Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals

Lucas Streichenberg[1,2,*], Elia Trevisan[1,*], Jen Jen Chung[2,3], Roland Siegwart[2] and Javier Alonso-Mora[1]

***Abstract*— Autonomous vehicles that operate in urban environments shall comply with existing rules and reason about the interactions with other decision-making agents. In this paper, we introduce a decentralized and communication-free interaction-aware motion planner and apply it to Autonomous Surface Vessels (ASVs) in urban canals. We build upon a sampling-based method, namely Model Predictive Path Integral control (MPPI), and employ it to, in each time instance, compute both a collision-free trajectory for the vehicle and a prediction of other agents' trajectories, thus modeling interactions. To improve the method's efficiency in multi-agent scenarios, we introduce a two-stage sample evaluation strategy and define an appropriate cost function to achieve rule compliance. We evaluate this decentralized approach in simulations with multiple vessels in real scenarios extracted from Amsterdam's canals, showing superior performance than a state-of-the-art trajectory optimization framework and robustness when encountering different types of agents.**

I. Introduction

With rising population density, cities are forced to enhance their mobility and transportation strategies. The City of Amsterdam aims to reduce the load on road infrastructure by transporting goods and people on the urban waterways [1]. This presents a great opportunity to operate Autonomous Surface Vessels (ASVs) such as Roboat [2] in urban canals. However, this is a very technically challenging task due to the complex and dynamic nature of the environment. Narrow canals, complex dynamics, static obstacles, and human-piloted vessels must be dealt with while obeying existing canal regulations [3]. Model Predictive Path Integral Control (MPPI) [4] offers a parallelizable sampling-based framework for solving motion planning tasks with such complex dynamics and discontinuous costs as those exhibited in our domain. Unlike methods based on constrained optimization, which need to rely on convex approximations of the free space and on inflating the ego and obstacle agents into ellipsoidal shapes for collision avoidance [5], MPPI can account for the exact and potentially non-convex shape of both static and dynamic obstacles. This promises to be a significant advantage in tight interaction-rich environments. Still, another important aspect of achieving safe and efficient navigation in crowded spaces is accounting for cooperation [6]. For these reasons, we propose a method to decentralize MPPI in order to navigate among non-communicating agents while providing interaction awareness and generating cooperative motion plans. We introduce awareness of navigation rules through discontinuous costs. Moreover, the proposed method can run in real-time thanks to our two-stage sample evaluation strategy and CPU parallelization.

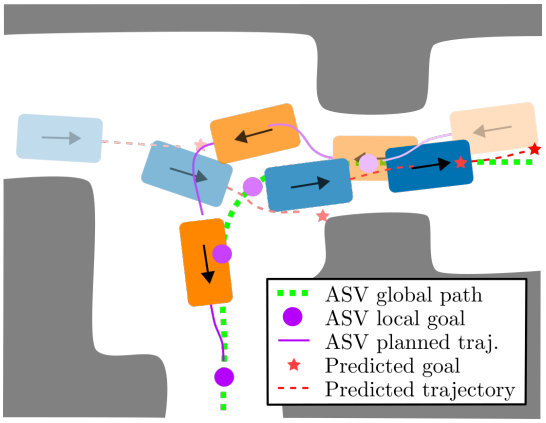


Fig. 1: An ASV running our method (orange) encounters a non-communicating vessel (blue). At each time step, the ASV is given its current position and a global path (dashed green line). Based on this, the ASV sets a local goal (purple circle) in front of itself on the global path. The ASV is also given the position and velocity of the non-communicating vessel. With these, the ASV estimates the local goal of the non-communicating vessel (red star) assuming constant velocity. Then, the ASV plans input sequences over a defined horizon resulting in trajectories for both vessels.

A. Related Work

Cooperative and interactive motion planning for robotics is a challenging problem with a vast literature [7].

The most well-known examples for planning in dynamic environments are the Dynamic Window Approach (DWA) [8], Reciprocal Velocity Obstacle (RVO) [9], its extension Optimal Reciprocal Collision Avoidance (ORCA) [10], and Artificial Potential Fields (APF) [11], [12]. While these methods are highly efficient, they often lead to reactive behaviors.

Model-free reinforcement learning algorithms have been successfully trained in simulation with hand-crafted reward functions to navigate among human crowds [13], [14], but generalization and collision avoidance are not guaranteed.

Game theoretic approaches have been implemented in the context of autonomous cars to perform lane changes [15], merging [16] and to solve unsignalized intersections [17]. However, they rely on a coarse discretization of the action space and do not scale well with the number of agents.

This research is supported by the project "Sustainable Transportation and Logistics over Water: Electrification, Automation and Optimization (TRiLOGy)" of the Netherlands Organization for Scientific Research (NWO), domain Science (ENW), and the Amsterdam Institute for Advanced Metropolitan Solutions (AMS) in the Netherlands.

*These authors have contributed equally.

[1]Cognitive Robotics Department, TU Delft, {e.trevisan, j.alonsomora}@tudelft.nl

[2]Autonomous Systems Lab, ETH Zurich {stlucas, rsiegwart}@ethz.ch

[3]School of ITEE, The University of Queensland jenjen.chung@uq.edu.au

Model Predictive Control (MPC) based on trajectory optimization is a popular approach when it comes to local motion planning. In a multi-agent setting, however, the actions of the other agents are required to proceed with the motion planning of the ego-agent. In [18] a distributed MPC was developed for motion planning with multiple drones relying on ideal communication. A way to avoid communication is to estimate each agent's state and predict their future motion using, for example, a constant velocity model [5], [19] or learning-based techniques [20]. However, as long as we first predict and then plan, large parts of the state-space can be perceived as unsafe by the motion planner [21]. This can be overcome by modeling interaction, such that the ego-agent is aware that its actions can influence the actions of the other agents around it. Unfortunately, accounting for such a model while planning with constrained optimization techniques can become computationally expensive [22].

In contrast to optimization-based methods, MPPI solves for the best control trajectory at each step by forward simulating the behavior of the full system. To achieve this, MPPI uses a parallelizable sampling-based framework [23] to rollout simulations, allowing it to find an approximate solution to non-linear, non-convex, discontinuous Stochastic Optimal Control (SOC) problems [4], [24]. Compared to other SOC methods such as iterative Linear Quadratic Gaussian (iLQG) [25] or Differential Dynamic Programming (DDP) [26], MPPI does not require linearization of the system dynamics or quadratic approximation of the cost function. This makes MPPI particularly well-suited to our target task of ASV navigation in urban canals since the regulation-based interactions explicitly give rise to non-differentiable costs. Moreover, MPPI's fast parallelizable computations could solve interaction-aware motion planning problems while still running in real-time. When deployed to centralized multi-agent systems, however, the classic MPPI approach shows a significant increase in the number of samples required with an increasing number of agents [27].

B. Contributions

We propose an interaction-aware motion planning method based on MPPI which can generate cooperative plans in environments with non-communicating agents accounting for the full dynamics of the system and the exact shapes of the obstacles. To summarize our contributions:

- We propose a decentralized architecture that can operate with limited or no communication under the assumption that other agents' states can be sensed exactly and that all the agents in the environment behave rationally.
- To reduce the number of samples required to plan for multi-agent systems, we propose a two-stage sample evaluation technique that improves sample efficiency.
- We formulate the objectives of the navigation task into an appropriate cost function to achieve rule compliance.

To demonstrate the performance of our method, we compare it to a state-of-the-art regulations-aware optimization-based MPC [5] in several simulated experiments set up in real sections of Amsterdam's canals. The proposed decentralized MPPI is also compared to the centralized version in environments with crowds of up to four interacting vessels. Finally, we demonstrate the robustness of the algorithm in scenarios where a human-driven vessel does not behave rationally and provide insights into the computation times.

II. Preliminaries

We start by describing the basic MPPI algorithm. Then, since MPPI relies on a model of the system to perform the simulated rollouts, we also define the relevant dynamics that describe the behavior of our multi-ASV system.

A. MPPI Algorithm

The presented work is based on the MPPI derivations by [24]. With this method, we can solve SOC problems for discrete-time dynamical systems of the form,

$$\mathbf{q}_{t+1} = \mathcal{F}(\mathbf{q}_t, \tilde{\mathbf{u}}_t), \quad \tilde{\mathbf{u}}_t \sim \mathcal{N}(\mathbf{u}_t, \mathbf{\Sigma}), \tag{1}$$

with state $\mathbf{q}$, time step t, nonlinear state transition function $\mathcal{F}$ and noisy input $\tilde{\mathbf{u}}$ with variance $\mathbf{\Sigma}$ and mean $\mathbf{u}$, where $\mathbf{u}$ is the input we command to the system. The algorithm samples K input sequences $\tilde{U}_k$, $k \in [1, K]$ from a distribution $\mathcal{N}(\mathbf{u}_t, \nu\mathbf{\Sigma})$ (with scaling parameter ν) [28] and simulates them into K state trajectories Q_k over a horizon T as,

$$Q_k = \left[\mathbf{q}_0, \mathcal{F}(\mathbf{q}_0, \tilde{\mathbf{u}}_{k,0}), \ldots, \mathcal{F}(\mathbf{q}_{k,T-1}, \tilde{\mathbf{u}}_{k,T-1})\right]. \tag{2}$$

Each sample is rated by computing the total cost S_k, which includes a stage cost and a terminal cost. *Importance sampling* weights w_k are then computed based on the cost of the sample k minus the minimum sampled cost S_{min} as,

$$w_k = \frac{1}{\eta} \exp\left(\frac{-1}{\lambda}(S_k - S_{min})\right), \quad \sum_{k=0}^{K-1} w_k = 1, \tag{3}$$

with normalization factor η and tuning parameter λ. The resulting control input sequence U^*, which approximates the optimal control input sequence, is computed with,

$$U^* = \sum_{k=0}^{K-1} w_k \tilde{U}_k. \tag{4}$$

Then, the first input u_0^* of the computed sequence U^* is applied to the system.

B. Vessel State and Dynamics

We define the multi-agent state similarly to [27]. The state of agent i is defined as the concatenation of its position, heading, and associated velocities,

$$\mathbf{q}_i = \begin{bmatrix}\mathbf{x}_i^\top & \mathbf{v}_i^\top\end{bmatrix}^\top = \begin{bmatrix}x_i & y_i & \phi_i & \dot{x}_i & \dot{y}_i & \dot{\phi}_i\end{bmatrix}^\top. \tag{5}$$

The full system state is then formed by stacking the individual states of each of the agents in the set $\mathcal{M} = \{0, 1, ..., m\}$,

$$\mathbf{q} = \begin{bmatrix}\mathbf{q}_0^\top & \mathbf{q}_1^\top & \ldots & \mathbf{q}_m^\top\end{bmatrix}^\top. \tag{6}$$

The ASV we use is modeled as a nonlinear second order system [29]–[31]. Since the vessel sails at low speeds, we discard Coriolis and centripetal effects. Therefore,

$$\begin{aligned}\dot{\mathbf{x}}_i &= \mathbf{R}(\phi_i)\,\mathbf{v}_i, \\ \dot{\mathbf{v}}_i &= \mathbf{M}^{-1}\left(\mathbf{B}\mathbf{u}_i - \mathbf{D}(\mathbf{v}_i)\,\mathbf{v}_i\right),\end{aligned} \tag{7}$$

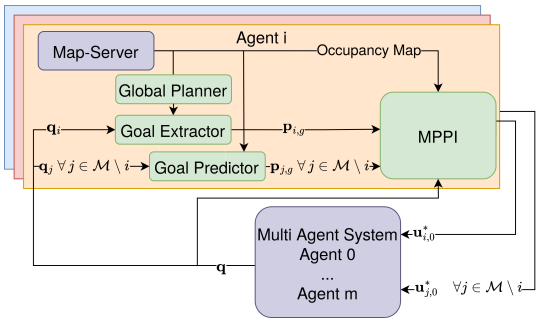


Fig. 2: Overview of the framework: At every time step, each agent receives the state of all the agents in the environment. With this, we extract a local goal for the ego-agent (agent i in figure) from the global planner and predict local goals for the other agents. We then solve the planning task with MPPI as if it was centralized, planning a control input sequence and corresponding trajectory for each agent. We then apply the first input $u^*_{i,0}$ of the sequence to the ego-agent.

where $\mathbf{R}(\phi_i)$ is the rotation matrix from body to inertial frame, $\mathbf{M}$ is the mass matrix, $\mathbf{B}\mathbf{u}_i$ are the applied forces (thrust) and $\mathbf{D}(\mathbf{v}_i)$ is the drag matrix.

C. Global Planning

To help navigate large maps, we provide the ego-agent with a global path. Such a path is generated via the ROS navigation stack with its path planning plugin [32]. Instead of directly tracking this global path, we look for a local goal $\mathbf{p}_g$ on the global path at a given look-ahead distance $r_{\mathbf{p}_g}$ (Fig. 5). Compared to just rigorously tracking the global path, this local goal approach gives the local planner more freedom to perform collision avoidance and other maneuvers.

III. Decentralized Interaction-aware Model Predictive Path Integral Control

In the following we outline the proposed architecture, state the changes to the classic MPPI framework and present the regulation-aware cost function along with the local goal prediction used for the decentralized computation of the cost.

A. Approach and Architecture

Our decentralized MPPI approach relies on each agent running its own MPPI solver for its local multi-agent system to anticipate the actions of other agents (see Fig. 2). That is, for agent i, the MPPI state and control output are defined as,

$$\begin{aligned}\mathbf{q}^i &= \begin{bmatrix}\mathbf{q}_i^\top & \mathbf{q}_j^{i\top}\end{bmatrix}^\top,\\ \mathbf{u}^i &= \begin{bmatrix}\mathbf{u}_i^\top & \mathbf{u}_j^{i\top}\end{bmatrix}^\top,\end{aligned} \quad \forall j \in \mathcal{M} \setminus i, \tag{8}$$

where $(.)^i_j$ signifies a variable that agent i estimates of agent j. In the centralized case described in Section II-A, the system state is fully observable and the inputs computed by the central controller will be those executed by each agent. When we move to the decentralized case, the positions and velocities of other agents must be communicated or observed. Furthermore, while each agent samples control actions for all other agents in the MPPI rollouts, at execution time, there is no guarantee that other agents will behave accordingly. To focus on the decentralized coordination problem, we make a few assumptions. First, we assume noise-free observations

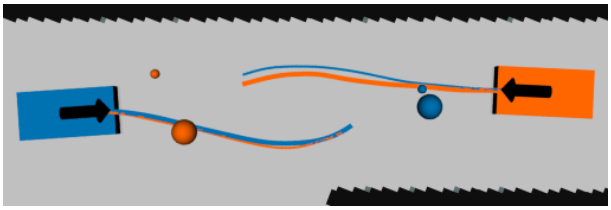


Fig. 3: Screenshot of two agents running our decentralized method with no communication inside our simulator. The left (blue) agent is given a local goal (large blue ball), its state, and the state of the other (orange) agent, based on which it predicts a local goal (small orange ball) for the obstacle vessel. The blue agent plans a sequence of inputs for both itself and the obstacle agent resulting in two trajectories, depicted by the blue paths. In orange, we can see the other agent applying the same algorithm. Note that even though we use constant velocity to predict the goal of the obstacles, both agents plan cooperation in the collision avoidance.

of the positions and velocities of other agents, i.e. $\mathbf{q}^i_j = \mathbf{q}_j$. Second, we assume that all agents behave rationally and that they are minimizing the same global cost. We later show in our experiments that the controller is still able to perform well when this assumption is violated. Third, in our experiments we only consider scenarios with homogeneous agents, meaning that they all have the same dynamics. This third assumption is not required in general as considering different dynamical models for different agents is possible as long as models are known. Fig. 3 shows a simulated encounter between two ASVs running our decentralized algorithm without communication.

B. Two-stage Sample Evaluation

With an increasing number of agents, it is increasingly likely for at least one agent to collide with a static obstacle in most rollouts. In the classical implementation of MPPI, this leads to most rollouts receiving a high cost and being effectively rejected, resulting in a very low sample efficiency. Therefore we propose to decouple the sampling into two stages as shown in Algorithm 1. Control-samples $U_{j,k}$ are evaluated in parallel for every agent $j \in \mathcal{M}$, predicting the set of individual trajectories $Q_{j,k}$ with agent-centric costs $S_{j,k}$ (eq. 9). At this point all samples with cost larger than the collision penalty $C_{\text{collision}}$ are immediately discarded (lines 3 and 4 of Algorithm 1). To build the expected number of system samples K we sample uniformly from the remaining non-colliding samples of each agent and unify these into full system samples, i.e. by stacking as in eq. (6). For each system sample Q_k the complete configuration cost S_k is evaluated by adding the stored agent-centric costs $S_{j,k}$ with any costs arising from collisions between vessels and regulation violations (line 14, Algorithm 1).

C. Cost Formulation

The sample cost $S_k, \forall k \in [1, K]$ evaluation is split into agent-centric and configuration costs. Both are considered instantaneous costs and are evaluated for every time-step t within the horizon T and each sample k. The **agent-centric** cost $S_{j,k,t}$ (in the following S_{agent}) is evaluated for agent j for sample k and defined as,

$$S_{\text{agent}} = C_{\text{static}} + C_{\text{rotation}} + C_{\text{tracking}} + C_{\text{speed}} + C_{\text{sample}}. \tag{9}$$

Algorithm 1: Decentralized MPPI for agent i (agent-specific superscripts are dropped for clarity)

Require: U ▷ previous control sequence (hot start)
Require: $\mathbf{q}$ ▷ current system state
1: $\mathbf{p}_{i,g} \leftarrow$ *receiveEgoLocalGoal*()
2: $\mathbf{p}_{j,g} \leftarrow$ *predictLocalGoal*$(\mathbf{q}_j)$ ▷ $\forall\, j \in \mathcal{M} \setminus i$
3: **for** each agent $j \in \mathcal{M}$ **do** ▷ independent rollouts
4: **for** each sample k **do**
5: $\mathcal{E}_{j,k} = [\epsilon_0, \ldots, \epsilon_{T-1}]$ ▷ $\epsilon_t \in \mathcal{N}(0, \nu\mathbf{\Sigma})$
6: $\tilde{U}_{j,k} = U_j + \mathcal{E}_{j,k}$
7: $Q_{j,k} \leftarrow$ *simulateSystem*$(\mathbf{q}_j, \tilde{U}_{j,k})$
8: $S_{j,k} \leftarrow$ *getIndividualCost*$(Q_{j,k}, \tilde{U}_{j,k}, \mathbf{p}_{j,g})$
9: **if** $S_{j,k} > C_{\text{collision}}$ **then**
10: *discardSample*$(Q_{j,k}, \tilde{U}_{j,k}, \mathcal{E}_{j,k}, S_{j,k})$
11: Uniformly sample from the remaining valid input sequences to rebuild K full system samples
12: **for** $k = 1 : K$ **do**
13: $\mathcal{S}_k = \sum_{j \in \mathcal{M}} S_{j,k} +$ *getConfigurationCost*(Q_k)
14: $[w_1, \ldots, w_K]$ = *importanceSampling*$([\mathcal{S}_0, \ldots, \mathcal{S}_K])$
15: return $U^* = U + \sum_{k=1}^{K} w_k \mathcal{E}_k$

C_{static} returns a constant penalty $C_{\text{collision}}$ if the vessel enters occupied space and C_{rotation} is based on a linear penalty for rotation velocities ($k_{\text{rot, slow}}$ for velocities $||\mathbf{v}||_2 < 0.5$m/s, k_{rot} otherwise). The tracking cost is,

$$C_{\text{tracking}} = k_{\text{tracking}} \frac{||\mathbf{p}_g - \mathbf{p}_t||_2}{||\mathbf{p}_g - \mathbf{p}_{t_0}||_2}, \tag{10}$$

where k_{tracking} is a scaling factor, $\mathbf{p}_g$ is the agent's local goal, $\mathbf{p}_t$ is the predicted agent position at time t and $\mathbf{p}_{t_0}$ is the vessel position at the start of the prediction horizon. C_{speed} is a constant penalty applied when the current speed is higher than the maximum speed. The sample cost is given by,

$$C_{\text{sample}}(\mathbf{u}_t, \epsilon_t) = \frac{1}{2}\gamma[\,\mathbf{u}_t^T \Sigma^{-1} \mathbf{u}_t + 2\mathbf{u}_t^T \Sigma^{-1} \epsilon_t], \tag{11}$$

with γ as a tuning parameter. The **configuration cost** $S_{k,t}$ (in the following $S_{\text{configuration}}$) is evaluated for every timestep t within the horizon T for every sample k and combines dynamic collisions and regulation violations,

$$S_{\text{configuration}} = C_{\text{dynamic}} + C_{\text{regulation}}. \tag{12}$$

Dynamic collisions are defined as those between multiple vessels and are penalized with the same constant $C_{\text{collision}}$ and the regulation cost $C_{\text{regulation}}$ is derived from the two main traffic rules (i) *avoiding to the right* in head-on encounters and (ii) *right of way* for crossing scenarios similar to COLREGs [33]. Regulation compliance is determined by a relative position and relative velocity check. Regarding the position, we check if there is a vessel with significant velocity ($||\mathbf{v}|| > 0.5m/s$) on starboard side within a given radius (Fig 4). Regarding the relative velocities, we evaluate if another vessel approaches from the right by,

$$||\mathbf{v}_i \times \mathbf{v}_j|| < ||\mathbf{v}_i||\,||\mathbf{v}_j|| \sin(-\frac{\pi}{2} + \delta), \tag{13}$$

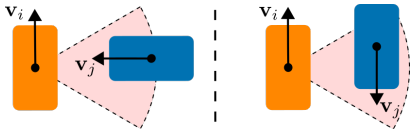


Fig. 4: Configurations considered as regulation violations. Left: Not giving right of way to a vessel approaching from starboard. Right: Avoiding an oncoming vessel to the left.

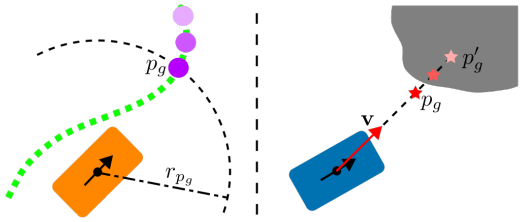


Fig. 5: Left: Extraction of the local goal of the ego vessel is performed by searching the global path backward until the goal position is within a radius r_{p_g} from the center of the vessel. Right: Local goal prediction for other agents is performed using a constant velocity model, then projected into unoccupied space if the goal is in collision with static obstacles.

where δ defines the angular margin, and if the other vessel is approaching the ego-vessel head-on via,

$$||\mathbf{v}_i \cdot \mathbf{v}_j|| < ||\mathbf{v}_i|| \cdot ||\mathbf{v}_j|| \cos(\pi + \delta). \tag{14}$$

Therefore if we detect a vessel on starboard side and the velocities satisfy (13), we consider the ego-agent as breaking the right-of-way rule (Fig. 4 left). If instead, we detect a vessel on starboard side with opposite velocity, we consider it as passing on the right (Fig. 4 right).

D. Local Goal Prediction

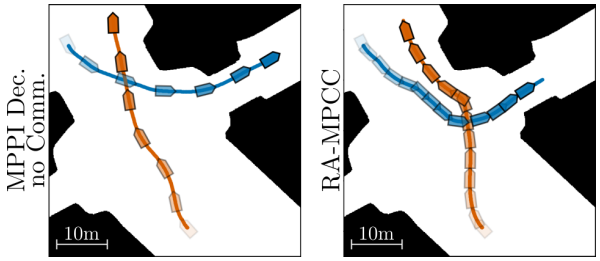
The proposed decentralized version of interaction-aware MPPI requires estimating the local goals for all non-ego vessels (line 2, Algorithm 1). We use a constant velocity model such that agent i estimates the goal of agent j as,

$$\mathbf{p}^i_{j,g} = k_s\, T\, \delta T\, \mathbf{R}\left(\phi^i_j\right) \mathbf{v}^i_j + \mathbf{p}^i_j, \tag{15}$$

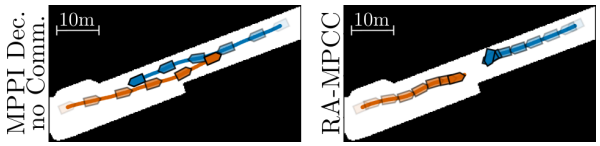
with k_s as a scaling factor and δT as step size. If the predicted goal is in collision, we project back along the vector towards $\mathbf{p}^i_j$ and choose the first unoccupied point as shown in Fig. 5.

IV. Experiments

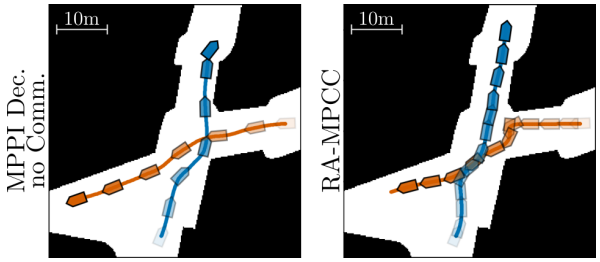
We perform extensive experiments in several maps taken from real canal sections of Amsterdam, namely the Herengracht (HG), Prinsengracht (PG), Bloemgracht (BG), and the intersection between Bloemgracht and Lijnbaansgracht (BGLG). In all experiments, the dimensions of both the map and the vessel are represented faithfully. In Section IV-A we compare our method in two-agent scenarios with an optimization-based MPC, in Section IV-B we test our method in interaction-rich four-agent scenarios, in Section IV-C we demonstrate the robustness to non-rational human-driven agents, while in Section IV-D we discuss the computation times of the proposed method. All experiments running MPPI use a horizon T of 100 time-steps with step-size $\delta T = 0.1s$, input variance $\Sigma = \text{diag}(0.5, 0.5, 0.01, 0.01)$ and exploration scaling factor $\nu = 12$. We use $K = \{2000, 6000\}$



(a) *Prinsengracht*. MPPI understands that the blue vessel can safely cross in front of the vessel with the right of way (orange) without slowing it down. RA-MPCC is not confident and ends up blocking the way, pushing the orange vessel out of its route.



(b) *Bloemgracht*. The MPPI agents cooperate to perform collision avoidance, while RA-MPCC ends up in a deadlock.



(c) *Bloemgracht-Lijnbaansgracht*. The MPPI agents correctly solve the crossing according to the right of way. RA-MPCC, not understanding interactions, deems much of the state-space as occupied therefore steering left and right. Eventually, the agent with the right of way has to stop and pass behind, violating the navigation rules.

Fig. 6: Comparisons between the decentralized MPPI without communications and RA-MPCC. Vessels on the figures are to scale (4m long) and are plotted every 5 seconds. Results for 100 runs are summarized in Table I.

samples for two- and four-agent scenarios, respectively. Each version of the MPPI shown in the experiments (centralized, decentralized, decentralized with no communications) uses our proposed two-stage sampling technique.

A. Comparison with Optimization-based MPC

We compare our method to a state-of-the-art optimization-based decentralized motion planner designed for ASVs in urban canals, namely the Regulations Aware Model Predictive Contouring Controller (RA-MPCC) [5]. The three scenarios on which we compare are an unprotected left turn (Fig. 6a), a head-on encounter (Fig. 6b) and a crossing (Fig. 6c). These three scenarios were then run 100 times with randomized initial conditions and global goals using the proposed centralized, decentralized, and decentralized with no communication MPPI as well as the RA-MPCC. For all controller types, the randomization was kept equal (i.e. same random seed). Table I summarizes the results, where we compare the number of runs that ended successfully, in a deadlock, or in a collision. Of the runs that ended successfully, we report the number of rule violations. We also report the average time to complete the scenario, defined as the moment in which all agents reach their goal, and the total average distance, which is the sum of the average distances traveled by all agents.

All controllers were encouraged through the cost function to keep a velocity around the speed limit (around 1.7m/s). From the table, however, it stands out that the RA-MPCC navigates much slower and therefore has much longer arrival times. This is both because of how its cost function is defined, but also because the RA-MPCC has to plan within convex obstacle-free areas, which can sometimes be quite small and slow down the pace. Instead, MPPI considers the exact occupancy map without any need for pre-processing.

While it is easy to give a discontinuous penalty to the MPPI whenever a sample violates a navigation rule, the RA-MPCC has to use continuous cost functions to encourage rule compliance. This, however, inadvertently introduces repulsive forces between the two agents, which then tend to push each other into corners, which is the main cause of deadlocks in the Prinsengracht and the Bloemgracht-Lijnbaansgracht scenarios. In the Bloemgracht head-on encounter, the RA-MPCC gets to its destination only about half of the time. Given that the RA-MPCC has to inflate the ego-agent in a set of circles, the obstacle agent into an ellipsoid, and the static obstacle map has to be pre-processed into convex regions, there is barely enough space to pass in the most narrow section of the canal. This, combined with the lack of understanding that the two agents can cooperate to solve the maneuver, leads to a large number of deadlocks.

Collisions with the RA-MPCC instead happen for two reasons. Number one, the method first approximates the static obstacle with polygons, which are then decomposed into convex shapes, to which we can then find linear constraints by solving a quadratic program. However, there is no guarantee that the polygons contain all of the original obstacle. Safety margins are added, but margins too large means that some narrow canals are simply impossible to navigate. Number two, the optimization can often just fail, especially in more difficult and risky situations. When this happens, the algorithm just applies zero input, and if the boat has enough momentum it can drift into a collision.

Moreover, while our interaction-aware MPPI could plan with a horizon of 100 steps, the RA-MPPC could only plan 20 steps to meet the 10Hz control loop.

TABLE I: Results for 100 runs of the experiments seen in Fig. 6 with randomized initial conditions and goals.

	Method	Successes-Deadlocks-Collisions	Rule Violations	Average Time	Total Average Distance
PG	Centralized	100 - 0 - 0	2	24.65s	82.15m
	Dec. Comm.	100 - 0 - 0	2	24.64s	82.14m
	Dec. No Comm.	100 - 0 - 0	2	25.12s	82.54m
	RA-MPCC	95 - 5 - 0	5	51.81s	73.20m
BG	Centralized	100 - 0 - 0	0	22.23s	74.93m
	Dec. Comm.	100 - 0 - 0	0	22.24s	74.79m
	Dec. No Comm.	100 - 0 - 0	0	22.18s	75.01m
	RA-MPCC	52 - 37 - 11	4	55.93s	67.64m
BGLG	Centralized	100 - 0 - 0	2	28.47s	87.24m
	Dec. Comm.	100 - 0 - 0	2	28.83s	88.26m
	Dec. No Comm.	100 - 0 - 0	2	29.01s	88.30m
	RA-MPCC	74 - 23 - 3	38	54.46s	84.20m

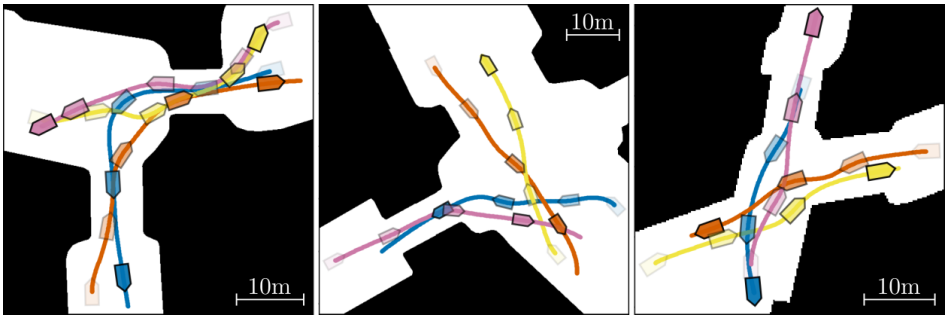


Fig. 7: Navigation among four autonomous vessels running decentralized MPPI without communication. Left: Narrow intersection with a bridge in Herengracht. Middle: Wide intersection with bridges in Prinsengracht. Right: Very narrow intersection in Bloemgracht. Vessels are plotted every 6 seconds.

TABLE II: Results for 20 runs of the experiments seen in Fig. 7 with randomized initial conditions and goals.

	Method	Successes-Deadlocks-Collisions	Rule Violations	Average Time	Total Average Distance
HG	Centralized	20 - 0 - 0	0	24.46	204.42
	Dec. Comm.	20 - 0 - 0	0	26.82	214.64
	Dec. No Comm.	20 - 0 - 0	3	33.18	225.48
PG	Centralized	20 - 0 - 0	0	23.24	214.92
	Dec. Comm.	20 - 0 - 0	0	23.82	216.99
	Dec. No Comm.	20 - 0 - 0	0	23.43	214.71
BGLG	Centralized	20 - 0 - 0	1	18.36	143.14
	Dec. Comm.	20 - 0 - 0	2	20.87	149.78
	Dec. No Comm.	19 - 0 - 1	0	20.13	144.86

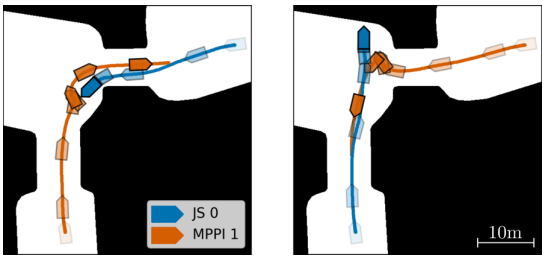


Fig. 8: Qualitative robustness evaluation for non-cooperative vessels. Left: The joystick-driven vessel (blue) avoids to the wrong side. The MPPI vessel (orange) avoids collisions by coming to a complete stop, then continues to the left. Right: The joystick-driven vessel (blue) blocks the MPPI, disregarding right of way. The MPPI agent (orange) avoids collisions, comes to a stop, then continues on its way. Vessels are plotted every 4 seconds.

B. Navigation in Crowded Environments

The proposed method is also capable of resolving scenarios with more than two agents. In Table II we show the results for 20 runs in crowded environments with four agents. The experiments are run in the maps shown in Fig. 7, where the vessels are exposed to many encounters in very narrow spaces. The results show that the centralized and decentralized method with shared local goals (with communication) can resolve the task successfully while behaving cooperatively. The decentralized version of our approach with no communication (thus predicting goals for other vessels) incurs in a collision in the tight Bloemgracht's intersection. With the four agents so close to each other, the vessels need to perform large avoidance maneuvers. This causes the estimated local goals and corresponding predictions to diverge drastically from the ground truth. However, similarly to the results in Table I, decentralization has little effect on arrival time, total distance, and rule violations.

C. Navigation among Human-piloted Vessels

As long as the human-piloted agent behaves rationally, the resulting trajectories are very similar to the ones presented in Fig. 6, where all agents are autonomous. Therefore, in Fig. 8 we demonstrate how the proposed decentralized MPPI with no communication can cope with irrational agents.

D. Computational Complexity

As previously found in the literature [27], we confirm that MPPI scales linearly with an increasing number of agents (with constant sample number K). Table III shows that the algorithm runs at about 10Hz with two agents, therefore in real-time, and down to less than 4Hz with five agents. We want to stress that this was achieved with parallelization of the sampling procedure over the CPU (Intel® Xeon® W-2123 CPU @ 3.60GHz × 8, 64 GB). Parallelizing this algorithm on a GPU would be highly beneficial, allowing for real-time control of several agents [27].

TABLE III: Average computation time t_c and standard deviation $\sigma_{t,c}$ for increasing number of agents

Number of agents	2	3	4	5
t_c (ms)	90.7	137.2	182.9	209.9
σ_{t_c} (ms)	6.7	8.9	17.5	26.0

V. Conclusions

Within this work, we developed an MPPI controller for decentralized interaction-aware navigation in urban canals. In multiple sets of randomized scenarios, we demonstrate that our method outperforms a state-of-the-art MPC in terms of success rate, deadlocks, collisions, rule violations, and arrival times. In extensive experiments among several rational autonomous agents and case studies with potentially non-cooperative human drivers, we show robust operation while providing insights into the limitations of the approach. We display that decentralizing the MPPI does not sacrifice performance. Moreover, we demonstrate that the method would be able to run in real-time with multiple agents. In the future, however, a GPU implementation would vastly reduce the computation time. Moreover, the sampling distribution can be improved, thus requiring fewer samples to obtain a good approximation of the optimal control.

REFERENCES

[1] Gemeente Amsterdam, "Uitvoeringsplan Transport over Water," 2020, in Dutch. [Online]. Available: http://openresearch.amsterdam/en/page/65638/uitvoeringsplan-transport-over-water

[2] Massachusetts Institute of Technology and Amsterdam Institute for Advance Metropolitan Solutions, "Roboat project." [Online]. Available: https://roboat.org/

[3] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, "Binnenvaartpolitiereglement," 2017, in Dutch. [Online]. Available: https://wetten.overheid.nl/BWBR0003628/2017-01-01/#DeelI_Hoofdstuk6_AfdelingII_Artikel6.04

[4] G. Williams, A. Aldrich, and E. Theodorou, "Model Predictive Path Integral Control using Covariance Variable Importance Sampling," *CoRR*, Sep. 2015.

[5] J. de Vries, E. Trevisan, J. van der Toorn, T. Das, B. Brito, and J. Alonso-Mora, "Regulations Aware Motion Planning for Autonomous Surface Vessels in Urban Canals," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 3291–3297.

[6] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 2153–2160, iSSN: 1050-4729.

[7] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, May 2018.

[8] 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, Mar. 1997.

[9] 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*, May 2008, pp. 1928–1935, iSSN: 1050-4729.

[10] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer, 2011, pp. 3–19.

[11] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 4293–4298, iSSN: 2153-0866.

[12] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

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

[14] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1343–1350, iSSN: 2153-0866.

[15] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement Learning with Iterative Reasoning for Merging in Dense Traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Sep. 2020, pp. 1–6.

[16] M. Garzón and A. Spalanzani, "Game theoretic decision making for autonomous vehicles' merge manoeuvre in high traffic scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 3448–3453.

[17] R. Tian, N. Li, I. Kolmanovsky, Y. Yildiz, and A. R. Girard, "Game-Theoretic Modeling of Traffic in Unsignalized Intersection Network for Autonomous Vehicle Control Verification and Validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2211–2226, Mar. 2022.

[18] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, Apr. 2020.

[19] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 236–243, iSSN: 2153-0866.

[20] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning Interaction-Aware Trajectory Predictions for Decentralized Multi-Robot Motion Planning in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2256–2263, Apr. 2021.

[21] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 797–803, iSSN: 2153-0866.

[22] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for Autonomous Cars that Leverage Effects on Human Actions," in *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.

[23] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 11, pp. P11 011–P11 011, Nov. 2005, publisher: IOP Publishing.

[24] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[25] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system," *International Journal of Control - INT J CONTR*, vol. 80, pp. 1439–1453, Sep. 2007.

[26] E. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proceedings of the 2010 American Control Conference, ACC 2010*. IEEE Computer Society, 2010, pp. 1125–1132.

[27] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, Feb. 2017, publisher: American Institute of Aeronautics and Astronautics.

[28] G. R. Williams, "Model predictive path integral control: Theoretical foundations and applications to autonomous driving," Doctoral Thesis, Georgia Institute of Technology, Atlanta, Mar. 2019, accepted: 2020-05-20T16:57:06Z Publisher: Georgia Institute of Technology. [Online]. Available: https://smartech.gatech.edu/handle/1853/62666

[29] W. Wang, L. Mateos, S. Park, P. Leoni, B. Gheneti, F. Duarte, C. Ratti, and D. Rus, "Design, Modeling, and Nonlinear Model Predictive Tracking Control of a Novel Autonomous Surface Vehicle," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2018.

[30] W. Wang, B. Gheneti, L. A. Mateos, F. Duarte, C. Ratti, and D. Rus, "Roboat: An Autonomous Surface Vehicle for Urban Waterways," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6340–6347, iSSN: 2153-0866.

[31] W. Wang, T. Shan, P. Leoni, D. Fernández-Gutiérrez, D. Meyers, C. Ratti, and D. Rus, "Roboat II: A Novel Autonomous Surface Vessel for Urban Environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 1740–1747, iSSN: 2153-0866.

[32] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The Office Marathon: Robust navigation in an indoor office environment," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 300–307, iSSN: 1050-4729.

[33] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, Dec. 2016.