robotics

MDPI

Article

RingBreak: A Lightweight Decentralized Controller for Symmetry Breaking in Multirobot Intersection Navigation

Vineetha Malathi [1], Pramod Sreedharan [1,*], Ganesha Udupa [1] and Liam Pastorelli [2]

[1] Department of Mechanical Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, Clappana P.O., Kollam 690525, Kerala, India; vineetha013@gmail.com (V.M.); ganesh@am.amrita.edu (G.U.)
[2] Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy; liam.pastorelli@mail.polimi.it
* Correspondence: pramods@am.amrita.edu

Abstract

Reciprocal waiting can stop decentralized robots at symmetric intersections even when local commands maintain separation. RingBreak uses a local conflict core estimate, queueing, clockwise orbiting, release point recovery, and an empirical velocity projection to restore goal-directed motion. Its transition conditions and contact metric are specified for reproducibility. On six fixed layouts repeated with three controller seeds, RingBreak completed 18 of 18 runs without contact; an MPC-CBF deadlock auction implementation completed 6 of 18 without contact. In 20 paired head-on and four-way conflict layouts, RingBreak and a tuned goal-directed artificial potential field (APF) controller both completed every run without contact, but RingBreak had a shorter mean run time (19.24 versus 35.19 s). A separate evaluation independently perturbed starts and goals in seven patterns. Both methods completed 60 of 70 such trials without contact; APF had a shorter mean run time (34.38 versus 38.97 s), and both usually timed out at the walled intersection. Within RingBreak, removing release reduced contact-free completion to 19 of 70, and removing separation projection reduced it to 3 of 50. These results support the faster clearance of the tested reciprocal conflicts and do not provide a general advantage or a formal safety or deadlock-freedom guarantee.

Keywords: decentralized multirobot navigation; deadlock resolution; artificial potential fields; reactive control; intersection coordination

1. Introduction

Decentralized multirobot navigation becomes difficult when robots interact in constrained spaces without a shared planner or explicit communication. Safety is not the only issue. In a head-on passage or at a four-way crossing, each robot can choose a locally sensible velocity and still leave the group in a slow negotiation. The motion then looks safe in a distance sense, but the task has effectively stopped. This paper treats that stalled interaction as a liveness problem rather than only a collision avoidance problem.

Classical local methods remain the starting point. Artificial potential fields combine goal attraction with obstacle repulsion [1]. Velocity obstacle and reciprocal velocity obstacle methods choose admissible velocities from relative geometry [2,3]. Dynamic window search is also useful when velocity and acceleration limits have to be respected [4]. MPC, CBFs, and deadlock-aware games add more structure to the decision process [5–7]. Yet, a local controller can still encounter a balanced configuration in which admissible commands make little progress.

Check for updates

Academic Editor: David Portugal

Received: 28 August 2026
Revised: 23 September 2026
Accepted: 24 September 2026
Published: 27 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

RingBreak is a reactive controller built around that escape motion. Its state machine has four modes: GOAL, QUEUE, ORBIT, and RELEASE. A robot first detects a local conflict core from nearby blockers. It may then wait briefly, enter a clockwise orbit with a rank-dependent radius, and leave through a release target before resuming direct goal tracking. The design is close to roundabout behavior, but the entry order, orbit lane, and exit decision are treated as separate local decisions.

This study makes two contributions: a specified queue–orbit–release controller with local conflict core estimation and a paired evaluation that separates goal completion from contact-free completion. The original fixed-layout comparison uses an MPC-CBF deadlock auction implementation. A controlled head-on and four-way comparison tests RingBreak against tuned goal-directed APF in reciprocal conflict layouts. Separately randomized comparisons and component tests probe behavior outside those layouts. These tests cannot establish formal deadlock freedom or universal superiority over simpler local methods.

Section 2 reviews related methods. Section 3 defines the controller and its transition rules. Section 4 gives both evaluation protocols, followed by the results in Section 5 and their limits in Section 6.

2. Related Work

A recent survey separates multirobot collision and obstacle avoidance into sensor-based, velocity obstacle, optimization-based, barrier function, and coordination-oriented methods [8]. Table 1 uses that grouping but only for the narrower problem considered here. The important split is between avoiding a near collision and forcing progress when the available local choices block each other.

Table 1. Thematic position of related multirobot collision avoidance methods.

Theme	Representative Work	Core Contribution	Main Limitation for This Study
Sensor-based obstacle avoidance	Dynamic window and survey methods [4,8]	Detect and avoid nearby obstacles using local sensing or short forward simulation.	May stop or oscillate in multirobot conflicts
Artificial potential fields	Goal attraction and short-range repulsion [1]	Gives a complete reactive navigation command at low computational cost	Local equilibria can halt goal progress
Velocity obstacle/ORCA	VO, ORCA, and uncertainty-aware Voronoi cells [2,3,9]	Avoid predicted collision using relative velocity or probabilistic safe cells	Can still struggle in dense or symmetric deadlock-like interactions
Model predictive control	MPC, MPC-ORCA, and forecast-driven MPC [5,10,11]	Plans short-horizon motion with constraints and dynamic feasibility	Adds computational load; local minima or stalling can remain
Control barrier functions	CBF-QP, CBF-DMPC, and social mini-game navigation [6,7,12]	Enforces safety constraints mathematically	Safety alone does not guarantee progress
Deadlock-aware methods	Social mini-games and roundabout methods [7,13]	Add priority, yielding, or circulation to restore liveness	May depend on priority rules, communication, or tuning
RingBreak	Present work	Reshapes traffic using queue, orbit, and release behavior	Needs broader randomized testing and hardware validation

2.1. Local Collision Avoidance

Potential fields are a relevant simple controller here because they address the same goal-reaching task: attraction points to the goal, and repulsion responds to nearby robots or

walls [1]. Their local equilibria motivate testing whether explicit entry and exit states add value. The comparison therefore uses a complete APF navigator, not an obstacle repulsion term alone.

Velocity obstacle methods express collision avoidance in velocity space and remain a common reference for multi-agent navigation [2,3]. Their appeal is practical: a robot can compute candidate velocities from relative position and velocity, without asking for a global schedule. Reciprocal variants split responsibility between agents, which is often enough in open space. Probabilistic Voronoi-cell methods extend the same idea toward uncertainty-aware safe regions and have been tested with ground and aerial robots [9]. But, a symmetric passage can still defeat these local rules. Every robot may pick a conservative, safe velocity, and none of those choices creates a passing order.

Dynamic window methods and similar reactive planners choose commands through short forward simulation under velocity and acceleration limits [4]. They are useful, but the score function carries a lot of responsibility. Too much weight on clearance can make stopping look best. Too much weight on goal progress can make a close pass look acceptable. This balance becomes awkward when several robots want the same small region at the same time.

2.2. MPC, CBFs, and Deadlock-Aware Navigation

MPC gives a receding-horizon way to handle constraints, and CBFs add inequality constraints for maintaining safety sets [5,6]. The combination is attractive because separation rules can be placed directly in the controller. Jiang and Guo used CBFs inside distributed MPC for coordinated multirobot control [12]. Freitas et al. combined time-varying artificial vector fields with MPC-ORCA for decentralized collision-free path-following [10]. Forecast-driven MPC has also been proposed for dense or symmetric multirobot avoidance [11]. These methods are stronger than simple reaction, but they still depend on computation time, horizon tuning, and how the local problem is initialized.

Recent social mini-game navigation uses discrete-time CBFs, receding-horizon optimization, and explicit liveness or deadlock logic for doorways and intersections [7]. That setting is close to the present one. The robots must not collide, and they should not negotiate forever. Another line of work creates temporary roundabouts, keeping robots in circulation until an escape condition is met [13]. That idea matters here because circulation is used as a progress mechanism, not only as a repulsive side effect.

The baseline in this benchmark comes from the same general family. It combines a short-horizon MPC objective, CBF-style separation constraints, and a pairwise deadlock auction step. It is a complete navigation controller, so it is a more useful comparison than a plain repulsive field. In the code used for the tests, the baseline is called MPC-CBF-DA. The comparison should be read as an implementation-level benchmark, not as a reproduction of every detail of a particular published controller.

2.3. Circulation-Based Deadlock Breaking

Deadlock resolution by circulation has become a recognizable idea in decentralized multirobot control. Circulation-embedded safety filters and roundabout-like methods use tangential components to avoid undesirable equilibria in multi-agent navigation. Ring-Break belongs to this practical family, but it does not only add a tangential velocity. Its distinguishing design choice is combining four local decisions: whether a robot should queue before entering the conflict zone, which concentric orbit lane it should occupy, when it has accumulated enough phase to leave, and how the release command is filtered with a lightweight safety projection.

3. RingBreak Controller

3.1. Robot and Workspace Model

Each robot is represented as a disc in the horizontal plane with radius $r = 0.16$ m. Robot i has position $p_i \in \mathbb{R}^2$, velocity $v_i \in \mathbb{R}^2$, start point s_i, and goal point g_i. The simulation step is $\Delta t = 0.1$ s. The preferred velocity is

$$v_i^{\text{goal}} = v_{\text{pref}} \frac{g_i - p_i}{\max(\|g_i - p_i\|, \epsilon)}, \tag{1}$$

with $v_{\text{pref}} = 0.82$ m/s, unless the robot is inside the goal tolerance. The commanded velocity is bounded by maximum speed 1.0 m/s and acceleration 1.5 m/s^2. Static walls, when present, are modeled as axis-aligned rectangles.

3.2. Relevant Neighbor Selection

At each step, robot i selects relevant neighbors from nearby robots within a neighborhood radius of 2.6 m. Let $d_{ij} = \|p_j - p_i\|$, $e_i = (g_i - p_i)/\max(\|g_i - p_i\|, \epsilon)$ and $e_j = (g_j - p_j)/\max(\|g_j - p_j\|, \epsilon)$. A candidate robot is treated as a blocker when it is already close, closing in relative motion, or visible in the forward sector with either opposing or crossing motion:

$$\begin{aligned} b_{ij} = {} & [d_{ij} < 1.15] \vee [(p_j - p_i)^\top (v_i - v_j) < 0.02] \\ & \vee \Big([e_i^\top \widehat{(p_j - p_i)} > -0.25] \wedge \big([e_i^\top e_j < 0.3] \vee [|e_i^\top \widehat{(p_j - p_i)}| < 0.95] \big) \Big). \end{aligned} \tag{2}$$

Here, $\widehat{x} = x/\max(\|x\|, \epsilon)$, and the square brackets denote Boolean predicates. The final blocker set $\mathcal{B}_i$ is limited to the six highest-scoring candidates, sorted mainly by distance and forward alignment. Distant robots are therefore ignored unless their geometry suggests an immediate role in the conflict.

3.3. Conflict Core and Ring Ordering

For each relevant neighbor j, the controller intersects the infinite lines through (s_i, g_i) and (s_j, g_j). Parallel lines use the midpoint $(s_i + s_j)/2$; a non-finite result uses $(p_i + p_j)/2$. These pairwise centers c_{ij} give the conflict core

$$c_i = \frac{\sum_{j \in \mathcal{B}_i} w_{ij} c_{ij}}{\sum_{j \in \mathcal{B}_i} w_{ij}}, \qquad w_{ij} = \frac{1}{\max(\|p_i - p_j\|, 0.2)}, \tag{3}$$

where $\mathcal{B}_i$ is the blocker set. After the core is available, each robot computes a clockwise phase to its goal side,

$$\phi_i = \text{mod}_{2\pi}(\angle(p_i - c_i) - \angle(g_i - c_i)). \tag{4}$$

Robots whose pairwise conflict centers lie near c_i form the local cluster. A deterministic rank then selects the ring radius:

$$\rho_i = \rho_0 + k_i \Delta\rho, \tag{5}$$

where $\rho_0 = 0.518$ m, $\Delta\rho = 0.071$ m, and k_i is the local order assigned by sorting $(\phi_i, \|p_i - c_i\|, \text{name}_i)$. The spacing is small, but it keeps robots from being assigned the same orbit.

3.4. Mode Transitions

The controller state m_i belongs to

$$m_i \in \{\textsc{Goal}, \textsc{Queue}, \textsc{Orbit}, \textsc{Release}\}.$$

The GOAL mode applies Equation (1). Let N_i denote a nonempty blocker set and D_i mean $\|p_i - c_i\| < 1.45$ m. The stagnation predicate S_i is true when goal-directed speed stays

below 0.06 m/s for at least 0.937 s or the local blocker geometry is judged symmetric. The symmetry test counts pairwise centers within 0.85 m of c_i: two such centers suffice, as does a blocker-distance spread below 0.35 m; a single blocker within 0.95 m also suffices. The wall-constrained predicate C_i holds when the measured free half-width at either the robot or the core is below 0.337 m. Table 2 gives the mode changes in evaluation order. A robot remains in its current mode if no row applies.

The release predicate R_i is true if the direct goal ray is clear. Otherwise it requires a clockwise exit phase below 122.7 degrees, a goal-heading/tangent dot product above 0.124, and a clear lateral peel ray. The forced release test can also become true after 0.623 s of orbiting when the exit phase is below 148.5 degrees; it still must meet the heading and peel clearance conditions. A robot must spend at least three steps in ORBIT before the transition to RELEASE is considered.

Table 2. State transitions evaluated once per control step; earlier rows within a mode take priority.

From	To	Condition
GOAL	QUEUE	$N_i \wedge C_i \wedge D_i$
GOAL	ORBIT	$N_i \wedge D_i \wedge S_i$
QUEUE	GOAL	$\neg N_i$
QUEUE	ORBIT	$\neg C_i \wedge S_i$
ORBIT	GOAL	$\neg N_i$
ORBIT	QUEUE	$C_i \wedge \lvert\mathcal{B}_i\rvert > 1$
ORBIT	RELEASE	at least three orbit steps and R_i
RELEASE	GOAL	no blockers, direct goal path clear, or the robot has passed the core
RELEASE	QUEUE	$C_i \wedge D_i$, more than 12 release steps, and $\neg R_i$
RELEASE	ORBIT	more than 16 release steps and $\neg R_i$

There is a limited progress property of the state machine: if the blocker set becomes empty, QUEUE, ORBIT, and RELEASE all return to GOAL on the next update. A robot already in RELEASE also returns when its direct goal ray clears or it passes the core. This is a condition for restoring a goal-directed command, not a guarantee of eventual goal arrival; a new blocker or a wall can interrupt the motion again. Table 3 summarizes the per-step execution order of the controller.

Table 3. One control step of RingBreak.

Step	Operation
1	Read the robot state, goals, nearby robots, and wall geometry.
2	Build $\mathcal{B}_i$ and estimate the conflict core c_i with Equation (3).
3	Stop the robot if the goal tolerance has already been met.
4	Enter QUEUE when GOAL mode has stalled near persistent blockers.
5	Assign a ring radius and enter ORBIT when the local order permits it.
6	Compute a release point and enter RELEASE when the exit test is satisfied.
7	Return from RELEASE to GOAL after the direct corridor opens.
8	Form the nominal velocity for the current mode.
9	Apply speed, acceleration, robot separation, and wall separation projection for the next step.

3.5. Orbit and Release Command

In ORBIT, $r_i = p_i - c_i$ is the radial vector, and $n_i = r_i / \max(\|r_i\|, \epsilon)$. The clockwise tangent is $t_i = [n_{iy}, -n_{ix}]^\top$, and the goal-side escape unit is $q_i = (g_i - c_i) / \max(\|g_i - c_i\|, \epsilon)$. The nominal orbit velocity is

$$v_i^{\text{orb}} = \alpha t_i + \beta v_i^{\text{goal}} + \gamma(\rho_i - \|r_i\|)n_i + \delta_i q_i, \tag{6}$$

where the tuning used in the benchmark has $\alpha = 0.299$, $\beta = 0.812$, and $\gamma = 0.834$. The four terms add circulation, goal bias, radial correction, and a peel toward the goal side. The peel gain is

$$\delta_i = (0.22 + 0.35a_i)\max\left(1 - \frac{\phi_i}{11\pi/18}, 0.2\right), \qquad a_i = \min\left(1, \frac{\Phi_i}{\pi/2}\right), \tag{7}$$

where Φ_i is the accumulated clockwise orbit phase after entering ORBIT. The outward bias therefore grows only after the robot has made some angular progress and is closer to the goal-side exit window. If only one blocker remains, the implementation adds a further outward bias. This avoids staying on the ring after the main congestion has already cleared.

Release mode commits the robot to a temporary target beyond the conflict core. Define

$$r_i^{\text{out}} = \max(\rho_i, \rho_0) + 0.58.$$

The target is

$$\tilde{g}_i = \begin{cases} g_i, & \text{if the direct goal ray is clear or } |\mathcal{B}_i| \le 1, \\ c_i + r_i^{\text{out}} q_i + 0.18 t_i, & \text{otherwise.} \end{cases} \tag{8}$$

The release command combines motion to that target with outward peel, rotational carry, and a small velocity-memory term:

$$v_i^{\text{rel}} = \min(v_{\text{pref}}, \|\tilde{g}_i - p_i\|)\widehat{(\tilde{g}_i - p_i)} + 0.20 q_i + 0.16\,\mathbb{I}[\tilde{g}_i \neq g_i] t_i + 0.08 v_i^{\text{last}}. \tag{9}$$

The carry and memory terms reduce abrupt stop-and-turn behavior. The release test checks goal heading, forward clearance, accumulated phase, and a timeout condition. In the benchmark, release clearance is 0.267 m, and the release timeout is 0.623 s.

3.6. Safety Projection

After the nominal command is computed, velocity is projected through speed, acceleration, wall, and pairwise separation limits:

$$v_i^{\text{cmd}} = \Pi_{\text{safe}}(v_i^{\text{raw}}, \mathcal{B}_i, W) = v_i^{\text{raw}} + v_i^{\text{safe}}, \tag{10}$$

where W is the set of walls, and v_i^{safe} is the correction introduced by the projection. For a nearby robot j, let $n_{ij} = (p_i - p_j)/d_{ij}$. Two sequential passes correct the command if $n_{ij}^\top v_i < n_{ij}^\top v_j - \lambda(d_{ij} - d_{\text{safe}})$, with $\lambda = 1.148$ and $d_{\text{safe}} = 0.346$ m. Wall handling reduces velocity toward an axis-aligned wall and adds a fallback repulsive term. Speed and acceleration caps are applied afterward. This local projection is empirical; it is not a formal invariance proof.

For discs of radius 0.16 m, the geometric contact distance is 0.32 m. The selected separation threshold exceeds that distance by only 0.026 m. Lowering it below 0.32 m removes even this nominal clearance margin, regardless of the mode logic. The threshold is necessary as a geometric design choice, although it is still insufficient for a safety proof under simultaneous motion and acceleration limits.

Speed and acceleration clipping bound each commanded step, but they do not alone preserve a robot–robot safe set. Simultaneous updates can make pairwise projections conflict, and an acceleration limit can delay a requested avoidance command. When no blockers or nearby walls are present, the preferred command points toward the goal and decreases goal distance until the goal tolerance is reached. That conditional statement does

not imply progress through arbitrary congestion. Ring ordering and release are intended to escape the local equilibria observed in the tests; finite-time deadlock resolution is not guaranteed for every geometry.

4. Benchmark Protocol

4.1. Scenario Set

The benchmark contains six scenarios selected to span low-conflict, pairwise conflict, and symmetric multirobot conflict. The same scenario files are used for RingBreak and MPC-CBF-DA. Table 4 summarizes the set. The narrow corridor and wall intersection scenarios available in the codebase were not included in the main six-case benchmark reported here.

Table 4. Benchmark scenarios used for main comparison.

Scenario	Robots	Pattern	Purpose
test_01	1	Straight line	Basic goal-reaching check with no robot–robot interaction.
test_02	2	Parallel lanes	Same-direction travel with little expected interference.
test_03	2	Head-on	Direct reciprocal conflict and pass-order formation.
test_04	2	Orthogonal crossing	Two paths meeting near the same center point.
test_05	4	Symmetric crossing	Four-way symmetry, with high deadlock risk.
test_06	4	Asymmetric crossing	Four-way case with small asymmetry in starts and goals.

4.2. Baseline and Metrics

The original baseline is an MPC-CBF-DA controller. It uses a local MPC objective, CBF-style robot and wall separation constraints, and a deadlock auction/projection step for pairwise liveness. The two-controller benchmark used the simulation setup in Table 5: $\Delta t = 0.1$ s, maximum 1200 steps per run, seeds 0, 1, and 2, and a 120 s per-run wall-clock timeout. These seeds altered the MPC auction bids but did not change robot starts or goals. They were repeated controller trials on six fixed layouts, rather than independent geometric trials.

Collision-free completion was the primary success metric: all robots entered a 0.15 m goal tolerance before the step limit with zero robot–robot and robot–wall contacts. Goal completion alone was retained as a secondary diagnostic. One robot contact event was recorded for each robot pair at each simulation step with a physics-engine contact point; continuous contact could have therefore added several events. Wall contacts were counted separately in the walled case. A separate episode count increased when a robot pair first entered contact after at least one contact-free step. These were simulated physical contacts, not a distance-warning counter. The earlier two-crossing baseline could reach both goals while retaining 32 contact pair-steps per run, which was goal completion but not collision-free completion. The deadlock flag was a controller diagnostic from deadlock-avoidance activation, not proof of a mathematical deadlock state.

Table 5. Common run settings and main controller parameters.

Item	Value
Simulator	PyBullet 3.2.7, fixed-step planar disc-robot simulation
Time step	0.1 s
Fixed-layout seeds	0, 1, 2
Perturbed test seeds	100–109
APF development seeds	20–22
Maximum steps	1200 per run
Per-run timeout	120 s
Robot radius	0.16 m
RingBreak preferred speed	0.82 m/s
RingBreak maximum speed	1.0 m/s
RingBreak maximum acceleration	1.5 m/s^2
RingBreak safe separation	0.346 m
RingBreak neighbor radius	2.6 m
RingBreak orbit speed	0.299 m/s
RingBreak release clearance	0.267 m
APF repulsion gain/radius	0.30/0.75 m
MPC horizon/barrier gain	3 steps/0.40 fixed-layout; 0.20 perturbed

4.3. Goal-Directed Potential-Field Comparison

Artificial potential fields combine goal attraction and short-range repulsion [1]. The APF controller evaluated here uses the same robot model, goal tolerance, time step, speed bound, acceleration bound, and contact metric as RingBreak. Its raw command is

$$u_i^{\mathrm{APF}} = v_i^{\mathrm{goal}} + 0.08 v_i^{\mathrm{last}} + k_r \sum_{j \neq i,\ d_{ij} < R} \left(\frac{1}{d_{ij}} - \frac{1}{R} \right) \frac{p_i - p_j}{d_{ij}^3} + u_i^{\mathrm{wall}}, \tag{11}$$

where u_i^{wall} applies the same repulsion law to the nearest point on each wall. The command is then clipped by the common speed and acceleration limits. Nine gain/radius combinations were screened on three development seeds and three scenarios. Selection favored collision-free completion, then goal completion, fewer contacts, and fewer mean steps; all nine settings tied on completion, and $k_r = 0.30$, $R = 0.75$ m had the lowest mean step count. The held-out geometry seeds 100–109 were not used to select these values. APF has no queue, orbit, or release state.

4.4. Controlled Conflict Tests

The conflict-focused comparison started with the exact two-robot head-on and four-robot symmetric crossing layouts. Nine controlled variants of each layout changed the common start-to-core distance (2.5 or 3.5 m instead of 3.0 m), rotated all paths together (22.5, 45, or 67.5 degrees), added 0.1 or 0.2 m to the approach distance of one opposing pair, or combined these changes. Goals remained opposite their starts, so opposing routes still met at the same core; only the arrival-offset cases relaxed simultaneous arrival. These were designed geometric variants, not independent random samples. RingBreak and APF use identical geometry in each pair, with unchanged controller parameters and motion limits. Low-speed time accumulated whenever a robot's commanded speed is below 0.05 m/s. The reported value is the mean, over cases, of the longest accumulated low-speed time among robots in each run; it does not represent one continuous stall.

4.5. Randomized and Component Tests

For each seed from 100 to 109, every start and goal coordinate was perturbed independently by a uniform offset in $[-0.20, 0.20]$ m. The resulting geometry was reused for

every controller on that scenario and seed. The set contained the head-on, two-crossing, four-way symmetric and asymmetric layouts, a four-way intersection with walls, and six- and eight-robot radial crossings. Thus RingBreak and APF were paired on 70 perturbed cases. The MPC implementation was also run on the first five seeds of the five two- and four-robot layouts. Its CBF gain was selected from 0.20, 0.40, and 0.80 on four separate development trials (two seeds and two layouts), using collision-free completion first; 0.20 was selected. The original RingBreak settings were held fixed.

Component tests used the same perturbed geometries. The variants removed queue entry, orbit entry, release, rank-dependent ring radii, or the separation projection one at a time; speed and acceleration caps remained in the last variant. These tests identify an effect within the implemented controller, not a formal necessity of any component. Controller time was measured around the per-robot command calculation, excluding simulation and logging. A sequential timing profile used three additional perturbed head-on seeds (200–202). The one-parameter-at-a-time sensitivity check used five separate seeds (30–34) and values at 80%, 100%, and 120% of the selected separation, orbit speed, release clearance, and stagnation time settings.

4.6. Reproducibility

Both controllers used the same benchmark driver, scenario definitions, seeds, step limit, and timeout. The evaluation produced per-run logs, CSV summaries, trajectory overlays, and performance plots. The original protocol contained 36 controller executions: six scenarios, three seeds, and two methods. Table 6 and the first three figures use those fixed-layout runs. The additional perturbed trials described below used separately generated starts and goals, with identical geometry for each paired controller run.

The experiments were run on an AMD Ryzen 5 7520U laptop with 14 GiB of system memory. The controller timing measurements used a sequential profile on the same machine; they excluded physics stepping, rendering, and file output. No run was stopped for memory exhaustion. Source scripts and per-run configurations retained the random seeds and all parameter values used for the tables.

Table 6. Fixed-layout comparison over seeds 0–2. Goal is goal completion regardless of contact; collision-free completion is discussed in Section 5.2. Collision counts are contact pair-steps per run.

Scenario	RingBreak			MPC-CBF-DA		
	Goal	Time	Coll.	Goal	Time	Coll.
Single robot	1.00	16.8	0.0	1.00	17.2	0.0
Two parallel	1.00	17.9	0.0	1.00	17.2	0.0
Two head-on	1.00	17.3	0.0	0.00	120.0	0.0
Two crossing	1.00	17.8	0.0	1.00	19.1	32.0
Four symmetric	1.00	20.5	0.0	0.00	120.0	0.0
Four asymmetric	1.00	20.4	0.0	0.00	120.0	0.0

5. Results

5.1. Scenario Outcomes

The first two scenarios are low-conflict checks. The last four put robots in direct or near-direct competition for the same central region. Figure 1 overlays representative trajectories for both controllers.

Table 6 gives scenario-level means over seeds 0–2. RingBreak reached every goal without contact. The baseline reached the goals in the single-robot, parallel-lane, and two-crossing scenarios, but its two-crossing runs recorded 32 contact pair-steps each. It did not complete the head-on or either four-way scenario within the 1200-step limit.

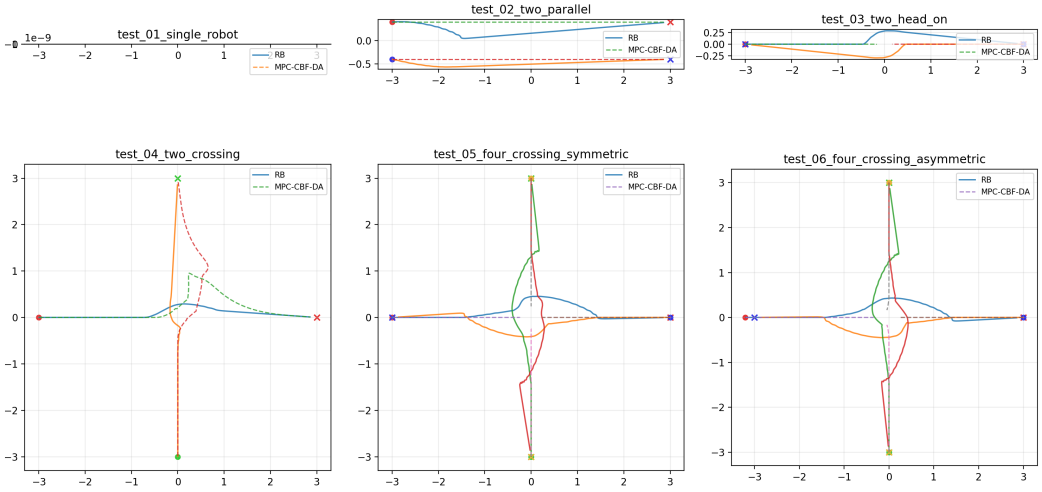


Figure 1. Trajectory overlays for RingBreak and MPC-CBF-DA. Solid trajectories represent RingBreak; dashed trajectories represent MPC-CBF-DA. Head-on and four-way crossing cases produce clearest visual separation.

5.2. Aggregate Comparison

Table 7 separates contact-free completion from goal-reaching. RingBreak completed 18 of 18 fixed-layout runs without contact; MPC-CBF-DA completed 6 of 18 without contact, although it reached the goals in 9 of 18. Its mean run time includes timeouts. Mean steps decrease from 689.17 to 184.50 with RingBreak, and mean contact pair-steps decrease from 5.33 to zero.

Table 7. Fixed-layout aggregate over 18 runs per controller. Contact-free completion is the primary outcome; mean time includes runs reaching the step limit.

Method	Contact-Free	Time (s)	Steps	Contacts	Deadlock Flags
RingBreak	18/18	18.45	184.50	0.00	30.83
MPC-CBF-DA	6/18	68.92	689.17	5.33	1277.33

Figure 2 plots the same success rate data by scenario, and Figure 3 shows completion time. In the single-robot and parallel-lane checks, the baseline is close to RingBreak, as those cases do not require explicit deadlock resolution. The difference appears when the geometry becomes balanced, especially in the head-on and four-way crossings.

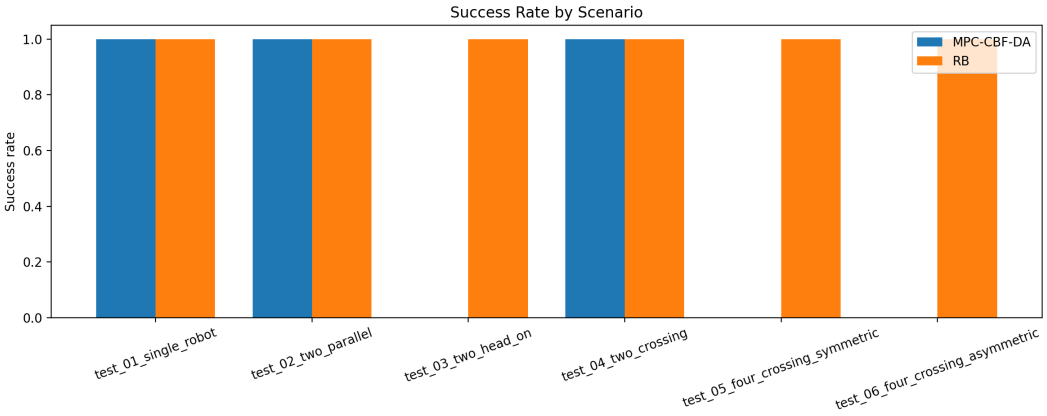


Figure 2. Goal completion by fixed scenario. Contact is recorded separately in Table 6.

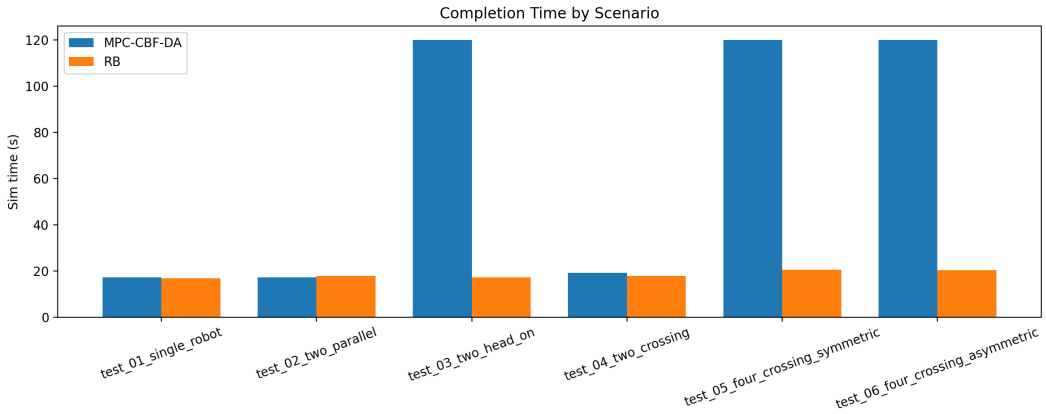


Figure 3. Mean completion time by scenario. Failed baseline runs reach the 120 s timeout; RingBreak clears the tested intersections in about 17–21 s.

5.3. Interpretation of Controller Behavior

The head-on and four-way crossing cases show why the ring-breaking maneuver was added. A direct goal command sends the robots toward the same local region. The safety filter then slows them as separation shrinks. If every robot follows that pattern, the group can keep losing speed without forming a priority order. RingBreak changes the shape of the interaction: one or more robots circulate clockwise, hold separate radii, and leave after the goal corridor is judged clear. Reciprocal waiting is turned into a short local detour.

The cost is extra intervention. In easy scenarios, orbit and queue flags can appear even when direct motion might have worked. In the two-parallel scenario, RingBreak records deadlock flags while still reaching the goals quickly and without collisions. This points to a conservative conflict detector. The present tuning favors clearing the intersection over reducing mode activations.

5.4. Controlled Reciprocal Conflict Comparison

Table 8 reports the exact layout and nine conflict-preserving variants of each pattern. Both methods reached every goal without contact. APF did not enter a permanent deadlock, but it accumulated more low-speed time and required more steps. On the exact four-way layout, RingBreak needed 205 steps, and APF needed 420; on the exact head-on layout, they needed 173 and 308 steps, respectively. Rotating an otherwise empty workspace does not constitute an independent test of environmental robustness.

Table 8. Controlled reciprocal conflict cases, ten geometries per pattern. All 40 controller runs reached every goal without robot contact. Steps, completion time, and longest per-robot low-speed time are means over ten geometries.

Pattern	RingBreak			Tuned APF		
	Steps	Time (s)	Low-Speed (s)	Steps	Time (s)	Low-Speed (s)
Head-on (2)	178.9	17.9	0.0	311.8	31.2	12.3
Four-way (4)	205.9	20.6	1.4	392.0	39.2	20.2

5.5. Paired Randomized Comparison

Table 9 reports ten independently perturbed geometries for each of seven patterns. The mean step count includes unsuccessful runs at the 1200-step limit. The two methods both had 60 collision-free completions across the 70 paired cases. APF completed all of the open layouts without contact, including the six- and eight-robot crossings, and generally required fewer steps on the larger crossings. RingBreak missed one asymmetric four-way case that APF completed. In the walled intersection, RingBreak completed one of ten trials

and APF none. The exact unperturbed walled layout was completed by RingBreak, so the perturbed failures expose sensitivity to the entry geometry rather than an inability to traverse that layout at all.

Table 9. Held-out randomized trials. Goal and contact-free columns are counts out of ten. Steps are means including timeouts at 1200 steps.

Pattern	RingBreak			Tuned APF		
	Goal	Safe	Steps	Goal	Safe	Steps
Head-on, 2 robots	10	10	175.7	10	10	180.4
Crossing, 2 robots	10	10	183.7	10	10	182.7
Symmetric, 4 robots	10	10	239.9	10	10	191.1
Asymmetric, 4 robots	9	9	394.2	10	10	190.6
Walled, 4 robots	1	1	1111.9	0	0	1200.0
Radial, 6 robots	10	10	282.3	10	10	221.3
Radial, 8 robots	10	10	340.1	10	10	240.4

Table 10 brings the three comparisons together. RingBreak cleared the controlled head-on and four-way conflicts faster than APF; both remained contact-free. This advantage did not extend to the broader randomized set, where APF had a shorter mean run time at the same contact-free completion count. The fixed-layout MPC result is a comparison with that implementation, not with all MPC-CBF controllers.

Table 10. Collision-free goal completion and mean simulated run time. Means include unsuccessful runs at the 120 s step limit; the fixed-layout comparator is MPC-CBF auction, while the other two use tuned APF.

Comparison	RingBreak	Comparator
Fixed (18 runs)	18/18; 18.45 s	MPC-CBF-auction: 6/18; 68.92 s
Controlled conflicts (20)	20/20; 19.24 s	APF: 20/20; 35.19 s
Randomized (70 runs)	60/70; 38.97 s	APF: 60/70; 34.38 s

Figure 4 shows the paired completion step difference in open layouts where both controllers reached their goals. Positive values mean APF used fewer steps. The one asymmetric case completed only by APF was excluded from this conditional time plot, as is the walled pattern where neither method completed the same perturbed case.

Only two paired cases disagree in collision-free completion: one favors RingBreak , and one favors APF. The two-sided exact McNemar test therefore gives $p = 1.00$. This does not establish equal performance; ten geometries per pattern give wide uncertainty. For example, a 10/10 completion count has a 95% Wilson interval of approximately 0.72–1.00. The step results indicate an efficiency cost for the extra orbit and release motion on the open multirobot patterns.

5.6. Component Tests and Sensitivity

Table 11 uses the same held-out starts and goals. Suppressing release made 51 of 70 runs fail to complete, usually with continued orbiting. Removing the local separation projection let 46 of 50 runs reach their goals, but only three finished without contact; these runs had 2777 robot contact pair-steps and 1479 wall contact steps. Removing queue entry lost one walled completion in the five tested two- and four-robot patterns. The no-orbit variant completed 59 of 70 without contact, one fewer than RingBreak, while its mean step count rose from 389.7 to 443.3. Ring rank made only a small difference in this sample.

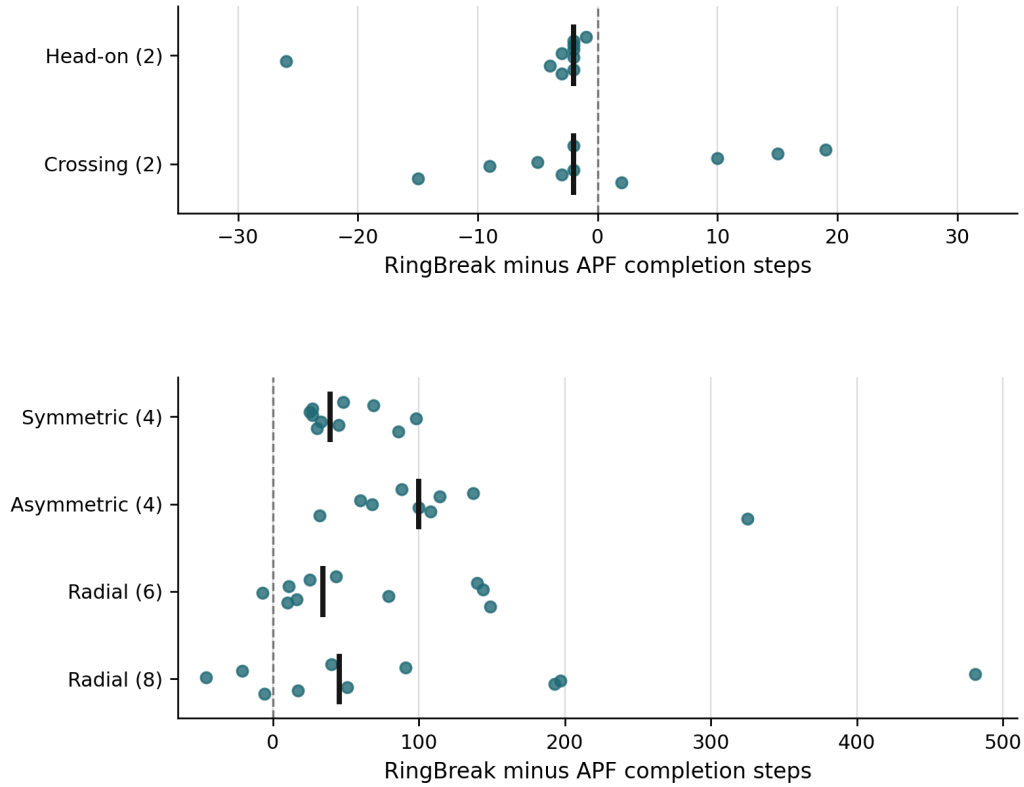


Figure 4. Paired RingBreak-minus-APF completion steps for jointly completed open-layout trials. Each point is one geometry; black ticks mark medians, and dashed lines mark zero. The two panels use separate horizontal scales.

Table 11. RingBreak component tests. The 50-run variants were evaluated on the five two- and four-robot patterns; orbit and release variants also include the radial cases.

Variant	Runs	Goal	Contact-Free	Mean Steps
Full RingBreak	70	60	60	389.7
No queue entry	50	39	39	438.7
No orbit entry	70	60	59	443.3
No release	70	19	19	966.4
Fixed ring radius	50	39	39	452.9
No separation projection	50	46	3	301.4

The parameter sweep in Table 12 kept all other settings fixed. Reducing the separation threshold by 20% put it below the 0.32 m physical contact distance and generated 10,952 robot contact pair-steps across 15 trials. Raising orbit speed improved completion in this small development set but that post hoc result was not used to retune the held-out comparison. Stagnation time changes within the tested range did not change the number of collision-free completions, probably because symmetry pressure often activates orbit before the time threshold.

Table 12. One-parameter sensitivity over 15 trials per setting. Entries count collision-free completions at 80%, 100%, and 120% of the selected value.

Parameter	80%	100%	120%
Separation threshold	7	12	12
Orbit speed	11	12	13
Release clearance	11	12	11
Stagnation time	12	12	12

5.7. MPC Comparison and Controller Time

Table 13 uses the first five perturbed geometries of each two- and four-robot pattern. The MPC barrier gain 0.20 achieved three contact-free completions in four development trials, compared with two each for 0.40 and 0.80. On the held-out head-on perturbations, MPC completed all five without contact, unlike its exact fixed head-on layout. It reached both goals in two of five orthogonal crossings, but none was contact-free; the crossing runs averaged 658.4 robot contact pair-steps. The symmetric and asymmetric four-way runs ended at the step limit. All five walled MPC runs hit the 120 s wall-clock limit before summary metrics were produced, so their step and contact counts are unknown.

Table 13. Five paired perturbed trials per pattern. Goal and safe columns are counts out of five; wall timeouts ended before metrics were written.

Pattern	RingBreak Safe	MPC Goal	MPC Safe	MPC Wall Timeouts
Head-on, 2 robots	5	5	5	0
Crossing, 2 robots	5	2	0	0
Symmetric, 4 robots	5	0	0	0
Asymmetric, 4 robots	5	0	0	0
Walled, 4 robots	0	0	0	5

The separate sequential profile in Table 14 excludes physics and file output. APF required the shortest controller time. RingBreak was much cheaper than the MPC implementation, although this small two-robot timing sample is not a general scaling measurement.

Table 14. Mean per-robot controller time over three perturbed head-on trials on the stated laptop.

Controller	Time per Robot Step (ms)
Goal-directed APF	0.032
RingBreak	0.165
MPC-CBF-DA	10.76

6. Discussion

6.1. What the Mode Logic Changes

On the fixed symmetric layouts, the MPC-CBF-DA implementation stopped short of the goals without robot contact. A separation constraint can permit this zero-progress outcome. The auction step adjusts pairwise progress, but its result is sensitive to geometry and tuning. The perturbed trials should therefore be read alongside the fixed-layout results, rather than as repetitions of the same failure.

RingBreak adds a transverse orbit command, and the release mode gives that motion an exit. In the component tests, removing release caused the largest loss of goal completion. Removing orbit had a smaller effect: the goal-plus-projection variant still completed many cases, although it was slower and missed some six- and eight-robot trials. Fixed ring radius and queue removal changed little in this sample. RingBreak cleared the controlled reciprocal conflicts faster than APF, although APF also completed every case. In the independently perturbed open patterns, APF was often faster. These experiments establish a role for release and projection within RingBreak, but not an overall advantage over APF.

6.2. Limits of the Present Evidence

The randomized set covers seven layout patterns with ten small coordinate perturbations each. This is more informative than the original three fixed-geometry seeds but still gives wide per-pattern intervals and does not sample arbitrary maps. The walled intersection is a clear counterexample to broad robustness: most perturbed runs stalled

without contact, despite success on its exact nominal geometry. The model uses planar discs and direct base-velocity commands. Heterogeneous dynamics, sensing uncertainty, moving obstacles, and physical robots remain outside the present evaluation.

The safety projection is empirical and does not prove forward invariance. Its removal caused many contacts, while the selected threshold still required careful tuning near the disc diameter. The MPC-CBF-DA controller is one implementation, not a ranking against all MPC-CBF designs. A certified safety filter or a formally verified release condition would require a separate design and proof.

6.3. Practical Implications

RingBreak uses local geometry, a mode state, and a limited blocker set. It needs no centralized coordinator or online optimization. Tuned APF was also lightweight and often faster in open crossings. Future work will test a progress-triggered APF–RingBreak combination: APF for ordinary motion, with RingBreak activated when reciprocal waiting persists. The switching rule and its safety and completion performance remain untested.

The controller is local to intersections and passages. Cluttered maps with long-horizon routing constraints need a higher-level planner. The next engineering task is to identify the geometries that trigger repeated orbit entry in walled intersections and test an explicit fallback.

7. Conclusions

RingBreak combines conflict core estimation, queueing, orbiting, release, and an empirical separation projection. It completed all six fixed benchmark layouts without contact. Across 20 controlled head-on and four-way conflict geometries, both RingBreak and APF completed every run without contact; RingBreak averaged 19.24 s versus APF's 35.19 s. In 70 independently perturbed cases, both achieved 60 contact-free completions, but APF averaged 34.38 s versus RingBreak's 38.97 s. Release and projection mattered within the present RingBreak implementation; queueing and ring rank had little measured effect in the tested layouts.

The walled intersection remains the clearest weakness: small changes to starts and goals often produced contact-free timeouts. These results do not establish a general performance advantage or formal safety guarantee. Future work should test a progress-monitoring fallback for walled crossings, broader map variations, and physical robots with measured sensing and drive limitations.

Author Contributions: Conceptualization, V.M. and P.S.; methodology, V.M. and P.S.; software and validation, V.M.; formal analysis, V.M. and P.S.; writing—original draft preparation, V.M.; writing—review and editing, V.M., P.S., G.U. and L.P.; visualization, V.M.; supervision, P.S., G.U. and L.P.; project administration, P.S. and G.U. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge Sri Mata Amritanandamayi Devi (Amma), Amrita Vishwa Vidyapeetham, for her inspiration and for providing financial support for the Article Processing Charges (APC) of this publication. During the preparation of this manuscript, the authors used OpenAI (GPT-5) for language editing, formatting assistance, and document review. The authors reviewed and edited the output and take full responsibility for the content of this publication.

Institutional Review Board Statement: Not applicable. This study did not involve human participants, human data, animals, or biological material.

Informed Consent Statement: Not applicable.

Data Availability Statement: The benchmark code, scenario files, controller configurations, generated tables, and generated figures are not publicly archived at this stage because the project is

under active manuscript preparation. Availability can be reconsidered according to journal and institutional requirements.

Acknowledgments: AI-assisted tools were used for language editing, formatting support, and LaTeX drafting assistance. The authors reviewed and edited the output and take full responsibility for the content of this manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
2. Fiorini, P.; Shiller, Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
3. van den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n-body Collision Avoidance. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–19. [CrossRef]
4. Fox, D.; Burgard, W.; Thrun, S. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
5. Mayne, D.Q.; Rawlings, J.B.; Rao, C.V.; Scokaert, P.O.M. Constrained Model Predictive Control: Stability and Optimality. *Automatica* **2000**, *36*, 789–814. [CrossRef]
6. Ames, A.D.; Xu, X.; Grizzle, J.W.; Tabuada, P. Control Barrier Function Based Quadratic Programs for Safety Critical Systems. *IEEE Trans. Autom. Control* **2017**, *62*, 3861–3876. [CrossRef]
7. Chandra, R.; Zinage, V.; Bakolas, E.; Stone, P.; Biswas, J. Deadlock-free, Safe, and Decentralized Multi-robot Navigation in Social Mini-games via Discrete-time Control Barrier Functions. *Auton. Robot.* **2025**, *49*, 12. [CrossRef]
8. Yang, G.; An, L.; Zhao, C. Collision/Obstacle Avoidance Coordination of Multi-Robot Systems: A Survey. *Actuators* **2025**, *14*, 85. [CrossRef]
9. Zhu, H.; Brito, B.; Alonso-Mora, J. Decentralized Probabilistic Multi-Robot Collision Avoidance Using Buffered Uncertainty-Aware Voronoi Cells. *Auton. Robot.* **2022**, *46*, 401–420. [CrossRef]
10. Freitas, E.J.R.; Vangasse, A.d.C.; Raffo, G.V.; Pimenta, L.C.A. Decentralized Multi-Robot Collision-Free Path Following Based on Time-Varying Artificial Vector Fields and MPC-ORCA. In *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*; IEEE: New York, NY, USA, 2023. [CrossRef]
11. Hailu, H.; Gebregziabher, B.; Raj, P. Forecast-Driven MPC for Decentralized Multi-Robot Collision Avoidance. In *2025 8th International Conference on Intelligent Robotics and Control Engineering (IRCE)*; IEEE: New York, NY, USA, 2025. [CrossRef]
12. Jiang, C.; Guo, Y. Incorporating Control Barrier Functions in Distributed Model Predictive Control for Multirobot Coordinated Control. *IEEE Trans. Control Netw. Syst.* **2024**, *11*, 547–558. [CrossRef]
13. Lee, W.; Sim, J.; Kim, J.; Jo, S.; Luo, W.; Nam, C. Merry-Go-Round: Safe Control of Decentralized Multi-Robot Systems with Deadlock Prevention. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: New York, NY, USA, 2025; pp. 4589–4595. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.