EXAMPLE-BASED PLANNING VIA DUAL GRADIENT FIELDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Path planning is one of the key abilities of an intelligent agent. However, both the learning-based and sample-based planners remain to require explicitly defining the task by manually designing the reward function or optimisation objectives, which limits the scope of implementation. Formulating the path planning problem from a new perspective, *Example-based planning* is to find the most efficient path to increase the likelihood of the target distribution by giving a set of *target examples*. In this work, we introduce Dual Gradient Fields (DualGFs), an offline-learning example-based planning framework built upon score matching. There are two gradient fields in DualGFs: a *target gradient field* that guides task completion and a *support gradient field* that ensures moving with physical constraints. In the learning process, instead of interacting with the environment, the agents are trained with two offline examples, *i.e.*, the target gradients and support gradients are trained by *target examples* and *support examples*, respectively. The support examples are randomly sampled from free space, *e.g.*, states without collisions. DualGF is a weighted mixture of the two fields, combining the merits of the two fields together. To update the mixing ratio adaptively, we further propose a fields-balancing mechanism based on Lagrangian-Relaxation . Experimental results across four tasks (navigation, tracking, particle rearrangement, and room rearrangement) demonstrate the scalability and effectiveness of our method. Our code and demonstrations can be found at sites.google.com/view/dualgf

1 INTRODUCTION

Planning paths to reach a goal is a fundamental function of an intelligent agent (Russell, 2010) and has a wide range of real-world applications, such as navigation (Patle et al., 2019), object tracking (Zhong et al., 2019), and object rearrangement (King et al., 2016). Existing planning algorithms, whether sampling-based (LaValle et al., 1998a; Karaman & Frazzoli, 2011) or learning-based (Kulathunga, 2021; Yu et al., 2020; Tamar et al., 2016), need manually design task-specific objective functions for optimisation or reward functions for learning. This severely limits the implementation scope of planning since many real-world tasks are hard to design the objectives/ reward with human priors, *e.g.*, tidying up a house, or rearranging a desktop.

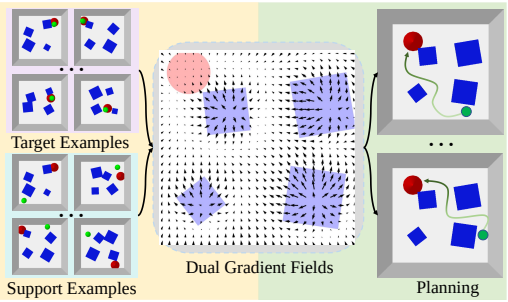


Figure 1: Our task setting. **Left:** The agent learns task specification from target examples and physical constraints from support examples during training. **Right:** The agent plans a path in novel condition during test phase.

In this paper, we consider a novel data-driven planning paradigm: *Example-based planning*, in which the user can specify the task by providing a set of *target examples*, rather than programming a task-specific objective. Benefiting from such a paradigm, example-based planning can scale to various tasks, particularly tasks with implicit goals, *i.e.*specifying the task with a target distribution instead of a specific goal state. Besides, the agent needs to infer the environmental constraints to safely move in a physical world. Previous approaches either learn physical constraints from interacting with the environment and collision penalty (Wu et al., 2022), offline demonstrations (Janner et al., 2022),

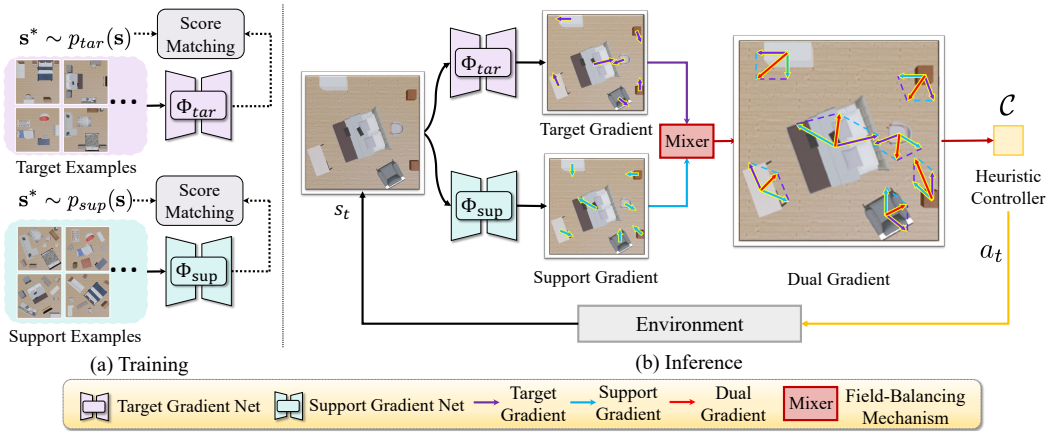


Figure 2: DualGF aims at estimating a target gradient field and a support gradient field. a): We train a target score network and a support score network from examples via score-matching objective. b): DualGF computes target and support gradient at each time step, where the *target gradient* guides the task completion (*i.e.*, move objects to tidy layout) and the *support gradient* directs the objects to avoid collision. DualGF adaptively mixes the two gradients via a field-balancing mechanism to output the dual gradient. Incorporated with a low-level controller, DualGF can interpret the dual gradient into an action.

or exhaustively sampling points at test time (LaValle et al., 1998a). However, online interaction is costly and unsafe, while offline demonstrations are expensive to collect and test-time sampling is time inefficient.

To this end, we propose a fully example-based planning framework that learns two gradient fields with different purposes from examples by score-matching (Vincent, 2011), namely *DualGF*. DualGF consists of two fields: A *target gradient field* and a *support gradient field*. The target gradient field estimates the gradient of the target distribution so as to provide the fastest direction to accomplish the task. The support gradient field learns to reverse the perturbed state back to the free space so as to help avoid collision. To combine the merits of the two fields, we further introduce a gradient mixer to adaptive balance the trade-off between the two gradients (keep safe vs. reach goal) when constructing the dual gradient field in execution. In practice, we can also incorporate the dual gradient field with a low-level controller to output primitive actions in control. The two gradients are trained from two sets of examples, respectively. For the target gradient field, we collect a set of target states sampled from the target distribution, such as a set of tidied rooms. For the support gradient field, we provide the agent with another set of examples (*support examples*) that are uniformly sampled from the free space, *i.e.*states without collision. The support examples are abundant and easy to obtain in real scenarios, *e.g.*, randomly initialised objects, which largely alleviate. As illustrated in Fig. 1, the agent can learn generalisable inference of task specification and physical constraints from target and support examples and planning in *unseen* environment.

Our experiments validate the generalisation ability of the DualGF planning framework across a variety of tasks, including classical planning tasks such as navigation (), tracking (), and planning tasks without explicit goal specification such as object rearrangement (Wu et al., 2022). Specifically, the proposed DualGF significantly outperforms the learning-based baselines in planning performance and efficiency while achieving comparable performance with reference approaches that use the ground truth model or test-time sampling. Ablation studies also demonstrate the effectiveness of the proposed support gradient field and field-balancing mechanism.

In conclusion, our contributions are summarised as follows: a) We reformulate the path planning problem in a data-driven paradigm, where we specify the tasks with examples rather than manually design objectives; b) We propose a novel score-based planning framework DualGF that can adaptively integrate two gradient fields trained from different example sets so as to output instructions to complete a task; c) We conduct experiments in four tasks to demonstrate the scalability of our method, and empirical results show that DualGF significantly outperforms the state-of-the-art methods in efficiency and safety.

2 RELATED WORK

Learning from Demonstration. Example-based planning can be viewed as a special case of Learning from demonstration (LfD). LfD is a long-studied problem aiming at learning a policy from only a set of expert trajectories. There are two mainstreams of LfD: Behavioural Cloning (BC) (Pomerleau, 1991; Ross & Bagnell, 2010; Ross et al., 2011), which learns a policy in a supervised manner; and Inverse Reinforcement Learning (IRL) (Fu et al., 2017; Liu et al., 2020; Kostrikov et al., 2018; Ziebart et al., 2008), which finds a cost function under which the expert is uniquely optimal. Different from LfD, we train the agent from only two sets of examples instead of whole demonstrations. There are some RL-based methods that consider an example-based setting: VICE (Fu et al., 2018) is similar to AIRL (Fu et al., 2017), but is designed for learning from a set of success examples. SQIL (Reddy et al., 2019) is a method modified from SAC (Haarnoja et al., 2018) that labels success examples with a reward of +1. RCE (Eysenbach et al., 2021), a modification of the actor-critic-based method (Fujimoto et al., 2018; Haarnoja et al., 2018), directly learns a value function from transitions and successful examples. Recently, Offline-RCE (Hatch et al., 2022) extends RCE to an offline setting. Different from these methods, our method requires neither interaction with the environment nor offline demonstrations.

Sampling-based Planning. Sampling-based planning algorithms, such as the probabilistic roadmap method (PRM, (Kavraki & Latombe, 1994)) and the rapidly exploring random tree (RRT, (LaValle et al., 1998b)), are dominant in traditional path planning algorithms. PRT and its variants: (Dobson & Bekris, 2014); (You et al., 2021); (Chai et al., 2022); (Hüppi et al., 2022), intend to generate a roadmap and find a collision-free path from a starting point to a goal region on the roadmap. RRT and its variants: (Strub & Gammell, 2020); (Gammell et al., 2020) (Li et al., 2022); (Strub & Gammell, 2022), iteratively build a tree by expanding towards the random sample point instead of directly connecting to the sample point. Different from these methods, our approach enables path planning in an end-to-end paradigm without test-time sampling, which is more time-efficient and scalable to high-dimensional tasks. There are also learning-based approaches for path planning, most of which are built up on reinforcement learning (Kulathunga, 2021; Yu et al., 2020; Tamar et al., 2016). Besides, the recently proposed (Janner et al., 2022) learns a diffusion-based generative model (Ho et al., 2020) from offline trajectories for planning. Different from these approaches, our method does not need reward engineering, online interaction or offline trajectories.

3 PROBLEM STATEMENT

The example-based planning can be described by a tuple $(\mathcal{S}, \mathcal{S}_f, p_{tar}, \rho, \epsilon)$ where $\mathcal{S}$ denotes the state space (*e.g.*, configurations of objects and obstacles) while $\mathcal{S}_f \subseteq \mathcal{S}$ denotes the free space (the valid states without object collision or transboundary). The planning agent $\pi : \mathcal{S} \to \mathcal{S}$ outputs a state within an epsilon-ball of the input state, *i.e.*, $||\pi(\mathbf{s}_t) - \mathbf{s}_t|| \leq \epsilon$. During training, the agent is given a set of target examples $S^*_{tar} = \{\mathbf{s}^*, \mathbf{s}^* \sim p_{tar}(\mathbf{s})\}$ where $p_{tar} : \mathcal{S} \to \mathbb{R}^+$ denotes a target distribution. Similar to (Wu et al., 2022), the goal of the planning agent is to find the most efficient path to increase the likelihood of the target distribution:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\rho(\mathbf{s}_0), \tau \sim \pi} \left[\sum_{\mathbf{s}_t \in \tau} \gamma^t \log p_{tar}(\mathbf{s}_t) \right], \text{s.t.} \quad \forall \pi(\mathbf{s}) \in \mathcal{S}_f \tag{1}$$

To this end, the agent needs to infer the target distribution, *i.e.*, knowing where to move can probably succeed, and the free space, *i.e.*, avoiding collisions between objects and obstacles.

In this work, we consider the offline setting of the example-based planning, where the agent has to infer physics constraints from support examples instead of interacting with the environment. The support examples are uniformly sampled from the free space $S^*_{sup} = \{\mathbf{s}^*, \mathbf{s}^* \sim p_{sup}(\mathbf{s})\}$ where

$$p_{sup}(\mathbf{s}) = \begin{cases} \dfrac{1}{|\mathcal{S}_f|}, & \mathbf{s} \in \mathcal{S}_f \\ 0, & \text{otherwise} \end{cases}, \quad |\mathcal{S}_f| = \int_{\mathcal{S}_f} ds \tag{2}$$

The rationale of 'replacing online interaction with examples' is that the support examples are enough for inferring the free space $\mathcal{S}_f$ while the online interactions are unsafe and costly. To this end, the agent should learn generalisable inference of free space instead of overfitting to the specific environment.

4 METHODOLOGY

Overview: To tackle example-based planning, our key idea is to learn gradient fields from examples. As illustrated in Fig. 2 (a), the *target gradient field* $\mathbf{\Phi}^{\theta}_{tar}$ and the *support gradient field* $\mathbf{\Phi}^{\phi}_{sup}$ is trained from the target examples and support examples via score-matching respectively. The target gradient guides the task completion by providing the fastest direction to increase the task likelihood. The support gradient helps avoid collision by pointing to the inner of the free space. Finally, we construct a policy that plans the next state based on the mixture of the two gradients, *i.e.*, the *dual gradient*. In the control scenario, a heuristic-based controller can be used to interpret the dual gradient into a low-level action.

4.1 PLANNING WITH THE DUAL GRADIENT FIELDS

We tackle the example-based planning by greedily maximising the target likelihood. The agent aims at searching a next state that maximises the target likelihood from the free space:

$$\underset{\mathbf{s}_{t+1}}{\operatorname{argmax}} \log p_{tar}(\mathbf{s}_{t+1}), \quad \text{s.t. } \log p_{sup}(\mathbf{s}_{t+1}) \geq \log \frac{1}{|\mathcal{S}_f|} \tag{3}$$

However, this objective is hard to optimise since the $\log p_{sup}(\mathbf{s}_{t+1})$ in the constraint is non-differentiable on the boundary of the free space $\partial\mathcal{S}_f$. As a result, we choose to 'soften' the support distribution by adding a small Gaussian noise $p^{\sigma}_{sup}(\hat{\mathbf{s}}) = \int p^{\sigma}(\hat{\mathbf{s}}|\mathbf{s})p_{sup}(s)d\mathbf{s}$, where $p^{\sigma}(\hat{\mathbf{s}}|\mathbf{s}) = \mathcal{N}(\mathbf{s}, \sigma^2 I)(\hat{\mathbf{s}})$. In this way, we replace the original objective in Eq. 3 with the following:

$$\underset{\mathbf{s}_{t+1}}{\operatorname{argmax}} \log p_{tar}(\mathbf{s}_{t+1}), \quad \text{s.t. } \log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) \geq c \tag{4}$$

where $c \in \mathbb{R}$ is a threshold indicating the conservative degree of the policy. The problem Eq. 4 can be further related to an unconstrained optimisation with a penalty term by the Lagrangian method:

$$\underset{\mathbf{s}_{t+1}}{\operatorname{max}} \underset{\lambda \geq 0}{\operatorname{min}} \log p_{tar}(\mathbf{s}_{t+1}) + \lambda \left(\log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) - c \right) \tag{5}$$

where $\lambda > 0$ is known as a Lagrangian multiplier. The $\mathbf{s}_{t+1}$ and λ can be updated iteratively for multiple steps to obtain a solution. Here, we choose to update the $\mathbf{s}_{t+1}$ and λ **only one step** for Eq. 5 since the s_{t+1} is close to $\mathbf{s}_t$ in our problem, *i.e.*, $||\mathbf{s}_{t+1} - \mathbf{s}_t||_2 \leq \epsilon$. :

$$\begin{aligned} \mathbf{s}_{t+1} &= \mathbf{s}_t + \mu_s \left(\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s}_t) + \lambda_t \nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_t) \right) \\ \lambda_{t+1} &= \text{ReLU} \left(\lambda_t - \mu_\lambda \left(\log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) - c \right) \right) \end{aligned} \tag{6}$$

where μ_s and μ_λ are two step sizes. Empirically, the latter is set as a fixed scalar $\mu_\lambda = 0.01$ while the former is properly set to ensure $||\mathbf{s}_{t+1} - \mathbf{s}_t||_2 \leq \epsilon$. In this way, our planning policy is derived as:

$$\begin{aligned} \pi_{dual}(\mathbf{s}_{t+1}|\mathbf{s}_t) = \mathbf{s}_t + \epsilon \frac{\mathbf{g}_{dual}}{||\mathbf{g}_{dual}||_2}, \quad \mathbf{g}_{dual} &= \underbrace{\mathbf{g}_{tar} + \lambda_t \mathbf{g}_{sup}}_{dual\,gradient} \\ \mathbf{g}_{tar} = \underbrace{\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s}_t)}_{target\,gradient}, \quad \mathbf{g}_{sup} &= \underbrace{\nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_t)}_{support\,gradient} \end{aligned} \tag{7}$$

Intuitively, the state is updated by the mixture of the *target gradient* $\nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s})$ and the *support gradient* $\nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_t)$ weighted by λ_t. As shown in Fig. 3, the target gradients is a vector field on the state space that points to the goal region, so as to guide the task completion. The support gradient fields point to the inner region of the free space. The closer a state is to the boundary of the free space, the larger the magnitudes of its support gradient will 'push' it into the inner region.

4.2 UPDATING MIXING RATE VIA SUPPORT GRADIENT FIELD

From Eq. 6, we can infer that λ_t is updated to maintain the 'safety level' $\log p^{\sigma}_{sup}(\mathbf{s})$ at a certain threshold c, since a larger λ_t will leads to an increase of $\log p^{\sigma}_{sup}(\mathbf{s})$ on the futher states, vice versa.

However, updating λ_t requires the exact computation of $\log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) - c$ which is hard to estimate directly. Thus, we seek to develop a practical field-balancing mechanism. We first approximate

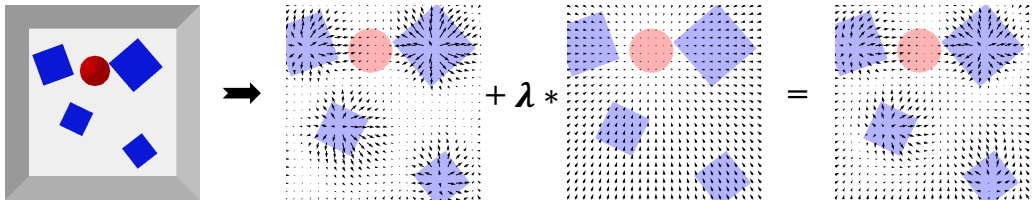


Figure 3: Visualisation of learned target, support and dual gradient fields on *Navigation* task. The L2-norm of support gradients is small in the free space but grows significantly faster when close to the boundary. As a result, the direction of dual gradients is consistent with the target gradient in the free space and consistent with the support gradient near the boundary.

$\log p^{\sigma}_{sup}(\mathbf{s}_{t+1})$ from $\log p^{\sigma}_{sup}(\mathbf{s}_0)$ via first-order Taylor expansion:

$$\begin{aligned}\log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) - c &= -\left(c - \log p^{\sigma}_{sup}(\mathbf{s}_0)\right) + \log p^{\sigma}_{sup}(\mathbf{s}_{t+1}) - \log p^{\sigma}_{sup}(\mathbf{s}_0) \\ &\approx -\left(c - \log p^{\sigma}_{sup}(\mathbf{s}_0)\right) + \sum_{k=0}^{t} \left\langle \nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_k), \mathbf{s}_{k+1} - \mathbf{s}_k \right\rangle\end{aligned} \tag{8}$$

We further assume that $\mathbf{s}_0$ is always initialised far away from the boundary of the free space so that $\mathrm{Var}_{\mathbf{s}_0 \sim \rho(\mathbf{s})}[\log p^{\sigma}_{sup}(\mathbf{s})] \approx 0$. Thus, the remaining term $c - \log p^{\sigma}_{sup}(\mathbf{s}_0)$ can be regarded as a constant c' in the sense of expectation:

$$\mathbb{E}_{\mathbf{s}_0 \sim \rho(\mathbf{s})}\left[c - \log p^{\sigma}_{sup}(\mathbf{s}_0)\right] = c - \mathbb{E}_{\rho(\mathbf{s})}[p^{\sigma}_{sup}(\mathbf{s})] = c - \underbrace{H(\rho(\cdot), p^{\sigma}_{sup}(\cdot))}_{constant} \stackrel{def}{=} c' \tag{9}$$

The whole planning framework is summarised below:

$$\begin{aligned}\pi_{dual}(\mathbf{s}_{t+1}|\mathbf{s}_t) &= \mathbf{s}_t + \epsilon \frac{\mathbf{g}_{dual}}{||\mathbf{g}_{dual}||_2}, \quad \mathbf{g}_{dual} = \nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s}_t) + \lambda_t \nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_t) \\ \lambda_{t+1} &= \mathrm{ReLU}\left(\lambda_t - \mu_\lambda \left(\sum_{k=0}^{t} \left\langle \nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_k), \mathbf{s}_{k+1} - \mathbf{s}_k \right\rangle - c'\right)\right)\end{aligned} \tag{10}$$

4.3 Incorporating DualGF with Low-level Controller

Except for planning, DualGF can also tackle the control tasks incorporated with a low-level controller $\mathcal{C} : \Delta\mathcal{S} \to \mathcal{A}$. For instance, if the atomic action space is velocity (with speed limit v_{max}), the dual gradient can be cast into a target velocity by a heuristic-based controller:

$$\mathbf{a}_t = \mathcal{C}(\mathbf{g}_{dual}) = \frac{v_{max}}{||\mathbf{g}_{dual}||_2} \cdot \mathbf{g}_{dual}, \quad \mathbf{g}_{dual} = \nabla_{\mathbf{s}} \log p_{tar}(\mathbf{s}_t) + \lambda_t \nabla_{\mathbf{s}} \log p^{\sigma}_{sup}(\mathbf{s}_t) \tag{11}$$

Under the general setting, when the dynamics is too complex to apply a heuristic-based controller, we can train an inverse dynamics model $\mathcal{P}^{-1} : \Delta\mathcal{S} \to \mathcal{A}$ to translate the state-change output by the DualGF into an action:

$$\mathbf{a}_t = \mathcal{P}^{-1}(\mathbf{s}_t, \mathbf{s}_{t+1}) = \mathcal{P}^{-1}(\mathbf{s}_t, \pi_{dual}(\mathbf{s}_t)) \tag{12}$$

In this work, we focus on the effectiveness of the high-level module trained under the fully example-based setting, *i.e.*, the dual gradient fields, instead of complexity of the dynamics. Hence, we conduct experiments on tasks with velocity-based action space (*e.g.*, linear or angular velocity) in Sec. 5.

4.4 Learning Gradient Fields from Examples

Finally, we seek to estimate the target and support gradient fields mentioned above to realise our planning framework. Thanks to the score-based generative modeling, we can obtain a guaranteed estimation of these gradient fields from examples.

Preliminary: The Denoising Score-Matching (DSM) proposed by (Vincent, 2011) aims at estimating the *score function* of a data distribution $\nabla_{\mathbf{s}} \log p_{data}(\mathbf{s})$. Given a sample set $\{\mathbf{s}^* \sim p_{data}(\mathbf{s})\}$, DSM pre-specifies a noise distribution $q_\sigma(\widetilde{\mathbf{s}}|\mathbf{s})$, *e.g.*, $\mathcal{N}(0, \sigma^2 I)$, and trains a *score network* $\mathbf{\Phi}_\theta(\cdot)$ to denoise the perturbed data samples:

$$\mathcal{L}(\theta) = \mathbb{E}_{\widetilde{\mathbf{s}} \sim q_\sigma(\widetilde{\mathbf{s}}|\mathbf{s}), \mathbf{s} \sim p_{data}(\mathbf{s})} \left[||\mathbf{\Phi}_\theta(\widetilde{\mathbf{s}}) - \frac{\mathbf{s} - \widetilde{\mathbf{s}}}{\sigma^2}||_2^2 \right] \tag{13}$$

This objective guarantees the optimal score network satisfies $\mathbf{\Phi}^*_\theta(\mathbf{s}) = \nabla_{\mathbf{s}} q_\sigma(\mathbf{s})$ almost surely. When σ is small enough, we have $\nabla_{\mathbf{s}} q_\sigma(\mathbf{s}) \approx \nabla_{\mathbf{s}} \log p_{data}(\mathbf{s})$, so that $\mathbf{\Phi}^*_\theta(\mathbf{s}) \approx \nabla_{\mathbf{s}} \log p_{data}(\mathbf{s})$.

Training: Adopting DSM, we train a *target score network* $\mathbf{\Phi}^\theta_{tar} : \mathbb{R}^{d_a + d_c} \to \mathbb{R}^{d_a}$ and a *support score network* $\mathbf{\Phi}^\phi_{sup} : \mathbb{R}^{d_a + d_c} \to \mathbb{R}^{d_a}$ from target examples S^*_{tar} and support examples S^*_{sup} respectively, where d_a denotes the dimension of agent-state space (*e.g.*, agent's position and orientation) and d_c denotes the dimension of conditional-state space (*e.g.*, agent's category, obstacles' state).

With $\mathbf{s}_a \in \mathbb{R}^{d_a}$ and $\mathbf{s}_c \in \mathbb{R}^{d_c}$ denoted as agent-state and conditional-state respectively ($\mathbf{s} = [\mathbf{s}_a, \mathbf{s}_c]$), the training objectives of both score networks are as follows:

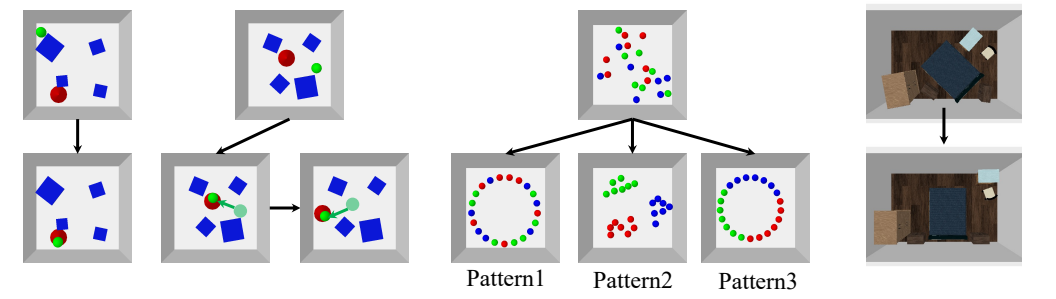
$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{q_\sigma(\widetilde{\mathbf{s}}_a|\mathbf{s}_a), p_{tar}(\mathbf{s})} \left[||\mathbf{\Phi}^\theta_{tar}([\widetilde{\mathbf{s}}_a, \mathbf{s}_c]) - \frac{\mathbf{s}_a - \widetilde{\mathbf{s}}_a}{\sigma^2}||_2^2 \right] \\ \mathcal{L}(\phi) &= \mathbb{E}_{q_\sigma(\widetilde{\mathbf{s}}_a|\mathbf{s}_a), p_{sup}(\mathbf{s})} \left[||\mathbf{\Phi}^\phi_{sup}([\widetilde{\mathbf{s}}_a, \mathbf{s}_c]) - \frac{\mathbf{s}_a - \widetilde{\mathbf{s}}_a}{\sigma^2}||_2^2 \right] \end{aligned} \tag{14}$$

We adopt a variant of DSM (Song et al., 2020) that conducts DSM under different noise scales simultaneously. We defer more implementation details to Appendix. C.1.

5 Experiment Setups

5.1 Tasks

To validate the rationality of our task setting, we conduct experiments on classical path planning tasks with explicit task reward: *Navigation* and *Tracking*. We also conduct experiments on tasks without explicit goal specification to evaluate the ability to infer the goal specification from examples: *Ball Rearrangement* and *Room Rearrangement*, where the *Ball Rearrangement* uses explicit target distribution for better numerical analysis and the *Room Rearrangement* is the hardest task to demonstrate the effectiveness of our method in the complex scenario. The *Navigation and Tracking* are implemented in PyBullet (Coumans & Bai, 2016–2021), while the *Ball Rearrangement* and *Room Rearrangement* basically follow the setting of (Wu et al., 2022). The task demonstrations are illustrated on Fig. 4. We defer complete task details to Appendix A.



(a) Navigation (b) Tracking (c) Ball Rearrangement (d) Room Rearrangement

Figure 4: Examples of four tasks used in experiments. *Navigation* and *Tracking*: To reach/track a goal location with dynamic obstacles that changes in each episode. *Ball Rearrangement:* To rearrange balls into certain patterns. *Room rearrangement:* To rearrange furniture into a reasonable layout.

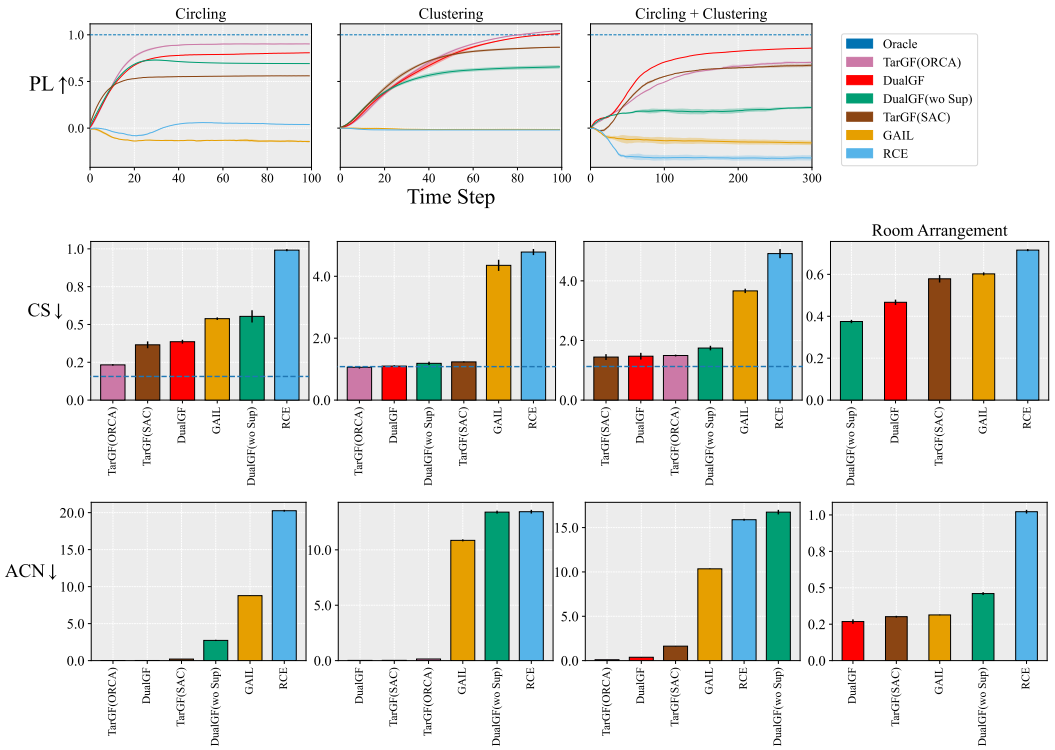


Figure 5: Comparative Results on *Ball Rearrangement* and *Room Rearrangement*. For each metric, the first three columns demonstrate the results on *Ball Rearrangement*. The CS and ACN results on *Room Rearrangement* are plotted in the last column.

5.2 Evaluation

We collect trajectories over 5 random seeds for evaluation. For each random seed, we collect 100 trajectories for *Navigation*, *Tracking*, and *Ball Rearrangement* and 8×83 (for each unseen room, we collect 8 trajectories from different initial states) trajectories for *Room Rearrangement*. We calculate the following metrics on the trajectories (for more details, we defer to Appendix. B):

Task Reward (TR) calculates the averaged cumulative reward of an episode : $\mathbb{E}[\sum_{\mathbf{s}_t \in \tau} r_t]$

Pseudo-Likelihood (PL) measures the similarity between a given state and a target distribution by assigning a *pseudo-likelihood function* $\mathbf{F}_{proxy} : \mathcal{S} \to \mathbb{R}^+$. At each time step t, the PL-curve reports the averaged PL across all trajectories $\mathbb{E}[\mathbf{F}_{proxy}(\mathbf{s}_t)]$.

Coverage Score (CS): measures the diversity and fidelity of the terminal states $S_T =$. CS reports the Minimal-Matching-Distance(MMD) (Achlioptas et al., 2018) between terminal states of a method S_T and a fixed set of examples S_{gt} from p_{tar}: $\sum_{\mathbf{s}_{gt} \in S_{gt}} \min_{\mathbf{s}_T \in S_T} ||\mathbf{s}_{gt} - \mathbf{s}_T||$.

Averaged Collision Number (ACN) reports the averaged collision number $\mathbb{E}[\mathbf{c}_t]$ at each time step over the trajectories: $\mathbf{c}_t = \sum r_{i,j}$, where $r_{i,j}$ equals to 1 only if i-th and j-th object collides.

5.3 Baselines

We compare our framework (*i.e.*, *DualGF*) with the following learning-based baselines and planning-based baselines. The implementation details are deferred to Appendix. C.2.

Leaning-based Baselines: *TarGF(SAC):* is the learning-based framework of TarGF (Wu et al., 2022), where a residual policy built upon the target gradient field is learned to avoid collisions. *GAIL:* A classical inverse RL method that trains a discriminator as the reward function. This method trains two classifiers from target and support examples, respectively. *RCE:* The SOTA example-

Table 1: Comparative results on *Navigation* and *Tracking*. The obstacles is always fixed in *Navigation (Static)* while changes for each episode in *Navigation (Dynamic)*.

		Navigation (Static)		Navigation (Dynamic)		Tracking	
		TR ↑	ACN ↓	TR ↑	ACN ↓	TR ↑	ACN ↓
Learning Based	RL (SAC)	-59.4 ± 2.8	8.9 ± 0.4	-74.1 ± 0.8	38.1 ± 0.0	-91.9 ± 1.6	37.3 ± 1.2
	GAIL	-102.5 ± 2.6	27.3 ± 2.8	-89.8 ± 1.8	37.5 ± 2.4	-91.4 ± 1.6	39.2 ± 1.8
	RCE	-65.8 ± 6.3	43.9 ± 1.1	-69.5 ± 3.8	40.8 ± 1.7	-71.3 ± 7.7	39.9 ± 3.0
	TarGF (SAC)	-126.3 ± 0.0	25.2 ± 3.0	-98.4 ± 0.7	30.0 ± 0.9	-99.8 ± 5.7	8.6 ± 0.1
	DualGF (wo Sup)	-98.0 ± 1.3	75.2 ± 1.3	-84.8 ± 2.6	68.3 ± 3.2	-89.0 ± 1.2	62.9 ± 2.4
	DualGF	**-26.7 ± 1.7**	**4.0 ± 0.8**	**-33.2± 0.2**	**9.4 ± 0.1**	**-45.4 ± 0.2**	**4.1 ± 0.7**
Planning Based	PRM (Reference)	-20.2 ± 2.0	8.5 ± 2.2	-19.7 ± 1.6	6.3 ± 2.1	-42.6 ± 0.5	9.7 ± 0.9

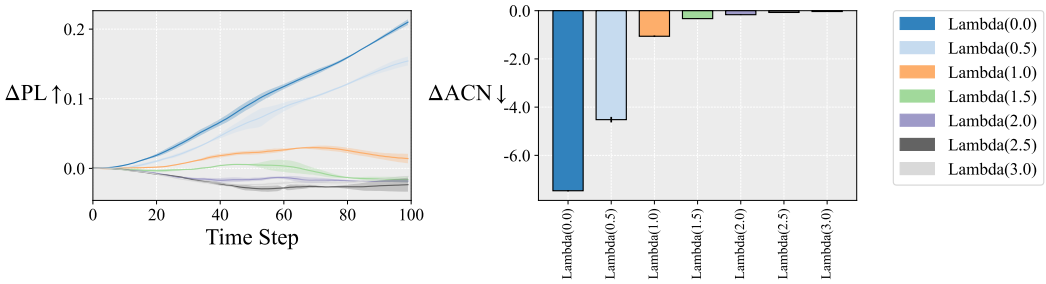


Figure 6: Ablation results on *Clusting*, where ΔMetric = Metric(DualGF) - Metric(Ablated).

based reinforcement learning method. *RL(SAC):* The agent is trained under ground truth task reward and collision signals.

Planning-based Baselines: *TarGF(ORCA):* is the model-based framework, serves as *a reference* of *Ball Rearrangement* since it requires ground truth model. *Probabilistic Road Map (PRM):* is a classic sampling-based method serving as *a reference* of *Navigation*.

Heuristic-based Baseline: *Oracle:* We perturb the target examples via a small Gaussian noise to obtain the terminal states at an oracle level.

6 Experiment Results

6.1 Baseline Comparison

As seen from the quantitative result in Tab. 1 and Fig. 5, *DualGF* significantly outperforms baselines and achieves comparable performance with reference methods such as *TarGF(ORCA)* and *PRM*:

In *Navigation and Tracking*: 1) *DualGF* outperforms learning-based baselines with a large margin. The classical planning algorithm *PRM* achieves strong results in task reward (TR) since it can search for a near-optimal path without any collision to reach the goal in most cases. However, even if we have provided exhaustive sampling budget (*i.e.*, 2000) for *PRM* in each episode, *PRM* still fails to search for a solution in some cases. As a result, *DualGF* outperforms *PRM* in ACN on *Navigation (Static)* and *Tracking* due to the failure cases of *PRM*. 2) Notably, the *PRM* is of poor time efficiency compared with *DualGF*. As shown in Tab. 2, *PRM* takes about 6 times of *DualGF* in *Navigation* and 110 times in *Tracking* since *PRM* needs to search a path again when the agent reaches the goal. 3) The *Navigation and Tracking* is non-trivial. The vanilla reinforcement learning approach *RL(SAC)* ranks the second in the simplest task *Navigation (Static)*, yet it fails to keep the advantage in more challenging tasks, *i.e.*, *Navigation (Dynamic)* and *Tracking*.

	Milliseconds per Step	
	Navigation	Tracking
DualGF	**3.3 ± 0.2**	**3.4 ± 0.2**
PRM	19.7 ± 2.3	376 ± 35.9

Table 2: The inference time per step averaged over 100 trajectories.

In *Ball Rearrangement*: 1) As Fig. 5 illustrated, *DualGF* achieves the best performance compared with learning-based baselines in PL and ACN. In the hardest task *Circling + Clustering*, *DualGF* even outperforms than *TarGF(ORCA)* that plans with the ground truth model. These demonstrate the effectiveness of our method to scale to high-dimensional tasks. 2) The CS of *DualGF* is comparable with other competitive baselines such as *TarGF(SAC)*. Hence, our method is not trivially overfitting to a single mode. 3) The classifier-based methods (*i.e.*, *GAIL* and *RCE*) fail due to the training collapse of the classifier. In practice, we observe that classifier-based baselines achieve appealing results during the training process yet fail to converge at a high-performance level due to the over-exploitation of classifier.

In *Room Rearrangement*: 1) *DualGF* achieves the best performance compared with baselines in both CS and ACN. This indicates that our method can generalise well to the diverse conditional attributes such as different size, number and composition of objects. 2) The planning-based methods either suffer from scalability issues that *PRM* and Rapidly-exploring Random Trees (RRT) (LaValle et al., 1998a) cannot search for a feasible solution in 10 minutes. Besides, *TarGF(ORCA)* assumes the agent is circular in shape which is infeasible in *Room Rearrangement*. Hence, we do not compare them in *Room Rearrangement*.

6.2 Ablation Studies and Analysis

We further conduct ablation studies to analysis the effectiveness of the key components of our method, *i.e.*, the *support gradient field* and the *field-balancing mechanism*. We defer futher ablations on key hyperparameters (*i.e.*, initial λ and c') to Appendix D.1. For more accurate numerical analysis, we conduct experiments on *Clustering* since the pseudo likelihood on this task is cloest to the ground truth as discussed in Appendix. A.

Effectiveness of the Support Gradient Field: 1) The task performance of *DualGF wo Sup* is worse than *DualGF* in all tasks across both *Navigation and Tracking* and *Ball Rearrangement*, where *Ours wo Sup* is the ablated version of *DualGF* that fixes the mixing rate to zero $\lambda \equiv 0$. Besides, the ACN of *DualGF* is significantly lower than *DualGF wo Sup* These results indicate that the support gradient field can help avoid collision effectively and further improve the task performance. 2) In *Room Rearrangement*, *DualGF wo Sup* achieves better CS than *DualGF*, yet performs worse than baselines in ACN. Meanwhile, *DualGF* outperforms the baselines in both CS and ACN. This indicates support gradient field helps to find a better trade-off between task completion and collision avoidance.

Effectiveness of Field-balancing Mechanism: We illustrate the advantage of *DualGF* over *DualGF(Fixed)* on PL and ACN on Fig. 6, where *DualGF(Fixed)* is the ablated version of *DualGF* with mixing rate lambda is fixed at an initial level. With the initial lambda decrease, the advantage of *DualGF* increases significantly in both PL and ACN. This indicates our field balancing mechanism can moderate the sensitivity to the choice of initial mixing rate λ_0.

We visualise the target, support and the dual gradient fields in Fig. 3 to get a better understanding of how DualGF balances task completion and collision avoidance. The results show that *DualGF* establishes a good trade-off between the task completion and collision avoidance. More visualisations are demonstrated on our website sites.google.com/view/dualgf.

7 Conclusion

In this study, we first reformulate the path planning problem in a novel data-driven paradigm where the agent learns to plan with two sets of examples in an offline manner. We further propose a novel score-based planning framework (DualGF) to tackle this problem. There are two gradient fields in DualGFs: a *target gradient field* that guides task completion and a *support gradient field* that ensures moving with physical constraints. Moreover, an adaptive gradient balance mechanism is introduced to combining the merits of the two fields together. Experiments demonstrate the scalability of our method, and empirical results show that it significantly outperforms the state-of-the-art methods in efficiency and safety.

Limitations and Future Work: This work does not consider the complex dynamics where the heuristic-based controller cannot handle. In the future, we may extend our framework with a co-trained inverse dynamics model to enlarge the scope of our applications. In the future, it is also necessary to extend the framework for vision-based control and the embodied robot.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Qisen Chai, Yujun Wang, Yong He, Caijing Xu, and Zhisong Hong. Improved prm path planning in narrow passages based on pso. In *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 41–46. IEEE, 2022.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Andrew Dobson and Kostas E Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *The International Journal of Robotics Research*, 33(1):18–47, 2014.

Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. *arXiv preprint arXiv:2103.12656*, 2021.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10933–10942, 2021.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *arXiv preprint arXiv:1805.11686*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39(5):543–567, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Kyle Hatch, Tianhe Yu, Rafael Rafailov, and Chelsea Finn. Example-based offline reinforcement learning without rewards. *Proceedings of Machine Learning Research vol*, 144:1–17, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Matthias Hüppi, Luca Bartolomei, Ruben Mascaro, and Margarita Chli. T-prm: Temporal probabilistic roadmap for path planning in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, 2022.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Lydia Kavraki and J-C Latombe. Randomized preprocessing of configuration for fast path planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 2138–2145. IEEE, 1994.

Jennifer E King, Marco Cognetti, and Siddhartha S Srinivasa. Rearrangement planning using object-centric and robot-centric action spaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3940–3947. IEEE, 2016.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Geesara Kulathunga. A reinforcement learning based path planning approach in 3d environment. *arXiv preprint arXiv:2105.10342*, 2021.

Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998a.

Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998b.

Chenming Li, Han Ma, Peng Xu, Jiankun Wang, and Max Q-H Meng. Biait*: Symmetrical bidirectional optimal path planning with adaptive heuristic. *arXiv preprint arXiv:2205.06940*, 2022.

Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020.

BK Patle, Anish Pandey, DRK Parhi, A Jagadeesh, et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606, 2019.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. accepted. IEEE, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Marlin P Strub and Jonathan D Gammell. Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3191–3198. IEEE, 2020.

Marlin P Strub and Jonathan D Gammell. Adaptively informed trees (ait*) and effort informed trees (eit*): Asymmetric bidirectional sampling-based path planning. *The International Journal of Robotics Research*, 41(4):390–417, 2022.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Mingdong Wu, Fangwei Zhong, Yulong Xia, and Hao Dong. Targf: Learning target gradient field for object rearrangement. In *Thirty-sixth Conference on Neural Information Processing Systems*, 2022.

Hong You, Gang Chen, Qingxuan Jia, and Zeyuan Huang. Path planning for robot in multi-dimensional environment based on dynamic prm blended potential field. In *2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 5, pp. 1157–1162. IEEE, 2021.

Jinglun Yu, Yuancheng Su, and Yifan Liao. The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Frontiers in Neurorobotics*, 14:63, 2020.

Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1467–1482, 2019.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A TASK DETAILS

A.1 TRACKING AND NAVIGATION

Navigation and Tracking requires the agent to reach a static location or track a dynamic location while avoids collision with the obstacles. There are four rectangular obstacles and a particle agent. The scale, position and orientation of the obstacles are randomised before each episode in both tracking and navigation. The goal location is randomly reset at the start of the episode. Note that in tracking, the goal location will be further reset when reached by the agent. In this task, the target distribution is assigned as a Gaussian distribution centred on the goal location, *i.e.*, $\mathcal{N}(\mathbf{g}, 0.1^2 I)$. The state space is concatenation of agent's position, goal location and geometry observations of obstacles and the action space is two dimensional linear velocity.

State and Action Spaces: $\mathbf{s} = [\mathbf{s}_a, \mathbf{s}_c]$ where the agent state is 2-D location $\mathbf{s}_a = [x, y] \in [-1, 1]^2$ and condition-state is the 2D goal location and obstacles' states $\mathbf{s}_c = [g, o_1, o_2, o_3, o_4]$ where $g \in [-1, 1]^2$ and o_i denotes the state of the i-th obstacle. The obstacle state is the normalised position, orientation and side-length $o_i = [pos, ori, side]$ where $pos \in [-1, 1]^2, ori = [sin(yaw), cos(yaw)] \in [-1, 1]^2, side \in [-1, 1]$.

Horizon: 100.

Initial Distribution: Before each episode, we first randomly initialises 4 box-shaped obstacles and then randomly sample the agent's state that do not collide with the obstacles.

Dynamics: The floor and wall are all absolutely smooth planes. The arena is a 0.25m x 0.25m area, with the agent's radius being 0.015m. We set friction coefficients of all obstacles and agent to be 10.0 to penalise the collision.

Target Examples: We collect 10000 examples for each task as target examples. Each example is of different obstacles and goal.

A.2 BALL REARRANGEMENT

Ball Rearrangement includes three sub-tasks: *Circling*, *Clustering*, and the hybrid of the first two, *Circling + Clustering*. In both sub-tasks, there are three sets of 7 balls that belong to different categories, *i.e.*, red, green and blue. The state and action space is the joint positions and linear velocities of all balls respectively. The agent are required to rearrange all balls to maximise the target likelihood of the joint state while avoiding collisions. The interpretations of target distribution of each task are as follows: *Clustering* requires all balls to form into three clusters by colour; Both *Circling* and *Circling + Clustering* require all balls to form into a circle that can centred in anywhere in the arena, except that the latter further requires that the balls of the same type are adjacent to each other.

State and Action Spaces: There are 21 balls in environment. The state space is the concatenation of all sub-states of all balls: $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_2 1]$. For each ball, the sub-state is $\mathbf{s}_i = [\mathbf{s}_a, \mathbf{s}_c], \mathbf{s}_a = [x, y], \mathbf{s}_c = [c]$ where $x, y \in [-1, 1]$ denotes the two dimensional position and $c \in \{0, 1, 2\}$ is a class label. The action space is also the concatenation of all sub-actions of all balls: $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_2 1]$ where $\mathbf{a}_i \in [-1, 1]^2$ is a 2 dimensional linear velocity.

Horizon: 300 for *Circling + Clustering* and 100 for *Circling* and *Clustering*.

Initial Distribution: We first uniformly sample rough locations for each ball, then we eliminate overlaps between these positions by stepping the physical simulation.

Dynamics: The floor and wall are all absolutely smooth planes. All the balls are bounded in an 0.3m x 0.3m area, with the radius being 0.025m. We set the friction coefficients of all balls to 100.0 since we observe that setting a small(*e.g.*not larger than 1.0) friction coefficient does not significantly affect the dynamics. Besides, to increase the complexity of the dynamics, we set masses and restitution coefficients of all green and blue balls to 0.1 and 0.99, respectively. The masses and restitution coefficients of all red balls are set to be 10 and 0.1, respectively. We observe that under these dynamics, the collision may significantly harm the efficiency of the rearrangement process. Hence, the agent have to adapt to the dynamics for more efficient object rearrangement.

Target Examples: We collect 100,000 examples for each task as target examples.

A.3 Room Rearrangement

Room Rearrangement is built on a large-scale, synthetic indoor scene dataset 3D-FRONT (Fu et al., 2021). Following (Wu et al., 2022), we use 756 of 839 rooms for training and 83 for testing. The agent is required to learn scene priors from the target examples (*i.e.*, training set) and rearrange objects into reasonable layout during the test phase. The state space is concatenation of observations of each object and the room boundary information. The action space is concatenation of two dimensional linear velocity and one dimensional angular velocity of each object.

Dataset and Simulator: We clean the 3D-Front dataset Fu et al. (2021) to obtain bedrooms that consist of four walls and three to eight objects. We augment each room by flipping two times and rotating four times to get eight augmentations. We then import these rooms into iGibson Shen et al. (2021) to run the physical simulation. The 756x8 rooms are used for *target examples* that are used for training of the target score network, the classifier-based baselines and the VAE in goal-conditioned baselines. The other 83x8 rooms are used to initialise the room in the test phase, so the room rearrangement task is performed on the test dataset. The rooms in the training dataset are only used for learning the prior knowledge to arrange the room.

State and Action Spaces: The state consists of a aspect ratio $r_a \in \mathbb{R}^+$ and an object state $\mathbf{s}_o \in \mathbb{R}^{K\times 6}$ where K denotes the number of objects. The aspect ratio $r_a = \tanh(\frac{b_x}{b_y})$ where b_x and b_y denotes the horizontal and vertical wall bounds. The object state is the concatenation of sub-states of all the objects $\mathbf{s}_o = [\mathbf{s}^1, \mathbf{s}^2, ...\mathbf{s}^i, ..., \mathbf{s}^K]$ where the sub-state of the i-th object $\mathbf{s}^i \in \mathbb{R}^6$ is consists of 2-D position, 1-D orientation, 2-D bounding bound and a 1-D category label. The action is also a concatenation of sub-actions of all the objects $\mathbf{a}_o = [\mathbf{a}^1, \mathbf{a}^2, ...\mathbf{a}^i, ..., \mathbf{a}^K]$. For the i-th object, the action $\mathbf{a}^i \in \mathbb{R}^3$ consists of a 2-D linear and an 1-D angular velocity. The whole action space is normalised into a $3 \times K$ dimensional unit-box $[-1, 1]^{3\times K}$ by the velocity bounds. Note that the agent-state $\mathbf{s}_a$ in this task is the concatenation of agents' positions and orientations while the condition-state $\mathbf{s}_c$ in this task is the aspect ratio and the concatenation of agents' labels and 2-d bounding boxes.

Horizon: Each training episode contains 100 steps, instead of 100 steps used in Wu et al. (2022).

Initial Distribution: To guarantee the initial state is accessible to the high-density region of the target distribution, we sample an initial state in two stages: First, we sample a room from the 83x8 rooms in the test dataset. Then we perturb this room by 1000 Brownian steps.

Dynamics: For more efficient environment reset and physical simulation, we build a 'proxy simulator' based on PyBullet Coumans & Bai (2016–2021) to replace the iGibson simulator. We use iGibson to load and save the metadata of each room. Then we reload these rooms in the proxy simulator, where each object is replaced by a box with the same geometry. We set the friction coefficient of all the objects in the room to zero, as the dynamics of the room are complex enough.

B Metric Details

B.1 Task Reward(TR)

This is specifically used in *Tracking* and *Navigation*, since these tasks have naturally well-defined reward function (*e.g.*, goal distance). It calculates the averaged cumulative reward of an episode : $\mathbb{E}[\sum_{\mathbf{s}_t \in \tau} r_t]$

B.2 Pseudo-Likelihood (PL)

measures the similarity between a given state and a target distribution by assigning a *pseudo-likelihood function* $\mathbf{F}_{proxy} : \mathcal{S} \to \mathbb{R}^+$. At each time step t, the PL-curve reports the averaged PL across all trajectories $\mathbb{E}[\mathbf{F}_{proxy}(\mathbf{s}_t)]$. Hence, PL-curve indicates the efficiency and final performance of each method. We do not report the PL-curves on *Room Rearrangement*, since it is hard to describe the human preferences and scene priors by hand-engineering.

The PL curve is only reported in *Ball Rearrangement*. The specific pseudo likelihood functions are listed as follows:

Circling: The pseudo likelihood function is defined as $\mathbf{F}_{proxy}(\mathbf{s}) = \exp^{-(\sigma_\theta+\sigma_r)}$, where σ_θ and σ_r denote the standard deviation of the angle between two adjacent balls and the distances from each ball to the centre of gravity of all balls, respectively. Intuitively, if a set of balls are arranged into a circle, then the σ_r and σ_θ should be close to zero, achieving higher pseudo likelihood.

Clustering: Different from (Wu et al., 2022), our target distribution is a six-mode Gaussian instead of a two-mode Gaussian:

Defining the joint centres' positions as a latent variable $C = (C_r, C_g, C_b)$ where C_r, C_g and C_b denotes centres of red, green and blue balls respectively and the above six modes as $\{c_i\}_{1\leq i\leq 6}$, the C obeys a categorical distribution $p(C = c_i) = \frac{1}{6}$. The pseudo likelihood function is a Gaussian Mixture Model:

$$p_{GMM}(\mathbf{s}) = \sum_{1\leq k\leq 6} p(C = c_k) \prod_{1\leq i\leq \frac{K}{3}} \mathcal{N}(C_r, 0.05I)(\mathbf{s}^i) \prod_{\frac{K}{3}\leq i\leq \frac{2K}{3}} \mathcal{N}(C_g, 0.05I)(\mathbf{s}^i) \prod_{\frac{2K}{3}\leq i\leq K} \mathcal{N}(C_b, 0.05I)(\mathbf{s}^i) \tag{15}$$

Circling+Clusterng: The pseudo likelihood function is defined as $\mathbf{F}_{proxy}(\mathbf{s}) = \exp^{-(\sigma_\theta+\sigma_r)} \cdot \exp^{-(\sigma_R+\sigma_G+\sigma_B)-\sigma_C}$ where σ_R denotes the standard deviation of the angle between two adjacent red balls, and σ_G and σ_B and σ_C denotes the standard deviation of the positions of red, green and blue centres. Intuitively, the first term $\exp^{-(\sigma_\theta+\sigma_r)}$ measures the pseudo likelihood of balls forming a circle and the next term $\exp^{-(\sigma_R+\sigma_G+\sigma_B)+\sigma_C}$ measures the pseudo likelihood of balls being clustered into three piles.

B.3 Coverage Score (CS):

CS measures the diversity and fidelity of the terminal states $S_T =$. CS reports the Minimal-Matching-Distance(MMD) (Achlioptas et al., 2018) between S_T and a fixed set of examples S_{gt} from p_{tar}: $\sum_{\mathbf{s}_{gt}\in S_{gt}} \min_{\mathbf{s}_T\in S_T} ||\mathbf{s}_{gt} - \mathbf{s}_T||$. This metric can detect mode-collapsing: If an agent only moves objects to a single mode, the terminal states may far away from other modes which leads to high value of CS (*i.e.*, bad performance).

For Ball Rearrangement: To calculate the coverage score, we sample fixed sets examples from the target distribution serving as S_{gt} for *Circling*, *Clustering*, and *Circling+Clustering*, respectively. We sample 20 examples for for *Circling* and *Circling+Clustering* and 50 examples for *Clustering*. Since the balls in the same category can actually be viewed as a two-dimensional point cloud, we measure the distance between two states by summing the CDs between each pile of balls by category.

For Room Rearrangement: The coverage score is calculated by averaging the coverage score in each room condition, since the state dimension differs in different rooms. For each room in 83 test rooms, we calculate the coverage score between the eight ground truth states and eight rearrangement results and then the averaged coverage score over the 83 rooms is taken as the final coverage score for a method. We measure the distance between two states by calculating the average L2-distance between the positions of the corresponding objects.

B.4 Averaged Collision Number (ACN)

ACN reports the averaged collision number $\mathbb{E}[\mathbf{c}_t]$ at each time step over the trajectories: $\mathbf{c}_t = \sum r_{i,j}$, where $r_{i,j}$ equals to 1 if i-th and j-th object collides and 0 otherwise.

C Implementation Details

For all learning-based baselines, we implement all the methods including ours in the same network architecture design and basically the same capacity (*e.g.*, hidden dimensions).

C.1 DUALGF

Training Objective: The complete training objective for both target and support networks is the SDE-based score-matching objective proposed by Song et al. (2020):

$$\mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{\mathbf{s}(0)\sim p_{data}(\mathbf{s})}\mathbb{E}_{\mathbf{s}(t)\sim p_{0t}(\mathbf{s}(t)|\mathbf{s}(0))}[\mathbf{\Phi}_\theta(\mathbf{s}(t),t)-\nabla_{\mathbf{s}(t)}\log p_{0t}(\mathbf{s}(t)\mid\mathbf{s}(0))\|_2^2]. \tag{16}$$

where $p_{0t}(\mathbf{s}(t)\mid\mathbf{s}(0))=\mathcal{N}(\mathbf{s}(t);\mathbf{s}(0),\frac{1}{2\log\sigma}(\sigma^{2t}-1)\mathbf{I})$ and $\sigma=25$ is a hyper-parameter.

Using this objective, we can obtain the estimated score *w.r.t.* different levels of the noise-perturbed target distribution $p^t_{data}(\mathbf{s}(t))=\int p_{0t}(\mathbf{s}(t)\mid\mathbf{s}(0))p_{data}(\mathbf{s}(0))d\mathbf{s}(0)$simultaneously. In this way, we can assign different noise-level t for different tasks conveniently. For all tasks, our method chooses $t=0.01$.

Network Architecture: In *Navigation* and *Tracking*, our score networks are implemented as fully connected layers with hidden size equals to 256. In *Ball Rearrangement* and *Room Rearrangement*, we preprocess the state as a fully connected graph that initialises the nodes by states of each object. The score networks pass the input graph through two-layers of FC for feature extraction and two edge-convolution layers for message passing to obtain the output score.

C.2 BASELINES

C.3 TARGF

The training objective of TarGF is same as Eq. 16. And we set $t=0.01$ for all experiments for fair comparison.

For TarGF(SAC): The implementation is based on the codes released by authors, except that we set $t=0.01$ for both reward estimation and residual policy for fair comparison.

For TarGF(ORCA): We set $\tau=0.1$ and the simulation duration of each timestep $\Delta t=0.02$. For each agent(object), ORCA only considers the 2-nearest agents as neighbours, since in our experiments ORCA often has no solution when the number of neighbours is larger than 2.

C.4 CLASSIFIER-BASED BASELINES

These baselines refer to the *RCE* and *GAIL* in experiments.

RCE and is implemented based on the codes Eysenbach et al. (2021) released by RCE's authors. We only modify $\gamma=0.95$, the training steps decrease to 0.5 million (*i.e.*, the same number of training steps as other methods) and the model architecture. The architecture of actor and critic networks is implemented the same as ours(*i.e.*, the same feature extraction layers and edge-convolutional layers).

GAIL trains two classifiers by discriminating two example sets with agent's rollouts respectively. The discriminator takes only one state instead of two adjacent states as input, since the 'ground truth' reward(*i.e.*, likelihood) is defined on the current state. At each training step, we update the discriminator by distinguishing between the agents' and the expert's states(for one step) and then update the RL policy under the reward given by the discriminator(for one step).

C.5 RL(SAC) IN NAVIGATION AND TRACKING

Is implemented from GAIL that replaces the classifiers with ground truth task rewards.

C.6 PRM

We implement PRM from an off-the-shelf open-sourced repository PythonRobotics Sakai et al. (2018) with 16.3k stars. This implementation is also used for navigation task. We keep the default configuration and properly rescale our configurations to fit the API.

D ADDITIONAL ANALYSIS

In this section, we seek to evaluate the sensitivity of our method to the two key hyperparameters, *i.e.*, λ_0 and c'. The experiments are conducted on *Clustering* for better quantitative evaluation.

D.1 SENSITIVITY OF MIXING RATE λ

We evaluate *DualGF* with different levels of the initial lambda λ_0. As shown in Fig. 7, the PL and CS do not significantly change when the $\lambda \geq 1.0$. This indicates our method is not sensitive to λ when it is above a positive threshold.

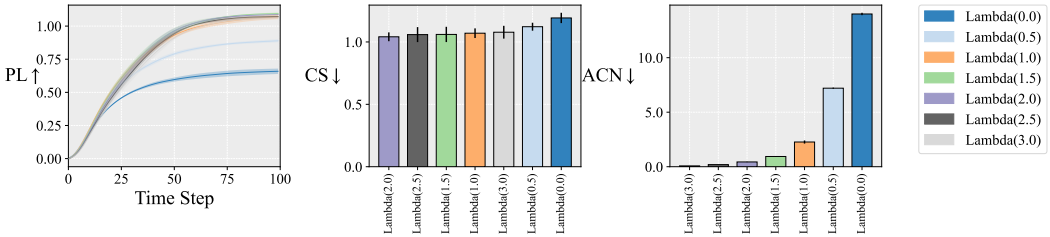


Figure 7: Ablation results on *Clusting*.

D.2 SENSITIVITY OF THRESHOLD c'

We evaluate *DualGF* with different levels of the initial threshold c'. As shown in Fig. 8, the PL and CS do not significantly change with all different levels of threshold but the ACN becomes better with the increase of the theshold. This indicates our method is not sensitive to c'.

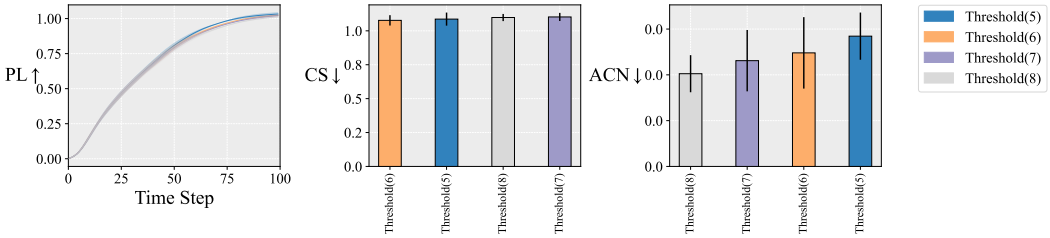


Figure 8: Ablation study results on *Clusting*. We fix the threshold $c' = 8$ and evaluate DualGF with different initial lambda.