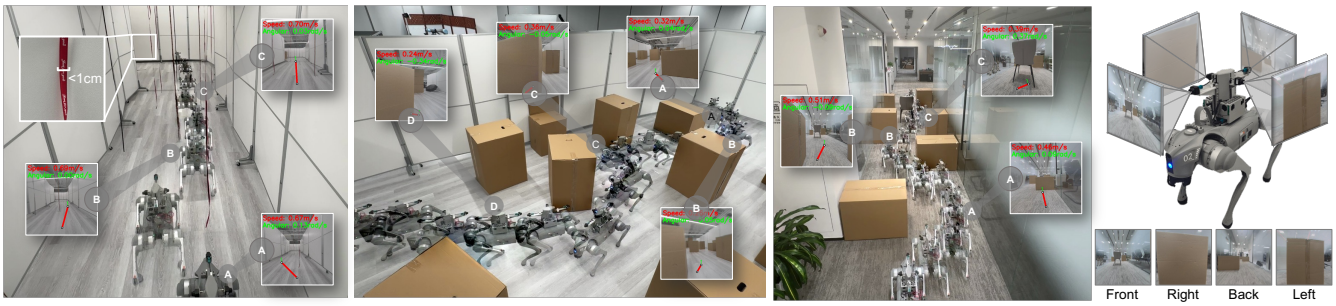
MM-Nav: Multi-View VLA Model for Robust Visual Navigation via Multi-Expert Learning



(a) Real-world demonstrations **(b)** Hardware setup

Fig. 1: **Real-world demonstration and hardware setup of MM-Nav**. Our method demonstrates strong navigation capability within challenging environments, including avoiding thin rope, squeezing between crowded and transparent objects.

***Abstract*— Visual navigation policy is widely regarded as a promising direction, as it mimics humans by using egocentric visual observations for navigation. However, optical information of visual observations is difficult to be explicitly modeled like LiDAR point clouds or depth map, which subsequently requires intelligent models and large-scale data. To this end, we propose to leverage the intelligence of Vision-Language-Action (VLA) model to learn diverse navigation capabilities from synthetic expert data in a teachers-student manner. Specifically, we implement the VLA model, MM-Nav, as a multi-view VLA (with 360° observation) based on pretrained large language models and visual foundation models. For large-scale navigation data, we collect expert data from three reinforcement learning (RL) experts trained with privileged depth information in three challenging tailor-made environments for different navigation capabilities: *'reaching'*, *'squeezing'*, and *'avoiding'*. We iteratively train our VLA model using data collected online from RL experts, where the training ratio is dynamically balanced based on performance on individual capabilities. Through extensive experiments in synthetic environments, we demonstrate that our model achieves strong generalization capability. Moreover, we find that our student VLA model outperforms the RL teachers, demonstrating the synergistic effect of integrating multiple capabilities. Extensive real-world experiments further confirm the effectiveness of our method.**

I. Introduction

Visual navigation has garnered considerable attention in the robotics field [1]–[9], as it requires robots to reach target locations based on visual inputs. Visual observations provide detailed and rich environmental information for navigation while remaining cost-effective. However, interpreting such informative visual data and planning appropriate navigation actions remain challenging [2], [3], demanding highly intelligent models and large-scale navigation datasets.

To this end, existing methods [1]–[3], [7], [9] address this challenge through learned policies that implicitly interpret visual inputs and predict subsequent actions. Advanced approaches have made initial progress by training on real-world passive navigation or touring videos to emulate general navigation capabilities. However, these methods remain constrained by limited observational perspectives (*e.g.*, only front-view) and the relatively spacious environments common in navigation datasets [10], [11], which hinders their applicability in more challenging scenarios.

In this context, visual navigation faces a conflict: real-world navigation data are mainly collected from single-camera setups and lack extremely challenging or hazardous scenarios (*e.g.*, avoid damaging robots); whereas synthetic navigation data allow customizable camera configurations and can generate data reflecting challenging navigation capabilities, yet they suffer from the sim-to-real gap [10], [12], [13] due to non-photorealistic imagery.

To resolve this conflict, we propose MM-Nav, a multi-view VLA model that captures 360° observations around the robot and learns navigation capabilities from multiple synthetic RL experts. This mechanism combines the best of both worlds: the generality of VLAs and the diverse navigation data available in synthetic environments. Specifically, we construct the VLA model with four horizontally distributed camera views to cover the surrounding environment and employ an action head to predict velocity commands for robot control. To maintain inference efficiency, we carefully design the tokenization of historical and current observations, enabling the VLA model to achieve an inference speed of approximately 7 Hz, which is comparable to existing visual navigation methods [2], [7].

To train MM-Nav for strong navigation capabilities, a large amount of navigation data is required. To this end, we construct synthetic environments and train individual reinforcement learning (RL) experts with three distinct navigation capabilities: reaching, squeezing, and avoiding. Leveraging data from these RL experts, we pre-train our model on their successful demonstrations. Then, we adopt an online teacher-student training strategy (in a DAgger manner [14]) to iteratively fine-tune our model with data

from different navigation capabilities. Here, we employ a capability-balanced data aggregation strategy that enables goal-oriented training, which we find leads to faster training convergence.

We conducted extensive experiments in both synthetic and real-world environments. The results show that our method exhibits strong navigation performance across environments with different capabilities, even outperforming specifically trained RL experts. This demonstrates that our approach achieves superior navigation performance by leveraging the synergy among diverse capabilities. Furthermore, extensive real-world experiments confirm our method's robust zero-shot sim-to-real transfer in challenging environments.

II. Related Works

Learning-based small navigation model. A considerable amount of learning-based navigation relies on models with a smaller parameter count. These models are often computationally efficient for specific tasks but face limitations in generalization and adaptability to complex scenarios. They can be broadly categorized into imitation learning and reinforcement learning. Imitation Learning (IL) acquires navigation capabilities by learning from expert demonstrations. This approach is relatively data-efficient as the expert data inherently contains effective strategies [2]–[4], [6], [10], [11], [15]–[18]. For instance, NavDP utilizes an A* planner with access to privileged information to generate high-quality expert trajectories. Reinforcement Learning (RL) offers a structured approach to end-to-end navigation by leveraging modern simulation technologies [19]–[23]. These methods are often limited to model size and meet strong difficulties in handling sim-to-real gap. In contrast, our method leverages VLA, utilizing VLA's inherent generalization capabilities to address the sim-to-real transfer challenge.

Learning-based large navigation model. Large model learning [24]–[27], particularly with Vision-Language-Action models [28], has significantly advanced navigation by leveraging powerful generalization and semantic understanding capabilities [7]–[9], [29], [30]. An approach employs off-the-shelf large models in a zero-shot manner [31]–[34], converting visual scenes into text descriptions for a Large Language Model to perform high-level planning [35]–[39]. However, this vision-to-text abstraction creates an information bottleneck, often limiting the agent to sparse landmarks and static environments. End-to-end VLA models like NaVid [8] and Uni-NaVid [7] tackle tasks ranging from object-search navigation and human-following to open-ended instruction following, all through a unified language-guided policy. In contrast to these models, which are constrained by a discrete action space (e.g., FORWARD, TURN LEFT) [7], [8], our VLA policy directly outputs continuous velocity commands, enabling the agile and instantaneous responses required for robust real-world deployment.

Visual-only navigation. Visual-only navigation relies solely on RGB images to guide movement, which is more economical and lightweight for practical deployment. A primary challenge for such methods is the implicit nature of 3D geometry in 2D images, as pure RGB input lacks direct depth information, making robust obstacle avoidance non-trivial [40]. Foundational models like GNM [1], ViNT [2], Scaling [5], and NoMaD [3] have demonstrated impressive generalization through imitation learning on large-scale datasets, but their reliance on single-camera views limits their spatial awareness. To mitigate safety risks, CARE [6] estimates RGB depth to add a collision avoidance layer on top of a pre-trained policy. Our work addresses these limitations using a four-camera surround-view system for 360° perception [4]. To tackle the challenge of learning from depthless RGB data, we leverage the comprehensive ability of VLA. We adopt the policy distillation paradigm, proven effective in works like X-Nav [41] and COMPASS [42]. But with a key innovation: we train multiple RL experts on specific capabilities using privileged depth information. This expertise is then distilled into a unified VLA policy that robustly navigates by inferring environmental geometry from surround-view RGB images only.

III. Method

A. Overview

Task Definition. We formulate the task as learning a velocity control policy π for an omnidirectional robot to navigate safely to a specific point goal in a cluttered and dynamic environment. At each time step t, given the position of a point goal $g_t = [g_t^x, g_t^y]$ and online captured multi-view RGB frames $O_t = \{I_{t-k+1:t}^N\}$ where $I_t^N \in \mathbb{R}^{3\times H\times W}$ (N denotes the number of cameras, including 4 camera views $\{I^{\text{front}}, I^{\text{right}}, I^{\text{back}}, I^{\text{left}}\}$), the policy $\pi(O_t, g_t) \mapsto a_t$ predicts an action $a_t = [v_t^{\text{x}}, v_t^{\text{y}}, v_t^{\text{yaw}}]$, representing omnidirectional velocity. We adopt velocity control because it allows the robot to execute arbitrary omnidirectional motions on a two-dimensional plane, thus enhancing its responsiveness in obstacle avoidance [20], [22], [42]. The objective is to ensure the velocities generated by the policy π are collision-free and reach the designated goal.

Pipeline Overview. Our pipeline comprises two steps: (1) training of multiple RL experts with different capabilities and initial VLA finetuning, (2) teachers-student online training iteration between RL experts and VLA. *First*, we use reinforcement learning to train three capability-specific experts in simulation, including reaching, squeezing, and avoiding. These capabilities are obtained specifically by training privileged RL experts in tailored scenarios. *Second*, different capability navigation trajectories from RL experts are generated and collected, which are then used to directly train a VLA model to initialize a general navigation policy mastering different capabilities. *Finally*, the basic VLA model (student) is deployed in the simulation environment, and we online collect the expert action of the RL teachers for further finetuning the VLA model. This mechanism is conducted iteratively until performance converges.

B. RL Experts for Different Navigation Capabilities

We construct three distinguishing environments for training RL experts to obtain different navigation capabilities:

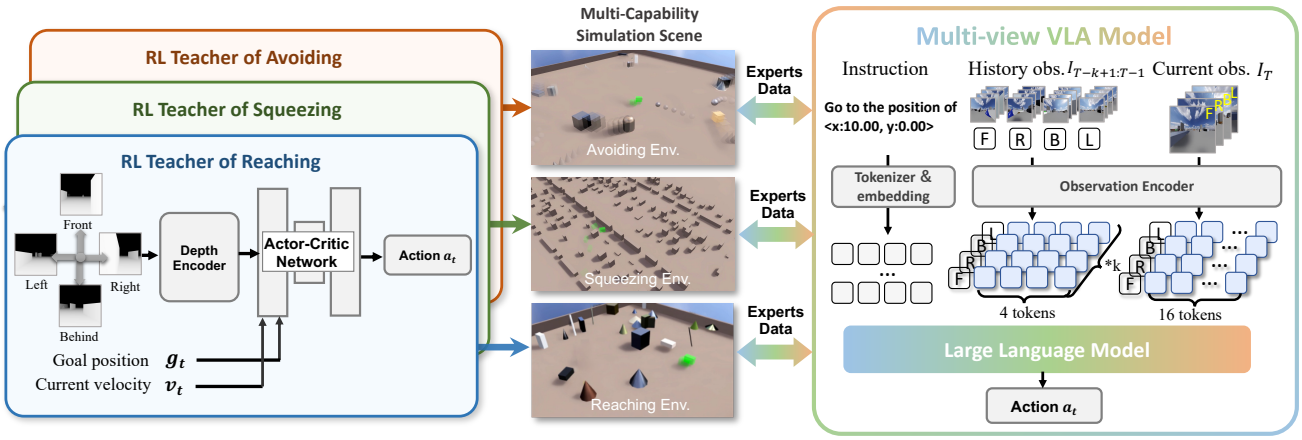


Fig. 2: **Pipeline of MM-Nav**. Our proposed teachers-student training pipeline. Independent RL teachers are trained in different scenes for multi-capability and distill knowledge to the VLA student. Further, the student is deployed in the capability-specific simulation scene to be iteratively fine-tuned.

(1) reaching: approach and reach a specific point goal while avoiding static obstacles, (2) squeezing: squeeze through cluttered and narrow gaps between obstacles and walls, and (3) avoiding: actively avoid crowded dynamic obstacles moving at random speed, as illustrated in Fig. 3.

Reaching environments. We construct the reaching scene which contains randomized static obstacles, generated in different shapes (cuboids, cones, cylinders, long poles, etc.), textures, and sizes. As in [20], the high diversity of the scene improves the expert's generalization ability, as well as the diversity of the collected data. Robots and their corresponding goals are initialized at random positions on the terrain, with initial distances of up to 30 m.

Squeezing environments. The static narrow-squeezing scene consists of densely placed pillars randomly distributed across the terrain and walls with narrow, randomized gaps. The expert must navigate safely and smoothly through these passages using visual feedback from four surround-view cameras. For example, observations from the left and right cameras help determine whether the robot can pass between adjacent obstacles.

Avoiding environments. The dynamic avoidance scene contains densely placed dynamic obstacles moving at velocities between 0.5 m/s and 1.5 m/s, requiring the RL expert to actively avoid collisions. These dynamic obstacles exhibit diverse sizes and geometries, such as spheres, cubes, rods, and cones. Similarly to the reaching scene, the distances between the robot and the target are sampled with values up to 10 m, and we make sure no collision happens when the robot is initialized.

Simulation setup. We construct our environments based on IsaacLab [43] for its modular design and strong reinforcement learning support. Following [10], in all three scenes, the robot is abstracted as a cuboid of size $[0.70m, 0.35m, 0.50m]$ in simulation to improve computational and rendering efficiency.

RL experts architecture. Although trained in three different scenes, all RL experts share a similar training methodology. We employ the Proximal Policy Optimization (PPO) [44] algorithm in simulation with 128 parallel robots. The observation at each time step t is defined as:

$$O_t^{RL} = [d_t^{\text{front}}, d_t^{\text{right}}, d_t^{\text{back}}, d_t^{\text{left}}, a_{t-1}, g_t] \tag{1}$$

where $d_t^{\text{front}}, d_t^{\text{right}}, d_t^{\text{back}}, d_t^{\text{left}}$ are the depth images from four cameras mounted on the respective sides of the robot, a_{t-1} is the last applied action and g_t is the relative position of the goal. As demonstrated in Fig. 2. Each depth image from the four views is encoded into a feature vector by ResNet-18 [45], which is concatenated with the last action a_{t-1}, the goal position g_t, and a history token from the last hidden layer of a three-layer MLP. The concatenated features are fed to the MLP to predict the velocity action $a_t = [v_{\text{x}}^t, v_{\text{y}}^t, v_{\text{yaw}}^t]$. The last hidden state of the MLP is stored and passed to the next time step as the history token. The output a_t is then multiplied and clipped within the maximum velocity limits $v_{\max} = [1.5m/s, 1.0m/s, \pi/4.0\ rad/s]$:

$$v_t = \max\{\min\{a_t, 1.0\}, -1.0\} \times v_{\max} \tag{2}$$

The resulting velocity v_t is then applied in the simulation. During training, Gaussian noise is added to proprioceptive velocity observations to improve robustness.

RL training rewards and termination conditions. The reward encourages reasonable, goal-directed, and collision-free behavior:

$$r_{\text{Cap.}} = \alpha_{\text{Cap.}} r_{\text{goal}} + \beta_{\text{Cap.}} r_{\text{step}} + \gamma_{\text{Cap.}} r_{\text{reg}} + \delta_{\text{Cap.}} r_{\text{col}} \tag{3}$$

Where Cap. denotes the reaching, squeezing, and avoiding experts; r_{goal} rewards progress toward the goal; r_{step} penalizes each step to encourage efficiency; r_{reg} regularizes actions to discourage undesirable behaviors (e.g., retreating, circling); and r_{col} penalizes collisions with obstacles or other robots. To guide and specialize the three RL experts, their reward coefficients $\alpha_{\text{Cap.}}, \beta_{\text{Cap.}}, \gamma_{\text{Cap.}}, \delta_{\text{Cap.}}$ are designed to differ. Specifically, we set $\beta_{\text{Cap.}} = -0.05$ and $\delta_{\text{Cap.}} = -15$ for all experts. For the regularization terms, we assign $\gamma_{\text{reaching}} = 0.05$, $\gamma_{\text{squeezing}} = 0.02$, $\gamma_{\text{avoiding}} = 0$, thereby enforcing

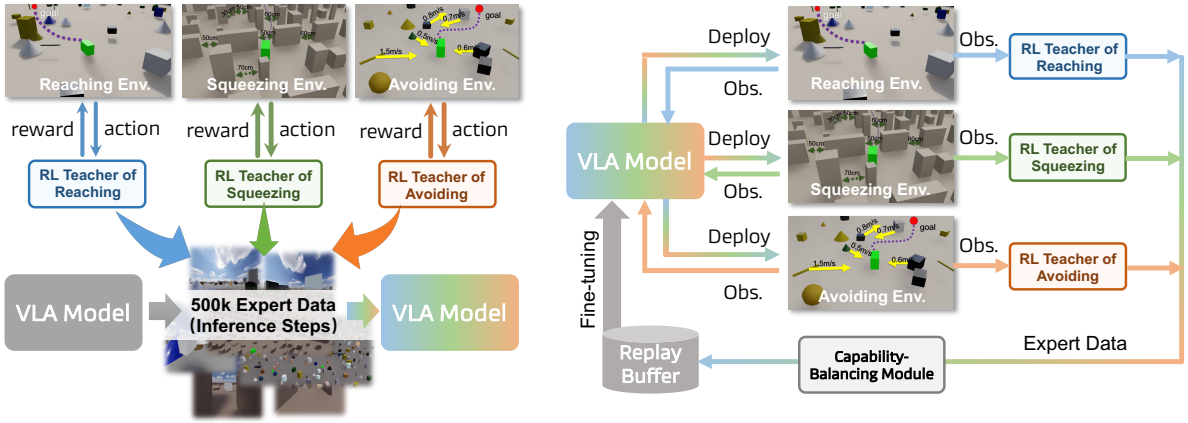


Fig. 3: **Training Strategy of MM-Nav.** Stage 1: We first train RL experts in different environments and collect successful trajectories for VLA fine-tuning. Stage 2: We then collect RL expert data online based on VLA observations and dynamically balance the training data ratio.

stronger regulation in the relatively easier reaching scene while applying weaker or no regularization in the more challenging squeezing and avoiding scenes. Finally, for the goal-reaching reward, we set $\alpha_{\text{reaching}} = \alpha_{\text{avoiding}} = 1.2$ and $\alpha_{\text{squeezing}} = 1.5$, ensuring that the squeezing expert remains decisive when traversing narrow gaps.

C. Student VLA Model

Visual Observation Encoding. To support a 360° observation, we extend a video-based VLA model [8] to support multi-view observations. Given a sliding window (length $k = 8$) of navigation history of four-view RGB images $O_T = I^N_{T-k+1:T}$, which include a front-view, right-view, back-view, and left-view of robots. We then encode all frames into a sequence of visual tokens $E^{\text{visual}} \in \mathbb{R}^{P\times C}$ ($P = 576$ is the patch size and C is the token dimensions) using a visual foundation model (implemented with SigLIP [46]) and a cross-modal projector (a two-layer MLP [25]). Similar to previous methods [7], [8], we conduct grid-based average pooling of the visual tokens, and use fine-grained visual tokens $E^{\text{fine}} \in \mathbb{R}^{16\times C}$ for latest observation and coarse-grained visual tokens $E^{\text{coarse}} \in \mathbb{R}^{4\times C}$ for navigation history. Finally, we can organize the visual token as:

$$E_{\text{visual}} = \{E^{\text{F/R/B/L\_coarse}}_{T-k+1}, ..., E^{\text{F/R/B/L\_coarse}}_{T-1}, E^{\text{F/R/B/L\_fine}}_{T}\}, \quad (4)$$

where all four camera views (Front, Right, Back, Left) are used at each timestep. The use of a sliding window for token selection helps maintain a reasonable visual token sequence length (192 tokens), thereby ensuring consistent inference speed.

Action Forwarding. With organized visual tokens, we format the relative point goal g_t into a textual prompt and encode the prompt as language tokens E_{text}. Here, using a textual prompt to represent the point goal allows us to co-tune the navigation data with open-world QA data, which has been widely proven to mitigate the sim-to-real gap [7], [9]. We then feed the concatenation of the visual tokens E_{visual} and language tokens E_{text} into a large language model (implemented using Qwen2 [47]) to obtain the predicted action token E_{action}. Finally, the action token is processed by an action head (implemented with two-layer MLP) to predict the velocity ($\mathbf{v}_T = [v_x, v_y, v_{yaw}]$) of the robot.

Here, we apply mean-square-error loss for action prediction $L_{\text{action}} = \text{MSE}(\mathbf{v}^{\text{pred}}, \mathbf{v}^{\text{gt}})$, and retain the cross-entropy loss L_{QA} for open-world question-answering data, similar to [9]. The overall loss is defined as: $L = \beta L_{\text{action}} + L_{\text{QA}}$, where the β is set to 5 to balance the magnitude of the two loss terms. Notably, through a tiered architecture that incorporates visual tokens, our method achieves an inference speed of 7 Hz using a 7B-parameter LLM, while maintaining strong performance in challenging scenarios. This framework could be further improved by integrating acceleration techniques such as quantization [48]–[50].

D. RL Experts–VLA iteration

To distill expertise from multiple RL experts into the student VLA model, we design a two-stage training process.

Initial expert data collection and VLA finetuning. To endow the VLA model with initial navigation capability, we first collect a diverse set of trajectories generated by RL experts in simulation. For each expert, we run 64 parallel robots in their respective simulation environments, recording the observations, goals, and corresponding expert actions. Only trajectories that successfully reach the goal are retained, ensuring that erroneous actions from imperfect RL experts are excluded. All retained trajectories are aggregated into a dataset containing 500k steps, each represented as $[O_i, g_i, a_i]$, where a_i is the ground truth velocity. During collection, we randomized the height of the camera within $[0.4m, 0.5m]$ and the field of view (FOV) of each camera within $[100°, 140°]$. As shown in Fig. 6, after being trained on the large-scale simulation dataset, the initial VLA model performs well in the reaching scene but poorly in others, and in all cases underperforms the corresponding RL expert.

Teachers-student online training iteration. We collect online expert data iteratively using a data aggregation manner.

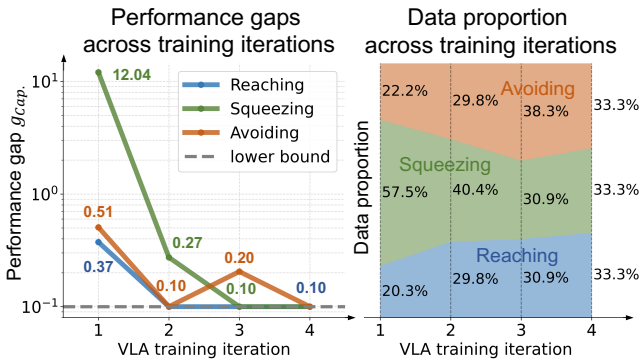


Fig. 4: **Plot of online training iteration.** (Left): the performance gaps $g_{Cap.}$ between the VLA model and the RL experts. (Right): the different proportions of our online collected data. After the fourth iteration, VLA outperforms all experts (in WTT), resulting in equal data ratio.

The preliminarily trained VLA model is deployed in the three simulation scenes to collect actions from the corresponding RL teachers. However, the varying complexities of the three capabilities result in uneven performance of VLA across the three scenes, like performing well in the reaching task but underperforming in the more challenging squeezing and avoiding tasks. This motivates the need for a more balanced use of training data from different scenes. To address this issue, we propose a capability-balanced data aggregation method. It leverages weighted travel time (WTT) W (the average time of successful episodes divided by success rate) to measure the performance gap between the VLA model and the RL experts. The gap is defined as

$$g_{\text{Cap.}} = \max\left\{0, \frac{W^{\text{VLA}}_{\text{Cap.}} - W^{\text{RL}}_{\text{Cap.}}}{W^{\text{RL}}_{\text{Cap.}}}\right\} + \epsilon \tag{5}$$

where $W^{\text{RL}}_{\text{Cap.}}$ represents the WTT of the reaching, squeezing, and avoiding experts, respectively, and $W^{\text{VLA}}_{\text{Cap.}}$ represents the WTT of the current VLA model in each scene. We set ϵ to 0.1 to maintain a set of data when the VLA model already outperforms the RL expert. Based on this gap, we compute the data proportion $p_{\text{Cap.}}$ for each capability as

$$p_{\text{Cap.}} = \frac{g^{\alpha}_{\text{Cap.}}}{\sum_{\text{Cap.}} g^{\alpha}_{\text{Cap.}}} \tag{6}$$

with $\alpha = 0.3$ to smooth the distribution. Intuitively, larger performance gaps lead to higher proportions of the corresponding expert data being aggregated, as illustrated in Fig. 4. Finally, using the computed proportions $p_{\text{Cap.}}$, we aggregate the expert datasets in a capability-balanced manner and fine-tune the VLA model. This enables the transfer of additional collision-avoidance and navigation skills. After fine-tuning, the model is re-evaluated, and the process is repeated iteratively with updated $p_{\text{Cap.}}$ values until no further improvement is observed.

E. Implementation Details

RL Training strategy. Each RL expert is trained in Isaac Lab [43] on an NVIDIA RTX 4090 GPU for 8-12 hours, using $N = 128$ parallel environments. The policy adopts a history-aware actor-critic architecture. Both the actor and critic are implemented as three-layer MLPs with hidden dimensions of [512, 256, 128] and use the ELU activation function [51]. Regarding the sensory input, the depth values from the cameras are clipped to [0.01m, 4.0m] to filter out noise. To encourage initial exploration, the action distribution is initialized with a noise standard deviation of 0.2.

VLA training strategy. Our initial VLA model is fine-tuned on 8 NVIDIA H100 GPUs for about 5 hours, totaling 40 GPU hours. We leverage the pre-trained weights of both visual encoders (SigLIP [46]) and LLM (Qwen2-7B [47]). The initial training contains 500k steps from three RL experts and 100k visual question answering (VQA) data. For VQA data, we use a subset of the dataset in [52] and the frames are sampled at 1 FPS, following [8]. Each iteration of teacher-student training costs about 2 hours and contains 200k steps of online collected expert data and 40k VQA data.

Deployment strategy. We deploy our method on the Unitree GO2 robot. The VLA model is executed on a server equipped with an NVIDIA RTX 5090 GPU, while the onboard computer on the robot continuously sends requests and receives velocity commands. We use four fisheye cameras mounted on the front, right, back, and left sides of the robot to obtain the four-view real-time images, which are then undistorted into normal perspective images and fed into the VLA model. Given the velocity v_t inferred by the VLA model, the low-level controller follows the velocity until the next response. The average response fps is about 7 Hz, which is sufficient for our navigation task.

IV. Experiments

A. Experiment Setup

Simulation environment setup. We evaluate our method in IsaacLab [43]. Unlike the training environment, three tailored and fixed scenes are designed to specifically assess the three target capabilities. To compare with baseline methods, we further design a combined test scene featuring static obstacles, dynamic obstacles, and narrow gaps, all of which fall outside the VLA model's training distribution. Additionally, we construct a `Mixed` environment that includes squeezing, reaching, and avoiding scenarios. All objects within Mixed scene are rendered with materials visually distinct from those used in training. The difficulty of the squeezing segment in the Mixed scene is also intentionally reduced to ensure the test remains balanced. Each simulation episode is terminated under one of three conditions: the robot successfully reaches the goal, it collides with an obstacle, or the episode times out. The time limit is set to 90 seconds for the three capability-specific scenes and extended to 120 seconds for the more complex Mixed scene.

Real-world environment setup. We constructed four diverse real-world scenes, including *Narrow Zigzag Corridor*, *Thin Obstacle Avoidance*, *Dynamic Environment*, *Cluttered Static Environment*, to evaluate the sim-to-real transfer and generalization capabilities of our method. Each scene is designed to

TABLE I: Quantitative comparison and ablation study on IsaacLab simulator in three capability-specific scenes and a mixed scene. Each method was evaluated over 100 episodes per scene.

Methods	Reaching			Squeezing			Avoiding			Mixed		
	SR(%)↑	CR(%)↓	WTT(s)↓	SR(%)↑	CR(%)↓	WTT(s)↓	SR(%)↑	CR(%)↓	WTT(s)↓	SR(%)↑	CR(%)↓	WTT(s)↓
iPlanner [53]	19	81	93.4	2	94	881.25	15	85	73.4	4	94	736.9
ViPlanner [54]	43	57	42.2	4	96	452.8	22	78	36.36	18	82	215.4
NavDP [10]	69	31	**27.3**	18	82	115.4	27	73	30.0	23	77	178.6
Ours	**80**	**20**	31.0	**71**	**19**	**42.2**	**68**	**32**	**20.9**	**47**	**26**	**127.5**

test specific skills learned in simulation, as well as robustness to novel objects and materials not present in the training data.

Metrics and baselines. We evaluate the methods based on three metrics: (1) Success rate (SR), (2) Collision rate (CR), (3) Weighted travel time (WTT, the average time of successful episodes divided by success rate). For each method and scene, 100 episodes are rolled out to compute the three metrics.

B. Quantitative Results

The simulation benchmark results are summarized in Table I. We configured each baseline following its original setup. Overall, our proposed method achieves the highest SR, the lowest CR, and the shortest WTT across almost all test scenes, indicating that it not only ensures collision-free motion but also plans more efficient paths. Here, NavDP [10] performs competitively in the Reaching scene but is fundamentally constrained by its single, front-facing camera, which limits its FOV. This limitation leads to characteristic failures in more complex scenarios. In the Avoiding scene, the policy is vulnerable to dynamic obstacles approaching from the sides or back. In cluttered static environments, it struggles with obstacles that are hard to perceive from a single forward perspective; for instance, it often fails to detect thin objects like poles and ropes. ViPlanner [54] and iPlanner [53] exhibit the same FOV constraint and, in our tests, their control policies are less agile and responsive than NavDP's, resulting in overall lower performance, especially in the Reaching scenes. In contrast, our 360° surround-view and vision-only perception, paired with continuous velocity control, mitigates these failure modes and enables robust navigation in dense, dynamic, and constrained environments.

C. Qualitative Results

In four real-world scenes, MM-Nav demonstrates robust sim-to-real transfer ability, as shown in Fig. 1. In the *Narrow Zigzag Corridor* and *Cluttered Static Environment* scene, the robot makes precise adjustments to pass through these boxes without collision, exhibiting impressive agility. In the *Thin Obstacle Avoidance* scene, our robot successfully avoids these thin fabric strips, which are difficult to detect for LiDAR sensors [21] (a task at which the robot's native LiDAR-based avoidance system completely fails). In the *Office Corridor Generalization*, the model successfully navigates around unseen objects, like a chair, and handles challenging materials such as partially transparent glass walls. This confirms its ability to generalize a robust understanding of traversability from simulation to the real world.

D. Ablation Study

Performance gains across online training iterations. The performance of the initial VLA model and its variants after the first four training iterations was assessed in simulation; the results are shown in Fig. 6. After the initial behavior-cloning training, the VLA model exhibits clear performance gaps across all three capabilities, particularly in the squeezing capability. Through several online training iterations, the model gradually improves its performance. In the first iteration, the capability-balanced data aggregation method places emphasis on squeezing, leading to substantial improvements. Consequently, the performance of the VLA model after the first iteration becomes comparable to, or even surpasses, that of the RL experts. In the second and third iterations, the reaching capability is further enhanced, even though the VLA model had already surpassed the performance of the corresponding RL expert. This may be attributed to the improved integration of squeezing and avoiding capabilities, which facilitates a stronger understanding of the relatively simpler reaching task. After the fourth iteration, the performance across the three tasks converges, and no additional iterations are conducted.

Capability-balanced data aggregation method. Comparison experiments are conducted to evaluate the advantage of our capability-balanced data aggregation method. Based on the same initial VLA model, we train one set of iterations using our balanced data and another one with unbalanced data (Fig. 6). In both training setups, the total number of samples per iteration is kept constant. It is shown that the capability-balanced method is able to complement the lacking capability in time, thus achieving faster and more stable training. Although the unbalanced way achieves better performance in reaching the scene, it fails to efficiently learn from squeezing data and avoiding data. In a word, the capability-balanced method helps fuse the data from different RL experts together and prevents the VLA model from overlooking specific capabilities.

Combination strategy of the experts. It is further investigated how combining three capability-specific RL teachers contributes to developing a more comprehensive and powerful VLA student. We trained three VLA models using the same total amount of data, but each derived from a single RL expert. Additionally, we train a single RL expert in the mixed scene requiring all three capabilities, comparing it with the three independent RL experts used in our method. As shown in Tab. II, the RL expert trained in the mixed scene possesses all three capabilities but, due to capacity limits, cannot match the specialized experts in any single

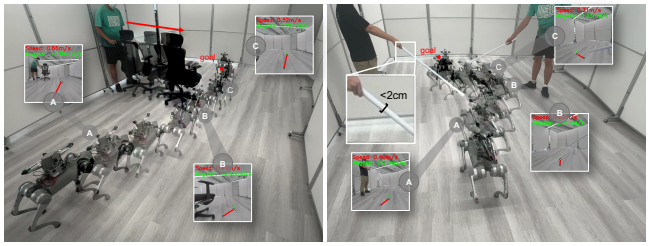


Fig. 5: **Real-world Experiments.** (Left): As the robot advances, an operator suddenly pushes a wheelchair into its path (Right): Two human-held thin rods are crossed across the corridor.

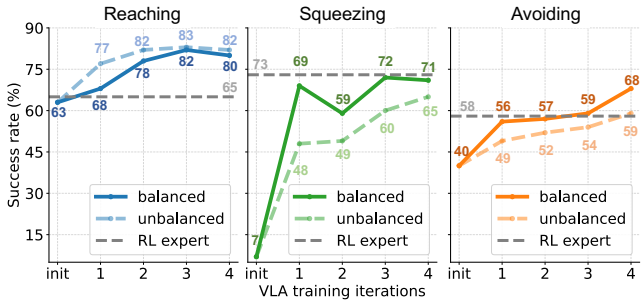


Fig. 6: Success Rate of the VLA model after each iteration.

capability, resulting in compromised performance across all scenes. The models trained with a single type of data achieve strong performance in their corresponding scenarios but generalize poorly to tasks that require unseen capabilities. In contrast, the VLA model trained with mixed data demonstrates significantly stronger cross-capability performance, outperforming all models trained with a single data source. The superior performance of the mixed-trained VLA model can be attributed to the complementary and mutually reinforcing nature of different capabilities. While models trained on a single expert specialize in their corresponding domain, they lack the broader contextual knowledge that other capabilities provide. For example, training solely with a squeezing expert may bias the model toward exploiting static obstacle properties, while limiting its ability to recognize and actively avoid dynamic obstacles. In contrast, mixed training exposes the student to diverse skill dimensions, enabling them to build shared representations that capture abstract structures spanning multiple domains.

V. Conclusion

We have presented MM-Nav, a multi-view VLA model that acquires robust visual navigation skills from a collective of specialized RL experts. Our training process consists of two key stages: first, an initial VLA finetuning phase where a student policy learns from a large offline dataset collected from the RL teachers; second, an online teachers-student training iteration where the student is deployed in simulation to receive on-the-fly, capability-balanced supervision for further refinement. MM-Nav demonstrates strong sim-to-real transfer and ultimately outperforms its RL experts, proving the synergistic benefit of learning multiple capabilities. Our work provides a scalable and effective blueprint for training a new generation of general-purpose visual navigation agents. Future work may include investigating the cross-embodiment potential of our training strategy and further advancing visual-only navigation through VLA and other methods.

TABLE II: **Comparison of Individual Navigation Capability.** We report the success rates (100 episodes) of VLA models and RL experts trained on either single or mixed datasets. The best overall result is highlighted in **bold**, and the best result within each category is indicated with underline.

Methods	Reaching	Squeezing	Avoiding
Reaching-RL	65	2	33
Squeezing-RL	30	73	19
Avoiding-RL	59	0	58
Mixed-RL	45	58	42
Reaching-VLA	70	4	35
Squeezing-VLA	44	64	28
Avoiding-VLA	63	0	62
Mixed-VLA	**80**	**71**	**68**

References

[1] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7226–7233. 1, 2

[2] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A foundation model for visual navigation," *arXiv preprint arXiv:2306.14846*, 2023. 1, 2

[3] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "Nomad: Goal masked diffusion policies for navigation and exploration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 63–70. 1, 2

[4] N. Hirose, F. Xia, R. Mart'ın-Mart'ın, A. Sadeghian, and S. Savarese, "Deep visual mpc-policy learning for navigation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3184–3191, 2019. 1, 2

[5] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678. 1, 2

[6] J. Kim, J. Sim, W. Kim, K. Sycara, and C. Nam, "Enhancing safety of foundation models for visual navigation through collision avoidance via repulsive estimation," *arXiv preprint arXiv:2506.03834*, 2025. 1, 2

[7] J. Zhang, K. Wang, S. Wang, M. Li, H. Liu, S. Wei, Z. Wang, Z. Zhang, and H. Wang, "Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks," *Robotics: Science and Systems*, 2025. 1, 2, 4

[8] J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang, "Navid: Video-based vlm plans the next step for vision-and-language navigation," *Robotics: Science and Systems*, 2024. 1, 2, 4, 5

[9] S. Wang, J. Zhang, M. Li, J. Liu, A. Li, K. Wu, F. Zhong, J. Yu, Z. Zhang, and H. Wang, "Trackvla: Embodied visual tracking in the wild," *arXiv e-prints*, pp. arXiv–2505, 2025. 1, 2, 4

[10] W. Cai, J. Peng, Y. Yang, Y. Zhang, M. Wei, H. Wang, Y. Chen, T. Wang, and J. Pang, "Navdp: Learning sim-to-real navigation diffusion policy with privileged information guidance," *arXiv preprint arXiv:2505.08712*, 2025. 1, 2, 3, 6

[11] A. Eftekhar, L. Weihs, R. Hendrix, E. Caglar, J. Salvador, A. Herrasti, W. Han, E. VanderBil, A. Kembhavi, A. Farhadi *et al.*, "The one ring: a robotic indoor navigation generalist," *arXiv preprint arXiv:2412.14401*, 2024. 1, 2

[12] H. Wang, J. Chen, W. Huang, Q. Ben, T. Wang, B. Mi, T. Huang, S. Zhao, Y. Chen, S. Yang *et al.*, "Grutopia: Dream general robots in a city at scale," *arXiv preprint arXiv:2407.10943*, 2024. 1

[13] X. Meng, X. Yang, S. Jung, F. Ramos, S. S. Jujjavarapu, S. Paul, and D. Fox, "Aim my robot: Precision local navigation to any object," *IEEE Robotics and Automation Letters*, 2025. 1

[14] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635. 1

[15] W. Xiao, H. Xue, T. Tao, D. Kalaria, J. M. Dolan, and G. Shi, "Anycar to anywhere: Learning universal dynamics model for agile and adaptive mobility," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 8819–8825. 2

[16] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "Sacson: Scalable autonomous control for social navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 49–56, 2023. 2

[17] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti *et al.*, "Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 238–16 250. 2

[18] W. Cai, S. Huang, G. Cheng, Y. Long, P. Gao, C. Sun, and H. Dong, "Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5228–5234. 2

[19] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," *arXiv preprint arXiv:2401.17583*, 2024. 2

[20] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "Navrl: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, 2025. 2, 3

[21] Z. Wang, T. Ma, Y. Jia, X. Yang, J. Zhou, W. Ouyang, Q. Zhang, and J. Liang, "Omni-perception: Omnidirectional collision avoidance for legged locomotion in dynamic environments," *arXiv preprint arXiv:2505.19214*, 2025. 2, 6

[22] J. Yao, X. Zhang, Y. Xia, Z. Wang, A. K. Roy-Chowdhury, and J. Li, "Towards generalizable safety in crowd navigation via conformal uncertainty handling," *arXiv preprint arXiv:2508.05634*, 2025. 2

[23] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs, "Poliformer: Scaling on-policy rl with transformers results in masterful navigators," *arXiv preprint arXiv:2406.20083*, 2024. 2

[24] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez *et al.*, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, vol. 2, no. 3, p. 6, 2023. 2

[25] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *NeurIPS*, 2023. 2, 4

[26] D. Zhu, J. Chen, K. Haydarov, X. Shen, W. Zhang, and M. Elhoseiny, "Chatgpt asks, blip-2 answers: Automatic questioning towards enriched visual descriptions," *arXiv preprint arXiv:2303.06594*, 2023. 2

[27] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024. 2

[28] W. Zhang, H. Liu, Z. Qi, Y. Wang, X. Yu, J. Zhang, R. Dong, J. He, H. Wang, Z. Zhang *et al.*, "Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge," *arXiv preprint arXiv:2507.04447*, 2025. 2

[29] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 42–48. 2

[30] A.-C. Cheng, Y. Ji, Z. Yang, X. Zou, J. Kautz, E. Biyik, H. Yin, S. Liu, and X. Wang, "Navila: Legged robot vision-language-action model for navigation," in *RSS*, 2025. 2

[31] Y. Long, W. Cai, H. Wang, G. Zhan, and H. Dong, "Instructnav: Zero-shot system for generic instruction navigation in unexplored environment," *arXiv preprint arXiv:2406.04882*, 2024. 2

[32] Y. Long, X. Li, W. Cai, and H. Dong, "Discuss before moving: Visual language navigation via multi-expert discussions," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 17 380–17 387. 2

[33] D. Shah, B. Osinski, B. Ichter, and S. Levine, "Robotic navigation with large pre-trained models of language," *Vision, and Action*, 2022. 2

[34] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504. 2

[35] G. Zhou, Y. Hong, and Q. Wu, "Navgpt: Explicit reasoning in vision-and-language navigation with large language models," *arXiv preprint arXiv:2305.16986*, 2023. 2

[36] G. Zhou, Y. Hong, Z. Wang, X. E. Wang, and Q. Wu, "Navgpt-2: Unleashing navigational reasoning capability for large vision-language models," in *European Conference on Computer Vision*. Springer, 2025, pp. 260–278. 2

[37] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009. 2

[38] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," *arXiv preprint arXiv:2210.05714*, 2022. 2

[39] Z. Qi, W. Zhang, Y. Ding, R. Dong, X. Yu, J. Li, L. Xu, B. Li, X. He, G. Fan *et al.*, "Sofar: Language-grounded orientation bridges spatial reasoning and object manipulation," *arXiv preprint arXiv:2502.13143*, 2025. 2

[40] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 10 371–10 381. 2

[41] H. Wang, A. H. Tan, A. Fung, and G. Nejat, "X-nav: Learning end-to-end cross-embodiment navigation for mobile robots," *arXiv preprint arXiv:2507.14731*, 2025. 2

[42] W. Liu, H. Zhao, C. Li, J. Biswas, S. Pouya, and Y. Chang, "Compass: Cross-embodiment mobility policy via residual rl and skill synthesis," *arXiv preprint arXiv:2502.16372*, 2025. 2

[43] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023. 3, 5

[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. 3

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 3

[46] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 975–11 986. 4, 5

[47] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024. 4, 5

[48] J. Lang, Z. Guo, and S. Huang, "A comprehensive study on quantization techniques for large language models," in *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*. IEEE, 2024, pp. 224–231. 4

[49] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022. 4

[50] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024. 4

[51] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, vol. 4, no. 5, p. 11, 2015. 5

[52] X. Shen, Y. Xiong, C. Zhao, L. Wu, J. Chen, C. Zhu, Z. Liu, F. Xiao, B. Varadarajan, F. Bordes *et al.*, "Longvu: Spatiotemporal adaptive compression for long video-language understanding," *arXiv preprint arXiv:2410.17434*, 2024. 5

[53] F. Yang, C. Wang, C. Cadena, and M. Hutter, "iplanner: Imperative path planning," *arXiv preprint arXiv:2302.11434*, 2023. 6

[54] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5243–5249. 6