Safe Gap-based Planning in Dynamic Settings

Max Asselmeier, Abdel Zaro, Dhruv Ahuja, Ye Zhao, and Patricio A. Vela

Abstract This chapter extends the family of perception-informed gap-based local planners to dynamic environments. Existing perception-informed local planners that operate in dynamic environments often rely on emergent or empirical robustness for collision avoidance as opposed to performing formal analysis of dynamic obstacles. This proposed planner, *dynamic gap*, explicitly addresses dynamic obstacles through several steps in the planning pipeline. First, polar regions of free space known as gaps are tracked and their dynamics are estimated in order to understand how the local environment evolves over time. Then, at planning time, gaps are propagated into the future through novel gap propagation algorithms to understand what regions are feasible for passage. Lastly, pursuit guidance theory is leveraged to generate local trajectories that are provably collision-free under ideal conditions. Additionally, obstacle-centric *ungap* processing is performed in situations where no gaps exist to robustify the overall planning framework. A set of gap-based planners are benchmarked against a series of classical and learned motion planners in dynamic environments, and dynamic gap is shown to outperform all other baselines in all environments. Furthermore, dynamic gap is deployed on a TurtleBot2 platform in several real-world experiments to validate collision avoidance behaviors.

M. Asselmeier · A. Zaro · Y. Zhao · P. A. Vela
Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308. e-mail: mass@gatech.edu, azaro3@gatech.edu, yzhao301@gatech.edu, pvela@gatech.edu

D. Ahuja
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308. e-mail: dahuja8@gatech.edu

This work supported in part by NSF Award #2235944. The work of Max Asselmeier is supported by the NSF Graduate Research Fellowship under Grant No. DGE-2039655. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

Acronyms

A3C	Asynchronous Advantage Actor-Critic
CADRL	Collision Avoidance with Deep Reinforcement Learning
CoHAN	Co-operative Human-aware Navigation
DRL-VO	Deep Reinforcement Learning - Velocity Obstacles
DGap	Dynamic Gap
DWA	Dynamic Window Approach
FOV	Field of View
GBP	Gap-based Planner
HA	Hungarian Algorithm
ICP	Iterative Closest Point
IQA	Iterative Quaternion Averaging
KF	Kalman Filter
LfLH	Learning from Learned Hallucination
LSTM	Long Short-Term Memory
MPC	Model Predictive Control
ORCA	Optimal Reciprocal Collision Avoidance
PN	Parallel Navigation
PGap	Potential Gap
PRM	Probabilistic Roadmap
PPO	Proximal Policy Optimization
PP	Pure Pursuit
RRT	Rapidly-exploring Random Trees
RVO	Reciprocal Velocity Obstacles
RA	Rectangular Assignment
RLCA	Reinforcement Learning for Collision Avoidance
ROS	Robot Operating System
TEB	Timed Elastic Bands
VO	Velocity Obstacles

1 Introduction

The family of gap-based planners (GBP) offers a free space-based approach to real-time navigation in previously unseen environments. GBPs seek to leverage the environmental affordances [1] available to the robot and restrict all planning efforts to these free regions, or *gaps*, in the environment. In the literature, this family of planners has shown great promise due to their ability to provide safety guarantees [2] under ideal assumptions including full field-of-view (FOV) sensing, first-order holonomic dynamics, and point-mass geometry. Beyond these ideal conditions, GBPs have still provided impressive navigation performance given restricted FOVs [2], noncircular robot geometries [3], and nonholonomic dynamics [4].

Despite this success, GBPs have only been extended to handling dynamic obstacle avoidance very recently [5, 6], a challenge that accurately reflects the previously unseen and constantly changing environments found in real life. Typically, local motion planners are designed with static environments in mind. Then, when the time comes to deploy these planners in dynamic real-world settings, the planners are often run with the hopes of high enough planning rates enabling sufficient reactive collision avoidance. However, relying on these emergent or empirical behaviors limits planner performance. It is more meaningful to design a planner with dynamic settings in mind, but this often requires each and every step of the planning pipeline to be revisited to account for these relaxed assumptions. In static settings, the currently available free space will remain as free space for all time. Naturally, the same applies for obstacle space. In dynamic settings, free space can turn into obstacle space, and vice-versa. If a motion planner is to fully exploit its environmental affordances, then both of these possibilities must explicitly be accounted for.

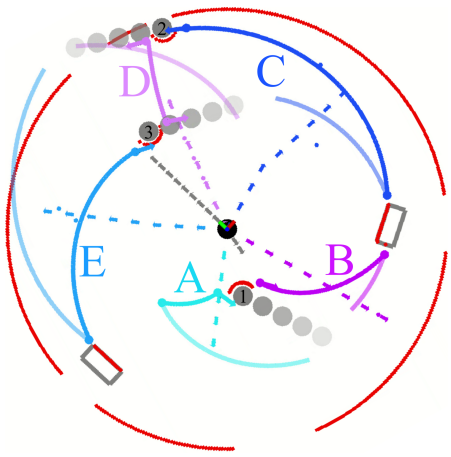


Fig. 1 Visualization of the dynamic gap planner. The black disk is the ego-robot and gray disks are dynamic agents. Transparent gray disks are future agent positions. Bold colored arcs labeled A-E are the instantaneous set of gaps. Transparent arcs are the predicted gaps obtained by propagating the gap dynamics models, shown as arrows, forward. Dashed lines are the candidate trajectories and the gap goals are colored points within the gap spans. The pink gap tube labeled D, which is comprised of all present and future pink gap arcs, is predicted to close as agents 2 and 3 cross each other. However, this gap will reopen, represented by the transparent pink arc. The gap tube trajectory plans up to the gap closure, idles in place while the gap is closed, and then continues through the reopened gap. Ungap trajectories, represented as gray dashed lines, are synthesized in the receding ungaps formed by agents 1 and 2.

To date, GBPs designed for dynamic settings [5, 6] have not been perception-informed, instead opting to rely on ground truth state information regarding other agents in the local environment. The intention of this proposed work is to design a perception-informed GBP for dynamic settings in order to explicitly account for both sensor data artifacts and potentially evolving environments. In [7], the authors introduced an initial version of the *dynamic gap* planner as a proof of concept. In this book chapter, the dynamic gap planner is documented in greater detail to further elucidate each intermediate step in the planning pipeline. The planner itself is extended by relaxing two limiting assumptions: (1) the "isolated gap" assumption in which gaps were previously not allowed to interact during gap propagation, and (2) the "unsafe ungap" assumption in which the polar spaces between gaps caused by

obstacles were deemed entirely uninhabitable and unsafe to plan within. In summary, the core contributions of the proposed work are as follows.

1. Proposing an alternative tracking paradigm which revolves around the tracking and prediction of free space
2. Adapting geometric and kinematic rules from guidance laws to the realm of GBPs to aid in dynamic gap propagation and feasibility analysis
3. Performing complex gap propagation analysis to enable planning through interrupted and switched gaps
4. Providing a proof of collision-free dynamic gap passage under ideal conditions along with supplementary safety modules for non-ideal conditions
5. Benchmarking against state-of-the-art local planners in the Arena-Rosnav simulation environment
6. Validating planner performance onboard a TurtleBot2 platform in dynamic real-world environments

A visual snapshot of active components during local planning for dynamic gap is portrayed in Figure 1. This planner is open-sourced[1] in Arena-Rosnav [8, 9, 10, 11] and available to test. The information flow for the planner can be seen in Figure 4.3. In this chapter, the task of navigation in dynamic environments is defined (Section 2) and common planning pipelines (2.1) and approaches (2.2) are surveyed and discussed. Social navigation considerations that become important in dynamic real-world settings are explained in Section 2.3. The paradigm of Planning in the Perception Space (PiPS) that GBPs fall under is discussed in Section 3. Preliminary concepts for GBPs in dynamic environments are defined (Section 4) before the proposed local planner is detailed (Section 5). Then, experimental results in Section 6 validate theoretical claims of collision avoidance (6.1), illuminate planning performance in more contrived setups (6.2), and contextualize dynamic gap planning performance among state-of-the-art planners (6.3). Analysis of the computational workload of the dynamic gap planner (6.4) demonstrates the efficiency of the PiPS framework, social compliance results (6.5) provide insight into the behavior of dynamic gap around pedestrians in simulation, and, lastly, hardware experiments (6.6) validate this proposed planning framework in the real world.

2 Navigation in Dynamic Environments

Mobile robot planning concerns the generation of a sequence of motion commands for a mobile robot platform to maneuver from an arbitrary start position to an arbitrary goal position. If this environment is entirely known and unchanging, then this task is purely a motion planning problem. If the environment is previously unknown, partially observable, and potentially even changing over time, then this task becomes a navigation problem. For this navigation problem, the robot must

[1] https://github.com/ivaROS/DynamicGap

also possess some means to reveal the environment structure online through visual sensing. This mobile robot may also be subject to geometric constraints due to its size and shape, kinematics constraints due to its composition and linkages, or dynamics constraints due to its actuation capabilities. In the context of this chapter, the robot must reach its goal position (1) under a predefined time limit and (2) without colliding into any static or dynamic artifacts in the environment in order for the task to be deemed a success.

2.1 Planning Considerations

When navigating in a dynamic environment, a handful of additional considerations must be made when compared to static environments: *data clustering*, *data association*, and *data estimation*. First, a navigation framework should have some way to *cluster* or abstract raw sensor data into meaningful primitives that enable local planners to efficiently avoid environment collisions. These primitives can take many forms, including point clusters [12, 13], disks [14], polygonal geometries [15, 16], or semantic labels [17]. Second, a navigation framework must find and *associate* pairings of these primitives between consecutive time steps in order to retain a history or memory of primitives. Depending on the format of the primitive, registration algorithms such as Iterative Closest Point (ICP) [18] or Iterative Quaternion Averaging (IQA) [19] can provide pairwise assignments. Other approaches treat this step as a Rectangular Assignment (RA) problem and solve via the Hungarian Algorithm (HA) [20]. Lastly, a navigation framework must *estimate* the unobservable portion of the state of the dynamic obstacles. This involves updating a set of prediction models that could be variants of Kalman filters (KF) [12, 21, 22], Gaussian processes [23], factor graphs [24], or neural networks [17].

Once all of these steps have been taken, prediction models can be factored into planning decisions through capturing them in cost functions or constraints in Model Predictive Control (MPC) formulations [25, 15], or embedding them in occupancy maps that can be searched over for paths [26, 21]. The methods cited here assign prediction models to individual obstacles and represent these prediction models with respect to a fixed frame. Performing obstacle tracking in the egocentric frame allows the local planner to leverage the benefits of operating in the perception space which will be discussed in Section 3.

2.2 Planning Approaches

Within the mobile robotics research community, navigation is a well-studied problem with a myriad of approaches. Many classical algorithms were originally designed for static environments and later saw extensions to explicitly account for dynamic environments. Graph search-based methods such as A* [27] can discretize the con-

tinuous configuration space of a robot and search over it for a collision-free path to the goal. Later, the methods D* [28] and D* Lite [29] were designed to handle the dynamic environment case. Sampling-based methods such as Probabilistic Roadmaps (PRM) [30], Rapidly-Exploring Random Trees (RRT) [31], and their asymptotically optimal extensions PRM* and RRT* [32] can be viewed as extensions of graph-based methods that opt to randomly sample the continuous configuration space instead of discretization. This family of planners has also seen extensions to dynamic environments [33, 34]. Some planning algorithms such as the Dynamic Window Approach (DWA) [35] choose to sample in the control space as opposed to the configuration space and then roll out these sampled control inputs. Similarly, dynamic versions of DWA have been developed [36]. More recently, trajectory optimization has proven to be a potent form of motion planning that can account for kinodynamical constraints while providing optimal trajectories. Methods including Timed Elastic Bands (TEB) [15] formulate local navigation tasks as optimal control problems which are also general enough to handle dynamic environments [16]. Some approaches such as velocity obstacles (VOs) [37, 38], reciprocal velocity obstacles (RVOs) [39], and optimal reciprocal collision avoidance (ORCA) [40] were initially designed with dynamic obstacles in mind. Methods from control theory have also been leveraged for effective collision avoidance strategies including Hamilton-Jacobi-Bellman reachability-based approaches [41, 42] and receding horizon-based methods [43, 44, 45, 46]. Lower-level safety filters such as control barrier functions [47, 48, 49] and backup controllers [50] can also modify control inputs to ensure safety during online deployment.

As of late, data-driven approaches to local navigation and collision avoidance have also received tremendous amounts of attention. Most of these learned planners are based on deep reinforcement learning [51, 52, 53, 54, 55, 56] due to the availability of physics simulations and self-supervised trial-and-error-based data collection methods. Such planners have seen significant use due to their apparent generalizability and scalability. However, these methods have yet to see widespread integration onto hardware platforms due to their lack of safety and interpretability. Extensive comparisons of classical and learned methods are still difficult to find [57, 58], and one of the motivations of this chapter is to provide a detailed benchmarking comparison of several state-of-the-art planners, both learned and classical. One of the particularly exciting facets of the proliferation of robotics research is the rise in publically available competitions and challenges to stimulate progress. Both the Benchmark Autonomous Robot Navigation challenge [59] and DynaBARN [60] challenges have been excellent places to locate and evaluate state-of-the-art local navigation planners.

2.3 Social Navigation

Under the context of navigation, planning in dynamic environments commonly means planning in the presence of humans. Social navigation refers to a robot's abil-

ity to navigate in a manner that goes beyond geometric efficiency, such as minimizing path length, to also account for social considerations, including human comfort, and norms. A socially compliant robot must be capable of recognizing people and prioritizing their safety accordingly. Beyond collision avoidance, the planner's behavior should aim to minimize discomfort, reduce confusion, and maintain predictability in its movements. Socially-aware navigation also involves the ability to convey intent, either through movement patterns or explicit signals, so that nearby humans can understand and anticipate the robot's actions. In situations where human and robot objectives may conflict, a socially compliant robot may even compromise its own efficiency to resolve the situation in a way that aligns with human expectations [61].

Failure to navigate in a socially compliant manner can cause discomfort and even increase the risk of collisions, especially in situations where human behavior is unpredictable. Social considerations are particularly important as robots become increasingly integrated into everyday social environments, supporting applications such as food delivery, healthcare settings, and office spaces. However, human behavior varies widely across contexts, individuals may be distracted by their phones, uncooperative towards robots, or hurried by their own tasks, introducing patterns that the robot may not be familiar with. This variability makes it challenging to develop navigation approaches that generalize effectively across real-world environments.

Numerous approaches have been proposed to address this. Classical geometric methods with hand-engineered cost functions and heuristics dominate industrial deployment due to their reliability and ease of software integration. Data-driven approaches, on the other hand, are able to address many of the shortcomings of classical based approaches, such as capturing nuanced human behaviors, predicting trajectories and incorporating social norms; however, they lack the same safety guarantees and explainability and have higher integration overhead [62].

In order to assess a planner's performance in a social setting, various metrics such as the velocity or time spent in personal space can be used, where personal space is often defined as some radius around humans [63].

3 Planning in the Perception Space

Most motion planners [31, 15, 64] opt to plan using Cartesian world-frame environment representations such as occupancy maps or voxel grids. These approaches contrast the perception space approach to planning which involves keeping sensory input in its raw egocentric form to take advantage of the computational benefits that come with foregoing intensive data processing. Applying the PiPS framework means that all local planning steps downstream must then be cast as egocentric decision-making processes.

Prior work from the authors has developed a hierarchical perception space navigation stack fit with efficient collision checking [65], egocentric environment representations [66], and local path planning [67, 2].

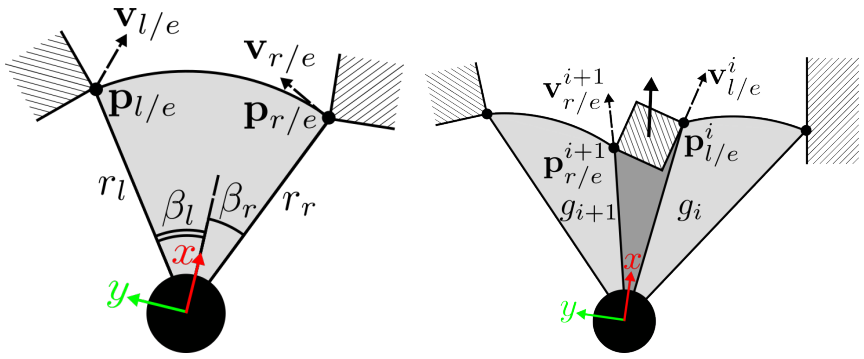
GBPs [68, 69, 70, 71, 3, 72, 73, 74, 75, 76, 6, 4, 5, 77] are a form of perception space-based planners that detect regions of collision-free space defined by leading or trailing edges of obstacles. This can be viewed as an alternative way of discretizing the environment, here in the egocentric polar space as opposed to discretizing in the Cartesian space for common methods such as occupancy maps [78, 79, 80]. Some attention has been given to gaps in dynamic environments [5, 6], but these methods do not develop their theory through a perception-informed approach, instead opting to use ground truth agent pose information. In the proposed work, explicit attention is paid towards how the dynamics of gaps must be ascertained from scan data in order to understand how the local gaps evolve over time.

4 Preliminaries

First, the primitive environmental affordances that the proposed planner will exploit for planning are defined and visualized.

4.1 Gap

Definition (Gap): A polar region of free space in $\mathbb{R}^2$ characterized by the area swept out between $\mathbf{p}_{l/e}$ and $\mathbf{p}_{r/e}$ where the subscripts l/e and r/e represent relative measurements between the left and right gap points, respectively, and the ego-robot. Gap points are also assigned velocities $\mathbf{v}_{l/e}$ and $\mathbf{v}_{r/e}$. Together, these form the left and right gap point states $\mathbf{X}_{l/e} = \begin{bmatrix}\mathbf{p}_{l/e} & \mathbf{v}_{l/e}\end{bmatrix}^T$ and $\mathbf{X}_{r/e} = \begin{bmatrix}\mathbf{p}_{r/e} & \mathbf{v}_{r/e}\end{bmatrix}^T$.



(a) Each gap G is characterized by left and right gap point states comprised of positions $\mathbf{p}_{l/e}$ and $\mathbf{p}_{r/e}$ and velocities $\mathbf{v}_{l/e}$ and $\mathbf{v}_{r/e}$.

(b) Each ungap U is built from the left gap point of a gap G_i, denoted as $\mathbf{p}^i_{l/e}$, and the right gap point of the next gap G_{i+1}, denoted as $\mathbf{p}^{i+1}_{r/e}$.

Fig. 2 Diagrams of (a) an example gap and (b) an example ungap.

4.2 Ungap

Definition (Ungap): A polar region of obstacle space in $\mathbb{R}^2$ characterized by the area swept out by adjacent gap points $\mathbf{p}_{r/e}^{i+1}$ and $\mathbf{p}_{l/e}^{i}$.
Note that the two points that define an ungap come from two different adjacent gaps. The set of ungaps considered during planning is restricted to the ungaps that are (a) dynamic and (b) receding. The criteria used to determine these classifications are further detailed in Section 5.1.3.

4.3 Gap Tube

Definition (Gap Tube): A sequence of gaps and gap lifespans that characterize how a gap evolves from time $t = 0$ to the end of the local planning horizon $t = T$.
In this chapter, a gap tube will be expressed as

$$\mathcal{T} = \{(G_0, T_0), (G_1, T_1), \ldots, (G_N - 1, T_N - 1)\}$$

where N is the number of unique gap models needed to characterize the gap tube during the local planning horizon and $T_0 + T_1 + \cdots + T_{N-1} = T$. The process through which these gap tubes are constructed during planning is detailed in Section 5.5.1.

5 Dynamic Gap Local Planning Module

5.1 Gap Generation

The step of gap generation is an egocentric form of data clustering and environment abstraction that allows GBPs to condense a high-dimensional perceptual input such as a laser scan into a small set of free space regions that can be passed through during planning.

5.1.1 Gap Detection

Gap detection involves the parsing of this scan $\mathcal{L}$ to obtain a set of detected gaps $\mathcal{G}_t^{\text{raw}}$ that describe the free space of the local environment at the current time t. Gaps are classified as radial if they are formed from a significant, instantaneous change in range across consecutive scan points or as swept if they comprise a large interval of scan points with maximum detectable ranges.

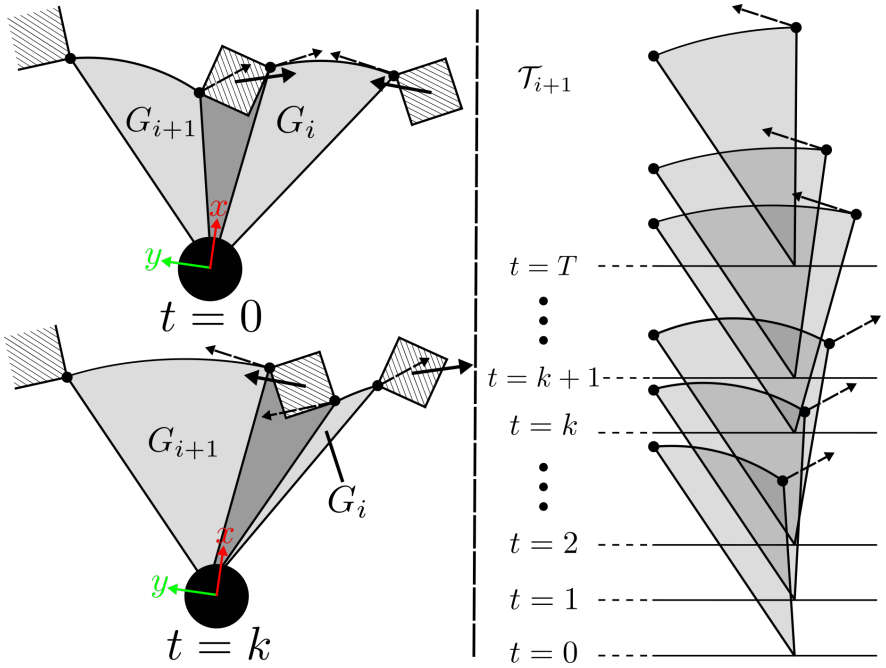


Fig. 3 Diagram of an example gap tube. At $t = 0$, the gap tube $\mathcal{T}_{i+1}$ begins with the gap G_{i+1}. However, at $t = k$, due to the two obstacles overlapping one another, the dynamics of the right point of gap G_{i+1} change. In order to reflect this change in the dynamics, the gap tube $\mathcal{T}_{i+1}$ visualized on the right side of the figure captures this second gap with altered dynamics.

5.1.2 Gap Simplification

Once this set of gaps has been extracted from the laser scan, an additional pass through $\mathcal{G}^{\text{raw}}$ is performed to remove any redundant gaps and potentially merge adjacent gaps, yielding a set of simplified gaps $\mathcal{G}^{\text{simp}}$ that represent the most s alient portions of free space in the local environment.

5.1.3 Ungap Detection

A set of ungaps $\mathcal{U}^{\text{simp}}$ can also be extracted from $\mathcal{G}^{\text{simp}}$ to use during planning. To do so, the set of simplified gap points

$$P^{\text{simp}} = \{\mathbf{p}^0_{r/e}, \mathbf{p}^0_{l/e}, ..., \mathbf{p}^N_{r/e}, \mathbf{p}^N_{l/e}\}$$

is sorted in counterclockwise order by bearing. Then a series of checks are performed on adjacent pairs of points to determine if the pair $(\mathbf{p}_i, \mathbf{p}_j)$ forms an ungap. First, a velocity check is made to determine if the pair of points are dynamic and moving in roughly the same direction:

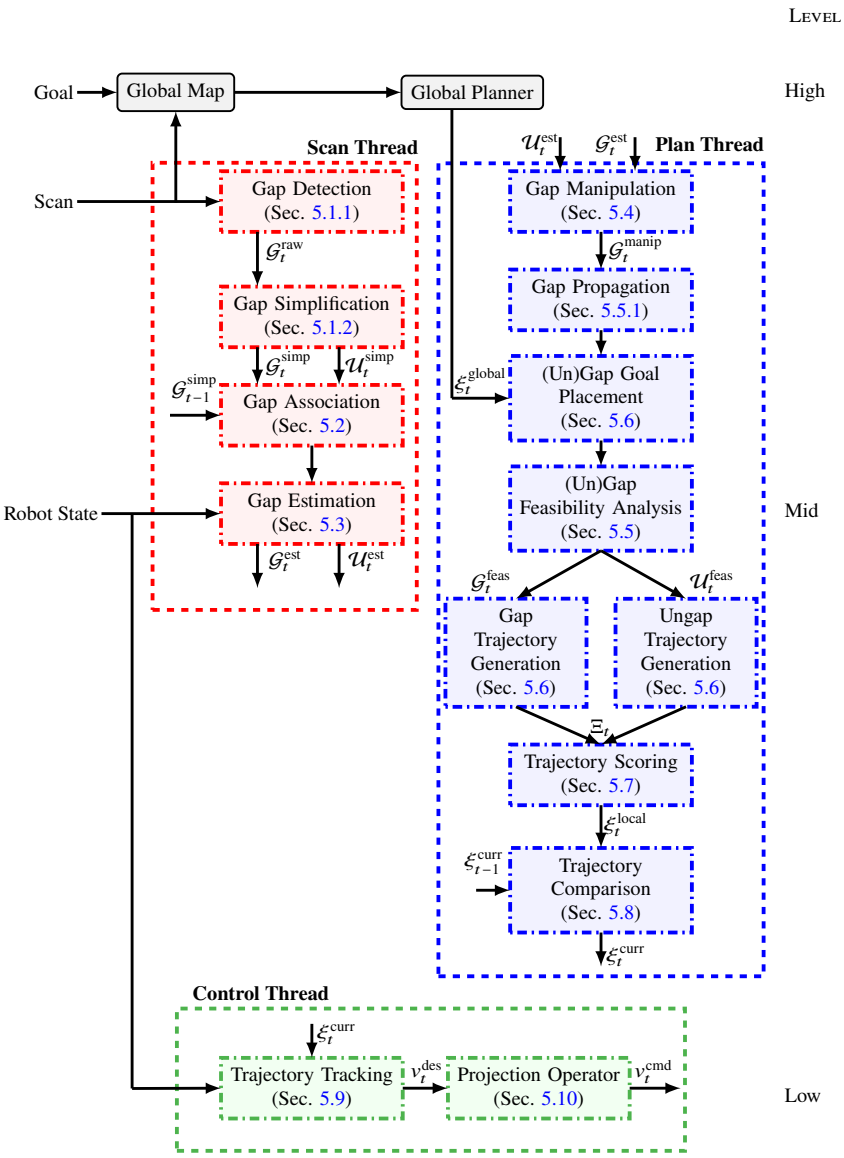
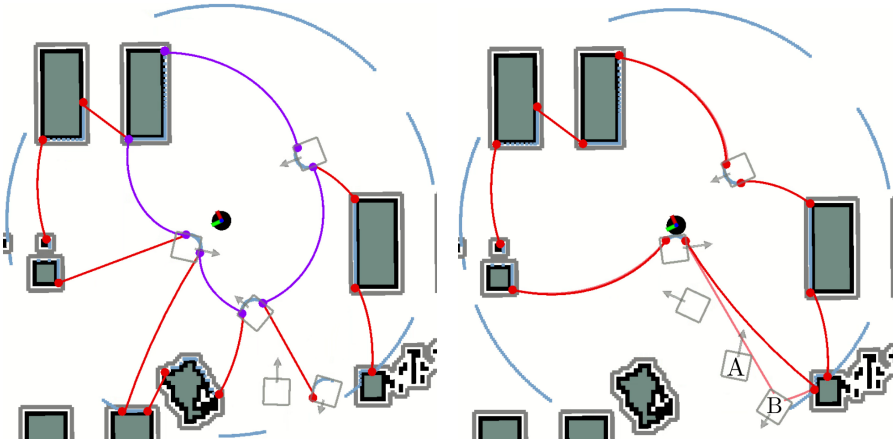


Fig. 4 Workflow for navigation framework. Within this framework, there are three active threads: scan, plan, and control. The scan thread (red) is run at the rate of the incoming laser scan, which is 25 Hz in simulation and on hardware. This thread acts on an incoming laser scan $\mathcal{L}$ and outputs a set of estimated gaps $\mathcal{G}_t^{\text{est}}$ and ungaps $\mathcal{U}_t^{\text{est}}$. The plan thread (blue) then synthesizes a set of candidate local trajectories Ξ_t through these estimated gaps and ungaps. Finally, the control thread acts to track the selected local trajectory ξ_t^{curr}. Both the plan and control thread are bound to run together at the prescribed planning rate which is 5 Hz in simulation and on hardware.



(a) Example set of raw and simplified gaps obtained from the gap detection policy in Sections 5.1.1 and 5.1.2. Blue points are the current laser scan, red arcs and points are the set of immediately detected raw gaps, and purple arcs and points are the simplified gaps.

(b) Example set of raw gaps associated across timesteps as detailed in Section 5.2. As agent A passes in front of agent B at time t, the gaps attached to agent B are no longer visible and subsequently lost during gap association.

Fig. 5 Visualizations of (a) gap detection and simplification and (b) gap association.

$$\|\mathbf{v}_i\|_2 \geq v_{\min} \wedge \|\mathbf{v}_j\|_2 \geq v_{\min} \tag{1}$$

and

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle > 0, \tag{2}$$

where $v_{\min}$ is a minimum speed threshold to consider an obstacle as dynamic. If the two points meet the criteria of Equations 1 and 2, then the points are attached to an ungap. Furthermore, if this ungap is determined to be receding, meaning that

$$\langle \mathbf{p}_i, \mathbf{v}_i \rangle > 0 \wedge \langle \mathbf{p}_j, \mathbf{v}_j \rangle > 0, \tag{3}$$

then this ungap is passed on to planning.

5.2 Gap Association

The set of simplified gaps $\mathcal{G}^{\text{simp}}$ captures the immediate free space, but in dynamic settings it is crucial to understand how this free space will evolve across the local planning horizon. Therefore, additional steps are taken to track gap points over time.

Association is performed on P_{simp} and represented as a RA problem where the cost is equivalent to the distance between points across consecutive steps, P_{t-1}^{simp}

and P_t^{simp}. This assignment problem is then solved with the HA [20], producing a minimum distance mapping between P_{t-1}^{simp} and P_t^{simp}. If the distance between two associated points exceeds a threshold $\tau_{\text{assoc}} \in \mathbb{R}^+$, then that point-to-point association is discarded and a new prediction model is instantiated for the respective point at time t.

5.3 Gap Estimation

The point-to-point associations provide an insight into how the set of gap points changes over time. This evolution is characterized by a second-order dynamics model with respect to the rotating ego-robot frame.

Given the desire to perform gap estimation in the perception space, the prediction models must represent the gap points with respect to the local robot frame which is constantly changing. Therefore, the constant velocity dynamics model taken with respect to an inertial frame is augmented to allow for translations and rotations of the robot. The state vector is defined as

$$\mathbf{X}_s = \begin{bmatrix} \mathbf{p}_{s/e} \\ \mathbf{v}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_s - \mathbf{p}_e \\ \mathbf{v}_s - \mathbf{v}_e \end{bmatrix}, \tag{4}$$

where $\mathbf{p}_{s/e} \in \mathbb{R}^2$ and $\mathbf{v}_{s/e} \in \mathbb{R}^2$ represent the position and velocity of the gap side s (left or right) relative to the ego-robot e, respectively. The system dynamics are then

$$\dot{\mathbf{X}}_s = \begin{bmatrix} \dot{\mathbf{p}}_{s/e} \\ \dot{\mathbf{v}}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{s/e} - \omega_e \times \mathbf{p}_{s/e} \\ \mathbf{a}_{s/e} - \omega_e \times \mathbf{v}_{s/e} \end{bmatrix}, \tag{5}$$

where ω_e represents the angular velocity of the ego-robot and $\mathbf{a}_{s/e}$ represents the linear acceleration of the gap side relative to the ego-robot. Gap points are assumed to travel at a constant velocity, which then simplifies Equation 5 to

$$\dot{\mathbf{X}}_s = \begin{bmatrix} \dot{\mathbf{p}}_{s/e} \\ \dot{\mathbf{v}}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{s/e} - \omega_e \times \mathbf{p}_{s/e} \\ -\mathbf{a}_e - \omega_e \times \mathbf{v}_{s/e} \end{bmatrix}. \tag{6}$$

This model is estimated with an extended KF given the nonlinear cross-product in Equation 6.

5.4 Gap Manipulation

Before the current set of gaps is passed on to trajectory generation, two pre-processing steps are taken in order to improve gap visibility and safety.

5.4.1 Radial Gap Conversion

After the gap simplification policy defined in Section 5.1.2, it is still possible to have radial gaps in the current set of gaps. Radial gaps can cause safety risks due to their low visibility beyond the gap arc. A real-world example can be a hallway corner where it is difficult to determine if there is an upcoming obstacle beyond the corner. To mitigate this source of danger, remaining radial gaps are converted into swept gaps by pivoting the gap representation about the closer of the two gap points. Only gap points that are not attached to ungaps are converted so as to not neglect the dynamics of the environment, and the prediction models of all gap points that are altered through this radial gap conversion step are modified to reflect the new gap state.

5.4.2 Gap Point Inflation

Any real world robot that this planner is to be deployed on will have some finite size. Therefore, gap representations must be inflated inwards to compensate for the inscribed radius of the ego-robot. A diagram for how this inflation is performed is shown in Figure 6. A user-defined inflation ratio $\tau_{\text{infl}} \in [1, \infty)$ can be adjusted to modulate how conservative this step should be when inflating the gaps inward. First, the inflated inscribed robot radius is calculated as

$$r_{\text{infl}} = \tau_{\text{infl}} \cdot r_{\text{inscr}}. \tag{7}$$

Then, the angle α that is formed between the line from $\mathbf{p}_e$ to the gap side point $\mathbf{p}_s$ and the line originating from $\mathbf{p}_e$ that is tangent to a circle of radius r_{infl} with its center at $\mathbf{p}_s$ is calculated as

$$\alpha_s = \arcsin\left(\frac{r_{\text{infl}}}{\|\mathbf{p}_s\|}\right). \tag{8}$$

The remaining angle β in the triangle formed by $\mathbf{p}_e$, $\mathbf{p}_s$, and the inflated gap side point $\mathbf{p}_s^{\text{infl}}$ can be calculated as

$$\beta_s = \frac{\pi}{2} - \alpha_s. \tag{9}$$

Finally, the distance h_{infl} from the original gap side point $\mathbf{p}_s$ and the inflated gap side point $\mathbf{p}_s^{\text{infl}}$ can be calculated as

$$h_s^{\text{infl}} = \frac{r_{\text{infl}}}{\sin(\beta_s)}. \tag{10}$$

This manipulation step generates a set of manipulated gaps $\mathcal{G}^{\text{manip}}$ which can be evaluated for feasibility.

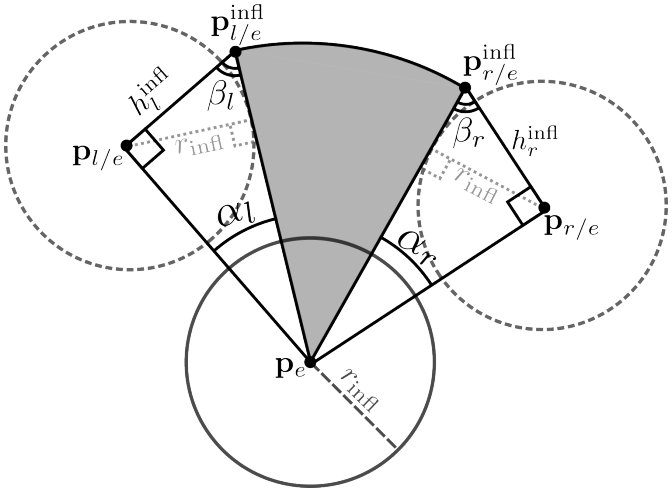


Fig. 6 Diagram representing an example inflated gap. Here, the ego-robot point $\mathbf{p}_e$, the left gap point $\mathbf{p}_{l/e}$, and right gap point $\mathbf{p}_{r/e}$ are displayed along with their inflated counterpart $\mathbf{p}_{l/e}^{\text{infl}}, \mathbf{p}_{r/e}^{\text{infl}}$. The inflation policy is identical for the both gap sides, with solely the rotation direction for the inflation point being the difference.

5.5 Gap Feasibility Analysis

In static settings, gap feasibility analysis is zeroth-order calculation. Determining whether or not a gap can be planned through is purely a geometric consideration: can the ego-robot fit within the gap? However, the transition to dynamic environments makes gap feasibility analysis a higher-order calculation.

In static settings, the currently available free space will remain as free space (and the same applies for obstacle space) for all time. In dynamic settings, free space can turn into obstacle space, and vice-versa. A gap currently exists, but might not in the future. A gap may not exist now, but will be created in the future. To account for these scenarios, the presently existent gaps are propagated forward in time to understand how they evolve, potentially changing shape or dynamics. By pruning away infeasible gaps, this step conserves the energy of the robot and avoids potentially dangerous gaps through which it would be difficult, if not impossible, for the robot to pass.

5.5.1 Gap Propagation

To understand the behavior of gaps over the local planning horizon, gap models (Eq. 6) are integrated forward under the constant velocity assumption. In order to remove the ego-robot motion from the gap state and solely analyze the motion of the gap itself, the *gap-only* dynamics are recovered from the prediction models by adding the ego-robot's velocity $\mathbf{v}_e$ to the relative velocity estimate $\mathbf{v}_{s/e}$ to obtain the gap-only velocity $\mathbf{v}_s$. This gap-only velocity is then used during propagation. The general workflow is detailed below and given in Algorithms 1 and 2.

Gap Point Propagation: First, the set of points from the manipulated gaps P^{manip} is collected and propagated forward (Algorithm 1, Line 8). The propagated points

are sorted in counterclockwise order by bearing (Line 9), and these sorted points are passed on to Algorithm 2 to extract a new set of gaps at that future timestep. The extracted gaps from consecutive timesteps $k-1$ and k are associated (Line 11) through a Rectangular Assignment problem that is slightly modified from the one in Section 5.2 in order to assign the propagated gaps from t_{k-1} to the gaps at t_k. Lastly, these newly associated gaps are merged into a temporal *gap tube* that captures how a gap at time $t = 0$ evolves in the local environment to time $t = T$ (Line 12). This set of gap tubes $\mathcal{T}^{\text{prop}}$ is evaluated for feasibility as detailed in Section 5.5.2.

Algorithm 1 Gap Point Propagation

1: **Given:** Manipulated gap points P^{manip}
2: **Return:** Set of propagated gap tubes $\mathcal{T}^{\text{prop}}$
3: */* Initialization */*
4: $\mathcal{T}^{\text{prop}} = \{\}$
5: $\mathcal{G}^{\text{prop}}_{k-1} = \mathcal{G}^{\text{manip}}$
6: */* Propagation */*
7: **for** each timestep t_k in planning horizon T **do**
8: Propagate P^{manip} to time t_k
9: Sort P^{manip} in counterclockwise order by bearing
10: Extract propagated gaps $\mathcal{G}^{\text{prop}}_k$ from sorted P^{manip} ▷ Algorithm 2
11: Associate propagated gaps $\mathcal{G}^{\text{prop}}_k$ with $\mathcal{G}^{\text{prop}}_{k-1}$
12: Merge associated propagated gaps $\mathcal{G}^{\text{prop}}_k$ into $\mathcal{T}^{\text{prop}}$
13: $\mathcal{G}^{\text{prop}}_{k-1} = \mathcal{G}^{\text{prop}}_k$
14: **end for**

Propagated Gap Extraction: Algorithm 2 shows how a set of gaps at future time t_k are extracted from a set of propagated gap points. This algorithm iterates through the ordered gap points (Lines 9 – 20) in search of consecutive points for which $\mathbf{p}_i^{\text{manip}}$ represents a right gap point and $\mathbf{p}_j^{\text{manip}}$ represents a left gap point. One example of this would be the pair of points $\mathbf{p}_0^0$ and $\mathbf{p}_1^0$ in Figure 7a. If the opposite occurs, and $\mathbf{p}_i^{\text{manip}}$ represents a left gap point while $\mathbf{p}_j^{\text{manip}}$ represents a right gap point, this represents a gap that exists in the environment but is currently unavailable due to obstacles passing across one another. An example of an unavailable gap is the pair of points $\mathbf{p}_0^{k-2}$ and $\mathbf{p}_1^{k-2}$ in Figure 7a. This unavailable gap is formed when agents 1 and 2 that initially form gap A pass across each other at the timestep t_{k-2}. A new gap will appear at a future timestep, so it is important to store this information for gap propagation. Ungap IDs are leveraged in this step to ensure that neighboring gap points that form an ungap are not mistaken for an unavailable gap.

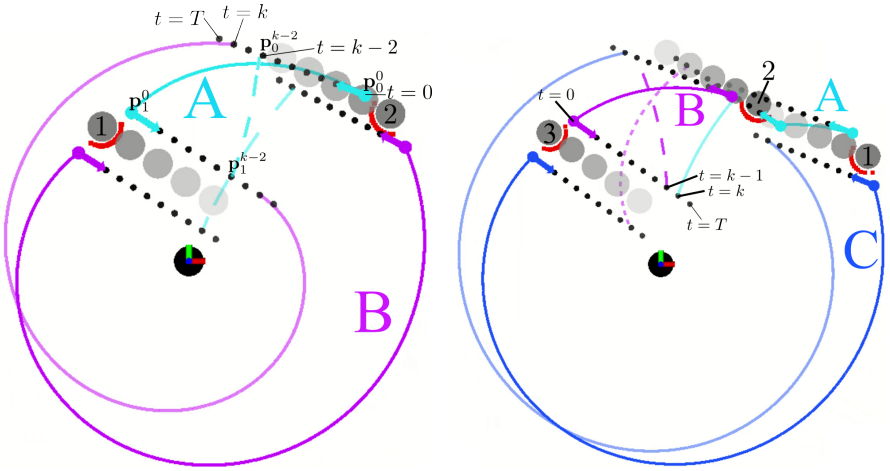
Once a set of propagated gaps $\mathcal{G}^{\text{prop}}_k$ has been extracted for future time t_k, a gap association step is performed between $\mathcal{G}^{\text{prop}}_{k-1}$ and $\mathcal{G}^{\text{prop}}_k$. This step is identical to the Rectangular Assignment problem discussed in Section 5.2, the only difference here being that entire gaps are associated instead of individual gap points. Since gaps are associated, the distance metric used is the sum of squared differences between the points of the two gaps G_i and G_j:

Algorithm 2 Propagated Gaps Extraction

Given: Manipulated gap points P^{manip}
Return: Set of propagated gaps $\mathcal{G}_k^{\text{prop}}$
/ Initialization */*
$N = |P^{\text{manip}}|$
$\mathcal{G}_k^{\text{prop}} = \{\}$
i_0 = index of first right gap point
$i \leftarrow 0$
/ Extraction */*
for $i = 0$ to N **do**
    $\mathbf{p}_i^{\text{manip}} = P^{\text{manip}}(i_0 + i)$
    if $\mathbf{p}_i^{\text{manip}}$ is not assigned to a gap **then**
        $l_i \leftarrow$ gap side for $\mathbf{p}_i^{\text{manip}}$
        $u_i \leftarrow$ ungap ID for $\mathbf{p}_i^{\text{manip}}$
        for $\Delta i = 1$ to N **do**
            j = $(i + \Delta i)$ % N
            $\mathbf{p}_j^{\text{manip}} = P^{\text{manip}}(i_0 + j)$
            if $\mathbf{p}_j^{\text{manip}}$ is not assigned to a gap **then**
                $l_j \leftarrow$ gap side for $\mathbf{p}_j^{\text{manip}}$
                $u_j \leftarrow$ ungap ID for $\mathbf{p}_j^{\text{manip}}$
                if $l_i \neq l_j$ and $u_i \neq u_j$ **then**
                    if l_i is right **then** ▷ Right to left: available gap
                        Add an available gap between $\mathbf{p}_i^{\text{manip}}$ and $\mathbf{p}_j^{\text{manip}}$ to $\mathcal{G}_k^{\text{prop}}$
                    else ▷ Left to right: unavailable gap
                        Add an unavailable gap between $\mathbf{p}_i^{\text{manip}}$ and $\mathbf{p}_j^{\text{manip}}$ to $\mathcal{G}_k^{\text{prop}}$
                    end if
                end if
            end if
        end for
    end if
end for

$$d_{ij} = \|\mathbf{p}_{l,i} - \mathbf{p}_{l,j}\|_2^2 + \|\mathbf{p}_{r,i} - \mathbf{p}_{r,j}\|_2^2. \tag{11}$$

From this association step, there can be several outcomes. If the associated gaps G_i and G_j share the same left and right points, then this gap was not interrupted by any other gaps during propagation and it can be left alone. However, if two associated gaps have different points, this indicates some form of a gap interruption. This interruption might be caused by a gap closing and eventually re-opening (as shown in Figure 7a), or by one gap being propagated into the space of a different gap (as shown in Figure 7b). In either of these scenarios, the gap dynamics change, and a new gap must be instantiated within this particular gap tube.



(a) Gap propagation for two crossing agents. The bold colored arcs labeled A and B are the two gaps detected at time $t = 0$. The lines of black points are the gap points propagated outwards from $t = 0$ to $t = T$. The transparent colored arcs represent the gaps formed at future time steps. Available gaps are represented by solid arcs and unavailable gaps are represented by dashed arcs. At time $t = k - 2$, the cyan arc is predicted to close, signaling that this gap is no longer available. At time $t = k$, the cyan gap points change dynamics, but the gap remains unavailable, marked by a second dashed arc. At $t = k$, the magenta arc changes gap dynamics but remains available.

(b) Gap propagation for two translating agents (labeled 1 and 2) and a third interrupting agent (labeled 3). The bold colored arcs labeled A -C are the three gaps detected at time $t = 0$. At time $t = k - 1$, the magenta arc is predicted to close, signaling that this gap is no longer available. At time $t = k$, the cyan gap is interrupted by agent 3, marked by a transparent cyan arc, but the gap remains available at this time. The magenta arc changes gap dynamics but continues to be unavailable as well Lastly, the blue gap changes dynamics but continues to be available.

Fig. 7 Visualizations of gap propagation algorithm.

5.5.2 Pursuit Guidance Analysis

In this section, the guidance law-based trajectory generation scheme that dynamic gap employs in order to determine if a gap tube is kinematically feasible will be outlined. Guidance laws [81] comprise a set of kinematic equations and feedback control laws that define collision course behavior between a pursuer and a target. While commonly affiliated with older forms of missile guidance, these laws have also seen use in many robotics applications [82, 83, 38].

Among the more established guidance laws, the two geometrical rules of Pure Pursuit (PP) and Parallel Navigation (PN) are the most popular. The PP rule, sometimes referred to as pursuit guidance, has the pursuer direct their velocity vector towards the target at all times, always keeping the target within the pursuer's line of sight. PP has seen a great deal of attention due to its simplicity [84, 85, 86], but this

guidance law only leads to a collision if the pursuer is capable of traveling at a speed faster than that of the target.

The PN, or constant bearing, rule [87, 88] has the pursuer direct their velocity vector such that the direction of the line of sight between the pursuer and target remains constant while the distance between them decreases. This geometrical rule is capable of yielding collision course conditions even if the pursuer is traveling slower than the target. Furthermore, for a non-maneuvering target, meaning a target that is not changing its speed nor its heading direction, PN is the optimal guidance law which yields a minimum intercept time. For a single gap tube, each gap in the gap tube is evaluated independently for feasibility. A gap tube is feasible if all gaps within the tube are feasible.

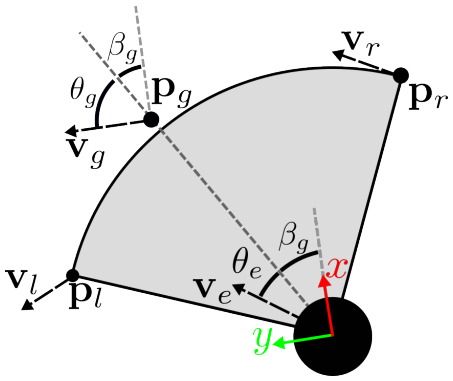


Fig. 8 Diagram for guidance law notation for a single gap. The bearing β_g denotes the bearing at which the gap goal is at for $t = 0$. The angles $\gamma_g = \theta_g + \beta_g$ and $\gamma_e = \theta_e + \beta_g$ represent the directions in which the gap goal position and the robot position move for interception.

This feasibility analysis employs the PN geometric rule [81] which assumes a constant gap goal speed v_g as well as a constant ego-robot speed v_e. This policy then defines the bearing $\gamma_e = \theta_e + \beta_g$ towards which v_e will be applied. With relation to Figure 8, the PN policy is defined as $\dot{\beta}_g = 0, \dot{r}_g < 0$, where

$$\begin{bmatrix} \dot{\beta}_g \\ \dot{r}_g \end{bmatrix} = \begin{bmatrix} \frac{v_g \sin(\theta_g) - v_e \sin(\theta_e)}{r_g} \\ v_g \cos(\theta_g) - v_e \cos(\theta_e) \end{bmatrix}, \tag{12}$$

and $r_g := \|\mathbf{p}_g\|$. In order for $\dot{\beta}_g = 0$ to hold,

$$v_g \sin(\theta_g) = v_e \sin(\theta_e). \tag{13}$$

In order for $\dot{r}_g < 0$, it must be that

$$v_e \cos(\theta_e) < v_g \cos(\theta_g). \tag{14}$$

Therefore, for a constant speed ratio $K := v_e / v_g$, it follows that the gap goal position can be attained if

$$\sin(\theta_e) = \frac{\sin(\theta_g)}{K}, \tag{15}$$

and

$$\cos(\theta_e) > \frac{\cos(\theta_g)}{K}. \tag{16}$$

If these conditions can be met, then the ego-robot will intercept the goal position at the time

$$t_{\text{intercept}} = \frac{r_g^0}{v_g} \cdot \frac{1}{K \cdot \cos(\theta_e) - \cos(\theta_g)}, \tag{17}$$

where r_g^0 is equal to r_g at $t = 0$. If these conditions can not be satisfied, then the given gap tube is deemed infeasible and discarded. If the conditions can be satisfied, but $t_f < t_{\text{intercept}}$, meaning that the gap will cease to exist before the ego-robot can intercept the goal position, then the gap tube is also deemed infeasible and discarded. Gap tubes that satisfy this condition are added to $\mathcal{T}^{\text{feas}}$.

5.5.3 Proof of Collision-Free Passage

Theorem 1: Given

1. A first-order holonomic ego-robot,
2. A feasible manipulated gap with constant velocity left and right points $\mathbf{p}_l, \mathbf{p}_r$,
3. An isolated local environment meaning that no other gaps will enter the gap in focus during the local time horizon,

then performing the PN policy towards the gap goal point $\mathbf{p}_g$ will yield collision-free gap passage.

Proof: Let the gap goal point $\mathbf{p}_g$ and velocity $\mathbf{v}_g$ be defined as a convex combination of the left and right gap point states,

$$\begin{aligned} \mathbf{p}_g &= \kappa \mathbf{p}_l + (1-\kappa)\mathbf{p}_r, \\ \mathbf{v}_g &= \kappa \mathbf{v}_l + (1-\kappa)\mathbf{v}_r, \qquad \kappa \in [0,1]. \end{aligned} \tag{18}$$

If the gap in focus possesses an angular span of greater than π radians, meaning that the gap is nonconvex and a convex combination may lie outside of the gap, the gap span can be artificially reduced to two points within the original gap span that form a convex polar triangle. It follows that

$$\beta_g = \arctan(\mathbf{p}_g) = \arctan(\kappa \mathbf{p}_l + (1-\kappa)\mathbf{p}_r). \tag{19}$$

Without loss of generality (by rotating the egocentric frame to align with the center of the initial gap), $\beta_g \in [\beta_r, \beta_l]$ given that arctan is a monotonically increasing function. Following the same argument for

$$\gamma_g = \arctan(\mathbf{v}_g) = \arctan(\kappa \mathbf{v}_l + (1-\kappa)\mathbf{v}_r), \tag{20}$$

it can be seen that $\gamma_g \in [\gamma_r, \gamma_l]$. Given that

$$\gamma = \beta + \theta, \tag{21}$$

it follows that $\theta_g \in [\theta_r, \theta_l]$. From Equation 15,

$$\theta_e = \arcsin\left(\frac{\sin\theta_g}{K}\right), \tag{22}$$

and given that arcsin is also a monotonically increasing function, this means that $\theta_e \in [\theta_{e/r}, \theta_{e/l}]$ where $\theta_e, \theta_{e/l}, \theta_{e/r}$ are the bearings at which the ego-robot must direct its velocity at to intercept the gap goal, the left gap point, and the right gap point, respectively. This indicates that under the PN policy, the ego-robot will intercept the gap goal point between the left and right gap points, therefore performing collision-free gap passage. □

While it is clear that in practice, few gaps will satisfy these constant velocity and isolated environment assumptions, the trajectory generation scheme for the proposed planner is still built upon a collision-free guarantee. Supplementary modules including gap propagation, scan propagation, and safety filters are all employed to mitigate risk in situations where assumptions are violated.

5.6 Gap Trajectory Generation

Gap Goal Placement: For a given gap G_i within a gap tube $\mathcal{T}_j$, the gap goal state $\mathbf{X}_g$ is a convex combination of the left and right gap point states. The position of a local waypoint $\mathbf{p}*$ from the global trajectory ξ^{global} can be used to bias this gap goal towards one side. For ungaps, the gap goal is inflated inwards to ensure that it can be reached without colliding with the obstacle creating the ungap.
Gap Trajectory Rollout: Gap tube trajectories are obtained by integrating the ego-robot position forward along the intercepting heading γ_e resulting from the pursuit guidance analysis in Section 5.5.2. Ungap trajectories are synthesized in the same manner.

5.7 Gap Trajectory Scoring

5.7.1 Pose-wise Scoring

In order to determine which trajectory to track, each trajectory is evaluated by an egocentric pose-wise cost based on proximity to local obstacles and a terminal pose cost based on proximity to a local waypoint along the global plan in order to encourage progress toward the global goal.

The cumulative cost for a trajectory ξ is

$$\mathcal{J}(\xi) = w\|\mathbf{p}[N] - \mathbf{p}^*\| + \frac{1}{N}\sum_{k=1}^{N} C(d(\mathbf{p}[k]; \mathcal{L}[k])) \tag{23}$$

where $d(\mathbf{p}; \mathcal{L})$ is the distance from the pose $\mathbf{p}$ to the laser scan $\mathcal{L}$, $w \in \mathbb{R}^+$ is a weighting factor, $\mathbf{p}^*$ is a local waypoint along the global path, and

$$C(d) = \begin{cases} \infty, & d \leq r_{\text{infl}} \\ c_{\text{obs}} e^{-w_2(d - r_{\text{infl}})}, & r_{\text{infl}} < d < r_{\max} \\ 0, & r_{\max} \leq d \end{cases}, \tag{24}$$

where $c_{\text{obs}}, w_2 \in \mathbb{R}^+$ are weighting factors, r_{infl} is the inflated radius of the ego-robot, and $r_{\max}$ is the maximum distance that is penalized. Pose-wise scores are averaged so as to not bias selection towards shorter trajectories.

5.7.2 Dynamic Scan Propagation

The trajectory cost formulation is adapted from [2], with one key difference: pose-wise scoring requires a laser scan for each timestep t along the trajectory. However, these future laser scans are not directly accessible. In practice, when gaps are propagated forward in time, a set of propagated laser scans are recovered and stored for later scoring by estimating the laser scan dynamics $\dot{\mathcal{L}}$. The algorithm used to back out these propagated scans from a set of predicted gaps is given in Algorithm 3. An example set of propagated scans obtained during planning is visualized in Figure 9.

In short, a dynamics model for the laser scan $\dot{\mathcal{L}}$ is approximated by assigning gap point dynamics models to each scan point if the neighboring gap point models are sufficiently similar. In this algorithm, the condition areSimilar that determines if two gap point states $(\mathbf{p}_{\text{LHS}}, \mathbf{v}_{\text{LHS}})$ and $(\mathbf{p}_{\text{RHS}}, \mathbf{v}_{\text{RHS}})$ are similar is evaluated as

$$\|\mathbf{v}_{\text{LHS}}\|_2 \geq v_{\min} \wedge \|\mathbf{v}_{\text{RHS}}\|_2 \geq v_{\min} \wedge \mathbf{v}_{\text{LHS}} \cdot \mathbf{v}_{\text{RHS}} > 0. \tag{25}$$

Given that propagated scan points can change both their bearing and range, care must be taken to ensure that propagated points are mapped to the correct scan point index.

The highest scoring candidate trajectory is compared against the currently executing trajectory to determine if a trajectory change should occur.

5.8 Gap Trajectory Comparison

The core idea behind safe hierarchical planners involves chaining together multiple safe local trajectories en route to the global goal. Therefore, a method of triggering a switch to a newly synthesized local trajectory must be defined. In this work, an

Algorithm 3 Dynamic Scan Propagation

Given: Current set of raw gap points P^{raw}
Given: Current laser scan $\mathcal{L}_0$
Return: Set of propagated laser scans $\{\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_T\}$
/ Assignment */*
$\dot{\mathcal{L}} = \{\}$
for $(\beta_i, r_i) \in \mathcal{L}_0$ **do**
  for $\mathbf{p}_j \in P^{\text{raw}}$ **do**
    $\beta_j = \arctan(\mathbf{p}_j)$
    if $\beta_j \leq \beta_i$ **then**
      $\mathbf{p}_{\text{RHS}} \leftarrow \mathbf{p}_j$
    else
      $\mathbf{p}_{\text{LHS}} \leftarrow \mathbf{p}_j$
      break
    end if
  end for
  if areSimilar($\mathbf{p}_{\text{LHS}}, \mathbf{p}_{\text{RHS}}$) **then** ▷ Equation 25
    $\dot{\beta}_i \leftarrow (\dot{\beta}_{\text{LHS}} + \dot{\beta}_{\text{RHS}})/2$
    $\dot{r}_i \leftarrow (\dot{r}_{\text{LHS}} + \dot{r}_{\text{RHS}})/2$
  else
    $\dot{\beta}_i = 0$
    $\dot{r}_i = 0$
  end if
end for
/ Propagation */*
for each timestep t_k in planning horizon T **do**
  $\mathcal{L}_k = \mathcal{L}_0 + \dot{\mathcal{L}} \cdot t_k$
end for

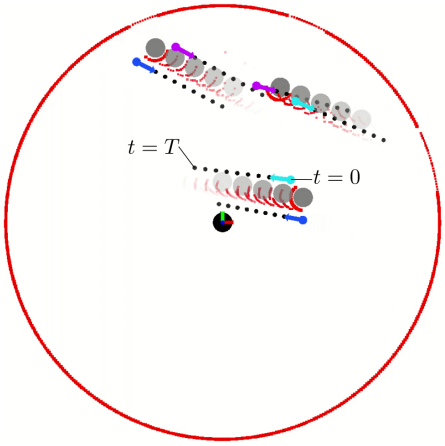


Fig. 9 Visualization of dynamic scan propagation. The colored points are the gap points along with their estimated velocities. Similar gap points are used to propagate scan points from $t = 0$ to $t = T$, represented by the increasingly transparent scan points.

event-based trajectory switching scheme is employed in which a trajectory switch only occurs if one of a set of conditions is met, as described below.

If the trajectory that is currently being tracked has either been completed (evaluated by proximity to the trajectory end or by the trajectory timing) or determined to be on a collision course, then a switch is triggered to the newly synthesized incoming trajectory. Similarly, if the gap that the currently tracked trajectory passes through has been determined to be infeasible as per Section 5.5, a switch is triggered.

5.9 Gap Trajectory Tracking

Once a local trajectory is chosen for tracking, a simple state feedback control law is deployed to track the trajectory.

5.10 Projection Operator

As a last resort safety filter to handle non-ideal circumstances including discrete time implementation and second-order dynamics, the proposed work also adapts the projection operator module which is detailed in [2].

6 Experimental Results

This planner is implemented as a C++ Robot Operating System (ROS) node through the `move_base` package [89]. All simulation tests are run one at a time in the Arena-RosNav simulation environment on a Dell Precision 3660 Tower with an Intel i9-12900K CPU with 16 cores (single-thread passmark of 4, 336; multi-thread score of 41, 322). All hardware tests are run on a Dell XPS 13 laptop with an Intel i5-10210U CPU with 4 cores (single-thread passmark of 2, 145; multi-thread score of 6, 152).

6.1 Experiment One: Assumption-satisfying Experiments

First, example trajectories are generated through single gaps to demonstrate how under ideal conditions, pursuit guidance-based policies can generate provably safe trajectories. A single gap is randomly generated and the PN policy is employed for trajectory generation. With the gap centered at the origin, left gap points are uniformly sampled from $\beta_l \in [\frac{\pi}{2}, \frac{3\pi}{2}]$, $r_l \in [0.25, 1.0]$ m. Right gap points are uniformly sampled from $\beta_r \in [\frac{-\pi}{2}, \frac{\pi}{2}]$, $r_r \in [0.25, 1.0]$ m. Gap point velocities are

uniformly sampled from all directions with magnitudes within the range $[0.0, 1.0]$ m/s. A finite robot radius of 0.20 m is also accounted for by the inflation policy in Section 5.4.2. A subset of gaps are shown in Figure 10.

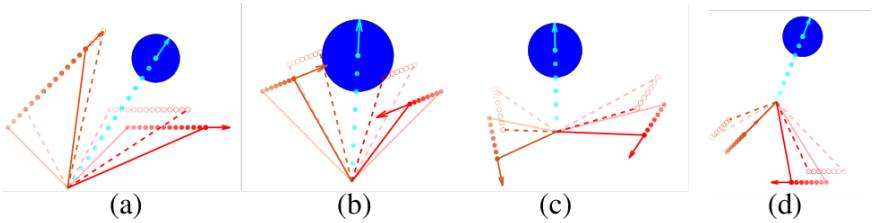


Fig. 10 Visualization of four Monte Carlo variations of gaps from Experiment One. Orange points and lines represent the left side of the gap while red points and lines represent the right side. Solid points and lines represent the original gap geometry whereas hollow points and dashed lines correspond to the inflated version of the gap. Transparent points represent positions at prior timesteps. The blue circle represents the robot along with its finite radius.

For this experiment, $10,000$ trials were run: $6,987$ trials end in collision-free gap passage, $2,668$ trials resulted in a kinematically infeasible gap due to the velocity limits of the robot, and for the remaining 345 trials, the robot was unable to pass through the gap before it closed. No collisions occurred during any trials.

6.2 Experiment Two: Canonical Scenarios

In this section, a collection of isolated canonical scenarios are demonstrated to provide further insights into how dynamic gap plans and predicts in dynamic settings.

6.2.1 Closing and Re-opening

In this scenario (Figure 11), the ability of dynamic gap to predict gaps to close and subsequently re-open is highlighted. In this scenario, the robot must move from the left side of the corridor to the right side beyond the two agents. The frame notations $i = n$ in Figure 11 refer to the planning loop iteration at which the frame was captured. At $i = 21$, the planner can propagate the central gap forward in time to determine that it will close and re-open during the planning horizon. Therefore, a piecewise trajectory is generated that leads the ego-robot up to the crossing gap, holds the ego-robot in place while the gap reopens ($i = 40$), and then leads the ego-robot through the newly opened gap ($i = 49$).

The original version of dynamic gap only propagated gaps forward up until the point at which their dynamics are altered, in this scenario being when the central gap closes. This means that the original dynamic gap planner has no way to predict that the central gap will reopen in the future for safe gap passage.

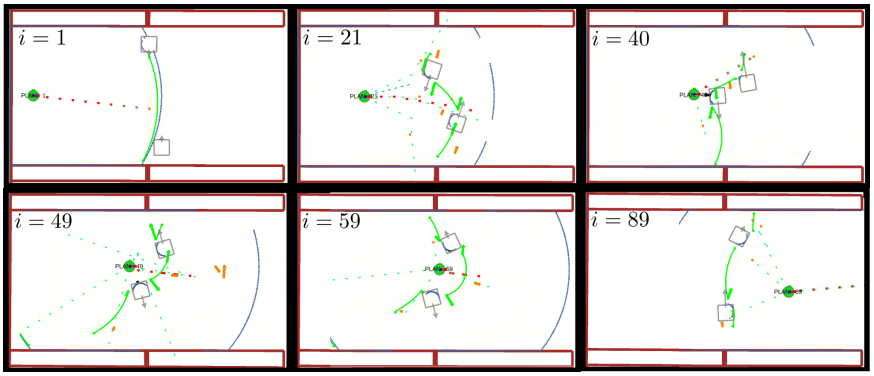


Fig. 11 Canonical Scenario One - closing and re-opening gap. The green circle with the attached frame is the ego-robot, and the gray squares are the agents. The dotted blue arc shows the current scan data, green arcs show the current manipulated gaps, orange arrows represent the position and velocity of the local gap goals, light blue pointed lines are the generated trajectories, and the red pointed line is the trajectory that is currently being executed by the robot.

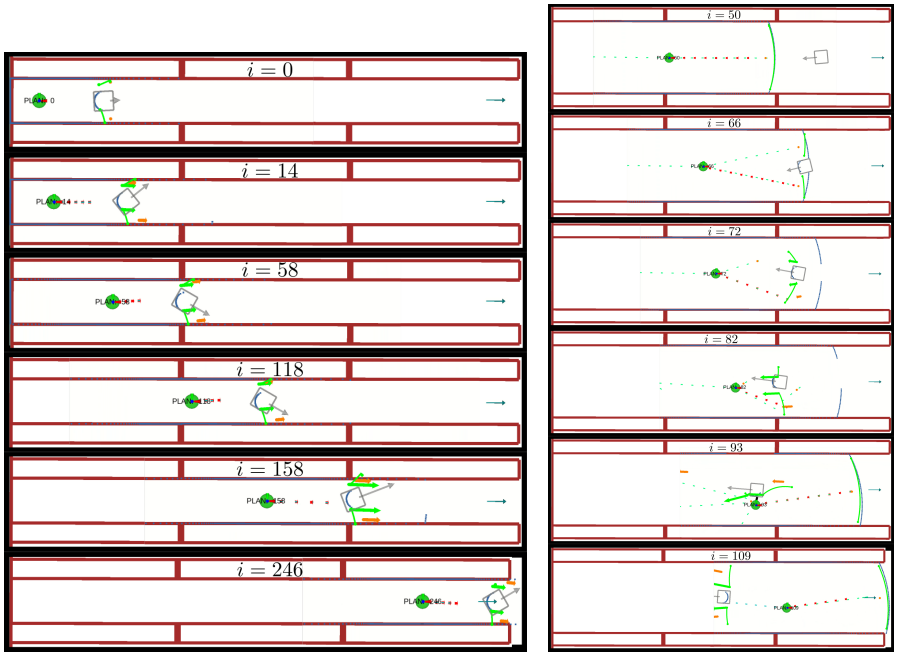
6.2.2 Corridor - receding

In this scenario (Figure 12a), the ability of dynamic gap to plan through a presently occupied region of space, referred to as the "ungap", is highlighted. In this scenario, the robot must move from the left side of the corridor to the right side, where the goal is represented by the blue arrow. In this setup, the corridor is deliberately designed to be too narrow for the ego-robot to move around the agent to pass it and continue on. There are two gaps detected on either side of the agent, but the trajectories are short and would cause collisions with the walls. Starting at frame $i = 14$, the planner can be seen to select the central ungap trajectory that allows the ego-robot to trail behind the agent. This ungap trajectory is consistently generated and tracked in all subsequent frames up until $i = 246$ when the robot reaches the goal.

The original dynamic planner could not generate trajectories through ungaps. Therefore, the planner would have to wait for the agent to clear out of the nearby space before it could plan a trajectory through the corridor.

6.2.3 Corridor - approaching

In this scenario (Figure 12b), the ability of dynamic gap to plan around an oncoming obstacle in a tight corridor is demonstrated. Similar to the last scenario, the planner must get the robot from the left side of the corridor to the right side, where the goal is represented by the blue arrow. In this setup, the corridor is wider compared to prior scenario, thereby allowing for the ego-robot to move around the agent to pass it and continue on. At iteration $i = 66$, the oncoming obstacle has been detected in the laser scan, but the gap point models have yet to converge to an accurate velocity estimate. By iteration $i = 72$, the approaching agent is both detected and predicted



(a) Canonical Scenario Two - receding corridor.

(b) Canonical Scenario Three - approaching corridor.

Fig. 12 Canonical Scenarios Two and Three - corridor.

to be traveling towards the ego-robot. The planner successfully predicts this agent forward, generating two trajectories that pass the agent on the left and right sides.

6.2.4 Four-way intersection

This scenario (Figure 13) highlights the ability of the dynamic gap planner to plan through a complex four-way intersection scenario. In this scenario, the planner must move the robot from the left branch of the intersection to the right branch, where the goal is represented by the blue arrow. In this setup, two agents pass through the top and bottom branches of the intersection.

By iteration $i = 35$, the agents are detected and the planner initially opts to pass between the two agents. However, at planning iteration $i = 60$ the two agents are attempting to de-conflict their own motion plans and staying in the center of the intersection, making it difficult for the planner to pass through them. Due to this, the planner moves the ego-robot away from the intersection at iteration $i = 80$. By iteration $i = 129$, the gap between the agents has re-opened and the planner generates a trajectory through it to continue through the intersection.

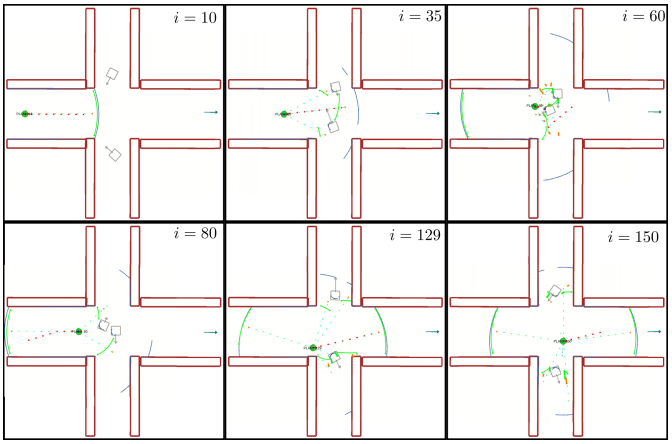


Fig. 13 Canonical Scenario Four - intersection.

6.3 Experiment Three: Simulation Benchmarking

To gain a better understanding of the performance level of the dynamic gap planner, a comprehensive series of simulation benchmarks are run with other state-of-the-art local perception-informed planners. The dynamic gap planner is integrated into the Arena-Rosnav [8] benchmarking environment and compared against four classical cost map-based planners and four learned planners. Additionally, two other GBPs are tested against including the static world predecessor of dynamic gap, known as potential gap, and the prior version of dynamic gap. The version of dynamic gap proposed in this chapter is benchmarked under both a holonomic robot model as well as a nonholonomic robot model. These baselines provide insights into the performance improvements obtained from the work presented in this chapter. All baselines are detailed in Section 6.3.1.

Three environments are used, shown in Figure 14, referred to as empty, factory, and hospital. With these environments, the authors aim to build a gradient of increasing environment structure to evaluate each planner's ability to not only navigate dynamic obstacles, in simulation represented as pedestrians, but also the static, non-trivial structure of these worlds such as corridors, rooms, and atria.

Within each environment, 15 dynamic agents are placed at predefined start points. Then, a path is generated from each agent's start to its goal which the agent subsequently tracks. Both the agents and the ego-robot have velocity limits of $v_x^{\max} = v_y^{\max} = \omega_z^{\max} = 1.0$ m/s. For each planner / environment combination, 25 tests are run, consisting of every combination of the five start and goal positions portrayed in Figure 14. These 25 trials are also run with no agents present to give insight into how the planners perform in a purely static version of each environment. For all trials, planners are given three minutes to reach the goal.

For navigation-level performance, two dependent variables are captured: whether or not the planner reached the goal under the prescribed time limit for the given

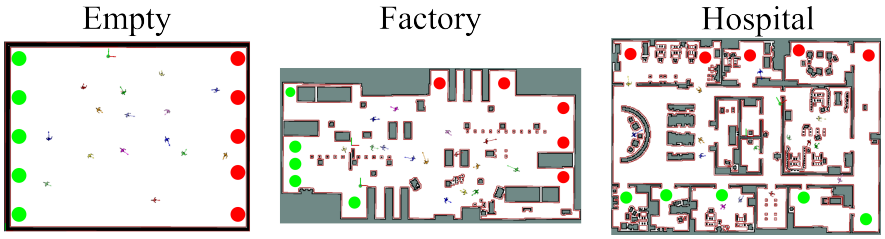


Fig. 14 Three simulation environments used during benchmarking. Sizes are Empty - 31 × 24 m, Factory - 38 × 19 m, Hospital - 34 × 24 m. Green dots represent the different start positions and red dots represent different goal positions for simulation benchmarking. Each combination is evaluated one time, yielding 5 × 5 = 25 trials for each environment.

trial, and whether or not the planner registered any collisions during the trial. Trials in which the planner reached the goal under the time limit without sustaining any collisions are considered a *success*. Trials in which the planner does not reach the goal before the time limit, but also does not sustain any collisions are depicted as *timed out*. Trials for which the planner reaches the goal under the time limit but sustains collisions are treated as *failed*. Trials that both did not reach the goal before the time limit and also sustained collisions are depicted as *failed and timed out*. Results are reported by environment in Sections 6.3.2, 6.3.3, and 6.3.4.

6.3.1 State-of-the-art Baselines

In this chapter, four classical cost map-based planners are evaluated:

Timed Elastic Bands (TEB) [15]: TEB is a local planner which formulates a soft constraint optimal control problem with respect to trajectory execution time, obstacle separation, and compliance with kinodynamic constraints. The planner performs blob detection on a local costmap to detect obstacles and runs KFs to estimate the dynamic obstacle states.

Co-operative Human-Aware Navigation (CoHAN) [90]: CoHAN is a tunable human-aware navigation planner designed to handle diverse human-robot interaction contexts through context-based planning modes and a social compliance cost function. From a planning perspective, CoHAN is a socially aware version of TEB.

Model Predictive Control (MPC) [91]: The authors from TEB also developed a receding horizon-based optimal controller that solves an optimal control problem much like the one formulated in the original TEB work.

Dynamic Window Approach (DWA) [35]: DWA randomly samples the robot's control space (v_x, v_y, ω) and rolls out the resultant trajectory for a short period of time. Each candidate trajectory is evaluated on proximity to obstacles, the global goal, and the global path as well as speed. Colliding trajectories are discarded, but the implementation that is benchmarked against has no dynamic obstacle prediction.

In addition, four data-driven planners are evaluated:

Deep Reinforcement Learning - Velocity Obstacles (DRL-VO) [55]: DRL-VO is a deep reinforcement learning approach with a reward function based on VOs that penalizes headings that are likely to lead to collisions using relative motion rather than distance. Inputs include pedestrian states, a sliding window of laser scans, and a local waypoint. Their policy is trained with the PPO (Proximal Policy Optimization) algorithm in the Gazebo simulation.

Reinforcement Learning for Collision Avoidance (RLCA) [52]: The RLCA method is a reinforcement learning model that directly maps the raw sensor data, in this case a laser scan, to a desired, collision-free steering command. For this specific network, a sliding window of the three past scans are passed into the model along with a local waypoint from the global path and the robot's current velocity. The reward function for this model is comprised of three terms. First, a reward term for making progress towards the local waypoint. Second, a flat penalty for collisions. Lastly, a regularization term to discourage large angular velocities. This formulation is solved using PPO in the Stage simulator [92].

Collision Avoidance with Deep Reinforcement Learning (CADRL) [93]: For CADRL, ground truth state information for nearby agents, —including position, velocity, and radius — is passed into model through a Long Short-Term Memory (LSTM) network, and the ego-robot state information — including a local waypoint from the global path, the preferred velocity, and the orientation — is then appended to the input. The reward function administers a positive value if the local waypoint is reached, a linearly decreasing negative value if an agent is close to the robot, and a flat negative value if an agent is in collision with the robot. This model is trained with Asynchronous Advantage Actor-Critic (A3C) algorithm using a custom simulation engine.

Learning from Learned Hallucination (LfLH) [56]: LfLH is a self-supervised framework that leverages automated hallucination to generate synthetic obstacle configurations during training rather than relying on trial-and-error exploration. An encoder-decoder structure is used for hallucination in which the robot state, goal state, and robot plan are passed to the encoder during training. The encoder then learns to predict probability distributions of nearby obstacles. The decoder samples from these obstacle distributions and generates a motion plan. Then, a motion planner is learned through behavior cloning by sampling the learned encoder to generate hallucinated obstacle perceptions and comparing the output action against the actions of an optimization-based motion planner that is treated as an expert under this framework.

Lastly, a family of GBPs are evaluated:

Potential Gap (PGap) [2]: Potential Gap is a gap-based planner designed for static environments. This planner detects the current set of gaps in the local environment, manipulates the set of gaps to ensure gaps are convex polar triangles, and then deploys an attractive potential and circulation field to drive the robot through gaps.

Dynamic Gap (DGap) [7]: This baseline is the prior version of the planner discussed in this chapter. This baseline propagates gaps only up until the point in which they close or propagate into another gap. This baseline also has no means to generate trajectories through ungaps.

6.3.2 Empty

Given that this environment is solely comprised of a single room, these trials can be used to evaluate each planner's pure dynamic obstacle avoidance mechanisms. Results for this environment are shown in Figure 15. For the static version of the empty environment, all planners succeeded on all 25 trials. The only notable takeaway from this set of tests is that the CADRL planner took roughly three times longer (79 – 91 seconds) to reach the goal compared to other baselines (28 – 36 seconds).

For the dynamic version of the empty environment, failure modes start to arise. In this setting, all unsuccessful trials are the result of collisions, and no timeouts were observed. Average performance across the classical cost map-based planners (the leftmost four benchmarks) is on par with the average performance for the learned planners (the middle four benchmarks), with the classical planners averaging a success rate of 42% while the learned planners average a success rate of 50%. However, the learned planners exhibit more variance in that CADRL performs worse (24% success rate) and LfLH performs better (76% success rate) compared to the average. Of the three environments, this is the one in which the set of learned planners performs the best. This is in part due to the environment closely resembling the training environments used to optimize the learned models, meaning that the scenarios encountered in this environment were more likely to exist inside of the distribution of data used to train the models.

While the costmap representation provides a way to discretely identify obstacles, this set of planners still struggled at times to avoid oncoming obstacles. These planners were often able to slightly adjust their heading to direct the ego-robot away from nearby agents. However, when agents entered the local costmap heading towards the ego-robot, the planners were not able to react in time to avoid the agent.

For the learned benchmarks, the planners all possessed the emergent property of coming to a stop directly in front of obstacles when the robot encroaches upon them, relying on non-adversarial motions of nearby agents to avoid collisions.

The GBPs all performed well in this setting. Gaps proved to be an effective representation due to their ability to naturally bias the planning space away from nearby obstacles. When agents did approach the ego-robot, the gap detection policy naturally partitions the planning space to either side of these agents which guides planning around the agents. The proposed version of dynamic gap deployed on a holonomic robot model outperforms all other baselines in this environment.

6.3.3 Factory

The factory setting consists of one large room with many smaller isolated static obstacles positioned within it, resembling real-world artifacts such as tables or pillars. The results for this environment are shown in Figure 16. Here, the larger regions of free space allow multiple agents to pass through the same part of the environment at one time. This structure causes more collisions than the empty environment.

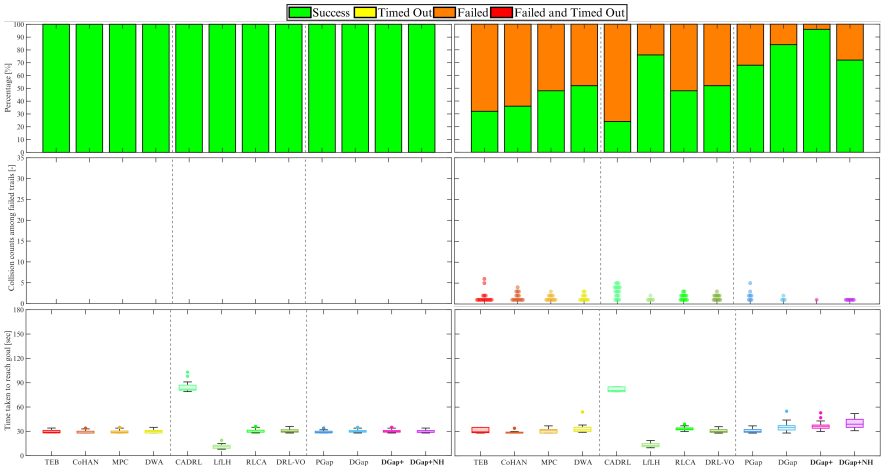


Fig. 15 Simulation benchmarking results for the Empty simulation environment. The left column represents trials with no agents in the environment, and the right column is for trials with 15 agents in the environment. The top row portrays navigation performance results for all benchmarks across the 25 trials. Green bars (success) are successful trials in which the planner reached the goal under the time limit without sustaining any collisions, yellow bars (time out) are trials for which the planner does not reach the goal before the time limit but also does not sustain any collisions, orange bars (failure) are trials for which the planner reaches the goal under the time limit but sustains collisions, and red bars (failure and time out) are for trials that do not reach the goal under the time limit and also sustain collisions. Results are shown in percentage format. The middle row depicts the collision counts amongst failed trials, meaning how many collisions actually occurred during each trial that registered a positive number of collisions. Lastly, the bottom row depicts average time taken to reach the goal for successful trials.

The four learned baselines all accept either a single raw laser scan or a sliding window of raw laser scans as their sensory input, meaning that these planners do not perform any form of environment abstraction. Building an environment representation can serve many purposes: reducing input size by distilling large sensor data into lower-dimensional formats, extracting additional unobservable features from the sensor data, and preprocessing the noisy sensor data to remove artifacts or inaccuracies. By foregoing this environment abstraction step, the task of extracting meaningful environment information from inputs becomes much more difficult for a learned model. The authors posit that this is one of the reasons for the significant performance drop amongst learned planners. The RLCA and DRL-VO models both struggled heavily with the static structure of the environment and timed out on a handful of trials because of their inability to get through narrow corridors or around tight corners. These planners also rarely backed up when in collision with the environment which can often help get the robot reset. The CADRL planner was immensely slow, but this ended up helping the planner navigate through the static environment. The other learned planners often over-rotated or over-compensated other maneuvers, but CADRL was instead very deliberate. This allowed it to reach the goal consistently, but at the cost of very limited collision avoidance abilities due

to its slow pace. Not only do the learned planners exhibit more failed trials, the individual failed trials also sustain a far greater number of collisions.

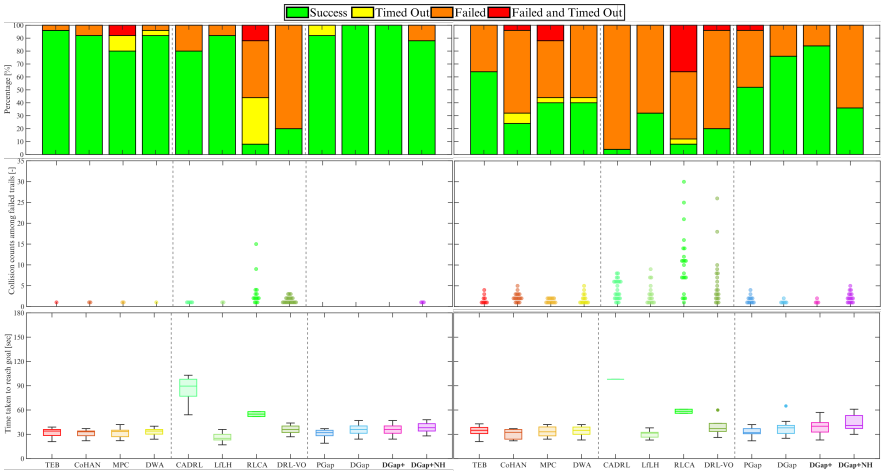


Fig. 16 Simulation benchmarking results for the Factory simulation environment.

The four classical benchmarks all use a local costmap that is propagated through the environment as the robot moves. This costmap takes in scan data and updates a discrete occupancy grid. Then, this occupancy grid is inflated, scored, and planned over. The Cartesian frame costmap is a very simple and intuitive environment representation that makes planning algorithms such as search and optimization easy to execute. Some baselines also perform motion estimation steps on this costmap to predict how the map might evolve in the future. The classical baselines perform better in the Factory environment than the learned baselines with no agents present as well as with 15 agents present. However, the classical planners occasionally clipped corners or hallways during navigation. Without agents, the classical baselines register success rates of $80-96\%$ compared to the success rates of $8-92\%$ from the learned baselines. With 15 agents, the classical baselines register success rates of $24-64\%$ compared to the $4-32\%$ success rates of the learned baselines.

The three gap-based benchmarks all plan using a gap-based environment representation comprised of polar arcs of free space. This environment representation requires minimal computation to synthesize and keeps the scan data in its raw polar format, bypassing the preprocessing steps required to maintain a costmap. Being able to build and propagate this gap-based representation pays dividends in practice, with the family of GBPs exhibiting the best navigation performance of all baselines with success rates of $36-84\%$ in the dynamic setting. The ability of dynamic gap to predict the feasibility of local gaps before committing to them proves paramount in avoiding collisions in this environment. The proposed version of the dynamic gap planner on the holonomic model outperformed the rest of the baselines with its added gap propagation abilities, although the proposed planner on the nonholo-

nomic model only achieved a success rate of 36%. This can largely be attributed to the model mismatch in which trajectories are planned through pursuit guidance techniques under a holonomic model, but then tracked under a nonholonomic model. The focus of this work was planning in dynamic environments, and future work will extend dynamic gap to a nonholonomic model.

6.3.4 Hospital

The hospital environment exhibits many smaller rooms connected through tighter passageways and corridors. Results for this environment are shown in Figure 17. Compared to the factory environment, fewer collisions are registered in this environment because the corridors and small rooms offered fewer opportunities for several agents to enter the same region and collide. Here, more trials ended in time outs, either due to the planner getting stuck in a tight space and not being able to make further progress in the environment, or due to the planner taking too long to reach the goal. Given that this environment is more sprawling, the time taken to reach the goal is higher on average, and more variance is seen in navigation times.

The CoHAN baseline tended to rotate and translate while trying to pass around corners. As a result, the planner would often bump into the corner and "roll" around it, resulting in several collisions. The MPC and DWA baselines occasionally drove backwards through corridors as well.

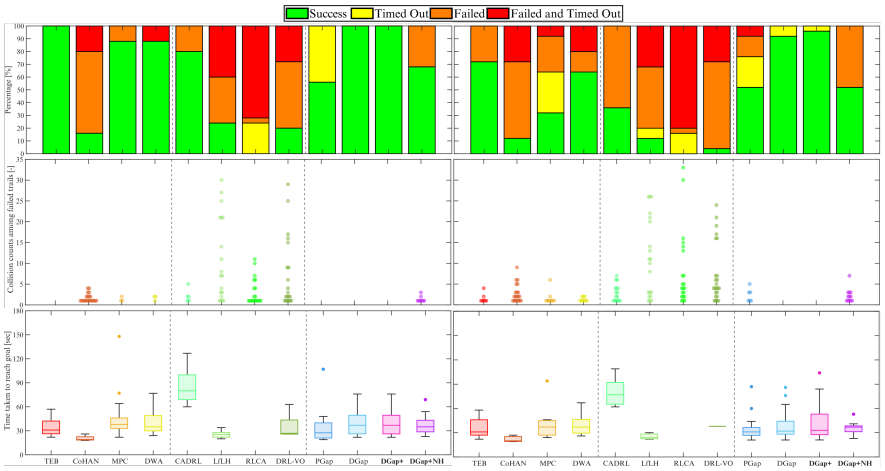


Fig. 17 Simulation benchmarking results for the Hospital simulation environment.

The learned planners exhibited more failed trials and higher collision counts. Overall, learned planners struggled with planning paths through narrow corners and corridors. The RLCA planner tended to over-rotate, leading the ego-robot to hit the walls surrounding corridor entrypoints. Once the robot hit the wall, the planner

was unable to output negative velocity commands to have the robot back up, and trial progress would typically end. The DRL-VO planner exhibited a strong bias in its output space distribution, solely outputting positive (counterclockwise) angular velocities. This would lead to situations where a simple hallway corner that could be passed through with a clockwise rotation could take very long due to the planner's inability to rotate in that direction. The authors mitigated this bias by shifting the interval of possible angular velocities towards negative values, but this only provided marginal performance improvements. The CADRL baseline would commonly output extremely small velocity values when the robot was close to static obstacles. This led to situations in which the planner would take a corner too tight and stall out next to the wall. The authors partially alleviated this issue by applying a small constant forward velocity to the system when there were no obstacles in front of the robot, enabling the robot to drive through these scenarios where the planner stalls out. The CADRL and LfLH planners both possessed a "rotate-in-place" mode that would guide the planner in aligning the ego-robot to point towards the next waypoint from the global goal. This helped these planners get through difficult portions of the environment.

Overall, the GBPs fared well in this setting. While not designed for dynamic settings, the planning behaviors of the potential gap planner still allowed it to navigate through this environment's internal structure. The two dynamic gap versions performed very well, with the holonomic proposed version narrowly outperforming its prior version with a success rate of 96% versus 92%.

6.4 Experiment Four: Timing

In this experiment, we report the average computation times in simulation and hardware for all baselines use during benchmarking. A more in-depth computational analysis is given for dynamic gap in Figures 18a - 18c. All timing experiments in simulation are run in one scenario of the Factory environment with 15 agents present.

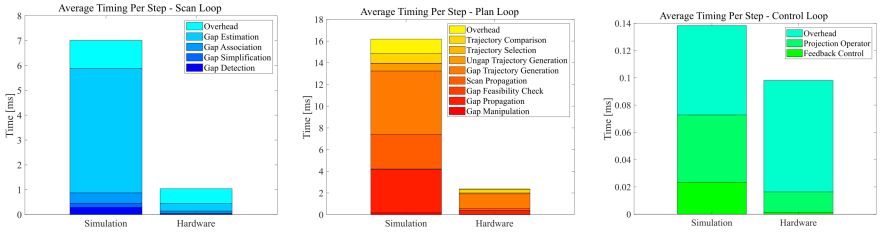
6.4.1 Dynamic Gap

6.4.2 Other Baselines

Planning rates and implementation details for other baselines are shown in Table 1.

Table 1 Reported Computation Times (Hz) for Simulation Benchmarks

TEB	CoHAN	MPC	DWA	RLCA	CADRL	DRL-VO	LfLH	PGap	DGap	DGap+
71	201	39	6	593	539	36	22	26	47	62



(a) Timing for scan thread portrayed in Figure 4.3.

(b) Timing for control thread portrayed in Figure 4.3.

(c) Timing for control thread portrayed in Figure 4.3.

Fig. 18 Timing breakdown for dynamic gap planner. The scan thread can run at ≈ 143 Hz in simulation and ≈ 959 Hz on hardware, but in practice it is set to the rate of the incoming sensor data in both simulation and hardware which is 25 Hz. The planning thread can run at ≈ 62 Hz (for an average of 5.76 gaps per planning loop) in simulation and ≈ 337 Hz (for an average of 5.80 gaps per planning loop) on hardware, but in practice it is set to 5 Hz within the `move_base` framework. The control thread is bound to the planning thread.

6.5 Experiment Five: Social Compliance Performance

A relative velocity cost function based on [90] is added to improve the social compliance of the robot. This new planning variant is referred to as DGap$^+$Social.

$$\text{cost}_{\text{rel\_vel}} = \frac{\left(\max\left(\mathbf{V}_{\text{rel}} \cdot \overrightarrow{P_r P_h}, 0\right) + \|\mathbf{V}_r\|\right)}{\left\|\overrightarrow{P_r P_h}\right\|} \tag{26}$$

The first term involves a dot product in order to penalize paths that are directed towards a human. Moreover, the maximum term restricts the cost to situations in which the robot is approaching a human rather than moving away from them. The second term penalizes high velocities in the vicinity of a human.

Results related to social compliance were also collected from the same experiments previously described in Section 6.3. As shown in Figure 19, DGap$^+$ and DGap$^+$Social achieved the highest success rates among all planners. However, they also exhibited relatively high time spent in personal space which is defined as a circle region surrounding the human with a radius of 0.5 m. This may be partially attributed to the fact that only successful trials were included in the plot. In densely crowded scenarios, many other planners likely failed early, producing no data for those challenging cases. In contrast, DGap methods successfully navigated through such environments which caused it to spend more time in close proximity to people, thereby increasing their average time in personal space.

Interestingly, classical planners such as MPC, DWA, and TEB showed both reasonably high success rates and comparable or lower time in personal space than most learned methods. This outcome is somewhat surprising, given that learning-based approaches are often claimed to yield superior social behavior. For instance, 90% of DWA, TEB, and MPC trials had personal-space durations under 3 seconds, whereas

90% of learned planners stayed under 4 seconds, excluding DRL-VO, which was below 7 seconds. The CoHAN planner had one of the lower success rates. This was likely due to its tendency to attempt to pass in front of pedestrians, as well as its occasional collisions with the environment when turning corners.

6.6 Experiment Six: Hardware Testing

In this section, the dynamic gap planner is deployed on a TurtleBot2, a differential drive platform in which ROS code is executed from an onboard laptop.

6.6.1 Test One: Static obstacle course

In this test, the ability of the dynamic gap planner to operate in a static setting is confirmed on hardware. The keyframes of this test are shown in Figure 20. In this scenario, the goal is placed at the top of the frame. At frame $i = 210$, the ego-robot selects the gap in the center of the frame. By frame $i = 468$, further gaps in the environment are detected. The planner selects the gap to the left of the frame and continues through this gap in frame $i = 610$. By frame $i = 675$, the planner has passed through this left gap and continues up the frame through another front-facing gap on its way to the goal.

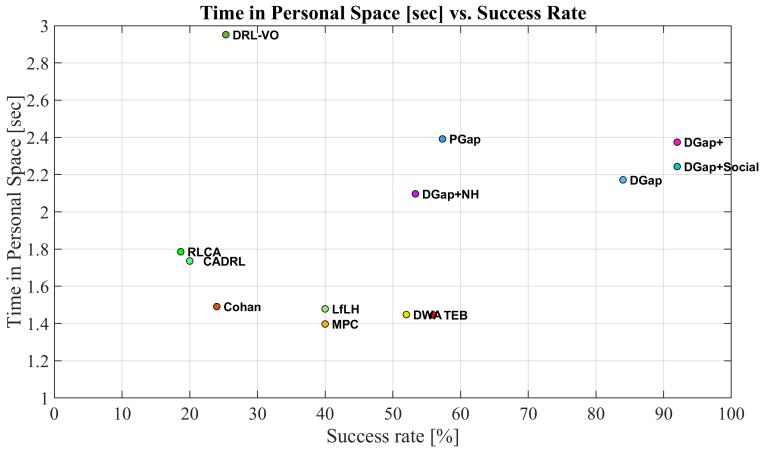


Fig. 19 Comparison of various navigation algorithms based on their time spent in personal space (in seconds) versus success rate (%).

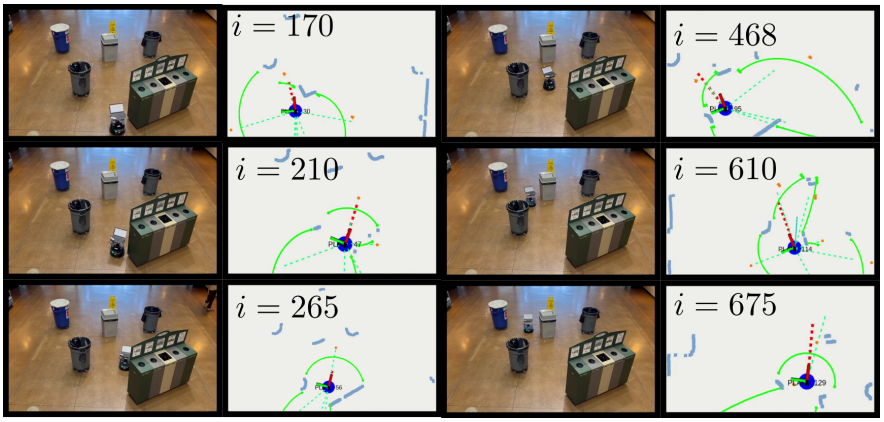


Fig. 20 Hardware experiment one: static obstacle course. Left column depicts frames from the hardware setup and right column depicts online environment visualizations from the planner. The ego-robot (starting at the bottom of the frame) must travel to the top of the frame while avoiding the array of static obstacles.

6.6.2 Test Two: Translating Agent

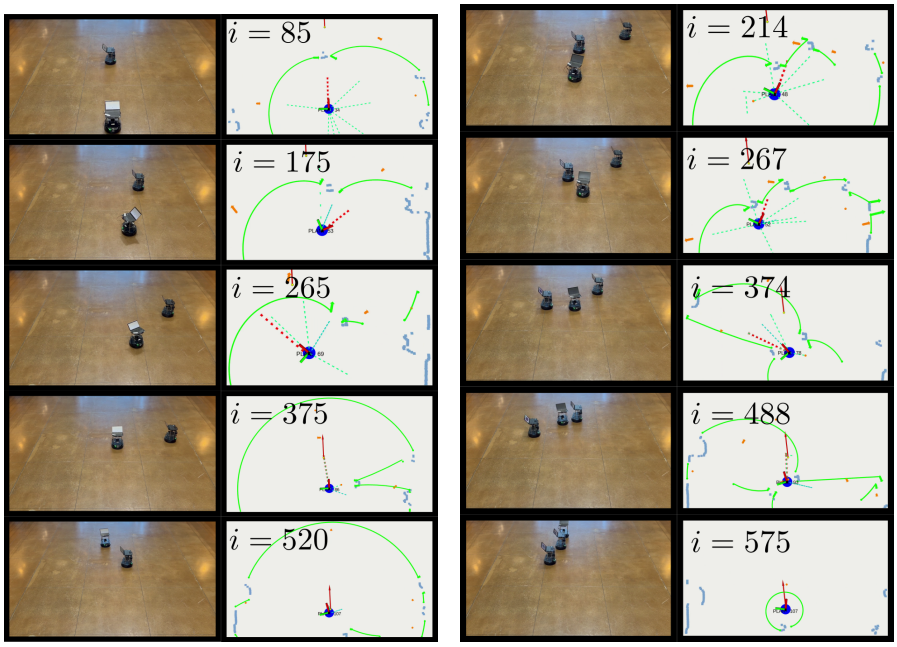
For this test, a second TurtleBot2 platform is programmed to track a straight line path between two waypoints at a constant velocity. The ego-robot must travel past this agent en route to the goal. The keyframes of this test are shown in Figure 21a.

At frame $i = 85$, the ego-robot is tracking a previously generated trajectory through the middle of the frame towards the agent. At frame $i = 175$, the agent completes the trajectory it had been tracking and selects a new trajectory that travels to the right of the frame. However, at frame $i = 265$, predicts a collision with the agent and switches to plan to the left of the agent. At frame $i = 375$, the agent reaches its right waypoint and begins to travel back to the left of the frame, but at this point the ego-robot has almost passed the agent. By frame $i = 520$, the ego-robot has passed the agent and reached the goal.

6.6.3 Test Three: Translating Gap

In this test, two agents are programmed to track a straight line path between two waypoints at a constant velocity, creating a translating gap that the ego-robot can pass through en route to the goal. Keyframes from this test are shown in Figure 21b.

At frame $i = 214$, the dynamic gap planner propagates the central gap forward in time and selects the trajectory that passes through this gap to be track. This trajectory continues to be tracked at frame $i = 267$, and at frame $i = 374$ this trajectory is completed. At this point, the planner generates trajectories through the central gap that are biased to the left of the gap. These trajectories are biased because of their



(a) Hardware experiment two: translating agent. Left column depicts frames from the hardware setup and right column depicts online environment visualizations from the planner. The ego-robot (starting at the bottom of the frame) must travel to a goal placed beyond the other TurtleBot (starting in the middle of the frame) which is translating back and forth.

(b) Hardware experiment three: translating gap. Left column depicts frames from the hardware setup and right column depicts online environment visualizations from the planner. The ego-robot (starting at the bottom of the frame) must travel to a goal placed beyond the two TurtleBots that are translating back and forth so as to form a gap between them.

Fig. 21 Hardware experiments two and three - translating scenarios.

propagated versions. At iteration $i = 488$, the ego-robot switches to a trajectory that will reach the goal, and by $i = 575$ the planner reaches the goal.

6.6.4 Test Four: Receding Corridor

In this test, the planner must maneuver the ego-robot around an agent travelling down a corridor. Keyframes from this test are shown in Figure 22. In this scenario, the corridor is wide enough for the ego-robot to travel around the agent. The planner opts to do so in frame $i = 212$ where the ego-robot begins to travel to the right of the agent as the planner select the gap to the right of the agent. During frames $i = 312$ and $i = 412$, the planner continues planning to the right of the agent. By frame $i = 512$, the ego-robot begins to cut back in front of the agent in order to reach the

goal which is in the center of the corridor. By frame $i = 645$, the ego-robot reaches the goal in front of the agent.

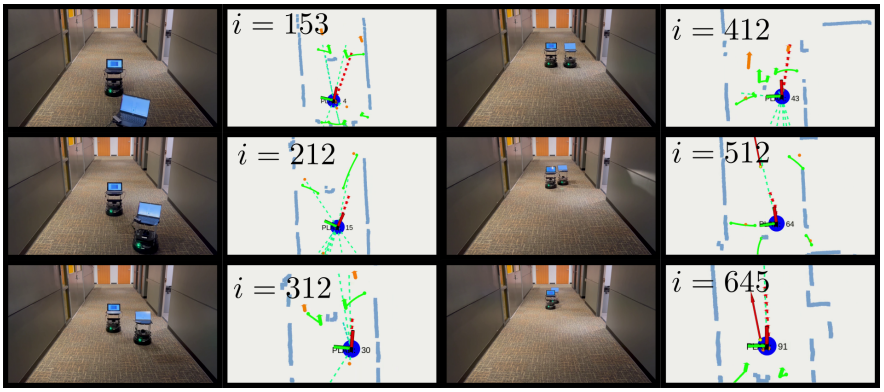


Fig. 22 Hardware experiment four: corridor - receding. Left column depicts frames from the hardware setup and right column depicts online environment visualizations from the planner. The ego-robot (starting at the bottom of the frame) must travel to the end of the corridor while the agent in front of it also moves through the corridor.

7 Conclusion

This chapter introduces a novel perception-informed gap-based planner known as dynamic gap. This planner is designed to operate in dynamic environments by tracking locally detected gaps of free space over time, propagating these gaps forward into the future to understand what subset of the free space remains feasible, and applying the PN geometric law from pursuit guidance theory to generate collision-free local trajectories under ideal conditions. Furthermore, the occupied polar regions referred to as ungaps are detected and planned within under circumstances in which no gaps exist. In this chapter, the dynamic gap planner is shown to outperform all other evaluated baselines in simulation benchmarks when tested on a holonomic robot platform. However, the planner performs worse when constrained to nonholonomic dynamics. This is largely due to the model mismatch between the holonomic model used to evaluate gap feasibility and generate local trajectories and the nonholonomic model that these trajectories are subsequently tracked on. In the future, the authors aim to integrate nonholonomic constraints directly into gap feasibility considerations and trajectory synthesis to mitigate this mismatch.

References

1. J. J. Gibson, *The Ecological Approach to Visual Perception*, 2014.
2. R. Xu, S. Feng, and P. A. Vela, "Potential Gap: A Gap-Informed Reactive Policy for Safe Hierarchical Navigation," *IEEE Robotics and Automation Letters*, 2021.
3. S. Feng, Z. Zhou, J. Smith, M. Asselmeier, Y. Zhao, and P. A. Vela, "GPF-BG: A hierarchical vision-based planning framework for safe quadrupedal navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
4. S. Feng, A. Abuaish, and P. A. Vela, "Safer Gap: A Gap-based Local Planner for Safe Navigation with Nonholonomic Mobile Robots," Mar. 2023, arXiv:2303.08243 [cs, eess].
5. H. Chen, S. Feng, Y. Zhao, C. Liu, and P. A. Vela, "Safe Hierarchical Navigation in Crowded Dynamic Uncertain Environments," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022.
6. E. C. Contarli and V. Sezer, "Predictive Follow the Gap Method for Dynamic Obstacle Avoidance," in *2024 13th International Workshop on Robot Motion and Control*, 2024.
7. M. Asselmeier, D. Ahuja, A. Zaro, Y. Zhao, and P. A. Vela, "Dynamic Gap: Safe Gap-based Navigation in Dynamic Environments," 2025.
8. L. Kästner, T. Buiyan, X. Zhao, L. Jiao, Z. Shen, and J. Lambrecht, "Arena-Rosnav: Towards Deployment of Deep-Reinforcement-Learning-Based Obstacle Avoidance into Conventional Autonomous Navigation Systems," Sep. 2021.
9. L. Kästner, T. Bhuiyan, T. A. Le, E. Treis, J. Cox, B. Meinardus, J. Kmiecik, R. Carstens, D. Pichel, B. Fatloun, N. Khorsandi, and J. Lambrecht, "Arena-Bench: A Benchmarking Suite for Obstacle Avoidance Approaches in Highly Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, Oct. 2022.
10. L. Kästner, R. Carstens, H. Zeng, J. Kmiecik, T. Bhuiyan, N. Khorsandi, V. Shcherbyna, and J. Lambrecht, "Arena-Rosnav 2.0: A Development and Benchmarking Platform for Robot Navigation in Highly Dynamic Environments," Jul. 2023.
11. L. Kästner, V. Shcherbyna, H. Zeng, T. A. Le, M. H.-K. Schreff, H. Osmaev, N. T. Tran, D. Diaz, J. Golebiowski, H. Soh, and J. Lambrecht, "Arena 3.0: Advancing Social Navigation in Collaborative and Highly Dynamic Environments," Jun. 2024.
12. D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2D lidar," *The International Journal of Robotics Research*, vol. 34, no. 7, 2015.
13. S. Vaskov, S. Kousik, H. Larson, F. Bu, J. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, "Towards Provably Not-at-Fault Control of Autonomous Robots in Arbitrary Dynamic Environments," Feb. 2019.
14. V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek, "Reactive Semantic Planning in Unexplored Semantic Environments Using Deep Perceptual Feedback," *IEEE Robotics and Automation Letters*, no. 3, Jul. 2020, conference Name: IEEE Robotics and Automation Letters.
15. C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control," in *2015 European Control Conference (ECC)*, 2015.
16. ——, "Track and include dynamic obstacles via costmap_converter," 2018. [Online]. Available: http://wiki.ros.org/teb_local_planner
17. T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data," Jan. 2021.
18. P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, Feb. 1992.
19. B. Mishra, D. Calvert, S. Bertrand, J. Pratt, H. E. Sevil, and R. Griffin, "Efficient Terrain Map Using Planar Regions for Footstep Planning on Humanoid Robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024.
20. H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

21. V. Guizilini, R. Senanayake, and F. Ramos, "Dynamic Hilbert Maps: Real-Time Occupancy Predictions in Changing Environments," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4091–4097.
22. Z. Xu, X. Zhan, B. Chen, Y. Xiu, C. Yang, and K. Shimada, "A real-time dynamic obstacle tracking and mapping system for UAV navigation and collision avoidance with an RGB-D camera," in *2023 IEEE International Conference on Robotics and Automation*, 2023.
23. R. Senanayake and F. Ramos, "Bayesian Hilbert Maps for Dynamic Continuous Occupancy Mapping," in *Proceedings of the 1st Annual Conference on Robot Learning*, Oct. 2017.
24. J. Poschmann, T. Pfeifer, and P. Protzel, "Factor Graph based 3D Multi-Object Tracking in Point Clouds," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020.
25. M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-Free MPC for Legged Robots in Static and Dynamic Scenes," Mar. 2021, arXiv:2103.13987.
26. R. Senanayake, L. Ott, S. O' Callaghan, and F. T. Ramos, "Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.
27. P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, 1968, publisher: Institute of Electrical and Electronics Engineers (IEEE).
28. A. Stentz, "The D*Algorithm for Real-Time Planning of Optimal Traverses."
29. S. K. a. M. Likhachev, "D* Lite."
30. L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, no. 4, 1996.
31. S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.
32. S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," 2011.
33. M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-PRM: Temporal Probabilistic Roadmap for Path Planning in Dynamic Environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
34. M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for Rapid Replanning in Dynamic Environments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. Rome, Italy: IEEE, 2007.
35. 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.
36. M. Missura and M. Bennewitz, "Predictive Collision Avoidance for the Dynamic Window Approach," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.
37. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993.
38. ——, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, 1998.
39. J. van den Berg, M. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008.
40. J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," in *Robotics Research*, B. Siciliano, O. Khatib, F. Groen, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
41. S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, 2020.
42. S. Vaskov, U. Sharma, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, "Guaranteed Safe Reachability-based Trajectory Design for a High-Fidelity Model of an Autonomous Passenger Vehicle," 2019.

43. B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," 2020.
44. M. Vahs, R. I. C. Muchacho, F. T. Pokorny, and J. Tumova, "Forward Invariance in Trajectory Spaces for Safety-critical Control," 2024.
45. 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 *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.
46. O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-Driven Parallel Trajectory Optimization in Dynamic Environments," *IEEE Transactions on Robotics*, 2025.
47. A. S. Lafmejani, S. Berman, and G. Fainekos, "NMPC-LBF: Nonlinear MPC with Learned Barrier Function for Decentralized Safe Navigation of Multiple Robots in Unknown Environments," 2022.
48. K. Long, K. Tran, M. Leok, and N. Atanasov, "Safe Stabilizing Control for Polygonal Robots in Dynamic Elliptical Environments," 2024.
49. O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to Train Your Neural Control Barrier Function: Learning Safety Filters for Complex Input-Constrained Systems," 2023.
50. D. R. Agrawal, R. Chen, and D. Panagou, "gatekeeper: Online Safety Verification and Control for Nonlinear Systems in Dynamic Environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.
51. L. Lützow, Y. Meng, A. C. Armijos, and C. Fan, "Density Planner: Minimizing Collision Risk in Motion Planning with Dynamic Obstacles using Density-based Reachability," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
52. P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," 2018.
53. M. Everett, Y. F. Chen, and J. P. How, "Collision Avoidance in Pedestrian-Rich Environments with Deep Reinforcement Learning," *IEEE Access*, 2021.
54. Z. Xu, G. Dhamankar, A. Nair, X. Xiao, G. Warnell, B. Liu, Z. Wang, and P. Stone, "APPLR: Adaptive Planner Parameter Learning from Reinforcement," 2020.
55. Z. Xie and P. Dames, "DRL-VO: Learning to Navigate Through Crowded Dynamic Scenes Using Velocity Obstacles," *IEEE Transactions on Robotics*, 2023, conference Name: IEEE Transactions on Robotics.
56. Z. Wang, X. Xiao, A. J. Nettekoven, K. Umasankar, A. Singh, S. Bommakanti, U. Topcu, and P. Stone, "From Agile Ground to Aerial Navigation: Learning from Learned Hallucination," 2021.
57. X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: a Survey," 2022.
58. Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking Reinforcement Learning Techniques for Autonomous Navigation," 2023.
59. X. Xiao, Z. Xu, A. Datar, G. Warnell, P. Stone, J. J. Damanik, J. Jung, C. A. Deresa, T. D. Huy, C. Jinyu, C. Yichen, J. A. Cahyono, J. Wu, L. Mo, M. Lv, B. Lan, Q. Meng, W. Tao, and L. Cheng, "Autonomous Ground Navigation in Highly Constrained Spaces: Lessons Learned From the Third BARN Challenge at ICRA 2024 [Competitions]," *IEEE Robotics & Automation Magazine*, 2024.
60. A. Nair, F. Jiang, K. Hou, Z. Xu, S. Li, X. Xiao, and P. Stone, "DynaBARN: Benchmarking Metric Ground Navigation in Dynamic Environments," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2022.
61. P. T. Singamaneni, P. Bachiller-Burgos, L. J. Manso, A. Garrell, A. Sanfeliu, A. Spalanzani, and R. Alami, "A survey on socially aware robot navigation: Taxonomy and future challenges," *Int. J. Rob. Res.*, vol. 43, no. 10, pp. 1533–1572, Feb. 2024.
62. A. H. Raj, Z. Hu, H. Karnan, R. Chandra, A. Payandeh, and L. Mao, "Rethinking Social Robot Navigation: Leveraging the Best of Two Worlds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

63. "Demonstrating Arena 3.0: Advancing Social Navigation in Collaborative and Highly Dynamic Environm... | Robotics: Science and Systems," 2025.
64. D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using RRT," in *2017 IEEE International Conference on Systems, Man, and Cybernetics*, 2017.
65. J. S. Smith and P. Vela, "PiPS: Planning in perception space," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6204–6209.
66. J. S. Smith, S. Feng, F. Lyu, and P. A. Vela, "Real-Time Egocentric Navigation Using 3D Sensing," in *Machine Vision and Navigation*, 2020, pp. 431–484.
67. J. S. Smith, R. Xu, and P. Vela, "egoTEB: Egocentric, Perception Space Navigation Using Timed-Elastic-Bands," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2703–2709.
68. V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, 2012.
69. M. Mujahed, D. Fischer, and B. Mertsching, "Admissible gap navigation: A new collision avoidance approach," *Robotics and Autonomous Systems*, 2018.
70. ——, "Safe Gap based (SG) reactive navigation for mobile robots," in *2013 European Conference on Mobile Robots*, 2013, pp. 325–330.
71. M. Mujahed and B. Mertsching, "A new gap-based collision avoidance method for mobile robots," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 220–226.
72. M. Mujahed, H. Jaddu, D. Fischer, and B. Mertsching, "Tangential Closest Gap based (TCG) reactive obstacle avoidance navigation for cluttered environments," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013.
73. M. Mujahed and B. Mertsching, "The admissible gap (AG) method for reactive collision avoidance," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
74. M. Mujahad, D. Fischer, B. Mertsching, and H. Jaddu, "Closest Gap based (CG) reactive obstacle avoidance Navigation for highly cluttered environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.
75. Z. Ullah, X. Chen, S. Gou, Y. Xu, and M. Salam, "FNUG: Imperfect Mazes Traversal Based on Detecting and Following the Nearest-to-Final-Goal and Unvisited Gaps," *IEEE Robotics and Automation Letters*, 2022, conference Name: IEEE Robotics and Automation Letters.
76. M. Demir and V. Sezer, "Improved Follow the Gap Method for obstacle avoidance," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017.
77. O. d. Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-Driven Parallel Trajectory Optimization in Dynamic Environments," 2024.
78. S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, 2012.
79. F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, no. 14, 2016.
80. M. Missura, A. Roychoudhury, and M. Bennewitz, "Polygonal Perception for Mobile Robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
81. N. Shneydor, *Missile Guidance and Pursuit: Kinematics, Dynamics and Control (1st ed.)*. Horwood Series in Engineering Science. Chichester, UK: Horwood Publishing, 1998.
82. "Implementation of the Pure Pursuit Path Tracking Algorithm." [Online]. Available: https://www.ri.cmu.edu/publications/implementation-of-the-pure-pursuit-path-tracking-algorithm/
83. L. Wellhausen and M. Hutter, "ArtPlanner: Robust Legged Robot Navigation in the Field," *Field Robotics*, vol. 3, no. 1, pp. 413–434, 2023.
84. A. Bernhart, "Polygons of pursuit," *Scripta Mathematica*, vol. 24, Jan. 1959.
85. ——, "Curves of general pursuit," *Scripta Mathematica*, vol. 24, Jan. 1959.
86. A. Bruckstein, "Why the ant trails look so straight and nice," *The Mathematical Intelligencer*, vol. 15, pp. 59–62, Jan. 1993.
87. V. Rajasekhar and A. G. Sreenatha, "Fuzzy logic implementation of proportional navigation guidance," *Acta Astronautica*, 2000.

88. Y. Ulybyshev, "Terminal Guidance Law Based on Proportional Navigation," *Journal of Guidance, Control, and Dynamics*, 2005.
89. "move_base Package." [Online]. Available: https://wiki.ros.org/move_base
90. P. Teja Singamaneni, A. Favier, and R. Alami, "Human-Aware Navigation Planner for Diverse Human-Robot Interaction Contexts," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.
91. C. Rösmann, A. Makarow, and T. Bertram, "Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups," in *2021 European Control Conference (ECC)*, 2021.
92. R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, 2008.
93. M. Everett, Y. F. Chen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," 2018.