

Article

Environment-Specific Route-Library Adaptation for Decentralized Multi-Robot Navigation via Hybrid RRT and Behavior Cloning in Grid-Based Industrial Environments

Yovel Atia ¹ and Chen Giladi ^{2,*} ¹ Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel² Department of Mechanical Engineering, Sami Shamoon College of Engineering (SCE), 84 Jabotinsky St., Ashdod 77245, Israel

* Correspondence: chengi1@sce.ac.il

Abstract

Decentralized multi-robot navigation in grid-based industrial environments must reach goals and avoid collisions without centralized control or direct communication. We study a hybrid framework pairing an offline Rapidly-Exploring Random Tree (RRT) expert with a Behavior Cloning (BC) local policy and route reuse, evaluated in a reproducible simulator; collisions are predicted blocked-move events, not physical contacts. Our central result is a quantitative analysis of environment-specific route-library adaptation: a route library generated for one map and deployed on another leaves robots blocked by unfamiliar obstacles, whereas regenerating it on the deployment map cuts collisions by 38–85% and task failures by 43–70% across two- to ten-robot fleets, and replicates on a third corridor layout (30 seeds; Mann–Whitney $p < 10^{-5}$; permutation $p_{\text{perm}} < 0.001$). Simpler remedies, filtering or repairing invalid routes, recover most of this gain, and a non-learning scripted connector matches the trained policy: the effect lives in the route library itself. A centralized prioritized-planning baseline bounds all communication-free variants from above; capping the online RRT baseline’s planning budget preserves its per-task collision quality but sharply raises task failures. A collision-history-sharing add-on coordinating through a shared map rather than messaging gives a limited, layout-dependent benefit not surviving multiple-comparison correction.

Keywords: decentralized multi-robot navigation; behavior cloning; RRT; route-library adaptation; imitation learning; collision-history sharing; reproducibility; grid-based environments; warehouse robotics



Academic Editor: Xinjun Liu

Received: 4 June 2026

Revised: 5 July 2026

Accepted: 20 July 2026

Published: 28 July 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 \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Mobile robots must reach their goals while avoiding collisions with obstacles [1,2]. In structured industrial settings—automated warehouses, in particular—fleets of such robots must additionally avoid collisions with one another [3]. Operating such fleets in a *decentralized* way—each robot deciding from local information, without a central coordinator or direct inter-robot communication [4]—improves robustness and scalability, but makes coordination harder; in dense decentralized fleets, the dominant failure mode is deadlock and collision [5].

Two families of methods dominate grid navigation. Sampling-based planners such as RRT explore the configuration space and return feasible paths [6], but in dynamic, multi-robot settings, they incur frequent, costly replanning. Learning-based policies such as

Behavior Cloning (BC) give fast, low-latency reactive control by imitating expert demonstrations, but degrade over long horizons and when the deployment environment diverges from the training data [7]. A *hybrid* framework that uses a pre-computed RRT route library together with a BC local policy [8]—reusing well-tested paths while reacting locally—can capture the strengths of both.

We study exactly such a framework and ask a question central to its practical use: *when does the learned navigation transfer to a new environment, and what governs its quality?* Our answer, obtained from a fully reproducible re-implementation that reuses the original trained network, RRT planner and route data verbatim, is that the decisive factor is the match between the route library and the deployment map. A library generated on one map and deployed unchanged on another leaves robots repeatedly blocked at the new obstacles; regenerating the library on the deployment map restores performance and sharply reduces both collisions and task failures. We deliberately position this contribution as a *quantitative analysis of environment-specific route-library adaptation*, not as a new navigation architecture: the BC network and the RRT planner are fixed throughout, and only the route library, the data-level carrier of the learned navigation skill, changes. To bound the effect, we compare against the simple library-level remedies a practitioner would try first (filtering, repairing, or regenerating invalid routes), against a non-learning scripted connector in place of the BC network, against a mainstream multi-robot path-finding baseline (prioritized planning with space–time A* over a shared reservation table [9,10], a standard decoupled multi-robot path-finding baseline commonly compared with conflict-based search and its bounded-suboptimal variants [11,12]), and against an online RRT baseline evaluated across planning-budget caps, clarifying the computational trade-off that motivates route reuse. We further evaluate—across repeated seeded trials—a lightweight collision-history-sharing add-on that needs no direct inter-robot messaging. The add-on was proposed as future work in an earlier unpublished M.Sc. thesis by the first author (hereafter the originating thesis) and is implemented and systematically evaluated here.

Our contributions are:

- A faithful, deterministic, seeded re-implementation of a hybrid RRT + Behavior Cloning decentralized navigation framework that reuses the original trained BC network, RRT planner, and route datasets, with the non-scalable multi-processing prototype replaced by a reproducible single-process scheduler (code and raw results available in the public repository).
- A quantification of *environment-specific route-library adaptation*: an environment-adapted route library cuts collisions by 38–85% (Mann–Whitney U on per-episode collision counts, all $p < 10^{-5}$, corroborated by permutation tests) and task failures by 43–70% relative to a Map2-naive library across five fleet sizes (2, 4, 6, 8, 10 robots) over 30 seeds, recovering much of the collision quality of the default-budget online RRT baseline at a fraction of its runtime; the effect replicates on a third, structurally different corridor layout (Map3).
- A baseline analysis that bounds the adaptation effect (Section 3.4): offline filtering, repair, and regeneration of invalid transferred routes; a non-learning scripted connector replacing the trained BC network; prioritized planning with space–time A* over a shared reservation table [9–11]; and an online RRT baseline across planning-budget caps, yielding a compute-versus-quality frontier for the reuse-versus-replan decision.
- A repeated-seed evaluation of collision-history sharing—indirect coordination through a shared map, with no direct messaging—within the full framework, finding only a limited benefit (on Map1, nominally significant reductions of 9% under selective and 8% under full participation, $p = 0.024$ and $p = 0.047$, neither surviving Holm correction; a second layout: not significant). This corrects the originating thesis’s

preliminary finding—a near-halving of collisions inferred from a single unseeded run—and shows that the effect is undetectable at low task throughput, where too few tasks accumulate to populate the shared map; we document this regime explicitly.

- A transparent, reproducible methodology in which every figure and number regenerates from the provided code and raw results, including the route-filtering and throughput conditions that determine whether each effect appears.

2. Related Work

Decentralized multi-robot navigation. Decentralized schemes avoid the single point of failure of centralized coordinators; approaches include cooperative reinforcement learning with a shared policy and inter-robot information exchange [3] and multi-agent RL more broadly [13], control-barrier functions [14,15], optimization-based trajectory planning [16], and swarm-intelligence approaches (reviewed for multi-UAV systems in [17]). Coping with deadlock and collision in dense fleets remains a persistent difficulty, including for decentralized aerial swarms [5]. **Planning and learning.** A* and sampling-based planners (RRT/RRT*) underpin robot path planning [6,18–20]; Behavior Cloning learns reactive policies from expert demonstrations [8,21–25], with well-known covariate-shift limits when the deployment distribution differs from the demonstrations [7]. Hybrid planner–policy methods combine a model-based planner with a learned model that guides it [26]; the framework we reuse couples an RRT expert, a BC policy, and route reuse, and—unlike hybrids that use BC only to smooth or refine planner paths—uses BC for decentralized reactive control and online RRT only for connecting segments. **Skill transfer and adaptation.** Transferring a learned policy to a new deployment environment—for example, from simulation to a real robot—is a core challenge for imitation-learning methods [27]; performance hinges on alignment between training and deployment distributions [7]. We give a concrete grid-navigation instance: adapting the expert route library to the deployment map restores performance lost to environment mismatch. Throughout, we use *route-library adaptation* for the specific, data-level mechanism studied here, and reserve *skill transfer* for the broader phenomenon of which it is one instance: the policy and planner are fixed, and what transfers, or fails to transfer, is the route library that carries the environment-specific part of the learned behavior. **Experience sharing.** Multi-agent policies can be improved by modeling other agents’ behaviors and interactions [28,29], by training against diverse partner populations [30], and by leveraging human experience [31]; we study a minimal instance that uses no direct messaging—only a shared collision-frequency map—and evaluate it across repeated seeded trials.

3. Materials and Methods

All components are implemented in a deterministic, seeded simulator available in the public repository (experiments/); every result regenerates from the provided code, with reproduction instructions provided in the public repository. The simulator is written in Python 3.12.3 (Python Software Foundation, Beaverton, OR, USA) and uses NumPy 2.3, SciPy 1.16 and pandas 2.3 for the simulation and analysis and PyTorch 2.9 for the trained network, with Matplotlib and seaborn for the figures. The simulator reuses the original implementation verbatim—the trained BC network (`trained_model_NN.pth`), the RRT planner, the offline route datasets and the per-step navigation logic—and replaces only the original multi-processing/shared-memory harness, which did not scale, with a single-process round-robin scheduler so that every episode is reproducible from its seed.

3.1. Environment and Metrics

N robots navigate a 50×50 occupancy grid in one of three layouts (Map1, Map2, Map3; Figure 1); obstacles are inflated by one cell to account for the robot footprint. Map1 and Map2 are the block layouts of the originating thesis; Map3 is a structurally different corridor layout added for this revision, with three shelf bands, staggered bottleneck doors, and a vertical divider at a matched obstacle density (Section 3.3). The base navigation is fully decentralized: each robot acts on local observations only, with no shared state and no inter-robot communication. The optional collision-history sharing of Section 3.5 is the single exception—it introduces one shared data structure (a global collision-frequency map) through which a subset of robots coordinate *indirectly*, in a stigmergy-style manner, without exchanging messages. Because this map is globally shared, the add-on is message-free but not information-free: it departs from fully decentralized operation, and we evaluate it separately as an optional add-on rather than as part of the base framework. Over a fixed step budget, a *task* is a robot reaching its goal, after which a new random start/goal pair is drawn; *tasks completed* measures throughput. A *collision* is a predicted blocked-move event—the robot’s three-step look-ahead lands in a cell occupied by a static obstacle or by another robot—counted per event, matching the originating thesis’s definition (interactions with obstacles or other agents), not a physical contact. A task that cannot make progress within a step cap is counted as a *failure*; the *failure rate* is the number of failed tasks divided by the total number of task attempts (completed plus failed). Because a fixed step budget yields different numbers of completed tasks across conditions, we report collisions both absolutely and *per completed task*, the workload-normalized metric.

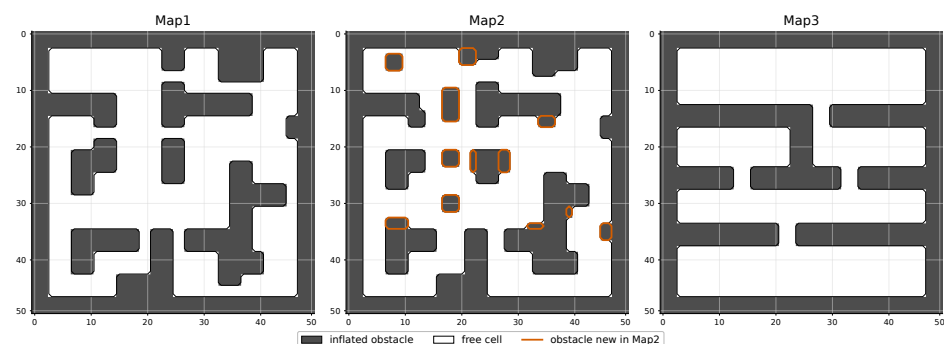


Figure 1. The three 50×50 grid workspaces (Map1, Map2, Map3): dark gray = inflated obstacle, white = free cell, with grid ticks every ten cells. Obstacles are inflated by the robot footprint. The vermillion outlines on Map2 mark the obstacles that are *new* relative to Map1—the cells the Map1-generated route library does not account for. Map3, added in this revision, is a corridor layout with three shelf bands, staggered bottleneck doors, and a vertical divider at an obstacle density matched to Map1.

3.2. Hybrid RRT + Behavior Cloning Framework

An offline RRT expert generates a library of collision-free routes on a map. At runtime, each robot (i) selects, from the library, routes whose bounding box brings it toward its goal and that are locally obstacle-free (the `Get_best_routes` filter); (ii) drives toward the selected route under the trained BC network, which maps the robot’s position, a *target waypoint on that route*, and a 5×5 local occupancy window to an incremental move—so the chosen route enters the policy through its target-waypoint input; and (iii) reconnects with a bounded online RRT segment toward the goal in two cases (Algorithm 1): as a fallback when no suitable library route is available (the nearest is too far to reach directly), and—when a three-step look-ahead along the chosen route predicts a blocked cell—probabilistically, logging the event and reconnecting with a probability that decreases as the task nears completion (more readily early in a task than near the goal). The BC network is

a 29-input multi-layer perceptron with hidden layers 256–256–64 (2 position + 2 target-waypoint + 25 occupancy inputs; ReLU activations; linear two-output head). We use the original trained weights and RRT/route code without modification.

BC training (from the originating thesis). Because we reuse the trained network verbatim, its training belongs to the originating thesis; we summarize it here for self-containment. Demonstrations were generated by an RRT* expert (MATLAB R2023b, The MathWorks, Inc., Natick, MA, USA; `plannerRRTStar`) on the Map1 grid with additional randomly placed obstacles at densities of 0–2%, sampling random start/goal pairs; each state along an expert route (current position, target waypoint, and the 5×5 local occupancy window) was labeled with the incremental expert step ($\Delta x, \Delta y$) to the next waypoint, i.e., a two-output regression formulation, with several thousand state–action samples per generation batch. Training minimized the mean-squared error with Adam (learning rate 10^{-3} , batch size 64) over an 80/20 train/validation split; the architecture comparison in the thesis (200 epochs) selected among a fully connected network, a convolutional variant, and gradient boosting, with the fully connected 256–256–64 network, the variant deployed in the released framework, performing close to the alternatives (the originating thesis reports an overall validation R^2 of 88% for the best model in that comparison). Beyond these settings, the thesis does not document the final training run that produced the shipped weights (in particular, the exact epoch count, random seed, and demonstration-set snapshot are unrecorded), which is one more reason we treat the network as a fixed artifact rather than retraining it. The network is trained once and frozen; whether a learned connector is necessary at all is tested directly by the scripted-connector ablation of Section 3.4. Figure 2 summarizes the offline and deployment phases, the route-library adaptation test condition, and the optional collision-history-sharing add-on; Algorithm 1 states the per-robot runtime decision step, and Table 1 lists the implementation settings, including the online-RRT iteration cap. In Algorithm 1, robots interact only through the shared map H : every robot writes its own collisions to H , while only the sharing subset (a fraction q_{share}) reads the aggregate to bias route choice—there is no direct robot-to-robot messaging. The routine `ONLINERRT( $x, g$ )` plans a fresh collision-free segment from x toward goal g , bounded by that cap.

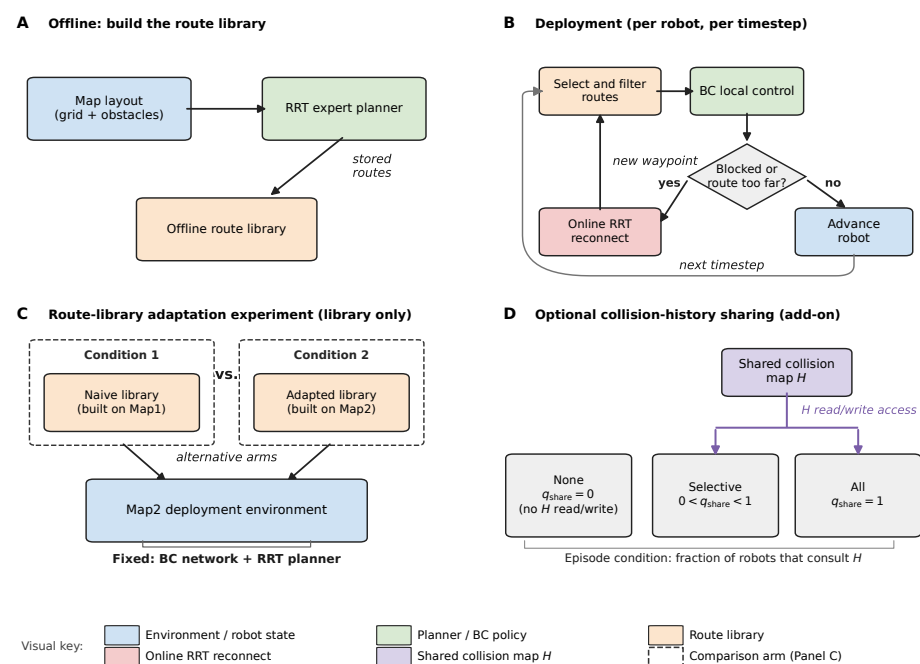


Figure 2. Hybrid RRT + Behavior Cloning framework. (A) Offline, an RRT expert uses the map to build the route library. (B) At deployment, each robot selects routes, applies BC local control, then

either reconnects with online RRT or advances to the next timestep. (C) The route-library adaptation experiment changes only the route library; BC network, RRT planner and Map2 deployment are fixed. (D) Optional sharing: the bracket marks non-zero- q_{share} conditions with H read/write access; q_{share} is an episode *condition*, not an output of H .

Algorithm 1 Per-robot decision step, run independently by each robot i on every scheduler tick. A predicted blocked-move is a counted event (Section 3.1), not a physical crash, and does not by itself halt the robot

Require: map M , route library R , BC policy π_{BC} , shared map H , sharing flag z_i , robot state x_i , goal g_i , counters c_i (steps) and h_i (collisions)

```

1:  $C \leftarrow \text{Get\_best\_routes}(R, x_i, g_i, M)$ 
2: if  $z_i$  and  $C \neq \emptyset$  then
3:    $r^* \leftarrow \arg \min_{r \in C} \sum_{c \in r} H[c]$  ▷ sharing robots prefer low- $H$  routes
4: else if  $C \neq \emptyset$  then
5:    $r^* \leftarrow$  route in  $C$  nearest  $x_i$ 
6: else
7:    $r^* \leftarrow$  current route if still valid, else  $\text{ONLINERTT}(x_i, g_i)$  ▷ nearest route too far/none available
8: end if
9:  $w \leftarrow$  next target waypoint on  $r^*$ 
10:  $a \leftarrow \pi_{\text{BC}}(x_i, w, W_{5 \times 5})$  ▷ BC drives toward  $w$ 
11: if three-step look-ahead along  $r^*$  predicts a blocked cell (obstacle or robot) then
12:    $\text{collisions} \leftarrow \text{collisions} + 1$ ; record cell in  $h_i$  ▷ counted event, not a crash
13:   with prob.  $1 - \rho_i$ :  $r^* \leftarrow \text{ONLINERTT}(x_i, g_i)$ ; recompute  $w, a$  ▷  $\rho_i$ : fraction of  $r^*$  done
14: end if
15:  $x_i \leftarrow$  advance by  $a$  along  $r^*$ ;  $c_i \leftarrow c_i + 1$ 
16: if  $x_i = g_i$  then ▷ task complete
17:    $H \leftarrow H + h_i$ ; register task; resample; reset  $c_i, h_i$ 
18: else if  $c_i > \text{step cap}$  then
19:   register failure; resample; reset  $c_i, h_i$ 
20: end if

```

Table 1. Implementation settings shared by all experiments.

Component	Setting
Grid size	50×50 occupancy grid
Obstacle inflation	1 cell (robot footprint)
Robot control	Decentralized; no direct inter-robot messaging
Collision-history sharing	Optional, via a shared map H (written and read; no messaging)
Collision definition	Predicted blocked-move (three-step look-ahead)
BC input	Robot position, target route waypoint, 5×5 local occupancy window
BC architecture	29-input MLP, hidden layers 256–256–64
Episode budget	2000 scheduler steps
Per-task step cap	300 steps (else counted as a failure)
Online RRT iteration cap	8000 (hybrid reconnect)/3000 (online baseline)
Seeds per condition	30
Fleet sizes	$N \in \{2, 4, 6, 8, 10\}$
Sharing fraction q_{share}	$\{0, 0.6, 1.0\}$ and a 0–1 sweep
Statistical tests	Mann–Whitney U , permutation (10^4), Spearman ρ

3.3. Route Libraries and Environment-Specific Adaptation

The framework’s prior knowledge lives in its route library. We compare two libraries in the Map2 deployment environment: a *Map2-naive* library (the “original” dataset, generated by the RRT expert on Map1) and a *Map2-adapted* library (the “alternative” dataset, whose routes were shaped by Map2’s obstacle configuration). Following the original pipeline,

each library is filtered for obstacle-free routes against the map on which it was *generated* (Map1 for the original library, Map2 for the adapted one) and then deployed on Map2; the Map2-naive library therefore retains routes that intersect Map2 obstacles, which is precisely the environment-mismatch the adapted library is meant to overcome. The two filtering stages are distinct: this offline filter is applied once against the *generation* map, whereas the runtime *Get_best_routes* filter (Algorithm 1) only rejects routes blocked in the robot's immediate local occupancy window (other robots and nearby cells). A Map1-generated route that crosses a Map2 obstacle outside that local window therefore survives both filters and stays active, so its three-step look-ahead eventually predicts a blocked move—the collision the diagnostics below quantify. This setup isolates the route-library-provenance effect: the BC network, planner, and deployment map are identical, and only the route library differs. Figure 3 makes the mismatch concrete: routes valid on Map1 (panel A) cross Map2's obstacles when deployed unchanged (panel B), and panels C,D illustrate the deployment consequence on two examples: a robot following a naive Map1 route reaches a blocked contact where the route would enter a Map2 obstacle, so online RRT must plan a fresh collision-free continuation to the goal—the runtime replanning that the static mismatch forces. Routes in panels C,D are shown after collision-checked simplification for display: every plotted segment is validated against the Map2 configuration-space obstacle, so simplification removes raw-planner jitter without moving a route into obstacle geometry. Across the entire library, 8.1% of the Map2-naive library's route cells—spanning 58% of its routes—fall inside Map2 obstacles. The Map2-adapted library has no such overlap by construction (its routes are generated by running RRT on Map2 directly); whether this static gap translates to a runtime benefit is the subject of Section 4 (Figures 4 and 5).

Table 2 quantifies the two libraries. They contain a comparable number of routes (3543 vs. 3592 raw; 3304 vs. 3256 after source-map filtering, within about 2%), so the Map2-adapted library is *not* simply larger than the Map2-naive one; its routes are modestly longer (median 33 vs. 37 cells) and cover somewhat more free space (1207 vs. 1363 unique cells). These differences are small relative to the non-trivial one: the Map2-naive library leaves 8.1% of its route cells, spanning 58% of its routes, inside Map2 obstacles, which is the static gap that drives the runtime collision difference reported in Section 4.

Table 2. Route-library diagnostics (both libraries deployed on Map2), computed from the released route datasets and map layouts. The two libraries hold a comparable number of routes (within about 2%), so the adapted library is not simply larger; their routes differ modestly in median length and free-cell coverage, but the decisive difference is that the Map2-naive library leaves route cells inside Map2 obstacles while the adapted one does not.

Library	Generated on	Raw Routes	Filtered Routes	Median Length	Free Cells Covered	Cells in Obstacles	Routes Overlapping
Map2-naive	Map1	3543	3304	33	1207	8.1%	58%
Map2-adapted	Map2	3592	3256	37	1363	0.0%	0%

To test whether the adaptation effect is specific to Map2's block geometry, this revision adds a third deployment environment. Map3 (Figure 1) is a corridor-style layout, three horizontal shelf bands with staggered bottleneck doors and a vertical divider, at an obstacle density (0.255) matched to Map1 (0.259). Its adapted library is generated with the same offline expert protocol as the existing libraries: seeded random start/goal pairs, the same RRT planner at the same iteration cap, 3600 raw routes stored and then filtered against the generation map, exactly as in Section 3.3. The Map3-naive condition deploys the Map1-generated library on Map3 unchanged, mirroring the Map2 experiment.

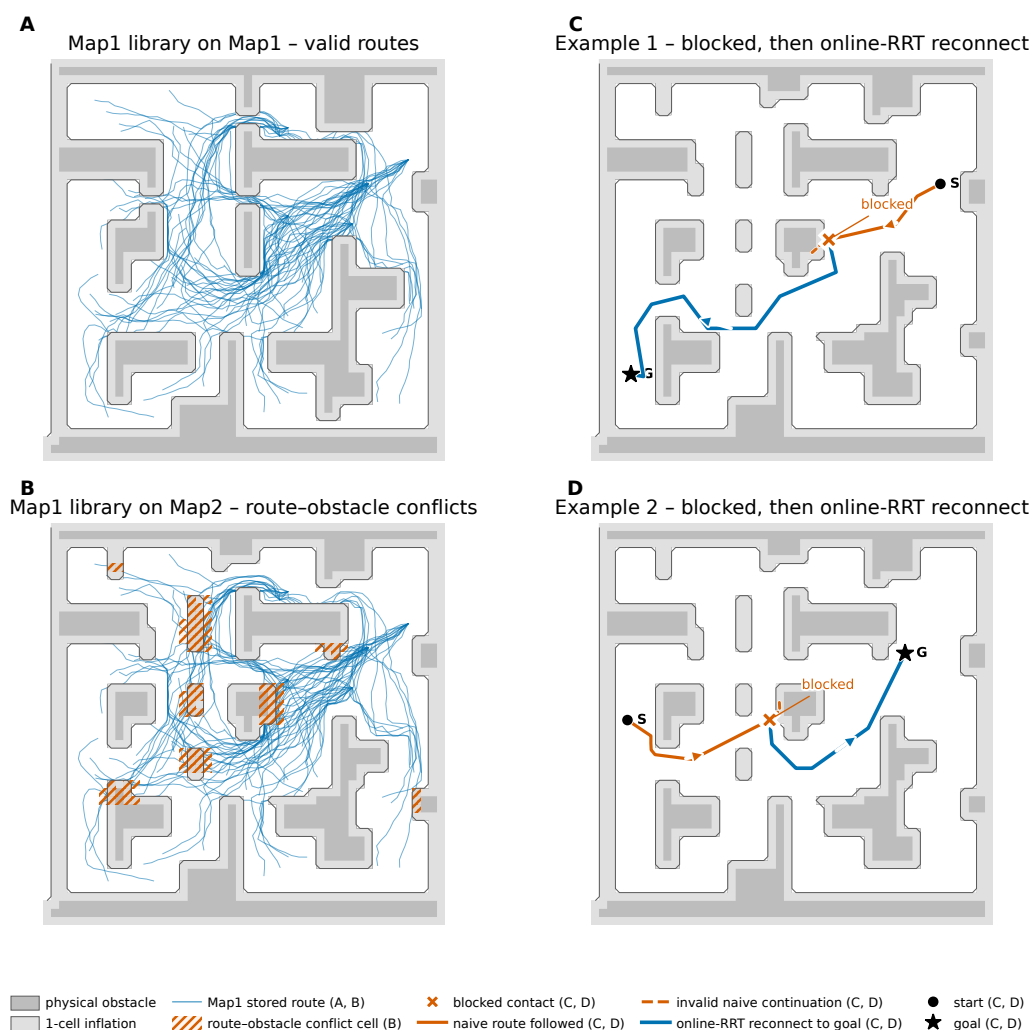


Figure 3. Route libraries encode the map on which they were built. Color and marker encodings are decoded in the bottom legend strip; the obstacle mask is shown in two gray tones (darker = physical obstacle, lighter = one-cell robot-footprint inflation) and is the same configuration-space mask the planner collision-checks against, so a valid route never overlaps it. (A) Sampled subset ($n = 80$) of the Map1-generated library on Map1 is valid. (B) The same sampled subset evaluated on Map2; orange hatching marks *route-obstacle conflict cells* (sampled route cells overlapping the Map2 obstacle mask). (C,D) Two examples of the deployment recovery step: the robot follows a naive Map1-library route (orange) from the start **S** until a blocked contact ($\times$) where the route would enter a Map2 obstacle (invalid continuation dashed); online RRT then plans a fresh collision-free continuation (blue) from the contact to the goal **G**. Arrows show travel direction.

3.4. Baselines and Ablations

Four baseline families, added in this revision, bound the adaptation effect from below and above. All reuse the exact episode protocol, metrics, and seeds of the main experiments.

Library-level remedies. Before deployment, the Map2-naive library is transformed offline in one of three ways a practitioner would try first: *filtering* drops every transferred route that intersects a deployment-map obstacle; *repair* keeps each invalid route's valid segments and reconnects the gaps with the same RRT planner running on the deployment map; *regeneration* replans each invalid route end-to-end on the deployment map. Valid routes pass through untouched, and each transform is applied once with a fixed seed, so all episodes deploy identical transformed libraries. Full regeneration of the entire library on the deployment map is the Map2-adapted condition itself, so the three remedies interpolate between the naive and adapted endpoints.

Non-learning connector. In the deployed framework, the trained BC network serves as a local connector that walks the robot onto a selected route whose start lies two to seven cells away. To test whether learning is necessary for this role, we replace it with a scripted greedy controller with the identical interface: unit steps toward the target with deterministic obstacle-avoiding deviations of $\pm 45^\circ$ and $\pm 90^\circ$, success within 1.5 cells, failure after 100 steps. Everything else, including the route library and planner, is unchanged.

Prioritized planning. As a mainstream multi-robot path-finding baseline we add decoupled prioritized planning [9] with space–time A^* over a shared reservation table [10], a standard decoupled multi-robot path-finding baseline commonly compared with conflict-based search and its bounded-suboptimal variants [11,12]. Robots plan in fixed priority order, reserving cells and edges in space–time; each new task triggers a replan around all current reservations. We state plainly that this baseline is *not* communication-free: the reservation table is shared, centralized information, so prioritized planning bounds what coordination through shared state can achieve on the same tasks rather than competing under the framework’s constraints. Execution-time blocked moves (which the reservation table makes structurally rare) are counted with the same collision metric.

Compute-budget frontier for online RRT. The online RRT baseline replans from scratch as tasks arrive, so its collision advantage is bought with planning compute. To make the comparison budget-aware (rather than only reporting its runtime), we rerun the online baseline with its per-call iteration cap reduced from the default 3000 to 1000 and to 300, tracing a compute-versus-quality frontier that can be compared directly against the hybrid’s runtime (Section 4).

3.5. Collision-History Sharing

As a lightweight add-on that needs no direct robot-to-robot message passing (proposed as future work in the originating thesis), a global per-cell map H accumulates the collisions logged during *completed* tasks: coordination, when enabled, is *indirect*—through this shared map rather than through messages exchanged between robots. The robots in a randomly chosen subset—a fraction q_{share} , fixed per episode—bias their route selection toward rarely-collided cells, selecting, among valid candidate routes, the one that minimizes the summed collision history $\sum_{c \in r} H[c]$ along its cells, while the rest plan normally. $q_{\text{share}} = 0$ is no sharing, $0 < q_{\text{share}} < 1$ selective, $q_{\text{share}} = 1$ full sharing. Every robot writes its own logged collisions into H upon completing a task, while only the sharing subset reads the aggregate; no robot exchanges messages with another. Collision records from tasks that *fail* (exceed the per-task step cap) are discarded with the task rather than added to H , so H reflects the collision history of completed tasks only. Because H accrues only as tasks complete, the mechanism can act only once enough collision history has been observed; we therefore run these episodes to a high task throughput (2000 scheduler steps, ≈ 245 completed tasks at five robots).

3.6. Experimental Design

With 30 seeds per condition (doubled from 15 in this revision to raise statistical power) we run eleven experiments (Table 3): (i) a *route-library adaptation* experiment on Map2 comparing the Map2-naive and Map2-adapted libraries across fleet sizes $N \in \{2, 4, 6, 8, 10\}$; (ii) an *online RRT baseline* over the same fleet sizes; (iii) a collision-history-sharing *ablation* ($q_{\text{share}} \in \{0, 0.6, 1.0\}$ at five robots) on both maps; (iv) an *access-fraction sweep* ($q_{\text{share}} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$); (v) a sharing *scalability sweep* ($N \in \{2, 4, 6, 8, 10\}$, with and without selective sharing); (vi) a *throughput-dependence sweep* (Map1, five robots, $q_{\text{share}} \in \{0, 0.6, 1.0\}$ over episode lengths from 300 to 2000 steps); and, new in this revision (Section 3.4), (vii) the *library-remedy* baselines (filter, repair, regener-

ate) on Map2; (viii) the *scripted-connector* ablation on Map2 with both libraries; (ix) a *prioritized-planning* baseline on Map2; (x) the online RRT *compute-budget* frontier (iteration caps 1000 and 300); and (xi) the *Map3 transfer* experiment (Map1-generated vs. Map3-generated library on Map3). Each of the main experiments runs for 2000 scheduler steps—about 245 completed tasks at five robots—so that the shared collision database fills before collision-rate comparisons are made; the throughput sweep additionally varies the episode length to expose the short-horizon regime in which the database stays nearly empty. We report mean $\pm$ SD, Mann–Whitney *U*-tests (one-sided, where directional), permutation tests (10^4 label permutations, reported alongside every headline Mann–Whitney result), Holm–Bonferroni correction within each comparison family, simulation-based post hoc power analyses for the sharing comparisons, and Spearman trends. Scripts (`sim_rrtbc.py`, `sim_rrt_online.py`, `sim_prioritized.py`, `scenario.py`, `lib_variants.py`, `gen_map3.py`, `run_experiments_rrtbc.py`, `run_experiments_rrt_online.py`, `run_experiments_revision.py`, `analyze_rrtbc.py`, `analyze_revision.py`, `make_heatmaps_rrtbc.py`) and raw results are available in the public repository.

Table 3. Experimental design. All conditions use 30 seeds; “library” is the route set deployed (naive = generated on Map1; adapted = generated on the deployment map; native = generated on the same map). Online RRT and prioritized planning use no library; the experiments below the rule are new in this revision.

Experiment	Map	Library	<i>N</i>	<i>q</i> _{share}	Steps
Route-library adaptation	Map2	naive vs. adapted	2, 4, 6, 8, 10	0	2000
Online RRT baseline	Map2	none (online)	2, 4, 6, 8, 10	—	2000
Sharing ablation	Map1, Map2	native	5	0, 0.6, 1.0	2000
Access-fraction sweep	Map1	native	5	0–1 (step 0.2)	2000
Sharing scalability	Map1	native	2, 4, 6, 8, 10	0, 0.6	2000
Throughput dependence	Map1	native	5	0, 0.6, 1.0	300–2000
Library remedies	Map2	naive: filter/repair/regen.	2, 4, 6, 8, 10	0	2000
Scripted connector	Map2	naive, adapted	2, 4, 6, 8, 10	0	2000
Prioritized planning	Map2	none (reservations)	2, 4, 6, 8, 10	—	2000
Online RRT budget caps	Map2	none (online)	2, 4, 6, 8, 10	—	2000
Map3 transfer	Map3	naive vs. adapted	2, 4, 6, 8, 10	0	2000

4. Results

The figures separate three effects. Figure 4 measures the result of route-library adaptation; Figures 5 and 6 diagnose the failure mechanism by locating and classifying collisions; and Figures 7 and 8 quantify the computational trade-off against online RRT. The final Figures 9–11 evaluate the optional collision-history-sharing add-on.

4.1. Environment-Adapted Route Libraries Restore Navigation Quality

Deploying the Map2-adapted library instead of the Map2-naive library on Map2 sharply improves navigation (Figure 4). At four robots, the Map2-naive library incurs 718 ± 62 collisions and a 0.23 task-failure rate, while the Map2-adapted library incurs 210 ± 34 collisions and a 0.08 failure rate—a 71% reduction in collisions (Mann–Whitney *U* on per-episode collision counts, $p < 10^{-5}$) and a 64% reduction in failures (the significance test is on collisions; failure rates are reported descriptively; full per-fleet-size numbers, bootstrap CIs and effect sizes in Table 4). The reduction is largest where obstacle mismatch dominates (85% at two robots) and remains substantial as agent–agent interference grows (55% at six, 38% at ten robots); the adapted library’s collision rate (per completed task) and failure rate stay below the naive library’s at every fleet size $N \in \{2, 4, 6, 8, 10\}$ tested, all with $p < 10^{-5}$. This is the environment-specific route-library-adaptation effect: the

BC network and planner are unchanged, so the gain comes entirely from aligning the route library with the deployment geometry. The collision heatmaps (Figure 5) make the mechanism visible—the Map2-naive library produces collisions inside Map2’s obstacle regions (routes inherited from Map1 cut through walls that did not exist there), whereas the Map2-adapted library confines residual collisions to natural high-traffic corridors.

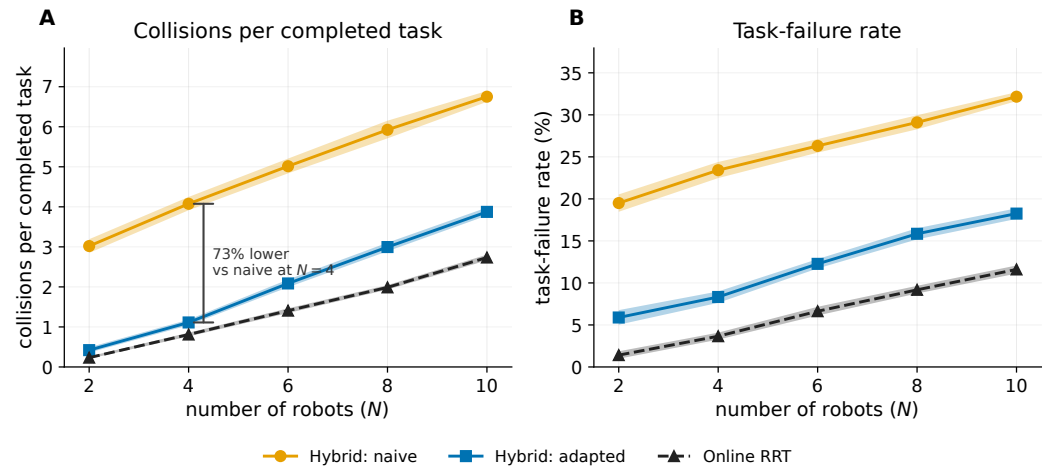


Figure 4. Map2 transfer performance versus fleet size N , over 30 seeds. Markers show seed means; (A) collisions per *completed* task and (B) task-failure rate (failed tasks as a percentage of attempted tasks). Shaded bands are percentile-bootstrap 95% confidence intervals for the mean across seeds. Three methods are compared: the hybrid framework with a Map2-naive route library (orange), the hybrid framework with a Map2-adapted library (blue), and online RRT (black, dashed). The Map2-adapted library substantially improves the hybrid framework relative to the Map2-naive library across fleet sizes—reducing collisions by, e.g., about 73% at $N = 4$ (bracket in panel (A))—while online RRT remains the lowest-rate replanning baseline on both metrics.

Table 4. Full route-library adaptation results on Map2 (30 seeds, 2000 scheduler steps). Tasks is the mean number of completed tasks (rounded to an integer); collisions are mean $\pm$ SD; collisions per completed task are reported with a percentile bootstrap 95% CI. The Reduction column is the percentage reduction in mean collisions relative to the Map2-naive hybrid; the Mann–Whitney U p -value (one-sided, naive > method), the one-sided permutation p_{perm} (10^4 resamples), and Cliff’s δ are computed against that same baseline. The Map2-adapted hybrid and the online RRT baseline both improve on the naive library at every fleet size, with $\delta = 1$ and $p < 10^{-5}$ throughout.

N	Method	Tasks	Collisions	Coll./Task [95% CI]	Fail Rate	Reduction	p	p_{perm}	δ
2	Map2-naive hybrid	93	279 $\pm$ 37	3.02 [2.85, 3.19]	0.20	—	—	—	—
	Map2-adapted hybrid	99	41 $\pm$ 18	0.42 [0.36, 0.49]	0.06	85%	$<10^{-5}$	<0.001	1.00
	Online RRT	118	27 $\pm$ 10	0.23 [0.20, 0.26]	0.01	90%	$<10^{-5}$	<0.001	1.00
4	Map2-naive hybrid	177	718 $\pm$ 62	4.08 [3.91, 4.25]	0.23	—	—	—	—
	Map2-adapted hybrid	190	210 $\pm$ 34	1.11 [1.04, 1.18]	0.08	71%	$<10^{-5}$	<0.001	1.00
	Online RRT	221	178 $\pm$ 27	0.81 [0.76, 0.86]	0.04	75%	$<10^{-5}$	<0.001	1.00
6	Map2-naive hybrid	255	1272 $\pm$ 94	5.01 [4.84, 5.20]	0.26	—	—	—	—
	Map2-adapted hybrid	275	573 $\pm$ 69	2.09 [1.99, 2.18]	0.12	55%	$<10^{-5}$	<0.001	1.00
	Online RRT	311	436 $\pm$ 46	1.41 [1.35, 1.46]	0.07	66%	$<10^{-5}$	<0.001	1.00
8	Map2-naive hybrid	329	1942 $\pm$ 124	5.92 [5.71, 6.15]	0.29	—	—	—	—
	Map2-adapted hybrid	350	1045 $\pm$ 87	3.00 [2.89, 3.10]	0.16	46%	$<10^{-5}$	<0.001	1.00
	Online RRT	396	787 $\pm$ 54	1.99 [1.93, 2.04]	0.09	59%	$<10^{-5}$	<0.001	1.00
10	Map2-naive hybrid	399	2689 $\pm$ 113	6.75 [6.62, 6.89]	0.32	—	—	—	—
	Map2-adapted hybrid	427	1654 $\pm$ 105	3.87 [3.78, 3.98]	0.18	38%	$<10^{-5}$	<0.001	1.00
	Online RRT	466	1274 $\pm$ 76	2.73 [2.66, 2.80]	0.12	53%	$<10^{-5}$	<0.001	1.00

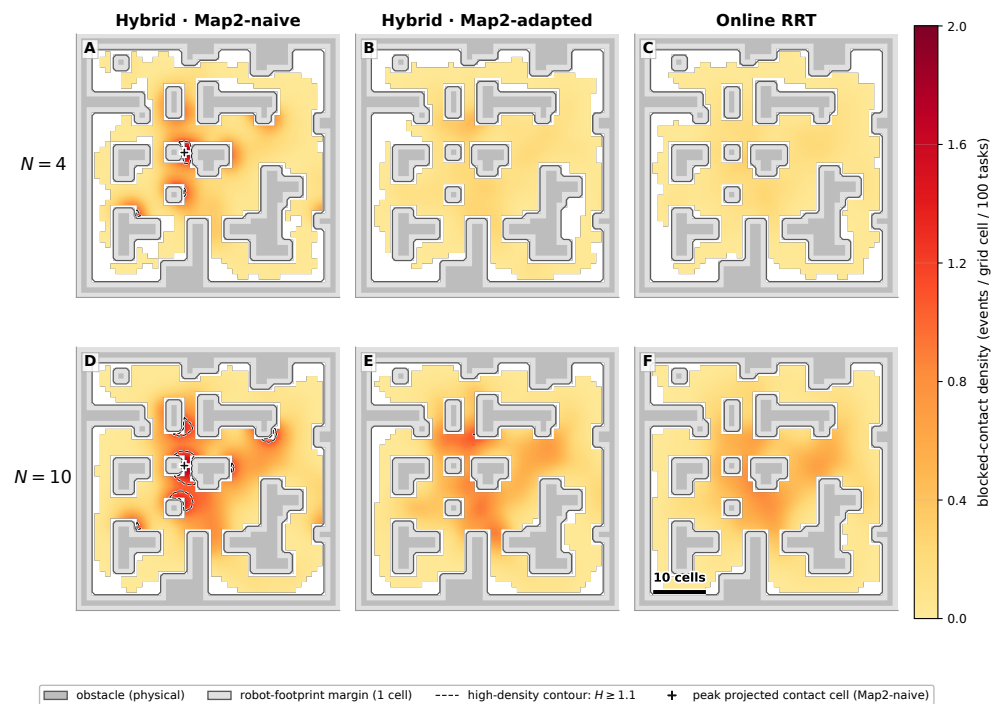


Figure 5. Projected blocked-contact density on Map2 (panels (A–F); rows = fleet size $N = 4$, $N = 10$; columns = method). A blocked contact is counted when a robot’s three-step look-ahead predicts the next cell is occupied; the event is assigned to the free cell immediately preceding the blocked transition and smoothed within the free-space mask with a mass-conserving Gaussian kernel, so the density sits on the reachable side of the geometry and never inside it. Obstacles are drawn opaque above the density in two grays (darker core = physical obstacle, lighter one-cell ring = robot-footprint/configuration-space inflation) with a boundary outline; white = zero (no displayed density). Color encodes blocked-contact events per grid cell per 100 completed tasks, aggregated over 30 seeds on a single shared scale (color bar); the scale bar in panel (F) spans ten cells. A dashed contour at the same absolute level $H \geq 1.1$ in every panel bounds the high-density regions, and a small crosshair marks the maximum of the plotted density in the Map2-naive panels (A,D); both annotations are read directly from the displayed field. The Map2-naive hybrid concentrates blocked contacts against the obstacles its transferred Map1 routes steer into, whereas the Map2-adapted hybrid and online RRT leave only diffuse residual contacts along the central corridors—reaching the high-density contour nowhere.

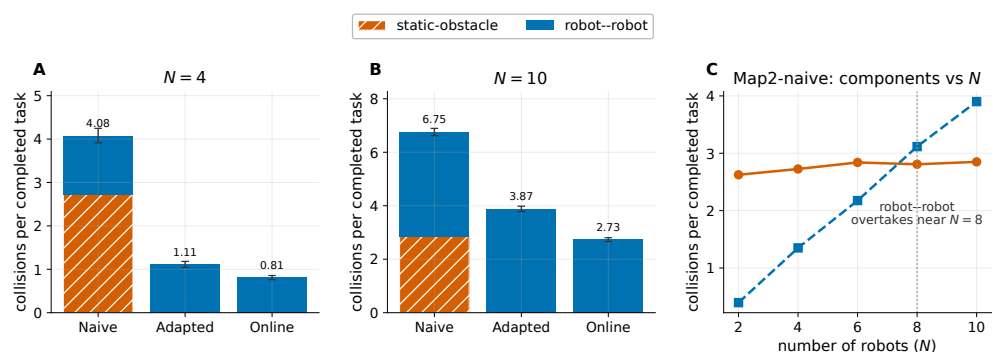


Figure 6. Collision composition on Map2 (collisions per completed task), split into static-obstacle (hatched) and robot–robot events. (A,B) Stacked bars at $N = 4$ and $N = 10$ for the three methods (error bars: bootstrap 95% CI of the total). (C) The Map2-naive library’s two components versus fleet size: a large static-obstacle component (absent from the other two methods) dominates at small fleets and persists, while robot–robot congestion grows with N and overtakes it near $N = 8$. The adapted hybrid and online RRT incur essentially no obstacle collisions (all fleet sizes in Table 5).

Table 5. Collision composition on Map2 at all fleet sizes (30 seeds, 2000 steps): static-obstacle and robot–robot collisions per completed task, and the static-obstacle fraction. The static-obstacle component is exclusive to the Map2-naive library and roughly constant across N (it dominates at small fleets and is overtaken by robot–robot congestion at the largest fleets); the adapted hybrid and online RRT incur no obstacle collisions at any N . Full-fleet complement to Figure 6.

N	Method	Static-Obstacle/Task	Robot–Robot/Task	Static Fraction
2	Map2-naive hybrid	2.62	0.40	87%
	Map2-adapted hybrid	0.00	0.42	0%
	Online RRT	0.00	0.23	0%
4	Map2-naive hybrid	2.73	1.35	67%
	Map2-adapted hybrid	0.00	1.11	0%
	Online RRT	0.00	0.81	0%
6	Map2-naive hybrid	2.84	2.18	57%
	Map2-adapted hybrid	0.00	2.09	0%
	Online RRT	0.00	1.41	0%
8	Map2-naive hybrid	2.81	3.12	47%
	Map2-adapted hybrid	0.00	3.00	0%
	Online RRT	0.00	1.99	0%
10	Map2-naive hybrid	2.85	3.90	42%
	Map2-adapted hybrid	0.00	3.87	0%
	Online RRT	0.00	2.73	0%

4.2. Library-Level Baselines Bound the Adaptation Effect

Tables 6 and 7 place the adaptation effect among the baselines of Section 3.4: Table 6 covers the offline library-level remedies and Table 7 the connector ablation and the alternative planners. The three offline library remedies recover most of the gap between the Map2-naive and Map2-adapted libraries: filtering (which keeps 1387 of the 3600 transferred routes) cuts collisions by 43–86% relative to the naive library across fleet sizes, repair by 39–86%, and full regeneration by 47–89%, bracketing the adapted library’s 38–85%; all fifteen comparisons survive Holm correction within the family ($p < 10^{-5}$, permutation tests in agreement). Filtering removes the obstacle-crossing routes (and with them most of the static-obstacle collision component) at the cost of a thinner library, repair additionally restores coverage, and regeneration attains the lowest collision and failure rates of the three at every fleet size. Replacing the trained BC connector with the scripted greedy controller does not degrade navigation; the scripted variant incurs *fewer* mean collisions at every fleet size (Mann–Whitney BC $>$ scripted, $p = 0.048$ and $p = 0.026$ at two and four robots, $p < 10^{-5}$ at six to ten; e.g., 1486 ± 112 vs. 1654 ± 105 per episode at ten robots) with a lower task-failure rate. The learned connector is therefore not what carries the framework’s navigation quality on these grids, which is consistent with, and sharpens, the central claim of this paper: the environment-specific knowledge lives in the route library, not in the connector that follows it (Section 5). The same holds on the Map2-naive library, where the scripted connector also incurs fewer mean collisions than the BC hybrid at every fleet size (e.g., 2431 ± 117 vs. 2689 ± 113 per episode at ten robots), so the connector choice does not interact with library provenance. Prioritized planning over a shared reservation table, which coordinates through centralized information the decentralized framework does not use, dominates every communication-free method on this benchmark: it completes roughly twice as many tasks (e.g., 865 vs. 427 per episode at ten robots) with near-zero blocked moves (0.02–0.14 per completed task) and a failure rate at or below 0.01, bounding from above what coordination through shared state achieves on the same tasks.

Table 6. Offline library-level remedies on Map2 (30 seeds, 2000 steps). Tasks is the mean number of completed tasks; collisions are mean $\pm$ SD per episode; collisions per completed task carry a percentile bootstrap 95% CI. Reduction, p and p_{perm} compare each method against the Map2-naive hybrid on mean per-episode collision counts: Reduction is the percentage reduction in mean collisions, p is a one-sided Mann–Whitney U -test (naive $>$ method) and p_{perm} is a one-sided permutation test (10^4 resamples) in the same direction. The filtered, repaired and regenerated rows apply offline transforms to the naive library before deployment; the Map2-adapted row deploys the library generated on Map2 itself. Row labels prefixed with “+” denote the Map2-naive library after the named offline transform (filtered, repaired or regenerated).

<i>N</i>	Method	Tasks	Collisions	Coll./Task [95% CI]	Fail Rate	Reduction	p	p_{perm}
2	Map2-naive hybrid	93	279 $\pm$ 37	3.02 [2.85, 3.19]	0.20	—	—	—
	+ Filtered library	99	40 $\pm$ 19	0.40 [0.33, 0.48]	0.11	86%	$<10^{-5}$	<0.001
	+ Repaired library	98	39 $\pm$ 17	0.41 [0.34, 0.48]	0.10	86%	$<10^{-5}$	<0.001
	+ Regenerated library	118	32 $\pm$ 10	0.27 [0.24, 0.30]	0.07	89%	$<10^{-5}$	<0.001
	Map2-adapted hybrid	99	41 $\pm$ 18	0.42 [0.36, 0.49]	0.06	85%	$<10^{-5}$	<0.001
4	Map2-naive hybrid	177	718 $\pm$ 62	4.08 [3.91, 4.25]	0.23	—	—	—
	+ Filtered library	195	204 $\pm$ 32	1.04 [0.98, 1.10]	0.15	72%	$<10^{-5}$	<0.001
	+ Repaired library	188	222 $\pm$ 44	1.18 [1.10, 1.27]	0.13	69%	$<10^{-5}$	<0.001
	+ Regenerated library	226	196 $\pm$ 33	0.87 [0.82, 0.93]	0.10	73%	$<10^{-5}$	<0.001
	Map2-adapted hybrid	190	210 $\pm$ 34	1.11 [1.04, 1.18]	0.08	71%	$<10^{-5}$	<0.001
6	Map2-naive hybrid	255	1272 $\pm$ 94	5.01 [4.84, 5.20]	0.26	—	—	—
	+ Filtered library	277	543 $\pm$ 63	1.96 [1.87, 2.05]	0.19	57%	$<10^{-5}$	<0.001
	+ Repaired library	274	555 $\pm$ 60	2.03 [1.95, 2.12]	0.16	56%	$<10^{-5}$	<0.001
	+ Regenerated library	329	469 $\pm$ 46	1.43 [1.37, 1.48]	0.13	63%	$<10^{-5}$	<0.001
	Map2-adapted hybrid	275	573 $\pm$ 69	2.09 [1.99, 2.18]	0.12	55%	$<10^{-5}$	<0.001
8	Map2-naive hybrid	329	1942 $\pm$ 124	5.92 [5.71, 6.15]	0.29	—	—	—
	+ Filtered library	357	986 $\pm$ 84	2.77 [2.67, 2.88]	0.21	49%	$<10^{-5}$	<0.001
	+ Repaired library	352	1016 $\pm$ 85	2.89 [2.79, 2.99]	0.20	48%	$<10^{-5}$	<0.001
	+ Regenerated library	425	874 $\pm$ 88	2.06 [1.98, 2.14]	0.15	55%	$<10^{-5}$	<0.001
	Map2-adapted hybrid	350	1045 $\pm$ 87	3.00 [2.89, 3.10]	0.16	46%	$<10^{-5}$	<0.001
10	Map2-naive hybrid	399	2689 $\pm$ 113	6.75 [6.62, 6.89]	0.32	—	—	—
	+ Filtered library	430	1521 $\pm$ 117	3.55 [3.42, 3.67]	0.24	43%	$<10^{-5}$	<0.001
	+ Repaired library	424	1641 $\pm$ 111	3.87 [3.77, 3.98]	0.23	39%	$<10^{-5}$	<0.001
	+ Regenerated library	515	1417 $\pm$ 113	2.76 [2.67, 2.85]	0.18	47%	$<10^{-5}$	<0.001
	Map2-adapted hybrid	427	1654 $\pm$ 105	3.87 [3.78, 3.98]	0.18	38%	$<10^{-5}$	<0.001

4.3. The Adaptation Effect Replicates on a Third Layout

Table 8 repeats the transfer experiment on Map3, whose corridor structure, bottleneck doors, and vertical divider differ qualitatively from Map2’s block geometry. Deploying the Map1-generated library unchanged leaves robots nearly unable to navigate: the corridor walls block most transferred routes, producing 30–38 collisions per completed task and task-failure rates of 0.77–0.80. Regenerating the library on Map3 removes 65–96% of collisions (largest at two robots and decreasing with fleet size as agent–agent interference grows, mirroring Map2), restores per-task collision rates to 0.49–4.90, and drops the failure rate to 0.14–0.25; every comparison survives Holm correction ($p < 10^{-5}$, permutation agreement, Cliff’s $\delta = 1.00$ at all five fleet sizes). This replication supports reading the Map2 result as a property of route-library provenance rather than of one particular layout, although two deployment layouts remain a limited sample of industrial geometry (Section 5).

Table 7. Alternative planners and connector ablation on Map2 (30 seeds, 2000 steps). The Map2-naive hybrid row is repeated from Table 6 as the shared reference for the Reduction column. Tasks is the mean number of completed tasks; collisions are mean $\pm$ SD per episode; collisions per completed task carry a percentile bootstrap 95% CI. Reduction, p and p_{perm} compare each method against the Map2-naive hybrid on mean per-episode collision counts: Reduction is the percentage reduction in mean collisions, p is a one-sided Mann–Whitney U -test (naive $>$ method) and p_{perm} is a one-sided permutation test (10^4 resamples) in the same direction. The scripted row replaces the trained BC connector with a greedy non-learning controller on the adapted library (the corresponding naive-library scripted comparison is summarized in the text); prioritized planning coordinates the fleet through a shared space–time reservation table, so unlike the hybrid it is not communication-free; the online RRT rows replan continuously under the stated iteration cap.

N	Method	Tasks	Collisions	Coll./Task [95% CI]	Fail Rate	Reduction	p	p_{perm}
2	Map2-naive hybrid	93	279 $\pm$ 37	3.02 [2.85, 3.19]	0.20	—	—	—
	Scripted connector (adapted)	99	33 $\pm$ 15	0.34 [0.29, 0.40]	0.02	88%	$<10^{-5}$	<0.001
	Prioritized planning	173	3 $\pm$ 2	0.02 [0.02, 0.02]	0.00	99%	$<10^{-5}$	<0.001
	Online RRT (cap 3000)	118	27 $\pm$ 10	0.23 [0.20, 0.26]	0.01	90%	$<10^{-5}$	<0.001
	Online RRT (cap 1000)	120	30 $\pm$ 13	0.26 [0.22, 0.30]	0.05	89%	$<10^{-5}$	<0.001
	Online RRT (cap 300)	147	46 $\pm$ 18	0.31 [0.27, 0.36]	0.44	84%	$<10^{-5}$	<0.001
4	Map2-naive hybrid	177	718 $\pm$ 62	4.08 [3.91, 4.25]	0.23	—	—	—
	Scripted connector (adapted)	192	194 $\pm$ 36	1.01 [0.94, 1.08]	0.04	73%	$<10^{-5}$	<0.001
	Prioritized planning	347	16 $\pm$ 3	0.05 [0.04, 0.05]	0.00	98%	$<10^{-5}$	<0.001
	Online RRT (cap 3000)	221	178 $\pm$ 27	0.81 [0.76, 0.86]	0.04	75%	$<10^{-5}$	<0.001
	Online RRT (cap 1000)	230	186 $\pm$ 28	0.81 [0.77, 0.86]	0.09	74%	$<10^{-5}$	<0.001
	Online RRT (cap 300)	290	225 $\pm$ 37	0.78 [0.73, 0.82]	0.48	69%	$<10^{-5}$	<0.001
6	Map2-naive hybrid	255	1272 $\pm$ 94	5.01 [4.84, 5.20]	0.26	—	—	—
	Scripted connector (adapted)	280	475 $\pm$ 74	1.70 [1.60, 1.80]	0.05	63%	$<10^{-5}$	<0.001
	Prioritized planning	521	35 $\pm$ 4	0.07 [0.07, 0.07]	0.00	97%	$<10^{-5}$	<0.001
	Online RRT (cap 3000)	311	436 $\pm$ 46	1.41 [1.35, 1.46]	0.07	66%	$<10^{-5}$	<0.001
	Online RRT (cap 1000)	332	462 $\pm$ 42	1.39 [1.34, 1.45]	0.14	64%	$<10^{-5}$	<0.001
	Online RRT (cap 300)	428	551 $\pm$ 55	1.29 [1.24, 1.34]	0.53	57%	$<10^{-5}$	<0.001
8	Map2-naive hybrid	329	1942 $\pm$ 124	5.92 [5.71, 6.15]	0.29	—	—	—
	Scripted connector (adapted)	363	891 $\pm$ 66	2.45 [2.39, 2.53]	0.07	54%	$<10^{-5}$	<0.001
	Prioritized planning	693	75 $\pm$ 10	0.11 [0.10, 0.11]	0.00	96%	$<10^{-5}$	<0.001
	Online RRT (cap 3000)	396	787 $\pm$ 54	1.99 [1.93, 2.04]	0.09	59%	$<10^{-5}$	<0.001
	Online RRT (cap 1000)	422	844 $\pm$ 55	2.00 [1.95, 2.05]	0.18	57%	$<10^{-5}$	<0.001
	Online RRT (cap 300)	557	1047 $\pm$ 90	1.88 [1.82, 1.96]	0.57	46%	$<10^{-5}$	<0.001
10	Map2-naive hybrid	399	2689 $\pm$ 113	6.75 [6.62, 6.89]	0.32	—	—	—
	Scripted connector (adapted)	434	1486 $\pm$ 112	3.43 [3.32, 3.54]	0.10	45%	$<10^{-5}$	<0.001
	Prioritized planning	865	120 $\pm$ 10	0.14 [0.13, 0.14]	0.01	96%	$<10^{-5}$	<0.001
	Online RRT (cap 3000)	466	1274 $\pm$ 76	2.73 [2.66, 2.80]	0.12	53%	$<10^{-5}$	<0.001
	Online RRT (cap 1000)	511	1350 $\pm$ 73	2.65 [2.59, 2.71]	0.22	50%	$<10^{-5}$	<0.001
	Online RRT (cap 300)	689	1616 $\pm$ 92	2.35 [2.29, 2.41]	0.60	40%	$<10^{-5}$	<0.001

4.4. Throughput, Collision Composition, and Planning Cost Across Fleet Density

Throughput (tasks completed within the fixed budget) increases with fleet size on Map2: the completed-task counts in Table 4 rise monotonically with N for every method. The collision rate per completed task also increases gradually with density as path intersections become more frequent, reproducing the qualitative scalability behavior reported in the originating thesis; the same throughput trend appears in the Map1 scalability sweep (Figure 9, left; Spearman $\rho = +0.98$ between fleet size and tasks, $p < 0.001$). Decomposing collisions by type (Figure 6) shows why: the Map2-naive library carries a large *static-obstacle* collision component—roughly 2.7–2.9 collisions per completed task at every fleet size (its routes cross Map2 walls)—that is essentially absent from the Map2-adapted hybrid and from online RRT, both of which plan around the static obstacles and so incur no obstacle

collisions. This map-mismatch component dominates the naive library's collisions at small fleets and persists undiminished as N grows, even as *robot-robot* congestion—the residual common to all three methods—rises with density and overtakes it at the largest fleets. Figure 6 shows $N = 4$ and $N = 10$; the complete breakdown for all fleet sizes is given in Table 5. This separates the map-mismatch failure mode (curable by adapting the library and roughly constant in N) from the remaining density-driven congestion.

Table 8. Transfer on Map3 (corridor/bottleneck layout; 30 seeds, 2000 steps). The Map3-naive hybrid deploys the Map1-generated route library on Map3 unchanged; the Map3-adapted hybrid uses a library generated on Map3 itself. Collisions are mean $\pm$ SD per episode and collisions per completed task carry a percentile bootstrap 95% CI. Reduction, p , p_{perm} and Cliff's δ are computed against the Map3-naive hybrid on per-episode collision counts, with p from a one-sided Mann–Whitney U -test (naive $>$ adapted) and p_{perm} from a one-sided permutation test (10^4 resamples) in the same direction.

N	Method	Tasks	Collisions	Coll./Task [95% CI]	Fail Rate	Reduction	p	p_{perm}	δ
2	Map3-naive hybrid	33	983 $\pm$ 63	30.47 [28.22, 32.97]	0.77	—	—	—	—
	Map3-adapted hybrid	90	44 $\pm$ 16	0.49 [0.43, 0.55]	0.14	96%	$<10^{-5}$	<0.001	1.00
4	Map3-naive hybrid	64	2086 $\pm$ 77	32.95 [31.46, 34.51]	0.78	—	—	—	—
	Map3-adapted hybrid	178	285 $\pm$ 42	1.61 [1.52, 1.70]	0.18	86%	$<10^{-5}$	<0.001	1.00
6	Map3-naive hybrid	92	3286 $\pm$ 100	36.27 [34.68, 38.11]	0.80	—	—	—	—
	Map3-adapted hybrid	264	673 $\pm$ 71	2.56 [2.45, 2.68]	0.19	80%	$<10^{-5}$	<0.001	1.00
8	Map3-naive hybrid	127	4476 $\pm$ 102	35.40 [34.29, 36.47]	0.79	—	—	—	—
	Map3-adapted hybrid	338	1277 $\pm$ 107	3.79 [3.65, 3.94]	0.22	71%	$<10^{-5}$	<0.001	1.00
10	Map3-naive hybrid	155	5862 $\pm$ 138	38.02 [36.96, 39.15]	0.80	—	—	—	—
	Map3-adapted hybrid	418	2047 $\pm$ 120	4.90 [4.76, 5.04]	0.25	65%	$<10^{-5}$	<0.001	1.00

Compared to the online RRT baseline (Figure 4), online RRT achieves the lowest collision and failure rates—unsurprisingly, as it computes a fresh path toward each goal—but at a steep and growing computational cost (Figure 8, Table 9): its search had to be bounded (capped RRT iterations) merely to remain tractable at higher densities, and each episode was markedly slower than the route-reusing hybrid, which invokes the planner only to reconnect. The environment-adapted hybrid closes most of the gap to online RRT (e.g., 1.11 vs. 0.81 collisions per task at four robots, 3.87 vs. 2.73 at ten) while avoiding fresh planning for every task—the efficiency trade-off that motivates the hybrid design. To make this comparison budget-aware rather than merely descriptive, the frontier in Figure 7 reruns the online baseline at reduced per-call iteration caps (1000 and 300, versus the default 3000) alongside the hybrid variants and prioritized planning. The outcome is more nuanced than the descriptive comparison suggests. Tightening the cap collapses the online baseline's runtime (922 to 88 to 53 s per episode at ten robots) while leaving collisions per completed task nearly unchanged (2.73, 2.65, 2.35): RRT's per-iteration cost grows with tree size, so most of the default-cap runtime is spent inside searches that run to the cap. What the larger budget buys is *reliability*: the task-failure rate rises from 0.12 (cap 3000) to 0.22 (cap 1000) to 0.60 (cap 300) at ten robots, so at the tightest cap a majority of assigned tasks are abandoned and the per-completed-task statistics cover only the easier tasks. At the intermediate cap, online replanning is competitive with the adapted hybrid on this benchmark (lower runtime and per-task collisions at a somewhat higher failure rate), so the budget-fair comparison narrows the hybrid's advantage: what remains is its far lower planner usage (at most 0.05 planner calls per completed task versus 1.2–2.5 for the default-cap online baseline, and up to 3.7 at the reduced caps, where failed attempts recur), which matters when each planning call is expensive. We report this narrowing openly; it sharpens, rather

than undermines, the contribution, which is the route-library adaptation analysis, not a superiority claim for the hybrid over online planning.

Table 9. Planning cost on Map2 (30 seeds, 2000 steps). Runtime is mean $\pm$ SD wall-clock for the navigation loop on one pinned CPU thread (11th Gen Intel Core i7-11700KF, 3.60 GHz; Intel Corporation, Santa Clara, CA, USA; Python 3.12.3), excluding scenario construction, figure rendering and disk I/O; the RRT-calls-per-task column is the number of online RRT planner invocations per completed task. Online RRT attains the lowest collision rate but at far higher planning cost; the adapted hybrid reuses routes and calls the planner rarely.

<i>N</i>	Method	Runtime (s)	RRT Calls/Task	Coll./Task	Fail Rate
2	Map2-naive hybrid	28.7 $\pm$ 4.9	0.07	3.02	0.20
	Map2-adapted hybrid	22.8 $\pm$ 2.8	0.02	0.42	0.06
	Online RRT	112.7 $\pm$ 79.7	1.15	0.23	0.01
4	Map2-naive hybrid	60.7 $\pm$ 6.5	0.08	4.08	0.23
	Map2-adapted hybrid	49.5 $\pm$ 2.9	0.03	1.11	0.08
	Online RRT	262.0 $\pm$ 158.0	1.44	0.81	0.04
6	Map2-naive hybrid	110.1 $\pm$ 10.7	0.12	5.01	0.26
	Map2-adapted hybrid	87.8 $\pm$ 7.3	0.04	2.09	0.12
	Online RRT	462.4 $\pm$ 251.3	1.77	1.41	0.07
8	Map2-naive hybrid	153.0 $\pm$ 13.7	0.14	5.92	0.29
	Map2-adapted hybrid	132.4 $\pm$ 9.8	0.05	3.00	0.16
	Online RRT	655.6 $\pm$ 330.1	2.07	1.99	0.09
10	Map2-naive hybrid	210.0 $\pm$ 19.6	0.16	6.75	0.32
	Map2-adapted hybrid	184.6 $\pm$ 12.1	0.05	3.87	0.18
	Online RRT	921.9 $\pm$ 470.9	2.45	2.73	0.12

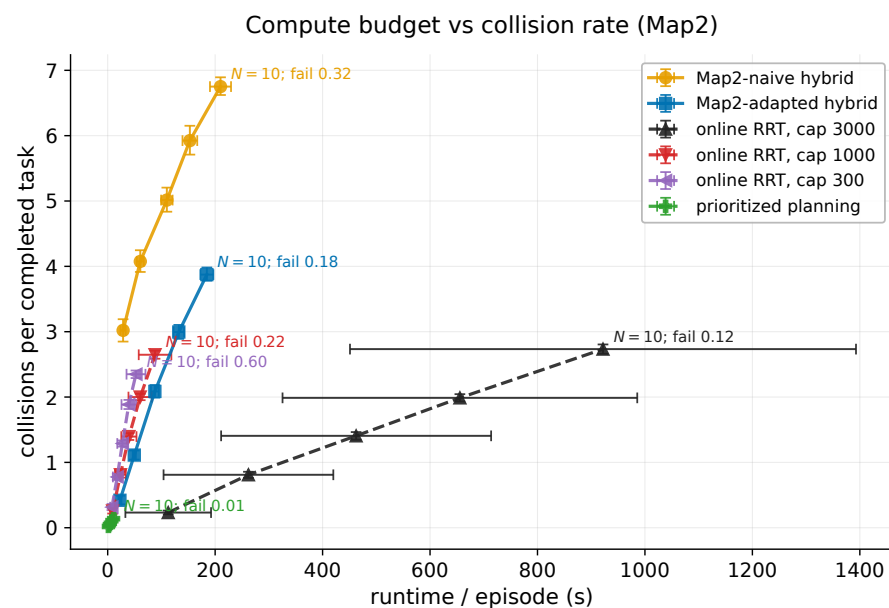


Figure 7. Compute-budget frontier on Map2: mean episode runtime against collisions per completed task for the hybrid variants, prioritized planning, and the online RRT baseline at three planning-budget caps (3000, 1000, 300 iterations per call), one marker per method and fleet size; each method's $N = 10$ marker is annotated with its task-failure rate. Reducing the online planner's budget collapses its runtime with little change in collisions per completed task, but its task-failure rate rises steeply (0.12, 0.22, 0.60 at ten robots for caps 3000, 1000, 300), so its completed-task statistics increasingly reflect only the easier tasks; prioritized planning, which coordinates through a centralized reservation table, dominates all communication-free methods.

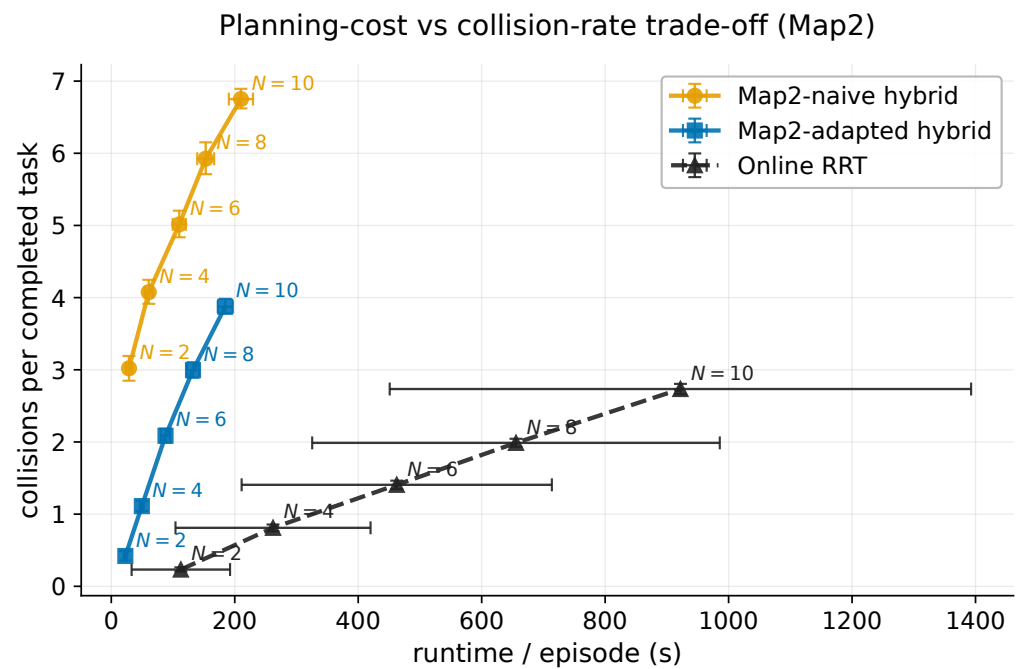


Figure 8. Planning-cost vs. collision-rate trade-off on Map2: mean wall-clock runtime per episode against collisions per completed task, one marker per method and fleet size ($N = 2, 4, 6, 8, 10$, annotated); horizontal bars are the across-seed runtime standard deviation and vertical bars the percentile bootstrap 95% confidence interval on collisions per completed task. Online RRT reaches the lowest collision rate but at roughly five times the Map2-adapted hybrid's runtime (e.g., 922 s vs. 185 s at ten robots, calling the planner ~ 2.5 times per task versus at most 0.05 for the Map2-adapted hybrid); the Map2-naive hybrid calls it more often, up to 0.16 per task); the Map2-adapted hybrid recovers most of that collision reduction at a fraction of the cost, while the Map2-naive hybrid costs about as much yet collides far more (Table 9).

4.5. Collision-History Sharing Gives Only a Limited Benefit

Evaluated at a high task throughput (2000 scheduler steps, ≈ 245 completed tasks at five robots), collision-history sharing yields only a limited benefit within the full framework (Table 10). In that table, the Change column and the significance tests refer to the mean per-episode collision count (the quantity tested), while Coll./task is the corresponding workload-normalized rate, which changes by a similar amount; p is the one-sided Mann–Whitney U -test against no sharing, p_{Holm} applies a Holm–Bonferroni correction across the four ablation comparisons, and the Significant column is judged after correction. On Map1 at five robots, the mean per-episode collision count falls from 367 ± 64 (no sharing) to 333 ± 53 under selective access ($q_{\text{share}} = 0.6$) and 339 ± 58 under full access: selective sharing gives a modest, nominally significant 9% reduction in collision count (Mann–Whitney U on these per-episode counts, one-sided $p = 0.024$; permutation $p = 0.014$), and full sharing an 8% reduction ($p = 0.047$; permutation $p = 0.041$); the corresponding collisions-per-completed-task rate falls from 1.45 to 1.34 (selective) and 1.38 (full), with the slightly smaller relative change reflecting the near-constant task count. The sharing analysis comprises four comparisons (two maps $\times$ two participation levels), which we treat as a single family and report both nominally and with a Holm correction (Table 10). Both Map1 comparisons are nominally significant, but neither survives the family-wise correction (Holm-adjusted $p = 0.097$ and $p = 0.14$); therefore, we read the sharing benefit as limited and layout-dependent rather than robustly established. A simulation-based post hoc power analysis places the power to detect the observed Map1 effects at $n = 30$ seeds per arm at 0.62 (selective) and 0.53 (full), and the minimal collision reduction detectable with 0.80 power at this sample size is about 15% of the no-sharing mean, so the effects of roughly

the observed magnitude sit near the detection threshold and smaller ones could escape it. We find no evidence that full sharing *increases* collisions relative to selective access (one-sided $p = 0.40$ in that direction), and no monotonic dose–response: the access-fraction sweep is essentially flat (Figure 10; Spearman $\rho = -0.06$, $p = 0.44$). On Map2, the effect is smaller and not significant (full 5%, $p = 0.16$; selective 2%, $p = 0.43$). The framework’s three-step look-ahead and environment-matched routes already keep collisions manageable, leaving little persistent congestion for the shared map to act on—so the near-halving reported from the originating thesis’s single preliminary run is not reproduced here. Crucially, this small benefit is *undetectable at low task throughput* (Figure 11): because the shared map H fills only as tasks complete (panel A), short episodes leave it nearly empty, and the collision-rate gap between full and no sharing is statistically indistinguishable from zero (panels B,C). In this regime, we cannot separate two compatible explanations—the mechanism has little accumulated history to act on, and few completed tasks give the test little power to detect such a small effect—so we report only that sharing has *no measurable effect* below roughly 1000 scheduler steps (the horizon at which the bootstrap interval first excludes zero in Figure 11), without claiming that the underlying effect is exactly zero.

Table 10. Collision-history sharing on the native route library of each map (five robots, 30 seeds, 2000 steps). One-sided permutation tests (p_{perm} , 10,000 resamples) are reported alongside the Mann–Whitney U -tests. Both Map1 conditions are nominally significant before correction, and neither survives Holm correction.

Map	q_{share}	Collisions	Coll./Task	Tasks	Change	p	p_{perm}	p_{Holm}	Significant
Map1	0.0	367 ± 64	1.45	254	—	—	—	—	—
	0.6	333 ± 53	1.34	249	−9.3%	0.024	0.014	0.097	no
	1.0	339 ± 58	1.38	245	−7.7%	0.047	0.041	0.140	no
Map2	0.0	380 ± 62	1.64	233	—	—	—	—	—
	0.6	374 ± 49	1.62	231	−1.6%	0.432	0.343	0.432	no
	1.0	362 ± 62	1.56	233	−4.7%	0.156	0.130	0.311	no

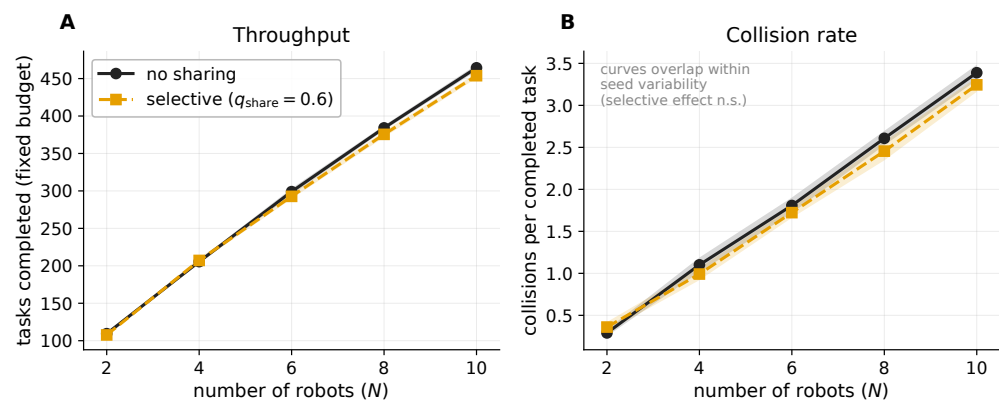


Figure 9. Collision-history sharing at realistic task throughput (Map1, 30 seeds): throughput (A) and collisions per completed task (B) versus fleet size, with and without selective sharing ($q_{\text{share}} = 0.6$); the shaded band around each curve is the percentile bootstrap 95% confidence interval across the 30 seeds. Selective sharing preserves throughput and produces only small changes in collision rate; the five-robot ablation (Table 10) shows that this selective-access effect does not survive multiple-comparison correction.

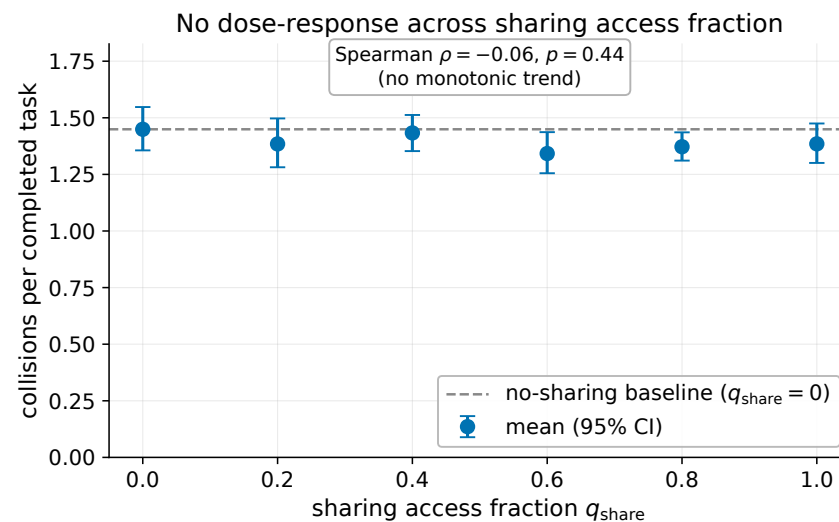


Figure 10. Access-fraction sweep (Map1, five robots, 30 seeds): collisions per completed task versus the fraction q_{share} of robots with collision-history access. Points are means with percentile bootstrap 95% confidence intervals (not connected, to avoid implying a dose-response); the dashed line is the no-sharing baseline. The sweep is essentially flat (Spearman $\rho = -0.06$, $p = 0.44$): there is no monotonic dose-response.

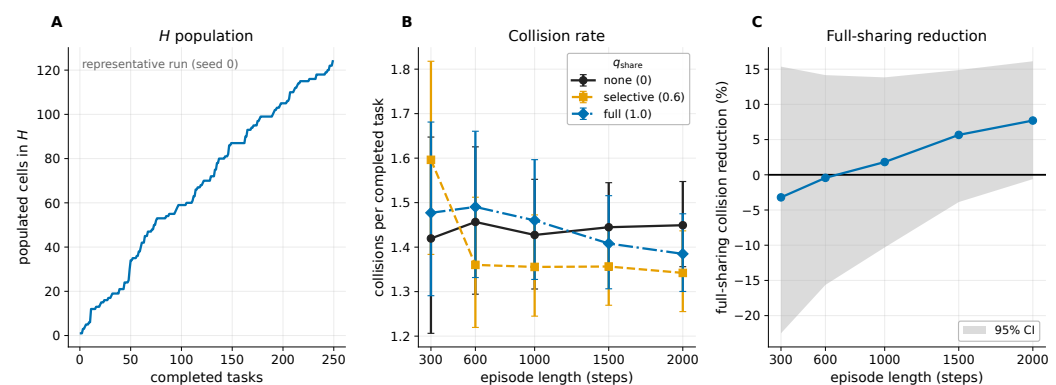


Figure 11. Throughput dependence of collision-history sharing (Map1, five robots). (A) The shared collision map H populates only as tasks complete (a representative run, seed 0). (B) Collisions per completed task for the three sharing levels ($q_{\text{share}} = 0, 0.6, 1.0$) across episode lengths; error bars are bootstrap 95% CIs. (C) Full-sharing collision reduction versus episode length, with a bootstrap 95% CI band (gray) about the zero reference line: the difference is statistically indistinguishable from zero at short horizons (when H is nearly empty) and becomes measurable only once enough completed tasks have populated H .

5. Discussion

5.1. What Governs Navigation Quality

The dominant factor in this decentralized framework is the alignment between the route library and the deployment environment. Because the BC policy imitates expert routes, its behavior is only as good as the routes it can draw on: a library generated for a different map carries that map's assumptions—including paths through cells that are obstacles in the new environment—and the robots inherit those mistakes. Adapting the library to the deployment map restores performance, transferring the learned navigation skill with the planner and policy unchanged. This is a concrete, measurable instance of the covariate-shift limitation of imitation learning [7], and of its remedy by distribution alignment.

This sensitivity to the gap between training and deployment conditions is not specific to our setup. Learning-based navigation is known to be sensitive to distribution drift

between the conditions a policy is trained on and those it meets at deployment [32], and methods that learn purely from demonstration similarly falter once a robot must act in cluttered configurations outside the demonstrated range [33]. Our route-library result is the grid-navigation instance of the same phenomenon: the policy is only as good as the offline routes it imitates, so realigning those routes to the deployment map is what restores performance. A complementary route to robustness, pursued mainly in manipulation, is to distill a large corpus of sampling-planner solutions into a single reactive policy that generalizes across scene layouts [34]; our results indicate that, for a fixed policy, the cheaper lever of regenerating the route library on the target map is already decisive.

Behavior Cloning is attractive here precisely because it is the simplest form of imitation learning—a supervised mapping from observations to actions that needs neither a reward function nor a dynamics model [35,36]. That simplicity is also its weakness: with no mechanism to correct for states unseen in the demonstrations, environment-specific assumptions are baked into the policy, which is why demonstration- or planner-derived priors are frequently paired with additional components to improve generalization [37]. Our hybrid retains BC for fast reactive control and delegates only the harder, environment-specific reconnection to the planner, which localizes the covariate-shift problem to the route library.

The BC network itself is trained once, offline, and remains *frozen* at deployment: no weight is updated during or between episodes. The framework’s response to dynamically changing congestion therefore comes not from the policy but from the machinery around it: the runtime route filter rejects routes blocked in the robot’s local occupancy window at the moment of selection, the probabilistic online-RRT reconnection replans around persistent blockages, and, when the optional add-on is enabled, the shared collision-history map biases route choice away from cells where congestion has repeatedly occurred. This division of labor is deliberate, as it keeps the learned component simple and auditable, but it also means the policy cannot specialize to congestion patterns it never saw during training; online adaptation of the policy parameters themselves, for example through adaptive-critic learning schemes [38], is an orthogonal extension outside the scope of this study. The scripted-connector ablation (Section 4.2) reinforces this reading from the other side: on these grids, a deterministic greedy controller with the same interface performs at least as well as the trained network, so the learned component is the replaceable part of the pipeline and the route library is the load-bearing one. We retain the BC connector as the faithful re-implementation target, but a deployment on comparable grid worlds could simplify it away; whether learned connectors pay off in richer dynamics, where greedy unit steps are no longer near-optimal, is a question this benchmark cannot answer.

5.2. The Role of Collision-History Sharing

Collision-history sharing is a low-cost way to steer robots away from cells that have repeatedly caused collisions, requiring no direct inter-robot messaging. Evaluated within the full hybrid framework and at a realistic task throughput, its benefit is limited: sharing yields small, nominally significant reductions on Map1 (9% under selective and 8% under full participation, neither surviving Holm correction across the sharing comparisons), while a second layout shows no significant change, and there is no clear dose–response. The mechanism is most useful where the base navigation leaves persistent congestion; here, the framework’s three-step look-ahead and environment-matched routes already keep collisions manageable, so little remains for the shared map to remove—and the near-halving reported from the originating thesis’s single preliminary run does not reproduce over repeated, seeded trials. The effect also depends on the shared map having accumulated enough history: at low throughput, the database stays nearly empty, and no

effect is detectable, so reporting this throughput dependence is essential to evaluating the mechanism fairly.

Collision-history sharing is an *indirect*, stigmergy-like form of coordination: robots influence one another only through a shared spatial record of where collisions recur, never through direct messages. This contrasts with the bulk of decentralized multi-robot navigation, where coordination is achieved through explicit communication—for example, decentralized multi-agent RRT in which agents pass a planning token and exchange planned trajectories to guarantee mutual collision avoidance [39]—and with classical reactive schemes that must resolve static-obstacle and inter-robot collisions at once and are prone to local-minima deadlock [40]. The appeal of a message-free shared signal is its near-zero coordination cost; our results temper that appeal by showing that, once the base navigation is strong, the residual congestion that such a signal can remove is small.

5.3. Generality and Relation to Planning–Learning Hybrids

The framework is within a broad and active effort to combine classical planning with learning, spanning sampling-based planners, supervised policies, and reinforcement learning [41]. Sampling-based planners such as RRT remain a workhorse for mobile robot path planning, and their cost is dominated by collision checking [42] and by parameters such as step size and iteration budget [43]; pairing them with learned components is increasingly common in industrial mobile robotics, for example, combining an RRT global planner with learned perception inside an ROS navigation stack [44]. Other hybrids augment a reinforcement-learning agent’s action space with a motion planner [45], or pair a demonstration-learned task model with a sampling-based planner so that a robot reproduces demonstrated behavior while still avoiding obstacles absent from the demonstrations [46]. Against this backdrop, our contribution is deliberately narrow: we reuse the original planner and policy verbatim and isolate the effect of route-library provenance, which is what lets the adaptation effect be attributed cleanly rather than confounded with architectural changes.

It is also worth delineating where the discrete-grid framework studied here stops and continuous-space methods with formal guarantees begin. Our robots move on a grid with an idealized kinematic model, so safety is enforced combinatorially (collision-checked routes and look-ahead) rather than analytically. In continuous state spaces, safe multi-robot coordination is instead certified through control-theoretic machinery, most prominently control-barrier functions [47], including safety-barrier certificates for collision-free multi-robot systems [48]. Likewise, when convergence *time* must be guaranteed, rather than measured, fixed-time stabilization theory [49] and adaptive-critic learning control [38] provide prescribed-performance alternatives. These families address controller-level guarantees for continuous dynamics that our discrete framework neither needs nor provides; conversely, the route-library provenance question we quantify arises whenever plans are reused across environments, regardless of the underlying controller. Extending the present framework toward continuous spaces or higher dynamicity would make those tools the natural complement, with the route library supplying environment-specific priors and the certified controller supplying safety and convergence guarantees.

5.4. Limitations

The study is in grid-based simulation; real deployments add sensing noise, continuous dynamics, and safety constraints. Sensing, in particular, is idealized: each robot reads its own pose and a 5×5 local occupancy window directly from the simulator, with no sensor noise, no localization uncertainty, and no perception latency, so the reported numbers should be read as an upper bound on what a physical fleet with real sensing would

achieve, and how the route-selection and look-ahead mechanisms degrade under noisy pose estimates remains untested. Behavior-cloning policies, in particular, carry a well-documented sim-to-real gap, so a simulation-trained policy needs physical validation before its numbers can be expected to transfer [50]. Absolute counts depend on modeling choices (the collision metric counts predicted blocked-move events; the shared map accumulates without decay; the access subset is fixed per episode), so the robust claims are the large route-library-adaptation effects and their directional consistency across seeds—not the precise magnitudes, and not the marginal collision-history-sharing effect, which does not survive multiple-comparison correction. The evaluation now spans three layouts, block geometry (Map1, Map2) and a corridor/bottleneck structure (Map3), but three hand-authored 50×50 grids remain a limited sample of industrial floor plans, so claims about warehouse environments in general should be read with that scope in mind.

5.5. Future Work

Several directions follow directly from these limitations. The environment-adapted route library here was produced offline, and real warehouse layouts drift over time; because offline plans are computed once and grow stale as the scene changes [51], generating such adaptation online and at scale is a natural next step, for example, by triggering incremental library repair (Section 3.4) from an online replanning behavior tree [51], or by handling moving obstacles through sequential neural control-barrier certificates [32]. Further directions include comparing against optimal multi-robot solvers (e.g., conflict-based search [11], whose suboptimal variants [12] extend the prioritized baseline added here), deep-reinforcement-learning navigation baselines (excluded here because training a new policy would confound the isolated route-library question), and continuous-space collision-avoidance methods (e.g., ORCA [52]), adding a decaying shared collision-history map, and validating the framework on physical robots.

6. Conclusions

We presented a faithful, fully reproducible re-implementation of a hybrid RRT + Behavior Cloning framework for decentralized multi-robot navigation in grid-based industrial environments, reusing the original trained network, planner and route data, and a quantitative analysis of environment-specific route-library adaptation within it. Our central finding is that navigation quality is governed by the match between the expert route library and the deployment environment: regenerating the library on the deployment map cuts collisions by 38–85% and task failures by 43–70% relative to a Map2-naïve library across five fleet sizes (2, 4, 6, 8, 10 robots) over 30 seeds, with Mann–Whitney and permutation tests in agreement, and the effect replicates on a third, structurally different corridor layout (65–96% collision reductions on Map3). The baselines added in this revision bound the effect: offline filtering, repair, or regeneration of invalid routes recovers most of the gain; a scripted connector in place of the trained BC network performs at least as well, locating the environment-specific knowledge in the route library rather than in the learned connector; prioritized planning over a shared reservation table bounds from above what centralized information achieves; and capping the online RRT baseline’s planning budget preserves its per-task collision quality at far lower runtime while sharply raising its task-failure rate, so the reuse-versus-replan decision trades planner usage against task reliability. A collision-history-sharing add-on that coordinates through a shared map rather than direct messaging, evaluated across repeated seeded trials, gives only a limited benefit—nominally significant 8–9% collision reductions on Map1 (not surviving multiple-comparison correction), but no significant change on a second layout—so the near-halving suggested by the originating thesis’s preliminary single run does not reproduce within the full framework,

and the effect is undetectable at low task throughput. By making the deterministic code and raw results available in the public repository—and by documenting the throughput and route-filtering conditions under which each effect appears—we aim to make these results directly reproducible and the framework a reliable basis for further work.

Author Contributions: Conceptualization, C.G.; methodology, Y.A. and C.G.; software, Y.A. and C.G.; validation, C.G.; formal analysis and investigation, Y.A. and C.G.; writing—original draft preparation, Y.A. and C.G.; writing—review and editing, C.G.; supervision and project administration, C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code, data, and trained model supporting the reported results are openly available in the GitHub repository at <https://github.com/ChenGiladi/skill-transfer-multi-robot-nav> (accessed on 19 July 2026). This repository also contains the artifacts reused from the originating thesis—the trained Behavior Cloning network weights, the RRT and route-generation code, and the route datasets—so the re-implementation can be verified independently of the thesis document itself.

Acknowledgments: During the preparation of this manuscript, the authors used Writefull and Claude Opus 4.8 (Anthropic) for the purposes of language editing and condensing portions of the text. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

References

1. Hekmati, A.; Gupta, K. On Safe Robot Navigation Among Humans as Dynamic Obstacles in Unknown Indoor Environments. In *Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*; IEEE: Kuala Lumpur, Malaysia, 2018; pp. 1082–1087. [\[CrossRef\]](#)
2. Silva, S.; Verdezoto, N.; Paillacho, D.; Millan-Norman, S.; Hernández, J.D. Online Social Robot Navigation in Indoor, Large and Crowded Environments. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: London, UK, 2023; pp. 9749–9756. [\[CrossRef\]](#)
3. Han, R.; Chen, S.; Hao, Q. Cooperative Multi-Robot Navigation in Dynamic Environment with Deep Reinforcement Learning. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: Paris, France, 2020; pp. 448–454. [\[CrossRef\]](#)
4. Xu, P.; Karamouzas, I. Human-Inspired Multi-Agent Navigation Using Knowledge Distillation. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: New York, NY, USA, 2021; pp. 8105–8112. [\[CrossRef\]](#)
5. Wu, S.; Chen, G.; Shi, M.; Alonso-Mora, J. Decentralized Multi-Agent Trajectory Planning in Dynamic Environments with Spatiotemporal Occupancy Grid Maps. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2024. [\[CrossRef\]](#)
6. Karaman, S.; Frazzoli, E. Incremental Sampling-Based Algorithms for Optimal Motion Planning. *arXiv* **2010**, arXiv:1005.0416. [\[CrossRef\]](#)
7. Codevilla, F.; Santana, E.; López, A.M.; Gaidon, A. Exploring the Limitations of Behavior Cloning for Autonomous Driving. *arXiv* **2019**, arXiv:1904.08980. [\[CrossRef\]](#)
8. Bojarski, M.; Testa, D.D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to End Learning for Self-Driving Cars. *arXiv* **2016**, arXiv:1604.07316. [\[CrossRef\]](#)
9. Erdmann, M.; Lozano-Pérez, T. On multiple moving objects. *Algorithmica* **1987**, *2*, 477–521. [\[CrossRef\]](#)
10. Silver, D. Cooperative Pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*; AAAI Press: Washington, DC, USA, 2005; Volume 1, pp. 117–122. [\[CrossRef\]](#)
11. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [\[CrossRef\]](#)
12. Barer, M.; Sharon, G.; Stern, R.; Felner, A. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS)*; AAAI Press: Washington, DC, USA, 2014; Volume 5, pp. 19–27. [\[CrossRef\]](#)

13. Zhang, R.; Hou, J.; Walter, F.; Gu, S.; Guan, J.; Röhrbein, F.; Du, Y.; Cai, P.; Chen, G.; Knoll, A. Multi-Agent Reinforcement Learning for Autonomous Driving: A Survey. *arXiv* **2024**, arXiv:2408.09675. [[CrossRef](#)]
14. Gao, Z.; Yang, G.; Prorok, A. Online Control Barrier Functions for Decentralized Multi-Agent Navigation. In *Proceedings of the 2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*; IEEE: Boston, MA, USA, 2023; pp. 107–113. [[CrossRef](#)]
15. Mestres, P.; Nieto-Granda, C.; Cortés, J. Distributed Safe Navigation of Multi-Agent Systems Using Control Barrier Function-Based Optimal Controllers. *arXiv* **2024**, arXiv:2402.06195. [[CrossRef](#)]
16. Kondo, K.; Tewari, C.T.; Peterson, M.B.; Thomas, A.; Kinnari, J.; Tagliabue, A.; How, J.P. PUMA: Fully Decentralized Uncertainty-Aware Multiagent Trajectory Planner with Real-Time Image Segmentation-Based Frame Alignment. *arXiv* **2024**, arXiv:2311.03655. [[CrossRef](#)]
17. Tang, J.; Duan, H.; Lao, S. Swarm Intelligence Algorithms for Multiple Unmanned Aerial Vehicles Collaboration: A Comprehensive Review. *Artif. Intell. Rev.* **2023**, *56*, 4295–4327. [[CrossRef](#)]
18. Li, J.; Wang, K.; Chen, Z.; Wang, J. An Improved RRT* Path Planning Algorithm in Dynamic Environment. In *Methods and Applications for Modeling and Simulation of Complex Systems*; Fan, W., Zhang, L., Li, N., Song, X., Eds.; Communications in Computer and Information Science; Springer Nature: Singapore, 2022; Volume 1713, pp. 301–313. [[CrossRef](#)]
19. Zhao, P.; Chang, Y.; Wu, W.; Luo, H.; Zhou, Z.; Qiao, Y.; Li, Y.; Zhao, C.; Huang, Z.; Liu, B.; et al. Dynamic RRT: Fast Feasible Path Planning in Randomly Distributed Obstacle Environments. *J. Intell. Robot. Syst.* **2023**, *107*, 48. [[CrossRef](#)]
20. Da Silva Costa, L.; Tonidandel, F. DVG+A* and RRT Path-Planners: A Comparison in a Highly Dynamic Environment. *J. Intell. Robot. Syst.* **2021**, *101*, 58. [[CrossRef](#)]
21. Florence, P.; Lynch, C.; Zeng, A.; Ramirez, O.; Wahid, A.; Downs, L.; Wong, A.; Lee, J.; Mordatch, I.; Tompson, J. Implicit Behavioral Cloning. *arXiv* **2021**, arXiv:2109.00137. [[CrossRef](#)]
22. Chi, Z.; Zhu, L.; Zhou, F.; Zhuang, C. A Collision-Free Path Planning Method Using Direct Behavior Cloning. In *Intelligent Robotics and Applications*; Yu, H., Liu, J., Liu, L., Ju, Z., Liu, Y., Zhou, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 529–540. [[CrossRef](#)]
23. Samak, T.V.; Samak, C.V.; Kandhasamy, S. Robust Behavioral Cloning for Autonomous Vehicles Using End-to-End Imitation Learning. *SAE Int. J. Connect. Autom. Veh.* **2021**, *4*, 279–295. [[CrossRef](#)]
24. Farag, W.; Saleh, Z. Behavior Cloning for Autonomous Driving Using Convolutional Neural Networks. In *Proceedings of the 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*; IEEE: Sakhier, Bahrain, 2018; pp. 1–7. [[CrossRef](#)]
25. Ran, L.; Zhang, Y.; Zhang, Q.; Yang, T. Convolutional Neural Network-Based Robot Navigation Using Uncalibrated Spherical Images. *Sensors* **2017**, *17*, 1341. [[CrossRef](#)] [[PubMed](#)]
26. Pan, Z.; Manocha, D. Feedback Motion Planning for Liquid Pouring Using Supervised Learning. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: Vancouver, BC, Canada, 2017; pp. 1252–1259. [[CrossRef](#)]
27. Jia, B.; Manocha, D. Sim-to-Real Robotic Sketching Using Behavior Cloning and Reinforcement Learning. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: Yokohama, Japan, 2024; pp. 18272–18278. [[CrossRef](#)]
28. Dharmavaram, A.; Gupta, T.; Li, J.; Sycara, K.P. SS-MAIL: Self-Supervised Multi-Agent Imitation Learning. *arXiv* **2021**, arXiv:2110.08963. [[CrossRef](#)]
29. Fang, B.; Zheng, C.; Wang, H. Fact-Based Agent Modeling for Multi-Agent Reinforcement Learning. *arXiv* **2023**, arXiv:2310.12290. [[CrossRef](#)]
30. Strouse, D.J.; McKee, K.R.; Botvinick, M.; Hughes, E.; Everett, R. Collaborating with Humans without Human Data. *arXiv* **2021**, arXiv:2110.08176. [[CrossRef](#)]
31. Hu, H.; Wu, D.J.; Lerer, A.; Foerster, J.; Brown, N. Human-AI Coordination via Human-Regularized Search and Learning. *arXiv* **2022**, arXiv:2210.05125. [[CrossRef](#)]
32. Yu, H.; Hirayama, C.; Yu, C.; Herbert, S.; Gao, S. Sequential Neural Barriers for Scalable Dynamic Obstacle Avoidance. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: Detroit, MI, USA, 2023; pp. 11241–11248. [[CrossRef](#)]
33. Ye, G.; Alterovitz, R. Demonstration-Guided Motion Planning. In *Robotics Research*; Christensen, H.I., Khatib, O., Eds.; Springer Tracts in Advanced Robotics; Springer International Publishing: Cham, Switzerland, 2017; Volume 100, pp. 291–307. [[CrossRef](#)]
34. Dalal, M.; Yang, J.; Mendonca, R.; Khaky, Y.; Salakhutdinov, R.; Pathak, D. Neural MP: A Generalist Neural Motion Planner. *arXiv* **2024**, arXiv:2409.05864.
35. Morga-Bonilla, S.I.; Rivas-Camero, I.; Torres-Jiménez, J.; Téllez-Cuevas, P.; Núñez-Cruz, R.S.; Perez-Arista, O.V. Behavioral Cloning Strategies in Steering Angle Prediction: Applications in Mobile Robotics and Autonomous Driving. *World Electr. Veh. J.* **2024**, *15*, 486. [[CrossRef](#)]
36. Zhan, E.; Zheng, S.; Yue, Y.; Lucey, P. Generative Multi-Agent Behavioral Cloning. *arXiv* **2018**, arXiv:1803.07612.

37. Zhang, Z.; Hong, J.; Enayati, A.S.; Najjaran, H. Using Implicit Behavior Cloning and Dynamic Movement Primitive to Facilitate Reinforcement Learning for Robot Motion Planning. *arXiv* **2024**, arXiv:2307.16062. [[CrossRef](#)]
38. Li, S.; Shao, X.; Wang, H.; Liu, J.; Zhang, Q. Adaptive Critic Attitude Learning Control for Hypersonic Morphing Vehicles Without Backstepping. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 8787–8803. [[CrossRef](#)]
39. Desaraju, V.R.; How, J.P. Decentralized Path Planning for Multi-Agent Teams in Complex Environments Using Rapidly-Exploring Random Trees. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*; IEEE: Shanghai, China, 2011; pp. 4956–4961. [[CrossRef](#)]
40. Zhaofeng, Y.; Ruizhe, Z. Path Planning of Multi-robot Cooperation for Avoiding Obstacle Based on Improved Artificial Potential Field Method. *Sens. Transducers* **2014**, *165*, 221.
41. Zhou, C.; Huang, B.; Fränti, P. A Review of Motion Planning Algorithms for Intelligent Robots. *J. Intell. Manuf.* **2022**, *33*, 387–424.
42. Yu, C.; Gao, S. Reducing Collision Checking for Sampling-Based Motion Planning Using Graph Neural Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4274–4289.
43. Heng, H.; Ghazali, M.H.M.; Rahiman, W. Comparative Analysis of Navigation Algorithms for Mobile Robot. *J. Ambient Intell. Humaniz. Comput.* **2024**, *15*, 3861–3871. [[CrossRef](#)]
44. Adiuku, N.; Avdelidis, N.P.; Tang, G.; Plastropoulos, A. Improved Hybrid Model for Obstacle Detection and Avoidance in Robot Operating System Framework (Rapidly Exploring Random Tree and Dynamic Windows Approach). *Sensors* **2024**, *24*, 2262. [[CrossRef](#)] [[PubMed](#)]
45. Yamada, J.; Lee, Y.; Salhotra, G.; Pertsch, K.; Pflueger, M.; Sukhatme, G.S.; Lim, J.J.; Englert, P. Motion Planner Augmented Reinforcement Learning for Robot Manipulation in Obstructed Environments. *arXiv* **2020**, arXiv:2010.11940.
46. Bowen, C.; Alterovitz, R. Closed-Loop Global Motion Planning for Reactive, Collision-Free Execution of Learned Tasks. *ACM Trans. Hum.-Robot Interact.* **2018**, *7*, 1–16. [[CrossRef](#)]
47. Ames, A.D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; Tabuada, P. Control Barrier Functions: Theory and Applications. In *Proceedings of the 2019 18th European Control Conference (ECC)*; IEEE: New York, NY, USA, 2019; pp. 3420–3431. [[CrossRef](#)]
48. Wang, L.; Ames, A.D.; Egerstedt, M. Safety Barrier Certificates for Collisions-Free Multirobot Systems. *IEEE Trans. Robot.* **2017**, *33*, 661–674. [[CrossRef](#)]
49. Polyakov, A. Nonlinear Feedback Design for Fixed-Time Stabilization of Linear Control Systems. *IEEE Trans. Autom. Control* **2012**, *57*, 2106–2110. [[CrossRef](#)]
50. Verma, A.; Bagkar, S.; Allam, N.V.S.; Raman, A.; Schmid, M.; Krovi, V.N. Implementation and Validation of Behavior Cloning Using Scaled Vehicles. In *Proceedings of the SAE WCX Digital Summit*, Online, 13 April 2021. [[CrossRef](#)]
51. De Luca, A.; Muratore, L.; Tsagarakis, N.G. Autonomous Navigation with Online Replanning and Recovery Behaviors for Wheeled-Legged Robots Using Behavior Trees. *IEEE Robot. Autom. Lett.* **2023**, *8*, 6803–6810. [[CrossRef](#)]
52. van den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n -Body Collision Avoidance. In *Robotics Research*; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–19. [[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.