Learning-Based Motion Planning for Dynamic Environments: From Foundational Algorithms to Emerging Paradigms

Zongyuan Shen[1], Shalabh Gupta[2], Shancheng Zhao[1], Dehua Zhou[1], Gao Wang[1], Rui Cheng[3], Yaming Ou[4], Zhongqiang Ren[5], Yikui Zhai[6], and C. L. Philip Chen[7], *Life Fellow, IEEE*

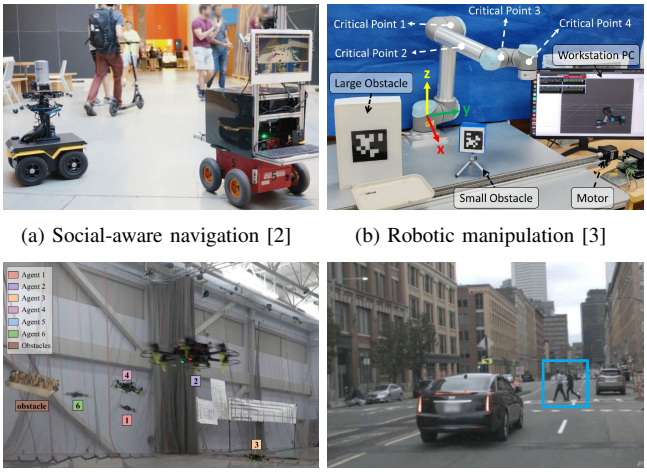
arXiv:2608.00625v1 [cs.RO] 1 Aug 2026

***Abstract*—Motion planning in dynamic environments is a fundamental problem in robotics, aiming to generate safe and efficient paths, trajectories, or control actions in the presence of moving obstacles, uncertain predictions, and multi-agent interactions. It has broad applications in autonomous driving, service robotics, warehouse logistics, human-robot collaboration, crowd navigation, and multi-robot systems. This survey reviews representative works published primarily between 2015 and 2025, with a particular focus on how recent learning-based advances extend, complement, or interact with classical planning foundations. We first revisit classical planning methods as algorithmic foundations and reference frameworks for learning-based extensions. We then propose a role-of-learning taxonomy that categorizes existing methods according to how learning participates in the planning pipeline, including direct policy learning, learning-augmented classical planning, hybrid planning, and training enhancement methods. For each category, we summarize the main problem settings, representative algorithms, key ideas, integration mechanisms, strengths, and limitations. We further analyze how observation representations, prediction uncertainty, interaction modeling, planner integration, safety constraints, and training strategies shape learning-based motion planning in dynamic environments. Finally, we discuss open challenges and future directions, including sim-to-real gap, safe and certifiable planning, dense crowd navigation, perception-planning coupling, and embodied AI.**

***Index Terms*—Motion planning, reinforcement learning, dynamic environments, collision avoidance, autonomous robots.**

I. Introduction

Autonomous robots are increasingly deployed in dynamic environments to perform different activities, such as public service, warehouse operations, and human-robot collaborative tasking. In these scenarios, robots must navigate among pedestrians, vehicles, other robots, and moving objects whose future trajectories may be uncertain or difficult to predict [1]. Unlike motion planning in static environments, dynamic environments require continuous plan adaptation rather than one-time path generation. The robot must update its trajectory or control actions in real time to avoid collisions, maintain steady progress, and ensure safe and uninterrupted navigation. Fig. 1 shows application examples of motion planning in dynamic environments using different robotic platforms, such as unmanned ground vehicles, industrial manipulators, unmanned aerial vehicles, and driverless cars.



(a) Social-aware navigation [2] (b) Robotic manipulation [3]

(c) Multi-agent navigation [4] (d) Autonomous driving [5]

Fig. 1. Application examples of motion planning in dynamic environments: (a) mobile robot navigation in public spaces [2], (b) robotic manipulation in dynamic workspaces [3], (c) aerial navigation among agents [4], and (d) autonomous driving with moving humans [5].

A. Motivation

The above requirements make motion planning in dynamic environments challenging due to a constant change in the feasible navigation space of the robot as obstacles move, appear, or disappear. Furthermore, the motion of pedestrians, vehicles, and other robots is often uncertain, making it difficult to predict future collision risks [6]. In particular, in human-populated or multi-agent scenarios, the robot must also account for robot-agent and agent-agent interactions, since the motion of each agent may influence the decisions of others [7]–[9]. Moreover, the planning decisions must be updated in real time while satisfying collision-avoidance, kinodynamic constraints, and task-progress requirements. Failure to balance these requirements may lead to unsafe motions, oscillatory behaviors, excessive conservatism, or the freezing robot problem.

Classical planning methods [10]–[13] have long provided the foundation for motion planning in dynamic environments.

[1]College of Information Science and Technology, Jinan University, Guangzhou 510632, China.
[2]Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA.
[3]Guangzhou Maritime University, Guangzhou 510725, China.
[4]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China.
[5]Global College, Shanghai Jiao Tong University, Shanghai 200240, China.
[6]School of Electronics and Information Engineering, Wuyi University, Jiangmen 529000, China.
[7]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China.

Fig. 2. Taxonomy of motion planning in dynamic environments.

These methods can be grouped into global and local replanning methods. Global replanning methods target optimality and provide long-range path guidance to produce feasible paths when environmental changes invalidate the current path, whereas local replanning methods repair robot trajectories over a short-horizon or generate control commands for real-time collision avoidance. The classical methods are widely used because they provide interpretable decision rules and explicit mechanisms for safety or feasibility with reasoning. However, their performance often depends on hand-crafted costs, manually tuned parameters, simplified interaction models, reliable obstacle prediction, and sufficient online computational resources.

To overcome these limitations, learning-based methods have been increasingly applied to motion planning in dynamic environments. By learning from demonstrations, interactions, or simulated experiments, these methods can build navigation policies, interaction patterns, adaptive costs, planning guidance, and dynamic-obstacle predictions from data. Importantly, learning is not limited to replacing planners with end-to-end policies; it can also enhance classical planners or be integrated with them in hybrid planners. In direct policy learning [14]–[16], the learned models serve as the primary decision-making module, whereas in learning-enhanced [17]–[19] and hybrid planning [20]–[22], the learned modules provide auxiliary guidance, intermediate decisions, or online adaptation while classical planners retain the primary structural planning roles. This diversity makes it necessary to review learning-based motion planning methods according to how the learning approaches contribute to the planning process.

B. Existing Surveys

Although motion planning in static environments has been extensively reviewed [23]–[26], surveys specifically dedicated to motion planning in dynamic environments remain relatively limited. The survey by Kamil et al. [27] focused mainly on local replanning methods, such as potential field-based, control-based, velocity obstacle-based methods. Mohanan et al. [28] provided an in-depth review of classical planning methods for dynamic environments. These surveys summarize developments up to 2015 but do not reflect the significant progress made in learning-based motion planning over the past decade. More recent surveys have focused on specific problem settings or application domains. Hewawasam et al. [29] briefly reviewed path planning for mobile robot navigation in dynamic environments. Liu et al. [30] provided a focused review on motion planning for industrial manipulators operating in dynamic environments, with particular attention to real-time planning performance in high-dimensional configuration spaces.

C. Contributions

Although existing surveys provide valuable insights, a unified and up-to-date review of learning-based motion planning in dynamic environments is still lacking. This survey addresses this gap by presenting a comprehensive and structured review of representative works primarily published between 2015 and 2025, while linking them to classical planning foundations. Specifically, before reviewing learning-based motion planning methods, we first revisit classical planning methods as algorithmic foundations and reference frameworks for later learning-based extensions. Rather than organizing the literature only by learning paradigms or classical algorithmic families, we adopt a role-of-learning taxonomy that characterizes how learning participates in the planning pipeline.

Accordingly, existing learning-based motion planning methods are organized into four main categories: direct policy learning, learning-augmented classical planning, hybrid planning, and training enhancement. The roles and key distinctions of these categories are summarized in Table I. For each category, we review the main problem settings, representative algorithms, key ideas, learning paradigms, integration mechanisms, and strengths and limitations. Fig. 2 shows overall organization of these categories and the methodologies therein.

D. Organization

The remainder of this paper is organized as follows. Section II reviews classical planning foundations for dynamic environments, including global replanning and local replanning methods. Section III discusses direct-policy learning methods, where learned models serve as the primary navigation decision maker. Section IV reviews learning-enhanced classical planning methods, in which learning is used to improve classical planners through learned parameter, search guidance, or obstacle prediction. Section V discusses hybrid planning methods that couple learned modules with classical planners in the online planning pipeline. Section VI reviews training enhancement methods that improve policy learning through expert guidance or diverse scenarios. Finally, Section VII concludes the survey and discusses future research directions.

II. Classical Planning Foundations

Classical planning methods provide the algorithmic foundations for navigation in dynamic environments. They define

Table I: Taxonomy and comparison of learning-based motion planning methods in dynamic environments.

Category	Main Idea	Role of Learning	Generated Output	Strengths	Limitations
Direct Policy Learning	Learn navigation policies directly from observations	Primary navigation decision-making module	Navigation actions, local trajectories, and subgoals	Fast and compact end-to-end navigation decision making	Limited generalization, interpretability, and safety guarantees
Learning-Augmented Classical Planning	Use learned module to enhance, rather than replace, classical planners	Auxiliary module	Planner parameters, search guidance, and obstacle predictions	Improve adaptability in complex scenarios, maintain interpretability of classical planners	Sensitive to the quality and generalization of learned auxiliary information
Hybrid Planning	Couple learned decision module with classical planners in the online planning pipeline	Navigation decision making module coupled with classical planner	Navigation actions, policy switching decisions, and subgoals	Balances learning adaptability with safety and interpretability of classical planners	Sensitive to module coupling, learned-output quality, and policy generalization
Training Enhancement	Improves policy learning through expert guidance or diversified scenarios	Policy improvement at training stage	Expert-guided initial policies, and diverse training scenarios	Improves training efficiency, robustness, and generalization	Performance depends on expert quality, training diversity, and scenario coverage

core mechanisms that are later enhanced or integrated with learning-based modules, including sampling, graph search, velocity selection, potential-field design, and control command generation. This section groups the representative classical methods into two types: global and local replanning methods.

A. Global Replanning Methods

Global replanning methods provide long-range planning by updating and repairing the robot trajectory when dynamic obstacles or environmental changes invalidate it or pose a risk. In this category, the existing methods can be classified into sampling-based and search-based methods.

1) Sampling-Based Methods: Sampling-based methods typically extend Probabilistic Roadmap (PRM) [31], Rapidly-exploring Random Tree (RRT) [32], and RRT* [33] to dynamic environments. They can be broadly classified as reactive or active. Reactive methods rely on current observations to update paths, enabling fast responses but lacking foresight in complex environments. Active methods incorporate predicted future states of dynamic obstacles for risk-aware replanning. However, their performance depends on prediction accuracy and may degrade under limited sensing or dense crowds.

a) *Reactive Methods:* Early reactive methods reused previous search efforts to some extent to accelerate replanning. Bruce and Veloso [34] rebuilt the tree from scratch while biasing new samples toward the previous path. Ferguson et al. [35] improved tree reuse by pruning invalidated nodes and their descendants to retain a single-tree structure. Zucker et al. [36] further preserved valid tree components by pruning only colliding nodes and reconnecting the resulting disjoint trees through biased sampling. To fully reuse previous search efforts, Otte and Frazzoli [10] maintained the same graph without pruning nodes and repaired the goal-rooted subtree through rewiring cascades after environmental changes. Later variants improve convergence, reduce computation, or extend the idea to kinodynamic replanning [37]–[39].

More recent methods develop advanced heuristics to guide tree repair. Chen et al. [40] biased new samples toward low-cost regions captured by a Gaussian mixture model trained online. Other methods use dual-tree, reverse-tree, potential-field, or multi-objective guidance to accelerate reconnection and improve path quality [41]–[44]. Shen et al. [6] performed fast tree-repair at hot spots that lie at the intersection of different disjoint trees, reducing unnecessary random exploration and improving computational efficiency.

b) *Active Methods:* Early methods mainly used prediction-based risk to regulate node insertion and pruning. Chiang et al. [45] gradually increased risk tolerance over time to account for growing prediction uncertainty. Chi et al. [46] expanded low-risk nodes in a time-embedded tree within a predictive horizon and then improved path quality through rewiring.

Later methods improved active replanning through enhanced risk modeling and search structures. Cai et al. [47] expanded the tree by adding low-risk nodes, where the risk is evaluated based on collision probability and the likelihood of entering dense human crowds. Hakobyan and Yang [48] quantified node risk with a safety loss based on conditional value-at-risk, which assigns higher risk to nodes that may lead to severe safety violations. This allows the planner to distinguish nodes with similar collision probabilities but different severities of potential safety violations. Sun et al. [49] expanded multiple trees to explore the environment in parallel and used them to guide the main tree toward safer regions, thereby improving search efficiency and robustness in cluttered environments.

2) Search-Based Methods: Search-based methods provide another classical foundation for global replanning. They represent the configuration space as a graph, grid, or lattice, and utilize fundamental graph-search algorithms, such as Dijkstra's algorithm [50] and A* [51]. Subsequently, these methods employ techniques such as incremental graph repair, spatio-temporal safety reasoning, and kinodynamic primitive search.

a) *Incremental Graph Repair:* These methods improve replanning efficiency by updating only the affected parts of the graph when environmental changes modify edge costs. Early methods, such as D* [52], D* Lite [53], and Lifelong Planning A* (LPA*) [11], repair A*-like solutions according to newly acquired environmental information. Ren et al. [54] extended this line of work to multi-objective replanning by maintaining a Pareto-optimal set of paths under vector-valued edge costs. Li et al. [55] combined forward and backward searches to improve the reuse of previous search results during replanning.

b) *Spatio-Temporal Safety Reasoning:* These methods

search for collision-free paths in the spatio-temporal domain. Safe Interval Path Planning [56] associates each configuration with collision-free time intervals and searches over these intervals rather than every discrete time step, thereby reducing the search space while preserving completeness and optimality under its assumptions. Several variants improve its efficiency and applicability [57]–[60]. Another line of work builds time-varying graphs. Cao et al. [61] constructed a graph from the Delaunay triangulation of pedestrian positions, where edges define time-varying gates between neighboring pedestrians. An A*-based search is then used to find a feasible channel through these gates. Huang et al. [62] constructed a dynamic connected visibility graph with safe intervals on graph edges, from which spatially and temporally distinct initial paths and safe corridors are generated for trajectory optimization.

c) *Kinodynamic Primitive Search:* These methods extend graph search from geometric path planning to kinodynamic trajectory generation by expanding short-horizon motion primitives that satisfy robot motion constraints while avoiding dynamic obstacles. Lin et al. [63] proposed a search-based online partial motion planner for car-like robots, which generates motion primitives by discretizing time and control inputs and searches a state-time graph for feasible motion.

Later methods incorporate prediction, risk modeling, and hierarchical design. Chen et al. [64] used a particle-based dynamic map to predict future occupancy and an A*-based search to generate a reference path, which is converted into spatio-temporal safety corridors for trajectory optimization. Chen et al. [65] combined local risk-aware primitive search with global reference guidance, enabling the robot to follow the global path when safe, deviate locally when blocked, and reconnect once a safe path becomes available. Qi et al. [66] generated a coarse trajectory through spatio-temporal motion-primitive search and refined the path and speed profile via multilayer optimization. Wiman and Tiger [67] used a high-resolution lattice for short-term collision-free planning and a low-resolution lattice for long-horizon guidance.

Motivation for Learning-Based Planning Methods: Classical global replanning methods support long-range path generation and online path repair, but their performance is often constrained by sampling efficiency, graph resolution, heuristic design, and prediction accuracy. These limitations motivate learning-based planning methods that improve global replanning by learning sampling biases, search costs, heuristic functions, risk maps, or obstacle predictions. In hybrid frameworks, classical global replanners provide long-range guidance, while learned local modules handle collision avoidance, kinodynamic constraints, or interaction-aware decision making.

B. Local Replanning Methods

Local replanning methods provide short-range planning by updating and repairing the robot trajectory when dynamic obstacles or environmental changes invalidate it or pose a risk. These methods generate control commands for a short time-horizon based on the local information. Existing methods are grouped into model predictive control (MPC), velocity obstacle, potential field, and dynamic window methods.

1) Model Predictive Control Methods: MPC methods formulate local replanning as a finite-horizon optimal control problem. At each planning step, MPC predicts the robot's future states over a finite horizon and incorporates predicted obstacle motion into safety constraints. It then optimizes control inputs under robot dynamics and safety constraints.

A central concern is how to formulate safety constraints for collision avoidance using predicted obstacle trajectories and uncertainties. Existing methods commonly represent obstacles as convex polyhedra (e.g., cuboids) or smooth differentiable surfaces (e.g., ellipsoids) [68]. Polyhedral models can encode collision avoidance via linear inequality constraints, but can be computationally expensive in crowded scenarios. Smooth surface representations reduce the number of constraints and are better suited to nonlinear optimization, but may yield conservative solutions. Based on these representations, collision avoidance can be formulated as deterministic constraints [69], chance constraints [68], [70], [71], scenario constraints [72], or control barrier function-based constraints [73]–[75].

Another concern is deadlock caused by local minima in the optimization landscape. Existing methods mainly address this through high-level guidance or proactive cost design. Groot et al. [76] generated homotopically distinct global paths using a visibility-probabilistic roadmap [77], tracked them with parallel MPC planners, and selects the MPC trajectory with the lowest cost. Arul et al. [78] introduced a terminal state cost based on expected time-to-goal and time-to-collision, which favors the states with better goal-reaching potential and more free space ahead. This encourages the robot to detour rather than freeze in locally safe but deadlocked configurations.

2) Velocity Obstacle Methods: Velocity obstacle methods perform local collision avoidance in velocity space. The concept of Velocity Obstacles (VO), introduced by Fiorini and Shiller [12], characterizes the set of relative velocities that would lead to future collisions between the robot and nearby agents. By selecting a velocity outside the VO set at each planning step, the robot can maintain collision-free motion.

Oscillations may occur when agents simultaneously react to each other's previous velocity choices. To address this issue, Van den Berg et al. [79] proposed Reciprocal Velocity Obstacles (RVO), which assumes that both agents share collision-avoidance effort when constructing the VO set, thereby reducing overreactions. Snape et al. [80] introduced Hybrid RVO (HRVO) to further reduce oscillatory behaviors.

More recent methods relax the ideal cooperation assumptions. Liu et al. [81] used reinforcement learning (RL) to infer collision-avoidance responsibility, which is the relative share of the collision avoidance effort that the robot should take when interacting with nearby agents. Martinez et al. [82] used an adaptive control law to estimate the cooperation level, which describes how much those agents are expected to participate in the reciprocal collision avoidance of nearby agents, and accordingly adjust the robot's responsibility.

Other works extend VO method from holonomic models to more realistic motion models by incorporating smoothness requirements [83], acceleration limits [84], linear dynamics constraints [85], and nonholonomic motion constraints [86].

3) Potential Field Methods: Potential field methods define attractive potentials toward the goal and repulsive potentials around obstacles, guiding the robot by the negative gradient of the combined field. For dynamic environments, Ge and Cui [13] constructed potentials from relative position and velocity between the robot, goal, and obstacles. Later methods improved dynamic avoidance using global sampling guidance [87], stochastic reachable sets for uncertain obstacles [88], oval or bounded vector fields for smoother bypassing and goal convergence [89], [90], and vortex-based or emergency deflection mechanisms to escape local minima [91].

4) Dynamic Window Methods: Dynamic Window Approach (DWA) [92] performs local replanning by evaluating the candidate control commands under the robot's dynamic constraints. Given the velocity and acceleration limits of the robot, DWA constructs a dynamic window of admissible commands, scores each candidate according to certain criteria (e.g., goal progress and obstacle clearance), and executes the best command. Extensions for dynamic environments include dynamic-polygon obstacle representations [93], long-term DWA with Elastic Band refinement [94], [95], uncertainty-aware safe DWA [96], and gradient-aware collision costs based on obstacle-distance fields [97].

Motivation for Learning-Based Planning Methods: Classical local replanning methods provide efficient mechanisms for real-time collision avoidance, but their performance often depends on hand-crafted cost functions, manually tuned parameters, interaction assumptions, and reliable short-term perception or prediction. These limitations motivate the learning-based methods that improve local replanning by learning adaptive costs, planner parameters, obstacle predictions, or interaction-aware behaviors. Learning can either enhance classical planners with auxiliary guidance or be integrated with them in hybrid frameworks to combine data-driven decision making with model-based safety and feasibility.

III. Direct Policy Learning Methods

In learning-based motion planning, learning generally refers to using data or interaction experience to train parameterized models that support navigation decision making. In robotic applications, these models are commonly implemented as neural networks, such as Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Graph Neural Network (GNN). Depending on the training formulation, they can be optimized by supervised learning, imitation learning, RL, or hybrid training schemes. The learned models may serve different roles, such as extracting observation features, encoding agent interactions, predicting future states, or generating navigation actions.

Direct policy learning methods use learned models in the primary decision-making module for navigation in dynamic environments. Given sensor observations, robot states, agent states, or interaction models, the learned policies directly generate control commands, local trajectories, subgoals, or navigation behaviors. According to the policy formulation, existing methods can be grouped into three classes: observation-to-action, interaction-aware, and hierarchical learning methods.

A. Observation-to-Action Methods

Observation-to-action methods directly map robot observations to navigation actions. By reducing the reliance on explicit environmental modeling or agent-level reasoning, they provide a compact decision-making pipeline for navigation in dynamic environments. Existing methods can be classified into single-modal and multi-modal methods, depending on whether the policy relies on one sensing modality or combines complementary sensory information. Table II summarizes the key features of representative observation-to-action methods.

1) Single-Modal Methods: Single-modal methods learn navigation policies based on one primary sensing modality.

a) *Supervised Learning-Based Methods:* These methods learn navigation policies from labeled data. Long et al. [14] generated training data using Optimal Reciprocal Collision Avoidance (ORCA) [102] under different parameter settings and sensing-noise levels. The continuous velocity space is discretized, and a network is trained as a multi-class classifier to infer collision-avoidance velocities from LiDAR measurements and a goal-directed preferred velocity.

b) *Reinforcement Learning-Based Methods:* These methods learn navigation policies through reinforcement learning (RL) by improving the policy through interactions with the environment. Long et al. [98] proposed a LiDAR-based method that maps raw LiDAR measurements to control commands for decentralized multi-robot collision avoidance. The method formulates the task as a Partially Observable Markov Decision Process (POMDP) [103] and trains a shared policy using Proximal Policy Optimization (PPO) [104]. Fan et al. [105] extended this method to dense crowds by introducing feature-rich recovery points, which mitigate localization failures caused by crowd occlusions and degraded LiDAR-based SLAM. Heuvel et al. [99] further exploited temporal structure in LiDAR observations. The method constructs Temporal Accumulation Group Descriptors (TAGDs) from consecutive LiDAR measurements to capture obstacle motion trends. A Deep Deterministic Policy Gradient (DDPG)-based policy [106] then takes LiDAR measurements, TAGDs, upcoming waypoints of robot as inputs and processes them via separate spatial and temporal attention streams, producing feature embeddings, which are concatenated and processed to generate control commands. Fan et al. [107] compressed historical LiDAR scans into a 2D obstacle map that captures obstacle contours and motion cues. A PPO-trained network combines this temporal representation with the quadrotor state and target command to output horizontal acceleration commands. A dynamic obstacle-aware reward enlarges the risk region along obstacle motion directions, improving reactions to high-speed obstacles.

Strengths and Limitations: Single-modal methods are computationally efficient and easy to implement; however, relying on a single sensing modality limits the utilization of rich environmental information. Especially in complex scenarios, a single sensing modality may not fully capture the semantic cues, spatial structure, and dynamic obstacle motion. These limitations motivate multi-modal methods that incorporate complementary sensory information.

2) Multi-Modal Methods: Multi-modal methods learn policies from complementary sensing modalities (e.g., camera and

Table II: Comparison of representative observation-to-action methods.

Category	Reference	Year	Main Idea	Sensing Modality	Learning Paradigm	Policy Output
Single-Modal	[14]	2017	Discretizes the velocity space and learns a multi-class classifier from ORCA-generated data for collision-free velocity selection	LiDAR	Supervised learning	Control command
	[98]	2018	Trains a shared policy using PPO for decentralized multi-robot collision avoidance	LiDAR	Reinforcement learning	Control command
	[99]	2024	Uses a DDPG-based policy with spatio-temporal attention to capture obstacle motion trends from LiDAR observations	LiDAR	Reinforcement learning	Control command
Multi-Modal	[15]	2021	Trains a CNN-based policy as a continuous velocity regressor from DWA-generated data for control command generation	LiDAR + Camera	Supervised learning	Control command
	[100]	2021	Fuses semantic segmentation maps and LiDAR data into a PPO-based policy to improve navigation robustness	LiDAR + Camera	Reinforcement learning	Control command
	[101]	2022	Weights depth features using self-state attention and feeds them into a PPO-based policy	LiDAR + Camera	Reinforcement learning	Control command
	[16]	2023	Uses a VO-based reward to train a PPO-based policy for proactive collision avoidance	LiDAR + Camera	Reinforcement learning	Control command

LiDAR) to enrich the representation of dynamic environments.

a) *Supervised Learning-Based Methods:* These methods learn navigation policies from labeled data. Pokle et al. [108] trained a local planner via behavioral cloning from simulated human teleoperation and the ROS Navigation Stack with social costs [109]. Given a global path, LiDAR measurements, robot odometry, and nearby human trajectories, the method generates a local trajectory. Xie et al. [15] used DWA [92] as the expert to generate training data. The method adopts a CNN [110] that takes a short history of LiDAR measurements, human states, and a waypoint as inputs, and learns continuous velocity commands by minimizing the mean-square-error loss between expert and predicted velocities.

b) *Reinforcement Learning-Based Methods:* These methods learn navigation policies through RL. Huang et al. [100] incorporated visual semantics and LiDAR measurements into an RL-based policy. The method first converts RGB images into binary traversability maps, which are then fused with LiDAR measurements and processed by a PPO-trained policy. Han et al. [101] focused on enhancing depth perception from RGB images and LiDAR measurements. The method uses sparse-to-dense depth completion [111] to estimate a depth map, which is fused with LiDAR data into a 2D obstacle-distance representation. A self-state-attention module assigns adaptive weights to depth features, and the resulting representation is used for policy learning. Xie and Dames [16] combined preprocessed sensor observations with VO-based reward shaping in RL. The method converts a short history of LiDAR scans into a grid map to encode geometric structure. It detects and tracks pedestrians from RGB-D data, and encodes their relative states into pedestrian kinematic maps. These representations, together with a subgoal, are used as policy inputs. It further uses VO [12] to formulate an active heading reward, which encourages the robot to steer toward a collision-free direction while maintaining progress toward the goal.

Strengths and Limitations: Multi-modal methods provide both semantic information and geometric structure by combining visual inputs and LiDAR measurements. These methods help the policy perceive the environment more comprehensively and improve navigation robustness. However, they increase system complexity due to additional perception modules, sensor calibration, modality synchronization, and fusion mechanisms, thus leading to higher computational costs. Furthermore, errors in segmentation, depth estimation, or sensor fusion may also propagate to the policy, and the enlarged observation space can make training more difficult.

B. Interaction-Aware Methods

Interaction-aware methods are useful for navigation in crowded environments, where the robot's decision is influenced not only by individual agent states but also by mutual interactions among agents. As such, these methods explicitly model interactions among agents within RL frameworks and incorporate such relational information into policy learning. Existing methods encode interactions in different forms and can be classified into joint state-based, attention-based, and graph-based methods. The key differences among these interaction modeling strategies are summarized in Table III, while Table V compares representative interaction-aware methods.

1) Joint State-Based Methods: Joint state-based methods model robot-agent interactions through structured joint representations. By encoding information such as relative positions, velocities, distances, headings, and agent sizes, these methods allow the learned policy to reason about potential conflicts.

Chen et al. [2] learned a value network that estimates time-to-goal values from joint states. During execution, each agent

Table III: Comparison of interaction modeling strategies.

Modeling Strategy	Main Idea	Representation	Modeled Relation	Strengths	Limitations
Joint State-Based	Encodes interactions as joint states and infers conflicts from relative kinematic information	Structured joint state with robot and nearby-agent states	Robot-agent	Simple, compact, and computationally efficient	Limited ability to model complex interactions
Attention-Based	Learns weights over agents or interaction features to focus on decision-relevant information	Weighted agent or interaction features	Robot-agent, human-human	Focuses on influential agents and reduces irrelevant crowd information	Depends on reliable detection and may not explicitly encode structured relations
Graph-Based	Models the navigation scene as a graph and performs relational reasoning via message passing among entities	Graph with entities as nodes and interactions as edges	Relations among robots, humans, and static obstacle	Strong relational modeling ability and useful for dense or heterogeneous scenes	Requires accurate graph construction and usually has higher training and computational complexity

performs a one-step lookahead over candidate velocity actions and selects the action that maximizes the estimated value, which combines goal-reaching rewards and collision penalties. In this process, nearby agents are assumed to maintain their current velocities over a short time horizon. This method was later extended in [112] by augmenting the reward function with social-norm penalties, encouraging socially aware behaviors such as passing on the right and overtaking on the left. To reduce the reliance on online lookahead under simplified motion assumptions, Everett et al. [113] replaced value-based action evaluation with actor-critic policy learning. The method learns a policy that maps the robot state and nearby-agent states to a probability distribution over discrete actions. To handle a variable number of nearby agents, it uses a Long Short-Term Memory (LSTM) [114] network to encode nearby-agent states into a fixed-length representation.

Zhu et al. [115] instead focused on improving interaction-aware learning by modeling agent heterogeneity in terms of shape and speed. The method uses Orientated Bounding Capsules (OBCs) rather than homogeneous circles to represent robots and nearby agents, and incorporates capsule-based distances and relative orientations into the joint-state representation. It further introduces a velocity-related collision-risk metric, where faster agents induce larger forward risk regions.

Strengths and Limitations: Joint-state-based methods encode the robot and agents in a shared state space, allowing the policy to use their relative positions and velocities for collision avoidance and social navigation. However, they mainly model robot-agent interactions, without identifying influential agents or capturing how agent-agent interactions affect the robot in dense crowds. These limitations motivate later methods that capture relational dependencies among agents.

2) Attention-Based Methods: Attention-based methods construct state representations in which nearby agents and interaction features are weighted according to their relevance to robot's navigation decision. In this way, the model can focus on agents that are closer to the robot, more likely to collide, or likely to affect path selection, while also emphasizing key interaction features, such as relative distance, relative velocity, heading direction, time-to-collision, and local crowd density.

Chen et al. [116] constructed latent pairwise interaction features for robot-human pairs using an MLP, with inputs including the robot's goal distance and velocity, and the human's relative position, velocity, radius, and distance to the robot. To capture human-human interactions, the method constructs a local map for each nearby human, where grid cells encode the occupancy and velocities of other humans around that person. This local map provides a coarse spatial representation of how surrounding humans may influence the motion of the current human. The resulting interaction features are fed into a self-attention-based pooling module, which assigns different importance weights to nearby humans and aggregates the weighted features into a compact crowd representation for action selection. This method was later extended in [117] by processing humans and static obstacles through separate channels, enabling more appropriate navigation decisions.

While these methods improve spatial interaction modeling, they mainly reason about interactions at the current time step. Yang et al. [118] extended attention-based interaction modeling from single-frame spatial attention to spatio-temporal self-attention. The method uses the Transformer architecture [119] to encode multi-frame robot-human joint states. A spatial Transformer encoder captures robot-human interactions, while a temporal Transformer encoder captures human motion trends and their changing relations with the robot over consecutive frames. The resulting spatio-temporal representation is used in a value-based RL framework for action selection.

Strengths and Limitations: Attention-based methods improve interaction modeling by emphasizing decision-relevant agents, obstacles, or interaction features. However, they rely on accurate human detection and tracking, may miss structured relational dependencies among agents, and incur higher training and inference costs as the number of nearby agents or interaction features increases. Although they usually do not require manually labeled navigation labels, RL-based attention policies still require a large number of interaction samples, which are commonly collected from simulators or generated using demonstration policies for initialization.

3) Graph-Based Methods: Graph-based methods model crowd navigation as relational reasoning over agents, obstacles, and their interactions. Nodes represent entities such as the robots, humans, or static obstacles, while edges describe their spatial relations, interaction strengths, or temporal connection. Compared with joint state-based or attention-based methods,

Table IV: Comparison of graph-based interaction modeling strategies.

Graph Type	Main Idea	Node	Edge	Strengths	Limitations
Spatial Graph	Models interactions among entities at current time step	Robots and humans	Encodes inter-agent spatial interactions	Simple and effective for current-frame relational reasoning	Limited ability to capture temporal evolution and entity heterogeneity
Spatio-Temporal Graph	Extends graph reasoning from spatial relations to temporal evolution	Robots and humans	Spatial edges encode inter-agent relations; temporal edges encode motion continuity	Captures changing interactions and supports more anticipatory decisions	Requires temporal history and increases model complexity
Heterogeneous Graph	Models different entity and relation types in one graph	Robots, humans, and static obstacles	Encodes different types of spatial interactions	Suitable for cluttered and multi-entity scenes	Requires accurate entity classification and more complex graph construction

graph structures provide a more explicit way to model robot-human, human-human, robot-robot, robot-obstacle, and group-level relations. According to the graph structure and relation modeling mechanism, existing methods can be grouped into spatial graph, spatio-temporal graph, and heterogeneous graph methods. The key differences among these graph modeling strategies are summarized in Table IV.

a) *Spatial Graph Methods:* These methods formulate the robot and humans as a spatial graph to model interactions.

Chen et al. [120] introduced relational graph learning for crowd navigation by constructing a directed graph, where each node represents an agent and each edge indicates an inferred relation between two agents. This relation can be interpreted as an influence weight, such as how strongly a nearby human affects the robot's decision or how one human influences another human's motion. Robot and human states are embedded into latent agent features using an MLP, from which a relation matrix is inferred through a pairwise similarity function. Based on this matrix, a GNN [123] propagates information among agents by aggregating neighboring node features according to the inferred relation weights. This message-passing process produces interaction-aware representations for both the robot and humans, which are used for value estimation and future human-state prediction, respectively. The learned value and prediction models are further used in multi-step lookahead planning for safe and efficient action selection.

Chen et al. [124] further incorporated human attention into graph-based navigation. Instead of relying only on automatically inferred relations, the method learns attention weights from human gaze data and uses them to modulate the GNN adjacency matrix. The weighted graph allows the robot to aggregate crowd information according to the relative importance of nearby humans before value-based action selection. Jiang et al. [7] focused on edge-level asymmetric interactions and uses an edge-wise gating mechanism to control how much information is passed from each neighbor during message passing, enabling the policy to distinguish different influence levels in human-human and human-robot interactions.

b) *Spatio-Temporal Graph Methods:* These methods capture interactions among agents across both spatial and temporal dimensions. By encoding how inter-agent relations evolve over time, they help the robot anticipate changing interaction patterns and make navigation decisions with greater foresight.

Liu et al. [121] developed a Decentralized Structural-Recurrent Neural Network (DS-RNN) based on S-RNN [125]. The method constructs a spatio-temporal graph, where nodes represent agents, spatial edges encode human-robot relations at the same time step, and a temporal edge encodes the robot's motion continuity by connecting the robot node across two adjacent time steps. Separate RNN modules process spatial-edge, temporal-edge, and robot-node factors, and the resulting representation integrates spatial interactions, temporal motion information, and robot state for policy generation.

Unlike the above method, which models humans as independent individuals, Lu et al. [8] introduced group-aware spatio-temporal graph reasoning for crowd navigation. The method detects human groups and represents them as additional graph nodes, so that the graph contains the robot, obstacles, individual humans, and human groups. This allows the robot to reason about group-level social constraints, such as avoiding group intrusion, following or overtaking groups, and cooperatively passing oncoming groups. A spatio-temporal graph attention network estimates spatial relation strengths through attention, propagates information with GNNs, and uses LSTMs to capture the temporal evolution of these relations. The resulting spatio-temporal feature of the robot node is combined with the robot state for value-based action selection.

c) *Heterogeneous Graph Methods:* These methods model navigation scenes containing different types of entities (e.g., humans, robots, static obstacles) and their relations. By assigning type-specific node features and relation-specific message passing, these methods allow learned policies to distinguish dynamic-agent interactions, robot-robot interactions, and constraints imposed by static obstacles.

Liu et al. [122] modeled heterogeneous interactions between the robot and different types of obstacles using a heterogeneous spatial graph. Nodes represent the robot, humans, circular obstacles (e.g., trash bins), and line obstacles (e.g., walls), while edges encode interactions among these entities. A GNN aggregates heterogeneous node features using different transformations for different edge types, so that interactions with humans, circular obstacles, and line obstacles are encoded differently. The updated features are then fed into actor-critic networks to generate navigation actions.

Zhou et al. [9] extended heterogeneous graph reasoning to multi-robot crowd navigation. The method distinguishes the controlled robot, humans, and other robots, and constructs spatial graphs for different interaction types (e.g., human-

Table V: Comparison of representative interaction-aware methods.

Category	Reference	Year	Main Idea	Learned Model	Interaction Cue	Modeled Relation
Joint State-Based	[113]	2018	Learns a policy that maps joint states to a probability distribution over actions	LSTM-based actor-critic network	Robot state and LSTM-encoded agent states	Robot-agent
	[115]	2023	Incorporates agents' shape and speed heterogeneity into joint state and collision risk	Heterogeneity-aware value network	OBC-based distances and relative orientations	Robot-agent
Attention-Based	[116]	2019	Aggregates robot-human and human-human interaction features through self-attention	Self-attention pooling value network	Pairwise robot-human features, human-centered crowd maps	Robot-human, human-human
	[118]	2023	Uses transformers to encode multi-frame robot-human interactions	Transformer value network	Multi-frame robot-human joint states and spatio-temporal human motion cues	Spatio-temporal robot-human relation
Graph-Based	[120]	2020	Infers pairwise agent relations and propagates them through GNN for value estimation and human state prediction	Relational GNN with value estimation and motion prediction heads	Inferred relation matrix from latent robot and human state features	Robot-human, human-human
	[121]	2021	Converts a spatio-temporal crowd-navigation graph into separate RNN factors for policy learning	RNN-based actor-critic network	Robot-human spatial edge features, robot temporal features, robot node features	Robot-human spatial relations, robot's motion continuity
	[122]	2024	Models navigation scenes containing different types of obstacles and relations using heterogeneous graph	Heterogeneous GNN-based actor-critic network	Multi-type edge features	Relations among robot, humans, static obstacles

robot, human-human, and robot-robot). A GNN propagates and aggregates information across these graphs, producing a controlled-robot embedding that encodes interactions with humans and other robots. This embedding is then used by a value-based network for action selection.

Strengths and Limitations: Graph-based methods explicitly model interactions among humans, robots, and obstacles, enabling the learned policy to reason about their mutual influences during navigation. By encoding different interaction types (e.g., robot-human, human-human, robot-obstacle, and robot-robot), these methods are suitable for dense crowds, cluttered environments, and multi-robot scenarios. However, their performance depends on accurate graph construction and reliable perception inputs, such as agent detection and tracking, category recognition, and group identification. More expressive graph models also increase training complexity, limiting robustness in unseen or rapidly changing scenarios.

C. Hierarchical Learning Methods

Hierarchical learning methods decompose a learned navigation policy into high-level decision-making and low-level motion execution. The high-level policy provides intermediate guidance (e.g., subgoals), while the low-level policy converts this guidance into control commands. Table VI summarizes the key features of representative hierarchical learning methods.

Existing methods differ in the form of guidance produced by the high-level policy. Jing et al. [126] adopted a subgoal-based hierarchy for long-range navigation. The high-level policy generates subgoals that guide the robot away from dense crowds while maintaining progress toward destination. The low-level policy then takes the subgoal as guidance and outputs control actions for collision avoidance and motion execution. Chen et al. [127] trained the high-level policy to adapt navigation modes (e.g., obstacle avoidance or goal pursuit) according to environmental complexity. The method characterizes the surrounding environment using two LiDAR-based metrics, namely the variation rate and entropy of environment structure, and uses them to regulate navigation modes. Du et al. [129] built a hallway map based on the temporal arrival intents of nearby agents, allowing each agent to select a hallway by balancing travel distance and congestion. A learned congestion predictor then serves as a high-level policy that decides whether the agent should enter the selected hallway or wait. Okunevich et al. [131] used a high-level social module to assign a learned social value to the local trajectory induced by each low-level RL action. The final action is selected by considering the navigation value and the learned social value.

Other methods further improve how high-level decisions are trained or updated. Gao et al. [128] incorporated low-level execution feedback into high-level policy training. When the low-level policy fails to reach the selected subgoal due to collision or timeout, the high-level policy receives a penalty, encouraging it to select more feasible and reliable subgoals. Gao et al. [130] introduced a congestion-aware subgoal update mechanism, where local congestion is estimated from LiDAR observations and used to adapt the subgoal update threshold. Higher congestion prompts earlier subgoal updates in crowded regions, reducing lingering near the current subgoal and helping the robot escape crowded traps.

Strengths and Limitations: Hierarchical learning methods use high-level guidance to improve the learning efficiency and decision quality of low-level policies. However, their performance depends on the quality of this guidance, as poor high-level decisions may mislead the low-level controller. Training and coordinating multiple policies also introduce additional design complexity, and mismatches between hierarchy levels

Table VI: Comparison of representative hierarchical learning methods.

Reference	Year	Main Idea	Hierarchical Type	High-Level Decision	Low-Level Execution	Inter-Level Coupling
[126]	2024	Learns high-level subgoal selection and low-level control for long-range navigation	Subgoal-based hierarchy	Subgoal	Control command	Top-down
[127]	2024	Learns the high-level policy to adapt navigation modes according to the complexity of environment	Behavior-adaptive hierarchy	Behavior mode	Control command	Top-down
[128]	2024	Uses low-level collision or timeout failures as feedback for high-level subgoal learning	Feedback-enhanced hierarchy	Subgoal	Control command	Bidirectional
[129]	2025	Uses hallway-level congestion reasoning to decide when and where to pass through constrained areas	Congestion-aware hierarchy	Hallway enter/wait decision	Control command	Top-down
[130]	2025	Uses environment congestion to adaptively update subgoals	Congestion-aware hierarchy	Subgoal	Control command	Top-down

may degrade navigation performance.

IV. Learning-Augmented Classical Planning Methods

Learning-augmented classical planning methods retain classical planners as the primary decision-making modules for navigation, while using learned modules to provide auxiliary online information. In these methods, the learned module adapts planner parameters (e.g., objective-function weights and obstacle-clearance margins), provides search guidance, or predicts future obstacle motion. The classical planner then uses this learned information to generate the final trajectory or control command. According to the role of learning modules, these methods can be grouped into planner parameter learning, search guidance learning, and obstacle prediction methods. Table VII summarizes the key features of representative learning-augmented classical planning methods.

A. Planner Parameter Learning Methods

Planner parameter learning methods adapt the numerical parameters of classical planners using learning techniques. In these methods, the learned module adaptively adjusts parameters, such as objective-function weights, forward simulation horizons, and obstacle-clearance margins. These updated parameters are then used by the classical planner to compute the final trajectory or control command.

Zhu et al. [17] integrated RL with a classical trajectory optimization pipeline. A policy trained using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [132] adaptively adjusts objective weights and constraint-related parameters (e.g., safety distance threshold and maximum speed) based on LiDAR observations, robot states, previous actions, local goals, and coarse guidance information. The adapted parameters are then used by the optimization pipeline to compute the trajectory and corresponding control commands.

Han et al. [18] trained an RL policy to learn when and how to deviate from the nominal MPC-based action for collision avoidance. The MPC module generates a nominal action for reference path tracking, while a human avoidance component uses a gated recurrent unit to encode human trajectories and multi-head attention to model human-human and robot-human interactions, generating a collision-avoidance action. A learned fusion module then predicts a blending coefficient that combines human-avoidance action with the MPC action to produce the control command.

Some methods adapt the parameters of local planner using RL. Chang et al. [133] used Q-learning [134] to tune the weights of the DWA evaluation function and the forward simulation horizon according to the robot pose, goal position, and nearby-obstacle information. Dobrevski and Skočaj [19] further trained a deep RL policy to predict the weights of the DWA evaluation function from a short history of LiDAR measurements, goal position, and the robot velocities. The use of recent observations allows the policy to implicitly capture obstacle motion in dynamic scenes. In both methods, the final control commands are generated by DWA. Chang et al. [135] used a transformer model [136] to predict the desired obstacle-clearance margin from LiDAR point clouds and the robot's local path. The predicted margin adapts obstacle clearance to the available free space along the planned path and updates parameters of the Timed Elastic Band (TEB) [137], including the minimum obstacle distance and costmap inflation radius.

Strengths and Limitations: Planner parameter learning methods improve the adaptability of classical planners to changing environments by adjusting their parameters online. However, their performance depends on whether the learned parameters generalize to unseen scenarios. Poor parameter adaptation may make the planner unsafe or overly conservative. In addition, these methods may still require careful reward design, bounded parameter ranges, and diverse training scenarios.

B. Search Guidance Learning Methods

Search guidance learning methods use learned modules to bias the search or plan evaluation process of classical planners. In these methods, the learned module produces sampling distributions, confidence maps, reward maps, or risk values. The classical planner then uses these learned quantities to guide sampling, path extraction, or trajectory evaluation, while preserving its original planning mechanism.

Chen et al. [40] learned an online sampling distribution to guide the expansion of an RRT*-based search tree [33].

Table VII: Comparison of representative learning-augmented classical planning methods.

Category	Reference	Year	Main Idea	Classical Module	Learned Output	Final Output
Planner Parameter Learning	[133]	2021	Uses Q-learning to adaptively tune parameters of DWA evaluation function	DWA	Evaluation-function weights, forward simulation horizon	Control command
	[17]	2025	Uses learned policy to adaptively adjust optimization objectives based on environmental context	Trajectory optimization planner	Objective-function weights, constraint-related parameters	Local trajectory
	[18]	2025	Learns blending coefficients to fuse human-avoidance actions with MPC path-tracking actions	MPC	Action blending coefficients	Control command
	[135]	2025	Uses a transformer-based model to estimate safety margins for TEB parameter adaptation	TEB	Safety margin	Local trajectory
Search Guidance Learning	[40]	2019	Learns a sampling distribution from elite tree nodes to guide RRT*-based replanning	RRT*	GMM-based sampling distribution	Global path
	[138]	2023	Uses a learned collision-risk evaluator to guide DWA candidate action scoring	DWA	Collision risk	Control command
	[139]	2025	Learns socially informed reward maps to generate ORCA-trackable reference trajectories	ORCA	Reward map encoding scene navigability and implicit social preferences	Control command
Obstacle Prediction	[140]	2023	Incorporates human trajectory predictions into MPC cost function for proactive collision avoidance	MPC	Human trajectory predictions	Control command
	[141]	2023	Uses conformal prediction to calibrate learned trajectory forecasts for probabilistically safe MPC	MPC	Human trajectory predictions with conformal prediction regions	Control command
	[142]	2025	Integrates diffusion-based joint human trajectory prediction into a bilevel MPC that jointly optimizes robot actions and human predictions	MPC	Human trajectory predictions compatible with robot's plan	Control command

When the current path is invalidated by dynamic obstacles, the method reuses and repairs the existing search tree by pruning invalid nodes and edges while retaining reusable tree components. It then collects promising nodes into an elite set according to their cost-to-come and heuristic cost-to-go values, and fits a Gaussian mixture model [143] to these elite samples. New samples are drawn from the learned distribution, together with goal-biased and random samples, to focus tree expansion on promising regions while maintaining sampling diversity.

Qin et al. [144] trained a CNN through supervised learning to predict a confidence map from a global path, LiDAR measurements, robot odometry, and human states. The confidence map encodes socially compliant path preferences and is converted into a cost map, from which an A*-based search extracts a local path for the low-level controller. Kathuria et al. [139] used inverse RL to learn a reward map from few-shot expert demonstrations, encoding scene navigability and implicit social interaction preferences. During online deployment, a short-horizon reference trajectory is generated from the reward map and executed by an ORCA-based local controller.

Yang et al. [138] combined an RL-based risk evaluation network with DWA. The method first computes an occupancy grid map, where each grid value indicates the probability of human occupancy within a future time window. A risk evaluation network then estimates human risk from this map. The DWA simulates short-horizon trajectories for candidate velocity actions and scores them by combining the learned human risk with static-obstacle clearance. The robot executes the action with the highest score.

Strengths and Limitations: Search guidance learning methods encode useful planning knowledge through sampling distributions, confidence maps, or reward maps, thereby biasing the search process toward promising regions. This improves the efficiency and adaptability of classical planners in dynamic environments. However, their performance depends heavily on the quality of the learned guidance. Inaccurate or outdated guidance may mislead the planner in rapidly changing environments. These methods often require representative training data or additional mechanisms to ensure that learned guidance does not compromise safety or completeness.

C. Obstacle Prediction Methods

Obstacle prediction methods provide anticipatory information for proactive replanning in dynamic environments. Classical methods, such as Kalman filter [145] and particle filter [146], can predict obstacle states and uncertainty. However, they typically rely on assumptions about the underlying motion model and probability distribution, which may not hold in real-world applications. In contrast, learning-based methods [147]–[150] can predict complex motion patterns from data without specifying motion dynamics or distributional forms.

Nishimura et al. [151] used learning-based human forecasts to induce a stochastic collision-cost distribution and optimized an entropic risk measure over this distribution within the MPC

Table VIII: Comparison of hybrid coupling strategies.

Hybrid Type	Main Idea	Coupling Structure	Operational Level	Classical Output	Learned Output
Global-Local Planning	Uses a classical global planner for long-range guidance and a learned module for local execution	Hierarchical	Classical and learned modules operate at different levels	Global path	Control command, local trajectory
Multi-Policy Adaptation	Selects among learned and classical policies according to the navigation contexts	Parallel	Classical and learned modules operate at the same level	Control command, waypoint	Control command, policy switching decision, waypoint
Hybrid Local Planning	Embeds learned intermediate quantities into local planning for final command generation	Cascaded	Classical and learned modules operate at the same level	Corrected, optimized, or dynamically feasible control command	Nominal control command, velocity selection cue, intermediate reference

framework. This allows the controller to account for both the expected collision cost and its uncertainty. Heuer et al. [140] added a penalty term to the MPC cost function to discourage candidate actions that would bring the robot close to predicted human positions. However, in interactive crowd navigation, human motion prediction and robot planning are inherently coupled, as the robot's planned motion may influence human responses, while predicted human motion affects the safety and efficiency of the robot plan. Samavi et al. [142] addressed this issue by integrating human trajectory prediction with MPC-based motion planning. The method first uses a diffusion model [152] to generate joint trajectory prediction samples for all humans in the scene. These samples are then incorporated into a bilevel MPC formulation that jointly optimizes the robot plan and ORCA-refined human predictions, with the upper level optimizing robot actions and the lower-level ORCA problems enforcing collision-free predicted human motion with respect to other agents and static obstacles.

A remaining limitation is that many learning-based predictors do not explicitly provide calibrated uncertainty estimates, which may lead to unsafe or unnecessarily conservative planning decisions when the prediction is inaccurate. Lindemann et al. [141] addressed this issue by using conformal prediction [153] to quantify the uncertainty of learning-based trajectory predictions. The method constructs valid prediction regions around the predicted positions of dynamic obstacles using prediction errors measured on a calibration dataset. These regions are then incorporated into an MPC formulation. By planning trajectories that avoid these regions, the MPC can provide probabilistic safety guaranties while still exploiting learned trajectory forecasts. Strawn et al. [154] extended conformal prediction to a RL-based planning framework by proposing a conformal predictive safety filter for pre-trained learning-based controllers. The method constructs conformal prediction regions around the predicted positions of dynamic agents. A safety filter is then trained to closely follow the nominal RL control sequence while modifying it when necessary to avoid the predicted agents and their uncertainty regions.

Strengths and Limitations: Obstacle predictions enable planners to make anticipatory decisions by reasoning about future obstacle motions. However, their performance may degrade in highly dynamic or densely populated environments, where accurate prediction is hindered by limited sensing, high obstacle density, and computational constraints.

V. Hybrid Planning Methods

Hybrid planning methods integrate learned modules with classical planning in the online navigation pipeline. This distinguishes them from direct-policy learning methods, where learned models serve as the primary decision maker. Within direct-policy learning, hierarchical learning methods decompose navigation into high-level decision-making and low-level execution, but both levels are typically learned modules. In contrast, hybrid planning integrates learned modules with classical planners through hierarchical, parallel, or cascaded coupling structures. It also differs from learning-enhanced classical planning, where the classical planner remains the primary decision maker and learning mainly provides auxiliary information. In hybrid planning, learned and classical components jointly determine online navigation behavior.

In hybrid planning methods, learned modules may provide navigation actions, short-horizon trajectories, or intermediate references, while classical components provide global paths or control commands. According to the coupling strategy between learning and classical planning, existing methods can be grouped into global-local planning, multi-policy adaptation, and hybrid local planning methods. The key differences among these coupling strategies are summarized in Table VIII, while Table IX compares representative hybrid planning methods.

A. Global-Local Planning Methods

Global-local planning methods decompose navigation into a global guidance layer and a local execution layer. In these methods, classical planners are used as global planners to generate geometric paths towards the goal. Learned modules then operate at the local level, using onboard observations and global guidance to generate navigation actions or short-horizon trajectories. This design allows the robot to track the global path while adapting to nearby dynamic obstacles, collision-avoidance requirements, and kinodynamic constraints.

Wang et al. [155] used the A* algorithm [51] to generate a global path and incorporated it into an RL-based local planner. The method first encodes local observations as image channels, while the global guidance is added as an additional guidance channel. These channels form the input at each time step. A sequence of such inputs over recent time steps is then fed into a Double Deep Q-Learning Network (DDQN) [156]. The network uses 3D CNN layers to extract spatial features and an LSTM layer to capture temporal information, and outputs

the discrete motion actions. Guldenring et al. [157] adopted a similar hybrid framework. A classical global planner provides waypoint guidance. An RL-based local planner is trained using PPO. The learned local planner takes LiDAR observations and upcoming waypoints as inputs, and outputs control commands to follow the global path while avoiding obstacle collisions.

Angulo et al. [158] proposed an RL-based steering function for classical planners (e.g., RRT [32]). Given LiDAR measurements, current and target state information, the learned policy generates kinodynamically feasible local trajectory to connect two states during planning, enabling the planner to account for non-holonomic constraints as well as static and dynamic obstacles. Xu et al. [20] combined RRT$^{\mathrm{X}}$ algorithm [10] with a Q-learning-based local controller for kinodynamic motion replanning in dynamic environments. RRT$^{\mathrm{X}}$ algorithm maintains and repairs a global path as the environment changes, and decomposes the path into a set of boundary-value problems between consecutive waypoints. For each boundary-value problem, the Q-learning controller drives the robot toward the next waypoint under unknown system dynamics, external disturbances, and intermittent control updates.

Strengths and Limitations: Global-local planning methods take advantage of both the long-range guidance and interpretability of classical global planners and the adaptability of learned models. However, their performance degrades when the global guidance becomes outdated in rapidly changing scenarios. Learned local modules may also deviate from the global guidance, and their effectiveness depends on policy generalization and compatibility with the global planner.

B. Multi-Policy Adaptation Methods

Multi-policy adaptation methods combine classical planning modules with learned modules at the same operational level. These modules may produce low-level commands, such as velocities, or higher-level decisions, such as intermediate waypoints. During execution, navigation behavior is adapted online by selecting between learning-based and classical planning-based policies according to the navigation context.

Fan et al. [21] introduced a scenario-aware policy adaptation method that classifies navigation contexts into simple, complex, and emergent scenarios based on the distance to nearby obstacles and predefined distance thresholds. Thus, to maintain simplicity, the robot switches its policy depending on the scenario, such as a PID controller for simple scenarios, the RL-based policy for complex scenarios, and a conservative policy for emergent scenarios. Semnani et al. [162] adopted a similar framework, which selects control commands using an RL-based policy under normal scenarios, while switching to a classical method [163] under high-risk situations.

Unlike the above methods that rely on hand-crafted policy switching rules, Wu et al. [159] learned both the navigation policies and the risk-aware switching method, allowing policy selection to adapt to different navigation contexts. A flow-based density model [164] first estimates the probability density of latent feature-action pairs from the offline dataset. A higher density indicates that the corresponding state-action pair is more consistent with the training data. The density information is then used to train a Lyapunov density model (LDM) [165], which predicts whether the future state-action trajectory will remain within the training-data distribution. During execution, the robot follows the learned policy when the LDM indicates low distributional-shift risk; otherwise, it switches to a safe action (e.g., emergency stop). Matsumoto et al. [166] proposed a learning-based policy switching method using graph normalizing flows [167]. The human states are encoded and passed through a graph normalizing flow to compute a likelihood-based switching score. During execution, the robot uses the learning-based navigation policy if the current human configuration has a high likelihood under the training distribution, and switches to ORCA-based policy otherwise. Zhu and Hayashibe [168] proposed a learned safety supervisor, which decides whether to execute the learned policy or switch to a conservative maneuver for safe obstacle avoidance.

Different from the above methods that switch among local control policies, Hong et al. [22] proposed a waypoint adaptation strategy that modifies intermediate navigation goals. The method uses a Conditional Variational Autoencoder (CVAE) [169] to learn collision-avoidance waypoint distributions based on the static obstacles, human states, and robot's start and goal poses. During execution, if no human conflict is detected, the robot follows the waypoints provided by the classical global planner; otherwise, it navigates to a CVAE-generated waypoint for collision avoidance.

Strengths and Limitations: Multi-policy adaptation methods combine the adaptability of learned policies with the reliability of classical planners, improving robustness across diverse navigation contexts and reducing reliance on a single policy. However, their performance depends on reliable scenario or risk assessment. Hand-crafted policy switching rules may not generalize well to different environments, while learning-based switching strategies require representative training data. Moreover, frequent or incorrect switching may lead to conservative, oscillatory, or inconsistent behaviors.

C. Hybrid Local Planning Methods

Hybrid local planning methods adopt a cascaded coupling between learned modules and classical local planners. In cascaded coupling, a learned module may first generate a nominal action, which is then corrected by a classical module to produce the final executable command. Alternatively, a classical module may first construct a dynamically feasible control command, from which the learned policy obtains the final command for collision avoidance.

Sathyamoorthy et al. [160] used a model-based correction module to refine the nominal control commands generated by a learned policy, aiming to reduce the freezing robot problem. The method tracks humans, predicts their future positions, and constructs the Potential Freezing Zone (PFZ), which represents a conservative region where the robot may become stuck. Based on the learned control command, the model-based module applies an angular correction to avoid entering the freezing zone while maintaining progress toward the goal, and outputs the corrected velocity as the final navigation command.

Patel et al. [161] used DWA to construct the dynamically feasible velocity set and RL to select the final command from

Table IX: Comparison of representative hybrid planning methods.

Category	Reference	Year	Main Idea	Classical Module	Classical Output	Learned Output
Global-Local Planning	[155]	2020	Incorporates A*-based global guidance into a RL-based local planner for obstacle avoidance	A*	Global path	Control command
	[158]	2023	Learns an RL-based steering function to generate adaptive motion primitives for kino-dynamic replanning	RRT	Target state pairs for local connection	Kinodynamically feasible local trajectory
	[20]	2024	Uses RRT^{X} to maintain a collision-free global path and uses a Q-learning-based local controller to follow waypoint sequence under unknown dynamics and disturbances	RRT^{X}	Global path	Control command
Multi-Policy Adaptation	[21]	2020	Switches among PID, RL, and conservative policies according to the scenario complexity	PID controller	Control command	Control command
	[159]	2023	Learns navigation policies and risk-aware policy switching based on distributional-shift estimation	Safety fallback mechanism	Safe action	Risk assessment, control command
	[22]	2023	Uses CVAE-generated temporary waypoints for local dynamic obstacle avoidance and resumes navigation along the original path	A*	Global path	Waypoint for dynamic obstacle avoidance
Hybrid local Planning	[160]	2020	Uses a model-based module to correct RL-generated velocities and avoid PFZs in human crowds	PFZ-based velocity correction module	Corrected velocity away from PFZ	Nominal velocity for collision avoidance
	[161]	2021	Uses DWA to construct a dynamically feasible velocity space and trains RL policy to select velocity for dynamic obstacle avoidance	DWA	Dynamically feasible velocity candidates	Final velocity for collision avoidance
	[81]	2024	Learns an escape velocity that encodes the robot's collision-avoidance responsibility and moving direction, and uses ORCA to compute final velocity	ORCA	Goal-directed final velocity	Nominal velocity for collision avoidance

this set. The method first generates velocity candidates that satisfy the robot's acceleration and nonholonomic constraints. For each candidate velocity, it evaluates safety and goal-progress information over recent time steps, forming a compact observation that describes the quality of this candidate. Given this observation, the learned policy selects one feasible velocity pair for execution.

Brito et al. [176] integrated a learned intermediate-reference module into MPC-based local planning in dynamic environments. The learned policy encodes the robot state, nearby-agent states, and the global goal, and outputs a local position increment as an intermediate reference for MPC. This reference captures short-term interaction effects with surrounding agents, while MPC optimizes local control commands under kinodynamic and collision-avoidance constraints.

Liu et al. [81] integrated RL with ORCA to implicitly learn the robot's collision-avoidance responsibility and avoidance direction in crowds. Instead of assuming that each agent takes half of the avoidance responsibility, a PPO-trained policy infers an escape velocity from the robot state and nearby-human states. The magnitude of this escape velocity reflects the robot's avoidance responsibility, while its direction determines the avoidance direction. ORCA then uses the learned escape velocity to construct a collision-avoidance constraint and compute the final robot velocity.

Xu et al. [177] trained a PPO-based UAV navigation policy using separate representations for static and dynamic obstacles. Static obstacles are encoded as a voxel map, while dynamic obstacles are represented by bounding boxes, reducing the sim-to-real gap compared with raw image inputs. The learned policy outputs goal-directed velocity commands, which are checked by a VO-based safety shield and projected into safe velocity regions when they may lead to future collisions.

Srisuchinnawong et al. [178] introduced an online unsupervised learning module to augment classical local planners. The module correlates obstacle feedback with the local planner command and learns a proactive correction command at a higher update rate. The learned command is added to the planner command, allowing the robot to compensate for limited update rates of local planners and motion model mismatch while balancing smooth motion and collision avoidance.

Strengths and Limitations: Hybrid local planning methods use cascaded coupling between learned modules and classical local planners to decouple planning requirements (e.g., dynamic feasibility, collision avoidance, and goal-directed motion). This allows each module to focus on the aspect it handles best, thereby improving the robustness, interpretability, and deployability. However, their performance depends on the reliability of learned intermediate quantities and may still be constrained by the assumptions, constraint formulations, or

Table X: Comparison of representative training enhancement methods.

Category	Reference	Year	Main Idea	Generalization Aspect	Learning Formulation	Enhancement Support
Expert-Guided Training	[170]	2018	Uses BC to initialize a navigation policy from expert demonstrations and refines it through GAIL	Socially compliant navigation across common pedestrian-interaction scenarios	BC + GAIL	BC-pretrained policy from expert demonstrations
	[171]	2021	Uses knowledge distillation to shape the reinforcement learning reward	Human-like collision avoidance in unseen multi-agent scenarios	Knowledge distillation	Expert policy-based reward shaping from human demonstrations
	[172]	2024	Uses IL to initialize a navigation policy from expert demonstrations and refines it through TRL	Non-homogeneous scenarios	IL + TRL	IL-pretrained policy from expert demonstrations
Scenario Diversification	[173]	2021	Improves generalization by training on simple canonical indoor layouts that compose into more complex scenarios	Unseen complex indoor environments	Compositional multi-layout learning strategy	Set of canonical layouts with common geometric patterns
	[174]	2025	Improves generalization by training policy with diverse human behaviors generated via domain randomization	Diverse pedestrian behaviors and real-world human-robot interactions	Domain randomization	Randomized ORCA pedestrian-behavior parameters
	[175]	2025	Improves policy robustness by generating diverse yet realistic simulated human behaviors via diversity-aware crowd model	Unseen crowd scenarios	RL-based diverse crowd generation	Diversity-aware crowd model

local minima of the underlying classical planner.

VI. Training Enhancement Methods

Training enhancement methods improve learning-based navigation policies during the training process. These methods introduce expert guidance or diversify the training scenarios to reduce exploration difficulty and improve robustness and generalization. According to the type of training support, existing methods can be broadly divided into expert-guided training and scenario diversification methods. Table X summarizes the key features of representative training enhancement methods.

A. Expert-Guided Training Methods

Expert-guided training methods incorporate expert demonstrations into the training process of RL-based navigation policies to reduce exploration difficulty and impose useful navigation priors.

Tai et al. [170] used expert demonstrations to improve the training efficiency and social compliance of navigation policies. The method uses Behavior Cloning (BC) [179] to train an initial policy from expert trajectories. It then uses Generative Adversarial Imitation Learning (GAIL) [180] to refine the policy by learning from entire trajectories and capturing how consecutive observations and actions evolve over time, thereby improving temporal consistency and social compliance. Xu and Karamouzas [171] incorporated human demonstrations as soft guidance through knowledge distillation [181]. The method first derives expert policies from human demonstrations. Instead of cloning these expert policies, it applies knowledge distillation to shape the RL reward based on the deviation between the learned action and expert actions. This allows the policy to benefit from imperfect demonstrations without being strictly constrained by them. Qin et al. [172] used a model-based expert policy to improve the training efficiency and generalization of the low-level collision avoidance policy. The method first trains an Imitation Learning (IL)-based policy [182] to learn expert collision-avoidance actions from NH-ORCA [183]. The initial policy is then fine-tuned through Transfer Reinforcement Learning (TRL) [184] under non-homogeneous scenarios.

Strengths and Limitations: Expert-guided policy learning helps improve training efficiency by providing useful prior behaviors. However, its performance depends on the quality and coverage of the expert information. Policies may also generalize poorly when deployed in scenarios that differ from the expert demonstrations or training environments.

B. Scenario Diversification Methods

Scenario diversification methods improve policy generalization by exposing learning-based navigation policies to diverse static obstacle layouts and human behaviors during training.

Perez et al. [173] improved policy generalization to indoor environments with complex layouts through a compositional multi-layout training strategy. Instead of manually designing complex training scenarios, the method trains the policy on a set of simple layouts (e.g., corridors, doorways, and crossings). These layouts capture common geometric patterns in the environments, enabling the learned policy to generalize to unseen scenarios composed of similar structural elements.

Sen et al. [174] improved the generalization of human-aware robot navigation through domain randomization. The method randomizes ORCA parameters that control human behaviors, including human radius, velocity, and time horizon. These variations expose the policy to different personal-space requirements, walking speeds, and anticipatory avoidance be-

haviors, thereby improving its adaptability to diverse human behaviors in real-world environments.

Wu et al. [175] generated diverse human behaviors through a diversity-aware crowd model. The method assigns latent control codes to simulated human policies and trains a discriminator to infer these codes from trajectory features. By rewarding trajectories that are distinguishable across different codes, the model encourages diverse behaviors, such as different separation distances. A constraint is further imposed to keep these behaviors goal-directed and collision-aware rather than unrealistic or random.

Strengths and Limitations: Scenario diversification improves the generalization of learned policies by expanding the training distribution over obstacle layouts and human behaviors. However, its effectiveness depends on how well the diversified scenarios cover real-world variations. Hand-crafted layouts or simulated crowd behaviors may still miss rare interactions, complex social norms, or unexpected human reactions.

VII. Conclusion and Future Directions

This paper presents a comprehensive review of representative works published primarily between 2015 and 2025. Classical planning methods are first revisited as algorithmic foundations, followed by a structured review of direct-policy learning, learning-augmented classical planning, hybrid planning, and training enhancement methods. For each category, this review summarizes the main problem settings, representative algorithms, key ideas, integration mechanisms, strengths, and limitations. Overall, recent studies show that learning is no longer limited to replacing classical planners with end-to-end policies. Learned models can serve as primary navigation policies, provide planner parameters, search guidance, obstacle predictions, intermediate references, policy-switching decisions, or training support. This diversity has expanded the design space of motion planning in dynamic environments, enabling better perception utilization, interaction modeling, online adaptation, and integration with classical planning structures. At the same time, existing methods still face challenges in generalization, safety verification, sim-to-real transfer, dense crowd navigation, prediction uncertainty, and robust learning-classical integration.

The current challenges of learning-based methods motivate several future research directions, as discussed below.

A. Sim-to-Real Transfer and Real-World Robustness

Although learning-based motion planning methods have achieved promising simulation results, their deployment in real-world dynamic environments remains challenging. The sim-to-real gap arises from noisy sensor observations, localization and tracking errors, actuation delays, robot-model mismatch, and unmodeled environmental factors. This gap is further amplified by the random behavioral patterns of interacting agents. For example, pedestrians, vehicles, and robots may suddenly stop, change direction, yield, ignore the robot, or move in groups in ways that are difficult to reproduce in simulation. As a result, policies trained in simplified or idealized simulators may exhibit degraded performance, unsafe reactions, or overly conservative behaviors on physical robots.

Future research should develop learning-based planning systems that are explicitly designed for robust real-world deployment. Promising directions include high-fidelity simulators with realistic sensing, actuation, and crowd behaviors; domain randomization and domain adaptation to reduce distribution mismatch; real-world demonstrations and human-in-the-loop data collection to capture natural interaction patterns; and online adaptation or continual learning to improve policy robustness after deployment. In addition, learning-based planners should be combined with classical safety layers, such as MPC, CBFs, ORCA, DWA, or emergency-stop mechanisms, to handle unexpected failures during real-world execution.

B. Safe and Certifiable Learning-Based Planning

Safety verification remains a critical challenge for learning-based motion planning in dynamic environments. Direct-policy learning methods, especially RL-based policies, often use high-dimensional nonlinear models whose decision boundaries are difficult to interpret or formally verify. Their outputs can be sensitive to perception errors, prediction uncertainty, distribution shifts, and unseen interaction patterns. This issue becomes more challenging in dynamic environments, where moving obstacles and human agents continuously change the feasible motion space, and closed-loop interactions between the robot and surrounding agents can lead to unpredictable future states. As a result, empirical metrics such as success rate or collision rate are insufficient to fully characterize whether a learned planner can remain safe under rare, uncertain, or adversarial conditions.

Future research should move from empirical collision avoidance toward certifiable learning-based planning. One promising direction is to combine learned policies with formal safety mechanisms, such as MPC constraints, control barrier functions, reachability analysis, and conformal prediction. These mechanisms can provide safety envelopes or corrective actions when learned modules generate risky decisions. Another direction is to quantify and propagate uncertainty from perception, prediction, and policy outputs into the planning process, so that the robot can reason about risk rather than relying on deterministic predictions. Future systems should include failure detection and recovery mechanisms that identify when the learned module is operating outside its reliable domain.

C. Dense Crowd and Multi-Agent Interaction Modeling

Dense crowd navigation remains challenging for learning-based motion planning. In such environments, motion planning is not merely a multi-obstacle avoidance problem, but a closed-loop multi-agent interaction problem. The robot must reason about heterogeneous interactions among humans, robots, obstacles, and groups, while also considering social norms, occlusions, and uncertain human responses. Existing interaction-aware methods have improved relational reasoning through joint-state representations, attention mechanisms, and graph-based models. However, many methods still rely on pairwise interaction modeling or local agent-level features, which may be insufficient for representing crowd-level patterns such as group motion, lane formation, bottlenecks, queues,

and opposing flows. In dense crowds, perception and tracking errors also become more frequent due to occlusions, and the computational cost of attention or graph-based reasoning can increase rapidly with the number of nearby agents.

Future research should develop scalable and socially aware interaction models for dense multi-agent environments. One direction is to move beyond individual-agent representations and learn crowd-level structures, such as density fields, flow patterns, group formations, and congestion regions. Another direction is to design more efficient graph and attention mechanisms, including sparse graphs, hierarchical graphs, agent clustering, and local-global attention, to support real-time planning in crowded scenes. In addition, learning-based planners should better couple interaction-aware prediction with motion planning, so that the robot can reason about how its own actions may influence surrounding agents.

D. Active Perception and Planning Coupling

Many existing learning-based motion planning methods treat perception as an upstream module that provides inputs to the planner, such as detected obstacle states, predicted trajectories, risk maps, or semantic information. However, this sequential pipeline can be insufficient in dynamic environments. The robot's motion not only determines its future position, but also affects what it can observe and how surrounding agents may react. Occlusions, limited fields of view, tracking failures, and prediction uncertainty can cause the planner to make short-sighted or unsafe decisions if they are not explicitly considered during planning. Therefore, motion planning in dynamic environments should not only react to current observations and predicted trajectories, but also actively choose motions that improve future observability and reduce prediction uncertainty.

Future research should further couple active perception, prediction, and planning in a unified framework. One direction is to develop perception-aware planning methods that consider information gain, visibility improvement, tracking reliability, and occlusion reduction together with goal progress and collision avoidance. Another direction is to incorporate prediction uncertainty directly into planning costs, constraints, or risk measures, so that the robot can slow down, wait, or select more conservative paths when future agent motions are highly uncertain. In addition, prediction models should consider how surrounding agents may react to the robot's planned motions, since pedestrians, vehicles, and other robots may change their behaviors when the robot slows down, yields, detours, or approaches them. Belief-space planning, POMDP-based formulations, and learning-based active sensing policies may provide useful tools for reasoning under partial observability.

E. Embodied AI for Semantic and Task-Level Planning

Existing learning-based motion planning methods focus on mapping sensor observations, agent states, or interaction representations to navigation actions. However, real-world dynamic environments often require more than geometric collision avoidance and local interaction modeling. Robots must also understand scene semantics, task contexts, human intentions, and social norms. For example, a robot may need to distinguish between walkable and restricted areas, infer whether pedestrians are queuing or crossing, decide whether to wait or detour, and adapt its navigation behavior according to language instructions or task-level requirements. Embodied AI techniques, including vision-language models, large language models, world models, and foundation models, provide new opportunities to incorporate semantic understanding and high-level reasoning into motion planning.

Future research should integrate embodied AI with motion planning in a grounded and safety-aware manner. Rather than directly replacing low-level planners or controllers, embodied AI modules can provide semantic maps, task constraints, intermediate goals, social rules, candidate behaviors, or planner-switching decisions for classical planners, learned policies, or hybrid planning frameworks. Such integration may help robots connect language instructions, visual perception, human intention reasoning, and executable motion plans in dynamic environments. However, several challenges remain. Foundation models may produce hallucinated or poorly grounded outputs, and high-level semantic decisions must be translated into constraints, costs, or references that are feasible, verifiable, and compatible with real-time planning. Therefore, embodied AI should be coupled with reliable low-level planning, safety filters, and uncertainty-aware verification mechanisms.

REFERENCES

[1] J. Z. Hare, J. Song, S. Gupta, and T. A. Wettergren, "POSE.R: Prediction-based opportunistic sensing for resilient and efficient sensor networks," *ACM Transactions on Sensor Networks*, vol. 17, no. 1, pp. 1–41, 2020.

[2] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 285–292, 2017.

[3] J. Liu, J. Yang, J. Mao, T. Zhu, Q. Xie, Y. Li, X. Wang, and S. Li, "Flexible active safety motion control for robotic obstacle avoidance: A CBF-guided MPC approach," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2686–2693, 2025.

[4] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, "Robust mader: Decentralized multiagent trajectory planner robust to communication delay in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1476–1483, 2024.

[5] Y. Sun, W. Zuo, H. Huang, P. Cai, and M. Liu, "Pointmoseg: Sparse tensor-based end-to-end moving-obstacle segmentation in 3-d lidar point clouds for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 510–517, 2021.

[6] Z. Shen, J. P. Wilson, S. Gupta, and R. Harvey, "SMART: Self-morphing adaptive replanning tree," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7312–7319, 2023.

[7] H. Jiang, N. Bhujel, Z. Lin, K.-W. Wan, J. Li, S. Jayavelu, and X. Jiang, "Learning relation in crowd using gated graph convolutional networks for drl-based robot navigation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 6, pp. 5085–5095, 2024.

[8] X. Lu, A. Faragasso, Y. Wang, A. Yamashita, and H. Asama, "Group-aware robot navigation in crowds using spatio-temporal graph attention network with deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 4140–4147, 2025.

[9] X. Zhou, S. Piao, W. Chi, L. Chen, and W. Li, "HeR-DRL:heterogeneous relational deep reinforcement learning for single-robot and multi-robot crowd navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4524–4531, 2025.

[10] M. Otte and E. Frazzoli, "RRT$^{\mathrm{X}}$: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[11] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[12] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[13] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, pp. 207–222, 2002.

[14] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[15] Z. Xie, P. Xin, and P. Dames, "Towards safe navigation through crowded dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4934–4940, 2021.

[16] 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.

[17] Z. Zhu, R. Wang, Y. Wang, Y. Wang, and X. Zhang, "Environment-adaptive motion planning via reinforcement learning-based trajectory optimization," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 16704–16715, 2025.

[18] J. R. Han, H. Thomas, J. Zhang, N. Rhinehart, and T. D. Barfoot, "DR-MPC: Deep residual model predictive control for real-world social navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 4029–4036, 2025.

[19] M. Dobrevski and D. Skočaj, "Dynamic adaptive dynamic window approach," *IEEE Transactions on Robotics*, vol. 40, pp. 3068–3081, 2024.

[20] Z. Xu, G. P. Kontoudis, and K. G. Vamvoudakis, "Online and robust intermittent motion planning in dynamic and changing environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 12, pp. 17425–17439, 2024.

[21] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *International Journal of Robotics Research*, vol. 39, no. 7, pp. 856–892, 2020.

[22] Y. Hong, Z. Ding, Y. Yuan, W. Chi, and L. Sun, "Obstacle avoidance learning for robot motion planning in human–robot integration environments," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 4, pp. 2169–2178, 2023.

[23] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018.

[24] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 295–318, 2021.

[25] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.

[26] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023.

[27] F. Kamil, S. Tang, W. Khaksar, N. Zulkifli, and S. A. Ahmad, "A review on motion planning and obstacle avoidance approaches in dynamic environments," *Advances in Robotics & Automation*, vol. 4, no. 2, pp. 134–142, 2015.

[28] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[29] H. S. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 353–365, 2022.

[30] J. Liu, H. J. Yap, and A. S. M. Khairuddin, "Review on motion planning of robotic manipulator in dynamic environments," *Journal of Sensors*, vol. 2024, no. 1, p. 5969512, 2024.

[31] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[32] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[34] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2383–2388, 2002.

[35] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *IEEE International Conference on Robotics and Automation*, pp. 1243–1248, 2006.

[36] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 1603–1609, 2007.

[37] L. Huang and X. Jing, "Fast asymptotically optimal path planning in dynamic, uncertain environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2846–2852, 2023.

[38] L. Huang and X. Jing, "Asymptotically optimal lazy lifelong sampling-based algorithm for efficient motion planning in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8861–8867, 2024.

[39] J. Silveira, K. Cabral, S. Givigi, and J. A. Marshall, "Real-time fast marching tree for mobile robot motion planning in dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 7837–7843, 2023.

[40] Y. Chen, Z. He, and S. Li, "Horizon-based lazy optimal RRT for fast, efficient replanning in dynamic environment," *Autonomous Robots*, vol. 43, no. 8, pp. 2271–2292, 2019.

[41] C. Yuan, G. Liu, W. Zhang, and X. Pan, "An efficient RRT cache method in dynamic environments for path planning," *Robotics and Autonomous Systems*, vol. 131, p. 103595, 2020.

[42] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7244–7251, 2021.

[43] C.-C. Lee and K.-T. Song, "Path re-planning design of a cobot in a dynamic environment based on current obstacle configuration," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1183–1190, 2023.

[44] B. Cui, R. Cui, W. Yan, Y. Wang, and S. Zhang, "RT-RRT: Reverse tree guided real-time path planning/replanning in unpredictable dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5380–5387, 2024.

[45] H.-T. L. Chiang, B. HomChaudhuri, A. P. Vinod, M. Oishi, and L. Tapia, "Dynamic risk tolerance: Motion planning by balancing short-term and long-term stochastic dynamic predictions," in *IEEE International Conference on Robotics and Automation*, pp. 3762–3769, 2017.

[46] W. Chi, C. Wang, J. Wang, and M. Q.-H. Meng, "Risk-DTRRT-based optimal motion planning algorithm for mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1271–1288, 2019.

[47] K. Cai, W. Chen, C. Wang, S. Song, and M. Q.-H. Meng, "Human-aware path planning with improved virtual doppler method in highly dynamic environments," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1304–1321, 2023.

[48] A. Hakobyan and I. Yang, "Distributionally robust risk map for learning-based motion planning and control: A semidefinite programming approach," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 718–737, 2023.

[49] Z. Sun, B. Lei, P. Xie, F. Liu, J. Gao, Y. Zhang, and J. Wang, "Multi-Risk-RRT: An efficient motion planning algorithm for robotic autonomous luggage trolley collection at airports," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 2, pp. 3450–3463, 2024.

[50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[51] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[52] A. Stentz *et al.*, "The focussed D* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[53] S. Koenig and M. Likhachev, "D* lite," in *18th National Conference on Artificial Intelligence*, pp. 476–483, 2002.

[54] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective path-based D* lite," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3318–3325, 2022.

[55] C. Li, H. Ma, J. Wang, and M. Q.-H. Meng, "Bidirectional search strategy for incremental search-based path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7311–7317, 2023.

[56] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 5628–5635, 2011.

[57] M. Phillips and M. Likhachev, "Planning in domains with cost function dependent actions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, pp. 74–80, 2011.

[58] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4708–4715, 2012.

[59] J. P. Gonzalez, A. Dornbush, and M. Likhachev, "Using state dominance for path planning in dynamic environments with moving obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 4009–4015, 2012.

[60] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, 2022.

[61] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *IEEE International Conference on Robotics and Automation*, pp. 5551–5557, 2019.

[62] S. Huang, Y. Wu, Y. Tao, and V. Kumar, "Safe interval motion planning for quadrotors in dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 2780–2786, 2025.

[63] J. Lin, T. Zhou, D. Zhu, J. Liu, and M. Q.-H. Meng, "Search-based online trajectory planning for car-like robots in highly dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 8151–8157, 2021.

[64] G. Chen, S. Wu, M. Shi, W. Dong, H. Zhu, and J. Alonso-Mora, "RAST: Risk-aware spatio-temporal safety corridors for MAV navigation in dynamic uncertain environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 808–815, 2023.

[65] G. Chen, P. Peng, P. Zhang, and W. Dong, "Risk-aware trajectory sampling for quadrotor obstacle avoidance in dynamic environments," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 12, pp. 12606–12615, 2023.

[66] Y. Qi, B. He, R. Wang, L. Wang, and Y. Xu, "Hierarchical motion planning for autonomous vehicles in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 496–503, 2023.

[67] E. Wiman and M. Tiger, "Safe lattice planning for motion planning with dynamic obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 9287–9294, 2025.

[68] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3620–3625, 2020.

[69] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[70] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[71] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for MAVs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[72] O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Scenario-based trajectory optimization in uncertain dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5389–5396, 2021.

[73] Z. Jian, Z. Yan, X. Lei, Z. Lu, B. Lan, X. Wang, and B. Liang, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *IEEE International Conference on Robotics and Automation*, pp. 3679–3685, 2023.

[74] Z. Huang, R. Li, W. Chen, Z. Lin, Z. Wu, and B. Zhang, "Risk euclidean distance-based model predictive path integral to safety-critical obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1906–1913, 2025.

[75] A. Saviolo, N. Picello, J. Mao, R. Verma, and G. Loianno, "Reactive collision avoidance for safe agile navigation," in *IEEE International Conference on Robotics and Automation*, pp. 16125–16132, 2025.

[76] O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 110–126, 2025.

[77] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[78] S. H. Arul, J. J. Park, and D. Manocha, "DS-MPEPC: Safe and deadlock-avoiding robot navigation in cluttered dynamic scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2256–2263, 2023.

[79] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*, pp. 1928–1935, 2008.

[80] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[81] Z. Liu, W. Na, C. Yao, C. Liu, and Q. Chen, "Relaxing the limitations of the optimal reciprocal collision avoidance algorithm for mobile robots in crowds," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5520–5527, 2024.

[82] D. Martinez-Baselga, E. Sebastián, E. Montijano, L. Riazuelo, C. Sagüés, and L. Montano, "AVOCADO: Adaptive optimal collision avoidance driven by opinion," *IEEE Transactions on Robotics*, vol. 41, pp. 2495–2511, 2025.

[83] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4584–4589, 2010.

[84] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 3475–3482, 2011.

[85] D. Bareiss and J. Van den Berg, "Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 3847–3853, 2013.

[86] L. Zhao, J. Zhao, Z. Liu, D. Yang, and H. Liu, "Solving the real-time motion planning problem for non-holonomic robots with collision avoidance in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10510–10517, 2022.

[87] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *IEEE International Conference on Robotics and Automation*, pp. 2347–2354, 2015.

[88] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

[89] M. Boldrer, M. Andreetto, S. Divan, L. Palopoli, and D. Fontanelli, "Socially-aware reactive obstacle avoidance strategy based on limit cycle," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3251–3258, 2020.

[90] L. Huber, J.-J. Slotine, and A. Billard, "Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3113–3132, 2022.

[91] Y. Wang, H. Wu, X. Xu, Y. Sun, and X. Zeng, "Enhanced adaptive artificial potential field for UAV navigation in dynamic 3D environments with lightweight spherical obstacle map," *IEEE Robotics and Automation Letters*, vol. 11, no. 1, pp. 682–689, 2026.

[92] 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.

[93] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *IEEE International Conference on Robotics and Automation*, pp. 8620–8626, 2019.

[94] Z. Jian, S. Zhang, L. Sun, W. Zhan, N. Zheng, and M. Tomizuka, "Long-term dynamic window approach for kinodynamic local planning in static and crowd environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3294–3301, 2023.

[95] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *IEEE International Conference on Robotics and Automation*, pp. 802–807, 1993.

[96] S. Yasuda, T. Kumagai, and H. Yoshida, "Safe and efficient dynamic window approach for differential mobile robots with stochastic dynamics using deterministic sampling," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2614–2621, 2023.

[97] Z. Zhang, Y. Xue, N. Figueroa, and K. Åkesson, "Gradient field-based dynamic window approach for collision avoidance in complex environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 19669–19674, 2025.

[98] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 6252–6259, 2018.

[99] J. de Heuvel, X. Zeng, W. Shi, T. Sethuraman, and M. Bennewitz, "Spatiotemporal attention enhances lidar-based robot navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4202–4209, 2024.

[100] X. Huang, H. Deng, W. Zhang, R. Song, and Y. Li, "Towards multi-modal perception-based navigation: A deep reinforcement learning method," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4986–4993, 2021.

[101] Y. Han, I. H. Zhan, W. Zhao, J. Pan, Z. Zhang, Y. Wang, and Y.-J. Liu, "Deep reinforcement learning for robot collision avoidance with self-state-attention and sensor fusion," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6886–6893, 2022.

[102] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*, pp. 3–19, Springer, 2011.

[103] M. L. Littman, "A tutorial on partially observable markov decision processes," *Journal of Mathematical Psychology*, vol. 53, no. 3, pp. 119–125, 2009.

[104] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[105] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1178–1185, 2019.

[106] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[107] X. Fan, M. Lu, B. Xu, and P. Lu, "Flying in highly dynamic environments with end-to-end learning approach," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3851–3858, 2025.

[108] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese, *et al.*, "Deep local trajectory replanning and control for robot navigation," in *IEEE International Conference on Robotics and Automation*, pp. 5815–5822, 2019.

[109] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 709–715, 2014.

[110] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[111] F. Ma and S. Karaman, "Sparse-to-Dense: Depth prediction from sparse depth samples and a single image," in *IEEE International Conference on Robotics and Automation*, pp. 4796–4803, 2018.

[112] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1343–1350, 2017.

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

[114] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[115] K. Zhu, B. Li, W. Zhe, and T. Zhang, "Collision avoidance among dense heterogeneous agents using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 57–64, 2023.

[116] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *International Conference on Robotics and Automation*, pp. 6015–6022, 2019.

[117] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5671–5677, 2020.

[118] Y. Yang, J. Jiang, J. Zhang, J. Huang, and M. Gao, "ST ^{2}: Spatial-temporal state transformer for crowd-aware autonomous navigation," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 912–919, 2023.

[119] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[120] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 10007–10013, 2020.

[121] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 3517–3524, 2021.

[122] H. Liu, W. Dong, S. Mao, C. Wang, and Y. Gao, "Sample-efficient learning-based dynamic environment navigation with transferring experience from optimization-based planner," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7055–7062, 2024.

[123] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[124] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.

[125] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5308–5317, 2016.

[126] X. H. Jing, X. Xiong, F. H. Li, T. Zhang, and L. Zeng, "A two-stage reinforcement learning approach for robot navigation in long-range indoor dense crowd environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5489–5496, 2024.

[127] P. Chen, Q. Liu, Y. Li, and S. Ma, "An environmental-complexity-based navigation method based on hierarchical deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 5119–5125, 2024.

[128] Y. Gao, J. Wu, X. Yang, and Z. Ji, "Efficient hierarchical reinforcement learning for mapless navigation with predictive neighbouring space scoring," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 5457–5472, 2024.

[129] Y. Du, J. Zhang, X. Cheng, and S. Cui, "Cooperative motion planning in divided environments via congestion-aware deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2295–2302, 2025.

[130] J. Gao, X. Pang, Q. Liu, and Y. Li, "Hierarchical reinforcement learning for safe mapless navigation with congestion estimation," in *IEEE International Conference on Robotics and Automation*, pp. 8849–8855, 2025.

[131] I. Okunevich, A. Lombard, T. Krajnik, Y. Ruichek, and Z. Yan, "Online context learning for socially compliant navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 5042–5049, 2025.

[132] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[133] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Autonomous Robots*, vol. 45, pp. 51–76, 2021.

[134] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application*, vol. 7, no. 1, pp. 279–301, 2020.

[135] M. Chang, J. Jang, D. Han, W. Choi, S. Kim, H. Park, and H. Choi, "OPPA: Online planner's parameter adaptation for enhanced mobile robot navigation," in *IEEE International Conference on Robotics and Automation*, pp. 3861–3867, 2025.

[136] J. J. Damanik, J.-W. Jung, C. A. Deresa, and H.-L. Choi, "LiCS: Navigation using learned-imitation on cluttered space," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 2000–2007, 2024.

[137] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6, VDE, 2012.

[138] H. Yang, C. Yao, C. Liu, and Q. Chen, "RMRL: Robot navigation in crowd environments with risk map-based deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7930–7937, 2023.

[139] T. Kathuria, K. Liu, J. Jang, X. J. Yang, and M. Ghaffari, "Learning implicit social navigation behavior using deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 5146–5153, 2025.

[140] L. Heuer, L. Palmieri, A. Rudenko, A. Mannucci, M. Magnusson, and K. O. Arras, "Proactive model predictive control with multi-modal human motion prediction in cluttered dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 229–236, 2023.

[141] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5116–5123, 2023.

[142] S. Samavi, A. Lem, F. Sato, S. Chen, Q. Gu, K. Yano, A. P. Schoellig, and F. Shkurti, "SICNav-Diffusion: Safe and interactive crowd navigation with diffusion trajectory predictions," *IEEE Robotics and Automation Letters*, vol. 10, no. 9, pp. 8738–8745, 2025.

[143] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[144] L. Qin, Z. Huang, C. Zhang, H. Guo, M. Ang, and D. Rus, "Deep imitation learning for autonomous navigation in dynamic pedestrian environments," in *IEEE International Conference on Robotics and Automation*, pp. 4108–4115, 2021.

[145] M. Lu, X. Fan, H. Chen, and P. Lu, "FAPP: Fast and adaptive perception and planning for uavs in dynamic cluttered environments," *IEEE Transactions on Robotics*, vol. 41, pp. 871–886, 2024.

[146] G. Chen, Z. Wang, W. Dong, and J. Alonso-Mora, "Particle-based instance-aware semantic occupancy mapping in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 1155–1171, 2025.

[147] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos, "PIE: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6262–6271, 2019.

[148] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 683–700, Springer, 2020.

[149] P. Kothari, S. Kreiss, and A. Alahi, "Human trajectory forecasting in crowds: A deep learning perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7386–7400, 2021.

[150] L. Knoedler, C. Salmi, H. Zhu, B. Brito, and J. Alonso-Mora, "Improving pedestrian prediction models with self-supervised continual learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4781–4788, 2022.

[151] H. Nishimura, B. Ivanovic, A. Gaidon, M. Pavone, and M. Schwager, "Risk-sensitive sequential action control with multi-modal human trajectory forecasting for safe crowd-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 11205–11212, 2020.

[152] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[153] G. Shafer and V. Vovk, "A tutorial on conformal prediction.," *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[154] K. J. Strawn, N. Ayanian, and L. Lindemann, "Conformal predictive safety filter for RL controllers in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7833–7840, 2023.

[155] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[156] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[157] R. Guldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6053–6060, 2020.

[158] B. Angulo, A. Panov, and K. Yakovlev, "Policy optimization to learn adaptive motion primitives in path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 824–831, 2023.

[159] J. Wu, Y. Wang, H. Asama, Q. An, and A. Yamashita, "Risk-sensitive mobile robot navigation in crowded environment via offline reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7456–7462, 2023.

[160] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[161] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 6057–6063, 2021.

[162] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[163] S. H. Semnani, A. H. de Ruiter, and H. H. Liu, "Force-based algorithm for motion planning of large agent," *IEEE Transactions on Cybernetics*, vol. 52, no. 1, pp. 654–665, 2020.

[164] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," *Advances in neural information processing systems*, vol. 32, 2019.

[165] K. Kang, P. Gradu, J. J. Choi, M. Janner, C. Tomlin, and S. Levine, "Lyapunov density models: Constraining distribution shift in learning-based control," in *International Conference on Machine Learning*, pp. 10708–10733, PMLR, 2022.

[166] K. Matsumoto, Y. Hyodo, and R. Kurazume, "Crowd-aware robot navigation with switching between learning-based and rule-based methods using normalizing flows," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4823–4830, 2024.

[167] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky, "Graph normalizing flows," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[168] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2023.

[169] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[170] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *IEEE International Conference on Robotics and Automation*, pp. 1111–1117, 2018.

[171] P. Xu and I. Karamouzas, "Human-inspired multi-agent navigation using knowledge distillation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8105–8112, 2021.

[172] J. Qin, Q. Liu, Q. Ma, Z. Wu, and J. Qin, "A non-homogeneity mapless navigation based on hierarchical safe reinforcement learning in dynamic complex environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 10237–10244, 2024.

[173] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 1140–1146, 2021.

[174] N. A. Sen, D. Kulić, and P. Carreno-Medrano, "Domain randomization for learning to navigate in human environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1625–1632, 2025.

[175] J. Wu, Y. Wang, T. Chen, J. Jiang, Y. Wang, Q. An, and A. Yamashita, "Diversity-aware crowd model for robust robot navigation in human populated environment," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6376–6383, 2025.

[176] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, 2021.

[177] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "NavRL: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3668–3675, 2025.

[178] A. Srisuchinnawong, J. Bæch, M. P. Hyzy, T. Kounalakis, E. Boukas, and P. Manoonpong, "Unsupervised multiple proactive behavior learning of mobile robots for smooth and safe navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 11796–11803, 2024.

[179] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[180] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[181] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[182] C. Yan, J. Qin, Q. Liu, Q. Ma, and Y. Kang, "Mapless navigation with safety-enhanced imitation learning," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 7, pp. 7073–7081, 2022.

[183] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium*, pp. 203–216, Springer, 2013.

[184] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13344–13362, 2023.