GRU Density Prediction for Gap-Based Navigation

Evaluating three planner configurations across 50 episodes in arena / Dynamic Gap

No GRU · *Current-density GRU* · *Future-density GRU (k = 5)*

Experimental setup

Same robot, same scenarios, 50 episodes per configuration

A — No GRU (baseline)

Dynamic Gap planner with the learned density cost disabled entirely.

Trajectory selection uses only geometric and scan-based costs. Establishes the floor and the ceiling for any GRU based evaluation.

B — Current-density GRU

GRU predicts a gap sector's density at the present timestep.

Inputs: 6 base features + change-in-density + aspect ratio. Regularized with dropout 0.1.

C — Future-density GRU (k=5)

GRU predicts what a gap's density will be 5 steps (~200 ms) into the future.

Same 8 input features. Only the training target changes: gt_sector_density shifted back 5 rows per gap track.

What every run measures

Each episode terminates in one of three ways — GOAL_REACHED, COLLISION, or TIMEOUT. Beyond the outcome we track total collision events (an episode can accumulate more than one), path length on successful runs, and time spent inside pedestrian personal space.

Path length is the metric that separates these three configurations most sharply. A robot that reaches the goal in 33 m and one that reaches it in 53 m are not doing the same thing, even though both count as a success.

Precursor: Important notes before the results

One asymmetry in the experimental design that shapes every comparison that follows

The configs are not identical across runs

The current-density 50-episode run was executed with the ORIGINAL planner config — before the parameter tuning sweep.

The future-density 50-episode run was executed with the TUNED config that came out of that sweep (Q_f = 0.05, integrate_maxt = 3.0, max_pose_to_scan_dist = 0.8).

So part of the future-density model's advantage on path length is attributable to the tuned planner, not the model.

The 15-episode head-to-head (later in this deck) DOES hold the config fixed, and is the cleaner model-vs-model comparison.

What is held constant

- Same scenarios: gru_learn_scenarios on 50ped_daniel env
- Same robot and episode ordering
- Same 50 episodes, 0–49
- Same evaluation harness and metrics

What varies

- Presence and target of the GRU
- Q_f, integrate_maxt, max_pose_to_scan_dist
- gruGapDensityCostWeight

SECTION A

No GRU — the baseline

Dynamic Gap with the learned density cost switched off. No neural network participates in trajectory scoring.

No GRU — what the planner uses

There are no learned features. This is the reference behaviour.

GRU features: none

useGruGapFeatureDensityCost_ = false.

The trajectory evaluator scores candidates purely on obstacle proximity (Q, pen_exp_weight), goal progress (Q_f), and scan-coverage validity (max_pose_to_scan_dist). No density prediction enters the cost.

Why we need this run

Without it, an improvement in the GRU runs is unattributable. Any claim that the density model reduces collisions has to be measured against a planner that never sees a density score at all.

Planner configuration

Parameter	Value
Q_f (goal progress)	0.01
Q (obstacle penalty)	1.0
pen_exp_weight	5.0
integrate_maxt	5.0 s
integrate_stept	0.5 s
max_pose_to_scan_dist	0.5 m
inf_ratio	1.0
assoc_thresh	1.0 m
rgc_angle	1.0 rad
gruGapDensityCostWeight	— (off)

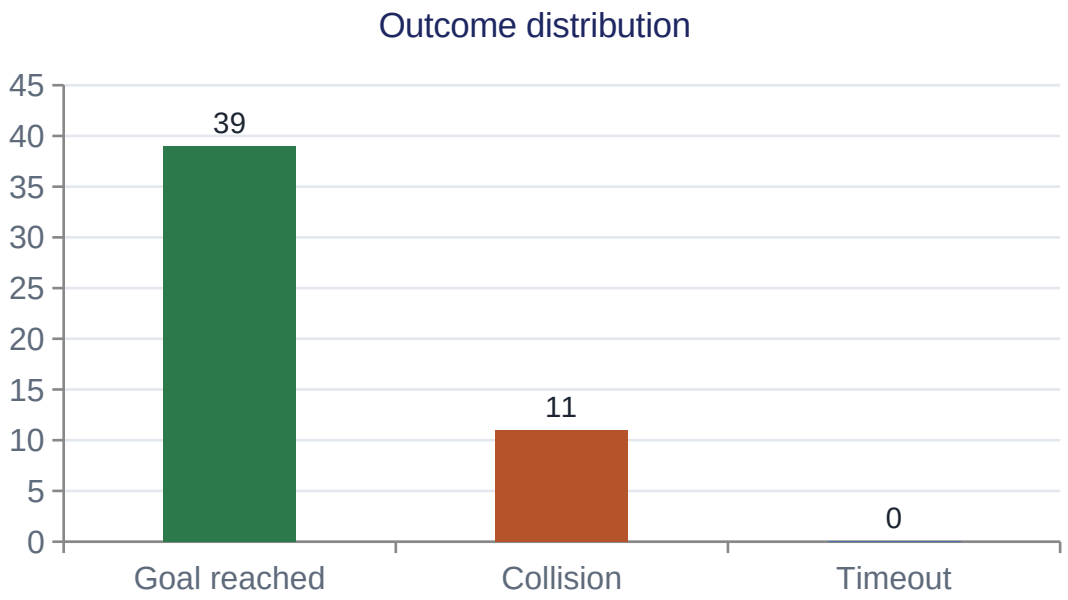
No GRU — results over 50 episodes

39 / 50
GOAL REACHED (78%)

11 / 50
ENDED IN COLLISION (22%)

0
TIMEOUTS

33.5 m
MEAN PATH LENGTH (successes)



Key figures

Total collision events: 14 (an episode can register more than one)
Episodes with ≥ 1 collision: 11 of 50 (22%)
Mean collisions per episode: 0.280

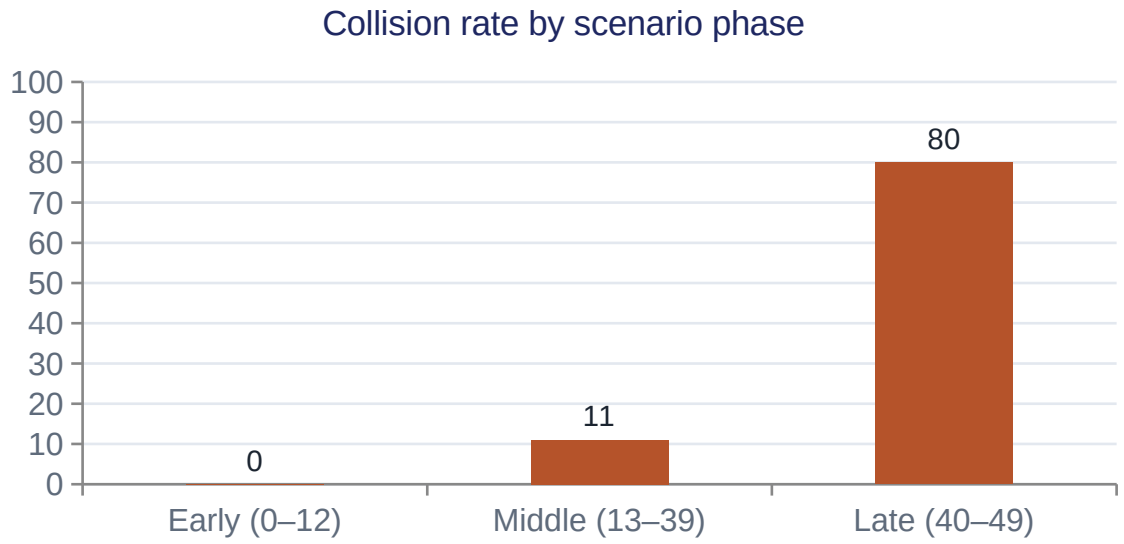
Path length: mean 33.5 m · median 32.7 m · max 41.8 m
Zero successful episodes exceeded 50 m.

Time in pedestrian personal space: 286.9 s (mean)

The planner is efficient and predictable. Every successful run is a short, direct path.

No GRU — where it fails

Collisions are not spread evenly. They are almost entirely a late-scenario phenomenon.



Collision episodes

22, 30, 31, 41, 42, 44, 45, 46, 47, 48, 49

Episodes 0–21 are almost perfectly clean — a single collision in 22 episodes. Episodes 41–49 collide 8 times out of 10.

This is a property of the scenario, not the planner. The late episodes place the robot in a genuinely harder crowd configuration.

Split up: 0 - 12; 13- 39; 40 - 49
What percentage of episodes in that chunk lead to a collision.

Collisions happen early in the path

Mean path length at the moment of collision: 28.2 m (range 21.9–37.8 m). The robot commits to a gap, that gap closes, and it hits something quickly. It does not wander first.

Why no timeouts

With Q_f = 0.01 and no density cost competing against it, goal progress is the only force acting on the terminal cost. The robot always drives forward. It either arrives or crashes — it never dithers.

SECTION B

GRU — density improvements

The GRU predicts a gap sector's density right now. Two derived input features and dropout regularization are added on top of the six base features.

Current-density GRU — model contents

Eight input features, one output: gt_sector_density at the present timestep

Six base features (read directly from the CSV)

sector_density
sector_dynamic_raw_gap_point_count
contained_raw_gap_point_count
sector_area
sector_angle_rad
sector_radius

These give the model the raw geometry of a gap sector and how many obstacle points fall inside it.

+ density_rate_of_change

ρ(t) − ρ(t−1) computed per tracked gap. Tells the model whether a gap is filling or emptying — trend, not just state.

+ aspect_ratio

1 / sector_angle_rad, clipped at 1e-4. Distinguishes a long narrow wedge from a short wide one at equal area.

Regularization & architecture

dropout = 0.1 (between GRU layers and before the linear head)
hidden_size = 128 · num_layers = 2 · seq_len = 10 · L1 loss
Trained on ~922 K rows (scenarios 0–24 combined)

Why these three additions

Instantaneous density is a snapshot: it cannot tell a safe gap from one about to close. Rate-of-change supplies the trend. Aspect ratio supplies the shape the area term hides. Dropout closed the train/val gap from 0.054 to 0.026.

Current-density GRU — configuration

The 50-episode run used the original planner config. Tuning came afterwards.

Config used for the 50-episode run

Parameter	Value
gruGapDensityCostWeight	**0.5**
Q_f	0.01
integrate_maxt	5.0 s
max_pose_to_scan_dist	0.5 m
maxGruPredictionAgeSec	3.0 s

How the GRU enters the cost

terminal_cost = (Q_f × goal_progress) + (weight × predicted_density)

With Q_f = 0.01 and weight = 0.5, a gap with predicted density 0.5 contributes 0.25 to terminal cost, while goal progress over a 5 m gap contributes 0.05. Density is roughly 5× more influential than reaching the goal.

Why each subsequent parameter changed

Q_f : 0.01 → 0.05
Raised because the robot was taking long, wandering routes. Q_f weights goal progress in the terminal cost; at 0.01 it was overwhelmed by the density term, so the planner had almost no incentive to actually advance.

integrate_maxt : 5.0 → 3.0 s
Lowered to cut per-cycle compute and reduce falling back to an idle trajectory. I feel like this helped with idling too.

max_pose_to_scan_dist : 0.5 → 0.8 m
Too many trajectory poses were pruned as being too far from scan evidence, leaving nothing to execute. Increasing this meant gives our robot some clarification at every timestep.

gruGapDensityCostWeight : tested 0.5 → 2.5 → 0.75
Raising it made the robot dramatically worse (see next slide). It over-trusted a model that is uncertain in exactly the dense scenes where the weight matters most.

Current-density GRU — results over 50 episodes

40 / 50

GOAL REACHED (80%)

8 / 50

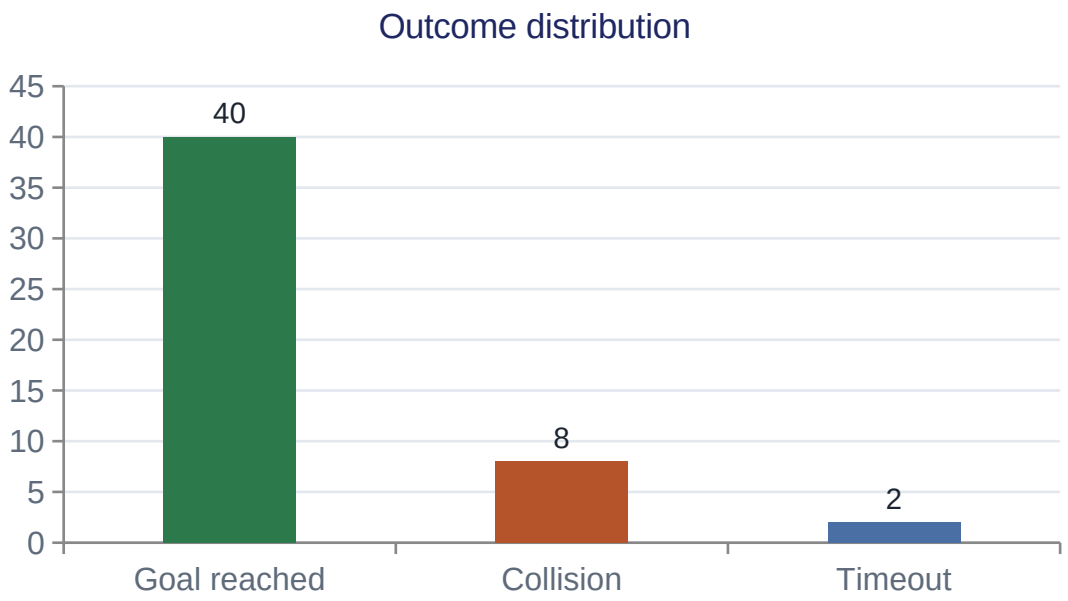
ENDED IN COLLISION (16%)

2

TIMEOUTS

52.7 m

MEAN PATH LENGTH (successes)



The headline hides the cost

Best collision numbers of any run: 11 total events, 9 episodes with ≥1 collision (18%), mean 0.220 per episode.

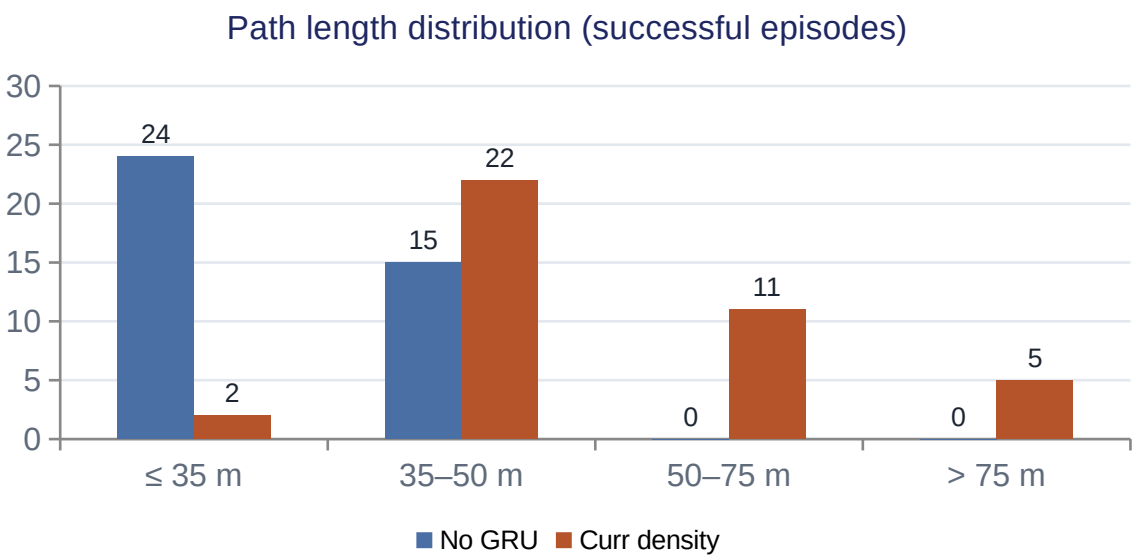
But the mean successful path is 52.7 m — 57% longer than the no-GRU baseline. Median 45.5 m, max 96.3 m.

16 of the 40 successful episodes exceeded 50 m. Only 2 finished under 35 m.

Time in personal space more than doubled: 286.9 s → 689.5 s.

Current-density GRU — analysis

Fewer collisions, but the robot bought them with distance



Wall-hugging and detours

Collision episodes: 7, 8, 11, 14, 40, 43, 47, 48, 49

Episode 11 collided after 79 m. Episode 14 after 117 m. Compare the no-GRU baseline, where the worst collision came at 38 m.

Walls carry near-zero density, so wall-adjacent trajectories always score well. Minimizing predicted density pulls the robot away from the direct route and into long arcs around the crowd.
Graph: **:** buckets of path length, grouping *successful* episodes by how long a path the robot walked to get there — "≤ 35 m", "35–50 m", "50–75 m", "> 75 m".

Both timeouts are 130 m walks

Episode 15: TIMEOUT after 131.2 m, zero collisions. Episode 40: TIMEOUT after 130.1 m, one collision.

The robot never stopped moving — it simply never converged on the goal. This is the density cost dominating Q_f = 0.01.

The trade-off, stated plainly

A direct 33 m path through a crowd is safer than a wandering 53 m path around it. Every extra metre is more time intersecting pedestrian trajectories. The model optimises density per gap; it does not optimise route completion.

Why we did not simply raise the density weight

Parameter sweep on the 15 hardest episodes (0–14)

Configuration	Goal reached	Collision	Timeouts	Coll. events	Mean path
w = 2.5, Q_f = 0.01	7/15 (47%)	7/15 (47%)	1	11	48.2 m
w = 0.75, Q_f = 0.03	9/15 (60%)	6/15 (40%)	0	12	37.8 m
w = 0.5, Q_f = 0.05 ✓	8/15 (53%)	7/15 (47%)	0	10	30.0 m

Raising the weight made it worse

At weight 2.5 the collision rate more than doubled versus weight 0.5, mean collisions per episode tripled (0.220 → 0.733), and a timeout appeared. Episode 6 alone logged 4 collision events across a 114 m path.

The likely mechanism

In dense scenes gap tracks break and reform constantly, so the GRU frequently runs on a partially-filled sequence buffer rather than a full 10-step context. Its predictions there are noisiest. A higher weight amplifies that noise straight into gap selection.

What actually fixed the path length was not the weight at all

Holding weight at 0.5 and instead raising Q_f to 0.05, dropping integrate_maxt to 3.0 s and raising max_pose_to_scan_dist to 0.8 m brought the mean path from 52.7 m down to 30.0 m and eliminated both timeouts. The density signal was never the problem — the balance against goal progress was.

SECTION C

GRU — future density as ground truth

Identical inputs. The training target is shifted forward: the model learns what a gap's density will be five planning steps (~200 ms) from now.

Future-density GRU — what changed

The inputs are unchanged. Only the label moves.

Same eight input features

The six base sector features, plus density_rate_of_change and aspect_ratio. Dropout 0.1, seq_len 10, L1 loss. Nothing about the architecture or the data collection changed.

The target shift

```
group[target] = group['gt_sector_density'].shift(-5)
```

Applied within each gap track, so a label never crosses a track boundary. Rows at the end of a track where t+5 does not exist become NaN and are dropped. The output feature is renamed future_sector_density_k5.

Why we tried it

Current density is a snapshot. In the collision analysis a gap's density rose from 0.12 to 2.79 over roughly six planning cycles — by the time instantaneous density flagged the danger the robot had already committed.

Five steps of lead time is meant to let the planner discount a gap that is currently passable but filling.

A bug that had to be fixed first

TrajectoryEvaluator.cpp hard-coded a lookup for the string "gt_sector_density". A future-density model publishes "future_sector_density_k5", so the lookup silently failed and every trajectory was scored with zero density cost.

The same bug existed in gru_gap_feature_node.py and suppressed all RViz markers.

Future-density GRU — results over 50 episodes

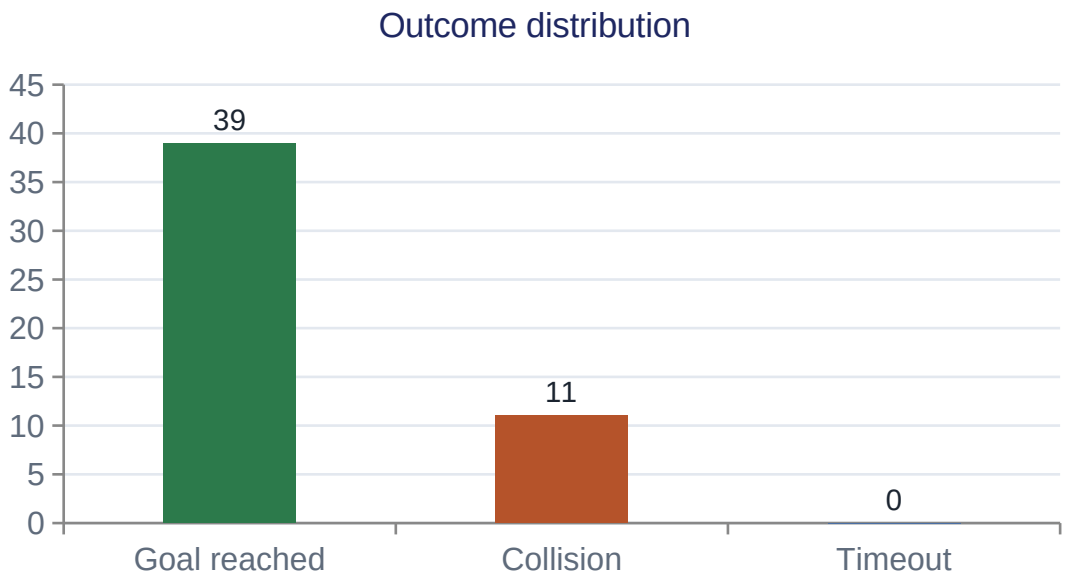
Run with the tuned planner config: weight 0.5, Q_f 0.05, integrate_maxt 3.0 s, max_pose_to_scan_dist 0.8 m

39 / 50
GOAL REACHED (78%)

11 / 50
ENDED IN COLLISION (22%)

0
TIMEOUTS

35.5 m
MEAN PATH LENGTH (successes)



Path efficiency almost fully recovered

Total collision events: 13 · episodes with ≥1 collision: 11 (22%) · mean 0.260 per episode.

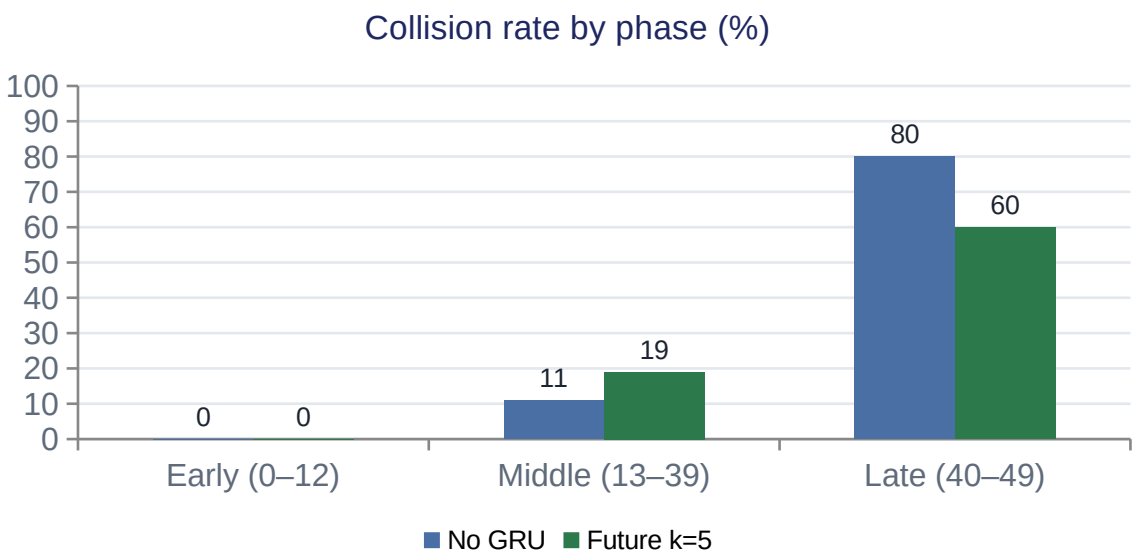
Mean path 35.5 m, median 33.9 m, max 63.7 m — against 52.7 m / 45.5 m / 96.3 m for current density.

22 successful episodes finished under 35 m (current density managed 2). Zero exceeded 75 m.

Time in personal space: 371.9 s, roughly half the current-density figure of 689.5 s.

Future-density GRU — analysis

Three episodes rescued, three episodes lost. The collisions moved rather than disappeared.



Rescued vs lost (against no GRU)

Rescued — 31, 41, 49
These collided without the GRU and now reach the goal. Episode 49's path fell from 22 m (crash) to 24 m (success).

Lost — 13, 27, 28
These were clean without the GRU. Under future density they collide after 47 m, 60 m and 54 m.

Net collision count: unchanged. The character of the failures changed completely.

Early-phase collisions eliminated

Episodes 0–12: zero collisions, exactly matching the no-GRU baseline, while current density collided 3 times there (23%). The lead time genuinely helps when the crowd is legible.

The middle-phase regressions are the tell

Episodes 27 and 28 are clean at 31 m and 39 m without a GRU. With future density the robot walks 60 m and 54 m and then hits something. It committed to a route the model forecast as safe, and the forecast expired.

Head to head — 50 episodes each

Green marks the best value in each row. No configuration wins every row.

Metric	No GRU	Current density	Future density (k=5)
Goal reached	39 (78%)	**40 (80%)**	39 (78%)
Ended in collision	11 (22%)	**8 (16%)**	11 (22%)
Timeouts	**0**	**2**	**0**
Total collision events	14	**11**	13
Episodes with ≥1 collision	11 (22%)	**9 (18%)**	11 (22%)
Mean collisions / episode	0.280	**0.220**	0.260
Mean path length	**33.5 m**	**52.7 m**	35.5 m
Median path length	**32.7 m**	**45.5 m**	33.9 m
Max path length	**41.8 m**	**96.3 m**	63.7 m
Mean collision path length	**28.2 m**	**58.0 m**	40.9 m
Time in personal space	**286.9 s**	**689.5 s**	371.9 s

Reading the table

Current density owns the collision column. No GRU owns the efficiency column. Future density is second on both — and is the only configuration that is never last.

The clean comparison — config held fixed

15 hardest episodes (0–14), identical planner config, only the model differs

Metric	Current density	Future density (k=5)
Goal reached	8/15 (53%)	**10/15 (67%)**
Ended in collision	7/15 (47%)	**5/15 (33%)**
Total collision events	10	**7**
Mean collisions / episode	0.667	**0.467**
Mean path length	**30.0 m**	32.1 m
Timeouts	0	0

With the config fixed, future density wins

30% fewer total collision events. 28% fewer episodes with a collision. Goal rate up 14 points.

Critically: three episodes improved (6, 8, 13) and not one regressed. That is a clean, one-directional result.

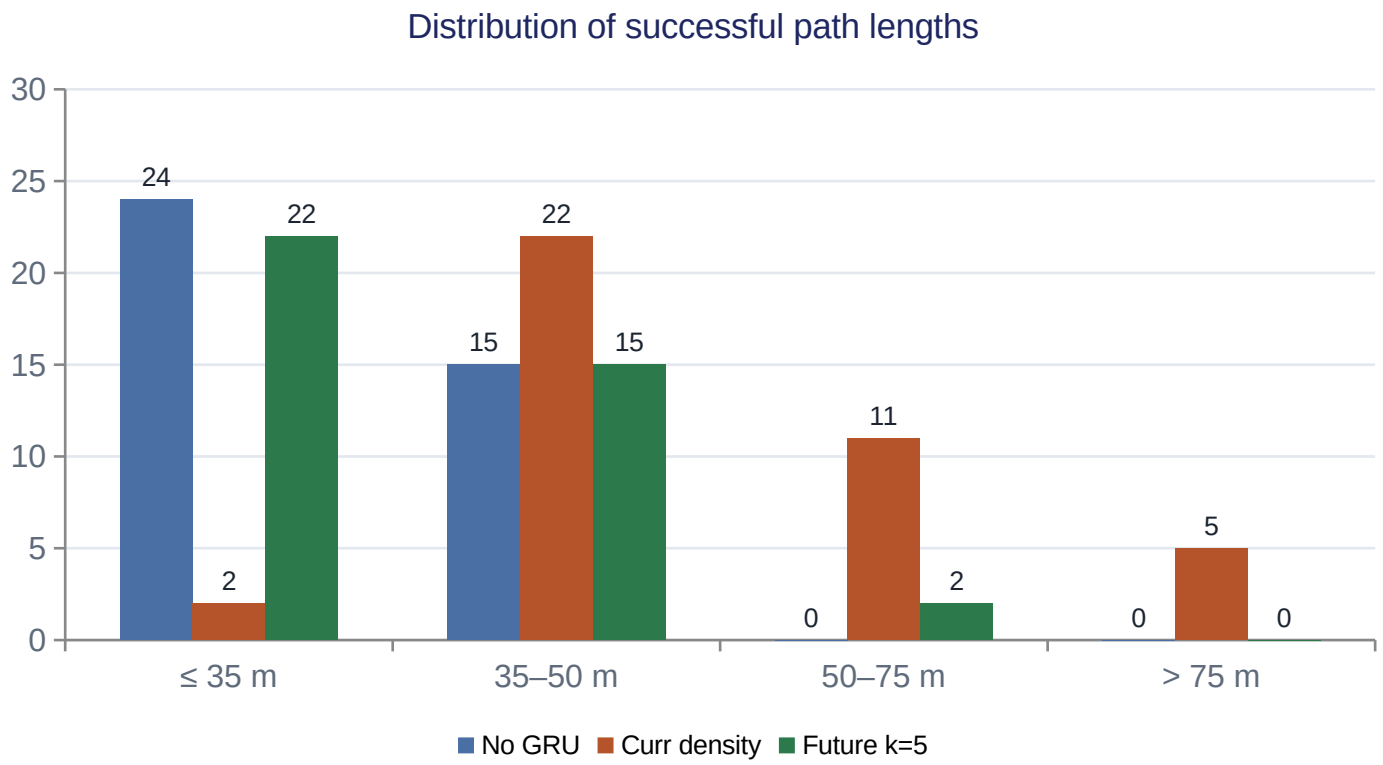
Why this slide matters more than the last

The 50-episode current-density run used the untuned config, so its 52.7 m paths conflate the model with the planner settings.

Here both models run on weight 0.5, Q_f 0.05, integrate_maxt 3.0 s, max_pose_to_scan_dist 0.8 m. The only difference is the training target.

Path length is the metric that separates them

Successful episodes only. A success at 90 m is not the same success as one at 30 m.



Why length equals risk

Every additional metre is more time the robot spends intersecting pedestrian trajectories. Distance is exposure.

Future density restores it

22 of 39 successes under 35 m, matching the no-GRU baseline's 24 of 39. Current density managed 2 of 40.

The personal-space figure tells the same story

Mean time inside pedestrian personal space: 286.9 s (no GRU) · 689.5 s (current density) · 371.9 s (future density). The current-density robot is not being more social — it is simply in the crowd for twice as long.

The situation determines the winner

Neither density target dominates. They fail in different places, for different reasons.

Sparse / legible crowds

Winner: Future density

Early episodes 0–12: future density and no GRU both record zero collisions. Current density collides 3 times (23%).

When pedestrian motion is predictable, a 200 ms forecast is accurate, and the planner gets real lead time on a gap that is about to close.

Dense / fast-changing crowds

Winner: Current density

Middle episodes 13–39: current density collides in 4% of episodes, future density in 19%.

Gap tracks break and reform constantly here. The GRU runs on a partially-filled sequence buffer, the forecast degrades, and a stale 200 ms prediction is worse than an accurate snapshot.

Genuinely hard geometry

Winner: nobody, yet

Late episodes 40–49 collide at 80% (no GRU), 50% (current) and 60% (future).

Both GRUs help versus no GRU, but all three fail more often than not. This is a scenario-difficulty problem, not a parameter problem — no amount of weight tuning fixes it.

The pattern

Future density buys lead time and pays for it with staleness. The denser and faster the scene, the worse that trade becomes.

Why the GRU is still the right direction

Read against the no-GRU baseline, both models move the needle where it matters most.

1. **Both GRUs beat the baseline in the hardest phase**

   Late episodes 40–49 collide 80% of the time without a GRU. Current density cuts that to 50%, future density to 60%. Where the scenario is hardest, the learned density score helps.

2. **The collision reduction is real and repeatable**

   Current density: 14 → 11 total collision events. On the fixed-config 15-episode test, future density: 10 → 7. Different runs, same direction.

3. **The costs are parameter problems, not model problems**

   Path inflation and timeouts were eliminated by Q_f, integrate_maxt and max_pose_to_scan_dist — not by touching the network. The density signal was never what was wrong.

4. **Prediction accuracy was never the bottleneck**

   The future-density model tracks ground truth closely across the whole density range. Navigation still degraded. That gap between good predictions and poor behaviour is a cost-function problem, and cost functions are tunable.

Where to go next

Four directions, ordered by expected value

1 — Blend the two targets

$(1 - \alpha) \cdot \rho_{current} + \alpha \cdot \rho_{future}$

Neither pure target wins outright, which is the signature of two complementary signals. Current density preserves path efficiency; future density supplies lead time. A blended label is the single most promising experiment.

2 — Sweep smaller k

k = 5 is ~200 ms of lookahead. Episodes 27 and 28 fail precisely because the robot commits to a route the forecast declared safe and the forecast expires.

k = 3 (~120 ms) shortens that commitment. Worth a 15-episode test at fixed config.

3 — Shorten seq_len in dense scenes

When a gap track breaks the buffer resets and the GRU predicts on 3–4 steps instead of 10. That is exactly where a high density weight does the most damage.

Retraining at seq_len 5–6 lets the buffer refill faster after a break, at the cost of temporal context in the easy scenes that are already clean.

4 — Collect data from the hard episodes

Episodes 40–49 defeat every configuration. The model has probably never seen enough of that crowd geometry to predict it well.

Targeted data collection there, then retraining, addresses a failure that no cost weight can reach.

In one sentence

The GRU reduces collisions. Which density target you choose
decides where it will fail.

Current density is safer in dense, fast-changing crowds and pays with 57% longer paths. Future density restores path efficiency and eliminates early-phase collisions, but its 200 ms forecast goes stale exactly where the crowd is hardest to read. With the planner config held fixed, future density is the better model — 30% fewer collision events, zero regressions. The next gain is a blended target, not another weight sweep.