Honda Research Institute EU
**Honda Research Institute Europe GmbH**
https://www.honda-ri.de/

# Finding the Easy Way Through-the Probabilistic Gap Planner for Social Robot Navigation

**Malte Probst , Raphael Wenzel, Tim Puphal, Monica Dasi, Nico Andreas Steinhardt, Sango Matsuzaki, Misa Komuro**

**2025**

**Preprint:**

This is an accepted article published in IEEE RO-MAN 2025. The final authenticated version is available online at: https://doi.org/10.1109/RO-MAN63969.2025.11217922 Copyright 2025 IEEE

# Finding the Easy Way Through – the Probabilistic Gap Planner for Social Robot Navigation

Malte Probst[1], Raphael Wenzel[1], Tim Puphal[1,2], Monica Dasi[1],
Nico A. Steinhardt[1], Sango Matsuzaki[3], Misa Komuro[3]

***Abstract*— In Social Robot Navigation, autonomous agents need to resolve many sequential interactions with other agents. State-of-the art planners can efficiently resolve the next, *imminent* interaction cooperatively and do not focus on longer planning horizons. This makes it hard to maneuver scenarios where the agent needs to select a good strategy to find gaps or channels in the crowd. We propose to decompose trajectory planning into two separate steps: *Conflict avoidance* for finding good, macroscopic trajectories, and *cooperative collision avoidance* (CCA) for resolving the next interaction optimally. We propose the Probabilistic Gap Planner (PGP) as a conflict avoidance planner. PGP modifies an established probabilistic collision risk model to include a general assumption of cooperativity. PGP biases the short-term CCA planner to head towards gaps in the crowd. In extensive simulations with crowds of varying density, we show that using PGP in addition to state-of-the-art CCA planners improves the agents' performance: On average, agents keep more space to others, create less tension, and cause fewer collisions. This typically comes at the expense of slightly longer paths. PGP runs in real-time on WaPOCHI mobile robot by Honda R&D.**

## I. Introduction

The goal of Social Robot Navigation (SRN) is to enable robots to seamlessly navigate public, unstructured, and potentially crowded environments. While this task is seemingly simple for humans, it poses significant challenges for autonomous agents [1]. Despite many important abstract questions like intention estimation, politeness, or contextual appropriateness [2], avoiding collisions with people in a cooperative, foresighted manner without being overly cautious remains an active topic in SRN research.

State of the art SRN algorithms can provide short-term plans of a few seconds in real time, which enables them to navigate through dense crowds effectively [3]. Despite these recent achievements, planners with shorter horizons have difficulties to create foresighted behavior. Cooperative collision avoidance (CCA) for longer horizons remains difficult, due to higher uncertainties and, when coupling planning and prediction, the resulting combinatorial explosion of potential future outcomes in more crowded environments [1], [4]–[6].

To address this problem from a slightly different angle, we follow a simple observation by [7] in the context of *conflict avoidance* for social navigation. The authors define a conflict between two agents as "a situation in which, if there is no change of direction or a change in speed by at least one of the parties, they will collide". Furthermore,

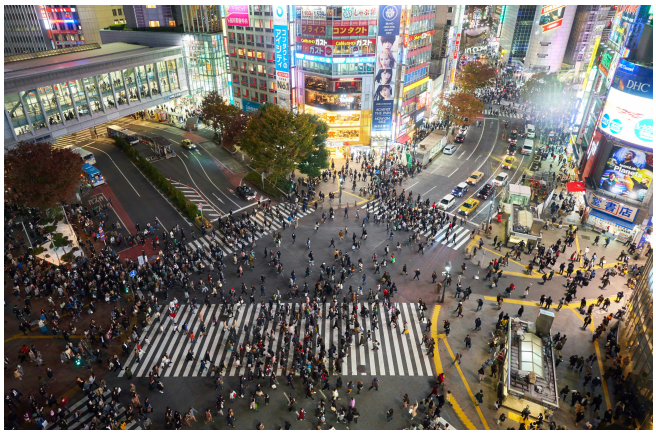

Fig. 1: Autonomous robots in public places like Shibuya Crossing, Tokyo need to be able to navigate smartly in crowded, unstructured environments. Here, areas with varying density can emerge which the robot can leverage to improve its motion through the crowd. *Photo by T. Volz*

the authors note that "not all conflicts end in a physical collision, but every collision is preceded by a conflict". Accordingly, in this paper, we propose to split the task of motion planning on a trajectory level for SRN into two distinct planning stages. First, a long-term planner which reduces the number of conflicts by finding foresighted, macroscopic trajectories, such as deviating slightly but early to evade an oncoming pedestrian, or finding potentially suitable gaps in a crowd of people. Second, a short-term planner which avoids collisions by resolving microscopic cooperative interactions. We make two main assumptions in our approach. First, we assume that the long-term planner can rely on the short-term planner to handle (cooperative) collision avoidance. Second, we assume that all other agents (e.g., pedestrians) also want to avoid collisions - at least to some extent. Given this premise, the long-term planner can anticipate some degree of cooperativity from other agents it will encounter in the future. Together, the assumptions relax the trajectory planning problem for the long-term planner significantly, since it does not have to discard conflicting trajectories at all costs, or model each potential future interaction in a detailed manner (e.g., by explicitly coupling prediction and planning). This paper presents the Probabilistic Gap Planner (PGP) as a long-term planner. PGP applies a well-known probabilistic risk model proven in both the Autonomous Driving (AD) and the robotics context [8]. We modify the probabilistic risk calculation to include PGP's cooperativity assumption.

[1]Honda Research Institute EU GmbH first.last@honda-ri.de
[2]Honda Research Institute JP Co., Ltd. tim.puphal@jp.honda-ri.com
[3]Honda R&D Co., Ltd. first_last@jp.honda

PGP guides the short-term planner towards areas with lower predicted density, i.e., less crowded trajectory options on the way towards the goal. PGP relies on the short-term planner to resolve all microscopic interactions without collisions.

In extensive experiments in simulation, we assess how adding PGP on top of state-of-the-art short-term planners improves the performance in scenarios with varying levels of crowdedness. The results show that PGP helps to find significantly less dense paths through crowds of people and reach the goal faster without compromising safety. Furthermore, we test PGP in indoor corridor scenarios on the WaPOCHI mobile robot by Honda R&D, where it runs in real time.

## II. Related Work

Hierarchical planning is a common approach in AD. Often, the problem is decomposed into route planning, path planning, maneuver planning (or decision making), and control [9]–[11]. The maneuver planning and control layers operate on trajectory level, and typically consider dynamic objects. For example, common approaches use lattice planners or discrete search techniques for coarse maneuver planning, and numerical optimization for control considering the vehicle dynamics. Due to the structure imposed by roads, lanes, or traffic rules it is often feasible to define high-level maneuvers or even parametrized maneuver primitives such as "lane change" or "right turn".

In robotics, hierarchical planning is also quite common. However, many architectures on real robots use a combination of path planning and collision avoidance [3]. It is less common to have multiple layers of trajectory planning, especially in the scope of the maneuver space. One reason could be that the planning problem is inherently less structured, since robots can, in principle, use the full 2D space for navigation.

Nevertheless, some approaches use multiple trajectory planners. [12] combine an s-t graph-based collision avoidance planner with a local planner superimposing corrections based on sensed obstacles. [13] combine a coarse lattice planner with numerical optimization for trajectory fine tuning, mainly focusing on static objects. Both approaches consider trajectories with predicted collisions infeasible and remove them already in their respective hierarchical layer.

In the context of SRN, a recent research focus has been on cooperatively solving interaction problems. Cooperative Collision Avoidance (CCA) algorithms go beyond standard reactive collision avoidance (CA) algorithms like DWA [14] by including assumptions about cooperativity. For example, ORCA [15] enables real-time, collision-free navigation for multiple agents by computing velocity constraints that ensure reciprocal, cooperative collision avoidance while optimizing individual motion objectives. [6] include cooperativity aspects in a planning algorithm based on braids, a representation used to describe trajectories of multiple agents in a shared space over time. [16] augment a model predictive control algorithm with a sophisticated learned prediction model, thereby coupling prediction and planning. [4] use iterative Bayesian updates over probability distributions to solve a mixed-strategy Nash equilibrium, finding cooperative solutions in real time.

However, many of the algorithms which explicitly consider trajectory-based interactions between agents face the problem of exponential complexity when more than a few other agents need to be considered, due to larger planning horizons or higher crowd density [6], [17]. By design, they typically focus on the next, *imminent* interaction at any point in time.

Summarizing, while there are approaches that use multiple hierarchical trajectory planners, they typically remove potentially colliding trajectories early on, which makes them susceptible to the *freezing robot* problem [18]. In the context of SRN, there are a number of effective CCA algorithms. However, they typically consider the next interaction with a few other agents at a time, rendering larger planning horizons or denser crowds computationally challenging.

## III. Probabilistic Gap Planner

PGP should avoid or reduce potential conflicts with other agents in the planned trajectory, assuming general cooperativity of agents. While PGP prefers planned trajectories with fewer conflicts, it does not perform collision avoidance, but relies on the CCA planner for this task. This relaxes planning and prediction significantly. Therefore, unlike most collision avoidance algorithms, PGP does not render a trajectory with a high level of predicted conflict, i.e., potential collision risk, infeasible. Instead, it reduces the expected utility of the trajectory, depending on the collision probabilities encountered along the way (see III-A). Therefore, PGP's main goal is to maximize the expected utility of the planned trajectory with respect to the long-term goal. To achieve this, PGP creates a set of candidate trajectories (III-B), evaluates the expected utility of each trajectory in the set (III-C), and, from the best one, extracts a subgoal for the CCA planner.

### A. Cooperative Collision Risk and Survival Analysis

PGP employs the probabilistic collision risk model of [8], [19], [20]. This approach has been used in the context of AD and Advanced Driver Assistance Systems (ADAS) for collision avoidance (see, e.g., [21], [22]) and calculates the integrated probability that a planned ego trajectory will collide with any of multiple other predicted trajectories in the future.

The model assumes that the location $X_{i,k}$ of agent $k$ at future time $i$ is a normal distribution with the predicted mean $\mu_{i,k}$ and standard deviation $\sigma_{i,k}$:[1]

$$X_{i,k} \sim \mathcal{N}(\mu_{i,k},\, \sigma_{i,k}^2)$$

Typically, $\sigma_{i,k}$ grows over the predicted time to reflect the growing uncertainty as to where an agent (ego or other) will be located in the future. Accordingly, the probability of two

[1]For the sake of readability, all following equations are 1-dimensional with respect to the agents' positional distributions. Analog to [21], the actual implementation is 2-dimensional. This is a straightforward extension of the 1d case, containing an extra rotation of the ellipsoid 2d Gaussian distributions.

agents colliding, i.e., occupying the same space at time step $i$ is defined as the product of their two Gaussian distributions (see [21], [22]).

In practice, we calculate the probability $p_{i,k}^{\text{coll}}$ of the ego agent colliding with other agent $k$ at time step $i$ as

$$p_{i,k}^{\text{coll}} = \exp(-\frac{||\mu_{i,\text{ego}} - \mu_{i,k}||_2^2}{2\tilde{\sigma}_i}) \cdot \frac{\tilde{\sigma}_{0,k}}{\tilde{\sigma}_{i,k}} \tag{1}$$

with

$$\tilde{\sigma}_{i,k} = \sigma_{i,\text{ego}}^2 + \sigma_{i,k}^2$$

Note that the exponential term of Eq. 1 equals one if the agents' predicted location distributions' means are identical. The second term normalizes the probability to the initial variances $\sigma_{0,\text{ego}}$ and $\sigma_{0,k}$ such that $p_{0,k}^{\text{coll}} = 1$ at $t = 0$, and $p_{0,k}^{\text{coll}} < 1$ for $t > 0$ if the variances grow over time. In other words, the collision probability typically decreases with growing variances, i.e., decreasing probability density.

However, the risk model assumes that the random variables are *conditionally independent*, whereas one of PGP's underlying assumptions is that agents are, in general, *cooperative*. That is, they will tend to avoid future collisions by changing course, speed, or both. In turn, this means that the predicted location distributions are, in fact, *not* conditionally independent and that the actual probability that the agents will be colliding is typically lower.

Estimating the true collision probability of dependent location distributions for two agents is difficult and would require coupling planning and prediction. Therefore, we introduce an *approximate* normalization factor $\tau \leq 1$ as

$$\tau = \frac{\sigma_{0,\text{ego}}}{\sigma_{i,\text{ego}}} \cdot \frac{\sigma_{0,k}}{\sigma_{i,k}} \tag{2}$$

In effect, $\tau == 1$ if the predicted collision is imminent ($i == 0$), or if both standard deviations remain constant over time (i.e., we are very certain about both agents' future locations). In all other cases, $\tau$ decreases linearly with growing standard deviation of either agent, since agents have more space to evade, i.e., cooperate.[2] If the actual cooperativity of a specific agent is known, can be estimated, or inferred by aspects of the perceived scene, it is possible to further extend Eq. 2. For example, $\tau$ could be kept close to its maximum of 1 if the other agent is a child, carrying a heavy box, or restricted by a closeby static object.

Finally, using $\tau$, we get the predicted *cooperative* collision risk as

$$p_{i,k}^{\text{coll, coop}} = p_{i,k}^{\text{coll}} \cdot \tau \tag{3}$$

Similar to the collision probabilities with dynamic objects, we calculate the probability $p_i^{\text{coll, static}}$ of colliding with a static object at time $i$. To keep things simple, we model the collision probability using the minimal lateral distance $d_i$ of any static object to the ego agent and use a 1d Gaussian for ego's location distribution. Consequently,

$$p_i^{\text{coll, static}} = 1 - \text{erf}\left(\frac{d_i}{\sigma_i\sqrt{2}}\right) \tag{4}$$

with $\text{erf}(z)$ being the Gauss error function defined as

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \text{d}t$$

Having calculated the predicted static object collision probability and cooperative collision probabilities, we now use the Survival Analysis from [8]. It integrates $p_i^{\text{coll,static}}$ and all $p_{i,k}^{\text{coll, coop}}$ from multiple other agents $k$ over time to get the probability $p_i^{\text{surv}}$ of "surviving" up time step $i$ by converting all probabilities to event rates (for more details refer to [8]). For PGP, we will use the survival probability $p_i^{\text{surv}}$ to discount the predicted utility of a planned trajectory in the evaluation (see Section III-C).

Considering the discrete case, the Survival Analysis calculates the total survival probability $p_i^{\text{surv}}$ of the ego agent as

$$p_i^{\text{surv}} = \prod_{j<i} \exp(-\sum_0^k p_{j,k}^{\text{coll, coop}} + p_i^{\text{coll, static}} + p^{\text{escape}}) \tag{5}$$

Here, $0 < p^{\text{escape}} < 1.0$ is the escape probability, which can be interpreted as the probability of an unforeseen event happening at any time step. In practice, this term will make the $p_i^{\text{surv}}$ decay exponentially over time, and fall to zero in the limit, even in the absence of any static or dynamic collision risks. Any non-zero collision probability with another agent or a static object will lead to a stronger decrease of the survival probability at time step $i$. As a consequence, if a trajectory has a high collision risk at time step $t$, e.g., by leading through a dense crowd of people or close to a static object, its expected utility at $t = j \geq i$ will be discounted stronger, compared to a trajectory without any close encounters (see Section III-C).

The computational complexity of the analytic risk calculation grows linearly in the number of considered agents. However, since collision probabilities for all agents are conditinally independent of each other, they can be implemented very efficiently using parallel features of standard CPUs, rendering computation times almost negligible in practice.

### *B. Predictions and PGP Trajectory Search Space*

PGP uses a classic predict-then-plan approach, decoupling prediction and planning. PGP's prediction step uses simple constant-velocity-constant-angle extrapolations of all other sensed agents.[3]

The total space of potential plans is very large, containing all kinematically possible ego trajectories within the prediction horizon. Here, PGP uses a fixed sample of trajectories to

[2]It is conceivable that we could parametrize the growth of $\sigma_{i,k}$ in a way that Eq. 1 already possesses the desired properties, rendering $\tau$ unnecessary. Nevertheless, we argue that it is easier to *explicitly* capture the general assumptions about an agent's motion in the growth of $\sigma_{i,k}$, and specifically include the approximated cooperation assumption via $\tau$

[3]PGP can use arbitrary trajectory predictions. A more sophisticated module could potentially predict pedestrians' future positions better, e.g., by factoring in their specific characteristics, or elements of the perceived scene. Known uncertainties about specific pedestrians can be accommodated by the risk model by individually modeling the growth of $\sigma_{i,k}$.

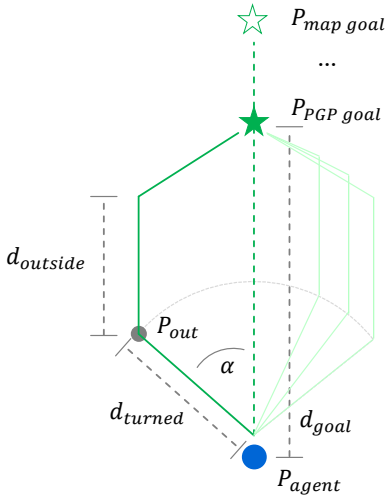


Fig. 2: PGP path creation. The agent (green dot) wants to reach the map goal (light green star). PGP calculates path variations around the straight line to the goal to create candidate trajectories.

cover potentially promising areas of the search space. The sample is based on variations of the direct path from the current ego position to the long-term goal (see Figure 2). The quality of each trajectory with respect to the robot's goal will be evaluated later. First, we derive a feasible goal for PGP within its prediction horizon $T$, at a distance of $d_{\text{goal}} = T * v^{\max}$, assuming $T$ seconds of constant maximum velocity $v^{\max}$ towards the goal on the map $P_{\text{map goal}}$. Then, we create paths consisting of three segments. The first segment of length $d_{\text{turned}}$ is directed away from the straight path to the goal at angle $\alpha$. The second segment of length $d_{\text{outside}}$ is, again, parallel to the straight line to goal. The third segment connects the second segment to the goal.

For each path, we create a constant velocity trajectory with $v = 1.0\ \frac{\text{m}}{\text{s}}$. We accommodate for the fact that the agent cannot turn arbitrarily fast by reducing velocity for a short time if considerable turning (i.e., more than 30°) is required. In that case, we approximate the required time to turn as $t^{\text{turn}} = \frac{\gamma^{\text{turn}}}{\dot{\gamma}^{\max}}$, with $\gamma^{turn}$ being the angle required to turn towards $P_{\text{out}}$, and $\dot{\gamma}^{\max}$ being the maximal turn rate. Then, we reduce the initial velocity for $t^{\text{turn}}$ seconds to $0.5 \cdot v$. Accordingly, the final set $L$ for evaluation contains trajectories fanning out at different angles in both directions, staying "outside" for a while or returning back immediately.

### C. PGP Trajectory evaluation, selection, and goal passing

PGP evaluates the expected utility $u_l^{\text{expected}}$ of each trajectory $l$ in the set of all trajectories $L$ by summing up all expected utilities over the planned horizon. For each time step $i$, we calculate the utility $u_{i,l}$ based on the planned velocity $v_{i,l}$ and the angle difference $\beta_{i,l}$ between the heading and a straight line towards the goal $P_{\text{PGP goal}}$ as:

$$u_{i,l} = \frac{v_{i,l}}{v_{\max}} * \frac{\cos(\beta_{i,l}) + 1}{2} \tag{6}$$

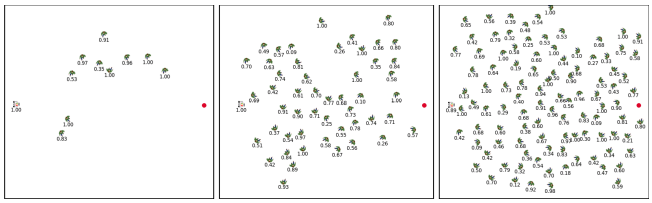

Fig. 3: Screenshots from three simulated scenarios with varying density (from left to right: 0.1, 0.5, and 1.0 agents / m$^2$). The agent (robot icon) needs to traverse a 10 x 10 m stage towards a goal (red dot) populated by groups of other agents moving around to random goals on the stage.

The second term normalizes $u_{i,l}$ to be between zero (heading away from goal) and one (heading towards goal).

We now weigh the utility by the probability of reaching this time step in the first place (see Eq 5). Therefore,

$$u_{i,l}^{\text{expected}} = p_{i,l}^{\text{surv}} * u_{i,l} \tag{7}$$

$$u_l^{\text{expected}} = \sum_i u_{i,l}^{\text{expected}} \tag{8}$$

PGP now selects the trajectory $l^*$ with the highest expected utility $u_{l*}^{\text{expected}}$, and uses it to calculate a subgoal which is passed to the collision avoidance planner. To calculate the subgoal, we simply use the trajectory's planned initial heading $\alpha_{l^*}$ (see Section III-B), and place the subgoal at a distance $d_{\text{subgoal}} = v^{\text{max, CCA}} * t^{\text{plan, CCA}}$, with $v^{\text{max, CCA}}$ and $t^{\text{plan, CCA}}$ being the desired velocity and planning horizon of the CCA planner. Hence, PGP biases the CCA planner to go into the initial direction of $l^*$.

## IV. Experimental Setup

### A. Scenario and Evaluation

We evaluate PGP in an open space scenario with a varying number of pedestrians (see Figure 3). The scenario is conceptually similar to the *Random* scenario of [23] and the *Crowd Navigation* scenario of [2]. Concretely, the agent needs to traverse a square stage of 10 x 10 m. Other pedestrians are spawned with random start and goal locations within the stage. All agents have a desired velocity of 1.0 $\frac{\text{m}}{\text{s}}$. We simulate all other agents in their groups with the Social Forces (SF) model [24].[4] We vary the density of other agents from $0.01 \frac{\text{agents}}{\text{m}^2}$ (one single other agent) to a maximum density of $1.0 \frac{\text{agents}}{\text{m}^2}$. Note that densities $> 0.72 \frac{\text{agents}}{\text{m}^2}$ already fall in a range where "the majority of persons would have their normal walking speeds restricted and reduced due to [...] avoiding conflicts" and the kind of scenario is "representative of the most crowded public areas, where it is necessary to continually alter walking stride and direction to maintain reasonable forward progress" [25, p. 8]. To impose more structure on the flow of pedestrians, we create groups of up to four people. Approximate spawning and goal locations are shared within a group. When any agent of a group has

[4]We use the SF implementation from `https://github.com/yuxiang-gao/PySocialForce`

arrived at its goal, we sample a new goal on the stage for the whole group to avoid standing agents blocking the stage. The preferred walking speed is $1.0\frac{\text{m}}{\text{s}}$ for all agents, including the ego agent.

With the variable density, the complexity of the scenario can vary drastically. It can be comparably simple, with only a few other agents that the ego agent needs to cooperate with. On the other extreme, it can be very complex, with many other agents, where continuous flow is interrupted, dynamic channels are forming and collapsing, and agents may get stuck due to suboptimal short-term decisions (or, simply, bad luck).

We evaluate the performance of the ego agent with the following metrics (see [2], [26]):

*1) Success Metrics:* We evaluate the performance by measuring the time the ego agent needs to reach the target, the total path length, and the collision rate while it is moving.[5]

*2) Quality/social Metrics:* We calculate the Space Violation Rate (SVR), also referred to as Space Compliance in [2]. SVR is the rate at which the agent is closer to any other agent than a given threshold (here, use a threshold of 1.0m). Same as for the collision rate, we only calculate the SVR when the agent is moving. Second, we use the Social Force model to calculate the average sum of Social Forces, i.e., the repulsive force between pairs of agents, affecting the ego agent. In contrast to threshold-based metrics such as SVR, the average Social Force also captures aspects of how a conflict evolves over time. We therefore interpret it as a proxy for social compliance, observing that the average force between two agents typically decreases faster if both agents act towards resolving the conflict, instead of only one agent performing all evasive actions alone.

### B. Tested algorithms

As baseline agents, we use three common approaches. First, we use an agent implementing the Dynamic Window Approach (DWA) [14]. DWA is a pure CA algorithm, and does not make any assumptions about cooperations. We use a *predictive* version of DWA, i.e., collision checks with others are based on their linear predictions. Second, we use an agent using Optimal Reciprocal Collision Avoidance (ORCA) [15]. ORCA assumes that agents cooperate for collision avoidance by taking "half of the responsibility for avoiding pairwise collisions". As a third baseline, we use a Social Forces (SF) agent [24] which is frequently used for crowd simulations. The underlying idea is that agents' movements are subject to a number of pulling and pushing forces.

To test our PGP algorithm, we extend each of the baselines with an upfront PGP planner. Specifically, in each time step, we run one PGP planning step. We then use the output of PGP, i.e., a goal in x/y coordinates, and give this goal to the respective collision avoidance algorithm.

Therefore, we evaluate the performance of three baseline agents (DWA, ORCA, SF) and three PGP-enabled agents (PGP+DWA, PGP+ORCA, PGP+SF).

[5]We assume that if an agent is standing, it should not be responsible for avoiding collisions.

### C. Parametrization

*1) Basic parameters:* For PGP, we use a prediction horizon of 8.0 seconds at 4 Hz, totaling in 32 planned samples. The agent's maximum velocity is 1 $\frac{\text{m}}{\text{s}}$.

*2) Risk calculation:* We parametrize the growth of all agents' Gaussian location distributions as follows: The time-dependent standard deviation $\sigma_i$ starts from an initial $\sigma_0 = 0.1666$, i.e., nearly all the probability mass is within a 1 m radius around the agent's predicted position. $\sigma_i$ increases over the predicted time, with a linear dependency on the predicted velocity at any particular time step. Larger velocities lead to larger increases, assuming that fast agents tend to show larger deviations than slow or standing agents. The total standard deviation is capped at a speed-dependent value $\sigma_{\text{max}}$. Specifically,

$$\sigma_{\text{max}} = \min(\beta * \sigma_0, \sigma_0 + \gamma * \max_i(v_i))$$

and

$$\sigma_i = \min(\sigma_{\text{max}}, \sigma_{i-1} + \delta * v_i)$$

In order to let $\sigma_i$ stay constant over time at standstill, and grow to three times the initial value over the total prediction horizon for the maximum velocity, we choose $\beta = 3.0$, $\gamma = 0.4$, and $\delta = 0.015$ for all agents.

*3) PGP Trajectory creation:* PGP creates the set $L$ of trajectories with the following parameters: A goal distance $d_{\text{goal}} = 8.0$ m, fan-out angle $\alpha \in [-80, -64, \ldots, 64, 80]$ deg, a segment length pointing outward of $d_{\text{turned}} = 2.5$m. For each fan-out-angle $\alpha$, we create two paths, one which immediately turns towards PGP's goal, the other one stays "outside" for 90% of the remaining distance before turning back, i.e., $d_{\text{outside}} \in [0, 0.9 * ||P_{\text{out}} - P_{\text{PGP goal}}||]$m.

*4) Simulation:* We run simulations on stages with increasing population density $\in [0.01, 0.1, 0.2, \ldots, 1.0]\frac{\text{agents}}{\text{m}^2}$, from a single other agent up to a total of 100 other agents, corresponding to one agent per $m^2$ on the 10 x 10 m stage. For each algorithm and density, we run 100 randomized scenarios, with identical random seeds across the evaluated algorithms.

*5) Benchmark algorithms:* The main parameters for the benchmark algorithms DWA, ORCA, and SF are listed in Table I.

| **DWA** | **ORCA** | **SF** |
|---|---|---|
| $t^{\text{plan}} = 2.0$s, dt $= 0.25$s, $v^{\text{min}} = 0\frac{\text{m}}{\text{s}}$, $v^{\text{max}} = 1\frac{\text{m}}{\text{s}}$, $\lvert\dot{\psi}\rvert^{\text{max}} = 1$rad, $\lvert\ddot{\psi}\rvert^{\text{max}} = 1.5\frac{\text{rad}}{\text{s}^2}$, $\lvert a\rvert^{\text{max}} = 1.5\frac{\text{m}}{\text{s}^2}$, dynamic window grid: 10 yaw rates * 10 velocities | $t^{\text{plan}} = 2.5$s, dt $= 0.25$s, $v^{\text{max}} = 1\frac{\text{m}}{\text{s}}$, $d^{\text{neighbor}} = 4.0$m, $n^{\text{neighbors}} = 5$, | $v^{\text{max}} = 1\frac{\text{m}}{\text{s}}$, A $= 5.1$, $\lambda = 3.0$, $\gamma = 0.35$, $n = 1$, $n' = 3.0$ |

TABLE I: Main parameters of benchmark algorithms; $\dot{\psi}$: yaw rate; $t^{\text{plan}}$: planning horizon

### D. Results

In the following, unless otherwise mentioned, all differences are statistically significant, using a Wilcoxon-signed-rank tests with $p \leq 0.05$.

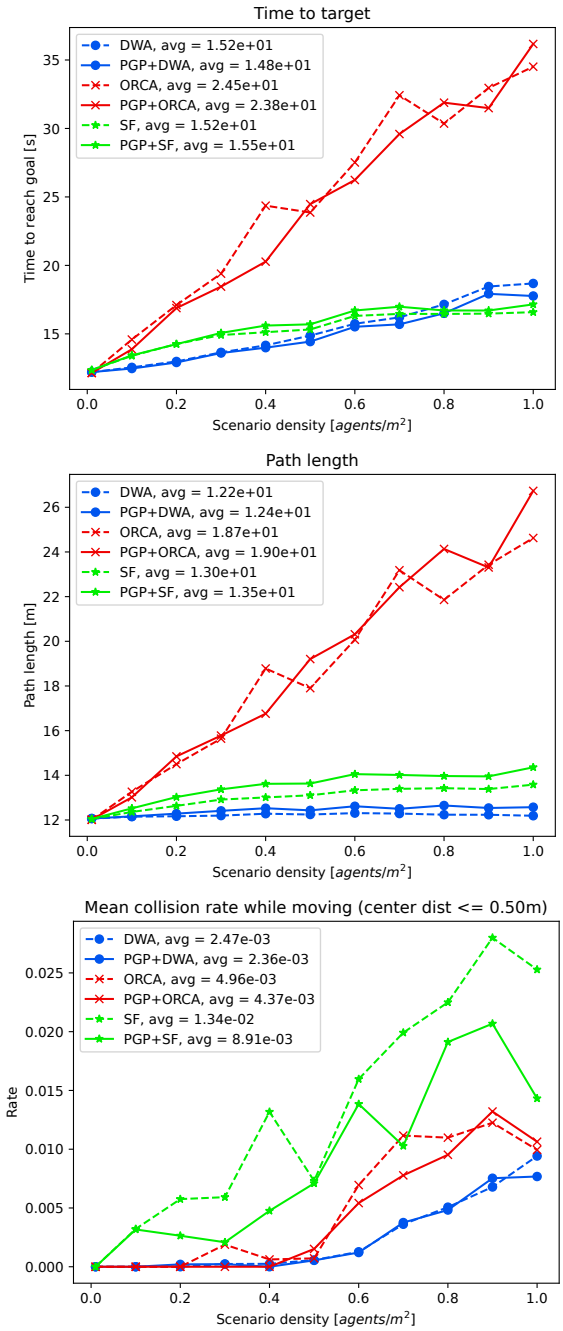


Fig. 4: Success metrics for open space scenario with increasing agent density

The performance metrics in Figure 4 show that all three (cooperative) collision avoidance algorithms (DWA, ORCA, SF) lead to very different behavior in the scenario. However, adding a PGP planner to the agent (solid lines) typicall improves the behavior of the agent compared to using the standalone (C)CA planners (dashed lines).

The DWA algorithm (blue lines, solid and dashed) deviates very little from the straight path towards the goal (middle plot). It shows the smallest collision rate across the three (C)CA algorithms, with only a moderate increase for the highest density scenarios (bottom plot).

With the addition of PGP, DWA is able to significantly reduce the average time to reach the goal from approximately 15.52 to 14.48 seconds across densities, with larger advantages for more crowded scenarios (top plot of Figure 4). At the same time, the overall average collision risk (where DWA already has the lowest total levels) decreases further from 0.25% to 0.24%.

The ORCA algorithm (red lines, solid and dashed) needs more time to reach the goal, compared to DWA and SF (top plot). This is especially true for denser crowds. In our simulations, ORCA agents tend to make large detours to avoid other agents, resulting in a significant increase in total path length with increasing density (middle plot). At the same time, ORCA shows a relatively low collision rate while moving (bottom plot).

For PGP+ORCA, the average change from 24.5 seconds to 23.8 seconds is not statistically significant. However, in terms of collision rate (bottom plot), PGP reduces the collision slightly for ORCA (0.44% vs 0.50%)

The SF agents (green lines, solid and dashed) also show a lower time-to-target than ORCA (top plot). For lower densities, SF agents are slower than DWA agents, but for the most crowded scenarios, they are slightly faster. Social Forces agents also stay mainly on-course, with a slightly longer path length compared to DWA (middle plot). Their fast pace, however, comes at the cost of a significantly increased collision rate compared to the other algorithms (bottom plot).

Combining the SF agents with PGP, the agent gets marginally slower (15.5s vs 15.2s). However, in terms of collision rate (bottom plot), PGP drastically reduces the collision while moving for the SF agent (0.89 % of time steps vs. 1.34 %).

Note that the reduction of collision risk (lower plot of Figure 4) only results from the biased goals for the (C)CA algorithms. The middle plot shows that, as expected, PGP significantly increases the average path length of the agents. This is not surprising, since without PGP, the (C)CA planners target is always the true goal, whereas PGP biases them to deviate from the straight line.

Next, we look at the quality/social metrics of the agents (Figure 5). As expected, all three (C)CA algorithms get closer to other agents in the high-density scenarios (left-hand plot). The Space Violation Rate while moving is lowest for ORCA (recall that it often takes a long way around), highest for SF, and in between for DWA. The average total Social Force on the ego agent (right-hand plot) across all time steps is highest for DWA, and significantly lower for SF and ORCA agents.

With respect to the quality/social metrics using PGP significantly improves both the average Space Violation Rate and the average total Social Force (right-hand side) across all (C)CA algorithms. This means, agents with PGP move into the personal space of other pedestrians significantly less often than without PGP. At the same time, they create less ”tension”, i.e., Social Force when moving through the denser groups of people.

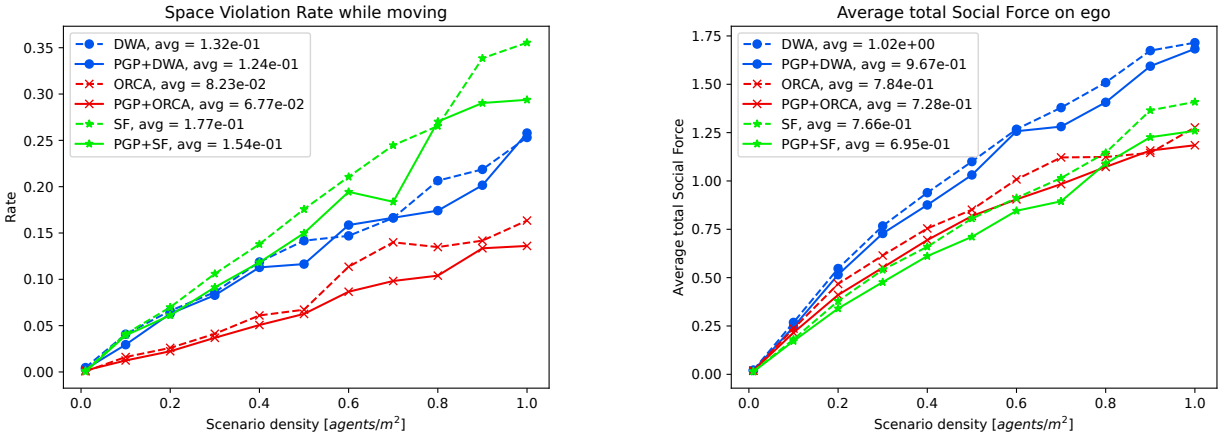


Fig. 5: Quality/social metrics for open space scenario with increasing agent density

In sum, using PGP to bias the (C)CA algorithms' goals significantly improves the quality of the behavior. At the same time, their performance increases in terms of collision avoidance, and, in the case of DWA, also in terms of time-to-target. This comes at the expense of slightly longer paths, and, for SF agents, at the cost of a marginally longer time-to-target.

## V. PGP on WaPOCHI mobile robot

We integrated PGP on the WaPOCHI robot of Honda R&D, an advanced prototype mobile robot. WaPOCHI is capable of navigating indoor and outdoor public spaces, leading and following registered users as well as carrying items.[6] The WaPOCHI platform offers a 360-degree field of view with stereo and monocular cameras, and it is equipped with an IMU, LiDAR bumpers, powerful computational hardware, and a mobile base. WaPOCHI perception in terms of static and dynamic objects is vision-based. It uses state-of-the-art machine learning models to detect dynamic and static objects. WaPOCHI's navigation stack contains interchangeable modules, connected by the system's ROS2 framework.

PGP is integrated in WaPOCHI as a ROS2 module, running in 10 Hz. It receives the robot's position and goal, its velocity and orientation, as well as positions and velocities for pedestrians and the boundaries of static objects. In each iteration, it outputs a subgoal to a DWA module for collision avoidance, which plans the final trajectory for the robot passed to the controller. We tested PGP in an indoor corridor scenario (see Figure 6). The robot's task was to navigate the corridors with a behavior that is not only collision-free, but foresighted and cooperative. In the tests, PGP improved the robots' behavior by biasing DWA's target towards areas with fewer predicted conflicts. In the example in Figure 6, it outputs a goal on the left-hand side of the corridor (blue star in right-hand plot) to pass behind the crossing pedestrian. This is consistent with the findings of the simulation study, where we see small but statistically significant improvements in space compliance rates for agents with PGP starting already at low densities. The proof-of-concept integration into WaPOCHI shows that PGP can improve a robot's behavior at relatively low additional computational cost, works with real-world data, and is conceptually simple to add to a mobile robotics platform.

[6] https://global.honda/en/stories/119-2402-hondaci.html

## VI. Conclusion

In this paper, we presented the Probabilistic Gap Planner (PGP), a novel trajectory-based long-term planner for Social Robot Navigation. PGP is based on the assumption that interacting pedestrians are, in general, cooperative. The approach decouples conflict avoidance (adjusting the motion behavior early on to traverse less dense areas) from cooperative collision avoidance (finding a collision-free, cooperative motion for an *imminent* interaction). PGP modifies an established probabilistic collision risk model to accommodate the assumption of cooperativity. It creates a set of prototypical trajectory variations leading to the global navigation goal. To evaluate them, PGP uses the so-called *survival analysis* of the risk model to weigh the expected future rewards with the probability of reaching them. We run extensive simulations with over 6.600 runs. We show that PGP guides the agents with state-of-the-art (C)CA algorithms to less crowded areas. Using PGP significantly improves the established quality/social metrics *Space Compliance* and average total *Social Force*. At the same time, using PGP improves the agent's performance in terms of their collision rate, and, in the case of DWA, also the time-to-target. Typically, these gains come at the expense of a slightly longer total path length, and the (limited) additional computational effort to run PGP. We integrated PGP on the Honda R&D WaPOCHI mobile robot, running in real-time at 10 Hz in a ROS2 environment. First tests with the robot show that PGP is able to guide the robot towards areas with less pedestrians in a corridor scenario. A promising area of research is improving PGP's trajectory space exploration strategy. Since the simple set of trajectory variations used in this paper already leads to significant improvements in agent behavior, using explorative sampling techniques or a search-based approach could enable the agents to find even better solutions.



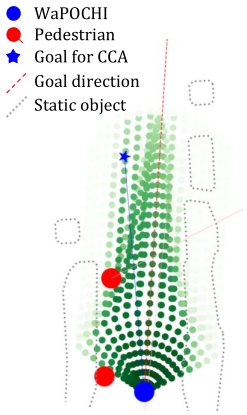


Fig. 6: Left: WaPOCHI mobile robot at Honda R&D, Tokyo, in a corridor scenario with PGP planner. The robot's map goal is at the end of the corridor. Right: Visualization of sensed input to PGP (grey: static objects, red: pedestrians), PGP's planning (green dots representing different evaluated trajectories, light color indicating low survival probability), and PGP output (blue star and dotted line indicating short-term goal direction to DWA). PGP biases the (C)CA planner towards the left side of the corridor (blue star in right plot), since the person wearing black is predicted to move to the right side.

## REFERENCES


[1] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. n. Steinfeld, and J. Oh, "Core Challenges of Social Robot Navigation: A Survey," *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, pp. 1–39, 2023.

[2] A. Francis, C. Pérez-D'Arpino, C. Li, F. Xia, A. Alahi, R. Alami, A. Bera, A. Biswas, J. Biswas, R. Chandra, H.-T. L. Chiang, M. Everett, S. Ha, J. Hart, J. P. How, H. Karnan, T.-W. E. Lee, L. J. Manso, R. Mirksy, S. Pirk, P. T. Singamaneni, P. Stone, A. V. Taylor, P. Trautman, N. Tsoi, M. Vázquez, X. Xiao, P. Xu, N. Yokoyama, A. Toshev, and R. Martín-Martín, "Principles and Guidelines for Evaluating Social Robot Navigation Algorithms," *ACM Transactions on Human-Robot Interaction*, vol. 14, no. 2, pp. 1–65, 2025.

[3] S. Macenski, T. Moore, D. V. Lu, A. Merzlyakov, and M. Ferguson, "From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2," *Robotics and Autonomous Systems*, vol. 168, p. 104493, 2023.

[4] M. M. Sun, F. Baldini, K. Hughes, P. Trautman, and T. Murphey, "Mixed Srategy Nash Equilibrium for Crowd Navigation," *The International Journal of Robotics Research*, publisher: SAGE Publications Sage UK: London, England.

[5] P. T. 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)*. IEEE, 2021, pp. 5817–5824.

[6] H. Khambhaita and R. Alami, "Viewing Robot Navigation in Human Environment as a Cooperative Activity," in *Robotics Research: The 18th International Symposium ISRR*. Springer, 2020, pp. 285–300.

[7] R. Mirsky, X. Xiao, J. Hart, and P. Stone, "Conflict Avoidance in Social Navigation—a Survey," *ACM Transactions on Human-Robot Interaction*, vol. 13, no. 1, pp. 1–36, 2024.

[8] J. Eggert and T. Puphal, "Continuous Risk Measures for ADAS and AD," in *Future Active Safety Technology Symposium*, 2017.

[9] P. Varaiya, "Smart Cars on Smart Roads: Problems of Control," *IEEE Transactions on automatic control*, vol. 38, no. 2, pp. 195–207, 1993.

[10] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[11] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A Review of Motion Planning for Highway Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2019.

[12] K. Kant and S. W. Zucker, "Planning collision-free trajectories in time-varying environments: a two-level hierarchy," *The Visual Computer*, vol. 3, pp. 304–313, 1988.

[13] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 57–66, 2020.

[14] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to C]ollision Avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

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

[16] V.-A. Le, B. Chalaki, V. Tadiparthi, H. N. Mahjoub, J. D'sa, and E. Moradi-Pari, "Social Navigation in Crowded Environments with Model Predictive Control and Deep Learning-Based Human Trajectory Prediction," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 4793–4799.

[17] 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," *The International Journal of Robotics Research*, vol. 43, no. 10, pp. 1533–1572, 2024.

[18] P. Trautman and A. Krause, "Unfreezing the Robot: Navigation in Dense, Interacting Crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[19] J. Eggert, "Predictive Risk Estimation for Intelligent ADAS Functions," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 711–718.

[20] F. Damerow and J. Eggert, "Risk-Aversive Behavior Planning under Multiple Situations with Uncertainty," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 656–663.

[21] T. Puphal, M. Probst, and J. Eggert, "Probabilistic Uncertainty-Aware Risk Spot Detector for Naturalistic Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 406–415, 2019.

[22] M. Probst, R. Wenzel, T. Puphal, M. Komuro, T. H. Weisswange, N. Steinhardt, B. Bolder, B. Flade, Y. Sakamoto, Y. Yasui *et al.*, "Automated Driving in Complex Real-World Scenarios using a Scalable Risk-Based Behavior Generation Framework," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 629–636.

[23] J. Wang, W. P. Chan, P. Carreno-Medrano, A. Cosgun, and E. Croft, "Metrics for Evaluating Social Conformity of Crowd Navigation Algorithms," in *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2022, pp. 1–6.

[24] D. Helbing and P. Molnar, "Social Force Model for Pedestrian Dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[25] J. J. Fruin, *Designing for Pedestrians: A Level of Service Concept*. Polytechnic University.

[26] Y. Gao and C.-M. Huang, "Evaluation of Socially-Aware Robot Navigation," *Frontiers in Robotics and AI*, vol. 8, p. 721317, 2022.