Safe Hierarchical Navigation in Crowded Dynamic Uncertain Environments

Hongyi Chen[1], Shiyu Feng[2], Ye Zhao[2], Changliu Liu[3], Patricio A. Vela[1]

***Abstract*— This paper describes a hierarchical solution consisting of a multi-phase planner and a low-level safe controller to jointly solve the safe navigation problem in crowded, dynamic, and uncertain environments. The planner employs dynamic gap analysis and trajectory optimization to achieve collision avoidance with respect to the predicted trajectories of dynamic agents within the sensing and planning horizon and with robustness to agent uncertainty. To address uncertainty over the planning horizon and real-time safety, a fast reactive safe set algorithm (SSA) is adopted, which monitors and modifies the unsafe control during trajectory tracking. Compared to other existing methods, our approach offers theoretical guarantees of safety and achieves collision-free navigation with higher probability in uncertain environments, as demonstrated in scenarios with 20 and 50 dynamic agents.**

I. Introduction

Deploying mobile robots ubiquitously requires that they can safely and reliably accomplish navigation tasks in crowded, dynamic, and uncertain real world settings. Traversing such environments can be challenging as the robot system is expected to plan online, handle the uncertainty, and establish the safe action to avoid multiple moving agents [1]. Current methods have made progress towards this goal, drawing from hierarchical navigation, control theory, deep learning, and optimization. This paper leverages progress on these fronts to establish an online, hierarchical approach to trajectory synthesis in crowded, dynamic environments.

A hierarchical navigation coordinates multiple approaches operating at different temporal and spatial scales [2–5], so as to exploit the advantages of the chosen approaches while offsetting their limitations. Gap-based planners detect passable free-space in local environments while relying on an approximate global path planner. However, current methods, like potential gap (PGap), are designed for static environments without dynamic agents [5–8]. Optimization-based planning methods generate collision-free optimal trajectories as long as the objective function and constraints are well defined. The challenge lies in real time computation requirement and designing the optimization problem [9]. Reactive algorithms, like potential field method (PFM) [10], control barrier function (CBF) [11], and safe set algorithm (SSA) [12], only consider one-step safe control calculation and can get stuck in local minima [13]. Learning-based planning and navigation in crowded environments lack safety guarantees, even if there is an extensive training phase [14–16].

We design a hierarchical navigation solution consisting of a high-level planner for long-term safety and robustness, and a low-level controller for online short-term safety guarantee. The planner layer itself is hierarchical and consists of three components. First, dynamic agents gap analysis (DAGap) is proposed to handle spatio-temporally evolving gaps and to synthesize trajectories for detected gaps. While the DAGap-generated trajectory considers specific pairwise agent groupings, the top two candidates warm start the convex feasible set (CFS) optimizer, which enforces hard safety constraints for all sensed agents while minimizing a trajectory optimizing objective function. To improve robustness, an uncertainty analysis module is proposed to estimate high-confidence bounds on the prediction errors of agents' positions for influencing a safety distance parameter. At the controller layer, adopting the fast reactive safe set algorithm (SSA) monitors and online modifies any unsafe actions to reduce collisions caused by computation delay or trajectory tracking errors. The key contributions are summarized below:

- Dynamic agent gap analysis for simplifying the candidate spatio-temporally evolving solution space and synthesizing candidate trajectories.
- Hierarchical use of DAGap multi-trajectory synthesis followed by CFS trajectory optimization for scaling the agents under consideration.
- High-confidence error bound estimation for use within the safety components of DAGap planning and CFS optimization, with provably high probability safety in uncertain environments.
- Analysis and benchmarking of the proposed solution relative to two hierarchical navigation methods ARENA [17] and DRRT-ProbLP [18], and empirically shown to be safer.

II. Related Work

A. Path Planning in Dynamic Environments

Robotic path planning in static environments is a thoroughly studied problem that can typically be solved very efficiently. However, planning in the dynamic environment, especially in crowded dynamic environment, is still challenging because time is added as an additional dimension to the search-space and requires real-time replanning to deal with unprecedented situations in the future. To overcome the online computation challenges, dynamic A* and other incremental variants of classical planning are proposed which can correct previous solutions when updated information is received, so that the ego vehicle can safely interact with several dynamic agents [19][20][21]. However, they assume the agents to be static and rely on repeatedly replanning to generate collision-free paths, which may cause suboptimality. Restricting the search space by filtering out unsafe subspace is also very common. For example, SIPP eliminates collision intervals and searches in contiguous safe intervals [22] and

Manuscript 1245 submitted to 61st IEEE Conference on Decision and Control .
Received March 31, 2022.

ICS filters out the inevitable collision states when searching the waypoints [23]. Besides, sampling-based planning is a classic concept and various strategies are proposed to handle moving agents, including sampling the control from robot's state $\times$ time space [24] and sampling the robot movement based on the distribution of target goal and agents [18]. Optimization-based algorithms are used to compute smooth and collision-free paths in dynamic environments by setting proper cost functions and constraints [9][25][26]. The downside is that the optimization problem is highly nonconvex which comes from the highly nonlinear inequality constraints, and is computationally expensive.

B. Reactive Collision Avoidance

Reactive approaches are extensively studied to compute an immediate action that would avoid collisions with obstacles. Energy-based reactive methods, include CBF [27], SSA [12] and sliding mode algorithm [28], usually design a scalar energy function to achieve set-invariant control. The underlying assumption is that the dynamics of the system should be known. With the dynamic information, they can correctly and quickly solve an optimization problem to drive the energy function in the negative direction whenever the system state is outside of the safe set. Different from energy-based methods, gradient-based reactive methods like potential field method and its variants don't need the knowledge about system dynamics and offer simple computations [5][29][30]. However, they lack of consideration for robot kinematics and dynamics and do not have sound safety guarantees. Optimal reciprocal collision avoidance [31] and reciprocal velocity obstacles [32] derive the collision-free motion based on the definition of velocity obstacles. But they assume all agents' movements follow certain policies while this assumption is not always valid in the real world. Besides, reinforcement learning methods are becoming popular for safe navigation in crowded environments, however, purely learning based methods lack the safety guarantee even after long time training [14][15][16]. Recent safe learning algorithms use CBF or SSA as action monitor to keep modifying actions generated from the policy and achieve a low collision rate for safety-critical tasks [13][33]. When these reactive methods are tested in challenging environments which require long-term planning, however, they may stuck in local minimal or lead to oscillations and cause collision because of their myopic property. To offset these limitations, some studies build hierarchical path planning consisting a low-level collision avoidance controller and a high-level planner [4][17].

C. Gap-based Navigation

Local planners using the representations of perception space can gain computational advantages by minimally processing the sensor data and recasting local navigation as an egocentric decision process [34][8]. Following this idea, gap-based approaches aim at detecting passable free-space, which is defined as a set of "gaps" comprised of beginning and ending points, from 1D laser scan measurements. Because of the detected collision-free regions, gap-based methods are compatible with other hierarchical navigation strategies to improve the safety of the synthesized trajectories [6][7]. For example, egoTEB combines the representation of gap regions with the trajectory optimization method timed-elastic-bands (TEB) [35], which produces and optimizes multiple trajectories with distinct topologies [8]. As a soft-constraint optimization approach, however, egoTEB cannot guarantee that optimized trajectory will fully satisfy all constraints and the poses of a trajectory may jump over an obstacle. Besides, the potential gap approach considers the integration of gap-based navigation with artificial potential field (APF) methods to derive a local planning module that has provable collision-free properties [5]. However, all previous gap-based navigation methods are designed for static environments without dynamic agents. The intent of this study is to explore more deeply the gap representations in environment with crowded dynamic agents.

III. Methodology

We first give an overview of PGap pipeline and our solution Hierarchical DAGap (H-DAGap) to show the difference between them in fig. 1. PGap pipeline detects the gaps, synthesizes the trajectories for each gap by following the gradient field and pick the one with highest score. The PGap navigation is designed to find out the affordance free space in static environment, while DAGap aims to search feasible trajectories in spatio-temporal space. Moreover, a computational efficient trajectory optimization method CFS and uncertainty analysis are exploited in the planner to improve the safety and robustness of trajectories. The high-level planning happens in child thread while the agent state estimation and low-level safe controller are executed in the main thread, see fig. 2. We will explain the details in the following subsections.

A. Dynamic Agent Gap Analysis

1) Inflated Agent Gap Detection: When a new laser scan-like measurement $\mathcal{L}$ (360°)–consisting of n measurements of dynamic agents within the maximum sensing range $d_{\max}$– is available, we first pass the measurement into a Kalman filter module to estimate the agents' positions and velocities. Then we use $\mathcal{L}$ in our dynamic agent gap (DAGap) analysis module, containing two components: inflated agent gap detection which leads to a set of gap $\mathcal{G}_{\mathrm{S}}$ and dynamic gap analysis which synthesizes a trajectory for each gap $G \in \mathcal{G}_{\mathrm{S}}$.

Inflated agent gap detection inflates the range measurement $\mathcal{L}$ of agents by expanding the radii of the agents (to r_{ins}) to allow the robot to be treated as a point [8]. To guarantee the distance between any points inside the gap region and the agents is larger than r_{ins} when passing through the gap, we calculate the tangent points from the robot to the inflated circle as gap endpoints, see fig. 4. Each agent's inflation radius is adjusted based on its Kalman filter estimation covariance to ensure safety with higher probability (refer to section III-C). Let i be a circular index of the inflated measurement $\mathcal{L}^{inf}$, we perform a clockwise pass through

Manuscript 1245 submitted to 61st IEEE Conference on Decision and Control .
Received March 31, 2022.

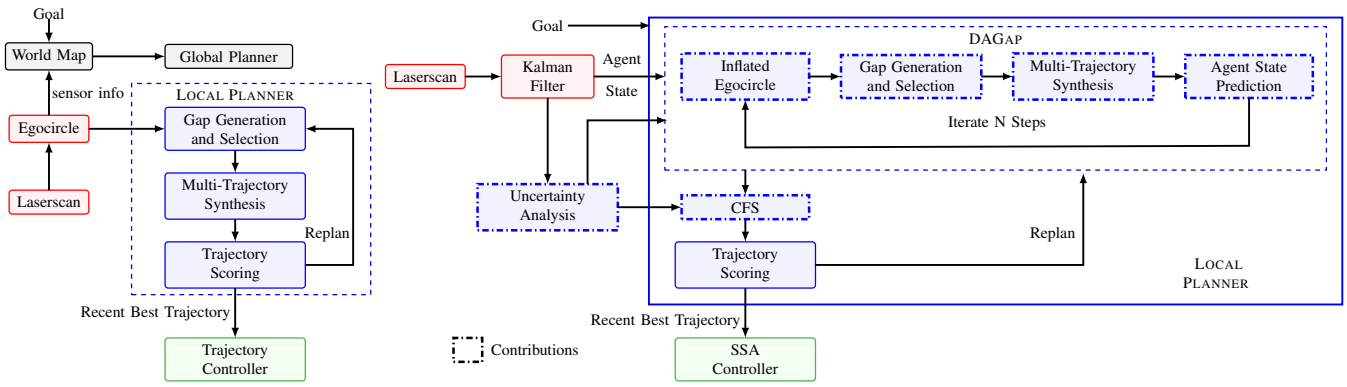


Fig. 1: Comparison between potential gap (PGap) in [5] (left) and our proposed Hierarchical DAGap (H-DAGap) (right). The goal is given from a globally scaled problem, DAGap operates on locally scaled problems using paired agents, CFS enlarges the problem to all sensed agents, and SSA operates at every step with the unsafe agents.

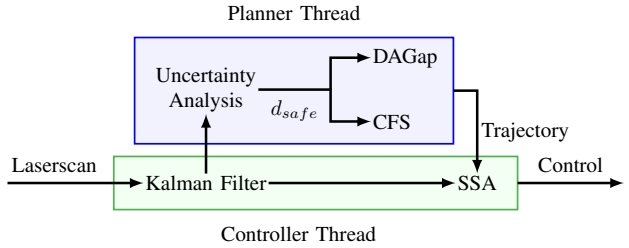


Fig. 2: Parallel computation architecture. The Kalman filter and SSA controller are run in one thread while the high-level planner is executed in another thread.

$\mathcal{L}^{inf}$ and look for candidate gaps satisfying the following two requirements:

1) Large range difference: $|\mathcal{L}_l^{inf}(i+1) - \mathcal{L}_r^{inf}(i)| > 2r_{ins}$
2) Large angle difference: $|\theta_l(i+1) - \theta_r(i)| > \theta_{thre}$,

l, r means the left or right tangent point, $\theta(i)$ is the scan angle associated to index i and θ_{thre} is a user defined parameter. The candidate gap is paired up by right tangent point of previous agent and left tangent point of next agent. When there is a wide open gap between two agents, for example agent1 and agent2 in fig. 3a, we split it into multiple passable gaps by putting static fake agents at a user defined interval. If there is only one or no agent detected, we simply use the straight-line local planner that travels in a straight line towards the target goal.

2) Dynamic Gap Analysis: Two problems need to be answered here: first, how to synthesize the trajectory to accommodate to the spatio-temporal dynamics of an open gap? For this question, we predict the agents' positions based on the Kalman filter, construct a predicted scan $\mathcal{L}$, and recover predicted gaps and gap regions for N steps into the future. Each gap defines a local gap goal. When all gap goals are connected to the robot, they define a star-like graph. N repeated single-step iterations following the PGap gradient field [5] using the predicted gaps synthesizes a set of candidate trajectories. This reactive approach is computationally fast.

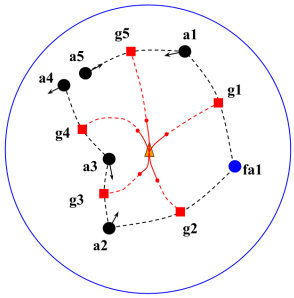
The predicted future gaps states might result in a new gap or have a previously-open gap close. These birth/death events need to be resolved. If a new gap is detected due to agents moving away from each other, for example the agent4 and agent5 in fig. 3b, we will create a trajectory for this newly-open gap at algorithm 1 line 9-10. Among all existing trajectories, we pick the one that is closest to the new gap as the path to expand from. This new trajectory will diverge from the old trajectory and drive towards to the new gap. In fig. 3b, the green trajectory to gap6 is copied from the trajectory to a previously-open but now closed gap5. The previously-open but now closed gap will be labelled as closed and trajectory updating will be halted. After planning finishes, we get all trajectories targeting to open gaps.

Algorithm 1 DAGap Planning

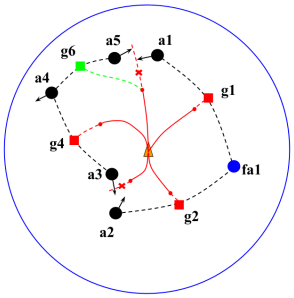
1: $\mathcal{T}_S \leftarrow TrajectorySet$
2: $\mathcal{G}_S \leftarrow GapSet$
3: **for** $planHorizon = 1, 2, \ldots, N$ **do**
4: $\quad \mathcal{G}_S \leftarrow$ gapDection($\mathcal{L}^{inf}$)
5: $\quad$ **if** $planHorizon == 1$ **then**
6: $\quad\quad$ initialize $\mathcal{T}$ for $G \in \mathcal{G}_S$ and add into $\mathcal{T}_S$
7: $\quad$ **end if**
8: $\quad$ **for** $G \in \mathcal{G}_S$ **do**
9: $\quad\quad$ **if** $\mathcal{T}$ of G doesn't exist in $\mathcal{T}_S$ **then**
10: $\quad\quad\quad$ create new trajectory $\mathcal{T}$ for gap G
11: $\quad\quad$ **end if**
12: $\quad\quad$ move $\mathcal{T}$ one step forward towards G using PFM
13: $\quad$ **end for**
14: $\quad$ Update the agents' $\mathcal{L}^{inf}$ with Kalman filter estimation
15: **end for**
16: **return** $\mathcal{T}_S$

B. Trajectory optimization and scoring

A set of trajectories $\mathcal{T}_S$ is generated from the algorithm 1, and each trajectory considers the specific pairwise agents forming its gap, which is acceptable in static environment. But for dynamic agents, they can drive towards the ego robot outside the gap region. It's risky if the ego robot doesn't



(a) DAGap synthesizes multiple trajectories (red, dashed lines) based on detected gaps.



(b) DAGap trajectory updates for newly-opened (green) and closed gaps (red x).

Robot Agents Fake Agents Gap Goals New Gap Goal

Fig. 3: DAGap synthesizes and updates trajectories.

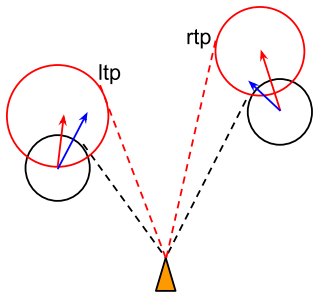


Fig. 4: Inflated agent gap detection. Black and red circles are the initial and expanded inflated circles. Red and blue arrows are the estimated and true agent's velocity. Ltp and rtp mean left and right tangent point, respectively.

do planning in advance, especially when the large agent moves fast. Thus the next challenge is to ensure the safety constraints for all agents during the planning horizon. The CFS optimizer, which can efficiently find optimal solutions that are strictly safe, is adopted to further modify these reference trajectories $\mathcal{T}_S$. Compared to other optimization methods like sequential quadratic programming (SQP), CFS incorporates the geometry information of the problem to improve the computational efficiency while solving the optimization, which is critical for online planning [9].

For each trajectory, robot with initial pose $\boldsymbol{x}_0$ at time t is suppose to reach a local goal position $\boldsymbol{x}_{lgoal}$ at time $t+T$. Let $T = N\Delta t$, where N is the planning horizon and Δt is the discrete time interval. The robot trajectory is denoted as $\boldsymbol{s} = [\boldsymbol{x}^{[0]}; ...; \boldsymbol{x}^{[i]}; ...; \boldsymbol{x}^{[N-1]}]$, where $\boldsymbol{x}^{[0]} = \boldsymbol{x}_0$, and $\boldsymbol{x}^{[N-1]} = \boldsymbol{x}_{goal}$. The trajectory of agent j is denoted as $\boldsymbol{s}_O^j = [\boldsymbol{o}_j^{[0]}; ...; \boldsymbol{o}_j^{[i]}; ...; \boldsymbol{o}_j^{[N-1]}]$, where $j \in \{1, 2, ..., M\}$ and M means the number of agents. Therefore, the robot should plan the trajectory that can reach the local goal while keeping the safety distance r_{ins} from all agents for every time step. Mathematically, the discretized optimization problem is formulated as:

$$\min_{\boldsymbol{s}} \ \|\boldsymbol{s} - \boldsymbol{s}_r\|^2_{Q_r} + \|\boldsymbol{s}\|^2_{Q_s}, \tag{1a}$$

$$s.t.\ D(\boldsymbol{x}^{[i]}, \boldsymbol{o}_j^{[i]}) \geq r_{ins}, \forall i, \forall j, \tag{1b}$$

$$\boldsymbol{x}^{[1]} = \boldsymbol{x}_0, \boldsymbol{x}^{[M]} = \boldsymbol{x}_{lgoal} \tag{1c}$$

where $\|\boldsymbol{s} - \boldsymbol{s}_r\|^2_{Q_r}$ penalizes the deviation from the new trajectory to the reference trajectory, and $\|\boldsymbol{s}\|^2_{Q_s}$ penalizes the properties of the new trajectory itself which ensures low velocity and acceleration magnitude. Constraint $D(\boldsymbol{x}^{[i]}, \boldsymbol{o}_j^{[i]}) \geq r_{ins}$ requires that the robot should keep safe at each planning step, where $D(.,.)$ computes the Euclidean distance between two points. The trajectory scoring eq. (2) combines global target goal efficiency and optimization cost. The trajectory with highest score will be selected to follow:

$$J(\mathcal{T}) = D(target, \boldsymbol{x}^{[M]}) - \|\boldsymbol{s} - \boldsymbol{s}_r\|^2_{Q_r} - \|\boldsymbol{s}\|^2_{Q_s} \tag{2}$$

C. Uncertainty Analysis and Replanning

In the above discussion, we set the safety distance to a fixed value r_{ins} which will work with perfect prediction about agents' trajectories. In uncertain world, however, the estimations of agents' positions and velocities from Kalman filter have errors that will propagate as planning horizon increases. Simply using the fixed safety distance r_{ins} may quickly drive the system into collision. To mitigate this problem, we estimate the high-confidence bound of prediction errors and enlarge the safety distance accordingly during planning. Assume the agent has constant velocity, we denote the current estimated agent state as $\boldsymbol{z}^{[0]} = [\boldsymbol{o}^{[0]}; \boldsymbol{v}]$, including position $\boldsymbol{o}^{[0]}$ and velocity $\boldsymbol{v}$, and its estimated covariance matrix as $\boldsymbol{\Sigma_0}$. The ground truth state is labeled as $\boldsymbol{z}_*^{[0]}$. The future agent state at step i can be predicted as $\boldsymbol{z}^{[i]} = F^i \boldsymbol{z}^{[0]}$, where F represents the transfer matrix. Suppose $\boldsymbol{z}^{[0]} \sim \mathcal{N}(\boldsymbol{z}_*^{[0]}, \boldsymbol{\Sigma}_0)$, the prediction for $\boldsymbol{z}^{[i]}$ should follow the distribution: $\boldsymbol{z}^{[i]} \sim \mathcal{N}(\boldsymbol{z}_*^{[i]}, \boldsymbol{\Sigma_i})$, where $\boldsymbol{\Sigma_i}$ is the covariance matrix propagated forward for i steps $\boldsymbol{\Sigma_i} = (F^T)^i \boldsymbol{\Sigma_0} F^i$. We pick out the submatrix of $\boldsymbol{\Sigma_i}$ corresponding to the position $\boldsymbol{o}^{[i]}$ and denote the position covariance matrix as $\boldsymbol{\Sigma_{i,o}}$. Then the error $\Delta \boldsymbol{o}^{[i]} = \boldsymbol{o}^{[0]} - \boldsymbol{o}_*^{[0]}$ should follow the chi-square distribution χ^2_N as eq. (3), N is the dimension of $\boldsymbol{o}^{[i]}$.

$$(\boldsymbol{\Delta o}^{[i]})^T \boldsymbol{\Sigma}^{-1}_{\boldsymbol{i,o}} \boldsymbol{\Delta o}^{[i]} \sim \chi^2_N \tag{3}$$

and we can obtain the following probability support on confidence bound value k_ϵ:

$$P((\boldsymbol{\Delta o}^{[i]})^T \boldsymbol{\Sigma}^{-1}_{\boldsymbol{i,o}} \boldsymbol{\Delta o}^{[i]} \leq k_\epsilon) > 1 - \epsilon \tag{4}$$

with the lemma 4 in [36], the following bound on the error $\boldsymbol{\Delta o}^{[i]}$ holds with probability $1 - \epsilon$:

$$-\sqrt{k_\epsilon \lambda_n} \leq \boldsymbol{v_n}^T \boldsymbol{\Delta o}^{[i]} \leq \sqrt{k_\epsilon \lambda_n}, \forall n \tag{5}$$

Manuscript 1245 submitted to 61st IEEE Conference on Decision and Control .
Received March 31, 2022.

where $\{\lambda_n\}'s$ and $\{\boldsymbol{v_n}\}'s$ are the eigenvalues and eigenvectors of $\boldsymbol{\Sigma_{i,o}}$, $n \in \{1, 2, ..., N\}$. Since the $\{\boldsymbol{v_n}\}'s$ are perpendicular basis, $\boldsymbol{\Delta o^{[i]}}$ can be represented as $\sum_n a_n \boldsymbol{v_n}$, where a_n is the coefficient.

$$\boldsymbol{v_n^T \Delta o^{[i]}} = \boldsymbol{v_n^T} \sum_k a_k \boldsymbol{v_k} = a_n \|\boldsymbol{v_n}\| \tag{6a}$$

$$-\frac{\sqrt{k_\epsilon \lambda_n}}{\|\boldsymbol{v_n}\|} \leq a_n \leq \frac{\sqrt{k_\epsilon \lambda_n}}{\|\boldsymbol{v_n}\|} \tag{6b}$$

$$\|\boldsymbol{\Delta o^{[i]}}\| = \|\sum_n a_n \boldsymbol{v_n}\| \leq \sum_n \sqrt{k_\epsilon \lambda_n} \tag{6c}$$

Theorem 1. *Using the bounds eq. (6c), we can expand the safety distance by $\sum_n \sqrt{k_\epsilon \lambda_n}$ to guarantees robust safety with probability at least $1 - \epsilon$ in uncertain environment.*

Proof: Since the prediction of the ground truth agent's position $\boldsymbol{o_*^{[i]}}$ is unbounded, we ensure a probability safety constraint between $\boldsymbol{x^{[i]}}$ and $\boldsymbol{o_*^{[i]}}$ for $\forall i$,

$$P(\|\boldsymbol{x^{[i]}} - \boldsymbol{o_*^{[i]}}\| \geq r_{ins}) < 1 - \epsilon \tag{7}$$

At worst case, $\|\boldsymbol{x^{[i]}} - \boldsymbol{o_*^{[i]}}\| = \|\boldsymbol{x^{[i]}} - \boldsymbol{o^{[i]}} + \boldsymbol{o^{[i]}} - \boldsymbol{o_*^{[i]}}\| = |\|\boldsymbol{x^{[i]}} - \boldsymbol{o^{[i]}}\| - \|\boldsymbol{o^{[i]}} - \boldsymbol{o_*^{[i]}}\||$. From eq. (6c), we know the bound on the uncertainty $\|\boldsymbol{o^{[i]}} - \boldsymbol{o_*^{[i]}}\|$ holds with probability $1 - \epsilon$. By expanding the safety distance d_{safe} between ego robot and estimated agent position $\|\boldsymbol{x^{[i]}} - \boldsymbol{o^{[i]}}\|$ to $r_{ins} + r$, where $r = \sum_n \sqrt{k_\epsilon \lambda_n}$, we can guarantee safety with probability at least $1 - \epsilon$.

From theorem 1, we can estimate the robust safety distance d_{safe} for every agent and replace the fixed r_{ins} used in DAGap and CFS, see fig. 2. If the d_{safe} becomes too large, the planner will become overly conservative and reject the path that is passable. To avoid this problem, we set an upper bound of the d_{safe} and record the first time step k this upper bound is reached. Replan will be evoked after k steps execution as the systems have less safety guarantee after that.

D. Safe Controller

Even though DAGap and CFS are efficient planning methods, their longer horizon mean that they are more computationally expensive compared to one-step reactive safe control methods. Moreover, challenging crowded environments mean that CFS may not have converged within a few iterations. Due to the real-time planning requirement, we don't run CFS for all synthesized trajectories until they converge. Instead we select the top two trajectories based on the efficiency score $D(target, \boldsymbol{x^{[M]}})$ as we notice the optimization cost doesn't change the rank ordering of top two candidates in most cases, and run CFS for only one iteration. Moreover, we adopt fast SSA in the low-level controller layer to further compensate the long planning time, and monitor the control at every step. Compared to other reactive algorithms like CBF which enforces constraints everywhere, SSA achieves better safety-efficiency trade-off in complex environment [13].

The key of SSA is to define a valid safety index ϕ such that 1) there always exists a feasible control input in control space that satisfies $\dot{\phi} \leq -\eta\phi$ when $\phi \geq 0$ and 2) any control sequences that satisfy $\dot{\phi} \leq -\eta\phi$ when $\phi \geq 0$ ensures forward invariance and asymptotic convergence to the safe set X_S, η is a positive constant that adjusts the convergence rate. In our problem, $X_S = \{x|\phi_0(x) \leq 0\}$, where ϕ_0 is defined as $d_{min}^2 - d^2$, d_{min} is the user defined minimal distance and d is the distance from the robot to the agent. Since the robot we adopt in testing is a second-order system, we add higher order term of ϕ_0 to ensure that relative degree one from safety index ϕ to the control input, and ϕ is defined as follows:

$$\phi = d_{min}^2 - d^2 - k \cdot \dot{d}. \tag{8}$$

where $\dot{d}$ is the relative velocity from the robot to the agent and k is a constant factor. As proved in [12][37], safety index ϕ will ensure forward invariance of the set $\phi \leq 0 \cap \phi_0 \leq 0$ as well as global attractiveness to that set. With safety index ϕ, we project the reference control u^r to the set of safe controls that satisfy $\dot{\phi} \leq -\eta\,\phi$ when $\phi \geq 0$, and $\dot{\phi}$ is expressed as

$$\dot{\phi} = \frac{\partial \phi}{\partial x} f + \frac{\partial \phi}{\partial x} g\, u = L_f \phi + L_g \phi\, u. \tag{9}$$

In crowded environment, we compute ϕ_j for every agent and add safety constraint whenever ϕ_j is positive. With the safety and dynamics constraints, SSA will solve the following one-step optimization problem through quadratic programming (QP) when triggered:

$$\min_{u \in U} ||u - u^r||^2 = \min_{u \in U} u^{\mathrm{T}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u - 2u^{\mathrm{T}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u^r \tag{10a}$$

$$s.t. L_f \phi_j + L_g \phi_j\, u \leq -\eta\, \phi_j, j \in \{1, 2...M\}. \tag{10b}$$

IV. Experiments

In this section, we will present the experiments, results and comparison of our H-DAGap relative to ARENA [17] and DRRT-ProbLP [18].

A. Benchmark Results

Experiments are conducted in a 2×2 empty world , see fig. 5, with two different test scenarios of increasing difficulty: 20 and 50 dynamic agents whose radius is 0.05 and velocity is sampled from a uniform distribution $\left[5e^{-3}, 2e^{-2}\right]$. Agents can move randomly in any direction in the map and collisions between them are not considered. The robot adopts the second order unicycle model and its speed range is $\left[0, 2e^{-2}\right]$. The overall scenario settings used in ours and two baseline papers are similar, including the relatively agent size to world size and the relatively agent velocity to robot velocity. But we don't assume the perfect lidar measurement is available, instead, the 360° field of view (FoV) measurement has errors following the Gaussian distribution $\mathcal{N}(0,\, 0.01^2)$. We use Kalman filter to track the position and velocity of each agent inside the 0.2 sensing range. Compare to the environment used in our previous safe learning work [13], we enlarge the agent size, speedup its velocity and add measurement uncertainty, which makes the task more challenging. H-DAGap is run in Python on Ubuntu

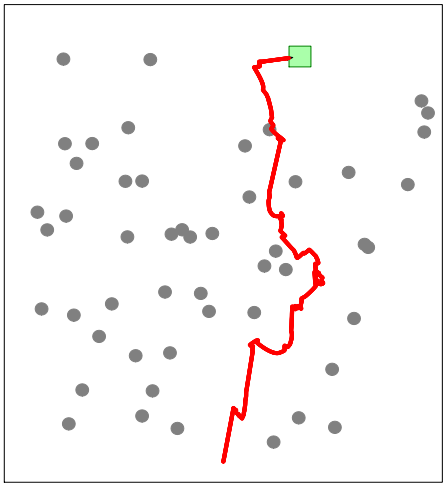


Fig. 5: Testing environment with 50 dynamic agents. The green square is the target goal, the gray circles are agents' positions at last time step and the red trial is the entire robot trajectory from the start to the goal.

20.04 of 3.7 GHz using Intel Core i7. We conduct 100 test runs in each scenario and use collision rate and success rate as evaluation metrics. A success trial means the robot reaches the goal within 3500 steps without any collision. The robot is allowed to continue driving after collision in benchmark papers, and we follow this rule but don't observe multiple collisions in one trial in H-DAGap experiments.

ARENA and DRRT-ProbLP are used for safe navigation in crowded dynamic environments and tested in environment with 20 dynamic agents in their papers. While our model can achieve 97% success and only 3% collision rate in 50 dynamic agents environment in table I, which is better than theirs performances in only 20 agents environment. For DRRT-ProbLP planner, the DRRT module only considers static obstacles and the avoidance of dynamic agents is entirely handled by ProbLP. ProbLP samples the robot movement direction based on the distribution of target goal and agents, then synthesizes and scores the trajectories. However, usually there are large number of possible safe trajectories in open continuous space and sampling-based method can't enumerate all of them and inevitably become suboptimal. For ARENA, it combines the traditional global planner A* and the Deep Reinforcement Learning based (DRL) local planner. But vanilla A* doesn't consider the dynamic agent and DRL doesn't consistently ensure safety constraint throughout execution. On the other hand, our H-DAGap considers and guarantee safety in both layers and in multi-modules. We leave the detailed analysis in the next section.

B. Discussion

In this part, we look into the contribution of different modules and each experiment is repeated for 100 times. Here, the test run will stop once the robot collides.

[4]Note: The success criteria in ARENA is goal attainment with less than two collisions.

TABLE I: Comparison between the H-DAGap and benchmark algorithms in empty world with 20 and 50 dynamic agents.

Model	20 agents		50 agents	
	Success	Collision	Success	Collision
H-DAGap	100%	0%	97%	3%
DRRT-ProbLP	N/A	4%	N/A	N/A
ARENA	$\leq$ 92.7% [4]	23.7%	N/A	N/A

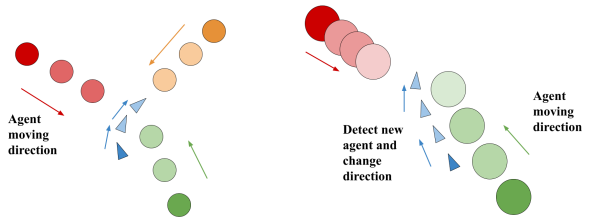
DAGap trajectory synthesis result: We compare the results of DAGap only and static gap detection (SGap), which synthesize trajectories for the gaps detected at current time step. CFS optimization or SSA modification are not applied, which will change the original trajectory. Compared to SGap, our DAGap method reduces the collision rate by 13% and 7% in 20 and 50 agents environments respectively, see table II. The reason is that DAGap considers the spatio-temporal dynamics of open gaps and filters out the trajectories towards gradually closing gaps. SGap guarantees safe passage in static environments, however, the originally open gap may become closed and can lead to collision in dynamic environments. Besides, the improvement of incorporating spatio-temporally evolving information is more significant in easy scenario because DAGap is less affected by the agents outside the gap region due to the lower agents density.

TABLE II: Experimental results of each module.

Model	20 agents		50 agents	
	Collision	Success	Collision	Success
SGap	49%	51%	76%	24%
DAGap	36%	64%	69%	31%
DAGap+CFS	8%	92%	29%	71%
DAGap+CFS+SSA	0%	100%	3%	97%

CFS trajectory optimization result: The collision rate drops from 69% to 29% in 50 agents scenario after CFS optimization and safety distance adjustment with uncertainty analysis. Compared to using CFS directly, DAGap provides good initial trajectories that can improve the safety of optimized trajectory. To better explain it, we need to define the feasibility of a trajectory: a feasible trajectory requires the distances between all neighbouring waypoints are smaller than a threshold value related to the ego robot's maximal velocity. An infeasible trajectory increases the risk of collision and is hard to be tracked by ego robot due to the large jump between waypoints. The feasibility rate of CFS optimized trajectory is around 87.4% when using DAGap to generate initial reference trajectory that drives towards the affordance free space, but is only 66.9% without DAGap.

SSA safe controller result: By replacing the feedback controller with the SSA safe controller, the collision rate drops to 3%. There are two main reasons behind: first of all, as we discussed above, CFS may generate dynamically infeasible trajectory due to the limited number of optimization iterations we can run in real-time and the challenging



(a) Trapping collision (b) Overtaking collision

Fig. 6: Trapping collision and overtaking collision. Robot and agents are shown as colored triangle and circles respectively which are dark at their initial positions and lighten as time progresses.

crowded dynamic environment. Tracking infeasible trajectory can cause collision. SSA can monitor and modify these unreasonable tracking controls online. Secondly, DAGap and CFS are long-term planners considering N steps safety. But the predicted error of trajectories of agents will compound as time goes, making the planned trajectory risky in the long-term future even we expand the safety distance. On the other hand, because of its fast and cheap computation property, SSA always uses the latest information to calculate one-step safe control and to reduce collision caused by uncertainty.

Collision analysis: Even applying all these techniques, we still have 3% collision rate in challenging scenario. The collisions can be categorized into two main cases: multi-agent traps and a fast, overtaking agent. In the first case, when the robot is trapped by multiple agents, see fig. 6a, SSA cannot find a control to meet all safety constraints because the robot will get closer to one of the agents no matter which direction it drives to. Instead, SSA will pick the "best" control which is staying at the current position. These occur due to the limited sensing radius of H-DAGap search space relative to the globaly search space and the greedy nature of SSA. The second case happens when a fast agent driving behind the robot and in collision overtakes it. The robot will follow the safest one-step control generated by SSA, which moves in the direction aligning with the agent's velocity. Even if the DAGap trajectory points in another direction, SSA modifies the original control to the safest single-step one based on the safety index. This escape-and-pursue situation usually continues for several steps until the robot meets another agent and needs to take a new control to avoid both. Then the fast agent behind will quickly catch up and is hard to bypass within one or two steps due to its big size. From the perspective of SSA, no control exists to satisfy the constraints in eq. (10b), however, we can avoid these collisions in two ways. First of all, we can improve the high-level planner design to check if we will drive into the trapping situation and detour in advance. Besides, we need better coordination between the high-level planner layer and low-level SSA layer to avoid their conflicts happened in second case. We leave these as our future works.

V. Conclusion

In this work, H-DAGap, a hierarchical navigation solution containing a multi-phase planner and a low-level safe controller, is discussed to solve the safe navigation problem in crowded, dynamic and uncertain environments. High-confidence error bound is estimated and used in the planner to achieve provably high probability safety in uncertain environments. Moreover, experimental benchmark and analysis are conducted in the simulation to show the effectiveness of H-DAGap. The implementation of H-DAGap is available at https://github.com/hychen-naza/H-DAGap.

References

[1] M. Hoy, A. S. Matveev, and A. V. Savkin. "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey". In: *Robotica* 33.3 (2015), 463–497.

[2] F. Bonin-Font, A. Ortiz, and G. Oliver. "Visual Navigation for Mobile Robots: A Survey". In: *Journal of Intelligent and Robotic Systems* 53 (Nov. 2008), pp. 263–296.

[3] D. Zhu and J.-C. Latombe. "New heuristic algorithms for efficient hierarchical path planning". In: *IEEE Transactions on Robotics and Automation* 7.1 (1991), pp. 9–20.

[4] J. Guldner, V. Utkin, and R. Bauer. "Mobile robots in complex environments: a three-layered hierarchical path control system". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*. Vol. 3. 1994, 1891–1898 vol.3.

[5] R. Xu, S. Feng, and P. A. Vela. "Potential Gap: A Gap-Informed Reactive Policy for Safe Hierarchical Navigation". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8325–8332.

[6] M. Mujahad et al. "Closest Gap based (CG) reactive obstacle avoidance Navigation for highly cluttered environments". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 1805–1812.

[7] M. Demir and V. Sezer. "Improved Follow the Gap Method for obstacle avoidance". In: *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2017, pp. 1435–1440.

[8] J. S. Smith, R. Xu, and P. Vela. "egoTEB: Egocentric, Perception Space Navigation Using Timed-Elastic-Bands". In: *IEEE International Conference on Robotics and Automation* ().

[9] C. Liu et al. "Convex feasible set algorithm for constrained trajectory smoothing". In: *2017 American Control Conference (ACC)*. 2017, pp. 4177–4182.

[10] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505.

[11] A. D. Ames, J. W. Grizzle, and P. Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 6271–6278.

[12] C. Liu and M. Tomizuka. "Control in a safe set: Addressing safety in human-robot interactions". In: *Dynamic Systems and Control Conference*. 2014, V003T42A003.

[13] H. Chen and C. Liu. "Safe and Sample-Efficient Reinforcement Learning for Clustered Dynamic Environments". In: *IEEE Control Systems Letters* 6 (2022), pp. 1928–1933.

[14] M. A. Kareem Jaradat, M. Al-Rousan, and L. Quadan. "Reinforcement based mobile robot navigation in dynamic environment". In: *Robotics and Computer-Integrated Manufacturing* 27.1 (2011), pp. 135–149. ISSN: 0736-5845.

[15] L. Liu et al. “Robot Navigation in Crowded Environments Using Deep Reinforcement Learning”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5671–5677.

[16] K. Li, Y. Lu, and M. Q.-H. Meng. “Human-Aware Robot Navigation via Reinforcement Learning with Hindsight Experience Replay and Curriculum Learning”. In: *arXiv preprint arXiv:2110.04564* (2021).

[17] L. Kästner et al. “Arena-Rosnav: Towards Deployment of Deep-Reinforcement-Learning-Based Obstacle Avoidance into Conventional Autonomous Navigation Systems”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 6456–6463.

[18] M. D’Arcy, P. Fazli, and D. Simon. “Safe navigation in dynamic, unknown, continuous, and cluttered environments”. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 238–244.

[19] A. Stentz. “The Focussed D* Algorithm for Real-Time Replanning”. In: *IJCAI*. 1995.

[20] M. Likhachev and D. Ferguson. “Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles”. In: *The International Journal of Robotics Research* 28.8 (2009), pp. 933–945.

[21] 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*. 2007, pp. 1603–1609.

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

[23] S. Petti and T. Fraichard. “Safe motion planning in dynamic environments”. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 2210–2215.

[24] D. Hsu et al. “Randomized kinodynamic motion planning with moving obstacles”. In: *The International Journal of Robotics Research* 21.3 (2002), pp. 233–255.

[25] C. Park, J. Pan, and D. Manocha. “ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments”. In: *Twenty-Second International Conference on Automated Planning and Scheduling*. 2012.

[26] W. Xu et al. “A real-time motion planner with trajectory optimization for autonomous vehicles”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2061–2067.

[27] A. Agrawal and K. Sreenath. “Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation.” In: *Robotics: Science and Systems*. Vol. 13. Cambridge, MA, USA. 2017.

[28] L. Gracia, F. Garelli, and A. Sala. “Reactive Sliding-Mode Algorithm for Collision Avoidance in Robotic Systems”. In: *IEEE Transactions on Control Systems Technology* 21.6 (2013), pp. 2391–2399.

[29] G. Li et al. “An efficient improved artificial potential field based regression search method for robot path planning”. In: *2012 IEEE International Conference on Mechatronics and Automation*. 2012, pp. 1227–1232.

[30] J. Guldner and V. Utkin. “Sliding mode control for gradient tracking and robot navigation using artificial potential fields”. In: *IEEE Transactions on Robotics and Automation* 11.2 (1995), pp. 247–254.

[31] J. v. d. Berg et al. “Reciprocal n-body collision avoidance”. In: *Robotics research*. Springer, 2011, pp. 3–19.

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

[33] R. Cheng et al. “End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks”. In: *Proc. AAAI Conf. Artificial Intelligence*. 2019, pp. 3387–3395.

[34] J. S. Smith et al. “Real-Time Egocentric Navigation Using 3D Sensing”. In: *Machine Vision and Navigation*. Ed. by O. Sergiyenko, W. Flores-Fuentes, and P. Mercorelli. Cham: Springer International Publishing, 2020, pp. 431–484.

[35] C. Roesmann et al. “Trajectory modification considering dynamic constraints of autonomous robots”. In: *ROBOTIK 2012; 7th German Conference on Robotics*. 2012, pp. 1–6.

[36] R. Cheng et al. “Safe multi-agent interaction through robust control barrier functions with learned uncertainties”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 777–783.

[37] W. Zhao, T. He, and C. Liu. “Model-free Safe Control for Zero-Violation Reinforcement Learning”. In: *Conference on Robot Learning, 8-11 November 2021, London, UK*. Ed. by A. Faust, D. Hsu, and G. Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, 2021, pp. 784–793.