Dynamic Gap: Safe Gap-based Navigation in Dynamic Environments

Max Asselmeier[1], Dhruv Ahuja[2], Abdel Zaro[3], Ahmad Abuaish[1], Ye Zhao[1], and Patricio A. Vela[1]

***Abstract*— This paper extends the family of gap-based local planners to unknown dynamic environments through generating provably collision-free properties for hierarchical navigation systems. Existing perception-informed local planners that operate in dynamic environments rely on emergent or empirical robustness for collision avoidance as opposed to performing formal analysis of dynamic obstacles. In addition to this, the obstacle tracking that is performed in these existent planners is often achieved with respect to a global inertial frame, subjecting such tracking estimates to transformation errors from odometry drift. The proposed local planner, *dynamic gap*, shifts the tracking paradigm to modeling how the free space, represented as gaps, evolves over time. Gap crossing and closing conditions are developed to aid in determining the feasibility of passage through gaps, and a breadth of simulation benchmarking is performed against other navigation planners in the literature where the proposed dynamic gap planner achieves the highest success rate out of all planners tested in all environments.**

I. Introduction

For mobile robots, collision avoidance is typically handled by a local planner which utilizes sensory information to evade obstacles. One family of local planners is the gap-based planner [1], which identifies passable regions, or "gaps", and synthesizes motion commands through them. With this emphasis on free space, gap-based planners are an approach based on the *affordances* of the environment [2], and they have shown great promise with capabilities of respecting dynamic, visual, and geometric constraints [3]–[7] as well as generating provably collision-free trajectories [8].

Despite this success, gap-based planners have only been extended to handling dynamic obstacle avoidance very recently [9], a challenge that accurately reflects the unknown, changing environments found in the real world. Reactive planners designed for static environments are often deployed in dynamic environments, relying on sufficient runtimes to adapt to the evolving environment. However, explicitly accounting for dynamic obstacles during planning allows for more assured collision avoidance behaviors.

This paper details an extension to prior work on the *potential gap* planner [8] involving augmentations to the planning framework to extend formal guarantees for safe navigation to dynamic environments under particular assumptions. Additional modules are also included to enable a more robust translation of planning success to non-ideal environments. This extension is referred to hereafter as *dynamic gap* and can be visualized in Figure 1. Egocentric free space tracking is integrated to predict how gaps will evolve over time, and guidance law-based policies are employed to synthesize collision-free trajectories that pass through moving gaps under ideal conditions. Additional modules are then applied to robustify planning performance in non-ideal conditions.

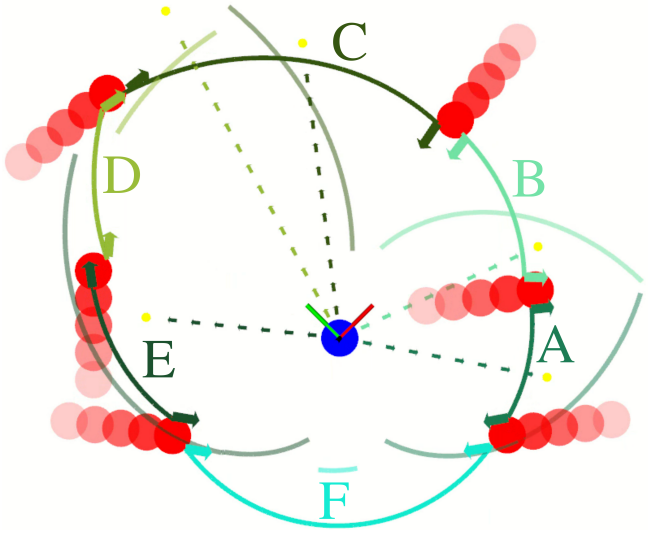

Fig. 1: Visualization of gaps and trajectories generated by dynamic gap. The central blue circle depicts the ego-robot while the red circles depict dynamic agents. The bold colored arcs labeled A-F are the instantaneous set of detected gaps and the transparent arcs are the predicted gaps obtained by propagating the gap dynamics models, shown as arrows, forward in time. Dashed lines are the candidate trajectories synthesized towards the gap goals shown in yellow. Gap F is predicted to close before the ego-robot can pass through, so it is deemed infeasible and not used during trajectory synthesis.

Our main contributions are summarized below:

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 gap-based planning to aid in dynamic gap propagation and feasibility analysis
3) Providing a proof of collision-free dynamic gap passage under ideal conditions along with supplementary safety modules for non-ideal conditions
4) Benchmarking against state-of-the-art local planners in the Arena-Rosnav simulation environment

This planner is open-sourced[1] in Arena-Rosnav [10]–[13] and available to test. The information flow for the planner can be seen in Figure 2.

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

[1]M. Asselmeier, A. Abuaish, Y. Zhao, and P.A. Vela are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308, USA. mass@gatech.edu

[2]D. Ahuja is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308, USA.

[3]A. Zaro is with the Department of Mechanical Engineering, University of California, Berkeley, Berkeley, CA, 94720, USA.

The work of Max Asselmeier is supported by the National Science Foundation 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.

II. Related Work

A. Perception Space and Gap-based Navigation

Most motion planners [14]–[16] opt to plan using Cartesian world frame environment representations such as cost 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. All local planning steps downstream are then cast as ego-centric decision making processes.

Gap-based planners [1], [3]–[7], [9], [17]–[22] 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 [23]–[25]. Some attention has been given to gaps in dynamic environments [9], [22], 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.

B. Guidance Laws

Guidance laws [26] 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 [27]–[29].

Among the more established guidance laws, the two geometrical rules of pure pursuit and parallel navigation are the most popular. The pure pursuit 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. Pure pursuit has seen a great deal of attention due to its simplicity [30]–[32], 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 parallel navigation, or constant bearing, rule [33], [34] 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, parallel navigation is the optimal guidance law which yields a minimum intercept time.

III. Dynamic Gap Local Planning Module

A. Gap Detection, Association, and Estimation

1) Gap Detection and Simplification

As input to the planner, we assume access to a 360° laser scan $\mathcal{L}$. Gap detection involves the parsing of this scan $\mathcal{L}$ to obtain a set of detected gaps $\mathcal{G}^{\text{det}}$ that describe the instantaneous free space of the local environment, further details regarding how this gap detection policy is defined can be found at [8]. Once this set of gaps has been extracted from the laser scan, an additional pass through $\mathcal{G}^{\text{det}}$ is performed to remove any redundant gaps and potentially merge adjacent gaps together, yielding a set of simplified gaps $\mathcal{G}^{\text{simp}}$.

2) Gap Association

The set of simplified gaps $\mathcal{G}^{\text{simp}}$ captures the immediate free space, but we seek to understand how this free space will evolve across our local planning horizon. Therefore, additional steps must be taken to track gap points over time.

At this stage, each gap $g \in \mathcal{G}^{\text{simp}}$ is characterized by a left gap point $\mathbf{p}_l = [x_l, y_l]^T$ and a right gap point $\mathbf{p}_r = [x_r, y_r]^T$. Association is performed on the set of points from the simplified gaps,

$$P_{\text{simp}} = \{\mathbf{p}_l^0, \mathbf{p}_r^0, ..., \mathbf{p}_l^N, \mathbf{p}_r^N\}.$$

The association step is represented as a rectangular assignment 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 Hungarian Algorithm [35], producing a minimum distance mapping between P_{t-1}^{simp} and P_t^{simp}. If the distance between two associated points exceeds a threshold τ_{assoc}, then that point-to-point association is discarded.

3) Gap Estimation

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

The state vector is defined as

$$\mathbf{X} = \begin{bmatrix} \mathbf{p}_{s/e} \\ \mathbf{v}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_s - \mathbf{p}_e \\ \mathbf{v}_s - \mathbf{v}_e \end{bmatrix}, \tag{1}$$

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

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{p}}_{s/e} \\ \dot{\mathbf{v}}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{s/e} - \omega_e \times \mathbf{p}_{s/e} \\ \mathbf{a}_{s/e} - \omega_e \times \mathbf{v}_{s/e} \end{bmatrix}, \tag{2}$$

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. We make a constant velocity assumption on the gap points, which then simplifies Equation 2 to

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{p}}_{s/e} \\ \dot{\mathbf{v}}_{s/e} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{s/e} - \omega_e \times \mathbf{p}_{s/e} \\ -\mathbf{a}_e - \omega_e \times \mathbf{v}_{s/e} \end{bmatrix}. \tag{3}$$

This model is estimated with an extended Kalman filter given the nonlinear cross-product in Equation 3.

B. Gap Propagation

Within static environments, gap feasibility solely depends on the geometric condition of whether or not the gap is wide enough to fit the robot. For dynamic environments, both geometric and kinematic considerations must be made to determine whether or not the robot can pass through the

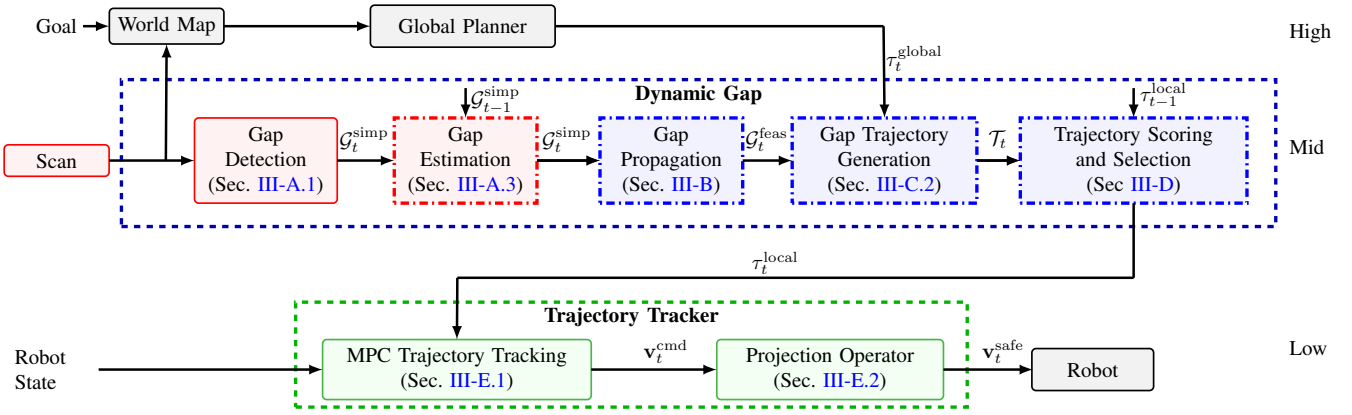

Fig. 2: Overall workflow for the proposed navigation framework. Red blocks correspond to perceptual modules run at the rate of the laser scanner. Blue blocks are the core planning loop, and green blocks are the lower-level trajectory tracking routine. Dashed outlines represent core contributions.

gap before the gap shuts. By pruning away infeasible gaps, this step not only conserves energy of the robot, but it also avoids potentially dangerous gaps through which it would be difficult, if not impossible, for the robot to pass.

To understand the behavior of gaps over the time horizon for which planning will be performed, gap models (Eq. 3) 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 $\mathbf{v}_s$. Gap propagation can terminate prematurely in two ways:

1) The gap closes, meaning if a gap that is shrinking over time reaches an angular span of 0 radians
2) The gap overlaps, meaning if a gap that is expanding over time passes beyond an angular span of 2π radians

At each time step t during integration, crossing and overlapping conditions are checked for each gap. These conditions will now be further detailed.

1) Crossing condition

First, we define the unit norm bearing vector for the gap side s at time t as $\boldsymbol{\eta}_s^t := \mathbf{p}_s^t / \|\mathbf{p}_s^t\|$. Then, we can calculate the clockwise angular gap span as

$$\alpha^t = \arctan\left(\frac{\boldsymbol{\eta}_l^t \cdot \boldsymbol{\eta}_r^t}{\boldsymbol{\eta}_l^t \times \boldsymbol{\eta}_r^t}\right), \tag{4}$$

which we use to define the bearing of the gap center as $\beta_c^t := \beta_l^t - \alpha^t/2$. This in turn allows us to obtain the unit norm bearing vector for the gap center, $\boldsymbol{\eta}_c^t = [\cos(\beta_c^t), \sin(\beta_c^t)]^T$. The left and right gap points have crossed each other if the two following conditions are true:

$$(\boldsymbol{\eta}_l^t \cdot \boldsymbol{\eta}_c^{t-1} > 0) \wedge (\boldsymbol{\eta}_r^t \cdot \boldsymbol{\eta}_c^{t-1} > 0), \tag{5}$$

meaning that the gap points form a convex polar arc, and

$$\alpha^t > \pi, \tag{6}$$

meaning that the clockwise angular span from the left gap point to the right gap point exceeds π radians. This scenario is visualized in part (a) of Figure 3.

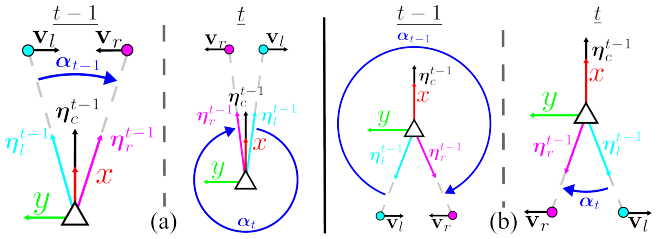

Fig. 3: (a) Crossing condition for a gap, where between timesteps $t-1$ and t, the gap's angular span reaches 0 radians. (b) Overlapping condition for a gap, where the gap's angular span reaches 2π radians.

2) Overlapping condition

A set of gap points have overlapped if the exact negation of the crossing conditions are met, meaning that

$$(\boldsymbol{\eta}_l^t \cdot \boldsymbol{\eta}_c^{t-1} < 0) \wedge (\boldsymbol{\eta}_r^t \cdot \boldsymbol{\eta}_c^{t-1} < 0), \tag{7}$$

and

$$\alpha^t < \pi. \tag{8}$$

This scenario is visualized in part (b) of Figure 3. From this gap propagation step, we obtain a gap lifespan t_f.

C. Gap Feasibility Analysis

Now that we have determined how long each gap will exist in the local environment for, we must determine if it is kinematically feasible for the ego-robot to pass through such gaps before they cease to exist.

1) Guidance Law Analysis

In this section, we outline the guidance law-based trajectory generation scheme that dynamic gap employs in order to determine if a gap is kinematically feasible.

We employ the parallel navigation geometric rule [26], for which we assume a constant gap goal point speed v_g as well as a constant ego-robot speed v_e. This policy then dictates the bearing θ_e in which v_e will be applied. With relation to Figure 4, the parallel navigation policy is defined as $\dot{\beta}_g = 0, \dot{r}_g < 0$, where

$$\begin{bmatrix} \dot{\beta}_g \\ \dot{r}_g \end{bmatrix} = \begin{bmatrix} \frac{v_g \sin(\theta_g) - v_e \sin(\theta_e)}{r_g} \\ v_g \cos(\theta_g) - v_e \cos(\theta_e) \end{bmatrix}, \tag{9}$$

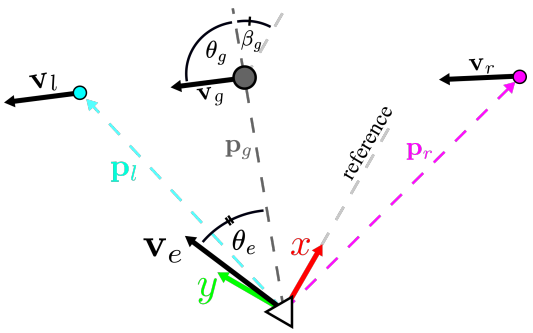

Fig. 4: Diagram for guidance law notation.

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

$$v_g \sin(\theta_g) = v_e \sin(\theta_e). \tag{10}$$

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

$$v_e \cos(\theta_e) < v_g \cos(\theta_g). \tag{11}$$

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

$$\sin(\theta_e) = \frac{\sin(\theta_g)}{K}, \tag{12}$$

and

$$\cos(\theta_e) > \frac{\cos(\theta_g)}{K}. \tag{13}$$

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

$$t_{\text{intercept}} = \frac{r_g^0}{v_g} \cdot \frac{1}{K \cdot \cos(\theta_e) - \cos(\theta_g)}, \tag{14}$$

where r_g^0 is equal to r_g at $t = 0$. If these conditions can not be satisfied, then the given gap 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 is also deemed infeasible and discarded. Gaps that satisfy this condition are added to $\mathcal{G}^{\text{feas}}$.

2) Collision-free Trajectory Generation

For each feasible gap $g \in \mathcal{G}^{\text{feas}}$, a collision-free trajectory τ is synthesized along the bearing θ_e for at least $t_{\text{intercept}}$ seconds to form the set of trajectories $\mathcal{T}$.

3) Proof of Collision-Free Passage

We aim to prove that performing the policy of parallel navigation towards the gap goal point $\mathbf{p}_g$ yields collision-free gap passage. Assumptions are (1) a first-order, point-mass, constant velocity, holonomic ego-robot, (2) constant velocity gap points, and (3) an isolated gap: no other gaps will enter the gap in focus during the local time horizon. 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,

$$\begin{aligned} \mathbf{p}_g &= \kappa \mathbf{p}_l + (1-\kappa)\mathbf{p}_r, \\ \mathbf{v}_g &= \kappa \mathbf{v}_l + (1-\kappa)\mathbf{v}_r, \qquad \kappa \in [0,1]. \end{aligned} \tag{15}$$

It follows that

$$\beta = \arctan(\mathbf{p}_g) = \arctan(\kappa \mathbf{p}_l + (1-\kappa)\mathbf{p}_r). \tag{16}$$

Without loss of generality (rotating the robot-centric 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

$$\gamma_g = \arctan(\mathbf{v}_g) = \arctan(\kappa \mathbf{v}_l + (1-\kappa)\mathbf{v}_r), \tag{17}$$

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

$$\gamma = \beta + \theta, \tag{18}$$

it follows that $\theta_g \in [\theta_r, \theta_l]$. From Equation 12,

$$\theta_e = \arcsin\left(\frac{\sin\theta_g}{K}\right), \tag{19}$$

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

D. Trajectory 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 is

$$\mathcal{J}(\mathcal{T}) = w\|\mathbf{p}[N] - \mathbf{p}^*\| + \sum_{k=0}^{N-1} C(d(\mathbf{p}[k]; \mathcal{L}[k])) \tag{20}$$

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

$$C(d) = \begin{cases} \infty, & d \le r_{\text{ins}} \\ c_{\text{obs}} e^{-w_2(d - r_{\text{ins}})}, & r_{\text{ins}} < d < r_{\max} \\ 0, & r_{\max} \le d \end{cases}, \tag{21}$$

where $c_{\text{obs}}, w_2 \in \mathbb{R}^+$ are weighting factors, r_{ins} is the inscribed radius of the ego-robot, and $r_{\max}$ is the maximum distance that we penalize.

The trajectory cost formulation is adapted from [8], with one key difference: pose-wise scoring requires a laser scan for each timestep t along the trajectory. We do not have direct access to this information in practice, so we propagate gaps forward in time and recover propagated scans.

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

E. Trajectory Tracking

Trajectories are generated for the ideal unit single-integrator system. However, the simulation environment Arena-Rosnav adopts the non-ideal double-integrator system. The input to the system is the velocity command with a constraint on the command rate, i.e., acceleration constraints. Although reference acceleration commands can be extracted from the trajectory to facilitate trajectory tracking using feed-forward control architecture, an MPC module is employed to robustly track the reference trajectory.

1) MPC Trajectory Tracking

For the double integrator system $\mathcal{D}(\mathbf{x}, \mathbf{u})$, the state and control are $\mathbf{x} = [p_x, p_y, v_x, v_y]^T$ and $\mathbf{u} = [a_x, a_y]^T$ respectively. The state and control are uniformly discretized with a time step $dt = 0.5$ over a maximum of $N = 10$ timesteps.

The MPC optimization program is given in Equation 22.

$$\min_{\mathbf{X},\mathbf{U}} \quad \|\mathbf{x}[N]_{\{1:2\}} - \mathbf{p}[N]^{\text{des}}\|^2_{Q_f} + \sum_{k=0}^{N-1} \left(\|\mathbf{x}[k]_{\{1:2\}} - \mathbf{p}[k]^{\text{des}}\|^2_Q + \|\mathbf{u}[k]\|^2_R \right) \tag{22a}$$

s.t.

$$\text{(Dynamics)} \quad \mathbf{x}[k+1] = \mathcal{D}(\mathbf{x}[k], \mathbf{u}[k]) \quad \forall k \in \{0, \cdots, N-1\} \tag{22b}$$

$$\text{(Initial state)} \quad x[0] = x_0 \tag{22c}$$

$$\text{(ZBF)} \quad A\mathbf{x}[j] \leq b \tag{22d}$$

$$\text{(Velocity)} \quad |\mathbf{x}[j]_{\{3,4\}}| \leq v_{max} \tag{22e}$$

$$\text{(Acceleration)} \quad |u[j]| \leq a_{max} \quad \forall j \in \{0, \cdots, N\} \tag{22f}$$

where the decision variables $\mathbf{X} = [\mathbf{x}[0], \cdots, \mathbf{x}[N]]^T$ and $\mathbf{U} = [\mathbf{u}[0], \cdots, \mathbf{u}[N-1]]^T$, the desired poses at discrete time k from the best trajectory τ^t_{best} are denoted by $\mathbf{p}[k]^{des} \in \mathbb{R}^2$, and the reduced projection of the first two elements of state is denoted by $x[k]_{\{1,2\}}$. The zeroing barrier function (ZBF) constraint ensures that control commands lie within the interval defined in Section III-C.3 for collision-free gap passage. The optimization problem (22) is solved using quadratic programming on MATLAB with an average compute time of 6.7 milliseconds, and the MATLAB-ROS Toolbox [36] was then used to communicate the MPC output to the system.

2) 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 from [8].

IV. Experimental Results

This planner is implemented as a C++ ROS node through the `move_base` package [37]. All benchmarks are run on a Dell Precision 3660 Tower with an Intel i9-12900K CPU with 24 cores. In the factory environment for Experiment 2 in Section $IV - B$, the dynamic gap planner runs at ≈ 50 Hz with an average of ≈ 4.5 gaps per planning loop.

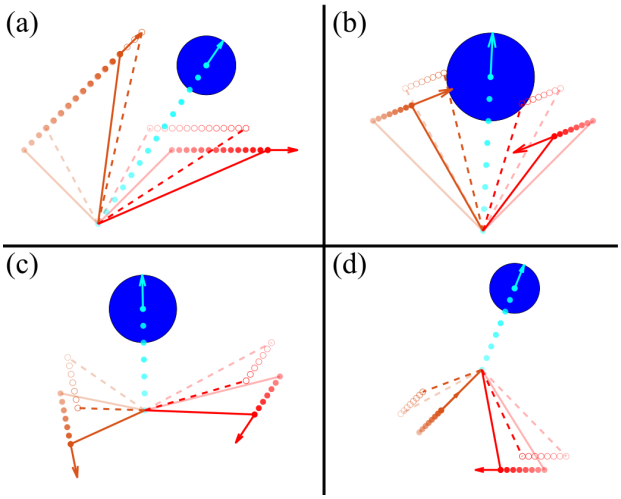

Fig. 5: Visualization of four Monte Carlo variations of gaps from Experiment 1. Orange points and lines represent the left side of the gap while red points and lines represent the right side. Solid points and lines represent the original gap geometry whereas hollow points and dashed lines correspond 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.

A. Experiment 1: Assumption-adhering Environments

First, we experimentally validate the proof of safe gap passage under ideal conditions. A single gap is randomly generated and the parallel navigation policy is employed to generate a trajectory through the given gap. 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 artificially inflating gap points inwards during planning. A subset of gaps are shown in Figure 5.

For this experiment, $10,000$ trials were run: $6,987$ trials end in the robot successfully passing through the gap, $2,668$ 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 given its finite radius. No collisions occurred during any trials.

B. Experiment 2: Assumption-violating Environments

To contextualize dynamic gap and understand the performance gap between ideal and non-ideal conditions, the planner is integrated into the Arena-Rosnav [10] benchmarking environment and compared against other state-of-the-art local planners. In this experiment, the classical motion planners DWA [38], TEB [15], and MPC [39] are tested along with the learned planners RLCA [40] and Trail [41]. Potential gap [8], the predecessor to dynamic gap, is also included.

Three environments are used, shown in column (a) of Figure 6, referred to as empty, factory, and hospital. With these environments, we aimed to build a gradient of increasing environment structure to evaluate each planner's ability to not only navigate dynamic obstacles, but also the static, non-trivial structure of these worlds.

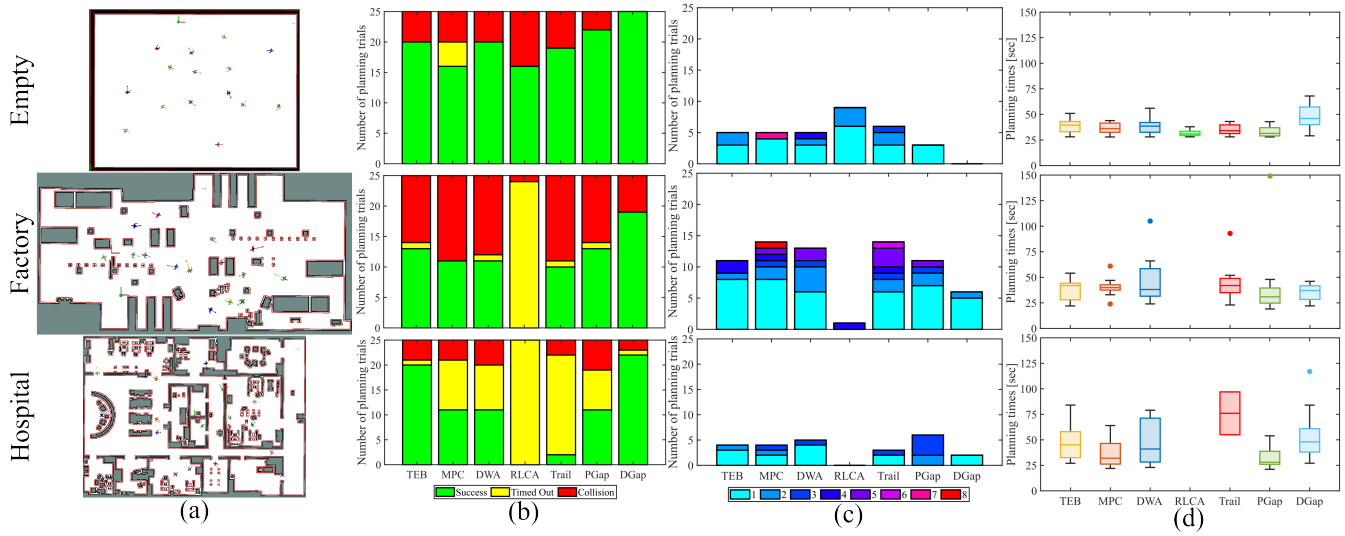

Fig. 6: . Benchmarking results from Arena-Rosnav. (a) Visualization of map with ego-robot and agents, (b) planning trial outcomes, (c) frequency of number of collisions per trial, and (d) average planning time per trial.

Within each environment, 15 dynamic agents are placed which maneuver between two manually defined waypoints within the environment. For each planner/environment combination, 25 trials are run, and results are reported by environment in Figure 6. For all trials, planners are given three minutes to reach the goal.

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. In this setting, the shortest path from start to goal is simply a straight line. Due to the lack of complexity in this setting, the variance in planning times is low, with most planners being able to reach the goal within $30 - 40$ seconds.

All other benchmarks perform fairly well in this setting, with success rates of $60 - 80\%$ across the board. Although, the set of classical planners, namely TEB and DWA, do perform slightly better than the learned planners. For the RLCA benchmark, the planner tended to come to a stop when the robot encroached upon obstacles, relying on non-adversarial motions of nearby agents to avoid collisions.

The ability of dynamic gap to predict the feasibility of local gaps before committing to them proved paramount in avoiding collisions in this environment. The proposed planner suffered no collisions over its trials, though this "patience" did result in longer planning times overall.

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 poles.

Here, the larger regions of free space allow for multiple agents to pass through the same part of the environment at one time. Because of this, more collisions are observed in this environment than the empty environment. General success rates for the benchmarks drops to roughly 50%, with the exception of RLCA failing every trial and dynamic gap registering an above average success rate of 72%. RLCA's performance can be explained in part by its neural network composition: inputs to this model are a sliding window of laser scan inputs, the relative goal position, and the robot's current state. While the scans provide a local environment representation, the model largely opts to drive in a straight line towards the goal. When structured obstacles such as walls are positioned between the robot and the goal, the planner struggles with maneuvering around such regions.

Hospital: The hospital environment exhibits many smaller rooms connected through tighter passageways and corridors. The primary failure mode observed in this setting was the benchmarks exceeding the allotted time limit of five minutes rather than colliding with obstacles in the environment.

As can be seen in column (c), far 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. Given that this environment is more sprawling, more variance is seen in planning times.

MPC and DWA stalled out on several trials when the robot was attempting to pass through thin entryways between rooms, eventually leading to a time out. Both TEB and dynamic gap performed well in this setting with success rates of 80% and 88%, respectively. The Trail benchmark did exhibit smoother overall trajectories in this environment, though the planner preferred to take wider turns around corners, often times overshooting and running into walls.

V. Conclusion

In this work, the perception-informed gap-based planning paradigm is extended to dynamic environments. Gap dynamics models are estimated and propagated forward in time to evaluate the feasibility of gap passage, and guidance laws are employed to generate collision-free trajectories. Targeted modules including scan propagation, MPC trajectory tracking, and projection operator control modifications help to bridge the gap between single gap safety guarantees and multi-gap real world settings. In the future, the authors aim to extended these safety guarantees to the nonholonomic case and enable gap propagation to account for the potential creation of gaps over the local planning horizon.

REFERENCES

[1] 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*, Oct. 2010.

[2] J. J. Gibson, *The Ecological Approach to Visual Perception*. New York: Psychology Press, Dec. 2014.

[3] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, 2012.

[4] M. Mujahed, D. Fischer, and B. Mertsching, "Admissible gap navigation: A new collision avoidance approach," *Robotics and Autonomous Systems*, May 2018.

[5] ——, "Safe Gap based (SG) reactive navigation for mobile robots," in *2013 European Conference on Mobile Robots*, 2013, pp. 325–330.

[6] 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.

[7] 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)*, Oct. 2013.

[8] R. Xu, S. Feng, and P. A. Vela, "Potential Gap: A Gap-Informed Reactive Policy for Safe Hierarchical Navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8325–8332, 2021.

[9] 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 (RoMoCo)*, July 2024.

[10] 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," Sept. 2021.

[11] 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*, Oct. 2022.

[12] 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," July 2023.

[13] 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," June 2024.

[14] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[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, pp. 3352–3357.

[16] 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 (SMC)*, Oct. 2017.

[17] 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," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[18] M. Mujahed and B. Mertsching, "The admissible gap (AG) method for reactive collision avoidance," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1916–1921.

[19] 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*, Apr. 2022.

[20] M. Demir and V. Sezer, "Improved Follow the Gap Method for obstacle avoidance," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2017.

[21] S. Feng, A. Abuaish, and P. A. Vela, "Safer Gap: A Gap-based Local Planner for Safe Navigation with Nonholonomic Mobile Robots," Mar. 2023.

[22] 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.

[23] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, Jan. 2012.

[24] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, Dec. 2016.

[25] M. Missura, A. Roychoudhury, and M. Bennewitz, "Polygonal Perception for Mobile Robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 10 476–10 482, iSSN: 2153-0866.

[26] N. Shneydor, *Missile Guidance and Pursuit: Kinematics, Dynamics and Control (1st ed.)*. Horwood Series in Engineering Science. Chichester, UK: Horwood Publishing, 1998.

[27] "Implementation of the Pure Pursuit Path Tracking Algorithm."

[28] L. Wellhausen and M. Hutter, "ArtPlanner: Robust Legged Robot Navigation in the Field," *Field Robotics*, vol. 3, no. 1, pp. 413–434, 2023.

[29] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, July 1998.

[30] A. Bernhart, "Polygons of pursuit," *Scripta Mathematica*, vol. 24, Jan. 1959.

[31] ——, "Curves of general pursuit," *Scripta Mathematica*, vol. 24, Jan. 1959.

[32] A. Bruckstein, "Why the ant trails look so straight and nice," *The Mathematical Intelligencer*, vol. 15, pp. 59–62, Jan. 1993.

[33] V. Rajasekhar and A. G. Sreenatha, "Fuzzy logic implementation of proportional navigation guidance," *Acta Astronautica*, pp. 17–24, Jan. 2000.

[34] Y. Ulybyshev, "Terminal Guidance Law Based on Proportional Navigation," *Journal of Guidance, Control, and Dynamics*, 2005.

[35] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[36] "ROS Toolbox." [Online]. Available: https://www.mathworks.com/products/ros.html

[37] "move_base Package." [Online]. Available: https://wiki.ros.org/move_base

[38] 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.

[39] 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)*, June 2021.

[40] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," May 2018.

[41] Z. Xie and P. Dames, "DRL-VO: Learning to Navigate Through Crowded Dynamic Scenes Using Velocity Obstacles," *IEEE Transactions on Robotics*, Aug. 2023.