University of Toronto

Trajectory Prediction and LLM Reward Tuning for Robot Social Navigation with Deep Reinforcement Learning

by

Jeongwoong (Daniel) Choi

A full-year MIE498 Undergraduate Thesis for the Faculty of Applied Science and Engineering

under the guidance

of

Dr. Goldie Nejat

Abstract

The challenge of enabling robots to navigate efficiently in spaces shared with humans is a multifaceted problem. Many current approaches consider prediction and planning in a coupled manner, allowing for a holistic view within a Partially Observable Markov Decision Process (POMDP). The robot agent would consider predictions such as human trajectories to ensure Cooperative Collision Avoidance (CCA) between robots and humans, highlighting the importance of social norms in robotic navigation. We advance this domain in this paper by replicating a leading trajectory prediction method using Spatial Occupancy Grid Maps (SOGMs) and a Convolutional Neural Network (CNN) specialized for point clouds to forecast human movements up to four seconds ahead. We also leverage NVIDIA's Omniverse Isaac Gym to develop a training environment for a Jackal robot with the Proximal Policy Optimization (PPO) algorithm, enhanced by Eureka, a novel reward tuning method using Large Language Models (LLMs). This integration facilitates zero-shot learning for navigation policies that consider dynamic social behaviors. Key results indicate a significant leap in the robot's ability to predict human trajectories and navigate in dynamic social environments. The use of Eureka for reward tuning resulted in a refined policy that outperformed the human-crafted policy by 50.7%, improving navigation success and efficiency in dynamic obstacle avoidance. Our conclusion emphasizes the efficacy of integrating deep reinforcement learning with trajectory prediction and LLM-based reward tuning in creating more adaptable, socially aware navigation systems for mobile robots.

Acknowledgments

I extend my heartfelt gratitude to Professor Goldie Nejat for her invaluable mentorship and the opportunity to undertake this significant research. I am deeply thankful to Aaron Hao Tan for his unwavering support and patience, and to Haitong Wang and Siddarth Narasimhan for their expert guidance in DRL simulation development. Lastly, I am forever grateful to my family and friends for their endless encouragement and support throughout this journey.

Table of Contents

List of Figures

List of Equations

1 Introduction

In the evolving field of robotics, the integration of autonomous mobile robots into our daily lives presents a frontier of technological advancement and innovation. However, to achieve full autonomy within human-centric environments, robots must not require human assistance in achieving their tasks. This poses a gap within the current autonomous social navigation methods in mobile robots: seamless navigation with prediction of human intent and trajectories [1]. For example, people can anticipate the intent of another to either shift their trajectories or slow down to find the most optimal action/path to account for the deviation in their original plan. The primary goal of this thesis is to develop a mobile robot social navigation system capable of predicting and adapting to human intent and trajectories, achieving seamless navigation in human-centric environments without human intervention. Current advances within the autonomous robotics field aim to achieve this behavior of seamless navigation for the surveillance, exploration, and social inclusion of mobile robots [2].

One area of interest for trajectory prediction is an extensive series of work that first proposed the self-supervised automated annotation of LiDAR point cloud map using Ray-tracing techniques [3]. The study integrates a novel convolutional network, KPConv [4] for extracting point clouds in the observed LiDAR map for segmentation and scene context understanding. This method differs from the traditional method of object detection and tracking, eliminating the need for manual labeling of images, relative to the concept of adaptive and self-supervised learning of autonomous systems. In addition to the trajectory prediction and within the context of this thesis, we will explore a navigation system that can unify different uncertainty cues, contextual information, and multi-modal data [5]. This research is significant for its potential to drastically improve the adaptability and safety of autonomous mobile robots in dynamic environments, making a substantial leap toward their seamless integration into society.

By including Deep Reinforcement Learning (DRL) in navigation, we can address a critical gap of traditional planning methods that lack robustness in dynamic and unpredictable environments. Traditional planners, often reliant on static models and predefined pathways, struggle to adapt to the real-time complexities of human movement and behavior. These systems can be faulty in scenarios where human actions are unpredictable or when environmental variables change unexpectedly (e.g., a crowded marketplace, sudden movements in public spaces, etc.), leading to inefficient path planning and potential safety risks. Therefore, we developed a simulation environment within the state-of-the-art NVIDIA Omniverse Isaac Gym [6]. This environment is specifically designed to parallelize the training process, significantly reducing the amount of time typically required for training sophisticated DRL policies.

Furthermore, we have leveraged other state-of-the-art advancements within the field of Large Language Models to refine the trained policy with human-level reward design [7]. Also known as “Eureka”, it distinguishes itself through its ability to generate and refine reward functions automatically, without needing task-specific prompting or predefined reward templates. It achieves this by using evolutionary optimization over reward code generated through the zero-shot capabilities of LLMs. A key feature we want to focus on utilizing is the “gradient-free in-context learning approach” to Reinforcement Learning from Human Feedback (RLHF). The pipeline will allow for the specification and refinement of reward functions based on human language prompts, translating these inputs into continuous refinement of the training policy. This capability is optimal for specifying different social norms and expectations in human-robot interactions, ensuring that the robot’s behavior aligns with human preferences and societal standards.

The purpose of this thesis is to address the identified research gap of current autonomous systems' inability to dynamically adapt to unpredictable human behavior and environmental changes in real time. The core contributions of this study encompass 1) the application of a self-supervised, Ray-tracing-based technique for LiDAR point cloud annotation, facilitating trajectory prediction capabilities, and 2) the employment of a novel reinforcement learning framework that leverages the zero-shot ability of Large Language Models for dynamic, human-centered reward tuning, setting a new standard for robot navigation in human environments. The future works section will also go over some limitations and how the RL navigation pipeline can complement our trajectory prediction base module for an end-to-end prediction and planning method.

2 Literature Review

In the domain of social navigation, environments are shared by multiple autonomous agents, each tasked with its navigational objectives. These agents, which can represent individuals or groups, operate within a shared space, where their movement decisions are made independently. Despite the absence of direct communication between agents, there is an implicit interaction driven by mutual observation. This interaction necessitates the anticipation of others' movements and requires continuous adaptation to the dynamically changing environment [8].

Formally defined, social navigation involves several agents, a_i greater than one, each positioned within a specific configuration in space at any given time $t \geq 0$. Their goal is to navigate to predetermined destinations g_i while ensuring they do not collide with static obstacles or intrude upon the safe space defined among moving agents, thereby adhering to socially accepted norms. During the planning phase, an agent formulates a path $\tau_i : [0,1] \rightarrow \mathbb{Q}$, from its current position to its destination. This path is determined by solving an optimization problem that minimizes a cost function of the form:

$$\begin{aligned} \tau_i = \arg\min_{\tau \in T} c_i(\tau_i) + \lambda_i c_i^s(\tau_i, \tilde{\tau}_{-i}) \\ s.t. \quad A_i(\tau_i(t)) \notin \mathcal{C}_{\text{obs}},\ \forall t \in [0,1], \\ A_i(\tau_i(t)) \cap A_j(\tilde{\tau}_j(t)) = \emptyset, \\ \forall t \in [0,1], j \neq i, \\ \tau(0) = q_i, \\ \tau(1) = g_i. \end{aligned} \tag{1}$$

This cost function [8] comprises an individual component, c_i related to the agent's navigation preferences and a social component, c_i^s that incorporates considerations of personal space and anticipates the future movements of other agents $\tilde{\tau}_{-i} = (\tilde{\tau}_2, ..., \tilde{\tau}_{nt})$ within the environment. It is crucial to note that agents lack access to the specific navigational intentions and parameters of their counterparts. This setup, characterized by individual decision-making in the presence of others without explicit communication, highlights the essence of social navigation.

The social robot navigation problem can be defined in two separate classes: considering prediction and navigation in both coupled or decoupled ways [8]. Specifically, the decoupled approach views prediction inputs as distinct information streams used to fulfill robot objectives, while the coupled method integrates agent predictions directly into the ego-robot's decision-making process.

2.1 Decoupled Prediction and Planning methods

A few trends that can be identified for decoupled methods are: 1) treating humans as dynamic, non-reactive obstacles; 2) navigation with interaction-agnostic models for uncertainties; and 3) navigation with hand-crafted social norms.

2.1.1 Dynamic, Non-responsive Obstacles

Early initiatives like RHINO [9] and MINERVA [10], pioneering robotic systems deployed in museums in Germany and the USA during the late '90s, marked significant advancements in robotic tour guide applications by interacting with human visitors and handling numerous guide requests. These robots, alongside successors like Rackham [11] and Robox [12], demonstrated autonomous navigation and service capabilities, further encouraged by competitions like the AAAI Mobile Robot Challenge [13]. Despite their achievements, these systems ignored the human element, treating humans as static obstacles, without considering their social behaviors or interactions. This approach, while effective for collision avoidance, led to less intuitive robot behaviors, often disrupting human pathways. This highlights the need for incorporating models that account for human behavior and uncertainty in navigating crowded environments.

2.1.2 Probabilistic Deduction of Uncertainties

The initial deployment of robots in crowded social settings highlighted the need for improved navigation strategies that account for the unpredictable nature of such environments. This led to a new focus on developing methods to navigate uncertainty. Innovations include a control framework that uses predictive uncertainty in decision-making [14], and Thompson et al.'s probabilistic human motion model aimed at better motion planning [15]. Although these approaches advanced the modeling of uncertainty, they generally viewed humans as solitary, non-interactive figures, neglecting the dynamics of group movement. This often led to an overestimation of potential collisions, contributing to what's known as the "freezing robot problem," where robots halt, unable to identify safe paths.

2.1.3 Formulated Social Norms

The push for social awareness in robot navigation has led to the development of motion planning frameworks that adhere to social norms and proxemics, aiming to increase human comfort. Inspired by proxemics theory [16], some research focuses on creating motion plans that respect personal space. Additionally, there's a trend towards learning social navigation cues from human behavior. Luber et al. used navigation prototypes learned from demonstrations to create dynamic cost maps that reflect both tangible and perceived aspects of human movement [17]. Techniques like Inverse Reinforcement Learning (IRL) have been employed to infer human navigation goals and integrate them into robot path planning [18], fostering more natural and human-like robot movement.

2.2 Coupled Prediction and Planning Methods

In active research fields such as autonomous driving, robotics, and manipulation, robots must be able to handle human encounters successfully to avoid unpleasant experiences. Specifically, from experiences regarding safety from collisions to ease of navigation, various challenging scenarios must be considered for the co-existence of robots and humans. Moreover, human intentions must be known for predicting different safety hazards. Therefore, knowledge of human intentions can be derived from future trajectories as an anticipatory metric for robots to not disrupt the flow of public crowds [19]. This type of coupled method is growing rapidly with various methods to implement social navigation which can be split into four sectors: 1) dynamic obstacle-based; 2) Deep Reinforcement Learning-based (DRL); 3) control-based optimizations and a newer field of 4) Language and Vision-Language model-based

2.2.1 Dynamic Obstacle-Based

Dynamic obstacle-based methods often consider different objects within the surrounding environments by tracking/isolating the region of interest to predict the obstacle trajectories and plan to avoid them [20]. With an increasing interest in Recurrent Neural Networks (RNNs) for sequential prediction, Trajectron++ is a novel method for jointly predicting the trajectories of all dynamic agents within the observed scene through graph-structured RNNs [21]. Methods like STG-DAT [22] implement a Generative Adversarial Network (GAN) to simulate possible futures and generate trajectories based on the relation within the graph network and the probability that it is real [23].

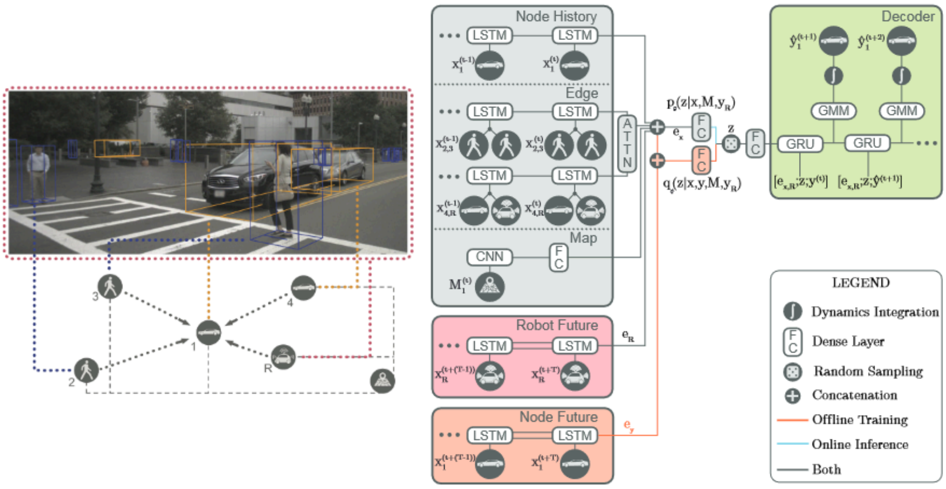

Figure 1: Trajectron++'s directed spatiotemporal graph and network architecture [21]

Some approaches use Occupancy Grid Map predictions (OGMs) to predict a dynamic risk density within a risk map that is computed from the occupancy and velocity field from LiDAR scene scans [24]. Relatively, there has been a promising extension from this work to use Spatiotemporal Occupancy Grid Map predictions (SOGMs) instead of OGMs to use space and time within OGMs to utilize annotated 3D LiDAR points for robot navigation [25]. This study implements an extensive pipeline that allows for the mobile robot to improve its navigational performance over time through self-supervised learning. By treating the observed LiDAR point clouds as a four-category entity, the network can predict future SOGMs of dynamic agents and organically avoid collisions.

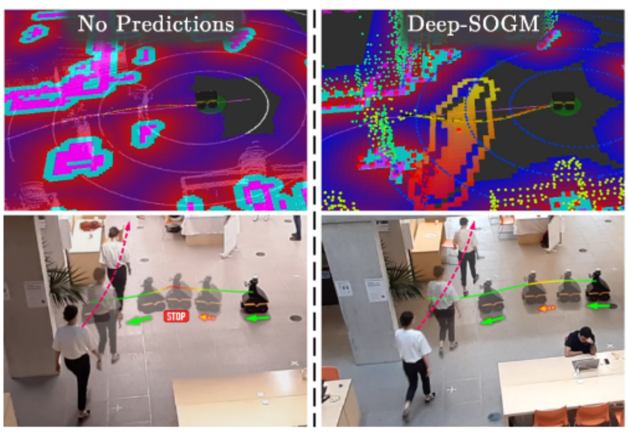

Figure 2: **Left;** Inefficient Social Navigation without predictions, **Right;** Efficient Social Navigation with SOGM predictions [25]

2.2.2 Deep Reinforcement Learning-Based

DRL methods focus on learning navigation policies that maximize the robot's ability to be socially cooperative [26], [27], [28]. More specifically, the robot seeks to maximize the designed reward function of avoiding crowds by minimizing the possibility of collisions. There also has been work to implement attention-based-RL or LSTM-RL networks to introduce an arbitrary number of surrounding agents [29]. Some common DRL algorithms for mobile robot navigation are Deep/Double Deep Q-Network (DQN/DDQN) [30], Proximal Policy Optimization (PPO) [31], and Soft Actor Critic (SAC) [32] which all have their strengths and weaknesses when it comes to training an optimal navigation policy. In terms of continuous actions, PPO is a good benchmark to start from due to its simple yet efficient nature in converging a variety of advanced tasks. Successor to the Trust Region Policy Optimization (TRPO) algorithm, PPO is a policy gradient method that revolves around small incremental policy updates through the ratio $r_t(\theta)$ of old and new policy updates. TRPO's Conservative Policy Iteration (CPI) is maximized with the surrogate objective of:

$$L^{CPI}(\theta = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta)\hat{A}_t \right] \tag{2}$$

Where π_θ is a stochastic policy and $\hat{A}_t$ an advantage estimator function at timestep t, with $\hat{\mathbb{E}}_t[\cdots]$ indicating the empirical average over a finite batch of samples. To prevent excessive policy updates, the probability ratio must be clipped to penalize ratios that move away from one:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right] \tag{3}$$

This removes the incentive for moving the $r_t(\theta)$ outside of the interval of $[1-\epsilon, 1+\epsilon]$ where ϵ being a hyperparameter of 0.2, employs an objective that maintains exploration and exploitation of advantageous policies. Further modifications can be done to augment the objective for general-purpose optimization which involves adding an entropy bonus $S[\pi_\theta](s_t)$ to the surrogate loss function or an advantage estimator:

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \quad (4)$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (5)$$

$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (6)$$

Despite the advances within DRL algorithms and methods, most suffer from the all-in-one nature of neural network policy deployment and can be difficult to troubleshoot issues, tune, and interpret. Also, there are challenges regarding being only able to learn from experience. If the encounter has never been learned, then the prediction of trajectories or states would be divergent from the feasible future [33].

2.2.3 Control-Based

Control-based optimization focuses on obtaining the best possible action the robot can take over a finite control horizon using simple velocity models. More recently, there have been advancements in providing better models for social navigation purposes, from state-of-the-art human trajectory prediction models such as Social-GAN [34] to Social-LSTM [22]. These models combined with MPC have shown significant potential for real-world application in terms of their ability to allow for a safer and predictable performance [35].

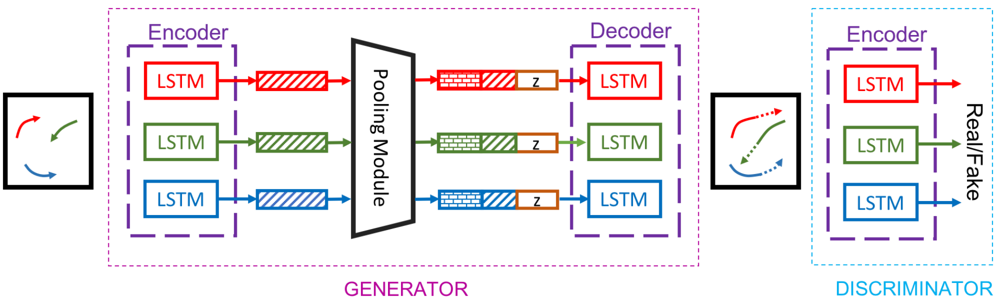

Figure 3: Social GAN's Generator/Discriminator Architecture [23]

2.2.4 Language and Vision-Language Model-Based

This emerging field leverages the advancements in Large Language Models (LLMs) and Vision-Language Models (VLMs) for robotic navigation and planning through natural language processing and visual understanding. Unlike traditional methods that rely on explicit annotations and supervised learning, this approach utilizes pre-trained models capable of interpreting complex, natural language instructions and visual cues without requiring domain-specific fine-tuning or annotated datasets. An example of this is the LM-Nav [36] by Shah et al, which integrates pre-trained navigation (ViNG), image-language association (CLIP), and language modeling (GPT-3) technologies. This system demonstrates the potential for robots to perform long-horizon navigation in complex environments based on high-level, natural language commands. This approach not only simplifies the interface for human-robot interaction but also enhances the robot's ability to generalize from vast, unannotated data sources, thereby reducing the reliance on expensive, labor-intensive data labeling processes.

Figure 4: LM-Nav [36] takes a set of raw observations from the surrounding environment in visual form, devising a set of plans using LLM, VLM, and VNM. This setup allows LM-Nav to effectively follow complex textual instructions using only visual data, without the need for model fine-tuning.

2.3 Social Navigation

Furthermore, to effectively achieve seamless navigation in social settings, specific challenges must be defined: 1) Complexity of Human Behavior, 2) Heterogeneity and Stochasticity of Human Behavior, and 3) Separation of Human Trajectory and Intention.

2.3.1 Complex Human Behavior

The first problem, understanding the complexities of human behaviors is a multi-faceted problem due to the variety of internal and external factors, such as personal intentions, interactions with others, social norms, and the environmental context. Most of such factors are not directly observable and must be inferred from often noisy perceptual cues of contextual information [37]. In addition, depending on the approach used for prediction, incorporating contextual cues such as current position or velocity becomes difficult [37]. Contextual information is essential in most state-of-the-art approaches such as planning methods as they require detailed semantic information of their surroundings to predict the future state of target agents [37]. Therefore, current methods pose limitations in effectively specifying the present scenario for navigation.

2.3.2 Heterogeneity and Stochasticity of Human Behavior

The second problem of stochasticity within human behaviors is an area of active research to propose a general model for predicting human behavior in all circumstances. By learning a multi-task prediction model on presented trajectories [38], a study was able to achieve an adaptable predictive system for human-robot collaboration systems. Furthermore, the latter problems allow for a more specific scope of combining trajectory and intention prediction to encapsulate all important aspects of human behavior.

2.3.3 Human Trajectory and Intention

The combination of human intention and trajectory to develop a flexible predictive system is also described in the previously mentioned study [38]. However, the approach is still limited to user-specified tasks and is unable to learn and adapt on its own. Therefore, by adjusting similar work in models that couple both trajectory and intention prediction, it could be implemented into an end-to-end mobile robot navigational pipeline that could adapt to different environments [39]. In essence, addressing the three challenges will contribute to improving contextual understanding of dynamic environments and their semantics [37]. This will significantly increase the adaptability of autonomous robots and vehicles in diverse scenarios, including accidents and unexpected human behaviors.

2.4 Conclusion of Literature Review

Moving forward, the limitations associated with dynamic obstacles and RL methods will be considered to improve mobile robot navigation in social environments. A notable limitation of dynamic obstacles lies in the inherent unpredictability and complexity of their movements within social settings, which are significant challenges for current navigation systems in accurately forecasting trajectories and adjusting paths in real time. Furthermore, RL methods, while effective in learning complex behaviors, often struggle with long training times, sample inefficiency, and difficulty in generalizing learned policies across varied and unpredictable environments. These challenges are particularly pronounced in dynamic social settings, where human behavior introduces a high degree of variability. These limitations will be explicitly addressed in the next sections, through the exploration of the current pipelines of SOGM predictions and RL navigation policy training with Large Language Models, aiming for advancements in the adaptability and efficiency of autonomous social robot navigation strategies.

3 Methodology

The main objective of this thesis is to provide a benchmark using a state-of-the-art trajectory prediction module [25] and to develop a simulation environment in Isaac Gym with LLM reward tuning. In the first half of this section, we will go over the extensive benchmarking from software to hardware done for the trajectory prediction module based on a series of foundational studies for using SOGMs. In the second half, we will delve into the details of implementing Omniverse Isaac Gym for mobile robot navigation tasks and the LLM reward function tuner, Eureka [40]. A sub-optimal model was produced for testing the trajectory prediction module using an NVIDIA GeForce RTX 3060. Isaac Gym policies were trained using an NVIDIA GeForce RTX 3070 TI with an average training time of 2-3 hrs for an optimal policy.

3.1 Occupancy Grid Maps for Trajectory Prediction

Trajectory prediction using occupancy grid maps function by predicting the probability of occupancy distribution over a 2D grid at individual future time steps. This is limited in terms of surrounding data interpretation due to the latent domain being constrained within a 2D environment. To interpret and transfer useful 3D features for OGM trajectory prediction, we can introduce both spatial and temporal patterns within the OGMs by stacking each 2D grid map to generate 3D SOGMs. This method is advantageous for multimodal predictions due to its point-centric background of using LiDAR point cloud segmentation instead of specific objects, where the critical scene data is largely dependent on correctly identifying certain objects. By treating the robot's environment as a single entity, the robot can learn better navigational decisions even with uncertain environment states using the predicted trajectories of dynamic obstacles.

To benchmark this method, the research detailed in papers [3], [25], [41] will be outlined to explain the implementation of Spatio-temporal Occupancy Grid Maps (SOGMs) for trajectory prediction within autonomous indoor navigation systems, utilizing unstructured 3D point clouds. Central to this approach is the application of Kernel Point Convolution (KPConv) [4], a technique designed to directly process point cloud data without the need for pre-structuring it into a grid format. This method enhances spatial data applicability by integrating Iterative Closest Point (ICP)-based Simultaneous Localization and Mapping (SLAM) algorithms for the creation of initial occupancy maps. These maps are then refined and interpreted through automated annotation processes, incorporating mathematical morphology for data enhancement and a triaging system for semantic filtering. Such filtering prioritizes computational efforts toward analyzing permanent structures and the ground, crucial for accurate localization and efficient path planning in hardware implementations.

3.1.1 Dataset

The network is trained on cumulative data from three hand-crafted sources; UTIn3D-Hall, UTIn3D-Atrium, and Sim 50% data which are all open-sourced on their GitHub repository [42]. UTIn3D-Hall has over 60,000 frames with over 6000 annotated points with 69% of the frames containing dynamic points. UTIn3D-Atrium has over 70,000 frames with over 7000 annotated points with 37.7% of the frames containing dynamic points. The simulated data has not been explicitly mentioned but exists on their old repository as mentioned.

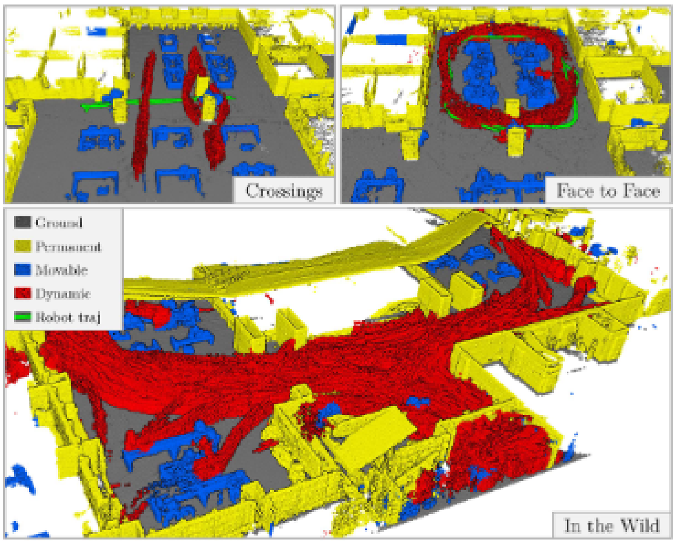

Figure 5: Different annotated frames of the UTIn3D dataset, robot trajectories are in green, and dynamic points are in red [25].

3.1.2 Data Preprocessing

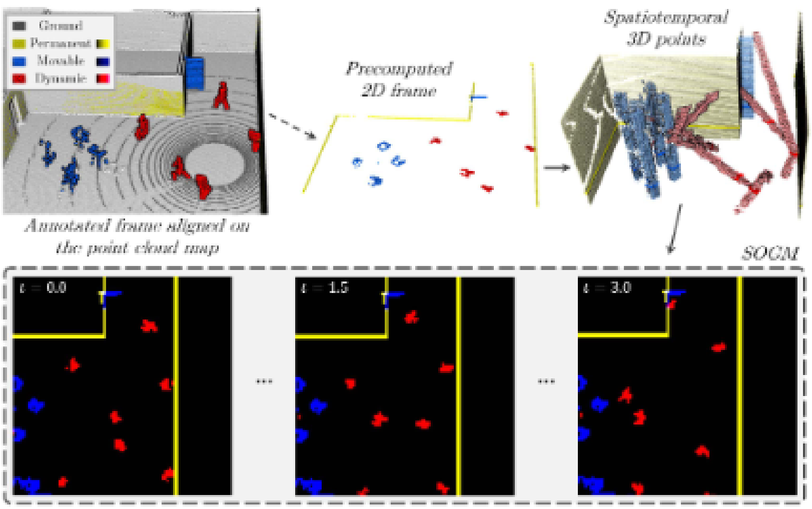

Figure 6: Each of UTIn3D Annotated LiDAR frames are saved and subsampled to a 2D Grid which are later stacked according to their timestamps to create 3D-grid SOGMs [25].

The architecture first requires the navigational environment to be mapped completely to optimize for segmentation and localization signals during runs. This is achieved through a raytracing-based SLAM with loop closures using the Open3D library [43]. The two main UTIn3D datasets are comprised of lidar frames, trajectories computed, and annotations which allows the bypass of reproducing initial mapping sessions and annotations. With the annotated input LiDAR frames, 2D point cloud structures are computed and saved for each of the LiDAR frames that are subsampled and noise-filtered to a grid size of 3cm. During training, the 2D structures of each frame are stacked into a 3D grid representation according to their timestamps with augmentations. This is an SOGM structure of spatial resolution dl_{2D} = 12 cm and temporal resolution dt = 0.1 s. Finally, two of the three channels; 1) Permanent and 2) Movables are merged for all timesteps of the SOGMs since they are static which helps the network learn to complete partially seen static objects.

3.1.3 Kernel Point Convolution

The merged and annotated SOGMs in Figure 6, along with an array of other point cloud features are fed into the Kernel Point Convolution Network for predicting point-wise labels [4], shown in Figure 7. This step is crucial for providing ground truth data as the network now provides ground truth signals for KPConv to predict point-wise labels for random LiDAR frames during inference.

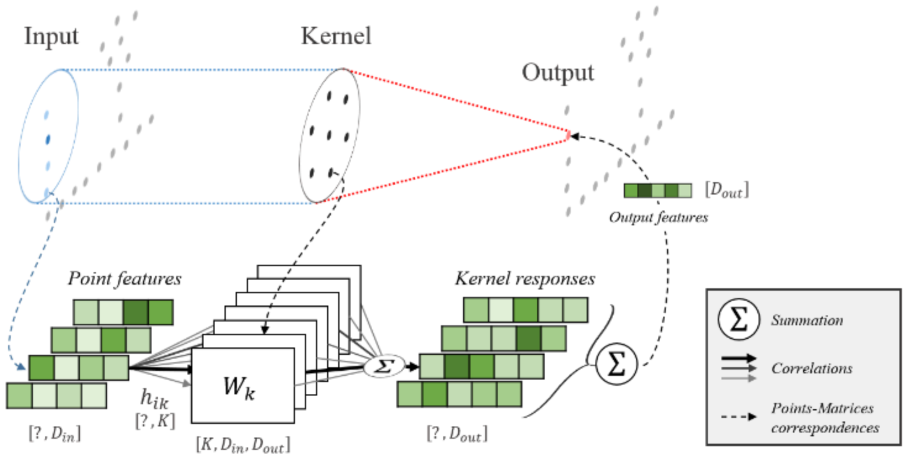

Figure 7: Kernel Point Convolution [4]; input points are not necessarily aligned with kernel points with varying quantities. Therefore, each point feature of the input must be multiplied by all the kernel weight matrices, concerning a correlation coefficient h_{ik}.

Derived from a general definition of a point convolution similar to an image convolution:

$$(F * g)(x) = \sum_{x_i \in N_x} g(x_i - x) f_i \tag{7}$$

KPConv employs a unique kernel function g, which operates on neighbor positions $\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}$ within a 3D space defined by a radius r. For any point $\mathbf{y}_i$, the kernel function is defined as:

$$g(\mathbf{y}_i) = \sum_{k<K} h(\mathbf{y}_i, \tilde{\mathbf{x}}_k) \mathbf{W}_k \tag{8}$$

where $\tilde{\mathbf{x}}_k$ are kernel points, $\mathbf{W}_k$ are associated weight matrices, and h is the correlation between $\tilde{\mathbf{x}}_k$ and $\mathbf{y}_i$, given by:

$$h(\mathbf{y}_i, \tilde{\mathbf{x}}_k) = \max\left(0, 1 - \frac{\|\mathbf{y}_i - \tilde{\mathbf{x}}_k\|}{\sigma}\right) \tag{9}$$

Here, σ is the influence distance of the kernel points. This method is distinguished by its adaptability to the observable point density and the straightforward nature of its gradient backpropagation, in contrast to more complex correlations like the Gaussian.

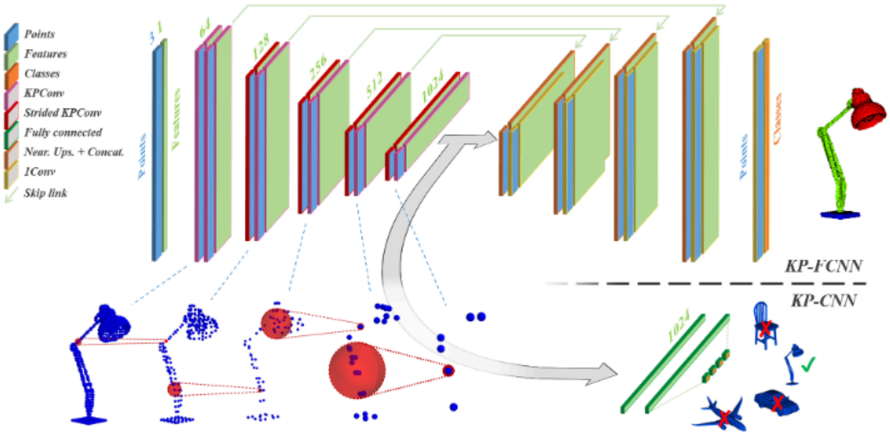

Figure 8: **Top**; KP-FCNN for point segmentation and labeling, **Bottom**; KP-CNN for classification [44]

3.1.4 Feed-Forward 3D-2D Network

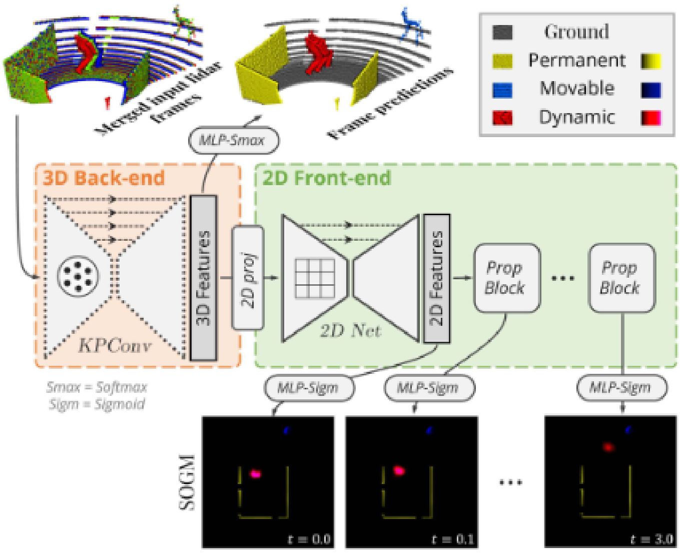

Figure 9: Full 3D-2D Feed Forward architecture going from merged input LiDAR frames to point-wise labels and SOGM trajectory predictions [25]

For the 3D-Backend, the specific architecture is the KP-FCNN as seen in Figure 8 which is a fully convolutional network for segmentation. The 5-layer encoder convolutional blocks are similar to bottleneck ResNet blocks and the decoder uses nearest upsampling to obtain the final point-wise features. Skip links are also used to transfer features between intermediate layers of both the encoder and decoder which also employ a unary convolution for upsampling. This is particularly necessary for handling noise and disruptive points.

In the context of KPconv's point-wise labels, by providing signals that allow more specific point features to be passed down to the 2D part of the network, it aids in training the network. The output is also used for both global planning and mapping in the pipeline. Along with the ground truth SOGMs, the network input array also contains 0.3 seconds of raw LiDAR frames that have been aligned and merged. Other than providing point-wise labels, the 3D point features are passed to the 2D-front-end with a grid projection using the same spatial resolution dl_{2D} as the SOGM. Then, it creates the 2D feature map by averaging features within cells of a grid. This map is processed by a U-Net architecture with three levels of two ResNet layers each, enhancing sparse information across the grid. The enhanced feature map predicts the initial SOGM time step, followed by propagation through blocks with two ResNet layers each, predicting subsequent SOGM time steps. The process aims for predictions up to 4 seconds, totaling 41 time steps. Despite redundancy in predictions for permanent and movable objects, this approach is maintained to improve class interaction learning and spatial awareness.

The network's loss function combines both the 3D and 2D layers. The total loss function, L_{tot}, is defined as:

$$L_{\text{tot}} = \lambda_1 L_{3D} + \lambda_2 \frac{\sum_{k<nT} L_{2D}^k}{nT} \tag{10}$$

where $\lambda_1 = 1.0$, $\lambda_2 = 10.0$. L_{3D} is the standard cross-entropy loss used in the original KPConv network, and L_{2D}^k is a binary cross-entropy loss applied to the layer k of the SOGM predictions:

$$L_{2D}^k = \sum_{i \in M_k} \text{BCE}(x_{k,i}, y_{k,i}) \tag{11}$$

where $x_{k,i}$ is the network logit at the pixel i of the time-step layer k in the SOGM, $y_{k,i}$ is its corresponding label, and BCE for binary cross-entropy. During inference, the predicted future SOGMs are converted into risk maps that are decoupled into static and dynamic risks where static risks are taken from the permanent and movable SOGM channels and dynamic risks from the dynamic channel. The dynamic risks are diffused in time and stacked into a single cost map with a single layer of static risks since it does not move. From here, the Timed Elastic Band (TEB) path planner receives the costmap to finally navigate the robot towards a goal avoiding dynamic/static risks.

3.2 Omiverse Isaac Gym Simulator For DRL

As our scope is to implement the trajectory prediction module into a separate navigation pipeline more suited for effective mobile robot social navigation using DRL, a simulator was developed within the latest Omniverse Isaac Simulator's built-in Gym environment utilizing OpenAI's Gym API and High-performance RL library; rl_games. Important features are outlined:

- **Simulator:** Omniverse Isaac Gym Environment (OIGE) [45].
- **RL Algorithm:** Proximal Policy Optimization (PPO) [31].
- **Observations:** 2D scan, distance to goal, angle to goal, linear and angular velocity.
- **Action Space:** Clearpath Jackal's [46] base linear and angular velocity.
- **Task:** Single-Robot waypoint navigation with dynamic/static obstacle avoidance
- **Reward Function:** Based on the difference between the previous and current distance to the goal, the agent is penalized if the current distance is larger than the previous distance. In addition, a minimum LiDAR depth threshold is specified to penalize the agent if it detects any obstacles closer to that threshold. The agent is also rewarded for reaching the goal and moving at all times without stopping.

3.2.1 Environment Design

In classic robotic simulators such as ROS, training a DRL policy requires heavy CPU usage. In contrast, OIGE leverages GPUs for parallelizing the training process, significantly accelerating DRL policy training. This capability allows OIGE to simulate a vast number of environments in parallel as shown in Figure 10, offering a substantial throughput increase over CPU-based simulations. The ability to sample a large number of environment states and actions in parallel drastically reduces the time required to train complex policies.

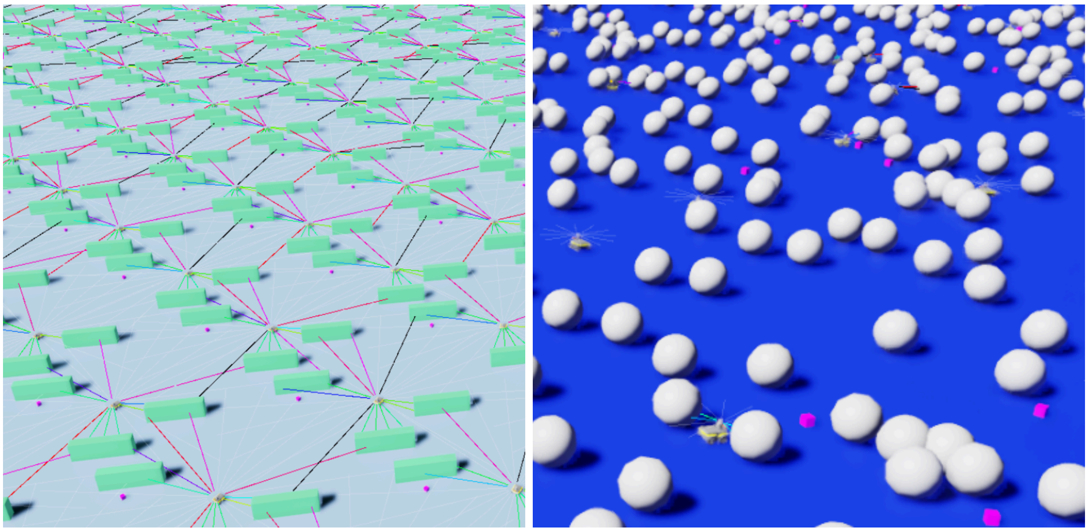

Figure 10: **Left;** Multiple training environments visualized in OIGE for waypoint navigation with static obstacles. **Right;** waypoint (pink cube) navigation with dynamic obstacles as moving spheres.

As for our Jackal robot, we mounted a 2D LiDAR sensor with 12 lasers for collecting depth information, the max and min range of the scans are 1 m and 0.1 m respectively. The short range of the 2D sensor is to ensure less noise while training but can be increased in the future for different tasks. The environment is built to be highly customizable and provides a significant baseline simulator for mobile robot navigation tasks.

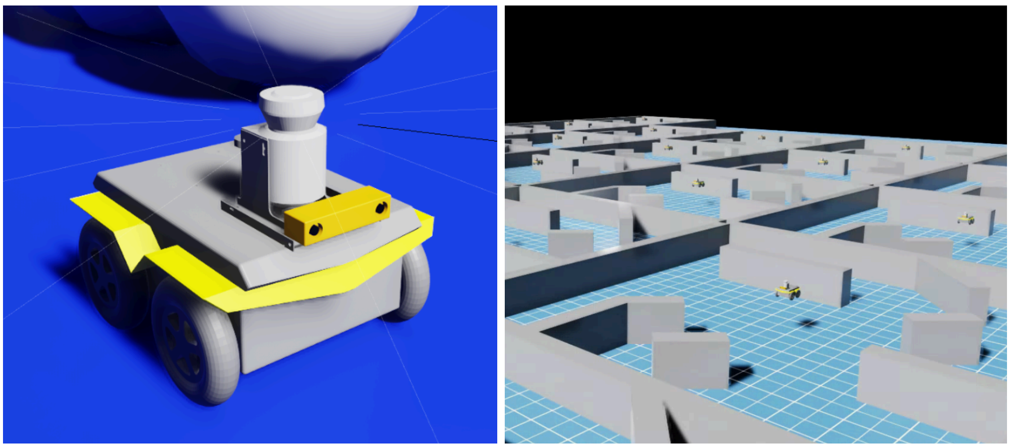

Figure 11: **Left;** Clearpath Jackal UGV Mobile Robot in USD format, provided within OIGE, wheel joints are configurable but the LiDAR sensor has been implemented from the existing APIs within Omniverse. **Right;** Individual maze configuration for each environment instance.

3.2.2 Observation Space

Given an agent in a dynamic environment, the observation space S at any time step t can be described as a multi-dimensional vector s_t, which includes both scalar and vector components representing the robot's sensory inputs and its state:

$$s_t = [l_1, l_2, \ldots, l_n, d_{\text{goal}}, \theta_{\text{goal}}, v, \omega] \tag{12}$$

where:

- l_i represents the i-th distance reading from a 2D LiDAR scan, with $i \in \{1, 2, \ldots, n\}$
- d_{goal} is the scalar distance to the goal location.
- θ_{goal} is the angle to the goal relative to the robot's heading.
- v and ω are the robot's current linear and angular velocities, respectively.

3.2.2 Action Space

The action space A is defined as the set of all possible actions the agent can take at any time step t, represented by a_t:

$$a_t = [v', \omega'] \tag{13}$$

where:

- v' is the desired linear velocity.
- ω' is the desired angular velocity.

These actions directly influence the robot's movement and are selected based on the policy being optimized.

3.2.3 Reward Function

The reward function $R(s_t, a_t, s_{t+1})$ evaluates the immediate reward received after transitioning from state s_t to state s_{t+1} due to action a_t, and is defined as:

$$R(s_t, a_t, s_{t+1}) = \alpha(d_t - d_{t+1}) - \beta \sum_{i=1}^{n} \mathbb{I}(l_i < \text{threshold}) + \gamma \mathbb{I}(\text{goal reached}) + \delta \mathbb{I}(v' > 0 \vee \omega' > 0) \tag{14}$$

where:

- $\alpha, \beta, \gamma, \delta$ are weighting coefficients.
- $\mathbb{I}(\cdot)$ is the indicator function, returning 1 if the condition is true, and 0 otherwise.
- d_t and d_{t+1} are the distances to the goal at times t and $t+1$, respectively.

3.2.4 PPO for Continuous Action Navigation

PPO seeks to fine-tune the policy $\pi_\theta(a_t|s_t)$ of the Jackal robot, maximizing the expected rewards over time by navigating towards a target while avoiding obstacles. With continuous action space of linear and angular velocities, there is a nuanced approach to policy optimization. As mentioned, PPO employs a specially designed objective function that leverages a clipped surrogate objective which controls the magnitude of policy updates and ensures smooth learning progress:

$$L_{\text{nav}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta) \hat{A}_{\text{nav},t}, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\text{nav},t}) \right] \tag{15}$$

In this context:

- $r_t(\theta)$ quantifies the ratio of the likelihoods under the new policy versus the old, capturing how the policy's preference for an action at state s_t evolves.
- $\hat{A}_{\text{nav},t}$ represents an advantage function estimator for navigation, measuring the relative benefit of taking action a_t in state s_t over the average. It incorporates navigation-specific rewards and penalties, such as the importance of maintaining a

safe distance from obstacles, the efficiency of path following, and the urgency of reaching waypoints.

- ϵ is a critical hyperparameter that dictates the clipping threshold, moderating the extent of policy updates to avoid destabilizing the learning process.

3.3 LLM Reward Function Tuning in OIGE

The Eureka framework, detailed in the Eureka paper [7], introduces a method for designing reward functions using Large Language Models, such as GPT-4. As mentioned, this approach leverages LLMs' capabilities for zero-shot generation and code writing to evolve reward functions through evolutionary optimization, without requiring task-specific prompts or predefined templates.

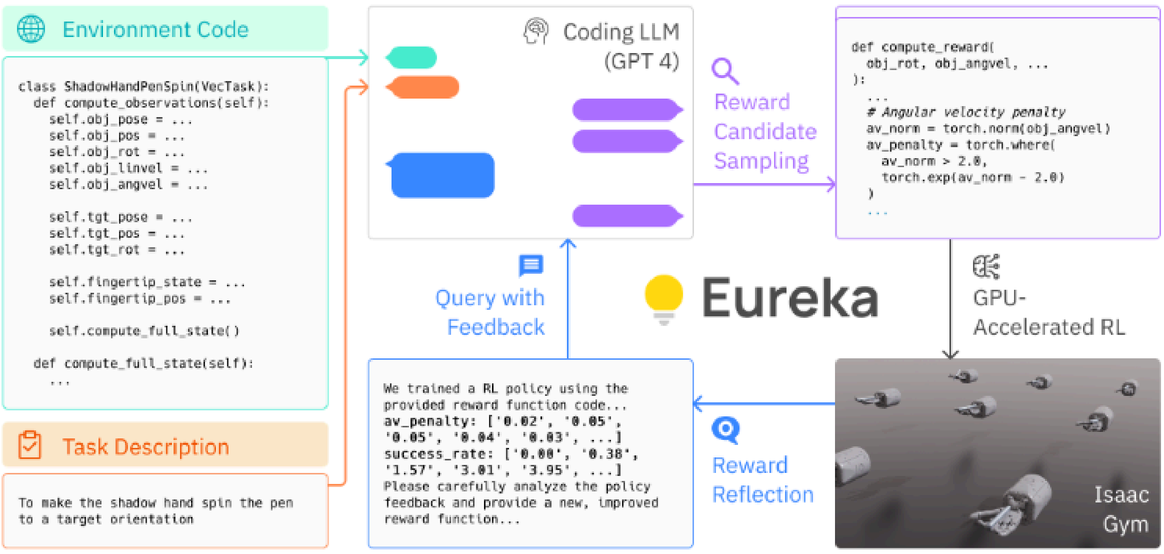

Figure 12: Eureka's iterative pipeline of providing human-level reward functions via LLMs [7]

Applied in Isaac Gym, Eureka significantly enhances the development of complex skills in reinforcement learning (RL) tasks. It has proven to outperform expert-designed rewards in 83% of tasks across 29 RL environments, leading to a 52% average improvement.

Eureka also facilitates a novel method for reinforcement learning from human feedback (RLHF) by integrating human inputs directly into reward tuning, improving both reward quality and safety without model retraining. One notable application is training a simulated Shadow Hand to perform pen-spinning tricks, demonstrating Eureka's potential to advance robotic control and manipulation through effective reward function design.

3.3.1 Eureka with Mobile Robot Navigation

Due to Eureka being implemented in an older version of Isaac Gym (Isaac Gym Preview 4) [45], the repository [47] has been heavily modified to work with the new OIGE. The proposed implementation is functional with a few technical limitations like speed and automated evaluation which will be detailed in the limitations section. Figure 13 outlines the modified Eureka pipeline according to our navigation task.

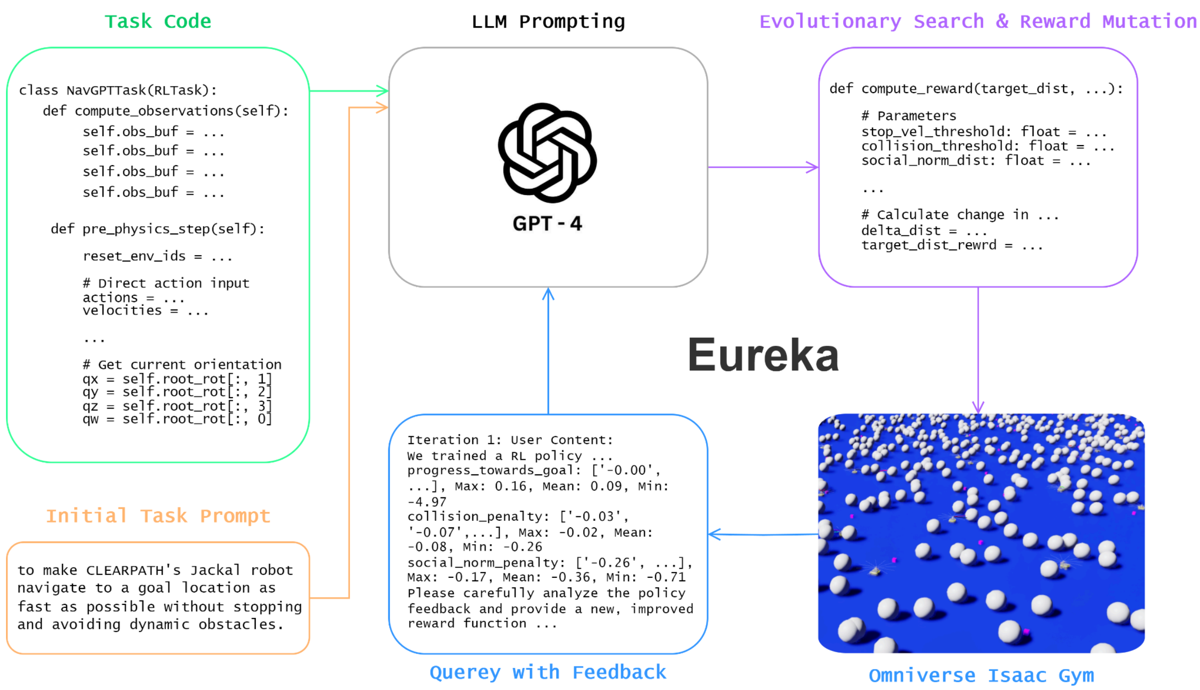

Figure 13: Modified visualizations according to how the navigation task is trained using Eureka. Note that the initial task prompt is different and the LLM will initialize using this starting point as seen in the reward mutation stage

3.3.2 Reward Tuning Dynamic Obstacle Task

For the task of waypoint navigation with dynamic obstacles, we implemented an environment shown in Figure 13, where spheres are treated as humans that move. Also, we provide the hand-crafted reward function to Erueka for correlation calculation between human-made and generated reward functions. Having a single executable code being output from the LLM will allow it to propose a new and improved reward function from an existing one based on textual feedback. This process iterates, taking the best reward from a previous iteration to generate more samples. This iterative optimization of the reward function continues until the chosen number of iterations is reached.

3.3.3 Evaluation and Integration

After undergoing multiple iterations of reward function tuning and improvement, the final stage in the Eureka pipeline involves evaluating the optimized reward functions and integrating them into the targeted RL environment for further testing and validation. This ensures that the evolved reward functions not only theoretically improve performance but also practically enhance the RL model's ability to learn and execute tasks more effectively. For a deeper look into the final evaluation and integration process within the context of mobile robot navigation, particularly focusing on navigating through dynamic obstacles:

- **Performance Metrics:** The final evaluation process involves testing against several key performance metrics. These include a metric that is indicative of “success” in task completion, efficiency (time or steps taken to complete the task), safety (avoidance of obstacles or unsafe states), and generalization across varying environmental conditions. This multi-metric assessment ensures that the reward function promotes balanced and robust skill development.

- **Comparative Analysis:** The evolved reward functions are compared against baseline reward functions, including the initial human-designed rewards and possibly other algorithmically generated rewards. This comparison highlights the relative improvements and identifies areas where the evolved reward functions excel or require further refinement.

4 Results and Discussion

There are two parts to the experiments that have been conducted. The first was to train our trajectory prediction model using the SOGM network which was tested on both the paper's Gazebo simulator and our real Clearpath Jackals. The second was training the Jackal on waypoint navigation with Eureka for dynamic obstacles using our OIGE simulator.

4.1 SOGM Prediction Training

To validate the training pipeline, we trained a sub-optimal model by modifying our variable batch size targeting $B = 1$, averaging 8.5 k points per batch with 1000 epochs, learning rate of 1e-2, momentum=0.98 and 500 steps per epoch with all other parameters being identical to the proposed model. This qualitative analysis confirms the validity of the model due to the general direction of the observable dynamic (red) predictions that align with the general path of the person's heading direction.

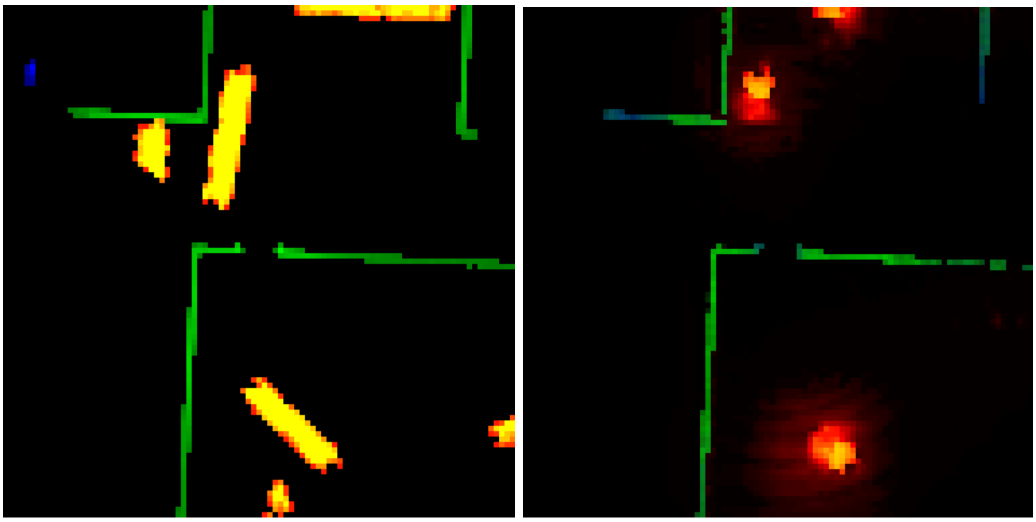

Figure 14: One training scenario result using sub-optimal training hyperparameters (due to hardware constraints at the time) **Left;** ground truth future SOGMs for the simulated path of humans (yellow), permanent walls (green), and movables (blue) produced using the stacked 2D features. **Right;** predictions made by the network for 4s into the future in (red) regions. This is in the general direction of the left image.

4.2 SOGM Network Simulation Experiments

The paper [25] also proposes a simulator that allows for visualization within a Gazebo Environment with predictions available within Rviz. The initial simulation results indicated that the model is unable to predict proper trajectories due to the sub-optimal model. However, as seen in Figure 15, it displays potential for training with better hardware to facilitate a prediction level similar to the paper.

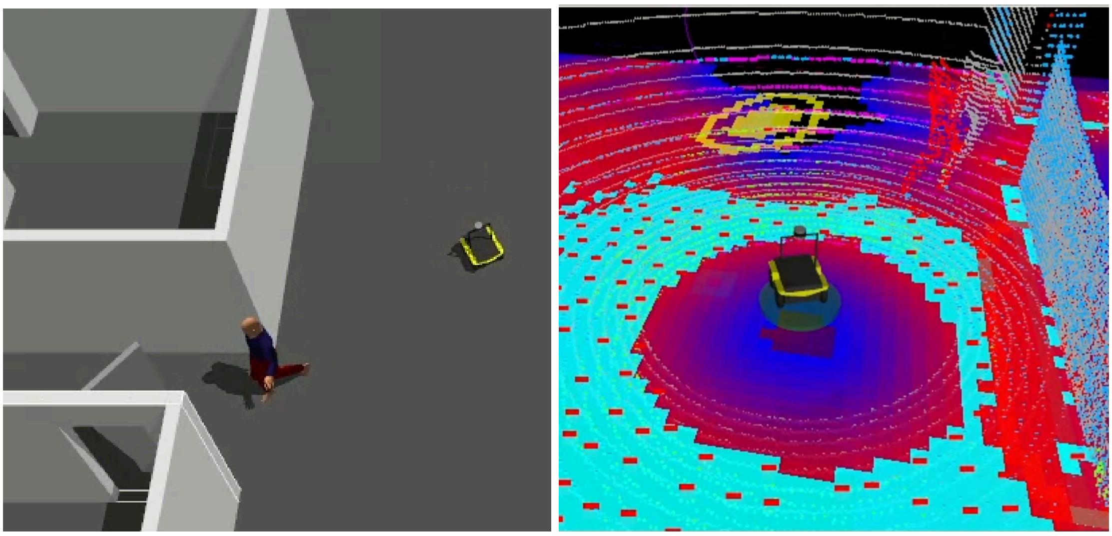

Figure 15: Initial simulation experiment with the sub-optimal model, **Left;** Gazebo environment involving dynamic humans walking around, **Right;** RVIZ visualization of the segmentation and prediction, the red region on the cost map is where the model hypothesizes that there are dynamic risks and trajectory predictions can be seen to be the yellow spot on the map. Again, the model is sub-optimal which results in inaccurate predictions.

Comparatively, the ground-truth results were visualized in Figure 16 to indicate the accuracy of the proposed dynamic obstacle prediction and trajectory planning. This showed effective planning around each person's trajectories.

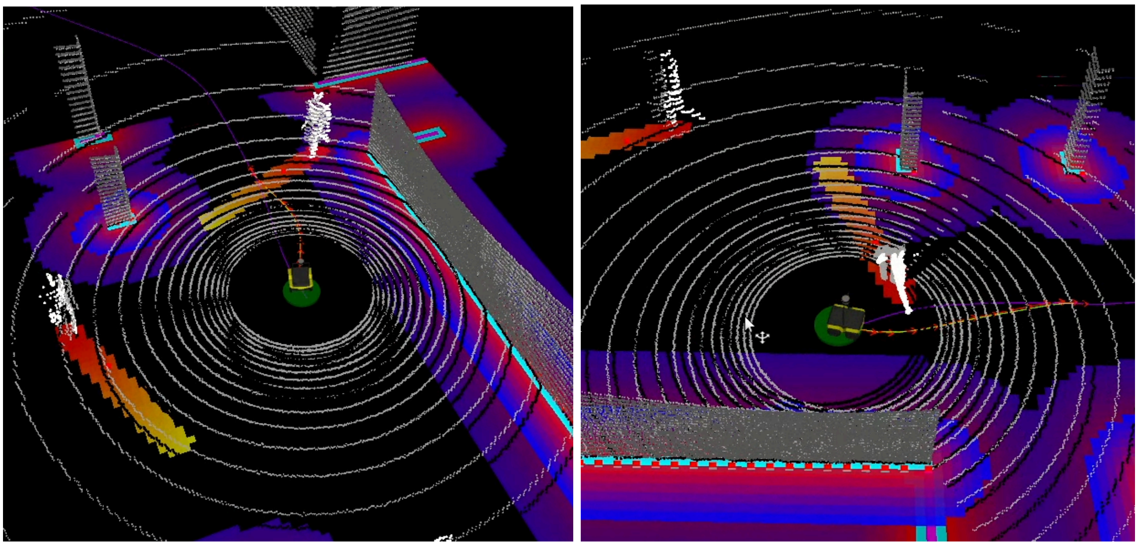

Figure 16: **Left;** TEB navigation with ground-truth SOGMs, previous timestep. **Right;** Future timestep, the jackal already has trajectory risks within the cost map that TEB is trying to minimize linearly. This allows the jackal to avoid the human naturally.

4.3 SOGM Network Hardware Experiments

To further test for functionality and limitations, the trained model was used to perform real-time predictions within a new environment outside of the University of Toronto Myhal Atrium and Hall settings where the paper's proposed model is trained. This provides a benchmark for implementing the model in hardware. Figure 16 shows the predictions being done in real time within our lab environment.

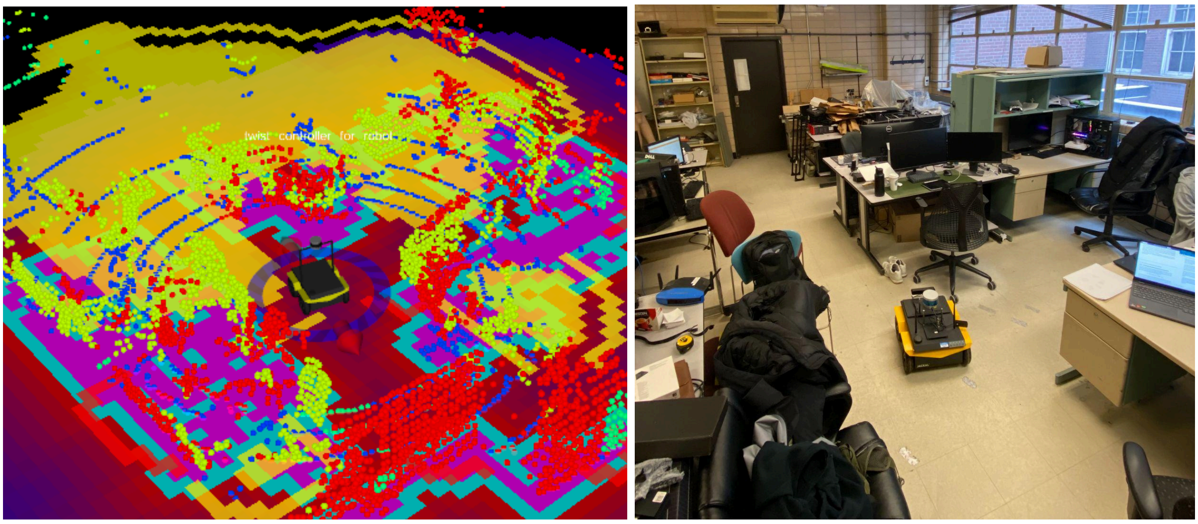

Figure 17: **Left;** Predictions shown within Rviz, the dynamic (red) and permanent (light

green) points are mixed with the trajectory predictions (yellow spot on ground). This indicates that the network's effectiveness is confined to environments closely resembling the training dataset. In contrast, the same model demonstrates a higher degree of accuracy in segmenting classes within a simulation environment. This discrepancy must be considered for the challenge of generalizing the model to different environments, as opposed to the controlled conditions of being within Myhal where it can more precisely differentiate between dynamic and static elements. **Right;** Real Jackal in the MC202 environment.

Overall, The SOGM Network's evaluation highlights hardware limitations and generalization challenges as barriers to performance. Advancements in hardware and model optimization are important for improved accuracy and generalizability. To achieve results similar to the ground-truth predictions, the training, and different hardware experiments must be conducted. The details for future implementations will be mentioned in the future works section.

4.5 Waypoint Navigation with Eureka

To consider the future implementation of the SOGM network in OIGE, we integrated navigation within our simulator populated with dynamic obstacles as shown in Figure 17. This approach aims at developing a policy that could serve as an alternative to traditional planners, such as TEB which is the planner used in the SOGM paper. The goal was to craft a policy that navigates efficiently towards a designated goal with socially compliant and robust behaviors across various scenarios the Jackal robot might encounter.

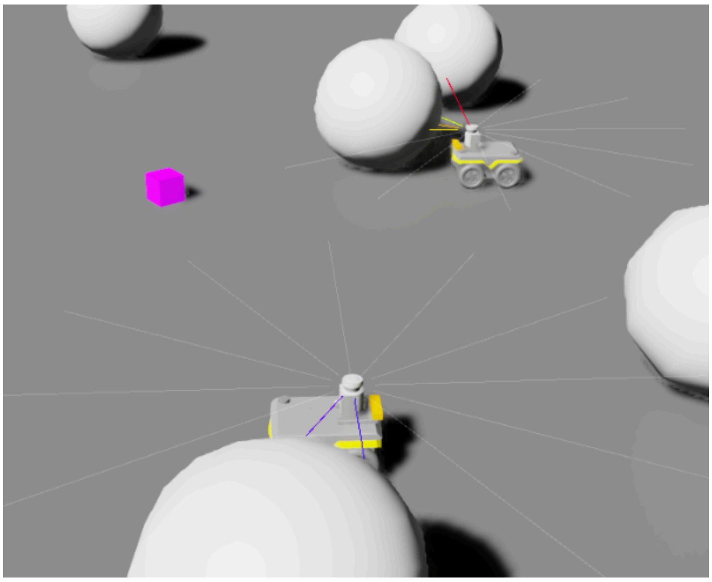

Figure 18: Jackal dynamic obstacle avoidance training environment with waypoints (pink cube)

The training regime for this navigation policy involved 128 parallel environments and a mini-batch size of 1024 over 5000 epochs. With this configuration, we produced a policy capable of navigating towards goals while avoiding dynamic obstacles. Building on the foundation laid by the initial training, the Eureka framework was implemented to further refine and enhance the performance of the developed navigation policy. Eureka was integrated to iteratively improve the policy's effectiveness in dynamic obstacle avoidance and goal-directed navigation. This optimized the navigation policy's performance, ensuring a higher degree of social compliance and adaptability to a wide array of environmental scenarios. With the goal of effective waypoint navigation, two important metrics were defined, a mean negative distance which is recorded after environment reset or 3000 episodes, and a mean correlation value between human and eureka rewards between -1 to 1. The mean negative distance is derived from Eureka's [40] implementation of the fitness function F for the quadcopter Isaac task that must reach a waypoint as well. This can be defined as the policy's success metric. Also, as seen in Figure 19, the harder the task is, the less correlated the Eureka rewards are which is the value being closer to one. The paper also claims a constant increase in performance with each sample and iteration due to the evolutionary search and mutation Eureka is capable of as mentioned previously.

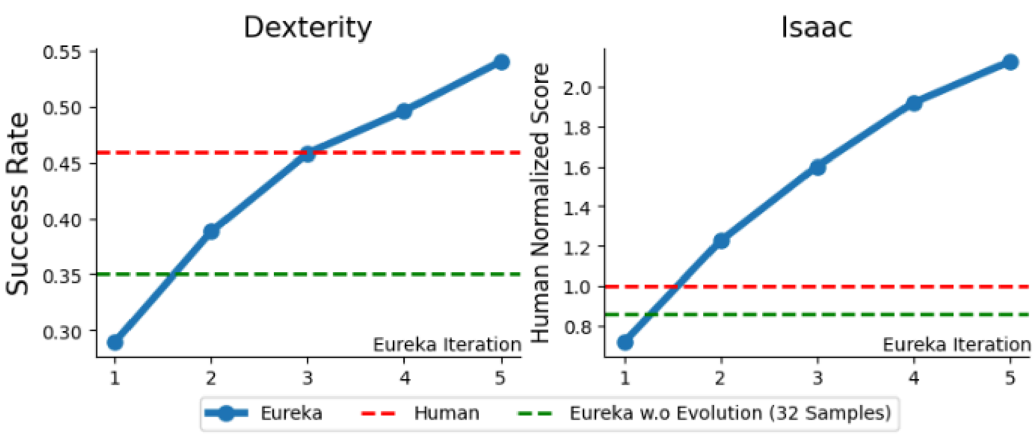

Figure 19: Eureka's Isaac and Dexterity task's ability to improve success metrics specific to each task [40]

4.6 Eureka Outputs and Evaluations

We begin the experiment by first specifying our task description with 4 samples and 2 iterations as a starting point as GPT-4 can produce more executable code with minimum sampling:

```
3 [2024-03-22 12:03:02,660][root][INFO] - Using LLM: gpt-4-0125-preview
4 [2024-03-22 12:03:02,660][root][INFO] - Task: NavGPT                   Task Description
5 [2024-03-22 12:03:02,660][root][INFO] - Task description: to make CLEARPATH's Jackal
  robot navigate to a goal location as fast as possible without stopping and avoiding
  dynamic obstacles. This environment involves a waypoint and dynamic sphere obstacles
  that move randomly in the environment. The robot needs to navigate to the goal
  location abiding social norms without collision. The robot is equipped with a 2D
  LIDAR sensor.
6 [2024-03-22 12:03:02,688][root][INFO] - Iteration 0: Generating 4 samples with
  gpt-4-0125-preview
7 [2024-03-22 12:04:10,431][root][INFO] - Iteration 0: Prompt Tokens: 911, Completion
  Tokens: 3067, Total Tokens: 3978
```

Figure 20: The description specifies many details specific to the task environment. Note that we include the social norm aspect for the LLM to consider when generating the initial reward function for exploration. We also include task-specific details to aid in generating executable code in fewer samples/iterations.

Out of the four samples in the first iteration, the 3rd and 4th samples are executed. The feedback prompts are initiated based on the success and correlation metrics. The best reward function is identified to be the 3rd sample since it is equal to the ID of 2:

```
[2024-03-22 19:27:12,879][root][INFO] - Iteration 0: Code Run 3 successfully
completed training!
[2024-03-22 19:27:20,695][tensorboard][INFO] - No path found after /home/asblab9/-
daniel/eureka_nav/omniisaacgymenvs/runs/NavGPTGPT/NavGPTGPT-2024-03-22_12-05-25/-
summaries/events.out.tfevents.1711123525.asblab9-desktop
[2024-03-22 19:27:28,677][tensorboard][INFO] - No path found after /home/asblab9/-
daniel/eureka_nav/omniisaacgymenvs/runs/NavGPTGPT/NavGPTGPT-2024-03-22_15-53-07/-
summaries/events.out.tfevents.1711137187.asblab9-desktop Best Sample Identification
[2024-03-22 19:27:28,743][root][INFO] - Iteration 0: Max Success:
-3.8836545944213867, Execute Rate: 0.0, Max Success Reward Correlation:
0.19617980521522216
[2024-03-22 19:27:28,743][root][INFO] - Iteration 0: Best Generation ID: 2
[2024-03-22 19:27:28,743][root][INFO] - Iteration 0: GPT Output Content:
Given the task description and the environment observations available within the
`NavGPTTask` class, we can devise a composite reward function that encourages the
Jackal robot to navigate to the goal location swiftly, avoid dynamic obstacles, and
comply with social norms. The reward function can utilize the target distance
(`self.target_dist`), the change in target distance between the current and previous
timestep (`self.prev_target_dists` and `self.target_dist` difference), the LIDAR
data (`self.lidar_data`), and the robot's velocities (`self.jackal_root_linvels` and
`self.jackal_root_angvels`) to formulate these incentives.
```

Figure 21: Eureka feedback for the first iteration. The execute rate being at 0 is a bug due to an inherent system modification. This iteration would have an execution rate of 50% since 2 out of the 4 generated samples ran. This is to do with the timing of recording the execution of each training session which will be detailed in the future works section.

The feedback continues to re-prompt the reward function generated by the best sample. This is to ensure more context to providing positive mutations to the reward function. Also, as seen in Figure 21, all scalar information of the reward metrics are saved and prompted as well to provide scores of the current rewards. This allows signals for adjusting scaling, implementing new reward calculations, and applying normalizations.

```
This script incorporates:
- A reward for moving towards the goal—a larger positive reward for closing the distance more
significantly.
- Penalties for proximity to obstacles detected by LIDAR to enforce obstacle avoidance.
- Penalties for stopping to maintain constant movement.
- A dynamic social norm adherence mechanism that penalizes the robot for getting too close to
obstacles, implying an encouragement for maintaining socially acceptable distances from
dynamic entities.

This composite reward design aims to guide the Jackal robot efficiently towards achieving the
task's objectives under the stated conditions, integrating essential behaviors for successful
navigation.
```

Scalar Reward Feedback from Tensorboard

```
[2024-03-22 19:27:28,744][root][INFO] - Iteration 0: User Content:
We trained a RL policy using the provided reward function code and tracked the values of the
individual components in the reward function as well as global policy metrics such as success
rates and episode lengths after every 500 epochs and the maximum, mean, minimum values
encountered:
progress_towards_goal: ['-0.00', '-0.00', '-0.01', '-0.00', '0.00', '-0.00', '-0.00', '-0.00',
'-0.00', '-0.00'], Max: 0.00, Mean: -0.00, Min: -0.02
collision_penalty: ['-0.07', '-0.10', '-0.12', '-0.35', '-0.08', '-0.09', '-0.13', '-0.09',
'-0.17', '-0.23'], Max: -0.02, Mean: -0.15, Min: -0.42
stop_penalty: ['-0.18', '0.00', '0.00', '-0.01', '-0.01', '0.00', '0.00', '-0.01', '0.00',
'0.00'], Max: 0.00, Mean: -0.01, Min: -0.25
social_norm_penalty: ['-0.42', '-0.23', '-0.47', '-0.82', '-0.39', '-0.35', '-0.41', '-0.32',
'-0.46', '-0.54'], Max: -0.13, Mean: -0.44, Min: -0.99
task_score: ['-5.28', '-6.03', '-5.21', '-5.26', '-4.60', '-5.65', '-5.11', '-5.34', '-5.06',
'-4.25'], Max: -3.88, Mean: -5.06, Min: -6.51
episode_lengths: ['90.00', '672.55', '1004.40', '404.60', '2213.21', '2590.23', '1951.92',
'1802.36', '2635.76', '1959.48'], Max: 5426.55, Mean: 1815.19, Min: 90.00
```

Figure 22: The scalar value feedback for all the reward metrics. Note that these reward metrics are LLM generated and are stored as a dictionary by the reward function.

Continuing to the next iteration, all 4 samples are executed, indicating that the previous reward function has mutated correctly. The performance has also improved as can be seen in Figure 23 with a success of -2.23 meters and 0.025 correlation. This also confirms the intuition of Eureka producing a lower correlation to human reward functions with increasing difficulty in tasks.

```
[2024-03-23 10:05:43,756][root][INFO] - Iteration 1: Max Success: -2.2314419746398926, Execute
Rate: 0.0, Max Success Reward Correlation: 0.025542423867751813
[2024-03-23 10:05:43,756][root][INFO] - Iteration 1: Best Generation ID: 1
[2024-03-23 10:05:43,756][root][INFO] - Iteration 1: GPT Output Content:
Analyzing the policy feedback, we observe that:
```

Tailored feedback for the next iteration

```
1. The value of `progress_towards_goal` is very low and doesn't vary much. This suggests that
the reward component for progress towards the goal is not effectively motivating the robot to
move towards the target. We need to adjust its scale or its temperature parameter to make it
more sensitive to changes in distance.

2. `collision_penalty` and `social_norm_penalty` both show significant impacts on the total
reward, with relatively high magnitude penalties. This indicates they are working as intended
but might be overly punitive, overshadowing the progress reward. Adjusting their scales could
provide better balance.

3. `stop_penalty` varies minimally and includes zeros, suggesting the robot doesn't often
completely stop, or the penalty is not sensitive enough to detect undesired low-velocity
states. It might need fine-tuning or re-scaling.

Given these insights, the improved reward function will adjust the temperature parameters and
scales of these reward components to enhance the training effectiveness:
```

Figure 23: Tailored feedback designed to enhance the next iteration of samples. This is an indicator that the more iterations we provide, the better the reward function.

The final reward function in Figure 24 is relatively simple to interpret as humans. However, if performed manually, the process of comprehending the scaling and normalization factors would have required days dedicated to experimentation with various values to ascertain an optimal solution. On the contrary, Eureka has the potential to iteratively improve to a degree surpassing the effectiveness of the initial reward function, without human intervention, through a sufficient number of iterations.

```
import torch
from typing import Dict, Tuple

@torch.jit.script
def compute_reward(target_dist: torch.Tensor, prev_target_dists: torch.Tensor, lidar_data:
torch.Tensor,
                   jackal_root_linvels: torch.Tensor) -> Tuple[torch.Tensor, Dict[str,
torch.Tensor]]:
    # Parameters
    stop_vel_threshold: float = 0.1
    collision_threshold: float = 0.5  # Minimum safe distance in meters
    social_norm_dist: float = 1.5  # Desired minimum distance to obstacles
    delta_dist_temp: float = 0.5  # Adjusted temperature parameter for progress towards goal
    obstacle_penalty_temp: float = 2.0  # Adjusted temperature parameter for obstacle penalty
    stop_penalty_temp: float = 2.0  # Adjusted temperature parameter for stop penalty

    # Calculate change in distance to goal, more sensitive to movement towards/away from goal
    delta_dist = prev_target_dists - target_dist
    target_dist_reward = (torch.exp(delta_dist / delta_dist_temp) - 1.0) * 5.0  # Increased
scaling factor

    # Detect potential collisions and apply penalty, re-scaled for balance
    min_lidar_dist, _ = torch.min(lidar_data, dim=1)
    close_to_obstacle_penalty = torch.exp(-min_lidar_dist / obstacle_penalty_temp)  # Adjusted
scaling
    collision_penalty = torch.where(min_lidar_dist < collision_threshold,  -0.5 *
close_to_obstacle_penalty, torch.tensor(0.0).to(target_dist.device))  # Reduced impact

    # Social norm adherence, adjusted scaling
    social_norm_penalty = torch.where(min_lidar_dist < social_norm_dist,
                                      -0.5 * torch.exp((social_norm_dist - min_lidar_dist) /
obstacle_penalty_temp), torch.tensor(0.0).to(target_dist.device))  # Reduced impact

    # Penalize stopping, more sensitive to low speeds
    speed = torch.norm(jackal_root_linvels, dim=1)
    stop_penalty = torch.where(speed < stop_vel_threshold, -torch.exp(-speed /
stop_penalty_temp), torch.tensor(0.0).to(target_dist.device))  # Adjusted impact

    # Aggregate the rewards and penalties with adjusted scaling
    total_reward = target_dist_reward + collision_penalty + stop_penalty + social_norm_penalty

    reward_components = {"progress_towards_goal": target_dist_reward, "collision_penalty":
collision_penalty,
                         "stop_penalty": stop_penalty, "social_norm_penalty":
social_norm_penalty}

    return total_reward, reward_components
```

Figure 24: The best reward function generated in the 2nd sample of the 2nd iteration. Note that there is a social_norm_dist parameter that is nuanced from our understanding of how it should function. This is one of the key advantages of Eureka, the ability to take general prompts and search for an aligned reward metric that works with human interpretability.

Finally, to confirm the improvements from the ground-truth reward function, we can investigate the graphs produced for Eureka's scalar value evaluations.

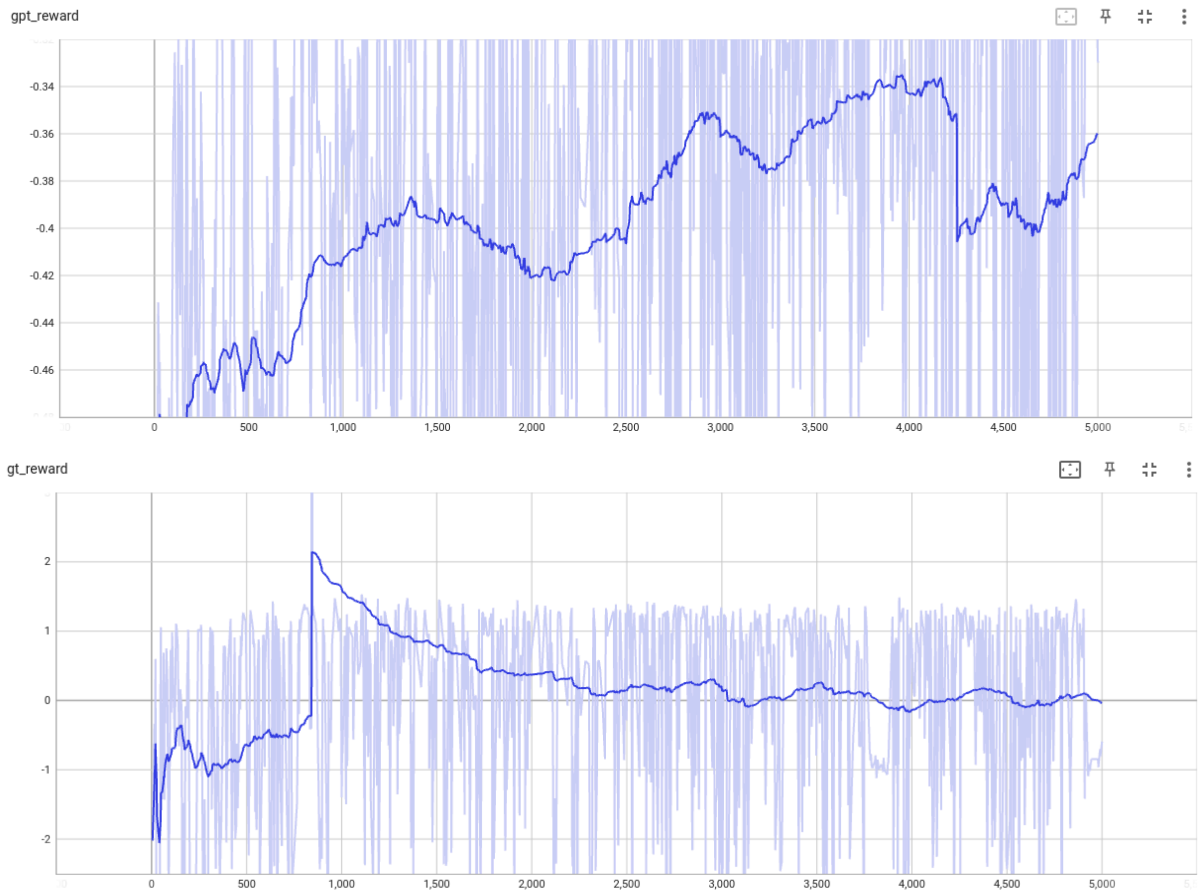

Figure 25: **Top;** Smoothed gpt_rewards which is the average reward per epoch, there is a general uptrend that would seem to continue past the specified 5000 epochs **Bottom;** Smoothed gt_reward is the ground truth rewards averaged from the original reward function every epoch.

Although direct comparison is nuanced due to variations in scaling and normalization, an analysis grounded in the operational mechanisms of both reward functions reveals that the gpt_reward demonstrates a greater efficacy in generating reward signals. Specifically, the original reward function was designed to progressively increase rewards as it neared the goal, offering a substantial bonus upon reaching the goal. Despite this, it fails to demonstrate a consistent improvement in average rewards, indicating a lack of significant learning progress. Conversely, the reward function generated by the language model consistently enhances its average rewards per epoch, suggesting ongoing learning and adaptation to feedback signals.

In Figure 26, it's noted that successes, defined as the negative mean distance to the goal, are consistently maintained at around -3.4. This metric shows an improvement with the increase in executed samples during the 2nd iteration, as depicted in Figure 27.

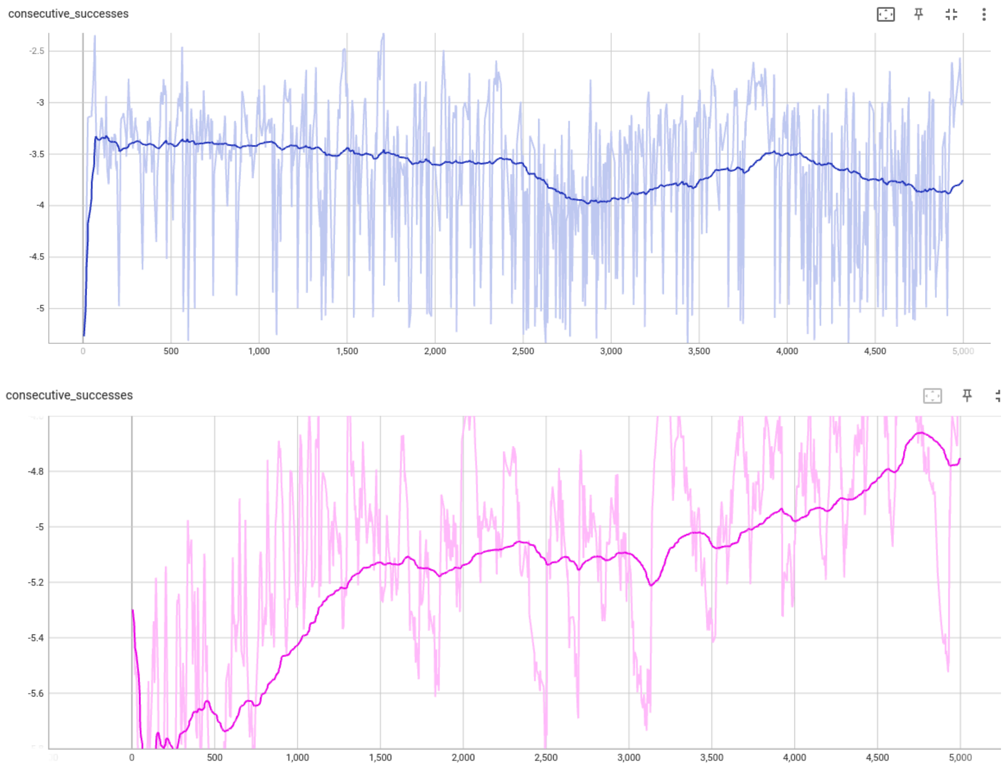

Figure 26: **Top;** Consecutive_successes defined as (mean -dist_to_target) in our task of waypoint navigation. The policy maintains a constant success distance that is closer than the **Bottom;** 1st iteration 2nd sample's consecutive success distance of -4.8.

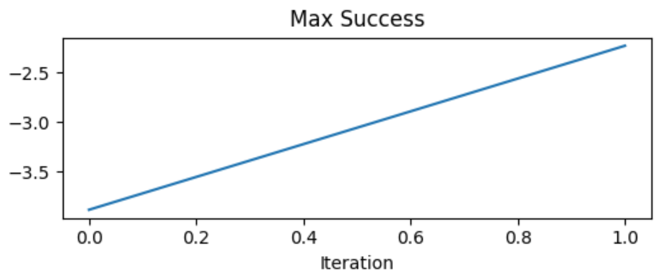

Figure 27: Max Success (mean -dist_to_target) over 2 samples in the 1st iteration and 4 samples in the 2nd iteration. Note that this is another sign of constant improvement over each mutation stage.

This observation validates our intuition regarding Eureka's capability to iteratively modify reward functions, leading to successive enhancements in the original reward function it commenced with. Remarkably, these improvements are observed after just two iterations, implying that Eureka has the potential to generate exponentially superior reward functions through additional iterations. Additionally, for a more tangible comparison, we can analyze the target distances achieved by comparing a reward function crafted by humans against one refined by Eureka, as illustrated in Figure 28.

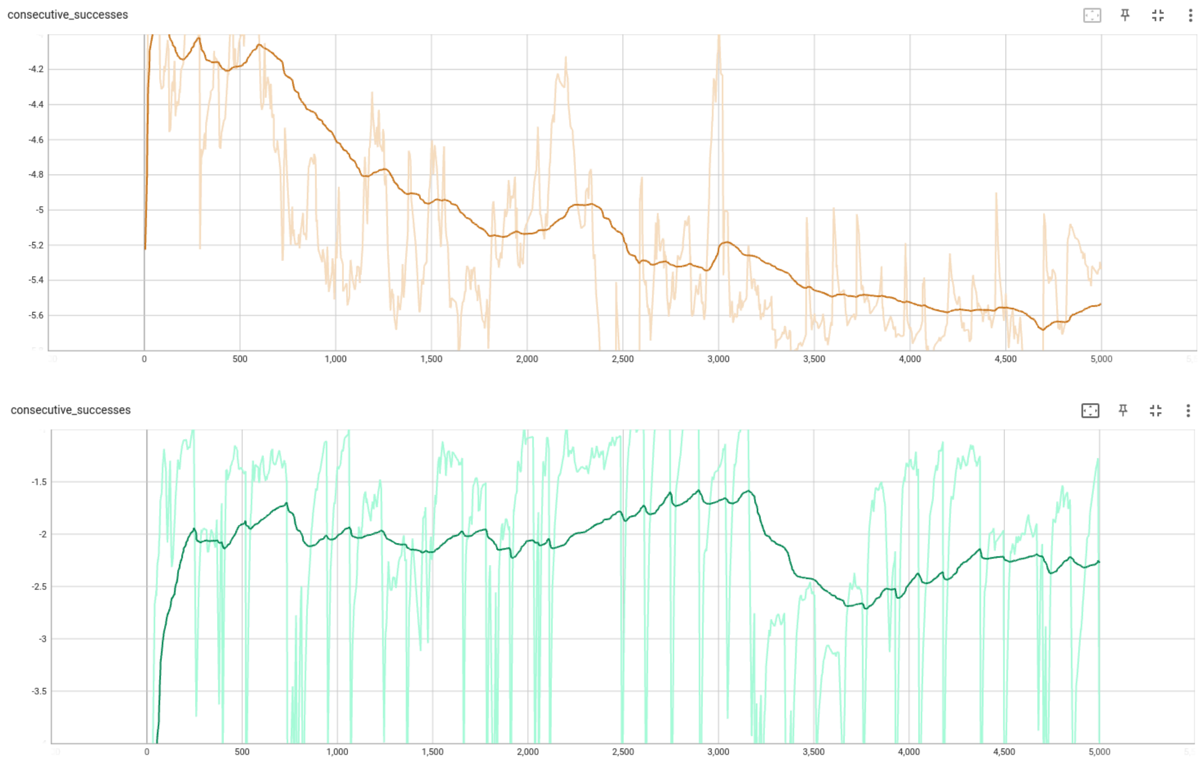

Figure 28: **Top;** Target distance achieved over 5000 epochs for human-crafted reward function, **Bottom;** Target distance achieved over 5000 epochs for Eureka-initialized and tuned reward function using 5 iterations with 3 samples.

We can see that Eureka's reward function performs significantly better than the Human-crafted reward function. Figure 29 shows a further quantified result relative to Figure 28 by defining an average success rate based on the achievement of a target distance closer than 2.0 meters which was chosen arbitrarily to be a sufficient success distance.

Figure 29: Average success rate achieved for both Eureka and human-crafted reward functions. Eureka (Blue) achieved a success rate of 56.5% and Human (Orange) achieved 5.80%.

Overall, there was a 50.7% increase in the success metric defined. Relatively, the simulation outcomes highlight Eureka's proficiency in both avoiding obstacles and successfully reaching the target, unlike the human-crafted approach that, despite being able to avoid obstacles, fails to effectively reduce the distance to the goal. This suggests the potential of using Eureka to enhance how reward functions are developed and iterated upon. Also, Refining the fitness function [7] a.k.a the consecutive success to include criteria like collision avoidance alongside target distance can significantly enhance the efficacy of Eureka for developing and iterating reward functions. This allows for a more targeted and effective training process by providing clear signals on both safety and efficiency in waypoint navigation tasks.

5 Conclusion and Future Works

In the development and evaluation of mobile robot social navigation through trajectory prediction and reinforcement learning (RL) enhanced by Large Language Model (LLM) tuning, we encountered several limitations that highlighted areas for future research and development. A notable challenge was the performance bottleneck introduced by heavy modifications to the Eureka framework, which surfaced by the constraints of Omniverse Isaac Gym. Specifically, the simulator's limitation to a single instance significantly slowed the iterative training process of the RL policies. This was compounded by an inefficiency in the execution recording mechanism, which, due to modifications, could not accurately track the execution time of each code iteration, affecting the overall speed and efficiency of the Eureka-based reward tuning process. Additionally, the Spatial Occupancy Grid Maps (SOGM) prediction network requires further training with enhanced hardware to improve its accuracy and reliability. The evaluation pipeline also requires refinement to fully leverage the predictive capabilities of the SOGM network. The integration of SOGM predictions with the robot's planning system presents another layer of complexity, requiring a robust framework to synchronize prediction and navigation seamlessly.

Addressing these challenges opens several future tasks aimed at elevating the efficacy of mobile robot navigation in social settings. Firstly, enhancing the generalizability of the SOGM network is important to ensure its applicability across diverse environments. This involves both hardware and software improvements alongside methodological refinements to adapt the network to varying contexts and dynamics. Also, Extending the prediction horizon beyond the current four seconds with more frequent updates could significantly improve navigation decisions, allowing for more proactive and less reactive maneuvers in dynamic environments. Integrating the trajectory prediction directly into the waypoint navigation DRL framework within Omniverse Isaac Gym represents a pivotal area of future work. By treating predicted trajectories as observable metrics or obstacles, the navigation policy can be enriched to account for future movements of dynamic entities, facilitating a deeper understanding of the environment and enhancing social compliance. This integration would enable the robot to anticipate and plan for human movements with greater accuracy, adhering to various social norms and expectations. Moreover, the incorporation of human feedback into the Eureka iteration process presents a promising direction for refining robot behaviors to be more human-centric. By systematically reviewing robot navigation behaviors and identifying areas for improvement, human evaluators can guide the evolution of reward functions to better capture the nuances of socially acceptable navigation. This feedback loop would ensure that the robot's behavior aligns more closely with human expectations, enhancing its utility and acceptance in shared environments.

In summary, the progression of mobile robot navigation in social contexts relies on addressing the technical limitations encountered in this research. By focusing on hardware and software enhancements, extending the prediction capabilities of the SOGM network, and tightly integrating predictive models with DRL-based navigation, future work can significantly advance the state-of-the-art in socially aware robot navigation. The iterative refinement of reward functions through human feedback and advanced LLM tuning further promises to align robot behaviors with complex human social norms, paving the way for robots that can navigate shared spaces with grace, efficiency, and social intelligence.

6 References

[1] K. D. Katyal, G. D. Hager, and C.-M. Huang, “Intent-Aware Pedestrian Prediction for Adaptive Crowd Navigation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, May 2020, pp. 3277–3283. doi: 10.1109/ICRA40945.2020.9197434.

[2] Y. Gao and C.-M. Huang, “Evaluation of Socially-Aware Robot Navigation,” *Front. Robot. AI*, vol. 8, 2022, Accessed: Nov. 12, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.721317

[3] H. Thomas, B. Agro, M. Gridseth, J. Zhang, and T. D. Barfoot, “Self-Supervised Learning of Lidar Segmentation for Autonomous Indoor Navigation.” arXiv, Dec. 10, 2020. Accessed: Oct. 16, 2023. [Online]. Available: http://arxiv.org/abs/2012.05897

[4] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “KPConv: Flexible and Deformable Convolution for Point Clouds.” arXiv, Aug. 19, 2019. Accessed: Oct. 16, 2023. [Online]. Available: http://arxiv.org/abs/1904.08889

[5] X. Jia *et al.*, “Think Twice before Driving: Towards Scalable Decoders for End-to-End Autonomous Driving.” arXiv, May 10, 2023. Accessed: Oct. 01, 2023. [Online]. Available: http://arxiv.org/abs/2305.06242

[6] V. Makoviychuk *et al.*, “Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.” arXiv, Aug. 25, 2021. Accessed: Mar. 14, 2024. [Online]. Available: http://arxiv.org/abs/2108.10470

[7] Y. J. Ma *et al.*, “Eureka: Human-Level Reward Design via Coding Large Language Models.” arXiv, Oct. 19, 2023. Accessed: Jan. 18, 2024. [Online]. Available: http://arxiv.org/abs/2310.12931

[8] C. Mavrogiannis *et al.*, “Core Challenges of Social Robot Navigation: A Survey.” arXiv, Mar. 16, 2021. Accessed: Mar. 17, 2024. [Online]. Available: http://arxiv.org/abs/2103.05668

[9] W. Burgard *et al.*, “The Interactive Museum Tour-Guide Robot,” presented at the Proceedings of AAAI-98, Jan. 1998, pp. 11–18.

[10] S. Thrun *et al.*, “MINERVA: a second-generation museum tour-guide robot,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Detroit, MI, USA: IEEE, 1999, pp. 1999–2005. doi: 10.1109/ROBOT.1999.770401.

[11] A. Clodic *et al.*, “Rackham: An Interactive Robot-Guide,” Oct. 2006, pp. 502–509. doi: 10.1109/ROMAN.2006.314378.

[12] K. Arras, N. Tomatis, and R. Siegwart, “Robox, a Remarkable Mobile Robot for the Real World,” Jan. 2002, pp. 178–187. doi: 10.1007/3-540-36268-1_15.

[13] T. D. Pete Bonasso, “Robots with AI: A Retrospective on the AAAI Robot Competitions and Exhibitions,” AAAI. Accessed: Mar. 21, 2024. [Online]. Available: https://aaai.org/papers/195-aaai96-195-robots-with-ai-a-retrospective-on-the-aaai-robot-competitions-and-exhibitions/

[14] A. Malinin and M. Gales, “Predictive Uncertainty Estimation via Prior Networks.” arXiv, Nov. 29, 2018. Accessed: Mar. 21, 2024. [Online]. Available: http://arxiv.org/abs/1802.10501

[15] S. Thompson, T. Horiuchi, and S. Kagami, “A probabilistic model of human motion and navigation intent for mobile robot path planning,” *2009 4th Int. Conf. Auton. Robots Agents*, pp. 663–668, Feb. 2009, doi: 10.1109/ICARA.2000.4803931.

[16] E. T. Hall, “A System for the Notation of Proxemic Behavior,” *Am. Anthropol.*, vol. 65, no. 5, pp. 1003–1026, 1963.

[17] M. Luber, L. Spinello, J. Silva, and K. Arras, “Socially-aware robot navigation: A learning approach,” presented at the Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2012, pp. 902–907. doi: 10.1109/IROS.2012.6385716.

[18] “Planning-based Prediction for Pedestrians.” Accessed: Mar. 22, 2024. [Online]. Available: https://www.cs.cmu.edu/~bziebart/publications/planning-based-prediction-pedestrians.html

[19] A. Francis *et al.*, “Principles and Guidelines for Evaluating Social Robot Navigation Algorithms.” arXiv, Sep. 19, 2023. Accessed: Nov. 08, 2023. [Online]. Available: http://arxiv.org/abs/2306.16740

[20] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang, “Interactive Trajectory Prediction of Surrounding Road Users for Autonomous Driving Using Structural-LSTM Network,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4615–4625, Nov. 2020, doi: 10.1109/TITS.2019.2942089.

[21] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data.” arXiv, Jan. 13, 2021. doi: 10.48550/arXiv.2001.03093.

[22] J. Li, H. Ma, Z. Zhang, J. Li, and M. Tomizuka, “Spatio-Temporal Graph Dual-Attention Network for Multi-Agent Prediction and Tracking,” *IEEE Trans. Intell. Transp. Syst.*, vol. PP, pp. 1–14, Jul. 2021, doi: 10.1109/TITS.2021.3094821.

[23] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks.” arXiv, Mar. 28, 2018. Accessed: Nov. 08, 2023. [Online]. Available: http://arxiv.org/abs/1803.10892

[24] Z. Huang, W. Schwarting, A. Pierson, H. Guo, M. Ang, and D. Rus, "Safe Path Planning with Multi-Model Risk Level Sets," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 6268–6275. doi: 10.1109/IROS45743.2020.9341084.

[25] H. Thomas, J. Zhang, and T. D. Barfoot, "The Foreseeable Future: Self-Supervised Learning to Predict Dynamic Scenes for Indoor Navigation," *IEEE Trans. Robot.*, pp. 1–19, 2023, doi: 10.1109/TRO.2023.3304239.

[26] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning." arXiv, Feb. 19, 2019. Accessed: Nov. 11, 2023. [Online]. Available: http://arxiv.org/abs/1809.08835

[27] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning." arXiv, May 04, 2018. Accessed: Nov. 11, 2023. [Online]. Available: http://arxiv.org/abs/1703.08862

[28] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "DenseCAvoid: Real-time Navigation in Dense Crowds using Anticipatory Behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, May 2020, pp. 11345–11352. doi: 10.1109/ICRA40945.2020.9197379.

[29] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning." arXiv, May 04, 2018. Accessed: Nov. 11, 2023. [Online]. Available: http://arxiv.org/abs/1805.01956

[30] R. Singh, J. Ren, and X. Lin, "A Review of Deep Reinforcement Learning Algorithms for Mobile Robot Path Planning," *Vehicles*, vol. 5, no. 4, Art. no. 4, Dec. 2023, doi: 10.3390/vehicles5040078.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms." arXiv, Aug. 28, 2017. Accessed: Mar. 14, 2024. [Online]. Available: http://arxiv.org/abs/1707.06347

[32] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. de S. L. Cuadros, and D. F. T. Gamarra, "Soft Actor-Critic for Navigation of Mobile Robots," *J. Intell. Robot. Syst.*, vol. 102, no. 2, p. 31, May 2021, doi: 10.1007/s10846-021-01367-5.

[33] U. Patel, N. Kumar, A. Sathyamoorthy, and D. Manocha, *Dynamically Feasible Deep Reinforcement Learning Policy for Robot Navigation in Dense Mobile Crowds*. 2020.

[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 961–971. doi: 10.1109/CVPR.2016.110.

[35] S. Poddar, C. Mavrogiannis, and S. S. Srinivasa, "From Crowd Motion Prediction to Robot Navigation in Crowds." arXiv, Mar. 02, 2023. Accessed: Oct. 05, 2023. [Online]. Available: http://arxiv.org/abs/2303.01424

[36] D. Shah, B. Osinski, B. Ichter, and S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action." arXiv, Jul. 26, 2022. doi: 10.48550/arXiv.2207.04429.

[37] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human Motion Trajectory Prediction: A Survey," *Int. J. Robot. Res.*, vol. 39, no. 8, pp. 895–935, Jul. 2020, doi: 10.1177/0278364920917446.

[38] A. Abuduweili, S. Li, and C. Liu, "Adaptable Human Intention and Trajectory Prediction for Human-Robot Collaboration." arXiv, Sep. 11, 2019. Accessed: Nov. 12, 2023. [Online]. Available: http://arxiv.org/abs/1909.05089

[39] G. Aydemir, A. K. Akan, and F. Güney, "ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation," presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 8295–8305. Accessed: Oct. 04, 2023. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Aydemir_ADAPT_Efficient_Multi-Agent_Trajectory_Prediction_with_Adaptation_ICCV_2023_paper.html

[40] Y. J. Ma *et al.*, "Eureka: Human-Level Reward Design via Coding Large Language Models." arXiv, Oct. 19, 2023. doi: 10.48550/arXiv.2310.12931.

[41] H. Thomas, M. G. de S. Aurin, J. Zhang, and T. D. Barfoot, "Learning Spatiotemporal Occupancy Grid Maps for Lifelong Navigation in Dynamic Scenes." arXiv, Sep. 16, 2021. Accessed: Oct. 15, 2023. [Online]. Available: http://arxiv.org/abs/2108.10585

[42] "GitHub - utiasASRL/Crystal_Ball_Nav: Like a Crystal Ball: Self-Supervised Learning to Predict the Future of Dynamic Scenes for Indoor Navigation." Accessed: Mar. 24, 2024. [Online]. Available: https://github.com/utiasASRL/Crystal_Ball_Nav

[43] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing." arXiv, Jan. 29, 2018. doi: 10.48550/arXiv.1801.09847.

[44] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds." arXiv, Aug. 19, 2019. Accessed: Oct. 15, 2023. [Online]. Available: http://arxiv.org/abs/1904.08889

[45] "9. Isaac Gym — Omniverse IsaacSim latest documentation." Accessed: Mar. 24, 2024. [Online]. Available: https://docs.omniverse.nvidia.com/isaacsim/latest/isaac_gym_tutorials/index.html

[46] "Jackal UGV - Small Weatherproof Robot - Clearpath." Accessed: Mar. 23, 2024. [Online]. Available: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

[47] "eureka-research/Eureka." eureka-research, Mar. 23, 2024. Accessed: Mar. 24, 2024. [Online]. Available: https://github.com/eureka-research/Eureka