CrowdSurfer: Sampling Optimization Augmented with Vector-Quantized Variational AutoEncoder for Dense Crowd Navigation

Naman Kumar*[1], Antareep Singha*[1], Laksh Nanwani*[1], Dhruv Potdar[1], Tarun R[1], Fatemeh Rastgar[2], Simon Idoko[2], Arun Kumar Singh[2], K. Madhava Krishna[1]

Fig. 1: The proposed method, **CrowdSurfer**, first generates a diverse set of samples using the VQ-VAE + PixelCNN pipeline, followed by inference time optimization via the PRIEST planner [1], resulting in trajectories that can traverse complex dynamic navigation scenarios successfully. From left to right we show HuskyA200, our Autonomous Wheelchair, and a simulated turtlebot navigating in crowded environments using **CrowdSurfer**.

***Abstract*— Navigation amongst densely packed crowds remains a challenge for mobile robots. The complexity increases further if the environment layout changes making the prior computed global plan infeasible. In this paper, we show that it is possible to dramatically enhance crowd navigation by just improving the local planner. Our approach combines generative modelling with inference time optimization to generate sophisticated long-horizon local plans at interactive rates. More specifically, we train a Vector Quantized Variational AutoEncoder to learn a prior over the expert trajectory distribution conditioned on the perception input. At run-time, this is used as an initialization for a sampling-based optimizer for further refinement. Our approach does not require any sophisticated prediction of dynamic obstacles and yet provides state-of-the-art performance. In particular, we compare against the recent DRL-VO approach [2] and show a 40% improvement in success rate and a 6% improvement in travel time.**

I. Introduction

Reliable, collision-free navigation in complex environments filled with human crowds is essential for deploying mobile robots in hospitals, offices, airports, etc [3], [4]. While this is a well-studied problem, one understated aspect of existing works is the reliance on a global plan computed based on prior maps. Put differently, existing planners show poor performance in the absence of a global plan or when it is rendered infeasible due to changes in the environment [1], [5].

* denotes equal contribution
[1] are with RRC, IIIT Hyderabad, India.
[2] are with University of Tartu, Estonia
Emails: (namanxkumar, antareepsinha12, lakshanshul, dhruvpotdar29, tarun.ramak)@gmail.com,(fatemeh.rastagar,simon.idoko)@ut.ee, aks1812@gmail.com, mkrishna@iiit.ac.in.
Project Page-https://smart-wheelchair-rrc.github.io/CrowdSurfer-webpage
Code-https://github.com/Smart-Wheelchair-RRC/CrowdSurfer

The presence of dense human crowds further necessitates complex on-the-fly decision-making as prior plans are of little use in this context. Existing works have tried to heavily leverage both imitation [6] and reinforcement learning (RL) [7], [8], [9] for navigation amongst dense crowds. The latter is particularly attractive since it not only gets rid of the necessity of obtaining expert demonstration but can also implicitly account for the interaction between the robot and the crowd. However, it is worth pointing out that only very few works like [2] have shown navigation amongst dense crowds in indoor environments with very tight spaces.

This paper shows that we can dramatically improve navigation in crowded tight spaces by just making the local planner more capable. In particular, a local planner capable of long-horizon planning at interactive rates can deliver strong performance even in the absence of any complex trajectory or interaction prediction of human crowds. To this end, our key **contribution** in this paper is the development of a local planner that combines generative modeling with inference-time optimization. Specifically, we use Vector Quantized Variational AutoEncoder (VQ-VAE) [10], to learn a discrete prior over the space of expert demonstrations. The discrete latent space is particularly suitable for capturing the inherent multi-modality of the expert demonstrations. We also train a PixelCNN [11] to sample from the learned priors and generate a distribution of trajectories during inference time, conditioned on the perception input.

We also perform inference time optimizations to ensure that the PixelCNN-generated trajectories exactly satisfy kinematic and collision constraints. This is achieved by using the learned posterior distribution as an initialization for a sampling-based trajectory optimizer built on top of our prior

work [1]. A unique aspect of [1] is that it uses a combination of convex optimization and gradient-free search to search across different potential homotopies for collision avoidance.

Although conceptually simple, our approach shows strong performance in various open-source benchmark environments augmented with human crowds controlled by the social-force model [12]. We specifically compare against DRL-VO [2] that combines RL with velocity-obstacle [13] based reactive collision avoidance. We achieve 40% and 6% improvement over DRL-VO in success rate and travel time respectively.

II. Related Works

Autonomous navigation capabilities still lack repeatability and robustness, especially in cluttered environments packed with dense crowds. Nevertheless, this problem has been well studied and we review the prominent works in this direction and contrast our approach with them.

Model-Based Planning: This class of planners is based on classical graph/sampling-based search, mathematical optimization, or a combination of both. Two such approaches that still form the backbone of autonomous navigation in both research and industry are Dynamic Window Approach (DWA) [14] and Timed Elastic Bands (TEB) [15]. In our recent paper [1], we showed that these two approaches substantially outperform even some recent approaches like Model Predictive Path Integral [16] and its variants like Log-MPPI [17]. A similar observation was also put forward in [5]. An important caveat in model-based planning is the requirement of a global planner, which in turn requires a prior map of the environments. In the absence of a global plan, all existing approaches show massive performance deterioration, even in the absence of dynamic obstacles [5], [1].

Learning-Based Approaches: Over the last decade, there has been a strong interest in applying RL to crowd navigation [7], [8], [9], [18]. These works rely on implicitly or explicitly modeling the interaction between the crowd and the robot and that between members of the crowd themselves. However, most RL approaches show results in spaces where the robot just needs to deal with the dynamic crowd. Recently, [2] has shown impressive performance in environments that resemble real-world settings that are highly cluttered as well as have tens of dynamic human obstacles. One critical drawback of purely learning-based approaches is that they typically struggle when encountered with novel scenarios. As we show later, even the performance of [2] deteriorates when tasked with longer runs.

Social navigation frameworks like [19], [20], [21] focus on collision avoidance while also adhering to social norms, enabling robots to anticipate and respect human movement/interaction in social spaces. In contrast, the focus of our work is to address the challenges of local navigation, which in turn, can also contribute towards improving human-aware navigation.

Our Contribution Over SOTA: Our proposed approach aims to find the middle ground between model-based planning and purely learning-based approaches. We aim to make the algorithmic parameters of the model-based planners adaptive to the environmental conditions. At the same time, we intend to improve performance in novel scenarios. We fulfill both these objectives by combining generative model (VQ-VAE+PixelCNN) based imitation learning with an expressive model-based planner PRIEST [1] capable of searching across multiple homotopies. As mentioned earlier, the VQ-VAE+PixelCNN can be seen as learning a prior over the optimal trajectories while PRIEST provides the inference-time refinement.

III. Methodology

A. Problem Formulation

We follow the trajectory optimization template and formulate crowd navigation in the following manner.

$$\min_{x(t),y(t)} c(x^{(q)}(t), y^{(q)}(t)), \tag{1a}$$

$$x^{(q)}(t_0), y^{(q)}(t_0) = \mathbf{b}_0,\ x^{(q)}(t_f), y^{(q)}(t_f) = \mathbf{b}_f, \tag{1b}$$

$$\dot{x}^2(t) + \dot{y}^2(t) \leq v_{max}^2,\ \ddot{x}^2(t) + \ddot{y}^2(t) \leq a_{max}^2, \tag{1c}$$

$$-\frac{(x(t) - x_{o,j}(t))^2}{a_j^2} - \frac{(y(t) - y_{o,j}(t))^2}{a_j^2} + 1 \leq 0, \tag{1d}$$

where $(x(t), y(t))$ and $(x_{o,j}(t), y_{o,j}(t))$ respectively denote the robot and the j^{th} obstacle position at time t. These obstacles could be either individual LiDAR points or a dynamic human. We model each type as a circular obstacle with radius a_j. The function $c(.)$ could be any arbitrary function (even non-smooth, non-analytic) of derivatives of the position-level trajectories. We can also leverage the differential flatness of a typical mobile robot to include control costs in $c(.)$. The vectors $\mathbf{b}_0$ and $\mathbf{b}_f$ in (1b) represent the initial and final values of boundary condition on the q^{th} derivative of the position-level trajectory. We employ $q = \{0, 1, 2\}$ in our implementation. Inequalities (1c) bound the maximum values of velocities and accelerations. In (1d), we enforce collision avoidance, assuming circular obstacle with radius a_j.

We can convert (1a)-(1d) into a finite-dimensional representation by parameterizing the trajectories as polynomials in the following manner.

$$\begin{bmatrix} x(t_1) \\ \vdots \\ x(t_{n_p}) \end{bmatrix}^T = \mathbf{P}\ \mathbf{c}_x, \begin{bmatrix} y(t_1) \\ \vdots \\ y(t_{n_p}) \end{bmatrix}^T = \mathbf{P}\mathbf{c}_y, \mathbf{W} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \tag{2}$$

where the matrix $\mathbf{P}$ is a matrix formed by time-dependent polynomial basis functions. The vectors $\mathbf{c}_x, \mathbf{c}_y$ are the coefficients attached to the individual basis functions. We can represent the derivatives like $\dot{x}(t), \ddot{x}(t), \dot{y}(t), \ddot{y}(t)$ in a similar manner as (2) using $\dot{\mathbf{P}}$ and $\ddot{\mathbf{P}}$

Using (2), we can represent (1a)-(1d) in the following compact form, wherein $\boldsymbol{\xi} = (\mathbf{c}_x, \mathbf{c}_y)$

$$\min_{\boldsymbol{\xi}} c(\boldsymbol{\xi}) \tag{3a}$$

$$\mathbf{A}\boldsymbol{\xi} = \mathbf{b}_{eq} \tag{3b}$$

$$\mathbf{g}(\boldsymbol{\xi}) \leq \mathbf{0}, \tag{3c}$$

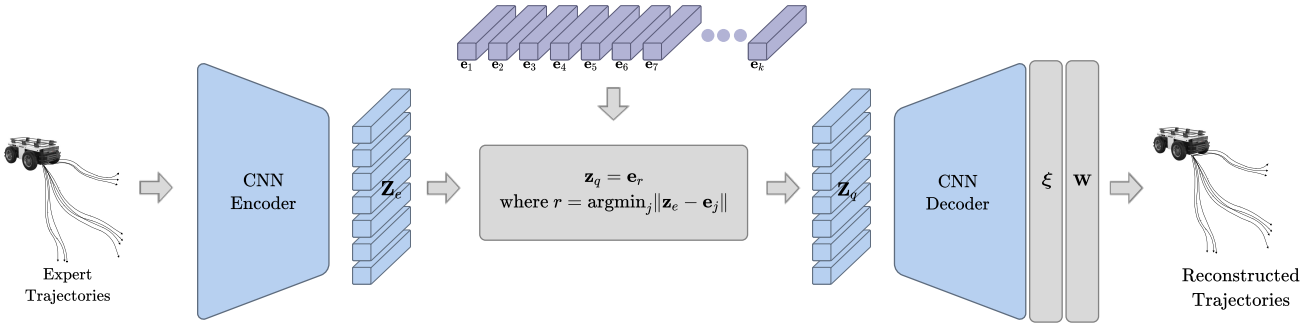


Fig. 3: The VQ-VAE pipeline is used to learn the discrete latent space of optimal demonstration trajectories. We train the VQ-VAE to predict polynomial coefficients that are converted to trajectories through (2). This ensures that the reconstructed trajectories have higher continuity and differentiability.

B. Learning the Discrete Latent Space of Solutions Through VQ-VAE

We intend to learn the solution space of (1a)-(1d) through demonstration of optimal trajectories in a similar style as our prior work [22] designed for autonomous driving. To this end, we first compress the demonstration trajectories to a discrete latent space. Such representation can capture the multi-modality in the solution trajectories such as avoiding obstacles from the left vis-a-vis from the right.

Our VQ-VAE architecture, illustrated in Fig. 3 adapts the original formulation [23] for trajectory generation. We use a CNN encoder to compress expert trajectories into a continuous latent space $\mathbf{Z}_e \in \mathbb{R}^{L \times D}$, where L denotes the number of latent vectors, each with dimension D. We denote the i^{th} latent vector in $\mathbf{Z}_e$ as $\mathbf{z}_{e,i}$. The fundamental feature of VQ-VAE is the transformation from continuous space to a discrete one: mapping $\mathbf{Z}_e$ to $\mathbf{Z}_q$. To achieve this, we introduce a latent embedding matrix $\mathbf{E} \in \mathbb{R}^{K \times D}$ called code-book with K discrete vectors $\mathbf{e}_j$. Each i^{th} latent vector in $\mathbf{Z}_e$ is assigned the nearest $\mathbf{e}_j$ vector based on the nearest neighbor equation detailed in (4), which makes up the i^{th} latent vector in $\mathbf{Z}_q$.

$$\boldsymbol{z}_{q,i} = \boldsymbol{e}_r, \text{ where } r = \arg\min_j ||\boldsymbol{z}_{e,i} - \boldsymbol{e}_j||_2^2 \quad \text{for } i \in \{1, 2, ..., L\} \tag{4}$$

The VQ-VAE decoder receives $\mathbf{Z}_q$ and reconstructs it to polynomial coefficients $\boldsymbol{\xi}$ and subsequently to trajectories using (2). The VQ-VAE model is optimized using the loss function detailed in (5), which comprises three parts. The first term is the reconstruction loss which ensures that the encoder/decoder pair effectively reconstructs the input expert trajectory. The second and third terms are called the codebook and commitment loss, which are used to update the code-book vectors during training and bypass the non-differentiable discretization presented in (4).

$$\mathcal{L}_{vqvae} = \|\mathbf{W}\boldsymbol{\xi} - \boldsymbol{\tau}_e\|_2^2 + \|\text{sg}[\mathbf{Z}_e] - \mathbf{E}\|_2^2 + \beta\|\mathbf{Z}_e(x) - \text{sg}[\mathbf{E}]\|_2^2 \tag{5}$$

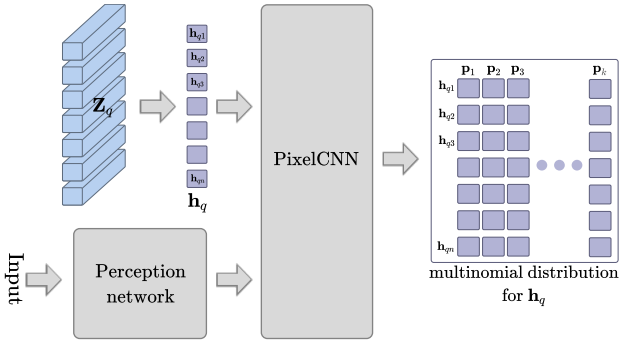


Fig. 4: It is possible to express the learned discrete latent space $\mathbf{Z}_q$ as a vector $\mathbf{h}_q$ whose elements signify which code-book vector is used to form the i^{th} row of $\mathbf{Z}_q$. Our PixelCNN model outputs a multinomial probability distribution over $\mathbf{h}_q$. The number of discrete probabilities p_k is equal to the number of code-book vectors. Thus, the PixelCNN model decides the probability that the i^{th} element of $\mathbf{h}_q$ is formed by the r^{th} code-book vector. Sampling from the multi-nomial distribution allows the generation of different $\mathbf{h}_q$ each leading to a distinct trajectory.

C. Using a Conditional PixelCNN for sampling from the VQ-VAE latent space

For sampling from the latent space of the VQ-VAE, we adapt the Conditional PixelCNN [11] that allows us to model the conditional distributions of the space. Using this model, we can generate diverse trajectories (via the VQ-VAE decoder) for different environments in which the robot operates, by simply conditioning it on features extracted from the occupancy grid map, dynamic obstacle states, and immediate heading to the goal.

We recall that the discrete latent space is a matrix $\mathbf{Z}_q$. Each row of this matrix is formed by $\mathbf{z}_{q,i}$ which in turn is related to the r^{th} code book vector through (4). Thus, it is trivial to encode the information about $\mathbf{Z}_q$ into a vector $\mathbf{h}_q$ in which each element stores the index of the corresponding codebook vector used to form $\mathbf{z}_{q,i}$ (see Fig.4). Hence, the training of the VQ-VAE in the last section provides us the ground-truth values of $\mathbf{h}_q$.

The PixelCNN model outputs a multinomial probability distribution over $\mathbf{h}_q$. We can sample from this distribution to generate different $\mathbf{h}_q$ samples and consequently diverse $\mathbf{Z}_q$. This in turn can be fed to the trained decoder of VQ-VAE to generate trajectory samples.

The defining feature of the PixelCNN model is that the probability distribution over $\mathbf{h}_q$ is generated in an auto-

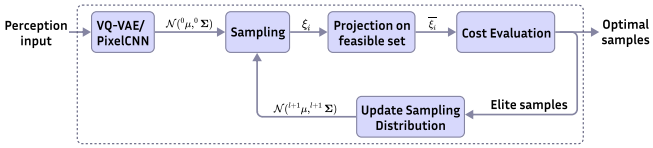


Fig. 5: At any given planning step, the VQ-VAE+PixelCNN combination takes in the perception input and generates samples for initialization of the PRIEST planner [1]. The planner subsequently refines the predicted distribution using a combination of projection-optimizer augmented cross-entropy method [24]

regressive manner through (6). That is, the prediction of each element of $\mathbf{h}_q$ depends on the prediction of the previous elements and the conditioning input. During the training phase, we form a cross-entropy loss over the ground-truth $\mathbf{h}_q$ (from VQ-VAE) and that predicted by PixelCNN to learn the parameters of the multinomial distribution.

$$p(\mathbf{h}_q|\mathcal{O}) = \prod_{i=1}^{L} p(\mathbf{h}_{q,i}|\mathbf{h}_{q,1}, \ldots, \mathbf{h}_{q,i-1}, \mathcal{O}) \tag{6}$$

In the inferencing phase, we start with $\mathbf{h}_q$ set to zeros and using the trained PixelCNN, we recursively generate indices $\mathbf{h}q, i$ through (6) based on the observed conditioning input, to obtain a multinomial distribution for each element of $\mathbf{h}_q$.

D. Inference-Time Optimization

Although, VQ-VAE and PixelCNN are powerful generative models, their generated trajectories may not exactly satisfy the different kinematic and collision constraints. Thus, we refine the PixelCNN-generated trajectory distribution with a sampling-based optimization PRIEST [1]. Specifically, we replace the Gaussian distribution initialization of PRIEST with our PixelCNN model (see Fig. 5).

E. Data Collection Pipeline and Perception Network Architecture

1) Data Collection Pipeline: The training data for the pipeline includes expert trajectories of a robot navigating highly dynamic environments, used to train the unconditional VQ-VAE, and observation data for training the PixelCNN model. All data was collected in a simulated setting using the PEDSIM social-force library [25] and the Gazebo simulator [26]. We teleoperated a Turtlebot2 within the open-source Lobby World environment, featuring 35 dynamic agents. The simulated positions of these agents (agent states) and LiDAR scans were used as ground truth inputs. The teleoperated expert trajectories were pre-processed using the PRIEST planner. This resulted in smoother trajectories that proved more conducive to training VQ-VAE.

For PixelCNN training, the observation data comprising of LiDAR scans and dynamic agent states were pre-processed to match the network's input requirements. LiDAR scans were converted to Occupancy Maps, and dynamic agent states were used to calculate agent velocities. Additionally, odometry data was processed to generate ego velocities and the heading-angle-to-goal.

2) Perception Network Architecture: The Perception Network encodes the observation space into a single conditioning vector which is passed to the PixelCNN. It accepts an occupancy map in the form of a single channel image $O \in \mathbb{R}^{H\times W\times 1}$, positions and velocities of dynamic obstacles for the past $T(=5)$ timesteps $D \in \mathbb{R}^{T\times 4\times M}$ where $M(=10)$ is the maximum number of dynamic obstacles. We also use the current heading of the mobile robot to the current goal in radians $-\pi \leq H \leq \pi$ as an input to the perception network.

Occupancy map O is encoded via four 2D convolutions with batched normalization, adaptive average pooling, and a series of fully connected (FC) layers to yield a single embedding. Each timestep of the dynamic obstacle input D is passed through a similar architecture as the occupancy map encoder, but utilizing 1D convolutions instead, resulting in T vectors. These are then passed through an LSTM and a FC layer to yield a single embedding. Finally, the heading H is processed through a FC layer and concatenated with the occupancy map and dynamic obstacle embedding. This concatenated vector is then passed through a FC layer for the final conditioning embedding $\mathcal{O}$.

IV. Validation and Benchmarking

The objective of this section is to answer the following questions.

1) Does our method generalize to multiple pedestrian densities and environments?
2) How does our method compare to the current state-of-the-art methods?
3) Is our method reliant on a global plan?

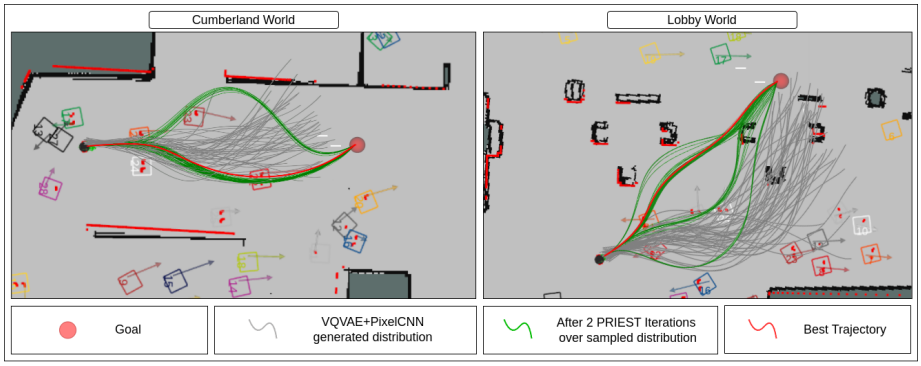
A. Implementation Details

The VQVAE codebook is set to 64 vectors, of size 4 each. The input occupancy map is generated using a 5m maximum observation radius and a 0.1 resolution. The number of PRIEST iterations is set to 2 (6x lower than the original), with 50 initial samples drawn from the distribution predicted by the PixelCNN. The PRIEST optimization is done using a maximum of 10 dynamic obstacles and 100 static obstacles (from the downsampled point cloud). All inference is done on an NVIDIA 3060 Mobile GPU with 6GB of VRAM but occupies less than 1GB in practice.

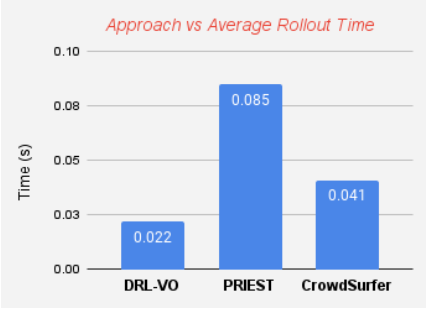
B. Simulation Configuration

1) Robot Configuration: We use the Turtlebot2 Robot equipped with a 2D LIDAR sensor used for mapping the environments before runs and as the primary sensor to navigate in the environment during each run. We use the open-source AMCL [27] library for localization during trials.

2) Environment Configuration: We naturally test in the Lobby World environment on which our pipeline has been trained. The other unseen environments we test on include Cumberland, Freiburg, Autolab, and Square world described in the DRL-VO paper as well as an additional custom Hospital world. All these environments had 35 dynamic pedestrians, and in the Lobby, Cumberland, and Square worlds, we additionally tested with 55 dynamic pedestrians.



(a) Trajectories at 3 stages of the pipeline. VQVAE + PixelCNN generate diverse primitives (GREY), which are then optimized via 2 PRIEST [1] iterations (GREEN) and the best trajectory as scored by PRIEST is chosen (RED).



(b) CrowdSurfer improves upon the computation times of PRIEST while also delivering better results in navigation than both PRIEST (Table III) and DRL-VO (Table I). Our pipeline, being a trajectory rollout approach, still achieves real-time frequencies of approx. 20 Hz, making it easily deployable on real-world robots (Fig. 1)

Fig. 6: (a) Qualitative Results and (b) Compute Time Comparison

C. Qualitative Results

Figure 6a shows the typical trajectories resulting from our pipeline in Cumberland and Lobby environments. The VQ-VAE+PixelCNN trajectories are shown in grey. As can be seen, these trajectories are very diverse but do not necessarily converge at the goal location. The trajectory distribution resulting after refinement from the PRIEST optimizer is shown in green. As can be seen, the refined trajectories are smoother, directed towards the goal, and show multi-modal behavior for avoiding collisions. We find that 2 iterations of PRIEST optimization are enough for all the experiments reported in this section.

D. Comparison with DRL-VO [2]

In order to measure the efficacy of our method, we compare it against DRL-VO [2], a state-of-the-art dynamic scene navigation algorithm. We provide test results, using the Gazebo simulator and the PEDSIM library, in 3 distinct environments and different densities of pedestrians. We first consider performance when the global plan is provided, to replicate the original DRL-VO testing conditions exactly. We compare using four commonly used metrics in the literature:

1) **Success Rate:** the fraction of trials where the robot reaches the goal without any collisions.
2) **Average Time:** average travel time across trials
3) **Average Length:** average length of the entire path to goal across trials
4) **Average Speed:** average travel speed across trials

For each environment and pedestrian density, we assign 20 successive goal points. Each goal point is a single trial. A trial is considered successful if the goal is reached without collisions. In case of failure, the current goal is used as the initial position, and trials continue from the next goal. The average speed and distance metrics are computed exclusively for successful trials, where the agent successfully reached its designated goal.

Table I shows the previously described metrics for three of the testing environments. Cumberland and Square World were unseen environments for our VQ-VAE+PixelCNN model. As can be seen, we show strong success rates on Lobby World which was seen during training. However, more importantly, our performance generalizes well to the unseen scenarios. In all environments, our method consistently outperforms DRL-VO with a success rate upwards of 0.8 with both 35 and 55 pedestrians. This is approximately 40% improvement over DRL-VO.

Additionally, our pipeline is able to traverse the benchmark scenarios with a higher average speed, and a lower average path length in most cases than those observed with DRL-VO. This is indicative of the reactive nature of DRL-VO, as compared to the long-horizon planning afforded by our pipeline. In some cases, as in Cumberland and Square World environments, DRL-VO takes a lower trajectory length by showing more aggressive planning behavior. However, our pipeline still yields a lower average time in these cases by planning ahead.

It is important to highlight that the DRL-VO performance, especially the success rate that we obtained is substantially lower than that reported in [2]. One reason for this is that we benchmark with substantially longer runs than [2]. Specifically, the start-to-goal distance in our benchmarking is almost twice what is used in [2].

DRL-VO compute times were typically lower at approximately 0.022 seconds, owing to its one-shot planning approach and reactive nature. However, our pipeline has a compute time adequately low for real-time functioning of approximately 0.041 seconds despite calculating trajectories over a horizon of 5 seconds with 50 timesteps as can be seen in Fig. 6b.

E. Adapting to Changing Map

In this subsection, we test the adaptivity of our approach and DRL-VO to the changes in the static environment. We take the Square World environment with 25 dynamic pedestrians and add additional static obstacles. These obstacles were not seen during the mapping and thus, the prior-compute plan

Planner Type	Environment	Method	Success Rate ↑		Average Time [s] ↓		Average Length [m] ↓		Average Speed [m/s] ↑	
			35	55	35	55	35	55	35	55
w/ Global Planner (Dijkstra)	Lobby World	DRL-VO	0.65	0.55	19.06	23.68	**12.2**	**13.5**	0.64	0.57
		CrowdSurfer	**0.85**	**0.80**	**17.58**	**21.67**	12.66	13.65	**0.72**	**0.63**
	Cumberland	DRL-VO	0.75	0.55	16.57	**18.57**	**10.11**	**11.14**	0.61	0.60
		CrowdSurfer	**0.90**	**0.85**	**15.16**	19.73	11.37	12.82	**0.75**	**0.65**
	Square World	DRL-VO	0.80	0.80	19.81	26.39	14.86	18.74	0.75	**0.71**
		CrowdSurfer	**0.90**	**0.90**	**18.5**	**24.55**	**14.43**	**16.94**	**0.78**	0.69
w/o Global Planner	Lobby World Cumberland	DRL-VO	0.45	0.40	25.41	**26.47**	**16.01**	**14.29**	0.63	0.54
		CrowdSurfer	**0.75**	**0.65**	**23.44**	28.75	16.17	18.11	**0.69**	**0.63**
		CrowdSurfer	**0.75**	**0.65**	**21.22**	**26.59**	**13.37**	**17.20**	**0.63**	0.47
	Square World	DRL-VO	0.65	0.60	31.40	35.86	22.61	23.67	0.72	**0.66**
		CrowdSurfer	**0.80**	**0.80**	**29.61**	**32.71**	**21.91**	**21.26**	**0.74**	0.65

TABLE I: Navigation results for three selected testing environments with 35 and 55 dynamic pedestrians. (Average Values are only for successful scenarios). Our pipeline shows a significant improvement in success rate in the navigational tasks performed in densely crowded scenarios as opposed to DRL-VO.

becomes infeasible on many runs. Table II summarizes the key results for 9 trials. As can be seen, the DRL-VO success rate dropped from 0.75 (Table I) to 0.44. In contrast, our approach showed a very marginal drop to 0.89 from 0.90 (Table I).

F. Reliance on Global Plan

Table I (second half) compares the performance of DRL-VO and our approach in the absence of guidance from the global plan. More specifically, the global plan is simply a straight line connecting the start and the goal. As can be seen, although the success rate reduces for both approaches, the performance falloff is steeper for DRL-VO dropping approximately 28% on average. In contrast, our approach is much more robust dropping approximately 15% on average.

G. Ablation: Comparison with PRIEST [1]

Our pipeline builds on PRIEST by augmenting it with a learned and environment-conditioned initialization through the VQ-VAE+PixelCNN model. Thus, this sub-section analyzes the impact that our learned model provides over the regular PRIEST pipeline. We consider the Cumberland environment with 35 dynamic agents as PREIST performed best in this setting. The results are summarized in Table III. We showcase the statistics for different computation budgets of PRIEST (2, 10, 12 iterations). As can be seen, the environment conditioned prior provided by VQ-VAE+PixelCNN dramatically improves the success rate along with average travel time, path length, and speed.

Method	**Success Rate ↑**
DRL-VO	0.44
Ours	**0.89**

TABLE II: Navigation Results with continuously changing environments

Method	**Success Rate ↑**	**Time[s] ↓**	**Length[m] ↓**	**Speed[m/s] ↑**
PRIEST-2	0.20	62.72	18.19	0.29
PRIEST-10	0.30	56.00	16.24	0.29
PRIEST-12	0.60	53.79	17.75	0.33
Ours	**0.90**	**31.40**	**23.5**	**0.75**

TABLE III: Ablation Study with PRIEST planner [1]. The PRIEST planner is run for different iterations (2, 10, 12)

H. Real World Configuration

We have tested our pipeline in the real world on two different robots:

- The Husky A200 mobile robot, which we have equipped with a SLAMTEC RPLIDAR S2 and an Intel® Realsense D455 RGBD Camera as the primary sensors
- A custom-made Autonomous Wheelchair equipped with a SLAMTEC RPLIDAR A3 and an Intel® Realsense D455 RGBD Camera as the primary sensors

For the detection of humans in the scene, we use two different methods and play around:

- A combination of YOLO [28] in the image frame and lidar for accurate depth information.
- Leg Tracker [29]: Detects leg-like patterns in lidar scans using clustering.

The testing was done in the lab at RRC, IIIT Hyderabad, and various static and dynamic obstacles were added during navigation, as can be seen in Figure 1.

V. Conclusion

This paper proposed a novel framework that couples generative prior models learned on expert data with inference time optimization to show results that are significantly better than prior art in highly populated pedestrian worlds containing as many as 55 moving pedestrians. While the generative priors are efficient in exploring the homotopies and are inherently multimodal they are not goal-conditioned and are not guaranteed to avoid collisions. The method of inference time batch projection ensures both these conditions are met with a very high degree of accuracy. Further, the efficacy of the proposed framework in the absence of a global plan or in changed environments where the planner is rendered ineffective is a cornerstone of this effort. In the future, we expect to condition the generative prior on both image and range data even as we focus on improving real-time performance with higher fidelity.

Acknowledgements

The author, Laksh Nanwani, thanks IHub-Data, IIIT Hyderabad, for extending their research fellowship. We also acknowledge IHub-Data for supporting this work.

REFERENCES

[1] F. Rastgar, H. Masnavi, B. Sharma, A. Aabloo, J. Swevers, and A. K. Singh, "Priest: Projection guided sampling-based optimization for autonomous navigation," *IEEE Robotics and Automation Letters*, 2024. 1, 2, 4, 5, 6

[2] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2700–2719, 2023. 1, 2, 5

[3] R. Valner, H. Masnavi, I. Rybalskii, R. Põlluäär, E. Kõiv, A. Aabloo, K. Kruusamäe, and A. K. Singh, "Scalable and heterogenous mobile robot fleet-based task automation in crowded hospital environments—a field test," *Frontiers in Robotics and AI*, vol. 9, p. 922835, 2022. 1

[4] A. Pratkanis, A. E. Leeper, and K. Salisbury, "Replacing the office intern: An autonomous coffee run with a mobile manipulator," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1248–1253. 1

[5] X. Xiao, Z. Xu, Z. Wang, Y. Song, G. Warnell, P. Stone, T. Zhang, S. Ravi, G. Wang, H. Karnan *et al.*, "Autonomous ground navigation in highly constrained spaces: Lessons learned from the barn challenge at icra 2022," *arXiv preprint arXiv:2208.10473*, 2022. 1, 2

[6] J. Bi, T. Xiao, Q. Sun, and C. Xu, "Navigation by imitation in a pedestrian-rich environment," *arXiv preprint arXiv:1811.00506*, 2018. 1

[7] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6015–6022. 1, 2

[8] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, "Robot navigation in a crowd by integrating deep reinforcement learning and online planning," *Applied Intelligence*, vol. 52, no. 13, pp. 15 600–15 616, 2022. 1, 2

[9] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 007–10 013. 1, 2

[10] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017. 1

[11] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in neural information processing systems*, vol. 29, 2016. 1, 3

[12] G. Shafabakhsh and M. Mohammadi, "Simulation of pedestrian movements using social force model," *Journal of Modeling in Engineering*, vol. 11, no. 34, pp. 49–62, 2013. 2

[13] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935. 2

[14] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997. 2

[15] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 european control conference (ECC)*. IEEE, 2015, pp. 3352–3357. 2

[16] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017. 2

[17] I. S. Mohamed, K. Yin, and L. Liu, "Autonomous navigation of agvs in unknown cluttered environments: log-mppi control strategy," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 240–10 247, 2022. 2

[18] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059. 2

[19] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "Sacson: Scalable autonomous control for social navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 49–56, 2024. 2

[20] X.-T. Truong and T. D. Ngo, "Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1743–1760, 2017. 2

[21] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *2013 European Conference on Mobile Robots*, 2013, pp. 331–336. 2

[22] S. Idoko, B. Sharma, and A. K. Singh, "Learning sampling distribution and safety filter for autonomous driving with vq-vae and differentiable optimization," *arXiv preprint arXiv:2403.19461*, 2024. 3

[23] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017. 3

[24] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018. 4

[25] C. Gloor, "PEDSIM: Pedestrian crowd simulation," 2016. [Online]. Available: http://pedsim.silmaril.org 4

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. Ieee, 2004, pp. 2149–2154. 4

[27] B. P. Gerkey, "Amcl - ros wiki." [Online]. Available: https://wiki.ros.org/amcl 4

[28] G. Jocher, A. Chaurasia, and J. Qiu, "Yolo by ultralytics," https://github.com/ultralytics/ultralytics, 2023. 6

[29] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 726–733. 6