Versatile Navigation under Partial Observability via Value-guided Diffusion Policy

Gengyu Zhang[1] **Hao Tang**[2] **Yan Yan**[1,†]
[1]Department of Computer Science, Illinois Institute of Technology, USA
[2]Robotics Institute, Carnegie Mellon University, USA
gzhang32@hawk.iit.edu, bjdxtanghao@gmail.com, yyan34@iit.edu

Abstract

Route planning for navigation under partial observability plays a crucial role in modern robotics and autonomous driving. Existing route planning approaches can be categorized into two main classes: traditional autoregressive and diffusion-based methods. The former often fails due to its myopic nature, while the latter either assumes full observability or struggles to adapt to unfamiliar scenarios, due to strong couplings with behavior cloning from experts. To address these deficiencies, we propose a versatile diffusion-based approach for both 2D and 3D route planning under partial observability. Specifically, our value-guided diffusion policy first generates plans to predict actions across various timesteps, providing ample foresight to the planning. It then employs a differentiable planner with state estimations to derive a value function, directing the agent's exploration and goal-seeking behaviors without seeking experts while explicitly addressing partial observability. During inference, our policy is further enhanced by a best-plan-selection strategy, substantially boosting the planning success rate. Moreover, we propose projecting point clouds, derived from RGB-D inputs, onto 2D grid-based bird-eye-view maps via semantic segmentation, generalizing to 3D environments. This simple yet effective adaption enables zero-shot transfer from 2D-trained policy to 3D, cutting across the laborious training for 3D policy, and thus certifying our versatility. Experimental results demonstrate our superior performance, particularly in navigating situations beyond expert demonstrations, surpassing state-of-the-art autoregressive and diffusion-based baselines for both 2D and 3D scenarios.

1. Introduction

Navigation is a critical component in mobile robotics and autonomous driving dependent on sequential planning, a process of evaluating and selecting an action sequence that most effectively achieves a specific goal. However, traditional autoregressive planning methods for navigation, as mentioned in [11, 14, 31], face two significant limitations. First, they select actions sequentially, where each decision is based on the previous one and the consequent state transitions. This step-by-step approach is not well-suited for tasks with longer horizons, as it lacks foresight. The problem worsens in partially observable settings, where increased uncertainty introduces greater computational demands to solve complex mathematical frameworks [13, 28]. Additionally, the necessity for instantaneous decision-making in real-time navigation can be at odds with the once-per-step inference rate of traditional planning methods. Second, these methods often require a substantial volume of data to learn effective policies for 3D navigation. In practice, however, gathering large datasets can be impractical due to environmental, logistical, or equipment constraints, resulting in a limited set of offline data. When faced with such data deficiency, traditional methods tend to yield suboptimal performance.

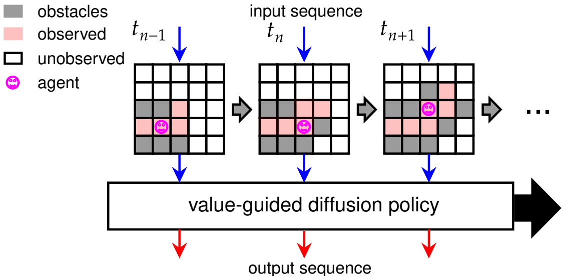

Figure 1. Our value-guided diffusion policy under partial observability. It processes local partial observations to generate action sequences adaptable for both 2D and 3D scenarios.

To overcome the limitations carried by autoregressive planning, we explore trajectory-level behavior synthesis. This novel approach capitalizes on the capabilities of generative models, particularly diffusion models [1, 4, 12, 17, 19, 24]. Unlike autoregressive methods that generate actions sequentially, diffusion-based approaches synthesize entire action trajectories simultaneously, enhancing multi-step planning efficiency during inference. However, to our best knowledge, no existing work of this class has actively explored their effectiveness under partial observability. Thus, through a significant modification, we adapt this concept for

[†] Corresponding author

use in partially observable environments. We model the navigation with an approximated partially observable Markov decision process (POMDP). This involves embedding a state estimation module in training a differentiable planner, which learns a value function to guide the agent's policy planning. This value function is derived by estimating the underlying decision model of expert demonstrations during training and iteratively computing optimal values during inference. Fig. 1 illustrates the plan generation module, in which the diffusion policy generates a plan for certain future timesteps in a closed-loop manner conditioned on the observed partial environment map. A plan in this context is essentially an action trajectory — a series of temporally sequential actions derived from a specific policy and state transition dynamics. The value function demonstrated in Fig. 3 and Fig. 4 ensures that the generated plans lead to at least near-optimal outcomes through trajectory optimization.

To overcome the challenge of data scarcity in 3D realistic navigation scenes, we propose adapting inputs into a format amenable to models trained on 2D data, allowing us to apply policies learned from the 2D domain to navigate in 3D environments. This is predicated on the abundance of 2D data, which ensures that a robust policy for the 2D domain can be learned. By constructing a point cloud from first-person-view (FPV) RGB-D inputs and transforming it to meet 2D standards, we can preserve the performance of the 2D policy in the 3D navigation. This transformation involves semantic segmentation of the point cloud using pre-trained models [5, 35], followed by projecting it onto a bird-eye view (BEV) grid map based on the result of segmentation. Consequently, the high-dimensional RGB-D inputs are converted into grid maps, serving as the basis for our diffusion policy to generate action plans. The 2D policy, already trained, can then infer actions for 3D scenes.

We evaluate the efficacy of our method with two established frameworks: an autoregressive planner, CALVIN [11], and a diffusion-based behavior cloner, Diffusion Policy [4]. Extensive experiments demonstrate the superiority of our method over these baselines in 2D mazes and real-world 3D indoor navigation scenes. Notably, the policy trained on 2D mazes is directly applicable to 3D settings by projecting the point cloud to 2D BEV plane, showing impressive scalability without additional training while still maintaining performance on par with the baselines. Further enhancements include training the model with both BEV grid maps converted from point cloud and egocentric RGB images, with supervision from a limited set of expert demonstrations in point cloud navigation. This dual-conditioning approach has been shown to boost the effectiveness of the policy.

2. Related Work and Preliminary

Differentiable planning. The concept of deep differentiable planning, a method that facilitates online plan generation and backpropagation of errors through these plans to train transition and reward estimators, was initially introduced by Value Iteration Networks (VINs) [22, 26, 31]. This approach is often employed in offline reinforcement learning [16] and imitation learning where data is limited. Subsequent works adapted VINs to partially observable scenarios, assuming a complete environmental map for localization tasks [14], and substituted the max pooling operation, which in VINs realizes the maximization in the Bellman equation of MDPs, with an LSTM structure. A recent enhancement to VINs introduced an additional mask for the explicit exclusion of invalid actions, thereby preventing collisions and allowing for more effective long-horizon navigation [11].

Diffusion models and diffusion policies. Diffusion models, as a prominent class of generative models, formulate the data creation process as an iterative denoising procedure [9, 27]. This approach can be interpreted as the parameterization of the gradients of data distribution [15, 29, 30], thereby linking diffusion models with score matching [10] and Energy-Based Models (EBMs) [7, 23]. The iterative, gradient-based sampling is particularly conducive to flexible conditioning [6, 21] and compositionality [8]. This led to the emergence of a promising new category of methods that harness the potential of diffusion models to extract effective behaviors from heterogeneous datasets and plan for unobserved scenarios during the training phase. Some of these approaches focus on the practical application of diffusion models for control policy behavioral cloning [4] or diffusion policy analysis in simulated environments [24]. Others explore the use of diffusion models in planning contexts, integrating a value function to facilitate planning for unseen tasks [2, 12, 33]. Researchers also utilize diffusion models in robot learning in conjunction with physics-augmented simulations. This approach is instrumental in designing and developing varied and functional soft robot systems, with an emphasis on their morphology and control mechanisms [32].

Diffusion models posit data generation as an iterative denoising process, $p_\theta(\mathbf{x}_{i-1}|\mathbf{x}_i)$, reversing the forward diffusion process $q(\mathbf{x}_i|\mathbf{x}_{i-1})$ which consistently adds noise to the original data sample. This process is also known as Stochastic Langevin Dynamics [34]. Here, $\mathbf{x}_i$ in the diffusion (forward) and reverse processes denotes the noisy data at the diffusion step i. The target data distribution that DPMs aim to recover from Gaussian noise, along with the corresponding denoising process, are as follows:

$$\begin{aligned} p_\theta(\mathbf{x}_{0:N}) &= p(\mathbf{x}_N)\prod_{i=1}^{N} p_\theta(\mathbf{x}_{i-1}|\mathbf{x}_i), \\ p_\theta(\mathbf{x}_{i-1}|\mathbf{x}_i) &= \mathcal{N}(\mathbf{x}_{i-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_i, i), \boldsymbol{\Sigma}_\theta(\mathbf{x}_i, i)), \end{aligned} \tag{1}$$

where $\mathcal{N}$ signifies a Gaussian distribution with mean $\boldsymbol{\mu}_\theta(\mathbf{x}_i, i)$ and variance $\boldsymbol{\Sigma}_\theta(\mathbf{x}_i, i))$, each is a function of the data sample $\mathbf{x}_i$ and step i. We adopt the notation used in [12] that denotes the number of diffusion steps with N and each step with i, distinguishing from T and t used for planning timesteps. The iterative sampling process of diffusion models facilitates flexible conditioning, allowing auxiliary guides to adjust the sampling procedure to retrieve trajectories with high returns or satisfy specific constraints. Incorporating trajectory optimization into the modeling process permits diffusion policies to enhance the performance of learned models in decision-making tasks.

Planning under partial observability. POMDP is a widely recognized mathematical framework for modeling decision-making scenarios with imperfect observation. In such contexts, an agent lacks direct access to the complete information necessary to fully describe the state of the system. A POMDP is formally defined as the tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, \mathcal{O}, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and Ω are the discrete state, action, and observation spaces, respectively. The state-transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, denotes the probability of transitioning from state s to s'. The reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, quantifies the immediate reward gained by executing action a in state s. The observation function, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$, specifies the likelihood of receiving observation o in state s by taking action a. Lastly, γ is the discount factor used in the Bellman equation for iterative optimal value derivation.

Under partial observability, since the agent cannot directly observe the underlying physical state, it instead maintains a probability distribution over $\mathcal{S}$, namely the belief state, that indicates its confidence about which state it is in. Over time, the belief state is updated in a Bayesian manner following Eq. (2), where it is first updated by action a and transition dynamics $\mathcal{T}$, and then corrected by observation o' and observation function $\mathcal{O}$. η is the normalization factor. This procedure is also called the state estimation.

$$b(s') = \eta \mathcal{O}(s', a, o') \sum\nolimits_{s \in \mathcal{S}} \mathcal{T}(s'|s, a) b(s), \qquad (2)$$

However, this presents a significant computational challenge for the optimal policy derivation of POMDPs. Consider a system with n physical states; the policy π must be defined across a $(n-1)$-dimensional continuous belief space, making it prohibitively expensive to solve by standard value or policy iteration. This challenge, known as the *curse of dimensionality*, is one of the two primary factors that contribute to the computational intractability of solving POMDPs exactly [25]. The other factor, termed the *curse of history*, arises from the exponential growth in the number of distinct action-observation histories to be evaluated for policy optimization as the planning horizon extends.

To mitigate these challenges, we adopt QMDP [18, 25], a heuristic that offers an approximate solution to POMDPs, effectively addressing both the curse of dimensionality and the curse of history. QMDP employs a simplified model that considers partial observability at the current planning step but assumes full observations for subsequent steps, which reduces computational complexity while still accounting for the uncertainty, thus offering a computationally efficient, approximate solution scaling to larger problems.

QMDP obtains the optimal Q function by solving the corresponding fully observable MDP via iterating the following Bellman equation until convergence.

$$Q^{k+1}(s, a) = \mathcal{R}(s, a) + \gamma \sum\nolimits_{s'} \mathcal{T}(s'|s, a) \max_{a'} Q^k(s', a'), (3)$$

where $k \in [1, K]$ denotes the current iteration round. Finally, we obtain the QMDP policy by:

$$\pi(b) = \arg\max_a \sum\nolimits_s Q^K(s, a) b(s). \qquad (4)$$

3. Methodology

This section introduces our novel navigation framework, which harnesses diffusion models for generating action trajectories in complex, partially observable environments. This framework comprises 1) the diffusion policy module and 2) the value network. The diffusion policy module, outlined in Sec. 3.2, lies in generating plans based on partial environment maps, enhancing the agent's decision-making as it gathers more environmental data. Our closed-loop planning process, underpinned by receding horizon control, ensures a smooth and coherent action trajectory formulation. To address the limitations of behavioral cloning in dynamic settings, we incorporate the value guidance as detailed in Sec. 3.3. This enhancement, critical in complex environments, drives the agent away from obstacles and dead ends. Our method integrates state estimation with QMDP to approximate the optimal value function and reinforce the policy's efficacy. We train these two modules separately and incorporate them for inference. A unique aspect of our approach, described in Sec. 3.4, involves adapting our robust 2D policy for 3D environments. We transform 3D RGB-D inputs into 2D BEV maps, allowing for a seamless transfer of 2D navigation policy to 3D scenarios. This method overcomes the challenges posed by the deficiency of real-world 3D data, thus facilitating efficient and accurate navigation in various settings.

3.1. Problem Formulation

We aim to address a trajectory optimization problem similar to that in [12] but under partial observability. In a discrete-time control system, where the dynamics are defined as $s_{t+1} = f(s_t, a_t)$ at state s_t and given action a_t, we seek to search for a plan, in the form of an action sequence, $\boldsymbol{a}^*_{0:T}$, that maximizes an objective function $\mathcal{J}$. This objective function is factorized over per-timestep Q values, $Q_t(b_t, a_t)$:

$$\begin{gathered} \mathcal{J}(b_0, \boldsymbol{a}_{0:T}) = \sum_{t=0}^{T} Q_t(b_t, a_t) = \sum_{t=0}^{T} Q_t(s_t, a_t) b_t(s_t), \\ \boldsymbol{a}^*_{0:T} = \arg\max_{\boldsymbol{a}_{0:T}} \mathcal{J}(b_0, \boldsymbol{a}_{0:T}). \end{gathered} \qquad (5)$$

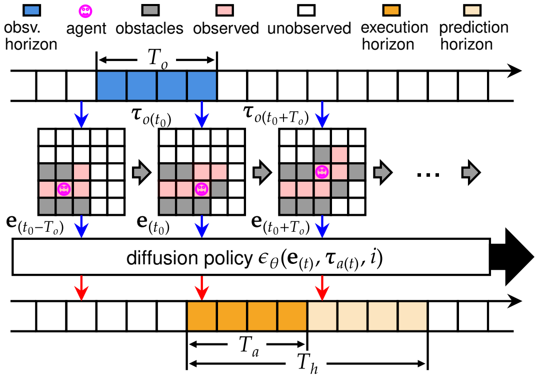

Figure 2. The architecture of diffusion-model-based plan generator. The top sequence represents local observations over time. The grids in the middle form the sequence of cumulative partial maps, which sufficiently encapsulate the agent's long-term memory and environment features. The bottom sequence represents the generated plan in the form of action trajectories. During training, the input of the framework at timestep t consists of the partial map, $\boldsymbol{e}_{(t)}$, and expert action trajectory, $\boldsymbol{\tau}_{a(t)}$; during inference, the input comprises $\boldsymbol{e}_{(t)}$ and a Gaussian noise of the same shape as $\boldsymbol{\tau}_{a(t)}$.

In our model, the belief b_t is updated according to Eq. (2) throughout the planning horizon T. We define the action trajectory at time t as $\boldsymbol{\tau}_{a,(t)} = (a_t, a_{t+1}, \dots, a_{t+T-1})$, which the diffusion model generates, conditioned on the partially observed environment $\boldsymbol{e}_{(t)}$. This environment map, $\boldsymbol{e}_{(t)}$, compiles the trajectory of local observations from timestep 0 to t. Given a previously obtained map $\boldsymbol{e}_{(t-T_o)}$, we have $\boldsymbol{e}_{(t)} = (\boldsymbol{e}_{(t-T_o)}, \boldsymbol{\tau}_{o,(t)})$, where $\boldsymbol{\tau}_{o,(t)} = (o_{t-T_o+1}, o_{t-T_o+2}, \dots, o_t)$ and T_o represent the observation horizon, as illustrated in Fig. 2. In our framework, T_o aligns with T_a, the horizon for action execution, which is further detailed in Sec. 3.2.

3.2. Diffusion-model-based Plan Generation

As depicted in Fig. 2, our framework utilizes a diffusion model to generate action trajectories from timestep t. These trajectories are conditioned on the partial environment map, $\boldsymbol{e}_{(t)}$, which aggregates information observed up to t. As a result, as the agent continues to explore its surroundings, its understanding of the overall environment gradually enhances, facilitating more rational decision-making.

The process of plan generation in our framework operates in a closed-loop manner. In each iteration, we input the partial environment map into the diffusion model. This map, which encapsulates sufficient statistics of the observation history, acts as the key condition, steering the diffusion model's conditional generation process. In a partially observable scenario, the agent uncovers new areas incrementally, gradually removing the mist and enriching the existing map with additional world information. To encourage the temporal coherence and smoothness of formulating action trajectories during planning, we adopt the receding horizon control strategy [20]. In practice, at each timestep t, the policy processes the current partial environment map, $\boldsymbol{e}_t$, forecasts actions for the next T_h steps (the prediction horizon), and then implements the initial T_a steps (the execution horizon) before the next planning cycle, thereby streamlining the decision-making process.

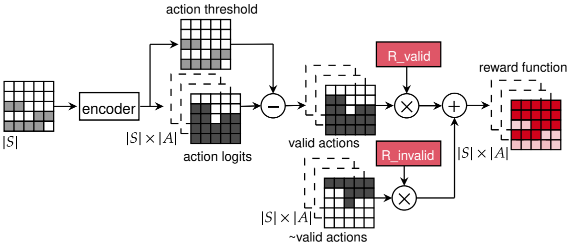

Figure 3. Reward function conditioned on the partial environmental map. The model learns a valid action mask that filters out invalid actions using soft thresholding. This learned embedding is subsequently used to construct the reward function.

We train a diffusion model to learn a robust policy that captures the conditional distribution $p(\boldsymbol{\tau}_{a,(t)}|\boldsymbol{e}_{(t)})$. We formalize the denoising diffusion process, theoretically in the form of $\boldsymbol{\tau}^{i-1}_{a,(t)} \sim p_\theta(\boldsymbol{\tau}^{i-1}_{a,(t)}|\boldsymbol{\tau}^{i}_{a,(t)})$ as

$$\boldsymbol{\tau}^{i-1}_{a,(t)} = \delta(\boldsymbol{\tau}^{i}_{a,(t)} - \alpha\boldsymbol{\epsilon}_\theta(\boldsymbol{e}_{(t)}, \boldsymbol{\tau}^{i}_{a,(t)}, i) + \boldsymbol{\epsilon}^i), \text{where } \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{6}$$

Here, δ denotes the step size, α represents the learning rate, and i is the diffusion step. We employ a mean squared error to account for the loss function:

$$\mathcal{L}_{\mathrm{MSE}}(\theta) = \mathbb{E}_{i,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon}^i - \boldsymbol{\epsilon}_\theta(\boldsymbol{e}_{(t)}, \boldsymbol{\tau}^0_{a,(t)} + \boldsymbol{\epsilon}^i, i)\|^2]. \tag{7}$$

3.3. Value-guided Exploration-safe Planning

Employing only the diffusion plan generator is essentially behavioral cloning [4, 24] under partial observability. This adaptation, however, retains inherent weaknesses in navigating complex and dynamic environments. A notable challenge arises when a policy conditioned on limited environmental data inadvertently leads the agent to a dead end. Since expert demonstrations do not cover such circumstances, the agent might struggle to backtrack and seek alternate paths. This exposes a common downside to diffusion-model-based behavioral cloning methods: a lack of deep environmental understanding.

Incorporating value guidance to direct the agent to the goal while avoiding obstacles presents an effective solution to this challenge. Several studies [2, 12, 33] have explored implementing value guidance in fully observable settings to enhance diffusion policies. To tackle partial observability, we augment one of our baselines [11] by integrating a state estimation module, utilizing QMDP to approximate the optimal value function under partially observable conditions.

Value function with state estimation. The state estimation module implements a Bayesian filter that maps a belief, an action, and an observation to the subsequent belief according to Eq. (2). This module comprises two components: 1) a

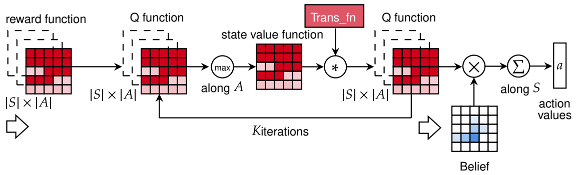

Figure 4. QMDP value iteration module. The learned reward function undergoes K rounds of iterations, consisting of alternating maximization over actions and convolution with the transition function $\hat{T}_m$. The outcome, soft-indexed by the current belief, derives the final action values of this QMDP planner.

prediction module that learns a state transition function that predicts the next belief in a translation-invariant manner for each action, and 2) a belief correction module that weights the updated belief by a jointly learned observation function. For the value function module (Fig. 3), we first train an action validity estimator to explicitly recognize valid and invalid actions for each state. To achieve this, our framework learns the logarithmic probability of each action, $\hat{A}_{\text{logit}}(s,a)$, and an action threshold function, $\hat{A}_{\text{thresh}}(s)$, both conditioned on the partial environment map. Specifically, $\hat{A}_{\text{logit}}$ and $\hat{A}_{\text{thresh}}$ are part of the same learned embedding, comprising $|A|+1$ channels, where $|A|$ denotes the size of the action space. The first $|A|$ channels form $\hat{A}_{\text{logit}}$, while the last channel serves as $\hat{A}_{\text{thresh}}$. Thus, we derive the valid action, $\hat{A}$, by applying a soft-threshold to $\hat{A}_{\text{logit}}(s,a)$ using $\hat{A}_{\text{thresh}}(s)$:

$$\hat{A}(s,a) = \sigma(\hat{A}_{\text{logit}}(s,a) - \hat{A}_{\text{thresh}}(s)),$$

where σ is the Sigmoid function. Once we acquire $\hat{A}$ and its negation, $\sim\hat{A}$, we construct the final reward function by merging them with the separately learned reward parameters: $\hat{R}_m$ for valid actions and $\hat{R}_f$ for invalid ones. In this reward learning framework, $\hat{R}_f$ is typically assigned large negative values, effectively reducing collisions during navigation. We formally define our new reward function as follows:

$$\begin{aligned}\mathcal{R}(s,a) = \hat{R}_f(1-\hat{A}(s,a)) + \\ \hat{A}(s,a)\sum_{s'}\hat{T}_m(s'|s,a)\hat{R}_m(s,a,s'),\end{aligned} \tag{8}$$

where $\hat{T}_m$ estimates the subsection of true state transition pertaining to valid actions. Given this enhanced reward function, the subsequent value iteration module, used to compute the optimal value function, adopts the design as depicted in Fig. 4. After K iterations, the resulting Q function is soft indexed by current belief to derive the approximated QMDP optimal value function, which we use to guide the diffusion policy in inference.

Algorithm 1 Best Plan Candidate Backtracking

```
Require: value function $\mathcal{Q}_{(t)}(s,a;\theta)$, diffusion policy $\boldsymbol{\epsilon}_\theta$, best plan memory $\mathcal{C}$, best plan $\hat{\boldsymbol{\tau}}^*_{a,(t)}$
$\hat{\boldsymbol{\tau}}^*_{a,(t')} \leftarrow$ empty, $\mathcal{C} \leftarrow \emptyset$
for $t = 0, 1, \ldots, T$ do
    $\mathcal{C} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{e}_{(t)}, \boldsymbol{\tau}^N_{a,(t)}, N)$ executed for $L$ times
    if $\hat{\boldsymbol{\tau}}_{a,(t')}$ is not empty then
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\hat{\boldsymbol{\tau}}^*_{a,(t')}\}$
    end if
    $\hat{\boldsymbol{\tau}}^*_{a,(t)} \leftarrow \arg\max_{\boldsymbol{\tau}\in\mathcal{C}} \frac{1}{|\tau|}\sum_{i=0}^{|\tau|-1} Q_{(t)}(s_{\boldsymbol{\tau}_i}, a_{\boldsymbol{\tau}_i};\theta) b_{(t+i)}(s_{\boldsymbol{\tau}_i})$ {Eq. (9)}
    $\Delta t \leftarrow \min(T_a, |\hat{\boldsymbol{\tau}}^*_{a,(t)}|)$
    Execute first $\Delta t$ actions of $\hat{\boldsymbol{\tau}}^*_{a,(t)}$
    Remove first $\Delta t$ actions from $\hat{\boldsymbol{\tau}}^*_{a,(t)}$
    $\hat{\boldsymbol{\tau}}_{a,(t')} \leftarrow \hat{\boldsymbol{\tau}}^*_{a,(t)}$
end for
```

Value-guided plan selection. We use a well-learned value function to guide the diffusion policy. The stochastic nature of diffusion models, stemming from noise sampling, enables us to generate diverse plans given the same conditions. By calculating the sum of action values along each plan, we determine the values of a set of action sequences and select the one with the highest value to execute. This design substantially enhances the navigation's success rate.

However, while the receding horizon control (Sec. 3.2) used in the diffusion plan generation encourages temporal coherence of predicted multi-step plans and strengthens their robustness against latency, it can lead to suboptimal plans. Specifically, when the policy predicts T_h steps of actions and executes the first T_a steps, a left-behind but optimal action a might be overwritten by some suboptimal a' in the re-planning starting from the end of the execution sequence. This issue occurs due to a covariate shift of testing observations from expert demonstration and the diffusion process's stochasticity. To cope with this issue, we propose maintaining a buffer to backtrack the best action trajectory candidates from the past, preserving optimal actions in at least one candidate to avoid inevitable failure. Eq. (9) demonstrates the criterion of selecting the optimal plan at timestep t. In this equation, $\hat{\boldsymbol{\tau}}^*_{a,(t)}$ represents the predicted optimal action trajectory selected from a set of trajectories $\mathcal{C}$, $\hat{\boldsymbol{\tau}}^*_{a,(t')}$ is the best plan candidate selected last time with executed actions removed, where t' is the last timestep a plan is selected. $Q_{(t)}$ refers to the learned Q function at the current timestep. The pseudocode for the backtracking process is provided in Algorithm 1, where N is the total number of diffusion steps for plan generation, and L is the number of candidates to generate each time. Note that we only apply backtracking during inference. Hence, the refined policy becomes:

$$\hat{\boldsymbol{\tau}}^*_{a,(t)} = \underset{\boldsymbol{\tau}\in\mathcal{C}\cup\{\hat{\boldsymbol{\tau}}^*_{a,(t')}\}}{\arg\max} \frac{1}{|\boldsymbol{\tau}|}\sum_{i=0}^{|\boldsymbol{\tau}|-1} Q_{(t)}(s_{\boldsymbol{\tau}_i}, a_{\boldsymbol{\tau}_i};\theta) b_{(t+i)}(s_{\boldsymbol{\tau}_i}). \tag{9}$$

3.4. 2D to 3D Policy Transfer

3D data scarcity poses a significant challenge due to constraints in the real world. Training models on such sparse datasets often leads to overfitting, compromising the ability to generalize to new environments. To circumvent this, we leverage the robust policy developed for the 2D domain,

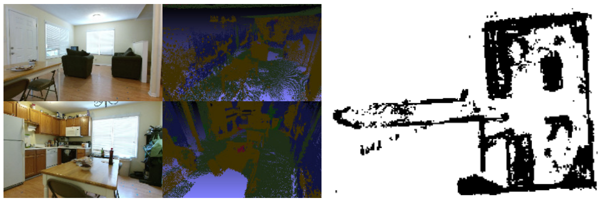

Figure 5. An illustration of constructing a point cloud for a given scene and its subsequent projection onto a BEV map. In this specific example, objects such as the table, chair, and various other furniture pieces in the kitchen, the two sofas and television cabinets in the living room, and the surrounding walls are identified as obstacles on the BEV map. Conversely, areas of the floor that remain uncovered by any objects are designated as free space.

aiming for a zero-shot application to 3D environments. This necessitates transforming 3D inputs into a format compatible with our established 2D policy. In the context of 3D embodied navigation, the agent processes first-person-view RGB-D images. To convert these into 2D BEV maps, we first construct a point cloud from accumulated RGB-D data and then apply pre-trained semantic segmentation models [35] to categorize various elements (Fig. 5). Key labels for crafting the 2D grid map include floors, indicating traversable areas, and walls or furniture, representing obstacles.

We follow a multi-step process to project these segmented components onto a BEV plane. Initially, we trim the point cloud along the Z-axis, which is absent in the BEV representation, to a fixed proportion and perform downsampling. Subsequently, we project the points onto a grid. Whether each cell is classified as free space or an obstacle hinges on the prevalence of points identified as the floor within it. This method allows us to replicate a 2D grid map analogous to those used in our 2D maze experiments, effectively bridging the gap between route planning of 2D and 3D navigation.

4. Experiments

4.1. Task Setups

GridMaze2D. This classic domain provides diverse synthetic environments and tasks for evaluating our method. In this domain, the agent is expected to explore an arbitrary partially observable maze, find the goal, and execute a termination action, *Done*, to finish the current task. If the agent terminates the task at the end of the goal, it successfully completes the mission. Otherwise, running into an obstacle, executing *Done* in the wrong state, or failing to terminate the task all lead to failure. Each environment of this domain corresponds to a unique 2D maze map that presents a BEV of that environment. The observed partial maps are part of the maze map. Please see the Appendix for more details about the composition and generation of partial maps.

The valid action space contains eight directional movements and a termination action, all of which are categorical. Hence, we use the bit encoding technique [3] to convert them into bit arrays for easy retrieval from Gaussian noise. During inference, we decode the sampled action trajectory back into categorical form. Please refer to the Appendix for technical details. The state space consists of the maze's full (X, Y) coordinates.

We simulate navigation in randomly generated mazes to collect expert trajectories. During each simulation, we record the agent's actions, positional coordinates (physical states), and partial environment maps (observation history) at each timestep throughout the trajectory. The expert, equipped with prior knowledge of the goal location, employs an informed search strategy like A^* to navigate toward the goal. Upon gathering sufficient expert trajectories, we partition them into training and validation sets, ensuring that the environments in the validation data remain unseen while training.

Active Vision Dataset. This dataset for 3D embodied navigation enables interactive navigation using real image streams, as opposed to synthetic rendering. AVD consists of 19 indoor environments, densely captured by a robot navigating on a 30cm grid with 30° rotational increments. The comprehensive image set from each scene allows for the simulation of various trajectories with a certain degree of spatial granularity. Additionally, AVD provides bounding box annotations for object instances, a feature we utilize to assess semantic navigation performance. In this domain, the agent's objective is to navigate an indoor environment to locate and reach a specified object.

The action setup is similar to that in the GridMaze2D domain, where the agent has the option to move in any of eight directions. Upon identifying and reaching the target object, the agent must actively execute *Done* command to terminate the current task. This time, we define the state space as the cell coordinates of the BEV map corresponding to each scene. A key difference from the GridMaze2D setup is that actions that lead to collisions with obstacles do not cause instant failure. Instead, the agent remains at the point until it navigates a clear path.

To assess the effectiveness of zero-shot policy transfer from GridMaze2D to the AVD domain, facilitated by point cloud projection, we chose 8 out of the 19 scenes containing a Coca-Cola glass bottle as our validation set. To evaluate CALVIN and our retrained policy with additional RGB inputs, we adopt cross-validation, using one scene for validation and the others for training.

4.2. Result Analysis

For the GridMaze2D domain, we train our model on 15×15 mazes with view range 2. To evaluate robustness against different observability levels, we test the learned policy across three view range settings. To evaluate the generalization capability, we test our model across unseen 15×15, 20×20, and 30×30 mazes. We compare our approach regarding the success rate of completing the navigation task with two

	CALVIN [11]	Diffusion Policy [4]	Ours
15×15 (vr=1)	0.832±0.030	0.024±0.015	**0.886±0.011**
15×15 (vr=2)	0.855±0.030	0.060±0.022	**0.906±0.010**
15×15 (vr=3)	0.900±0.026	0.110±0.031	**0.911±0.013**
20×20 (vr=2)	0.658±0.016	0.012±0.010	**0.713±0.020**
30×30 (vr=2)	0.326±0.030	0.000±0.000	**0.624±0.032**

Table 1. For each method, we train the model on 15×15 mazes with a view range equal to 2 and evaluate in three different maze sizes with three different view range settings. The results demonstrate their scalability to unseen and larger environments. Overall, our approach has better performance.

baseline methods: 1) CALVIN [11], an autoregressive differentiable planner, and 2) Diffusion Policy [4], a diffusion-based behavioral cloner. The results represent the mean and standard deviation across five trials, each encompassing 500 distinct maze simulations. For the AVD, we redeploy a model trained on 30×30 mazes and then transform the input RGB-D images into a point cloud. It is then projected onto a 2D partial environment map in each planning step. To evaluate the pre-trained model's zero-shot transfer to real-world scenes, we only feed it with the partial map, the same as in 2D mazes. To assess the model retrained with RGB-D inputs, we first feed the FPV images into the additional feature extraction module and then concatenate the output embedding with the partial map as the final input to the model. In this setup, we compare our method with CALVIN and its variant that employs our 2D-to-3D policy transfer technique, deriving the mean and standard deviation of 5 trials, each comprising 50 simulations per scene.

We first analyze the overall performance of each methodology regarding success rate in different domains. The numerical results shown in Tab. 1 reveal that CALVIN achieves solid performance with a mean success rate of 0.855 on 15×15 mazes with view range 2 (standard setup), implying that it learns a proficient value function and identifies a near-optimal policy based on it. However, when scaling to larger mazes, the performance of both two variants noticeably declines. The method achieves the best performance in only one test scene. This is likely due to the increased planning horizon. In more expansive environments, the planning horizon extends, requiring more rounds of value iteration to adapt effectively. Nevertheless, since the function approximation deprives the value iteration of its monotonic improvement property, simply applying the model inference for additional iterations does not always work in larger environments. CALVIN's performance in embodied indoor navigation (Tab. 2) is restricted by the small size of the dataset. The policy learned in scenes belonging to the training set is hard to generalize to unseen scenes in the validation set.

Diffusion Policy, equivalent to our framework without value guidance, attains a far lower success rate in GridMaze2D. This behavioral cloning approach hinges solely on conditional diffusion for policy derivation, neglecting the value function's role. As the maze expands, diffusion policy's effectiveness further diminishes, failing all navigation tasks in 30×30 mazes. This trend underscores the significance of value guidance in partially observable navigation, particularly when the target's location is unknown beforehand. Given the inferior performance, we exclude the diffusion policy from the comparative analysis in the more intricate AVD domain.

Scene	CALVIN-2D	CALVIN-3D	Ours (Zero-shot)	Ours (Retrain)
Home_001_1	0.692±0.037	0.720±0.052	0.769±0.038	**0.776±0.028**
Home_001_2	0.627±0.037	0.640±0.048	0.655±0.033	**0.732±0.030**
Home_002_1	0.735±0.035	0.740±0.048	0.728±0.034	**0.755±0.027**
Home_003_1	0.606±0.042	0.642±0.060	0.638±0.041	**0.686±0.031**
Home_003_2	0.558±0.033	0.590±0.043	0.603±0.033	**0.622±0.030**
Home_004_1	0.647±0.040	0.680±0.050	0.684±0.042	**0.695±0.036**
Home_007_1	0.587±0.038	**0.610±0.045**	0.584±0.039	0.601±0.035
Home_010_1	0.728±0.033	0.736±0.043	0.769±0.032	**0.781±0.028**
Mean succ. rate	0.635±0.032	0.682±0.047	0.679±0.040	**0.706±0.032**

Table 2. Performance of CALVIN and our method in AVD's embodied navigation and object searching tasks, where the goal is to locate a Coca-Cola glass bottle in an indoor scene. It presents the agent's success rates across various scenes. Our method, which achieves comparable results to CALVIN in zero-shot policy transfer from the 2D domain, surpasses CALVIN in scenarios retrained with additional RGB inputs, with an exception in one scene.

Our approach demonstrates a strong success rate of 0.906 in the standard setup of GridMaze2D, outperforming CALVIN and setting new state-of-the-art performance. Despite a performance dip in larger environments, the decline is gradual, underscoring our work's superior scalability. This success is largely due to the incorporation of multi-step action values in our value-guided plan selection for trajectory optimization (Eq. (9)) instead of focusing solely on the next step. This approach effectively mitigates potential collisions or repetitions during navigation. In the AVD domain, the superiority of our approach becomes more evident. Independent of limited scenes for policy learning, our policy transferred from GridMaze2D backed by extensive training data demonstrates improved generality and robustness, leading to better performance as shown in Tab. 2.

Fig. 6 illustrates a scenario where the three methods navigate the same maze. CALVIN falls into an indefinite loop due to the opposite actions suggested by the learned policy for observations of two consecutive steps. This is due to a combination of suboptimal modeling of the decision process and autoregressive single-step planning. Diffusion Policy fails early, especially after reaching a dead end, mainly due to two aspects. First, since the expert has full observation of the environment and is optimal, its demonstration for training never involves situations of encountering dead ends. Second, the behavior cloning essence of the Diffusion Policy is known to be effective in goal-conditioned planning. However, the agent cannot access the goal under partial observability until it is detected, significantly dropping the method's performance. On the contrary, our approach avoids loops by multi-step action prediction. It circumvents obsta-

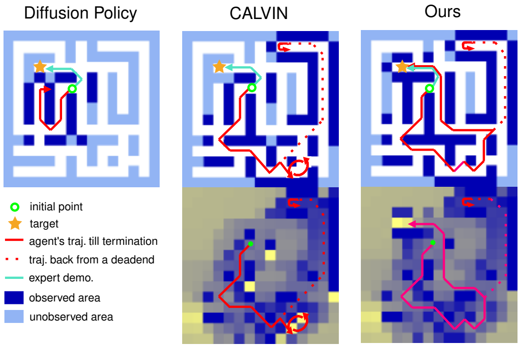

Figure 6. A scenario where three methods navigate the same maze. Diffusion Policy (left) collides with an obstacle after encountering a dead end, while CALVIN (middle) gets trapped in a repeating loop at a corner. Our approach (right), however, successfully backtracks from a dead end and identifies an alternate path to the goal, demonstrating its superior performance. Please note that the expert demonstration is gathered under **full** observability, with prior knowledge of the target's location. The heat maps illustrate the value learned at each spatial location by CALVIN and our framework, respectively, with brighter colors indicating higher values.

cles and safely backtracks from dead ends via effectively learned value guidance. This helps generalize the policy to unseen situations. Using the value as a guide also eliminates the need for access to the goal.

We then evaluate the robustness of the presented methods for different observability levels. As shown in Tab. 1, a smaller view range, indicating a lower observability level, generally leads to performance degradation. Among the three methods, ours exhibits superior performance and adaptability. As the view range increases from vr=1 to vr=3, our method consistently outperforms the others and is the least sensitive to the variation. CALVIN also shows good robustness and scalability, though it does not match our approach's performance. The Diffusion Policy struggles significantly in comparison, showing the least robustness and lowest performance across all observabilities. This analysis underscores the effectiveness and reliability of our method.

4.3. Ablation Study

We conduct a series of ablation studies to assess the contribution of each core component of our framework to performance in 15×15 grid mazes, and AVD embodied navigation. The full version can be represented as multi-samp.+val. guidance+best-plan memo.

Effect of multi-sampling. The Diffusion Policy [4], employs single-sampling. In contrast, our approach samples multiple times for each planning. The inherent stochasticity of diffusion sampling generates varied outputs, from which we choose the most frequent outcome for execution using a voting mechanism. This strategy modestly increases the success rate, underscoring the advantages of leveraging diffusion models for multiple rounds of sampling.

Ablation	GridMaze2D	AVD
Full version	**0.906±0.010**	**0.776±0.028**
single-samp.	0.060±0.022	0.024±0.012
multi-samp.+voting	0.114±0.025	0.082±0.026
multi-samp.+val. guidance	0.538±0.010	0.542±0.031
w/o PC to BEV projector	N/A	0.486±0.036

Table 3. Ablation experiments on navigation success rate in 15×15 GridMaze2D and Home_001_1 scene of AVD.

Effect of value guidance. Leveraging multi-sampling, we replace the voting mechanism for plan selection with a learned value function. Instead of choosing the most frequently sampled plan, we select the one with the highest multi-step Q value, as determined by the value function. This change markedly enhances performance, elevating success rates from 0.114 and 0.082 to 0.538 and 0.542 for GridMaze2D and AVD, respectively. This highlights the essential role of value-based guidance in our model's effectiveness.

Effect of best plan memory. We explore the significance of backtracking the past best plan, based on multi-sampling and value guidance. This mechanism is responsible for the performance gap between multi-samp.+val. guidance and the full version. The best plan memory addresses the issue of an optimal plan being replaced by a suboptimal one during re-planning in the context of receding horizon control. This underscores its crucial role in our methodology.

Effect of point cloud to BEV projector. Eliminating the semantic-segmentation-based projector hinders our framework's ability to apply the pre-trained policy for 2D domain to 3D navigation, necessitating the development of a new 3D-specific policy. To maintain the backbone of the diffusion-based plan generator, we adopt the lattice point net (LPN) used in CALVIN-3D for end-to-end policy learning. The complexity of this network alteration, coupled with the lack of ground-truth 2D maps for supervised projector training, leads to training difficulties, which causes a drop in success rate from 0.776 to 0.486. This emphasizes the importance of the semantic-segmentation-based projector in enabling the 2D policy's zero-shot transfer to 3D navigation.

5. Conclusion

This paper introduces a novel value-guided diffusion approach for trajectory-level plan generation, adept at navigating complex, long-horizon challenges under partial observability. Our approach exhibits remarkable versatility in both 2D and 3D environments and outperforms state-of-the-art methods. Extensive ablations underscore the importance of key constituents. Notably, our method effectively addresses the uncertainties inherent in partially observable environments, which is promising for real-world applications.

Acknowledgments: The first author especially thanks Bin Duan for his support during both submission and rebuttal. This research is supported by NSF IIS-2309073 and ECCS-212352101. This article solely reflects the opinions and conclusions of its authors and not the funding agencies.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *ICLR*, 2023. 1

[2] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *ICLR*, 2023. 2, 4

[3] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv:2208.04202*, 2022. 6

[4] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023. 1, 2, 4, 7, 8

[5] Xin Deng, Wenyu Zhang, Qing Ding, and Xinming Zhang. Pointvector: A vector representation in point cloud analysis. In *CVPR*, 2023. 2

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 2

[7] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv:1903.08689*, 2019. 2

[8] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *NeurIPS*, 2020. 2

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[10] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *JMLR*, 6(4): 695–709, 2005. 2

[11] Shu Ishida and João F. Henriques. Towards real-world navigation with deep differentiable planners. In *CVPR*, 2022. 1, 2, 4, 7

[12] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022. 1, 2, 3, 4

[13] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 1

[14] Peter Karkus, David Hsu, and Wee Lee. Qmdp-net: Deep learning for planning under partial observability. In *NeurIPS*, 2017. 1, 2

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 2

[16] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*, 2020. 2

[17] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *ICML*, 2023. 1

[18] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, 1995. 3

[19] Gabriel B. Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *CoRL*, 2022. 1

[20] David Q Mayne and Hannah Michalska. Receding horizon control of nonlinear systems. In *CDC*, 1988. 4

[21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 2

[22] Buqing Nie, Yue Gao, Yidong Mei, and Feng Gao. Capability iteration network for robot path planning. *IJRA*, 37(3):266–272, 2022. 2

[23] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *NeurIPS*, 2019. 2

[24] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. In *ICLR*, 2023. 1, 2, 4

[25] Joelle Pineau. *Tractable planning under uncertainty: exploiting structure*. Carnegie Mellon University, 2004. 3

[26] Daniel Schleich, Tobias Klamt, and Sven Behnke. Value iteration networks on multiple levels of abstraction. In *RSS*, 2019. 2

[27] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 2

[28] Edward J Sondik. The optimal control of partially observable markov decision processes. *PhD thesis, Stanford University*, 1971. 1

[29] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019. 2

[30] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 2

[31] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *NeurIPS*, 2016. 1, 2

[32] Tsun-Hsuan Wang, Juntian Zheng, Pingchuan Ma, Yilun Du, Byungchul Kim, Andrew Everett Spielberg, Joshua B. Tenenbaum, Chuang Gan, and Daniela Rus. Diffusebot: Breeding soft robots with physics-augmented generative diffusion models. In *NeurIPS*, 2023. 2

[33] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *ICLR*, 2023. 2, 4

[34] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011. 2

[35] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *arXiv:2304.06906*, 2023. 2, 6