can handle complex, multimodal data and make decisions accordingly.

General Problem and Significance

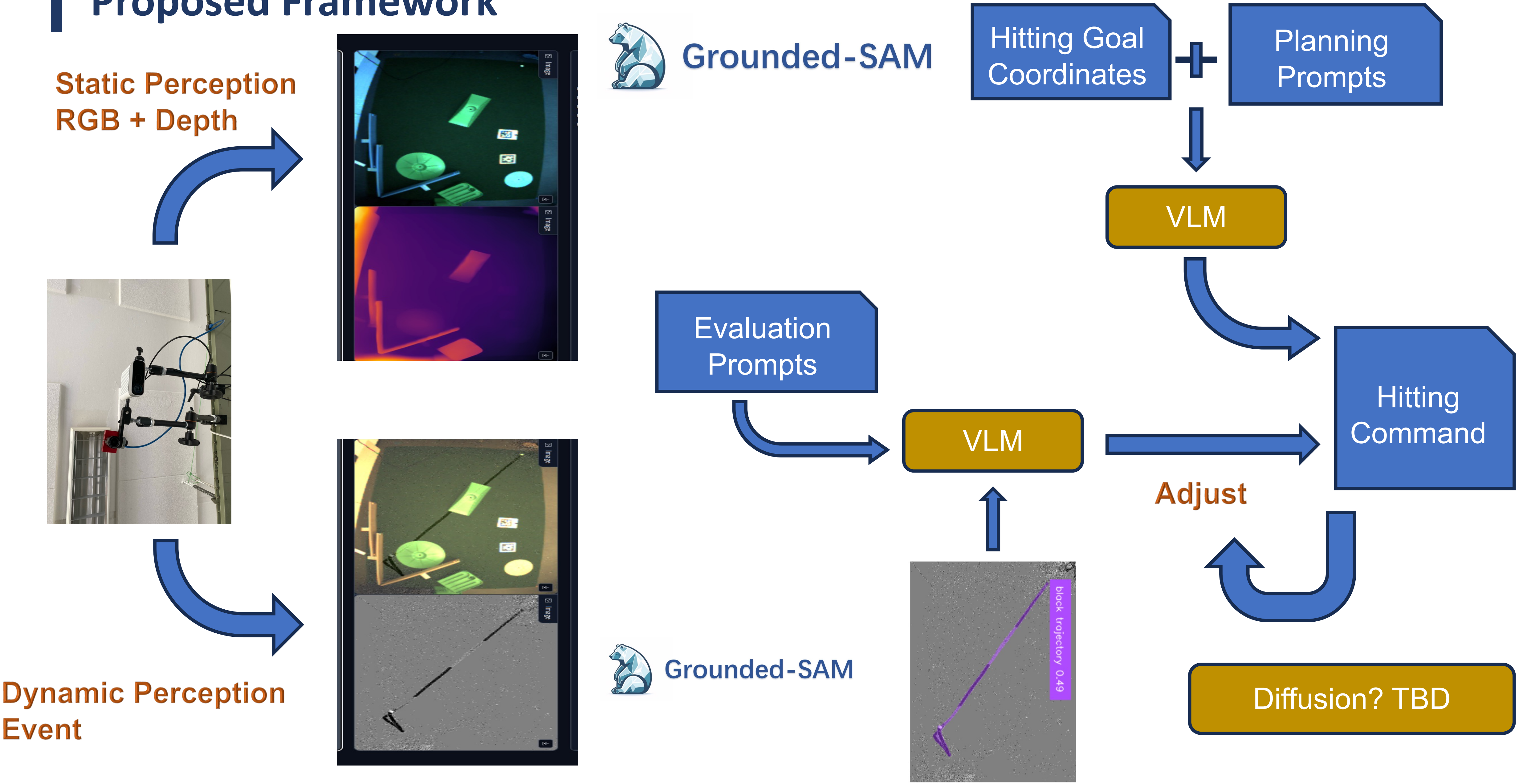


- Complex Trajectories Prediction and Kinodynamic understanding
- Trajectory Prediction
  - CNN, Diffusion
  - A type of behaviours forecasting, highly valuable for many scenarios like self-driving, collision detection
- Kinodynamic Understanding
  - A capability that is crucial for embodied AI
  - VLM currently performs very poor
- These two feature have not been synergistically combined, lacking the possibilities of improving the performances recurrently

Previous VLMs’ deficiencies

- Core VLM’s deficiencies
  - Only harness simple logic of positional relationships rather than the complex nature of trajectories
  - Can’t understand trajectories effectively (no VLM can read video as a continuous file, they just extract key frames)
- Applying VLM in robotics
  - Only serving as high-level planner
  - Only Deterministic Execution, no feedback and rethinking
  - Vision information only as assistance for understanding, with out vision -> same performance (especially “look before you leap”)

Proposed Framework



Static Perception
RGB + Depth

Grounded-SAM

Hitting Goal Coordinates + Planning Prompts

VLM

Evaluation Prompts

VLM

Hitting Command

Adjust

Dynamic Perception
Event

Grounded-SAM

Diffusion? TBD

Challenges

- Current VLM is not precise on details in vision – Especially in the Kinodynamic and Trajectory
- Sim2Real can simulate the environment and generate an agent that can perform reasonably well in real scenarios --- but takes high cost especially there is a huge gap between simulation and reality
- Hard coded control can solve certain scenarios hard-coded way but not adaptable for complex scenarios

Two Main Sub-Tasks

- **Perception**
  - Percept and understand obstacles in environments
  - Track high speed motion at high temporal resolution

- **Planning**
  - Evaluate feasibility of hitting to goal under certain obstacles setup
  - If feasible
    - Plan trajectory based on distances and positions of obstacles
    - Adjust strategy based on hitting recordings

Advanced Tools

Sub-tasks

Perception

- Percept and understand obstacles in environments
- Track high speed motion at high temporal resolution

Planning

- Evaluate feasibility of successful hitting
- Adjust strategy based on hitting recordings

Advanced Tools

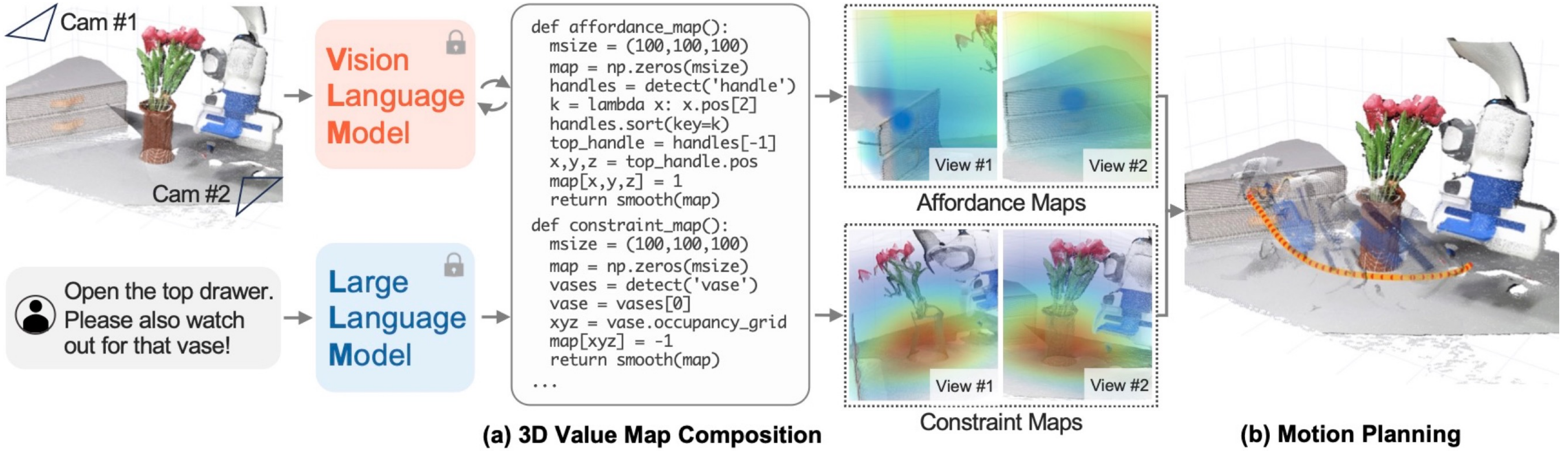
- **VLM – Grounded SAM**
- **Event Camera**

- **Depth Camera**
- **VLM – Llama Adapter**

Related Works

- **VLM+Robotics**
  - VoxPoser
  - MOKA Manipulation

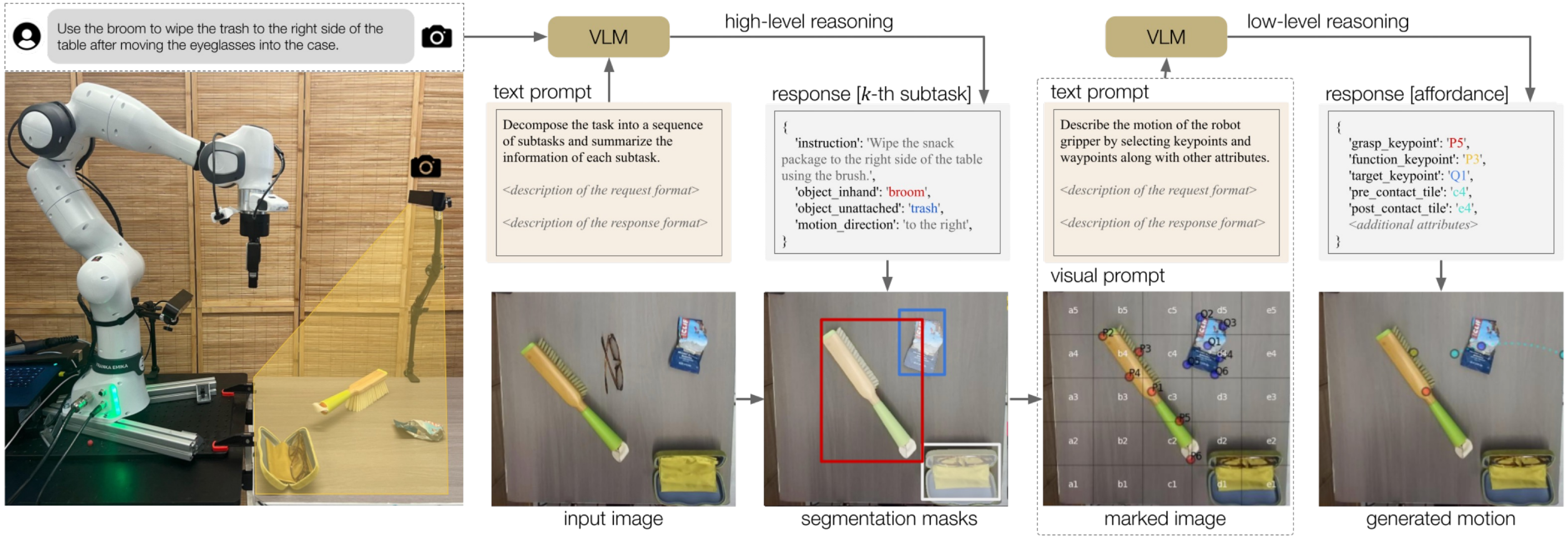
- **Limitaions**
  - **Static Tasks**
  - **No adjustment feedback**



```
def affordance_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  handles = detect('handle')
  k = lambda x: x.pos[2]
  handles.sort(key=k)
  top_handle = handles[-1]
  x,y,z = top_handle.pos
  map[x,y,z] = 1
  return smooth(map)
def constraint_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  vases = detect('vase')
  vase = vases[0]
  xyz = vase.occupancy_grid
  map[xyz] = -1
  return smooth(map)
...
```

(a) 3D Value Map Composition

(b) Motion Planning


Use the broom to wipe the trash to the right side of the table after moving the eyeglasses into the case.

Decompose the task into a sequence of subtasks and summarize the information of each subtask.

<description of the request format>

<description of the response format>

```
{
  'instruction': 'Wipe the snack package to the right side of the table using the brush.',
  'object_inhand': 'broom',
  'object_unattached': 'trash',
  'motion_direction': 'to the right',
}
```

Describe the motion of the robot gripper by selecting keypoints and waypoints along with other attributes.

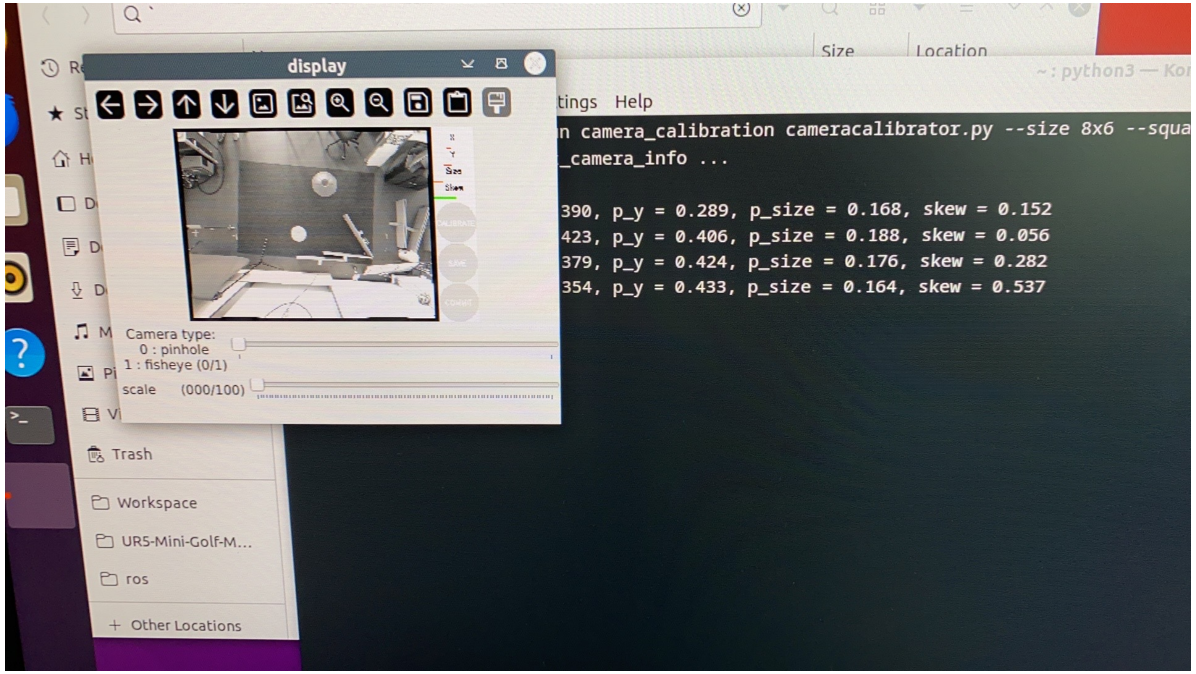
<description of the request format>

<description of the response format>

```
{
  'grasp_keypoint': 'P5',
  'function_keypoint': 'P3',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'e4',
  <additional attributes>
}
```

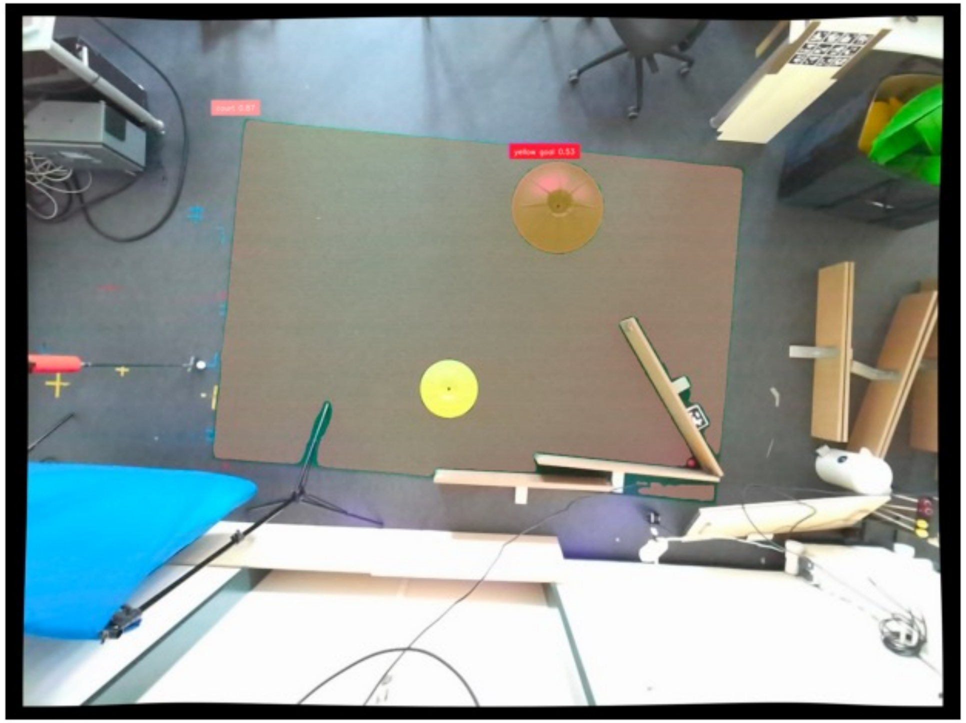
Current Experimental Results

- **RGBD**
  - Calibration
  - Visual Grounding



```
n camera_calibration cameracalibrator.py --size 8x6 --squa
_camera_info ...

390, p_y = 0.289, p_size = 0.168, skew = 0.152
423, p_y = 0.406, p_size = 0.188, skew = 0.056
379, p_y = 0.424, p_size = 0.176, skew = 0.282
354, p_y = 0.433, p_size = 0.164, skew = 0.537
```

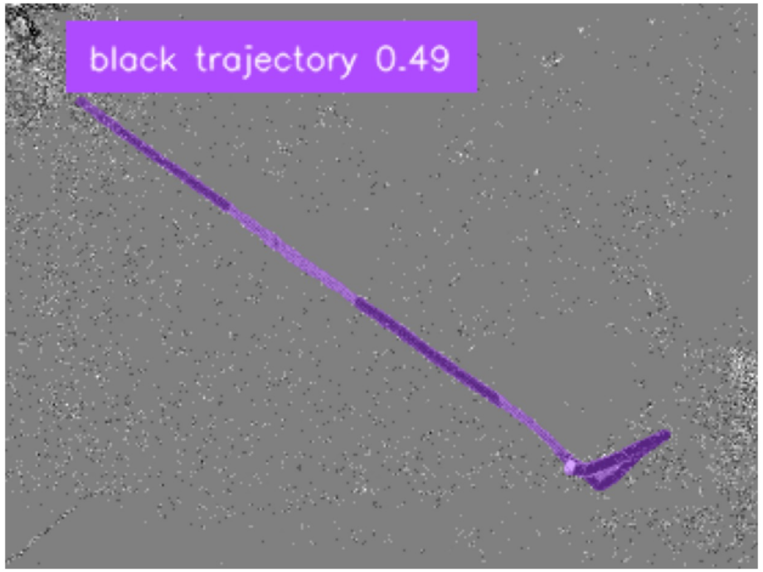


Current Experimental Results

- **Dynamic Perception**
  - Calibration

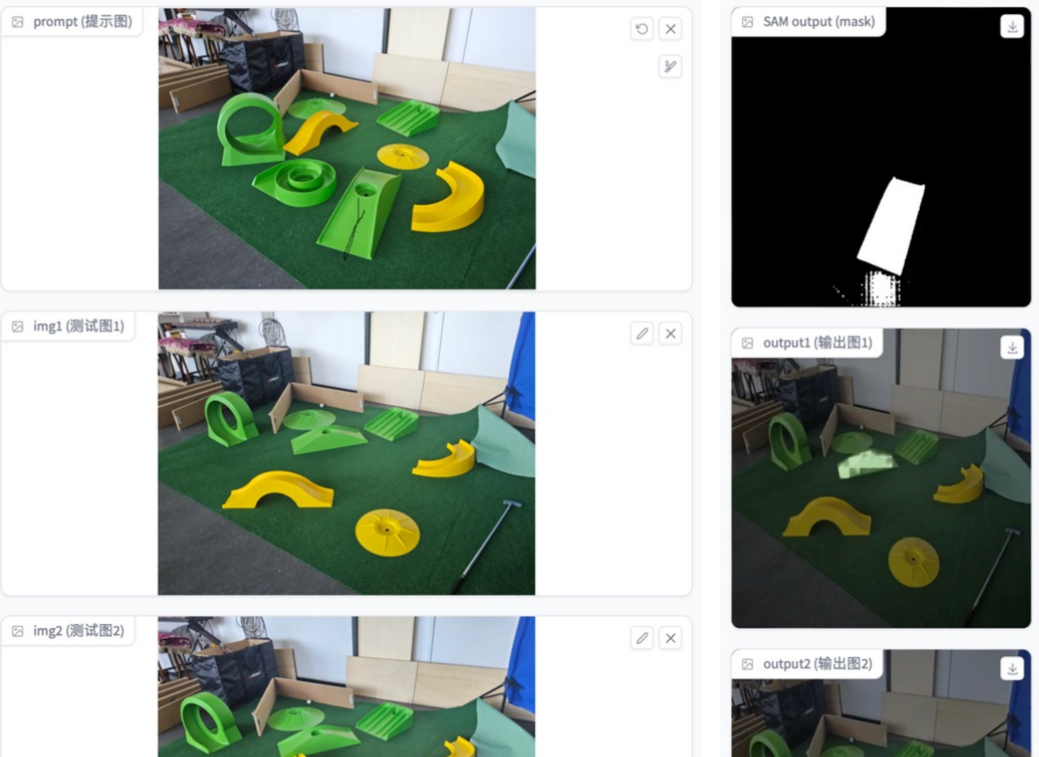
- Recording

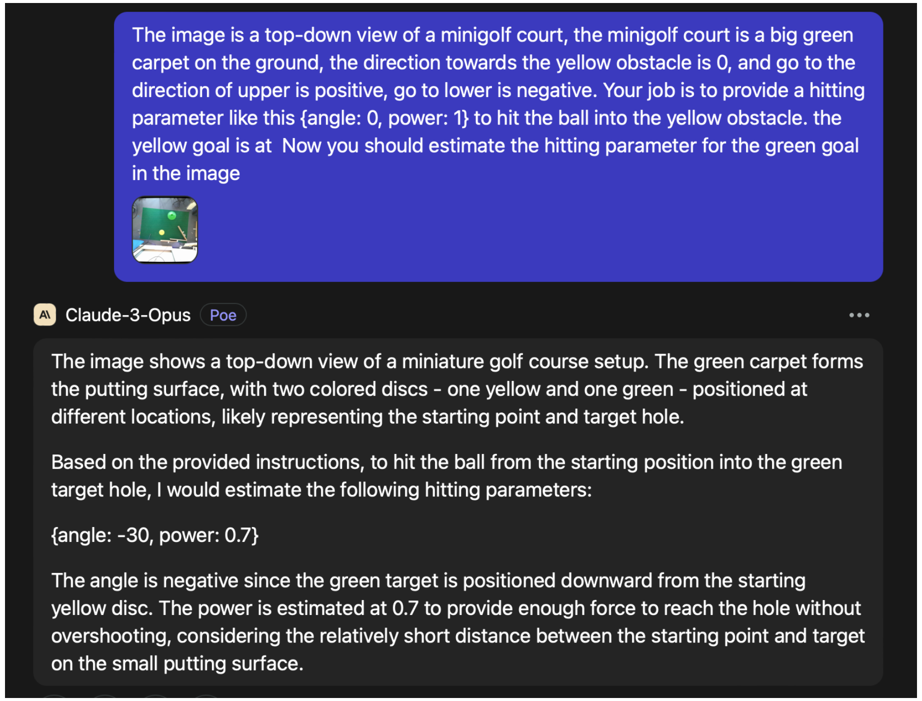
- Find trajectory



Current Experimental Results

- **Plan & Evaluate**
  - Analyse the court based on prompts
  - Generate planning trajectory based on image
  - Evaluate recorded trajectory





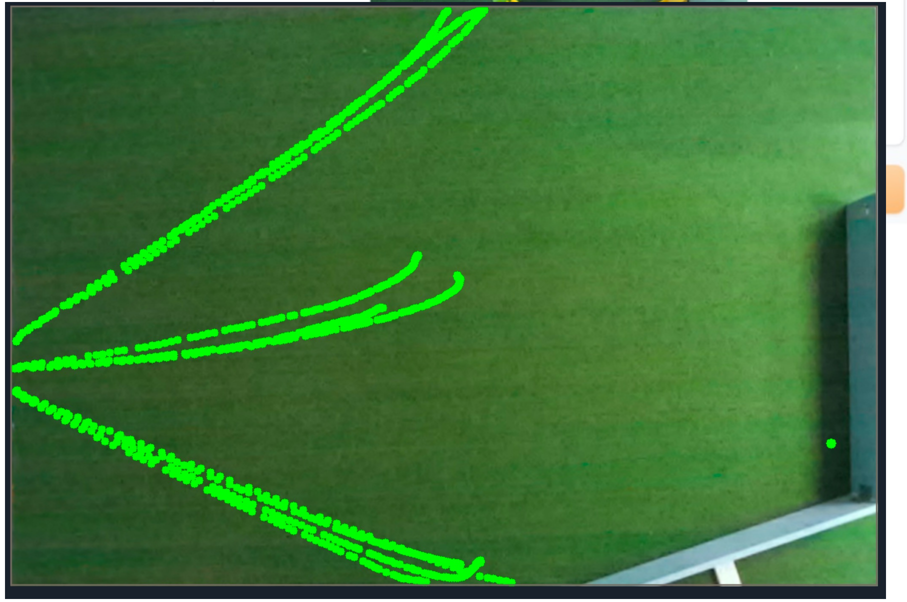
The image is a top-down view of a minigolf court, the minigolf court is a big green carpet on the ground, the direction towards the yellow obstacle is 0, and go to the direction of upper is positive, go to lower is negative. Your job is to provide a hitting parameter like this {angle: 0, power: 1} to hit the ball into the yellow obstacle. the yellow goal is at Now you should estimate the hitting parameter for the green goal in the image

The image shows a top-down view of a miniature golf course setup. The green carpet forms the putting surface, with two colored discs - one yellow and one green - positioned at different locations, likely representing the starting point and target hole.

Based on the provided instructions, to hit the ball from the starting position into the green target hole, I would estimate the following hitting parameters:

{angle: -30, power: 0.7}

The angle is negative since the green target is positioned downward from the starting yellow disc. The power is estimated at 0.7 to provide enough force to reach the hole without overshooting, considering the relatively short distance between the starting point and target on the small putting surface.



Current Progress in sum 0422

- **Perception**
  - Event Cam + RGB Cam to record
  - Blob + Yolo to generate high precision trajectories

- **Planning**
  - Baseline: GPT-4V + Prompt to generate hitting parameters and evaluate hitting results
  - Baseline: GPT-4V + Parameters + Prompt to generate hitting parameters and evaluate results