ComposableNav: Instruction-Following Navigation in Dynamic Environments via Composable Diffusion

Zichao Hu[1]*, Chen Tang[1], Michael J. Munje[1], Yifeng Zhu[1], Alex Liu[1], Shuijing Liu[1]
Garrett Warnell[1,2], Peter Stone[1,3], Joydeep Biswas[1]

[1]Department of Computer Science, The University of Texas at Austin
[2]Army Research Laboratory [3]Sony AI

Abstract: This paper considers the problem of enabling robots to navigate dynamic environments while following instructions. The challenge lies in the combinatorial nature of instruction specifications: each instruction can include multiple specifications, and the number of possible specification combinations grows exponentially as the robot's skill set expands. For example, "overtake the pedestrian while staying on the right side of the road" consists of two specifications: *"overtake the pedestrian"* and *"walk on the right side of the road."* To tackle this challenge, we propose ComposableNav, based on the intuition that following an instruction involves independently satisfying its constituent specifications, each corresponding to a distinct motion primitive. Using diffusion models, ComposableNav learns each primitive separately, then composes them in parallel at deployment time to satisfy novel combinations of specifications unseen in training. Additionally, to avoid the onerous need for demonstrations of individual motion primitives, we propose a two-stage training procedure: (1) supervised pre-training to learn a base diffusion model for dynamic navigation, and (2) reinforcement learning fine-tuning that molds the base model into different motion primitives. Through simulation and real-world experiments, we show that ComposableNav enables robots to follow instructions by generating trajectories that satisfy diverse and unseen combinations of specifications, significantly outperforming both non-compositional VLM-based policies and costmap composing baselines. [2]

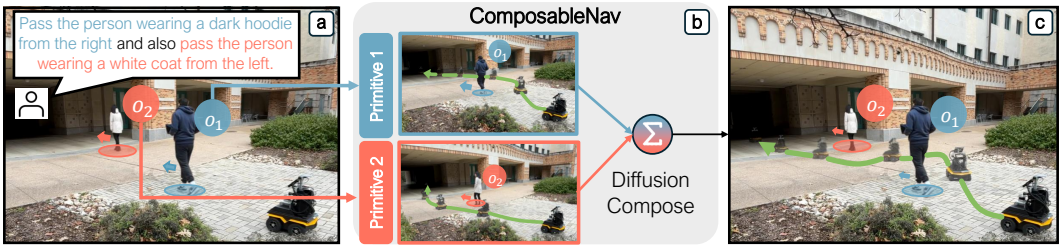


Figure 1: **Instruction-Following Navigation in Dynamic Environments.** Given an instruction that specifies how a robot should interact with entities in the scene (a), ComposableNav leverages the composability of diffusion models (b) to compose motion primitives to generate instruction-following trajectories (c).

1 Introduction

Developing robots that can effectively navigate by following instructions is an active research area aimed at enabling robots to operate within human-inhabited environments. Existing work has predominantly tackled the instruction-following navigation problem in *static* environments [1, 2, 3],

*Correspondence to: zichao@utexas.edu
[2]Project page: https://amrl.cs.utexas.edu/ComposableNav

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

where instructions specify the navigation goals. In contrast, we focus on instruction-following navigation in *dynamic* environments. Specifically, we consider the under-explored settings where the instructions describe specific robot interaction behaviors (*e.g.,* "yield to a pedestrian") with respect to the other dynamic obstacles or agents. Addressing this problem requires developing methods capable of grounding high-level instructions into fine-grained, low-level actions that account for the dynamic behaviors of other agents. Solving this problem would allow end users (human or AI agents) to customize robotic behaviors beyond their default settings, in ways that align with user preferences and nuanced social interactions.

A crucial challenge present in instruction-following navigation is that a single instruction may contain multiple specifications for the robot to follow, e.g., *"overtake the pedestrian while staying on the right side of the road"* consists of two specifications: *"overtake the pedestrian"* and *"walk on the right side of the road."*. Following an instruction amounts to simultaneously satisfying each of its constituent specifications. As the robot's capabilities and environment complexity increase, the space of possible combinations of such specifications grows exponentially. This combinatorial expansion makes popular learning-based methods, such as imitation learning [4] or reinforcement learning [5, 6], impractical as they demand substantial data and computational resources.

To address this challenge, we build our solution upon the idea of ***composition***. Rather than training a single model to handle an exponential number of possible combinations, we propose to train separate motion primitives for each individual category of specifications. At deployment time, we assume that an upstream module, such as a large language model, can decompose a natural-language instruction into a set of specifications online. The corresponding motion primitives can then be composed to generate a trajectory that follows the instruction. This approach significantly reduces complexity from exponential to linear: a relatively small set of motion primitives can support a combinatorially large space of instructions, enabling users to specify diverse robot behaviors required in real-world social navigation. Notably, in our setting, the relevant primitives are blended while composed, rather than stitched together sequentially [7, 8], since the robot's trajectory needs to satisfy all the specifications *simultaneously*.

To this end, we present **ComposableNav**, a composable, diffusion-based motion planner that composes motion primitives based on the instruction specifications to generate instruction-following motion trajectories. The core intuition motivating our approach is that diffusion models [9, 10] are highly effective at representing complex probability distributions, and that these models can be composed to form joint distributions [11]. Leveraging this property, we can separately train diffusion models to learn motion primitives, each represented as a distribution over trajectories that satisfy a specific instruction specification. At deployment time, ComposableNav composes the relevant motion primitives based on the provided instruction specifications, constructs the corresponding joint distribution, and samples a trajectory that simultaneously satisfies all specified instructions. Additionally, to avoid the onerous need for demonstrations of individual motion primitives, we introduce a two-stage training procedure consisting of supervised pre-training followed by reinforcement learning (RL) fine-tuning. Finally, to ensure real-time performance, we incorporate a model predictive controller (MPC) [12] and an online replanning strategy for low-latency action execution.

We demonstrate the effectiveness of ComposableNav through simulated and real-world experiments. With just six motion primitives (See Tab. 1 in Appendix for the instruction list), we build a testbed with 24 instructions featuring various unseen specification combinations. Our results show that ComposableNav excels at following unseen instructions compared to baseline approaches. Our main contributions are summarized as follows. (1) We introduce the use of *composition* as a strategy for instruction-following navigation in dynamic environments, making the problem tractable under limited data and computational resources for training. (2) We propose a diffusion-based learning method to model motion primitives as probability distributions, enabling their composition at deployment time. (3) We develop a two-stage training procedure—combining supervised pre-training and reinforcement learning fine-tuning—that effectively learns motion primitives without the need for specialized demonstration datasets for each primitive.

2 Related Work

Instruction Following Navigation. A key area in instruction-following navigation is vision-language navigation (VLN)[1, 2, 3], which combines natural language understanding with visual perception to guide agents through 3D environments[13, 14, 15, 16, 17, 18]. However, these methods assume static settings and overlook scenarios involving dynamic agents, where instructions specify interactions with moving agents. In contrast, social robot navigation focuses on enabling robots to operate in dynamic environments [19, 20, 21, 22, 23, 24, 25, 26, 27], but lacks instruction conditioning. Recent work has shown promise in using vision-language models (VLMs) to address this gap. CoNVOI [28] and Social-VLM-Nav [29] leverage VLMs' reasoning to interpret environment observations and suggest actions, but they face high inference latency and planning inconsistency. BehAV [30] addresses some of these issues by generating cost maps from VLM outputs, yet struggles with sample inefficiency in geometric planning. Our work proposes a novel alternative using diffusion models [9] to compose motion primitives, without relying on costly VLM inference.

Diffusion For Robotics. Diffusion models have emerged as powerful tools for solving a variety of robotics tasks [31, 32, 33, 34, 35, 36], with training typically performed using either supervised learning [32, 33, 35, 31] or RL [34, 37]. A distinctive advantage of diffusion models is their ability to guide the sampling process after training [38]. Janner et al. [39] first relate this guided sampling mechanism to the control-as-inference framework [40], demonstrating that classifier guidance enables the generation of motion plans for previously unseen goal configurations. However, designing such classifiers can be challenging. To address this, Luo et al. [36] interpret diffusion models as energy-based models, training separate models and composing them at inference time to generalize to novel environments. Building on this idea of composition, our work differs in that we do not assume access to diverse motion primitive datasets for supervised training. Instead, we propose using reinforcement learning-based [34] to fine-tune diffusion models, and composing them to generate trajectories that satisfy unseen combinations of specifications from an instruction.

3 Problem Formulation

We consider the problem of instruction-following robot navigation in dynamic environments, where the objective is to generate a motion trajectory τ that follows a given instruction I, based on the robot's observation O of the environment. We represent the motion trajectory τ as a sequence of 2D waypoints at fixed-time intervals, which are then tracked by a model predictive controller to produce fine-grained actions in real time. The observation O encodes the state of entities relevant to the instruction, such as the current and predicted positions of dynamic agents. Note that other representations are also possible, such as full SE(3) poses for τ or RGB images for O.

In this work, we assume an instruction I can be decomposed into a set of independent specifications $I \to \langle \phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(k)} \rangle$. Each specification $\phi^{(i)} : \tau \times O \to \{0, 1\}$ evaluates whether the trajectory meets the corresponding requirement, returning 1 if it does and 0 otherwise. To determine whether a trajectory τ follows an instruction I, τ must satisfy all relevant specifications. Formally,

$$\tau \text{ follows } I \text{ iff } \forall i \in [1, \cdots, k], \phi^{(i)}(\tau, O) = 1. \tag{1}$$

Solving this problem is challenging because the trajectory must simultaneously satisfy all specifications $\phi^{(i)}$, whose combinations can grow exponentially. In the following sections, we explain how leveraging diffusion models enables us to compose motion primitives and generate trajectories that can follow instructions during robot deployment.

4 Preliminaries

We provide a brief overview of the two key techniques used in ComposableNav, conditional diffusion models [10, 9, 41, 42] and denoising diffusion policy optimization (DDPO) [34].

4.1 Conditional Diffusion Models

In this work, we consider conditional diffusion probabilistic models [9, 10, 42], which belong to a family of generative models trained to represent a conditional distribution $p(x \mid c)$, where c is the

corresponding context. These models are trained to reverse a forward diffusion process $q(x_t \mid x_{t-1})$ that gradually adds Gaussian noise to the data $x_0 \sim p(x|c)$. To learn this reverse process, the model is trained to predict the noise ϵ at each step t using a denoising network, $f_\theta(x_t, t, c) \approx \epsilon$, where x_t is the noisy data at step t. The network is optimized using a training objective that penalizes the mean squared error between the predicted and actual noise value at step t:

$$\mathcal{L}_{\text{MSE}}(\theta) = \mathbb{E}_{x_0,\epsilon,t,c}\left[\|\epsilon - f_\theta(x_t, t, c)\|^2\right]. \tag{2}$$

This objective is derived from maximizing a variational lower bound on the data log-likelihood [9].

At inference time, the model generates a data sample by starting from Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and progressively denoising it using the learned denoising network for T steps. The reverse process at each timestep t follows a Gaussian distribution with a time-dependent covariance matrix $\sigma_t^2\mathbf{I}$, where σ_t^2 is treated as a hyperparameter:

$$p_\theta(x_{t-1} \mid x_t, c) = \mathcal{N}(x_t - f_\theta(x_t, t, c), \sigma_t^2\mathbf{I}), \tag{3}$$

This iterative process continues until a final sample x_0 is obtained, which approximates the true conditional distribution $p(x \mid c)$.

4.2 Denoising Diffusion Policy Optimization (DDPO)

ComposableNav follows the denoising diffusion policy optimization technique (DDPO) proposed by Black et al. [34] to use reinforcement learning (RL) to fine-tune diffusion models to generate the motion primitives corresponding to the instruction specifications. DDPO models the multi-step denoising process as a multi-step Markovian Decision Process (MDP), defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \rho_0, \mathcal{P}, R\rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, ρ_0 is the distribution of initial states, $\mathcal{P}$ is the transition kernel, and R is the reward function. We denote the timestep of this multi-step MDP as i. The denoising process is mapped into this MDP as follows:

$$\begin{aligned}
&s_i \triangleq \langle x_t, t, c\rangle \in \mathcal{S} \quad \pi(a_i \mid s_i) \triangleq p_\theta(x_{t-1} \mid x_t, c) \quad P(s_{i+1} \mid s_i, a_i) \triangleq \langle \delta_{x_{t-1}}, \delta_{t-1}, \delta_c\rangle \\
&a_i \triangleq x_{t-1} \in \mathcal{A} \qquad \rho_0(s_0) \triangleq \langle \mathcal{N}(0, \mathbf{I}), \delta_T, p(c)\rangle \quad R(s_i, a_i) \triangleq \begin{cases} r(x_0, c) & \text{if } t = 0, \\ 0 & \text{otherwise,} \end{cases}
\end{aligned} \tag{4}$$

where δ_y denotes the Dirac distribution with nonzero density only at y.

The key insight behind this technique is that the reverse process in a diffusion model is a Markovian process, where each denoising step $p_\theta(x_{t-1} \mid x_t, c)$ is modeled as a Gaussian distribution (see Eq. 3). By interpreting each denoising step as the policy $\pi(a_i \mid s_i)$ in an MDP, the policy itself becomes Gaussian, which allows for the exact evaluation of log-likelihoods and their gradients with respect to the diffusion model parameters. As a result, this formulation enables the use of policy gradient methods, such as PPO [43], to optimize the diffusion model's denoising network.

The DDPO algorithm alternates between (1) collecting denoising trajectories $\langle x_T, x_{T-1}, \ldots, x_0\rangle$ via sampling and (2) updating the model parameters using gradient descent. Finally, the policy gradient objective used in DDPO can be expressed as:

$$\mathcal{L} = \mathbb{E}\left[\sum_{t=1}^{T} \frac{p_\theta(x_{t-1} \mid x_t, c)}{p_{\theta_{\text{old}}}(x_{t-1} \mid x_t, c)} \nabla_\theta \log p_\theta(x_{t-1} \mid x_t, c) r(x_0, c)\right], \tag{5}$$

where the expectation is taken over trajectories generated using the previous model parameters θ_{old}.

5 ComposableNav

In this section, we present ComposableNav, a diffusion-based planner for instruction-following navigation. As shown in Fig. 2, ComposableNav first learns motion primitives via a two-stage training procedure (see Sec. 5.1). At deployment, given instruction specifications, it selects relevant primitives and composes them by summing the predicted noise from each diffusion model during the denoising process (Sec. 5.2). Finally, for real-time control, ComposableNav is paired with an MPC (see Appendix F.2). Please refer to our codebase for detailed implementation. [3]

[3] Code is released at https://github.com/ut-amrl/ComposableNav.

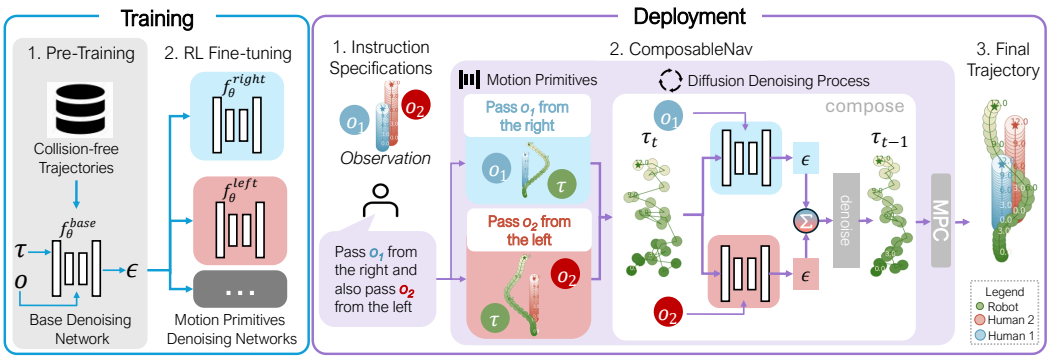


Figure 2: **ComposableNav Overview.** ComposableNav learns each motion primitive through a two-stage procedure: supervised pre-training (step 1) followed by RL fine-tuning (step 2). At deployment, it maps instruction specifications to the corresponding learned primitives and composes them by summing the predicted noise from each primitive's denoising network. The final trajectory is generated through the diffusion denoising process.

5.1 Learning Motion Primitives without Primitive-Specific Demonstration Data

Diffusion models for robotics are often trained in a supervised manner using large-scale demonstration datasets [36, 44, 32]. However, collecting such datasets for different robot motion primitives can be labor-intensive. To address this challenge, we make two key design choices. First, *dedicated datasets for each primitive behavior are costly to build, whereas general-purpose navigation datasets are far easier to obtain*. These general-purpose datasets only need to provide diverse, collision-free, goal-reaching trajectories in dynamic environments, which can be obtained from existing real-world datasets [45, 46] or generated in simulation [33]. Such data allows the pre-training of a base diffusion planner to generate diverse and feasible trajectories across various environments. Second, *verifying a solution is often easier than generating one*. In our setting, evaluating whether a trajectory satisfies an instruction (*e.g.,* with rule-based heuristics or vision-language models (VLMs)) is typically more straightforward than directly generating such a trajectory. Hence, for each primitive, we design a primitive-specific reward function that asserts instruction compliance and use RL to fine-tune the pre-trained base model [34]. Building on the two design choices above, we propose a two-stage framework: supervised pretraining followed by RL fine-tuning.

Supervised Pre-training. To pre-train a base diffusion model, we generate diverse trajectory data in simulation for simplicity and scalability. Following prior works [36, 33], we randomly synthesize environments with varying entities (*e.g.,* dynamic agents or terrain regions) and goal locations. We then use a geometric planner to generate a diverse set of collision-free, goal-reaching, and smooth trajectories. To account for dynamic environments, these trajectories must be *time-dependent*, so we employ a spatio-temporal Hybrid A* planner. In addition, we also want to capture the distribution of diverse feasible trajectories within the same environment (*e.g.,* both detouring left or right around an obstacle in front are feasible trajectories). Hence, we use a Rapidly-Exploring Random Tree planner to randomly generate candidate trajectories and then select waypoints along the trajectories as subgoals for Hybrid A* to track. Additionally, we also vary the hyperparameters for the planners (*e.g.,* velocity cost) to further enhance trajectory diversity.

Once a diverse set of time-dependent trajectories is generated, we pre-train a base diffusion model via supervised learning, using the objective in Eq. 2. The model learns a conditional denoising network $f_\theta^{(\text{base})}(\tau_t, t, O)$, which predicts the noise ϵ to denoise the trajectory τ_t at step t, conditioned on environment observations O. We adopt an object-centric representation for the observations, encoding each observation separately and then using a transformer encoder to attend over these embeddings to produce a global context feature. To handle varying trajectory lengths, we pad shorter trajectories with the final goal position to ensure uniform length during training.

RL Fine-tuning. We then fine-tune the base model *separately* for each motion primitive using RL, following the DDPO approach described in Sec. 4.2. For each primitive, we randomly generate simulation environments containing only the entities relevant to the corresponding instruction speci-

fication. The diffusion model then generates trajectories for these environments, which are evaluated using a reward function based on how well they align with the instruction. While the reward function can take various forms, we adopt a simple rule-based heuristic approach, as the primitives considered in our experiments are straightforward to evaluate (example shown in Appendix C). The resulting trajectories and rewards are stored in a replay buffer, and the model is updated using PPO [43]. Finally, after fine-tuning, we obtain multiple diffusion models $f_\theta^{\phi^{(i)}}(\tau_t, t, O)$, each representing a motion primitive associated with a specification $\phi^{(i)}$.

5.2 Generating Instruction-Following Trajectories via Composing Motion Primitives

ComposableNav models the instruction-following motion trajectories τ as a conditional distribution $p(\tau|\phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)})$, where each $\phi^{(i)}$ is a specification extracted from the instruction I, *i.e.,* $I \rightarrow \langle \phi^{(1)}, \cdots, \phi^{(k)} \rangle$, and each $o^{(i)}$ is the environment observation corresponding to $\phi^{(i)}$. We assume both the specifications and the environment observations can be extracted using off-the-shelf large language models and vision foundation models. Given the conditional independence assumption for each specification $\phi^{(i)}$ discussed in Sec. 3, the conditional distribution can be factorized as follows (derivation shown in Eq. 8):

$$p(\tau|\phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)}) \propto p(\tau) \prod_{i=1}^{k} \frac{p(\tau \mid \phi^{(i)}, o^{(i)})}{p(\tau)}. \tag{6}$$

Here, each conditional trajectory distribution $p(\tau \mid \phi^{(i)}, o^{(i)})$ corresponds to a motion primitive represented by a diffusion model with denoising network $f_\theta^{\phi^{(i)}}(\tau_t, t, o^{(i)})$. In contrast, the marginal trajectory distribution $p(\tau)$ is an unconditioned motion primitive, obtained by replacing the observation $o^{(i)}$ with a null input $\varnothing$, *i.e.,* $f_\theta^{\phi^{(i)}}(\tau_t, t, \varnothing)$, following the classifier-free guidance approach [42].

Following prior work [11, 36], we compose motion primitives by summing the predicted noise from denoising networks, with user-defined weights w_i controlling the guidance strength for the ith primitive (w_i is set to be 1 for all primitives in this work). The composed noise is:

$$\hat{\epsilon} = \frac{1}{k} \sum_{i=1}^{k} f_\theta^{\phi^{(i)}}(\tau_t, t, \varnothing) + \sum_{i=1}^{k} w_i (f_\theta^{\phi^{(i)}}(\tau_t, t, o^{(i)}) - f_\theta^{\phi^{(i)}}(\tau_t, t, \varnothing)). \tag{7}$$

Here, with separate diffusion models for each primitive, $p(\tau)$ is defined as the average of the unconditioned outputs $f_\theta^{\phi^{(i)}}(\tau_t, t, \varnothing)$ across all considered primitives.

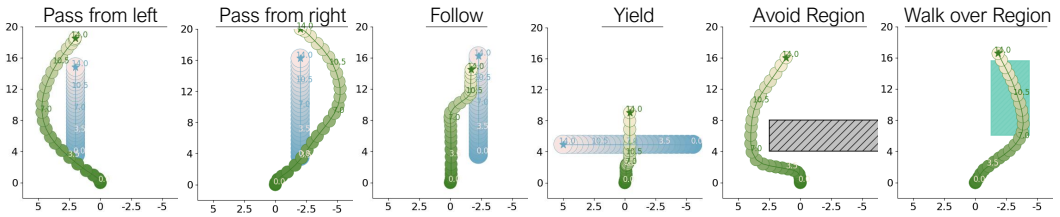
Finally, ComposableNav generates trajectories by iteratively applying the reverse diffusion process, starting from a noisy trajectory $\tau_T \sim \mathcal{N}(0, \mathbf{I})$ and denoising via $p_{\text{compose}}(\tau_{t-1} \mid \tau_t, \phi^{(1)}, o^{(1)}, \ldots, \phi^{(k)}, o^{(k)}) = \mathcal{N}(\tau_t - \hat{\epsilon}, \sigma_t^2 \mathbf{I})$. After T steps, the process yields a trajectory τ_0, drawn from a distribution concentrated on trajectories that satisfy all specifications of the given instruction (See Appendix B for an intuitive explanation of diffusion composition, based on the score-based interpretation [47]).

6 Experiments and Results

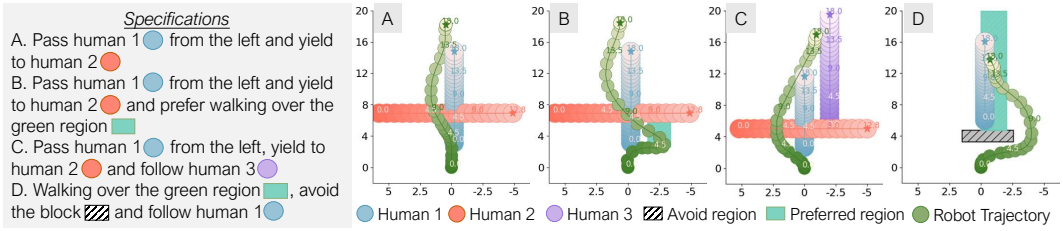
We evaluate ComposableNav in simulation and the real world to address the following questions: (1) Can ComposableNav learn individual motion primitives that satisfy each instruction specification without relying on demonstration data? (2) To what extent can ComposableNav compose motion primitives to generate trajectories that satisfy unseen combinations of specifications, in comparison to baseline approaches? (3) Can ComposableNav operate in real-time when deployed on a real-world robot and enable the robot to follow instructions in dynamic environments involving pedestrian interactions? In our experiments, we consider six navigation motion primitives, as shown in Fig. 3a and Appendix C, with training details provided in Appendix D.

6.1 Simulation Experiments

Environment Setup. Using six motion primitives, we built a testbed with 24 instructions featuring various unseen specification combinations (see Appendix E.3.2). Instructions are grouped by



(a) **Motion Primitives Learned in ComposableNav.**



(b) **Composed Robot Navigation Trajectories.** These examples show trajectories generated by combining learned motion primitives to follow complex instructions across diverse scenarios.

Figure 3: Qualitative Simulation Results.

complexity, based on the number of specifications—ranging from two to four—with each category comprising eight instructions. For each instruction, we randomly generate 20 test environments. Fig. 3b illustrates sample environments and corresponding robot behaviors, where dynamic humans are modeled as spheres and regions as rectangles.

Baselines. We compare ComposableNav with three VLM-based baselines (see Appendix E.2 for more details): (1) VLM-Social-Nav [29], which uses a VLM to select actions from predefined behaviors and translate them into a social cost function; (2) CoNVOI [28], which leverages a VLM to choose the robot's next waypoint from an annotated image; and (3) BehAV [30], which uses a VLM to generate instruction-grounded segmentation maps converted into cost maps for motion planning. Like ComposableNav, these methods pair their approach with a local motion planner [48].

Metrics. Success is defined using three criteria—Instruction Alignment (IA), Collision-Free (CF), and Goal Reaching (GR)—with a trajectory considered successful (SR=1) only if all are satisfied. IA is assessed via a rule-based function to check if the trajectory follows the given instructions (See Appendix C).

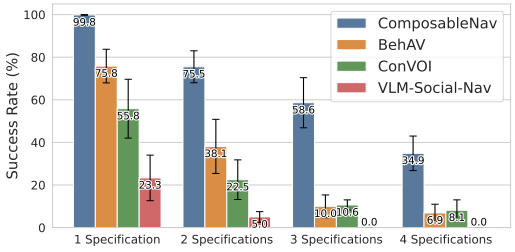


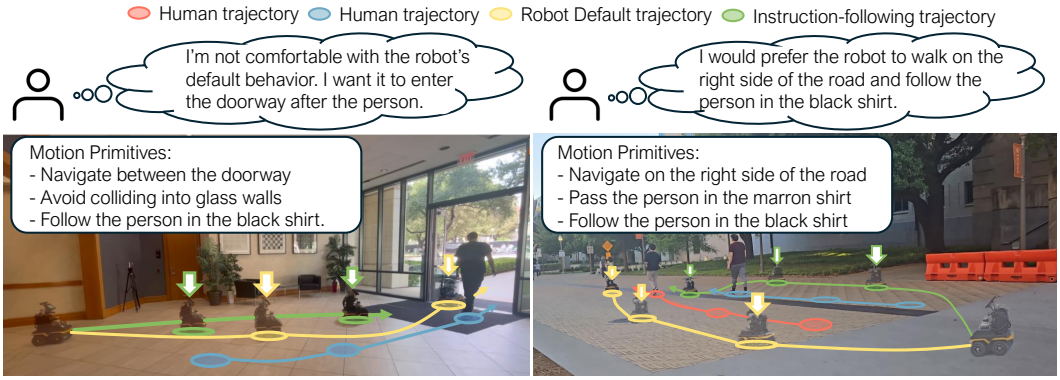
Figure 4: **Simulation Evaluation Results.** Bar plots show mean success rates with standard error bars. As instruction complexity increases, ComposableNav consistently outperforms baselines.

Results & Analysis. We first assess whether our two-stage training procedure enables diffusion models to learn individual motion primitives without direct demonstration data. As illustrated in Fig. 3a, the robot can generate navigation behaviors tailored to different primitives and environments. Also, the fine-tuned models execute motion primitives with near-perfect accuracy, whereas pre-trained models perform significantly worse (see Tab. 2 in Appendix).

Fig. 4 presents the quantitative results, with further analysis in Appendix E.3.2. ComposableNav consistently outperforms all baselines across all levels of instruction complexity. While the baseline methods exhibit moderate performance with a single specification, their success rates degrade rapidly as the number of specifications increases. In contrast, ComposableNav maintains fairly high success rates even under complex multi-specification instructions, achieving 58.6% with three and 34.9% with four specifications, where all baselines fall below 11%. This demonstrates the robustness of ComposableNav in following complex unseen combinations of instruction specifications.

Finally, Fig. 3b further illustrates how ComposableNav composes motion primitives to generate diverse trajectories in response to previously unseen combinations of instruction specifications. For instance, to follow instruction *A*, the trajectory must first slow down to yield to Person 2, then accelerate to overtake Person 1. *B* differs from *A* by introducing an additional specification to walk over the green region, prompting the robot to take a detour to the right. Similarly, *C* and *D* elicit complex navigation behaviors, such as following a human or avoiding a region in front.

6.2 Deployment on a Real Robot



(a) Indoor Narrow Doorway (b) Outdoor Open Environment

Figure 5: **Real world Experiments**. ComposableNav allows for customizing the robot's behavior.

We deployed ComposableNav on a Clearpath Jackal robot using only its onboard compute in real-world environments (see Appendix F.1). We tested six instructions in two common scenarios: (1) navigating a narrow doorway and (2) walking in an open outdoor space. To ensure safety, all instructions were first validated in simulation. Each instruction was tested in 10 repeated trials to evaluate real-world performance. Success rates, shown in Fig. 6, were consistently high, indicating that ComposableNav can be effectively deployed in real-world settings (additional results detailed in Appendix G).

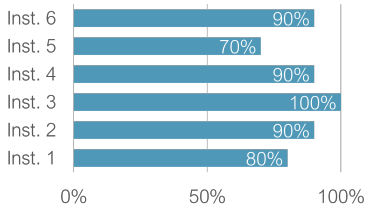


Figure 6: Real-World Deployment Success Rate

Fig. 9 also illustrates how ComposableNav enables users to customize robot behavior beyond the default navigation behavior. For example, the default geometric stack (yellow trajectory) prioritizes fast, collision-free navigation but may pass uncomfortably close to humans. In contrast, ComposableNav allows users to specify preferences, such as following a person through a doorway instead of overtaking them (green trajectory), supporting more human preference-aligned behavior. Finally, we measured system latency, which depends on the number of composed motion primitives. In the most complex case involving four primitives, initial trajectory generation averaged 0.4s, whereas trajectory replanning required only 0.06s (see additional results in Appendix F.1).

7 Conclusion

This paper presents ComposableNav, a composable diffusion-based motion planner that composes navigation motion primitives to generate trajectories satisfying diverse, previously unseen combinations of instruction specifications. To address the lack of primitive-specific demonstrations, ComposableNav employs a two-stage training pipeline: (1) supervised pre-training to learn a base diffusion model for dynamic navigation, and (2) reinforcement-learning fine-tuning that specializes this base model for each motion primitive. Our simulation and real-world experiments show that the design of ComposableNav enables it to learn and compose motion primitives effectively, and it can be deployed on real-world robots to operate in real time.

8 Limitations and Future Work

This work has several limitations that future research could address.

First, we only considered six commonly used navigation primitives when composing novel behaviors. These primitives are relatively simple and can be described with straightforward rule-based reward functions (see Sec. C). However, manually designing such reward functions does not scale well as the number of primitives grows. Importantly, our method is not tied to a particular way of crafting rewards. A promising and direct direction for improving scalability is to leverage vision-language models (VLMs) as verifiers—shown effective in DDPO [34]—to automatically learn diverse and complex behaviors without relying on handcrafted rewards.

Second, we assume that tasks such as parsing instructions into specifications and detecting relevant observations can be handled by existing methods, such as off-the-shelf LLMs and VLMs. Since our focus is on composable planning, we abstract these components away in our experiments. Future work may stack high-level VLM-based modules, as in prior studies [30, 49], on top of ComposableNav to close the loop. Furthermore, a high-level task planner [50, 51] could be introduced to further support long-horizon instruction-following navigation.

Third, although ComposableNav significantly outperforms baseline methods, it still shows a notable decline in success rate as the number of instruction specifications increases. This limitation may stem from the underlying composition strategy: following Liu et al. [11], our method composes diffusion models by summing the predicted noise from individual denoising networks. As Du et al. [38] highlight, however, this approach can lead to suboptimal results. Future work may explore more advanced sampling techniques for diffusion models—such as Hamiltonian Monte Carlo [52, 53]—to improve composition performance under higher instruction complexity.

Acknowledgments

This work has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) and the Learning Agents Research Group (LARG) at UT Austin. AMRL research is supported in part by NSF (CAREER-2046955, OIA-2219236, DGE-2125858, CCF-2319471), ARO (W911NF-23-2-0004), Amazon, and JP Morgan. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858), ONR (N00014-24-1-2550), ARO (W911NF-17-2-0181, W911NF-23-2-0004, W911NF-25-1-0065), DARPA (Cooperative Agreement HR00112520004 on Ad Hoc Teamwork) Lockheed Martin, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Chief Scientist of Sony AI and receives financial compensation for that role. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[1] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-long.524. URL https://aclanthology.org/2022.acl-long.524/.

[2] Y. Zhang, Z. Ma, J. Li, Y. Qiao, Z. Wang, J. Chai, Q. Wu, M. Bansal, and P. Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=yiqeh2ZYUh. Survey Certification.

[3] S.-M. Park and Y.-G. Kim. Visual language navigation: a survey and open challenges. *Artif. Intell. Rev.*, 56(1):365–427, Mar. 2022. ISSN 0269-2821. doi:10.1007/s10462-022-10174-9. URL https://doi.org/10.1007/s10462-022-10174-9.

[4] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[5] B. Singh, R. Kumar, and V. P. Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artif. Intell. Rev.*, 55(2):945–990, Feb. 2022. ISSN 0269-2821. doi:10.1007/s10462-021-09997-9. URL https://doi.org/10.1007/s10462-021-09997-9.

[6] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martin-Martin, P. Stone, K.-H. Zeng, and K. Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning, 2024. URL https://arxiv.org/abs/2409.16578.

[7] W. Liu, N. Nie, R. Zhang, J. Mao, and J. Wu. Learning compositional behaviors from demonstration and language. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=fR1rCXjCQX.

[8] V. Myers, C. Zheng, O. Mees, K. Fang, and S. Levine. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=qUSa3F79am.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[10] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[11] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum. Compositional visual generation with composable diffusion models. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, page 423–439, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19789-5. doi:10.1007/978-3-031-19790-1_26. URL https://doi.org/10.1007/978-3-031-19790-1_26.

[12] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016. doi:10.1109/ICRA.2016.7487277.

[13] S. Raychaudhuri, S. Wani, S. Patel, U. Jain, and A. Chang. Language-aligned waypoint (LAW) supervision for vision-and-language navigation in continuous environments. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4018–4028, Online and Punta Cana,

Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.328. URL https://aclanthology.org/2021.emnlp-main.328/.

[14] M. Z. Irshad, N. Chowdhury Mithun, Z. Seymour, H.-P. Chiu, S. Samarasekera, and R. Kumar. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 4065–4071, 2022. doi:10.1109/ICPR56361.2022.9956561.

[15] K. Chen, J. K. Chen, J. Chuang, M. Vázquez, and S. Savarese. Topological planning with transformers for vision-and-language navigation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11271–11281, 2021. doi:10.1109/CVPR46437.2021.01112.

[16] S. Liu, A. Hasan, K. Hong, R. Wang, P. Chang, Z. Mizrachi, J. Lin, D. L. McPherson, W. A. Rogers, and K. Driggs-Campbell. Dragon: A dialogue-based robot for assistive navigation with visual language grounding. *IEEE Robotics and Automation Letters*, 9(4):3712–3719, 2024.

[17] P. Chang, S. Liu, and K. Driggs-Campbell. Learning visual-audio representations for voice-controlled robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[18] 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.

[19] J. Holtz, S. Andrews, A. Guha, and J. Biswas. Iterative program synthesis for adaptable social navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6256–6261, 2021. doi:10.1109/IROS51168.2021.9636540.

[20] A. Wang, C. Mavrogiannis, and A. Steinfeld. Group-based motion prediction for navigation in crowded environments. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=kn0bbYqSowX.

[21] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard. Time dependent planning on a layered social cost map for human-aware robot navigation. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2015. doi:10.1109/ECMR.2015.7324184.

[22] Z. Xie, P. Xin, and P. Dames. Towards safe navigation through crowded dynamic environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4934–4940, 2021. doi:10.1109/IROS51168.2021.9636102.

[23] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3517–3524, 2021.

[24] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350, 2017. doi:10.1109/IROS.2017.8202312.

[25] A. H. Raj, Z. Hu, H. Karnan, R. Chandra, A. Payandeh, L. Mao, P. Stone, J. Biswas, and X. Xiao. Rethinking social robot navigation: Leveraging the best of two worlds. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16330–16337, 2024. doi:10.1109/ICRA57147.2024.10611710.

[26] Z. Zheng, C. Cao, and J. Pan. A hierarchical approach for mobile robot exploration in pedestrian crowd. *IEEE Robotics and Automation Letters*, 7(1):175–182, 2022.

[27] S. Liu, H. Xia, F. C. Pouria, K. Hong, N. Chakraborty, Z. Hu, J. Biswas, and K. Driggs-Campbell. Height: Heterogeneous interaction graph transformer for robot navigation in crowded and constrained environments. *arXiv preprint arXiv:2411.12150*, 2025. URL https://arxiv.org/abs/2411.12150.

[28] A. J. Sathyamoorthy, K. Weerakoon, M. Elnoor, A. Zore, B. Ichter, F. Xia, J. Tan, W. Yu, and D. Manocha. Convoi: Context-aware navigation using vision language models in outdoor and indoor environments, 2024. URL https://arxiv.org/abs/2403.15637.

[29] D. Song, J. Liang, A. Payandeh, A. H. Raj, X. Xiao, and D. Manocha. Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models. *IEEE Robotics and Automation Letters*, 10(1):508–515, 2025. doi:10.1109/LRA.2024.3511409.

[30] K. Weerakoon, M. Elnoor, G. Seneviratne, V. Rajagopal, S. H. Arul, J. Liang, M. K. M. Jaffar, and D. Manocha. Behav: Behavioral rule guided autonomy using vlms for robot navigation in outdoor scenes, 2024. URL https://arxiv.org/abs/2409.16484.

[31] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *Arxiv*, 2024.

[32] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[33] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923, 2023. URL https://api.semanticscholar.org/CorpusID:260191316.

[34] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

[35] M. Seo, Y. Cho, Y. Sung, P. Stone, Y. Zhu, and B. Kim. Presto: Fast motion planning using diffusion models based on key-configuration environment representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[36] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du. Potential based diffusion motion planning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Qb68Rs0p9f.

[37] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. In *arXiv preprint arXiv:2409.00588*, 2024.

[38] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. Grathwohl. Reduce, reuse, recycle: compositional generation with energy-based diffusion models and mcmc. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[39] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[40] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL https://arxiv.org/abs/1805.00909.

[41] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713845393.

[42] J. Ho and T. Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[44] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[45] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone. Socially CompliAnt Navigation Dataset (SCAND): A Large-Scale Dataset of Demonstrations for Social Navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, 2022. doi:10.1109/LRA.2022.3184025.

[46] A. Zhang, C. Eranki, C. Zhang, J.-H. Park, R. Hong, P. Kalyani, L. Kalyanaraman, A. Gamare, A. Bagad, M. Esteva, and J. Biswas. Toward robust robot 3-d perception in urban environments: The ut campus object dataset. *IEEE Transactions on Robotics*, 40:3322–3340, 2024. doi:10.1109/TRO.2024.3400831.

[47] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[48] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. Technical report, 1995.

[49] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=9_8LF30mOC.

[50] B. Liu, Y. Jiang, et al. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv:2304.11477*, 2023.

[51] Z. Hu, F. Lucchetti, C. Schlesinger, Y. Saxena, A. Freeman, S. Modak, A. Guha, and J. Biswas. Deploying and evaluating llms to program service mobile robots. *IEEE Robotics and Automation Letters*, 9(3):2853–2860, 2024. doi:10.1109/LRA.2024.3360020.

[52] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987. ISSN 0370-2693. doi:https://doi.org/10.1016/0370-2693(87)91197-X. URL https://www.sciencedirect.com/science/article/pii/037026938791197X.

[53] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

[54] J. Biswas and M. Veloso. Episodic non-markov localization: Reasoning about short-term and long-term features. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3969–3974, 2014. doi:10.1109/ICRA.2014.6907435.

[55] S. Zhou, Y. Du, S. Zhang, M. Xu, Y. Shen, W. Xiao, D.-Y. Yeung, and C. Gan. Adaptive online replanning with diffusion models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

A Derivation for Eq. 6

$$
\begin{aligned}
& p(\tau|\phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)}) && (8)\\
&= \frac{p(\tau, \phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)})}{p(\phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)})} && \triangleleft \text{ Bayes' Rule}\\
&\propto p(\tau, \phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)})\\
&= p(\tau) p(\phi^{(1)}, o^{(1)}, \cdots, \phi^{(k)}, o^{(k)} \mid \tau) && \triangleleft \text{ Bayes' Rule}\\
&= p(\tau) \prod_{i=1}^{k} p(\phi^{(i)}, o^{(i)} \mid \tau) && \triangleleft \text{ Conditional Independence}\\
&\propto p(\tau) \prod_{i=1}^{k} \frac{p(\tau \mid \phi^{(i)}, o^{(i)})}{p(\tau)} && \triangleleft \text{ Bayes' Rule}
\end{aligned}
$$

B Composing Diffusion Models via Score Function Interpretation

Diffusion models belong to the family of score-based generative models. These models learn to estimate the score function [47, 11], which is defined as the gradient of the log-probability density with respect to the input, i.e., $\nabla_x \log p(x)$. *Intuitively, the score function indicates the direction in which a data point should be moved to increase its likelihood under the data distribution.*

In the case of diffusion models, the denoising network $f_\theta(x_t, t)$ can be interpreted as being proportional to the score function. The denoising process can thus be viewed as iteratively moving the noisy sample x_t in the direction predicted by the model, gradually transforming it into a high-probability sample from the data distribution.

Given this interpretation, composing multiple diffusion models corresponds to computing the sum of their score functions, $\sum_{i=1}^{k} f_\theta^{(i)}(x_t, t)$, where each $f_\theta^{(i)}$ represents the score from the i-th diffusion model being composed and $\hat{\epsilon}$ is the composed score. The generative process for composing these models becomes [11]:

$$
p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_t - \sum_{i=1}^{k} f_\theta^{(i)}(x_t, t), \sigma_t^2 \mathbf{I}) \tag{9}
$$

This process can be understood as guiding the sample toward regions that are simultaneously high-probability under all the models being composed. Hence, the data can be seen as sampling from the joint distribution defined by all the composed diffusion models.

C Motion Primitives

In this work, we consider six commonly used navigation motion primitives, as listed in Tab. 1. In the following subsections, we present the pseudocode that outlines how we define the specification $\phi^{(i)}$ for each motion primitive using heuristic rule-based functions.

Table 1: Instruction Specifications for Navigation Motion Primitives

Motion Primitive (MP)	Instruction Specification
Pass a person from the left (L)	The robot should pass the person from the left side.
Pass a person from the right (R)	The robot should pass the person from the right side.
Follow behind a person (F)	The robot should stay in a specific region behind the person relative to the person's position.
Yield to a person (Y)	The robot should not cross the region in front of the person.
Walk through a region (W)	The robot's trajectory should overlap with the specified region.
Avoid walking through a specified region (A)	The robot's trajectory should not overlap with the specified region, which may be defined either by a terrain feature or by a person's location.

C.1 Pass a person from the left

Pseudocode 1 Motion Primitive $\phi^{(\mathrm{L})}$

```
def criteria_left (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    for t in time_at_each_waypoint:
        region = extract_region_left_of_obs(O, t)
        robot_position = extract_position(τ, t)
        if region.contains(robot_position):
            return 1
    return 0
```

C.2 Pass a person from the right

Pseudocode 2 Motion Primitive $\phi^{(\mathrm{R})}$

```
def criteria_right (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    for t in time_at_each_waypoint:
        region = extract_region_right_of_obs(O, t)
        robot_position = extract_position(τ, t)
        if region.contains(robot_position):
            return 1
    return 0
```

C.3 Follow behind a person

For this primitive, we consider a period of time to evaluate whether the robot has successfully followed the human, which is defined as the final few seconds before the robot reaches the goal.

Pseudocode 3 Motion Primitive $\phi^{(\mathrm{F})}$

```
def criteria_follow (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    time_period = get_relevant_time_period(time_at_each_waypoint)
    for t in time_period:
        region = extract_region_behind_obs(O, t)
        robot_position = extract_position(τ, t)
        if not region.contains(robot_position):
            return 0
    return 1
```

C.4 Yield to a person

Pseudocode 4 Motion Primitive $\phi^{(\mathrm{Y})}$

```
def criteria_yield (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    for t in time_at_each_waypoint:
        region = extract_region_in_front_of_obs(O, t)
        robot_position = extract_position(τ, t)
        if region.contains(robot_position):
            return 0
    return 1
```

C.5 Walk through a region

Pseudocode 5 Motion Primitive $\phi^{(\mathrm{W})}$

```
def criteria_prefer (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    for t in time_at_each_waypoint:
        region = extract_region(O, t)
        robot_position = extract_position(τ, t)
        if region.contains(robot_position):
            return 1
    return 0
```

C.6 Avoid walking through a region

Pseudocode 6 Motion Primitive $\phi^{(\mathrm{A})}$

```
def criteria_avoid (τ, O):
    time_at_each_waypoint = extract_time_at_each_waypoint(τ)
    for t in time_at_each_waypoint:
        region = extract_region(O, t)
        robot_position = extract_position(τ, t)
        if region.contains(robot_position):
            return 0
    return 1
```

D Training Diffusion Model

D.1 Model Design

We build upon a publicly available diffusion model implementation[4]. Our denoising network, f_θ, consists of a 1D UNet architecture augmented with a context encoder. Each observation—whether a dynamic human or a specific region—along with the goal, is first encoded using a multilayer perceptron (MLP). The dynamic human is represented as a predicted future trajectory, estimated under a constant velocity assumption. In contrast, a region is represented as a rectangle defined by the positions of its four corners. These embeddings are then passed through a vision transformer to generate context features, following the approach introduced in prior work [36].

[4] https://github.com/lucidrains/denoising-diffusion-pytorch

D.2 Training

To train the base model, we collected trajectories across three types of environments: (1) collision-free trajectories in dynamic settings, (2) trajectories that avoid specified regions, and (3) trajectories that intentionally traverse specific regions. The first two types were generated in simulation by randomly placing obstacles and planning trajectories to avoid them. For the third type, we sampled trajectories from the first two types, randomly selected a region that each trajectory passes through, and re-labeled this region as the observation to form training pairs.

We collected approximately 2 million collision-free trajectories and trained the denoising network for 2000 epochs, using a learning rate of 2×10^{-4} and a dropout rate of 0.1. Training followed the classifier-free guidance approach [42], where the model was conditioned on a null context (represented by zero vectors) with a probability of 20%, instead of using features extracted from the context encoder. We also applied an exponential moving average (EMA) to the model parameters during training, which stabilizes optimization and improves generalization by smoothing out noisy updates. Following prior work [33], the diffusion model performs a total of 25 denoising steps using an exponential noise schedule, generating a trajectory composed of a sequence of fixed-length, time-dependent waypoints.

To fine-tune the base denoising network for each motion primitive, we adapted the DDPO implementation[5]. For simplicity, we replaced the original vision-language model (VLM)-based reward function with a heuristic-based one, designed according to the specific definitions of each motion primitive in this work. During each training epoch, we generated 32 different environments and trained the model for a total of 1000 epochs. A noteworthy observation from our experiments is that a significantly lower learning rate greatly enhances training performance. Based on this finding, we adopted a learning rate of 1×10^{-6}, which is also consistent with results reported in the literature [6].

E Additional Simulation Experiment Details

E.1 Simulation ComposableNav Setup

We evaluate ComposableNav in a $20 \times 20\,\mathrm{m}^2$ 2D simulation arena. Dynamic humans are modeled as spheres, whose future positions are predicted under a constant-velocity assumption, while the static environment is represented as a rectangular region specified by the coordinates of its four corner points. For each instruction, 20 environments are randomly initialized, assigning initial positions and speeds to the entities based on the specific requirements of the instruction. The simulation operates at a control frequency of $\Delta t = 0.1\,\mathrm{s}$, and each episode lasts for a maximum of 300 timesteps, equivalent to 30.0 seconds.

E.2 Baseline Setup

In this work, we consider three baseline methods: VLM-Social-Nav [29], CoNVOI [28], and BehAV [30]. These baselines fall into two categories: the first two treat the VLM as a black-box policy that proposes a target action (e.g., next waypoint or velocity) for a geometric planner to track, while the third computes composable cost maps for planning.

None of these baseline methods is explicitly designed to solve the problem considered in this work. Therefore, we adapt them for our experimental setup. For VLM-Social-Nav and CoNVOI, we use the latest GPT-4.1 model and disregard the high inference latency associated with invoking a remote VLM and focus on evaluating their prediction accuracy. Additionally, we modify these approaches by providing annotated screenshots of the simulates scenes and prompting the VLMs for reasoning.

For BehAV, whose core idea is to create composable costmaps and plan trajectories over them (similar to Voxposer [49]), we simplify the setup by abstracting away the segmentation vision model.

[5] https://github.com/kvablack/ddpo-pytorch

Instead, we provide BehAV with ground-truth annotations obtained directly from the simulation environment and evaluate solely how well costmap-based planning enables instruction following.

For each baseline method, we conducted a grid search over various combinations of motion planner hyperparameters—specifically, different weights of the constituent cost functions—using a small tuning set. We then selected the hyperparameters that achieved the highest overall success rates.

E.3 Supplementary Quantitative Results

E.3.1 Learning Motion Primitives

Beyond demonstrating that the RL fine-tuning procedure enables the learning of effective motion primitives, we further analyze results obtained from comparing the pre-trained and fine-tuned models. As shown in Tab. 2, the pre-trained model, which is trained to generate collision-free, goal-reaching trajectories, consistently achieves these objectives across all evaluated motion primitives. In particular, the pre-trained model achieves a 100% success rate for the *"Avoid walking through a region"* motion primitive. This outcome is anticipated, as avoiding designated regions is closely aligned with the collision avoidance objective emphasized during the pre-training phase.

Table 2: Comparison of Fine-tuned and Pre-trained Diffusion Models for Representing Motion Primitives.

MP	Pre-trained Model				Fine-tuned Model			
	SR(%)↑	IA(%)↑	CF(%)↑	GR(%)↑	SR(%)↑	IA(%)↑	CF(%)↑	GR(%)↑
L	44.0	44.0	100.0	100.0	100.0	100.0	100.0	100.0
R	37.0	37.0	100.0	100.0	100.0	100.0	100.0	100.0
F	27.0	27.0	100.0	100.0	99.0	100.0	100.0	99.0
Y	48.0	48.0	100.0	100.0	100.0	100.0	100.0	100.0
W	34.0	34.0	100.0	100.0	100.0	100.0	100.0	100.0
A	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

E.3.2 Composing Motion Primitives

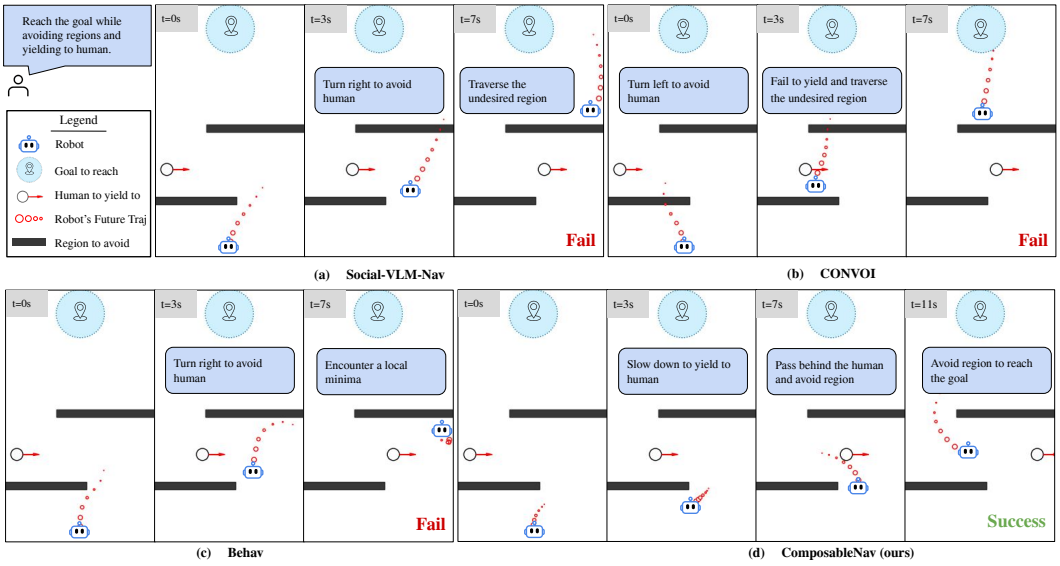


Figure 7: Qualitative Simulation Results

We present the quantitative results of 24 motion primitive combinations in our simulation testbed, as detailed in Tab. 3. Each combination (e.g., "L+R") corresponds to a specific natural language instruction (e.g., "Pass person 1 from the left and person 2 from the right"), which maps to a distinct robot motion trajectory illustrated in Fig. 1. Given the similarity between the primitives "pass a person from the left" and "pass a person from the right"—which differ only in direction—we unify

Table 3: Detailed Simulation Evaluation Results

#	Combination	VLM-based Policy								Compose Costmaps				Compose Primitives			
		VLM-Social-Nav (%)				Convoi (%)				Behav (%)				ComposableNav (ours) (%)			
		SR↑	**IA↑**	**CF↑**	**GR↑**	**SR↑**	**IA↑**	**CF↑**	**GR↑**	**SR↑**	**IA↑**	**CF↑**	**GR↑**	**SR↑**	**IA↑**	**CF↑**	**GR↑**
1 Primitive	L	55.0	55.0	100.0	100.0	70.0	75.0	95.0	100.0	65.0	65.0	100.0	100.0	**100.0**	100.0	100.0	100.0
	R	55.0	55.0	100.0	100.0	70.0	75.0	95.0	100.0	65.0	65.0	100.0	100.0	**100.0**	100.0	100.0	100.0
	F	5.0	5.0	100.0	100.0	0.0	0.0	95.0	100.0	70.0	75.0	100.0	95.0	**99.0**	100.0	100.0	99.0
	Y	25.0	25.0	95.0	100.0	40.0	40.0	95.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0
	W	0.0	0.0	100.0	100.0	55.0	55.0	100.0	100.0	55.0	55.0	100.0	100.0	**100.0**	100.0	100.0	100.0
	A	0.0	0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0
	Overall	23.3	23.3	99.2	100.0	55.8	57.5	96.7	100.0	75.8	76.7	100.0	99.2	**99.8**	100.0	100.0	99.8
2 Motion Primitives	L+R	5.0	5.0	100.0	100.0	0.0	0.0	100.0	100.0	0.0	0.0	100.0	100.0	**33.0**	34.0	86.0	100.0
	P+F	10.0	10.0	100.0	100.0	5.0	5.0	90.0	100.0	0.0	0.0	90.0	100.0	**85.0**	85.0	99.0	100.0
	Y+P	20.0	20.0	100.0	100.0	55.0	55.0	95.0	100.0	90.0	90.0	100.0	100.0	**90.0**	93.0	97.0	100.0
	Y+F	5.0	5.0	95.0	100.0	0.0	0.0	80.0	95.0	85.0	85.0	100.0	100.0	**85.0**	85.0	100.0	100.0
	W+P	0.0	0.0	100.0	100.0	20.0	20.0	95.0	100.0	5.0	5.0	100.0	100.0	**60.0**	61.0	97.0	100.0
	W+Y	0.0	0.0	100.0	100.0	35.0	35.0	95.0	100.0	40.0	40.0	100.0	100.0	**99.0**	99.0	100.0	100.0
	A+P	0.0	0.0	90.0	100.0	65.0	65.0	100.0	100.0	50.0	50.0	100.0	95.0	**67.0**	68.0	99.0	100.0
	A+F	0.0	0.0	100.0	100.0	0.0	0.0	100.0	100.0	35.0	40.0	100.0	35.0	**85.0**	85.0	100.0	100.0
	Overall	5.0	5.0	98.1	100.0	22.5	22.5	94.4	99.4	38.1	38.8	98.8	91.2	**75.5**	76.2	97.2	100.0
3 Motion Primitives	P+F+Y	0.0	0.0	100.0	100.0	5.0	5.0	85.0	100.0	0.0	0.0	100.0	100.0	**30.0**	34.0	91.0	100.0
	P+F+W	0.0	0.0	100.0	100.0	5.0	5.0	80.0	100.0	0.0	0.0	100.0	95.0	**58.0**	61.0	92.0	100.0
	P+Y+W	0.0	0.0	100.0	100.0	20.0	20.0	90.0	100.0	5.0	5.0	100.0	90.0	**38.0**	42.0	92.0	100.0
	W+W+Y	0.0	0.0	100.0	100.0	10.0	10.0	90.0	100.0	5.0	5.0	100.0	50.0	**87.0**	87.0	100.0	100.0
	A+A+Y	0.0	0.0	90.0	100.0	5.0	5.0	100.0	95.0	15.0	75.0	100.0	15.0	**86.0**	86.0	99.0	100.0
	A+W+Y	0.0	0.0	100.0	100.0	**20.0**	20.0	100.0	100.0	10.0	10.0	100.0	60.0	0.0	0.0	99.0	100.0
	A+P+F	0.0	0.0	100.0	100.0	15.0	15.0	95.0	100.0	0.0	0.0	100.0	90.0	**93.0**	95.0	94.0	100.0
	A+W+F	0.0	0.0	100.0	40.0	5.0	5.0	95.0	40.0	45.0	50.0	90.0	85.0	**77.0**	77.0	98.0	100.0
	Overall	0.0	0.6	98.8	92.5	10.6	10.6	91.9	91.9	10.0	18.1	98.8	80.6	**58.6**	60.2	95.6	100.0
4 Motion Primitives	A+W+F+Y	0.0	0.0	90.0	60.0	0.0.	0.0	90.0	60.0	30.0	30.0	95.0	60.0	**33.0**	35.0	94.0	98.0
	A+W+F+P	0.0	5.0	90.0	50.0	0.0	0.0	85.0	50.0	0.0	0.0	95.0	10.0	**59.0**	61.0	72.0	100.0
	A+W+A+Y	0.0.	0.0	100.0	50.0	35.0.	35.0	100.0	70.0	2.0.	20.0	100.0	25.0	**46.0**	46.0	76.0	100.0
	W+W+Y+A	0.0.	0.0	90.0	100.0	5.0.	5.0	65.0	100.0	0.0.	0.0	85.0	100.0	**71.0**	78.0	86.0	100.0
	W+P+Y+A	0.0.	0.0	70.0	100.0	25.0.	25.0	75.0	100.0	5.0.	5.0	70.0	100.0	**28.0**	34.0	78.0	100.0
	W+P+F+A	0.0.	0.0	80.0	80.0	0.0.	0.0	75.0	100.0	0.0.	0.0	80.0	100.0	**24.0**	31.0	81.0	100.0
	P+F+Y+A	0.0.	0.0	80.0	100.0	0.0.	0.0	85.0	100.0	0.0.	0.0	100.0	100.0	**18.0**	23.0	82.0	95.0
	A+W+Y+A	0.0.	0.0	60.0	100.0	0.0.	0.0	75.0	100.0	0.0.	0.0	90.0	55.0	0.0.	0.0	92.0	99.0
	Overall	0.0	0.6	82.5	80.0	8.1	8.1	81.2	84.4	6.9	6.9	89.4	68.8	**34.9**	38.5	82.6	99.0

them under a general instruction category: "Pass a person" (denoted as P) for motion composition purposes. Accordingly, we evaluate our method, ComposableNav, on its ability to handle both left and right variants using a single instruction specification.

The quantitative results are summarized in Tab. 3. Across all testbed scenarios, ComposableNav consistently outperforms all baseline methods in terms of success rate, particularly as the number of motion primitives in an instruction increases. While baseline methods perform reasonably with simple instructions, their performance deteriorates notably with more complex instruction sets.

Methods relying on Vision-Language Models (VLMs) as black-box policies perform particularly poorly. These models are not designed for such navigation tasks and often fail to maintain planning consistency, especially as instruction complexity grows. Similarly, BehAV performs adequately with one or two motion primitives but suffers as composition complexity increases. In addition, BehAV has the lowest goal-reaching rate, which suggests that such a costmap-based method tends to get trapped in local minima and is unable to complete tasks within the allotted time.

Furthermore, baseline methods exhibit high variance in performance across different instruction combinations. In many cases, they fail to generate any viable, instruction-following trajectory. In contrast, ComposableNav—*despite not being explicitly trained for any possible instruction composition*—demonstrates strong generalization capabilities and consistently higher success rates across a wide range of scenarios.

To complement the quantitative analysis, we provide a qualitative illustration in Fig. 7. Here, we observe that while VLM-based methods may initially steer the robot in the correct direction, they lack the responsiveness and consistency needed for sustained instruction following. These methods often begin to avoid regions or yield to pedestrians, but then fail to complete subsequent specifi-

cations. The BehAV method often gets stuck in local minima and fails to reach the goal within the allowed time. In contrast, ComposableNav effectively produces instruction-aligned behaviors, simultaneously satisfying all given specifications—such as avoiding restricted regions and yielding to oncoming pedestrians.

F Real-World Deployment

F.1 Robot Setup

We deploy ComposableNav on a Clearpath Jackal robot equipped with a Zed 2i camera for human tracking and an Ouster LiDAR for generating point cloud data to create an obstacle map for collision avoidance, as shown in Fig. 8. For localization, we apply ENML [54], which provides robust position estimation. The system is built using ROS and consists of a navigation stack with three main modules: a perception module, a diffusion planning module, and an MPC motion planning module. The perception module leverages Zed's internal human-tracking algorithm to detect and track humans while using Ouster's point cloud data to detect obstacles for collision avoidance. The diffusion planning module loads diffusion models corresponding to various motion primitives and composes the appropriate models based on instructions and environmental observations. The MPC motion planning module tracks the time-dependent trajectory generated by the diffusion planner and computes real-time motion control commands. All computations are executed entirely onboard, utilizing an Intel i7-9700TE CPU and an NVIDIA RTX A2000 GPU.

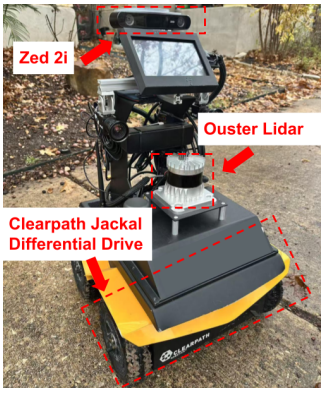


Figure 8: Robot Setup.

F.2 Real-time Deployment

In our deployment, the motion controller and trajectory replanning run at fixed frequencies: the motion controller operates at $10\,\text{Hz}$ (every $0.1\,\text{s}$), while trajectory replanning executes at $0.67\,\text{Hz}$ (every $1.5\,\text{s}$). We define *real-time* operation as producing outputs within the motion control cycle of $0.1\,\text{s}$, ensuring both control and replanning complete within this bound. Below, we describe the implementation details that enable ComposableNav to meet this requirement with onboard hardware.

F.2.1 Real-time Motion Control

To produce motion commands in real-time, we employ a Model Predictive Path Integral (MPPI) [12], a sampling-based MPC controller, to track the time-dependent trajectories generated by the composed diffusion models. During navigation, MPPI uses a differential drive kinematic model to predict the robot's future states and minimizes the deviation between these predictions and the target trajectory over a short planning horizon. It also enforces constraints on acceleration and velocity to guarantee feasible and safe control inputs.

F.2.2 Real-time Trajectory Replanning

To enable trajectory replanning on only the robot's onboard compute, we adopt the adaptive online replanning framework introduced in prior work [55], specifically the *Replan from Previous Context* method. The core insight is that the current trajectory is typically close to optimal, so instead of discarding it and replanning from scratch, ComposableNav perturbs the trajectory by applying a few forward diffusion steps $q(x_t \mid x_{t-1})$ and then partially denoises it to generate an updated trajectory conditioned on the latest observations.

In practice, we apply five forward diffusion steps to inject noise into the current trajectory, followed by five reverse diffusion steps to denoise it. During this process, the states already visited by the robot are fixed, and only the future segments are updated.

We introduce two key optimizations to ensure efficiency: (1) Since all fine-tuned diffusion models share an identical architecture derived from a common base, we leverage PyTorch's vectorized mapping operation (`vmap`) to batch their execution in parallel. (2) We further accelerate inference using PyTorch's compilation feature (`torch.compile`).

With these optimizations, ComposableNav achieves real-time replanning entirely on onboard compute, with latency results summarized in Tab. 4.

Table 4: ComposableNav Inference Latency on Robot Hardware

Latency	**# of Composed MPs**			
	1	**2**	**3**	**4**
Initial Plan (s)↓	0.144 ± 0.014	0.243 ± 0.009	0.329 ± 0.014	0.413 ± 0.010
Replan (s)↓	0.027 ± 0.002	0.036 ± 0.004	0.049 ± 0.003	0.060 ± 0.005

G Additional Robot Deployment Details

G.1 Quantitative Experiment Details

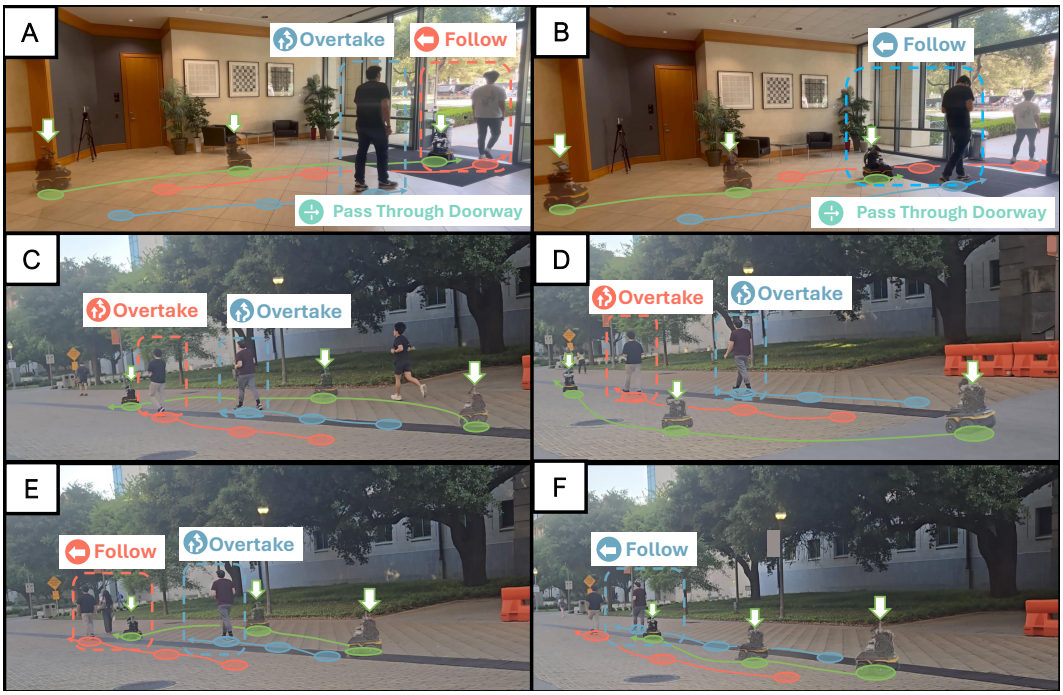


Figure 9: Quantitative Experiment Illustration

We evaluate ComposableNav in two common real-world scenarios: navigating through a narrow doorway and walking outdoors in an open environment, using a total of six instructions—two for the doorway scenario and four for the outdoor scenario, as illustrated in Fig. 9.

Doorway Instructions:

- **Instruction 1:** "Go through the doorway after the person wearing a black shirt and before the person wearing a white shirt" (Fig. 9A)
- **Instruction 2:** "Go through the doorway before the person wearing a black shirt" (Fig. 9B)

Outdoor Instructions:

- **Instruction 3:** "Pass both the person wearing a black shirt and a maroon shirt from the right side of the road" (Fig. 9C)
- **Instruction 4:** "Pass both the person wearing a black shirt and a maroon shirt from the left side of the road" (Fig. 9D)
- **Instruction 5:** "Pass the person wearing a maroon shirt and follow the person wearing a black shirt" (Fig. 9E)
- **Instruction 6:** "Follow the person wearing a black shirt" (Fig. 9F)

Each instruction was tested over 10 trials, and we report the corresponding success rates in Fig. 9. These experiments demonstrate how ComposableNav enables customizable robot behavior that aligns with human preferences. For instance, in the doorway scenario, a human operator might prefer the robot to be polite by following the person in a black shirt, or alternatively, instruct it to hurry and enter after the person in a white shirt. In the outdoor scenario, preferences may include keeping to a specific side of the road to follow the human flow, or adjusting the robot's pace to follow a specific individual, such as someone in a maroon or black shirt.

G.2 Additional Real-World Deployment Demo

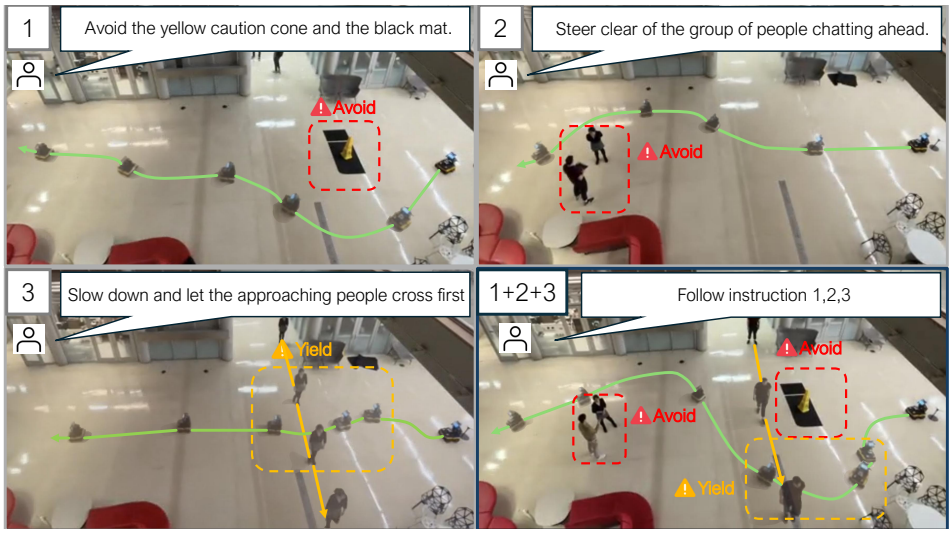
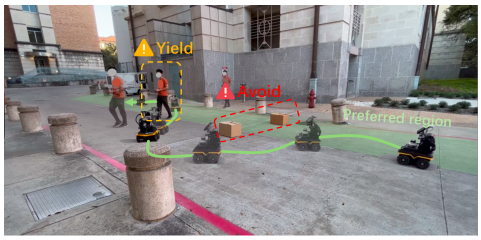


Figure 10: Real-World Composition Experiment

(a) Yield to the person in the white coat and follow the person wearing a dark hoodie.



(b) Avoid walking between the boxes, yield to the person, and stay on the right side of the road.

G.3 Deployment Failure Case Analysis

We conducted a qualitative analysis of the failure cases observed when deploying ComposableNav on the robot and identified two common issues. The first issue stems from human tracking errors. Since both the robot and the human are in continuous motion, the person may temporarily exit the camera's field of view—particularly when the robot turns—causing the system to lose track of them, even if they later reappear. While we applied a simple nearest-neighbor heuristic to reassign the human based on previous tracking data, occasional failures still occur, where the robot is unable to reliably re-identify the person. The second issue arises during replanning. We observed that the newly generated plan can sometimes diverge significantly from the original one. This can lead the MPPI controller to issue large acceleration or deceleration commands, resulting in jerky movements. Consequently, the robot may overshoot its intended state and struggle to stay on the planned path. We plan to address these issues in future work