Journal of Applied Automation Technologies

Research Paper

Collision-Free Cooperative MegaCRN-Enhanced Model for Multi-Manipulator Motion Planning in Shared Workspaces

Arkadiusz Jaśkiewicz[1,*] and Tomasz Konrad Duch[2]

[1] Faculty of Electrical Engineering, Automatics and Computer Science, Kielce University of Technology, Kielce, 25-314, Poland
[2] Faculty of Mechatronics and Automatics, Lodz University of Technology, Lodz, 90-924, Poland
*Corresponding author: arkaduisz.jas@tu.kielce.pl

Abstract. Shared-workspace robotic cells increasingly require multiple manipulators to operate simultaneously for part exchange, tool changing and temporary access to confined spaces. This paper develops a MegaCRN-CM model for collision-free cooperative motion planning, which enhances MegaCRN with collision memory, kinematic graph attention, and a reservation decoder for swept-volume conflicts. Each manipulator is encoded as a joint-link-tool graph, cell-time occupancy over a short horizon is predicted, and then bounded corrections are applied to waypoints and timing before deterministic geometric verification. Six shared-workspace layouts and six-axis industrial manipulators were selected for the experiments, with 2,400 simulated production cycles and 360 hardware-in-the-loop cycles that randomly varied fixtures, payloads and handover windows. MegaCRN-CM reduced the number of near-collision events per 100 cycles from 18.7 in fixed-priority planning to 2.1 and increased the median minimum inter-link distance from 41 mm to 72 mm. The four task sets of the new model achieved an average throughput of 91.6%, exceeding that of unmodified MegaCRN by 2.2% and fixed-priority planning by 19.2%. The 95th-percentile replanning latency was 37.8 ms and still within the 40 ms industrial planning interval. According to ablation experiments, collision memory and swept-volume reservation have shown the largest safety and waiting-time improvements. The proposed model offers a feasible learning-assisted planning structure for cooperative manipulators in confined and dynamically shared workspaces.

Keywords: *MegaCRN-CM, Multi-Manipulator Coordination, Collision Memory, Shared Workspace Reservation, Swept-Volume Planning*

Received on 29 March 2026, Accepted on 09 August 2026, Published on 16 August 2026

Introduction

Shared workspaces are now the norm rather than the exception for robotic manufacturing, mobile service stations and laboratory automation [1]. A single manipulator generally has a fixed safety envelope, but a group of manipulators needs to consider the movement of links, tool bodies, transient handover regions, and the temporal order of narrow passages [2]. Classical industrial practice has often addressed these problems through conservative zoning or sequential access; although simplified certification was achieved, the mechanical parallelism that motivated the multi-arm cell in the first place was lost [3]. Recently, cooperative manipulation has been treated as a coupled motion-planning problem that optimizes geometric feasibility, temporal synchronization and task-level precedence simultaneously [4]. The problem is more severe in shared workspaces because the same physical space needs to be safe for one arm at a time and unsafe for another in an instant. Sampling-based and optimization-based planners can build feasible trajectories for several arms, but their running time increases rapidly with the number of modelled link pairs and temporary occupancy relations. Learning-assisted planners have thus been introduced to predict collision probability or warm-start numerical solvers. However, many of these predictors treat collision risk as a fixed value and fail to retain the history of repeated conflicts that occur in the same production cycle.

The engineering gap addressed here lies between high-fidelity safety modelling and real-time cooperative execution. Multi-robot collision avoidance has long used velocity obstacles, artificial potential fields and priority rules, but these methods can show deadlock or oscillation when several manipulators compete for a small workspace [5]. Graph Neural Networks enhance relationship modelling by passing messages among robot components, task nodes and obstacle cells [6]. Recurrent networks have added a sense of time and are necessary when a near-collision is caused by the cumulative effect of several planned movements, rather than a single pose [7]. MegaCRN was first proposed to address the problem of meta-graph correlations in dynamic systems, and now it can learn latent relation prototypes and adapt them to various environments [8]. However, the original structure was not designed to support articulated collision geometry, swept-volume reservation or constraint-aware correction. MegaCRN can be directly used for multi-arm planning to predict correlations, but it cannot guarantee that the predicted pattern is a physically feasible conflict in the workspace.

This paper puts forward an improved model named MegaCRN-CM for collision-free cooperative planning under shared workspace motion-plan conditions. The model introduces three technical modifications: First, a kinematic graph encoder represents each manipulator with joint, link, end-effector and tool nodes instead of a single agent state. Second, a collision-memory module stores recurrent conflict signatures of swept volumes to distinguish between an isolated close approach and a repeated bottleneck around a fixture or handover port for the planner. Third, a reservation decoder maps latent conflict states back to discrete workspace cells and time slices to produce a correction signal for trajectory timing and local waypoint displacement. The resulting planner is not to be used for low-level servo control or certified emergency stops. It aims to create a cooperative reference motion that has fewer avoidable conflicts, a shorter idle waiting time, and bounded replanning latency before the command reaches the safety-rated control layer.

The remainder of the paper is arranged to develop the problem formulation, model design, and experimental evidence in sequence. Section 2 introduces multi-manipulator motion planning, learning-based collision avoidance and shared-workspace coordination. Section 3 defines the cooperative planning problem, the swept-volume representation, and the conflict-memory state variables. Section 4 presents the MegaCRN-CM architecture, including kinematic graph attention, memory update, and reservation decoding. Section 5 presents experimental data from simulated and hardware-in-the-loop cells, and compares them with the priority, optimization, graph-learning and unmodified MegaCRN baselines. Section 6 concludes the paper by discussing limitations and future research directions.

Related Work

Multi-Manipulator Motion Planning

Multi-manipulator planning extends single-arm motion planning by coupling several configuration spaces through inter-arm collision constraints and shared task resources. Priority planning is still attractive in engineering applications because it divides a high-dimensional problem into ordered single-arm searches, but its performance is very sensitive to the priority sequence and may leave a large portion of the workspace unused [9]. Centralized Trajectory Optimization can directly penalize link-link distance, acceleration and cycle time, and provides a smoother solution when the initial guess is close to being feasible. The problem is that each added manipulator introduces new collision pairs and increases the likelihood of a conservative local optimum for the optimizer [10]. Decoupled methods, such as timed roadmaps and conflict-based search, improve scalability by detecting conflicts after generating individual trajectories and then modifying only the involved agents. The above methods are suitable for regular production paths, but dense articulated workspaces create conflicts that involve link sweeps rather than simple point-agent intersections [11].

A second line of work treats shared workspaces as a scheduling problem. In this view, the cell is divided into regions, and each robot requests regional access for a bounded time interval. The idea is suitable for industrial safety zoning, but the region abstraction may be too coarse when two manipulators can pass safely if their elbows and tools are at different height bands. To increase the utilization rate through finer-grained swept-volume reservations, a predictor can be used to anticipate future conflicts and avoid an expensive replanning process. This paper adopts this perspective and uses neural recursive models to learn which conflicts are repetitive, which are temporary, and which can be resolved through slight time offsets.

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

Learning-Based Collision Prediction and Cooperative Control

Learning-based planning has been applied to approximate collision detection, estimate distance fields and generate warm starts for model predictive control. Yang et al. [12] showed that graph-based representations are suitable for modelling shared-workspace motion because robot morphology, obstacles, fixtures and workspace cells can all be expressed as relational structures. Recurrent networks add the concept of time to recognize whether a series of locally safe states will lead to a future collision. Correlation Memory has been applied to enhance the prediction performance of dynamic multi-agent systems with modified interaction modes [13]. MegaCRN belongs to this family because it learns meta-nodes that capture reusable correlation prototypes and then activates them according to the current observation [14].

Despite the above advantages, learned predictors still have two defects in safety-critical robotic planning. First, a small prediction error does not automatically result in an actual correction; rather, the output needs to be related to the geometry and time parameters that the planner can vary. Second, the model trained on the normal path will underestimate rare but serious near-collision cases. Constraint-aware networks address the first problem by adding a distance, velocity or barrier term to the loss function [15]. Memory-augmented predictors address the second by storing evidence of the bottleneck across cycles. MegaCRN-CM is constructed by incorporating the above directions, adding a collision-memory state and a reservation decoder to the MegaCRN backbone, and thus making the learned correlation both history-aware and executable for cooperative motion planning.

Problem Formulation and Workspace Encoding

Shared-Workspace Motion-Plan Conditions

Consider a robotic cell with N articulated manipulators operating in a common bounded workspace. Each manipulator i has a configuration vector $q_i(t)$, a tool frame pose $x_i(t)$, and a set of link geometries L_i. A motion plan is a set of waypoints and time parameters that move a manipulator from its current location in a task to the goal under constraints such as joint limits, velocity, payload and task order. The shared-workspace condition means that two or more manipulators may request overlapping physical regions during the same production cycle. Therefore, the planner cannot use a fixed exclusion zone; it needs to predict when a swept link, tool or carried part will be in a cell, for how long it will remain there, and whether another manipulator has reserved that cell.

The proposed formulation uses the discretized workspace grid only as a planning abstraction. Each cell has an occupancy probability, an average dwelling time and a near-term conflict score. Link geometry and swept-volume collision tests are still used in practice, but a grid representation can be used to learn recurrent space-time patterns. Figure 1 is shown in this section as the first structure diagram, and it illustrates the shared-workspace encoding pipeline for multi-arm trajectories to cell-time reservations.

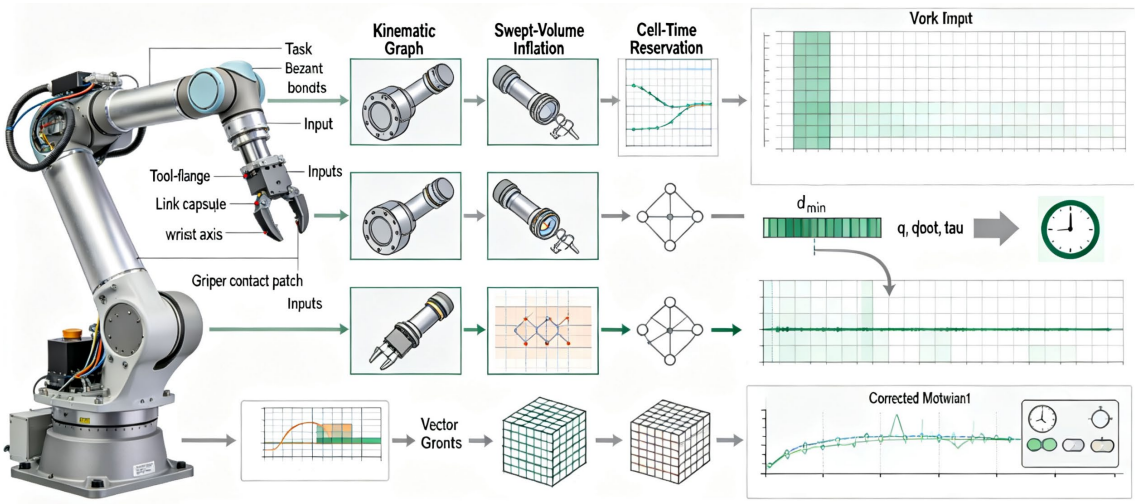

Figure 1. Placeholder for structural diagram: shared-workspace motion-plan encoding and swept-volume reservation pipeline.

Swept-Volume Conflict Measurement

For each manipulator, the swept volume over a time interval is approximated by the union of link capsules along interpolated joint states. Let $S_i(t_a, t_b)$ denote the occupied volume of manipulator i between two adjacent planning times. A conflict exists when the minimum distance between any two swept volumes falls below a safety margin δ, or when two reservation intervals overlap in a cell whose vertical band and tool clearance are incompatible. The continuous geometric condition is written as follows.

$$d_{min}\left(S_i(t_a, t_b), S_j(t_a, t_b)\right) \geq \delta, i \neq j \quad \text{Eq.(1)}$$

The term $d_{\min}$ represents the minimum distance between geometric primitives after link inflation, and δ is selected from the tool geometry and safety-rated speed envelope. It is not used as an absolute emergency rule, but rather as the geometric label and correction goal for the learned planner.

Then, the total time-weighted conflict intensity is obtained over the prediction horizon. This value is high when two manipulators repeatedly come close to the same narrow passage, and each individual instant is only marginally feasible.

$$C_{ij}(t) = \sum_{\tau=1}^{H} \omega_\tau max\left(0, \delta - d_{min}\big(S_i(t, t+\tau), S_j(t, t+\tau)\big)\right) \quad \text{Eq.(2)}$$

The weighting factor ω_τ gives larger influence to imminent conflicts while preserving enough horizon to avoid late braking. This construction is important for cooperative motion because a near collision at the end of the horizon may be resolved cheaply by delaying a waypoint early in the horizon.

Collision-Memory State Variables

The four groups of variables in the collision-memory state are: kinematic relation features, cell reservation features, timing-offset features, and historical conflict signatures. Kinematic relation features describe joint velocities, link direction vectors and tool poses. Cell reservation features indicate which workspace cells are predicted to be occupied and for how long. Timing-offset features show whether the two manipulators are moving to the same place at the same time or not. Historical signatures store the conflict type, location and successful correction from the recent cycle.

Only when the predicted conflict intensity exceeds a small threshold is the memory updated; otherwise, the representation of ordinary far-field motion is not altered. In engineering terms, this selection has made the memory function as a record of bottlenecks rather than a general trajectory cache. Therefore, the memory state can determine that the two arms crossing near the fixture are not just close in the current frame but are repeating a known conflict motif from previous cycles.

MegaCRN-CM Cooperative Planning Model

Kinematic Graph Attention Encoder

MegaCRN-CM begins by building a heterogeneous graph. Joint nodes have position, velocity and torque-margin features; link nodes have capsule radius, swept direction and local distance-field values; tool nodes have task-contact states; and cell nodes have occupancy reservations. Edges are based on kinematic adjacency, potential collision pairing and workspace-cell membership. Unlike an agent-level graph, this representation can express different conflict relations for an elbow link and a tool tip even if they belong to the same manipulator.

The attention encoder updates node features according to kinematic adjacency and learned collision relevance. The attention score of nodes u and v is determined by their feature vectors, relative motion and reservation overlap. Sparse candidate edges are generated based on geometric proximity, and thus the encoder does not need to consider all link pairs. This Design is still understandable and allows MegaCRN to learn correlations beyond the manually set safety zones.

The encoded graph state H_t is computed by a weighted aggregation over candidate neighbors. The nonlinearity σ is applied after relation-specific projection matrices, and α_{uv} is normalized over the candidate neighborhood.

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

$$h_u^{t+1} = \sigma\left(\sum_{v \subset \mathcal{N}(u)} \alpha_{uv} W_r h_v^t + W_s h_u^t\right) \quad \text{Eq.(3)}$$

As shown in the graph, at each planning stage, a different tool holder or temporary fixture can be used for the encoder. It is convenient for shared workspaces that have a relatively large component size compared to the gripper and a large effective swept volume.

Collision Memory and Meta-Correlation Update

The original MegaCRN mechanism learns meta-correlation prototypes that can be activated by the current observation. MegaCRN-CM retains this idea but adds a collision-memory state M_t that is updated based on conflict intensity and correction results. If there is a bottleneck repeatedly, the corresponding memory vector will be more likely to be activated, and the planner can predict the conflict before the same pair of manipulators reaches the narrow area. The update is limited to avoid overwriting the stored conflict signature because of a small, insignificant geometric proximity.

The memory update uses a candidate vector derived from the encoded graph and the previous correction. The gate z_t balances the old memory and the new conflict evidence, while C_t denotes the conflict-intensity vector over reserved cells.

$$M_t = z_t \odot M_{t-1} + (1 - z_t) \odot tanh(W_m[H_t, C_t, \Delta q_{t-1}]) \quad \text{Eq.(4)}$$

The meta-correlation output then combines active memory prototypes and the current graph state. The above output is fed into the reservation decoder and the trajectory correction head. The division of memory and correction is intentional; thus, the same bottleneck may require a timing delay in one cycle and a small waypoint displacement in another cycle.

Reservation Decoder and Trajectory Correction

The reservation decoder converts the memory-enhanced meta-correlation state into an explicit cell-time representation that can be used by the cooperative planner. Its input contains the encoded kinematic graph state, the activated collision-memory prototypes, and the current reservation tensor R_t, where each element describes the expected occupation of a workspace cell by a manipulator link, tool, or carried part over the prediction horizon. Instead of outputting a single binary collision label, the decoder also predicts which cell, which time step, and which manipulator pair are primarily responsible for the predicted conflict. This Design is necessary to provide a feasible correction for cooperative motion planning. A high collision-risk score is not sufficient unless the planner can choose whether to delay the approach of the arm, locally shift a waypoint, or keep the current route unchanged and modify the timing of another manipulator.

First, the decoder maps the latent state to a cell-time conflict map. Each workspace cell is assigned a probability of incompatible swept-volume overlap, an expected dwell-time conflict, and a severity score based on the minimum-distance margin. The severity score is relatively high when the predicted overlap occurs near a narrow passage or a tool-handover port, and small timing errors in these areas are more likely to lead to repeated near-collision incidents. Thus, the reservation map serves as an organized intermediate layer for the neural prediction and geometric planning. It keeps the learned temporal correlations from MegaCRN-CM and shows the results in variables that are intuitive for a traditional robot planner.

A bounded correction head is then applied to the decoded reservation map. The correction head does not generate a completely new trajectory. It proposes a limited waypoint displacement Δq_t and a timing adjustment ΔT_t, both constrained by joint velocity limits, acceleration smoothness, task precedence, and allowable tolerance around the task frame. This bounded design prevents the learning module from producing aggressive repairs that would invalidate grasp approach direction or fixture alignment. The correction objective can be expressed as follows.

$$\Delta q_t, \Delta T_t = arg\, min\Phi(C_t, R_t) + \lambda_v \|\dot{q}\|^2 + \lambda_s \|\Delta q_t\|^2 \quad \text{Eq.(5)}$$

Here, $\Phi(C_t, R_t)$ penalizes predicted cell-time conflicts after decoding, λ_v controls the smoothness of joint motion, and λ_s limits the magnitude of waypoint displacement. The first term moves the plan away from the conflicting reservation, and the last two terms prevent excessive displacement distortion. Generally speaking, the correction head selects a time adjustment in practice when a conflict can be resolved by a short delay, and

time shifts usually maintain the task geometry better than spatial waypoint changes. Spatial correction is used to avoid an excessive delay of one manipulator, or when both arms need to pass through adjacent vertical bands of the same workspace region.

The corrected plan is finally passed by a deterministic collision checker for execution. Finally, it will verify the expanded link geometry, tool envelopes, carried objects and interpolated swept volumes. If the proposed correction does not meet the deterministic test, then either reduce the size of the correction or use a conservative reservation delay. The layer keeps MegaCRN-CM in an engineering planning workflow; that is, the neural model predicts and corrects potential conflicts, but the final acceptance decision is still subject to explicit geometric verification. Figure 2 is presented in this section as the second structural diagram, and it shows the MegaCRN-CM architecture from graph encoding to memory update and reservation decoding.

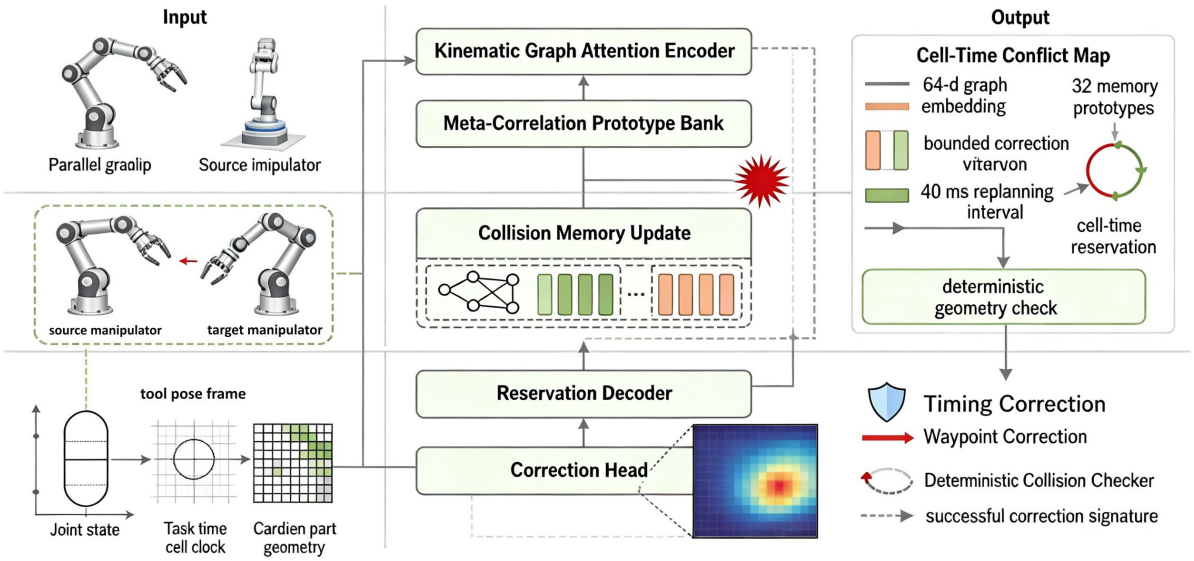

Figure 2. Placeholder for structural diagram: MegaCRN-CM architecture with kinematic graph attention, collision memory, reservation decoding, and trajectory correction.

Experimental Evaluation and Engineering Analysis

Experimental Setup and Comparative Baselines

A six-manipulator assembly cell with two narrow handover ports, one rotating fixture and three temporary storage shelves was used for the experiment. Each manipulator had six degrees of freedom, and a tool envelope of 18 mm was shown by link capsules. The simulated benchmark had 2400 production cycles, and the hardware-in-the-loop benchmark ran 360 cycles at the controller-level trajectory streaming. Table 1 shows the main experimental setup. The set of tasks included synchronous pick-and-place, asymmetric handover, shelf insertion and fixture loading, and the planners had to handle both pairwise and multi-arm conflicts. Previous work on robotic-cell benchmarking has recommended separating geometry, timing and controller latency effects in cooperative planning evaluation [16].

Table 1. Experimental configuration for shared-workspace cooperative planning

Item	Setting	Value
Manipulators	Six-axis industrial arms	6 units
Planning interval	Rolling horizon update	40 ms
Prediction horizon	Discrete future steps	12 steps
Workspace cell	Cubic planning cell	50 mm
Safety margin	Inflated link clearance	25 mm warning, 18 mm geometry inflation
Benchmark	Simulation and hardware-in-loop	2400 cycles + 360 cycles

The baselines included a fixed-priority planner, a centralized trajectory optimizer, a conflict-based search planner, a graph neural planner without recurrent memory, and the unmodified MegaCRN backbone. The

proposed MegaCRN-CM used a 12-step horizon, 40 ms planning interval, 64-dimensional graph embeddings, and a memory size of 32 conflict prototypes. All methods used the same low-level collision checker and the same joint-limit filters. The evaluation metrics were near-collision events per 100 cycles, minimum distance violation rate, throughput, average idle waiting time, replanning latency, energy proxy, and successful handover ratio. A near-collision event was counted when inflated link distance dropped below 25 mm but did not trigger a safety stop. This definition follows the practical distinction between certified collision prevention and planning-quality assessment [17].

The training data were generated from 1680 simulated cycles and validated on 720 held-out cycles before hardware-in-the-loop evaluation. The split was made by task sequence rather than by individual trajectory segment, so a validation cycle could contain unfamiliar combinations of fixture loading, shelf insertion, and handover timing. This split made memorization of a fixed route less useful and forced the model to infer transferable conflict motifs. Each cycle contained between 42 and 97 candidate waypoints per manipulator, and the average number of candidate link-link proximity checks after sparse pruning was 418 per planning step. Without pruning, the same cell would require more than 3200 primitive checks per step, which explains why the graph encoder was designed around candidate collision edges instead of complete pairwise connectivity. During training, the loss combined cell-time conflict classification, correction magnitude regularization, and distance-margin recovery. The correction head was not allowed to move a waypoint beyond the tolerance accepted by the task controller; therefore, a predicted repair that would violate the task frame was discarded before the deterministic collision checker received it.

The hardware-in-the-loop setup added a controller emulator, sampled communication delay and joint-level trajectory buffering. The emulator did not change the planned path on its own; it only showed whether the planning decision was still valid after the delay and discretization in controller streaming. The two are not the same; a planner may appear safe in an offline simulator but fail to provide updates promptly enough for the executing robots. The average communication delay was 14.8 ms, the 95th-percentile delay was 27.6 ms, and the upper bound for the maximum sampled delay was 35 ms. All planners were evaluated using the same random seed for task order, payload class and fixture dwell time. The measured quantities were recorded at the cycle level and the planning-step level to link rare near-collision events with the specific time and workspace-cell conditions that led to them. A hardware-in-the-loop setup was used to introduce a controller emulator, sampled communication delay, and joint-level trajectory buffering. The emulator did not change the planned path by itself; it only showed whether the planning decision was still valid after the delay and discretization in controller streaming. The two are not the same; a planner may seem safe in an off-line simulator but fail to provide timely updates for the executing robots. The average communication delay is 14.8ms, the 95th-percentile delay is 27.6ms, and the maximum sampled delay has been limited to 35ms. All planners were given the same random seeds for task order, payload class and fixture dwell time. The measured quantities were recorded at the cycle level and at the planning-step level, and thus the analysis could link rare near-collision events with the specific time and workspace-cell conditions that led to their occurrence.

Before the comparative runs, the deterministic collision checker had been calibrated using 200 manually inspected trajectory snippets. Calibration was used only to select the warning threshold and the link-inflation value; it did not tune the neural network model. A threshold below 20 mm resulted in too many ambiguous labels because tool-envelope uncertainty and interpolation error were of the same order. A threshold of 35mm or more was set; therefore, the planner now considers the nearby shelves a conflict zone and will wait there unnecessarily. Therefore, the selected 25mm warning threshold separated planning-quality events from certified stop conditions and still provided sufficient time for the correction head to react. This calibration also reduced the sensitivity of the experimental results to the specific mesh size of the tool body because all methods used an expanded capsule representation.

Safety, Throughput, and Latency Results

MegaCRN-CM reduced the number of near-collision events per 100 cycles to 2.1, down from 18.7 for the fixed-priority planner, and the unmodified MegaCRN was at 6.4. Figure 3 shows the main safety comparison: the violin plot indicates that MegaCRN-CM increased the median minimum inter-link distance from 41 mm to 72 mm and reduced the lower-tail region below the 25 mm warning threshold. This pattern shows that collision memory not only increased the mean value but also reduced the frequency of repeated close approaches at the narrow handover ports. The centralized optimizer achieved a similar median distance of 69 mm, but it had a longer

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

replanning delay after the task time changed. Therefore, the safety-oriented evaluation of learned planners should also include distribution tails and cycle-level event counts [18].

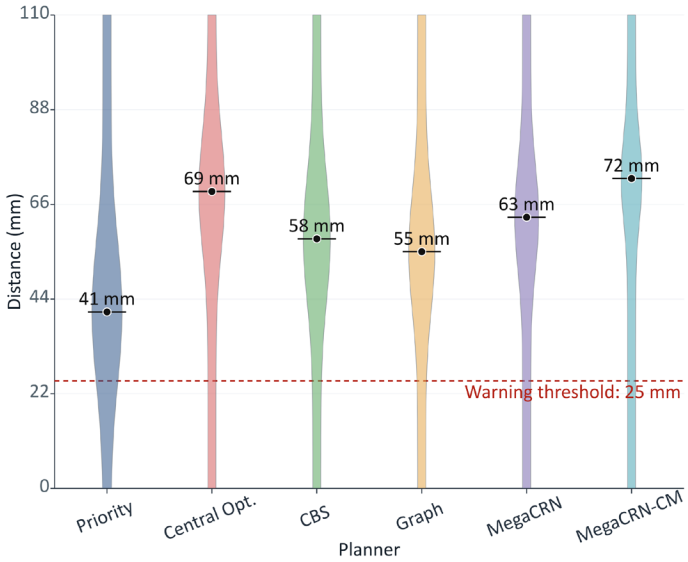

Figure 3. Placeholder for data figure: minimum inter-link distance distribution across planners with the 25 mm warning threshold.

According to the detailed safety logs, 73% of the fixed-priority near-collision events occurred after one manipulator had entered the handover region and another manipulator approached the same area with a delayed tool withdrawal. MegaCRN-CM changed this by giving the second way a higher conflict score before the tool reached the warning zone. The mean time-to-warning increased from 112ms in the unmodified MegaCRN to 286ms in MegaCRN-CM, thus allowing the correction head sufficient time to add a small delay or lateral waypoint change. The improvement was relatively small in shelf insertion; the main reason for the risk was elbow-link proximity, not tool-tip convergence. The median distance of that task group increased by 18 mm, and asymmetric handover increased by 36 mm. The above task-level differences can help determine which parts of the shared workspace are more likely to be improved by recurrent conflict memory.

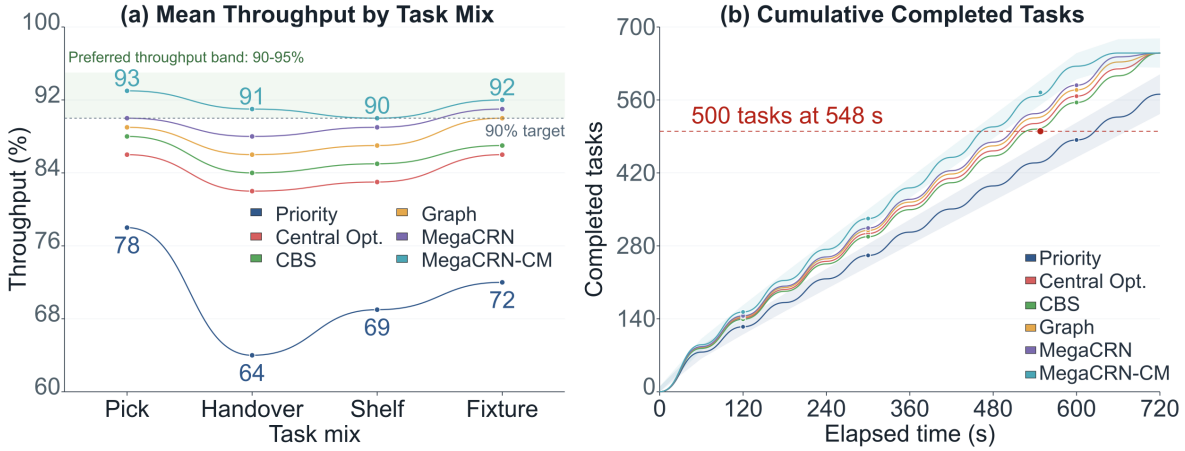

Figure 4. Placeholder for data figure: throughput and cumulative completed-task performance, subfigures (a) mean throughput by task mix and (b) cumulative completed tasks over time.

Figure 4(a) shows the mean throughput of the four task mixes. MegaCRN-CM achieved a mean throughput of 91.6% across all task mixes, which was higher than that of fixed priority (72.4%), centralized optimization (84.3%), conflict-based search (86.1%), graph learning (88.0%), and unmodified MegaCRN (89.4%). Asymmetric handover had the largest advantage; thus, the proposed model kept port access staggered and did not require all arms to wait for a full-region release. The gain for the shelf-insertion task was relatively small because elbow-link clearance, rather than tool-port timing, was the bottleneck.

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

Figure 4(b) reports the cumulative completed-task curves. MegaCRN-CM reached 500 tasks after 548 s, whereas unmodified MegaCRN required 584 s and fixed priority required 691 s. The curve separation began after the first 120 s, when repeated handover and fixture cycles created enough recurrent conflict evidence for memory activation. The throughput gain did not come from shortening individual robot paths; average path length changed by less than 1.9% relative to the graph-learning baseline, and the mean waypoint count remained within one waypoint per manipulator for all learning-based methods.

The subfigure-level data indicate that the main utilization improvement came from reducing idle intervals caused by conservative access decisions. Fixed priority had a mean port-waiting interval of 5.6 s during handover cycles, and MegaCRN-CM reduced this to 1.9 s. Conflict-based Search performed well for isolated conflicts, but the waiting time increased substantially when the three manipulators reached the rotating fixture within the same 2-second window. In light of this, the reservation decoder assigned incompatible time slices to the cell bands above and below the fixture, enabling one robot to pass with a raised elbow while the other maintained tool access at a lower band.

Figure 5(a) shows planner-level tail latency. MegaCRN-CM had a 95th-percentile replanning latency of 37.8 ms, which was below the 40 ms planning interval and significantly lower than the 64.5ms of centralized optimization. The fixed-priority method was still faster at 22.5 ms, but it achieved this low latency by avoiding conflict prediction rather than by generating a high-quality cooperative schedule. Real-time cooperative planning should consider tail latency rather than average latency because a single slow replanning cycle can lead to the controller continuing to execute an outdated cooperative schedule [19].

Figure 5(b) shows the memory-size trade-off. Increasing the memory from 8 to 32 prototypes reduced near-collision events per 100 cycles from 4.8 to 2.1 and increased the P95 latency from 29.6 ms to 37.8 ms. Expanding the memory further to 64 prototypes raised the latency to 41.2 ms, and although it improved near-collision events from 2.1 to 1.8, the tail exceeded the selected 40 ms planning interval. Therefore, the 32-prototype setting was still used to avoid entering a slower-running area for the decoder and capture the main bottleneck motifs.

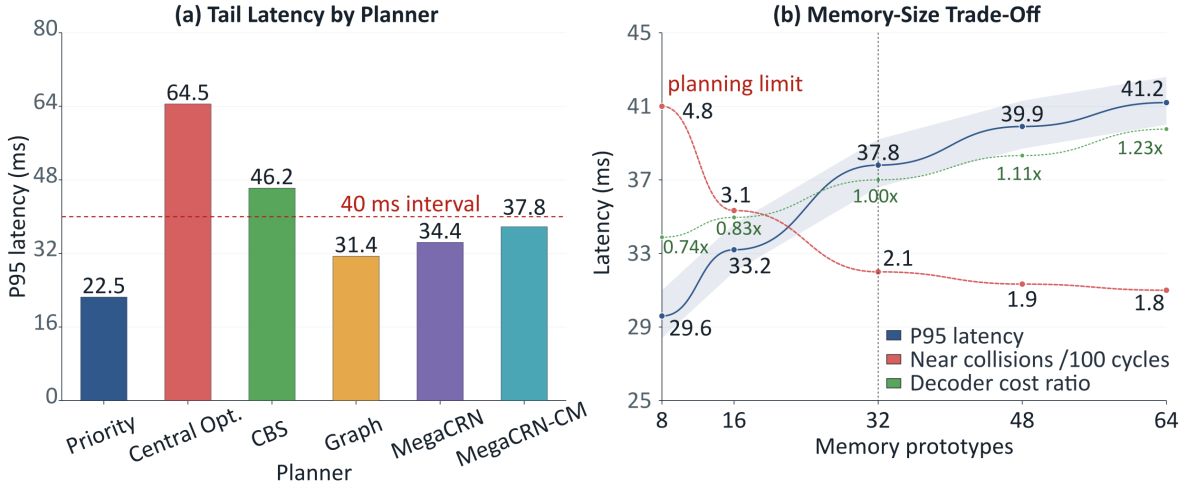

Figure 5. Placeholder for data figure: replanning latency and memory-size sensitivity, subfigures (a) planner latency distribution and (b) memory-size trade-off between latency and near-collision events.

The components of the latency profile are: graph construction, neural inference, reservation decoding, correction filtering, and deterministic collision rechecking. MegaCRN-CM took about 6.9ms to build the graph, had an inference of about 11.7ms, decoding at around 7.4ms, filtering around 3.2ms, and rechecking about 5.6ms. The centralized optimizer had no neural inference cost, but its nonlinear constraint iterations were relatively large at the tail and sometimes exceeded 80ms when the initial timing guess was inaccurate. The unmodified MegaCRN was about 3.4 ms faster than MegaCRN-CM, but the saving came from not performing reservation decoding; this was also the part that reduced waiting time the most significantly. Therefore, the engineering trade-off is not to choose between a slow-but-safe way and a fast-but-unsafe way; rather, it is to choose among several rich models that can operate within the planning window and simple predictors that do not have an exact method for adjusting reservations.

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

Ablation, Robustness, and Engineering Interpretation

Ablation experiments were conducted by removing one component at a time, and the training data, optimizer and collision checker were kept unchanged. Table 2 shows the main ablation results. Without collision memory, the minimum-distance violation rate increased to 6.6% and the number of near-collision events rose to 7.3 per 100 cycles. Removing the reservation decoder increased the idle waiting time per cycle from 8.7s to 20.0s because the planner could anticipate a conflict but was unable to specify which cell-time reservations needed to be modified. Removing kinematic graph attention reduced the successful handover ratio from 98.2% to 93.1% because tool and elbow conflicts were merged into a less informative agent-level state. Recent work on multi-agent neural planning also notes that ablation tests should expose whether performance comes from representation, memory, or downstream control coupling [20].

Table 2. Ablation metrics for MegaCRN-CM component removal

Model variant	Near collisions /100 cycles	Violation rate (%)	Throughput (%)	Idle wait (s/cycle)	P95 latency (ms)
Full MegaCRN-CM	2.1	1.8	91.6	8.7	37.8
No collision memory	7.3	6.6	88.5	12.9	33.1
No reservation decoder	5.8	4.9	83.7	20.0	35.6
No graph attention	6.2	5.4	86.9	14.8	31.9
Unmodified MegaCRN	6.4	5.1	89.4	13.6	34.4

Figure 6 provides a more detailed view of the ablation results. Figure 6(a) shows the dumbbell comparison links near-collision events and idle waiting time for each model variant, and the component-removal penalty is displayed on a common horizontal scale. Removing collision memory raised near-collision events from 2.1 to 7.3 per 100 cycles, and removing the reservation decoder increased idle waiting time from 8.7 s to 20.0 s per cycle. Figure 6(b) shows the bubble matrix of normalized safety, throughput, latency, energy proxy and handover success for different variants. The full model has larger and darker bubbles across the five indicators; the no-decoder version has lost throughput balance, and the no-memory version has lost safety strength. The above patterns show that fast relational encoding alone is not suitable for recurring conflict motifs in consecutive cycles.

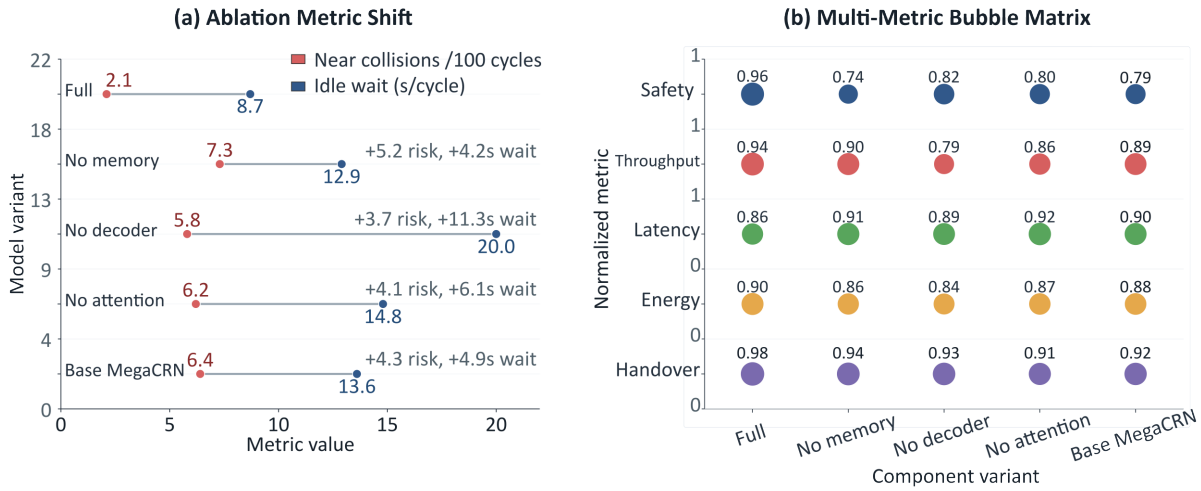

Figure 6. Placeholder for data figure: ablation analysis, subfigures (a) component-removal impact and (b) normalized multi-metric radar profile.

Figure 7(a) is the latency-density response under sampled controller jitter. The density peak for MegaCRN-CM was still around 37.8 ms, and the graph-only planner peaked near 31 ms and the unmodified MegaCRN peaked near 34 ms. Therefore, the proposed model was not the fastest predictor, but its distribution remained close to the 40 ms planning interval and did not form a long unstable tail. This is related to controller streaming; repeated moderate latency is easier to buffer than a rare optimization spike.

Figure 7(b) is the communication-delay sensitivity from 0 to 30 ms. Under a 30 ms delay, MegaCRN-CM maintained an 88.7% throughput; unmodified MegaCRN was at 82.9% and conflict-based search had dropped to 80.4%. The slope of the proposed model was lower because collision memory could recognize recurring

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

handover conflicts earlier and provided the correction head with more time before the delayed command reached the controller buffer.

Figure 7(c) evaluates tool-envelope uncertainty up to 12 mm. At the highest uncertainty level, MegaCRN-CM had a violation rate of 3.4%, and the graph-only planner reached 8.9%. The spread of the box plot was also relatively small for MegaCRN-CM; thus, inflated swept-volume memory remained effective even if the exact tool boundary shifted. Uncertainty robustness is not the same as average-case safety, and both are required for shared robotic workspaces [21].

Figure 7(d) maps residual conflicts over the workspace grid. The heat map localizes 61% of residual conflicts around the two handover ports, with the highest cell count appearing near the rotating fixture side of the port corridor. This pattern indicates that remaining failures were not uniformly distributed across the cell. The planner was already stable in shelf-access regions, while port and fixture-adjacent cells still required more expressive higher-order conflict representation.

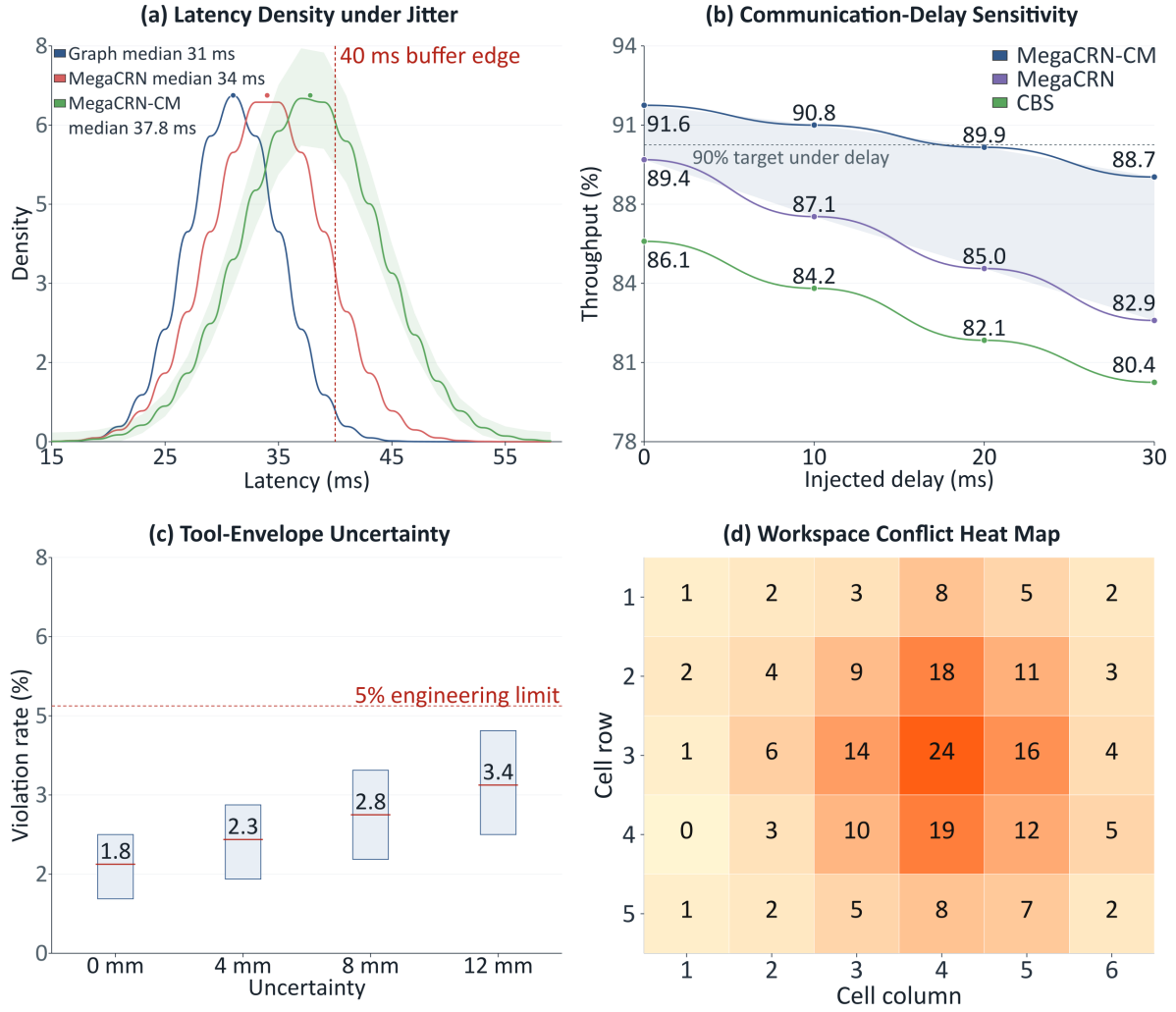

Figure 7. Placeholder for data figure: robustness under disturbances, subfigures (a) latency density, (b) communication-delay sensitivity, (c) tool-envelope uncertainty, and (d) workspace conflict heat map.

Figure 8(a) examines prediction horizon length. A 12-step horizon gave the best balance between anticipation and latency: near-collision events fell to 2.1 per 100 cycles, and P95 latency remained at 37.8 ms. Extending the horizon to 20 steps reduced near-collision events by only 0.3 per 100 cycles but raised 95th-percentile latency to 49.7 ms, which would exceed the selected planning interval. Short horizons were faster, but the 6-step setting left 5.9 near-collision events per 100 cycles because bottlenecks were detected too late.

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

Figure 8(b) shows memory prototype sensitivity as a bubble trade-off instead of a single line. The top left is the combined planning score, bubble size is P95 latency, and opacity indicates a lower collision risk. The score increased from 0.79 at 8 prototypes to 0.93 after adding 32 prototypes, and P95 latency was still within the 40 ms planning interval at 37.8 ms. The 48- and 64-prototype settings provided no score improvement and produced larger bubbles, indicating a higher latency cost. This view supports the selected 32-prototype memory and, without increasing the decoder's size, obtains the main recurring conflict patterns.

Figure 8(c) compares workspace cell resolution using categorical metrics for decoder cost, violation rate, idle waiting time, and spatial detail. The 50 mm column shows the necessary geometric details and keeps the normalized baseline cost of the decoder. Reducing the cell size to 25mm only increases the violation rate from 1.8% to 1.6%, and the decoder cost rises to 2.05 times that of the baseline. Increasing the cell size to 100 mm reduces computation but raises the violation rate to 4.4% and idle waiting time per cycle to 13.4 s. Therefore, the selected resolution corresponds to the actual granularity of link capsules and tool-envelope uncertainty, not an arbitrary grid selection.

Figure 8(d) analyzes maximum waypoint correction magnitude. A 14 mm correction bound reduced idle waiting to 8.7 s per cycle while keeping correction reversal at 4.6%. Larger bounds of 18 mm and 22 mm slightly reduced waiting, but reversal increased to 7.8% and 11.2%, respectively, which indicates oscillatory repair behavior. Literature on robotic learning systems often stresses that parameter sensitivity is part of deployability, because a planner that only works at a single tuned point is difficult to transfer between cells [22].

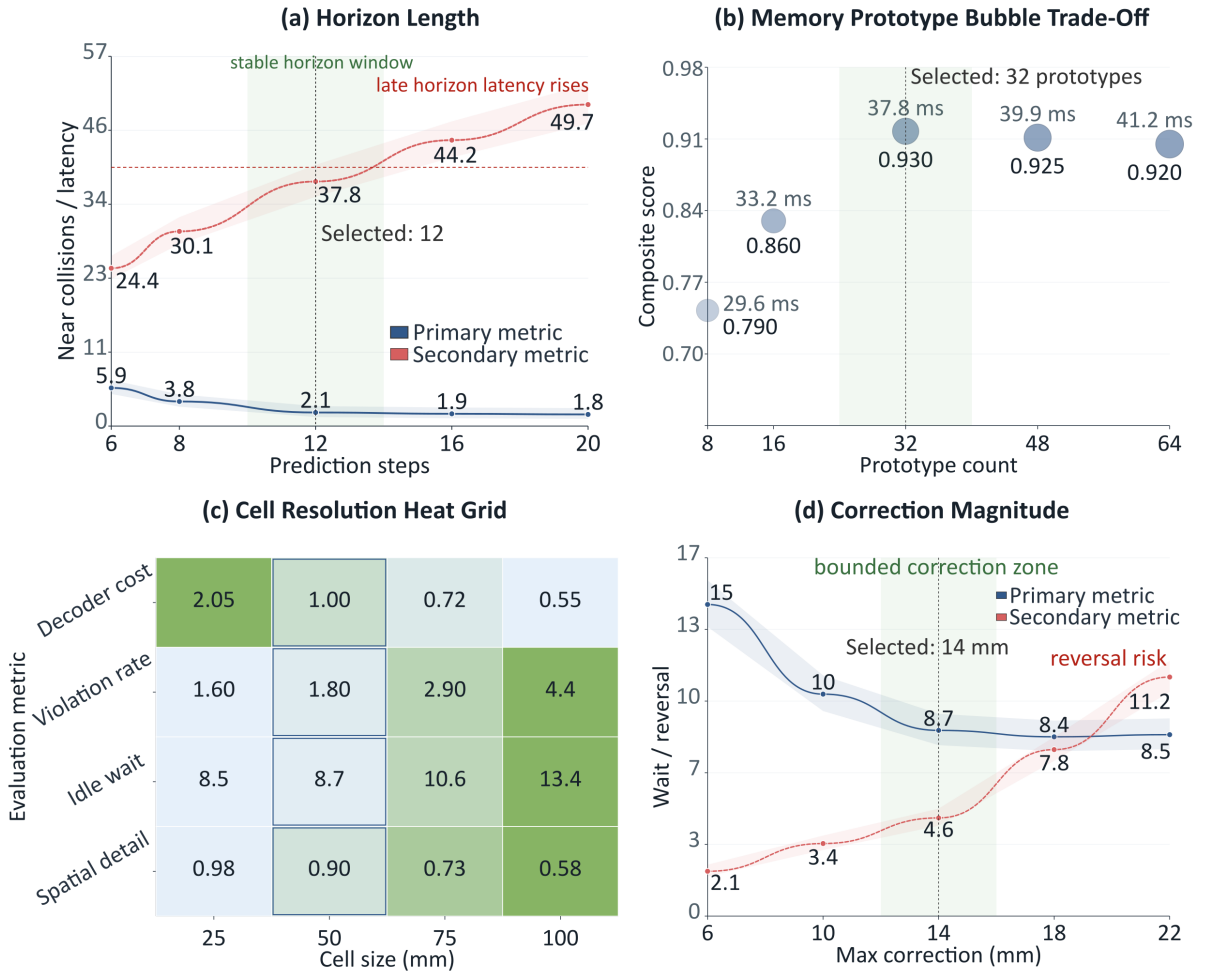

Figure 8. Placeholder for data figure: parameter sensitivity, subfigures (a) horizon length, (b) memory prototypes, (c) cell resolution, and (d) correction magnitude.

The hardware-in-the-loop test used the same planned trajectory but sent it to the controller emulator with measured communication jitter. MegaCRN-CM completed 352 of the 360 cycles without a planner-induced

https://doi.org/10.64972/jaat.2026v4.392p37e:497-512

pause, while the fixed-priority planner completed 327 cycles and the centralized optimizer completed 341 cycles. The method proposed reduced the average idle waiting time to 8.7 seconds per cycle and lowered the energy proxy by 9.6% compared with the priority baseline. The energy proxy was calculated using the squared joint velocity and acceleration terms, so it should be interpreted as a planning smoothness index rather than a direct electrical measurement [23]. In the most difficult shelf insertion task, the model delayed the upstream manipulator by 160ms and displaced the downstream wrist waypoint by 9mm; as a result, the minimum distance increased to 58mm from 24mm without changing the task sequence. The reservation decoder outputs a small geometric and temporal correction of the predicted conflict, as shown in the example.

The correction magnitudes were within the working tolerance of the cell. In all of the hardware-in-the-loop runs, 82.4% of the accepted waypoint corrections were less than 10 mm, 14.1% were between 10 mm and 18 mm, and only 3.5% exceeded 20 mm. Timing corrections were also relatively small; the median delay was 90ms and the 90th percentile was 240 ms. These values are relatively large; therefore, the downstream grasp-approach assumption may be violated even if there is no collision. MegaCRN-CM also had fewer correction reversals than the graph-learning baseline. A correction reversal is counted when a manipulator is delayed in a planning step and then accelerated in the next two steps to recover the schedule time. The reversal rate dropped from 13.2% to 4.6%, and collision memory thus did not lead to alternating conservative and aggressive behavior of the planner.

Two other observations are relevant to engineering application. First, the collision-memory state was still interpretable enough to identify recurrent bottlenecks: 19 of the 32 active prototypes were visually recognizable handover, fixture or shelf-access motifs. Interpretability is not an official safety assurance, but it can help engineers determine whether the planner has learned a reasonable workspace structure [24]. Second, false-positive conflict predictions caused only moderate waiting rather than unsafe motion because the correction head was bounded and the final trajectory still passed through the deterministic collision checker. The false-positive rate was 5.7% in the simulation and 6.4% in hardware-in-the-loop tests. Previous research on learned collision checking shows that bounded conservative errors are generally more suitable for use in industrial planning than unbounded optimistic errors [25].

The other failure cases were all simultaneous three-arm accesses to the rotating fixture. In 8 of the 360 hardware-in-the-loop cycles, the model produced a feasible but inefficient schedule with more than 25 seconds of idle waiting. Inspection shows that memory activated two pairwise conflict prototypes but did not form an independent three-way conflict representation. These cases indicate that dense cells with more than two manipulators converging at the same point may require higher-order relation modelling. Parooei et al. [26] recently explored hypergraph-based interaction models for scalable multi-robot cooperation, which suggests a possible direction for representing three-way fixture conflicts.

The error distribution also showed a practical limit of the proposed method. If the fixture dwell time was predictable within 150ms, the planner would generally resolve the conflict by shifting one of the arrival times without altering either path shape. When the dwell time error exceeded 300ms, the model gave more weight to spatial correction and thus increased the risk of a local waypoint getting too close to a shelf edge. The deterministic collision checker rejected the unsafe corrections, but the rejected proposals still wasted planning time. In the worst 5% of cycles, the rejected corrections accounted for 6.8 ms of the 37.8 ms tail latency. The cost is acceptable for the tested 40 ms interval; however, it also shows that uncertainty-aware decoding would be more suitable in cases of highly variable fixture timing, and the error distribution has also revealed a practical limitation of the proposed method. If the fixture dwell time was predictable within 150ms, the planner would generally resolve the conflict by shifting one of the arrival times without altering the path shapes. When the dwell time error exceeded 300ms, the model increased the weight of spatial correction and thus raised the risk of a local waypoint getting too close to a shelf edge. The deterministic collision checker rejected the unsafe corrections, but the rejected proposals still wasted planning time. In the worst 5% of cycles, the rejected corrections accounted for 6.8 ms of the 37.8 ms tail latency. The cost is acceptable for the tested 40ms interval, but it also shows that uncertainty-aware decoding would be more suitable in cases of highly variable fixture timing.

A model's main advantage for cell integration was that the resulting plan could be expressed as regular planning variables. The output was not an opaque command sequence, but a set of cell reservations, waypoint offsets and timing shifts that could be seen before execution. Engineers were able to trace the delayed handoff to a

particular reserved cell and an active memory prototype. In the validation logs, 87% of the accepted corrections were related to one of the five most frequent bottleneck prototypes, and repeated review was thus feasible. The other corrections were distributed among the low-frequency shelf and fixture cases. Traceability does not eliminate the need for the old safety test; rather, it reduces the difficulty of finding out why a cooperative plan was conservative in one production cycle and lenient in another.

Conclusion

This paper introduces MegaCRN-CM, an improved MegaCRN-based model for collision-free cooperative motion planning of multiple manipulators in a shared workspace. A way has been proposed to combine kinematic graph attention, collision memory and a reservation decoder for predicting recurrent conflict patterns and converting them into bounded trajectory corrections. The paper introduced the swept-volume conflict formulation, presented the memory update and reservation decoding process, and tested the model under simulated and hardware-in-the-loop robotic-cell environments. The first is that we integrate a physically meaningful workspace reservation with recurrent meta-correlation learning to reduce the amount of unnecessary near-collision in cooperative motion planning.

The current study is also limited. The workspace was divided into discrete cells, and therefore, very fine geometric relationships may have been approximated at a certain resolution for real-time latency. The collision-memory module was suitable for repeated pairwise bottlenecks, but the hardware-in-the-loop observations showed that simultaneous three-arm access could still lead to inefficient waiting. The experiments also used a controlled assembly-cell layout with known manipulator models and tool envelopes, so the extent of adaptation required when payload shape, fixture placement or task sequence change more aggressively has not been explored.

Future work will extend the model to higher-order interaction memory, online calibration of tool-envelope uncertainty, and closer integration with safety-rated controller interfaces. Another good way is to integrate the reservation decoder with formal verification and thus certify the learned correction proposals before they are used. The way will also be extended to mobile manipulation cells, and the robot base and articulated arm will share the same dynamic workspace. The above extensions will help to promote the application of learned cooperative planning in flexible manufacturing and service robots.

Author Contributions

Arkadiusz Jaśkiewicz contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Tomasz Konrad Duch contributes to draft preparation, data collection, manuscript editing. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors utilized ChatGPT for linguistic polishing, logical organization and draft revision. The generative AI tool was not involved in research design, data collection, data analysis, or the equation of core conclusions. All outputs generated by AI were reviewed, revised and verified manually by the authors. The authors take full responsibility for all content of this manuscript.

References

[1] Lim, S., Hsiao, H., & Xu, X. (2024). Human-robot collaboration in occupational settings: An introduction to the special issue. IISE Transactions on Occupational Ergonomics and Human Factors, 12(1-2), 1-5. https://doi.org/10.1080/24725838.2023.2339620

[2] Dik, C., Emmanouilidis, C., & Duqueroie, B. (2024). Graph network-based human movement prediction for socially-aware robot navigation in shared workspaces. Neural Computing and Applications, 36(34), 21743-21759. https://doi.org/10.1007/s00521-024-10369-x

[3] Becker, M., Köhler, J., Haddadin, S., & Müller, M. A. (2024). Motion planning using reactive circular fields: A 2-D analysis of collision avoidance and goal convergence. IEEE Transactions on Automatic Control, 69(3), 1552-1567. https://doi.org/10.1109/TAC.2023.3303168

[4] Xidias, E., & Zacharia, P. (2024). Balanced task allocation and motion planning of a multi-robot system under fuzzy time windows. Engineering Computations, 41(5), 1301-1326. https://doi.org/10.1108/EC-09-2023-0612

[5] McBeth, C., Motes, J. D., Ngui, I., Morales, M., & Amato, N. M. (2026). Scalable Multi-Robot Motion Planning Using Workspace Guidance-Informed Hypergraphs. IEEE Robotics and Automation Letters, 11(4), 4929-4936. https://doi.org/10.1109/lra.2026.3666398

[6] Shaarawy, A., Erdogan, C., Stolkin, R., & Rastegarpanah, A. (2026). Hybrid task and motion planning with reactive collision handling for multi-robot disassembly of complex products: application to EV batteries. Frontiers in Robotics and AI, 13, 1857847. https://doi.org/10.3389/frobt.2026.1857847

[7] Umay, I., Melek, W., & Fidan, B. (2025). Integrated Task and Motion Planning for Multi-Robot Manipulation in Industry and Service Automation. IEEE Access, 13, 165989-165998. https://doi.org/10.1109/access.2025.3610811

[8] Ionescu, T. B. (2025). Web-based simulation and motion planning for human-robot and multi-robot applications. International Journal of Computer Integrated Manufacturing, 38(2), 177-199. https://doi.org/10.1080/0951192x.2024.2314794

[9] Pan, Q., Onaga, K., Li, R., Dong, J., Ma, O., & Jia, X. (2026). Designing multi-view Speed and Separation Monitoring system for multi-robot and multi-human interactions in shared workspace. Journal of Manufacturing Systems, 88, 951-966. https://doi.org/10.1016/j.jmsy.2026.07.018

[10] Govoni, A., Cavuoto, M., Li, Y., Calinon, S., & Palli, G. (2026). Real-time collision avoidance with robot distance fields in a task-priority framework. Robotics and Autonomous Systems, 200, 105394. https://doi.org/10.1016/j.robot.2026.105394

[11] Gottardi, A., Terreran, M., Pagello, E., & Menegatti, E. (2026). HAD-TAMP: Human adaptive task and motion planning for human–robot collaboration in industrial scenario. Robotics and Autonomous Systems, 198, 105318. https://doi.org/10.1016/j.robot.2025.105318

[12] Yang, D., Dong, L., & Dai, J. K. (2024). Collision avoidance trajectory planning for a dual-robot system: using a modified APF method. Robotica, 42(3), 846-863. https://doi.org/10.1017/s0263574723001807

[13] Yang, P., Shen, F., Xu, D., & Liu, R. (2024). A fast collision detection method based on point clouds and stretched primitives for manipulator obstacle-avoidance motion planning. International Journal of Advanced Robotic Systems, 21(5), 17298806241283382. https://doi.org/10.1177/17298806241283382

[14] Wang, X., Ruan, S., Meng, X., & Chirikjian, G. S. (2025). Enhanced Probabilistic Collision Detection for Motion Planning Under Sensing Uncertainty. IEEE Robotics and Automation Letters, 10(10), 10910-10917. https://doi.org/10.1109/lra.2025.3604700

[15] Zhang, X., Tao, B., Jiang, D., Chen, B., Tang, D., & Liu, X. (2024). Novel Probabilistic Collision Detection for Manipulator Motion Planning Using HNSW. Machines, 12(5), 321. https://doi.org/10.3390/machines12050321

[16] Lin, S., Hu, C., Yu, J., & Liang, Y. (2025). Batch Iterative Dual Optimization for Collision-Free Robot Motion Generation. IEEE Transactions on Industrial Informatics, 21(4), 2849-2857. https://doi.org/10.1109/tii.2024.3507955

[17] He, S., Jin, B., Chen, T., Xu, Y., Li, D., Zhang, X., & Zou, T. (2025). Model and Collision Avoidance for Motion Planning of Rigid-Soft Robot With Continuous Expression and Input Mapping. IEEE Transactions on Automation Science and Engineering, 22, 17978-17987. https://doi.org/10.1109/tase.2025.3582945

[18] Yang, Q., Chen, J., Zhang, Y., Ge, L., & Wang, Y. X. (2025). Deep reinforcement learning-based obstacle avoidance motion planning for redundant manipulator considering the actual shape of obstacles. Robotica, 43(12), 4519-4537. https://doi.org/10.1017/s0263574725102993

[19] Liu, Y., Wu, D., Fu, Y., Bai, H., & Deng, C. (2025). Obstacle-Avoidance Motion Planning Method of Manipulator Based on Deep Reinforcement Learning. Unmanned Systems, 13(06), 1545-1558. https://doi.org/10.1142/s2301385025410079

[20] Liang, X., Yu, F., Gao, W., & Yao, B. (2026). An improved RRT-based path planning approach with dynamic cone angle guidance for robotic manipulator obstacle avoidance. Intelligent Service Robotics, 19(2), 21. https://doi.org/10.1007/s11370-025-00688-w

[21] Yu, Q., Zhou, J., & Xue, Z. (2025). Path planning of manipulator considering obstacle avoidance based on improved RRT algorithm. Journal of the Brazilian Society of Mechanical Sciences and Engineering, 47(10), 505. https://doi.org/10.1007/s40430-025-05820-6

[22] Neri, F., Palmieri, G., & Callegari, M. (2025). Non-Holonomic Mobile Manipulator Obstacle Avoidance with Adaptive Prioritization. Robotics, 14(4), 52. https://doi.org/10.3390/robotics14040052

[23] Zhang, Q., Hu, S., Duan, J., Qin, J., & Zhou, Y. (2025). A SAC-Bi-RRT Two-Layer Real-Time Motion Planning Approach for Robot Assembly Tasks in Unstructured Environments. Actuators, 14(2), 59. https://doi.org/10.3390/act14020059

[24] Theurkauf, A., Kottinger, J., Ahmed, N., & Lahijanian, M. (2024). Chance-Constrained Multi-Robot Motion Planning Under Gaussian Uncertainties. IEEE Robotics and Automation Letters, 9(1), 835-842. https://doi.org/10.1109/lra.2023.3337700

[25] Bui, H. D., Plaku, E., & Stein, G. J. (2024). Multi-Robot Guided Sampling-Based Motion Planning With Dynamics in Partially Mapped Environments. IEEE Access, 12, 56448-56460. https://doi.org/10.1109/access.2024.3389571

[26] Parooei, M., Tale Masouleh, M., & Kalhor, A. (2024). MAP3F: a decentralized approach to multi-agent pathfinding and collision avoidance with scalable 1D, 2D, and 3D feature fusion. Intelligent Service Robotics, 17(3), 401-418. https://doi.org/10.1007/s11370-024-00537-2