INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Master Thesis

IU University of Applied Sciences

Study Program: M. Sc. in Applied Artificial Intelligence

Safety-Critical Path Planning with LQR-CBF-RRT* in 2D Dynamic Environments

Benjamin Müller

Enrolment number: IU14085243

Speicherstraße 14, 19055 Schwerin

Supervisor: Dr. Usman Akhtar

Date of submission: December 12, 2025

Abstract

This thesis studies an LQR-CBF-RRT* planner for safety-critical motion planning in two-dimensional dynamic environments with moving circular obstacles. The planner combines asymptotically optimal sampling (RRT*), cost-consistent linear-quadratic-regulator (LQR) steering, and a time-varying control-barrier-function (CBF) guard that, at scale, primarily acts as a predictive rejection gate by accepting or rejecting candidate time-stamped edges during expansion and rewiring under time-indexed obstacle forecasts. Dynamic obstacles follow a bounded-velocity disc model; accordingly, safety claims are stated with respect to this predictive model rather than interactive or adversarial obstacle behavior. Safety is framed as an acceptance test on the collision proportion under a pre-specified target. Three research questions structure the evaluation: whether the planner keeps the collision rate below a 1% budget under a low-speed generator, what computational and geometric overhead the safety layer adds relative to an LQR-RRT* baseline, and how the CBF gain α, the nominal expansion speed v_{nom}, and the extension length `step_len` affect success rate and minimum signed obstacle distance. Due to the computational cost of repeated edge-wise safety checks and rewiring, the current implementation targets offline or global planning rather than real-time receding-horizon control. In 3000 roll-outs with obstacle speeds up to 1.0 $\mathrm{m\,s^{-1}}$, no collisions are observed and the upper endpoint of the two-sided $95\,\%$ Wilson score interval is 0.13%, satisfying the 1% target with margin. On 100 paired scenes, median planning time increases by a factor of about 8 while median path length remains similar. A $3 \times 3 \times 2$ factor sweep yields 1799 successes in 1800 trials and shows that clearance is chiefly shaped by nominal speed and extension length, with only mild sensitivity to the CBF gain.

Keywords: Motion planning; RRT*; Control Barrier Functions; LQR; Dynamic obstacles; Safety

Contents

Lists and Indices

a Abbreviations

2D	2-dimensional
3D	3-dimensional
BIT*	batch informed tree star
CBF	control-barrier function
CLF	control Lyapunov function
CPU	central processing unit
CSV	comma-separated values
FMT*	fast marching tree star
GPU	graphics processing unit
I/O	input/output
IQR	interquartile range
JSON	JavaScript object notation
LQR	linear-quadratic regulator
MPC	model predictive control
OMPL	open motion planning library
PRM	probabilistic roadmap
PRM*	probabilistic roadmap star
QC	quadratically constrained
QCQP	quadratically constrained quadratic program
QP	quadratic-program
RNG	random number generator
RQ	research question
RRT	rapidly-exploring random tree
RRT*	rapidly-exploring random tree star
RRTX	dynamic RRT
SST	stable sparse RRT
TV	time-variant

b List of Symbols

Symbol	Name	Role	Unit
$\mathcal{W}$	workspace	subset of $\mathbb{R}^2$	
p_k	robot position	state at discrete time t_k	m
$t_0,\ t_k,\ \Delta t$	initial time, grid, step	$t_k = t_0 + k\,\Delta t$	s
u_k	control input	velocity command	$\mathrm{m\,s^{-1}}$
$\mathcal{U}$	input set	box constraints with $u_{\min}, u_{\max}$	
p_0	initial position	given	m
$\mathcal{G},\ p_g,\ \varepsilon_{\text{goal}}$	goal ball, center, radius	$\|p - p_g\|_2 \le \varepsilon_{\text{goal}}$	m
$\mathcal{O}(t)$	obstacle set	union of moving discs at time t	
$c_i^0,\ c_i(t)$	obstacle center (init, time-t)	$c_i(t) = c_i^0 + v_i t$	m
r_i	obstacle radius		m
$v_i,\ v_{\max}$	obstacle velocity, bound	$\|v_i\|_2 \le v_{\max}$	$\mathrm{m\,s^{-1}}$
M	number of obstacles	positive integer	
δ_i	inflation margin	precheck buffer	m
$h_i(p,t)$	CBF barrier (per obstacle)	$\|p - c_i(t)\|_2^2 - r_i^2$	$\mathrm{m^2}$
$h_{\min}(p,t)$	minimum barrier	$\min_i h_i(p,t)$	$\mathrm{m^2}$
$\mathcal{S}(t)$	safe set	$\{p \mid h_{\min}(p,t) \ge 0\}$	
N	horizon length	trajectory length (steps)	
Q, R	LQR weights	$Q \succeq 0,\ R \succ 0$	
P	Riccati solution	discrete ARE solution	
K	LQR gain	$(R + B^\top P B)^{-1} B^\top P A$	
$u_{\text{nom},k}$	nominal input	$-K(p_k - p_k^\star)$	
A, B, I_2	system matrices	$A = I_2,\ B = \Delta t\, I_2$	
$p_k^\star$	local target	steering target at step k	m
$x_k,\ V_k$	tracking error, Lyapunov	$x_k = p_k - p_k^\star,\ V_k = x_k^\top P x_k$	
ℓ_{edge}	edge length	$\sum_j \|p_{j+1} - p_j\|_2$	m
v_{nom}	nominal speed	for timestamping	$\mathrm{m\,s^{-1}}$
J	trajectory cost	$\sum (x_k^\top Q x_k + u_k^\top R u_k)$	
$k_{\text{cbf}},\ p_{\text{cbf}}$	CBF gains	$\alpha(h) = k_{\text{cbf}} h^{p_{\text{cbf}}}$	

c List of Figures

d List of Tables

e List of Listings

1 Introduction

1.1 Problem Definition

Autonomous vehicles have progressed from controlled pilots to limited service. In the United Kingdom, the CAVForth project launched passenger operations in 2023 with five Level-4 buses on a 22 km route across the Forth Road Bridge between Ferrytoll and Edinburgh Park (Stagecoach UK Bus, 2023). In the United States, Waymo's rider-only service reported 7.14×10^6 fully driverless miles through October 2023 across Phoenix, San Francisco, and Los Angeles (Kusano et al., 2024, p. S66). Despite these milestones, motion planners still lack broadly applicable formal safety guarantees once obstacles move, and recent reviews by Lindemann et al. (2023) and Zhang et al. (2025) report persistent challenges in dynamic, unpredictable conditions.

Against this backdrop, two ingredients are central to the discussion: the search backbone and the safety filter. Rapidly-exploring random tree star (RRT*) is provably asymptotically optimal in static workspaces under standard assumptions (Karaman & Frazzoli, 2011, p. 864), while control-barrier function (CBF) enforces forward invariance via a quadratic-program safety filter (Ames et al., 2017, p. 3861), including time-varying formulations for dynamic scenes (Dai et al., 2025, pp. 1–3). Integrations of CBFs with sampling-based planners, such as CBF-rapidly-exploring random tree (RRT), focus on generating feasible, collision-free trajectories for nonlinear systems with dynamic obstacles and do not include RRT*-style rewiring or provide asymptotic optimality guarantees (Yang et al., 2019, pp. 22, 28). Published accounts that concurrently demonstrate asymptotic optimality and formal safety against moving obstacles are unreported in the screened corpus, and surveys identify this combination as an open challenge (Zhang et al., 2025).

1.2 Research Objectives and Questions

The objective of this thesis is to evaluate the safety, computational overhead, and parameter sensitivity of an LQR-CBF-RRT* planner in 2-dimensional (2D) dynamic environments. This thesis adopts the LQR-CBF-RRT* formulation (Yang et al., 2025) and extends it to dynamic 2D environments with moving circular obstacles, retaining the search structure and cost metric of sampling-based planning while enforcing per-step, time-aware safety through a CBF guard. Time is explicit during expansion and rewiring via node time stamps. Obstacle motion is forecast for feasibility checks along candidate edges, and the linear-quadratic regulator (LQR) steerer provides a locally cost-aware connection

primitive and edge-cost evaluation for parent selection and rewiring, while nearest-neighbor queries remain Euclidean for simplicity. The study ties the method to measurable outcomes through three research questions and an evaluation protocol designed for auditability and exact replay.

The study addresses three research questions (RQs):

RQ 1 **Safety**: For circular dynamic obstacles with speed $v_{\max} \leq 1.0\,\mathrm{m\,s^{-1}}$, does the dynamic-obstacle LQR-CBF-RRT* keep the sampled collision rate below $1\,\%$ (upper endpoint of the two-sided Wilson $95\,\%$ interval) over 3000 roll-outs?

RQ 2 **Performance**: What extra median wall-clock planning time and path-cost overhead does the CBF layer add compared to baseline LQR-RRT* under scene/seed parity?

RQ 3 **Parameter sensitivity**: How do the CBF gain $\alpha \in \{0.25, 0.5, 1\}$, the nominal velocity $v_{\mathrm{nom}} \in \{1.5, 3.0, 5.0\}\,\mathrm{m\,s^{-1}}$, and the extension length $\texttt{step\_len} \in \{6, 12\}$ affect (i) success rate and (ii) minimum signed obstacle distance under fixed planner and environment settings?

To link method and metrics, each question maps to a concrete method. The CBF guard supplies execution-time safety evidence for RQ 1. Paired runs with identical scenes and seeds isolate timing and cost overhead for RQ 2. A controlled grid over α, v_{nom}, and $\texttt{step\_len}$ exposes sensitivity for RQ 3. The audit trail and replay tools enable verification, failure analysis, and reproducibility across all three.

The thesis contributes a safety-critical extension of LQR-CBF-RRT* to 2D dynamic environments and a quantitative comparison against a baseline LQR-RRT* planner in terms of safety, planning time, and path cost.

The contributions are:

- An LQR-CBF-RRT* algorithm that integrates a cost-consistent LQR steerer with a per-step, time-variant (TV)-CBF guard during sampling, parent selection, and rewiring. The guard predicts obstacle motion and enforces a relative-velocity barrier inequality along attempted edges.
- A time-aware expansion rule that stamps nodes with nominal arrival times and updates them during rewiring, aligning feasibility checks and cost evaluation on a shared time base.
- An audit trail that records per-roll-out logs and minimal JavaScript object notation (JSON) artifacts for failure cases, enabling post-hoc inspection, animation, and independent verification of collisions and near misses.

- A paired evaluation against LQR-RRT* reporting planning time and geometric cost with robust summaries, plus Wilson intervals for collision proportions over 3000 roll-outs.
- A sensitivity study over α, v_{nom}, and `step_len` reporting success rates and minimum signed distances to quantify safety margins under controlled changes.

1.3 Structure

To keep the safety story precise, the scope is narrow and the document is organized accordingly. The state space is planar. Obstacles are time-varying discs with bounded velocities. The robot uses a velocity-input model with fixed limits. The planner does not include learned predictors, stochastic CBFs, or 3-dimensional (3D) kinematics. Within these boundaries, the thesis is organised as follows. Section 1 motivates the safety gap in dynamic environments, states the research questions, and frames the contribution and scope. Section 2 reviews optimal sampling-based planners, LQR steering, control-barrier functions, and prior hybrids using a structured search and a focused gap analysis. Section 3 formalises the problem, defines the dynamic-obstacle model and assumptions, and derives the LQR-CBF-RRT* algorithm with the exact safety checks used in experiments. Section 4 reports results for safety, overhead, and sensitivity, includes a discussion of observed patterns, and summarises threats to validity. Section 5 concludes with a concise summary, maps contributions to objectives, and outlines extensions to stochastic predictions, robust CBFs, and higher-dimensional motion. Appendices provide search logs and code snippets.

2 Literature Review

This section grounds the thesis in existing scholarship on optimal sampling-based planning, linear-quadratic steering, control-barrier functions, and their hybrids. It opens with an account of the systematic search used to assemble the reference corpus, then progresses through five thematic sections whose sequence mirrors the algorithmic pipeline studied in Section 3. A final gap analysis explains where the proposed dynamic-obstacle LQR-CBF-RRT* contributes new knowledge.

2.1 Review Methodology

The PRISMA 2020 protocol (Page et al., 2021) structures the literature review to ensure transparency and reproducibility. This methodology defines explicit stages: literature identification, initial screening, document retrieval, eligibility assessment, and final inclusion. PRISMA guidelines outline clear criteria at each phase, which helps to increase consistency and reliability in decisions regarding study relevance. The PRISMA 2020 flow diagram (Fig. 1) illustrates the number of records at each review stage. Documentation of search queries, screening decisions, inclusion criteria, and exclusion reasons are maintained in Excel spreadsheets. To support independent verification and review replication, these records are archived and publicly accessible through the companion repository in the folder prisma_tables [1].

[1] https://github.com/Leg0shii/LQR_CBF_rrtStar/tree/dyn-tvcbf

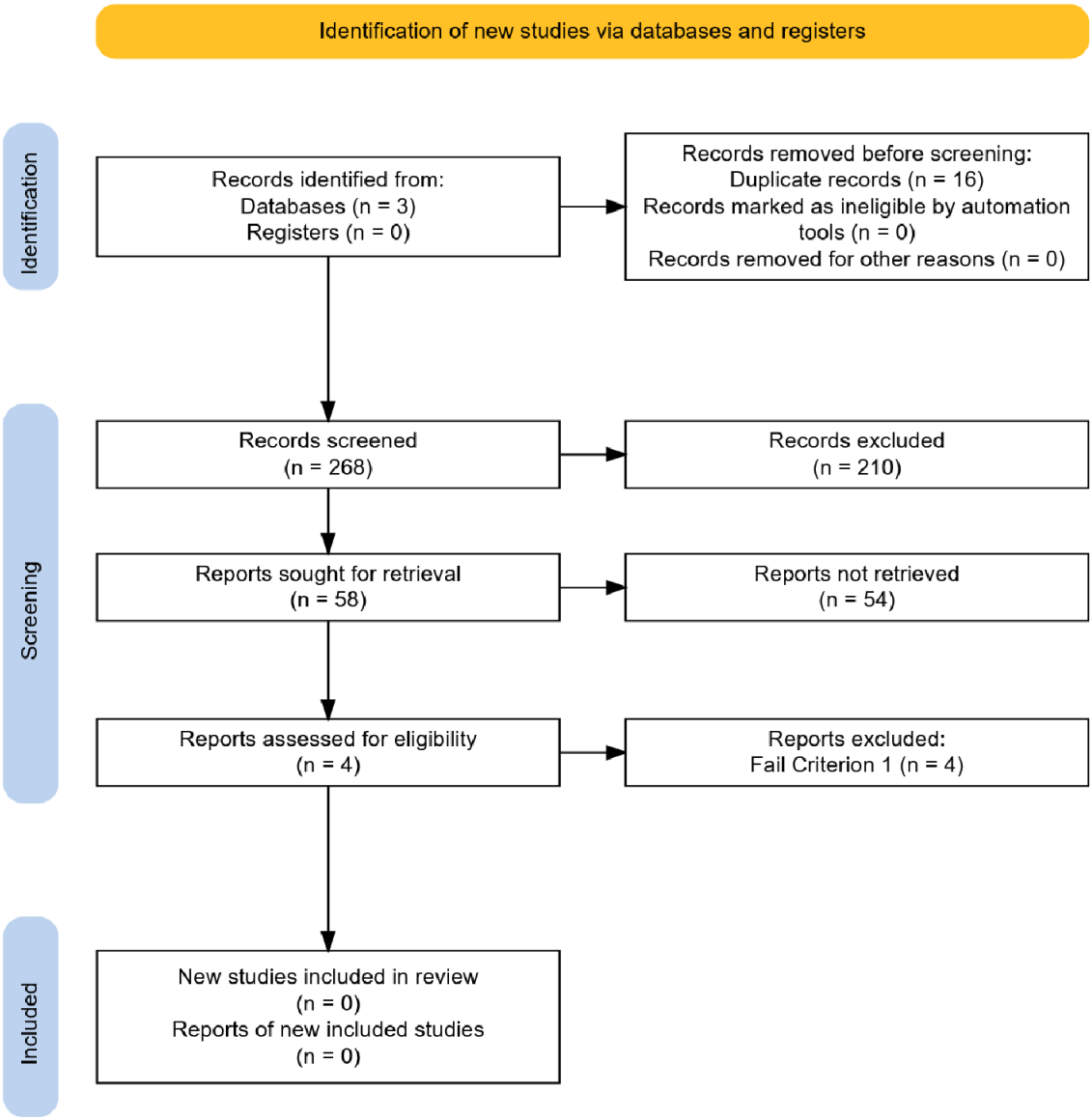

Fig. 1: PRISMA flow diagram illustrating study selection process. Source: Own illustration (modified) in accordance with Haddaway et al. (2022).

The methodology focuses specifically on optimal sampling-based planners, control-barrier functions, and linear-quadratic regulators for autonomous vehicles in dynamic environments.

2.1.1 Literature Collection

Following the PRISMA 2020 recommendations, three digital libraries, IEEE Xplore, SpringerLink, and Google Scholar, were queried. Query operators combined domain-specific keywords such as “CBF”, “RRT*”, “LQR”, “safe”, “planning”, “obstacles”, “autonomous driving”, “kinodynamic” and “dynamic”,

without altering the platforms' default filters. The exact search queries issued on each portal were:

- CBF gain parameter tuning
- CBF quadratic-program (QP) autonomous driving
- control barrier function autonomous
- dynamic obstacle avoidance vehicles
- kinodynamic RRT* autonomous vehicles
- LQR CBF RRT* vehicles
- probabilistic safe motion planning
- RRT replanning moving obstacles
- safe LQR RRT planning
- safe LQR steering vehicles
- time varying CBF planning

For each query the first ten hits, irrespective of publication year, access status, or document type, were recorded. This sweep captured journal and conference papers, technical reports, books, theses, and online courses. All records were exported to a structured Excel workbook that logs the title, author, publication year and more. The initial corpus totalled 284 entries.

2.1.2 Initial Screening

The workflow began with deduplication, which reduced the combined database exports to 268 unique records. Currency and methodological coherence were preserved by imposing explicit publication-year windows: optimal sampling-based planners were limited to 2009-2011, anchored by the probabilistic-optimality proofs for RRT* and probabilistic roadmap star (PRM*) (Karaman & Frazzoli, 2011). Steering functions and heuristics entered from 2012, when the discrete-time LQR-RRT* was introduced (Perez et al., 2012). CBF literature was accepted from 2014 onward, following the formalisation of forward-invariance conditions and quadratic-program safety filters (Ames et al., 2014). Hybrid LQR-CBF planners appeared after 2017 (Xu et al., 2018) and reproducible dynamic-obstacle benchmarks

were admitted from 2016-2017, when the ETH/UCY pedestrian datasets became established as standardised evaluation resources in planning and trajectory prediction research (Alahi et al., 2016).

Title screening kept any record with at least one optimal-planning string in the title: RRT*, LQR-RRT*, CBF, or CBF-QP. It dropped titles mentioning only RRT-Connect, generic potential fields, or non-ground-vehicle domains (e.g., unmanned aerial vehicles). Abstract screening kept a record when the abstract mentioned at least two keyword clusters: optimal sampling, LQR steering, CBF safety, dynamic-obstacle context, probabilistic safety, CBF gain tuning, or autonomous-vehicle tags. These filters left 58 records for full-text evaluation and set the median publication year at 2024, showing that most of the retained work is very recent.

2.1.3 Document Retrieval and Filtering

Full bibliographic information for the 58 surviving studies was exported to a dedicated spreadsheet and subjected to a full-text eligibility review. Full texts were then evaluated against five inclusion criteria consistent with the PRISMA 2020 ”Eligibility” stage (Page et al., 2021, p. 5):

1. **Methodological scope**: the study combines at least two core themes, namely asymptotically optimal sampling (e.g., RRT*), LQR-based steering or cost-to-go metrics, and CBF safety filtering.
2. **Scholarly quality**: the work is written in English and published in a peer-reviewed journal or conference, or archived as a reproducible pre-print that includes proofs, code, or data.
3. **Dynamic-obstacle relevance**: the evaluation addresses moving or time-varying obstacles pertinent to ground vehicles and reports quantitative safety or optimality metrics (e.g., collision rate, path cost, feasibility rate).
4. **Domain fit**: the dynamics model targets ground vehicles, studies centered on aerial, underwater, or humanoid platforms were excluded unless the mathematical development is explicitly platform-agnostic and transferable.
5. **Accessibility**: the full text must be openly available or reachable via university-library subscriptions.

Applying these criteria removed papers limited to static environments, purely theoretical CBF work, and studies that lacked an optimal planner. At the same time, the corpus retained method papers on

RRT*, LQR, and CBFs needed to support the main algorithm. In total, four studies met all inclusion criteria and advanced to the eligibility assessment.

2.1.4 Assessment for Eligibility

The full-text corpus of the four remaining papers was graded against eight binary criteria that operationalize the research gap stated in Section 1. A study advanced only when every criterion was met.

1. **Hybrid architecture**: The planner jointly employs (i) an asymptotically-optimal sampling backbone (RRT* or proven equivalent), (ii) an LQR-derived steering metric or feedback, and (iii) a CBF safety filter applied during tree growth, not just in post-processing.
2. **Dynamic-obstacle model**: Obstacles translate in $\mathbb{R}^2$ with non-zero velocity. Static-only papers were rejected. LQR-CBF-RRT* work that is expressly "offline" and excludes dynamic obstacles fails here.
3. **Ground-vehicle dynamics**: The plant is a unicycle, bicycle, or car-like model in the plane. Works centred on robot arms, unmanned aerial vehicles, 3D manipulators, or abstract point masses were excluded unless they gave a complete reduction to the planar unicycle case.
4. **Formal safety guarantee**: The paper proves forward-invariance (or an equivalent barrier certificate) for the closed-loop system under moving obstacles.
5. **Empirical stress-test**: At least 100 Monte-Carlo roll-outs in distinct dynamic scenes, with collision rate or risk explicitly reported.
6. **Comparative optimality**: Path-cost or time-to-goal is benchmarked against two or more dynamic-safe baselines under a common central processing unit (CPU) budget.
7. **Reproducibility**: Source code or parameter files are publicly available, or the authors provide enough detail to reproduce all figures without reverse engineering.
8. **Publication quality**: Peer-reviewed venue (top-tier robotics/control conference or journal) or an archival arXiv pre-print with accompanying artifacts. Grey literature without experiments was discarded.

Applying the eight eligibility guards left the corpus empty: no rigorously vetted study met the criteria. This zero-hit result points to a gap in safety-critical, asymptotically optimal planning for ground vehicles.

The next subsections therefore examine the algorithmic foundations and prior LQR- and CBF-based planners relevant to this thesis, starting with optimal sampling-based planning in Section 2.2 and culminating in the gap analysis in Section 2.7. The full screening log and exclusion notes appear in the companion repository.

2.2 Foundations and Extensions of Optimal Sampling-Based Planning

Karaman and Frazzoli describe sampling-based algorithms for optimal motion planning that draw random samples in a continuous state space, connect collision-free samples to build a graph of feasible trajectories, and return solutions whose best-path cost converges almost surely to the optimal value as the number of samples grows (Karaman & Frazzoli, 2011, pp. 847–848). In their RRT* variant, the tree expands through free space while a rewiring step ensures that vertices are reached through lower-cost paths, so the cost of the current best path decreases over time (Karaman & Frazzoli, 2011, pp. 855, 867). Fig. 2 illustrates this behaviour in a 2D maze, with the tree in blue and the current best path in magenta.

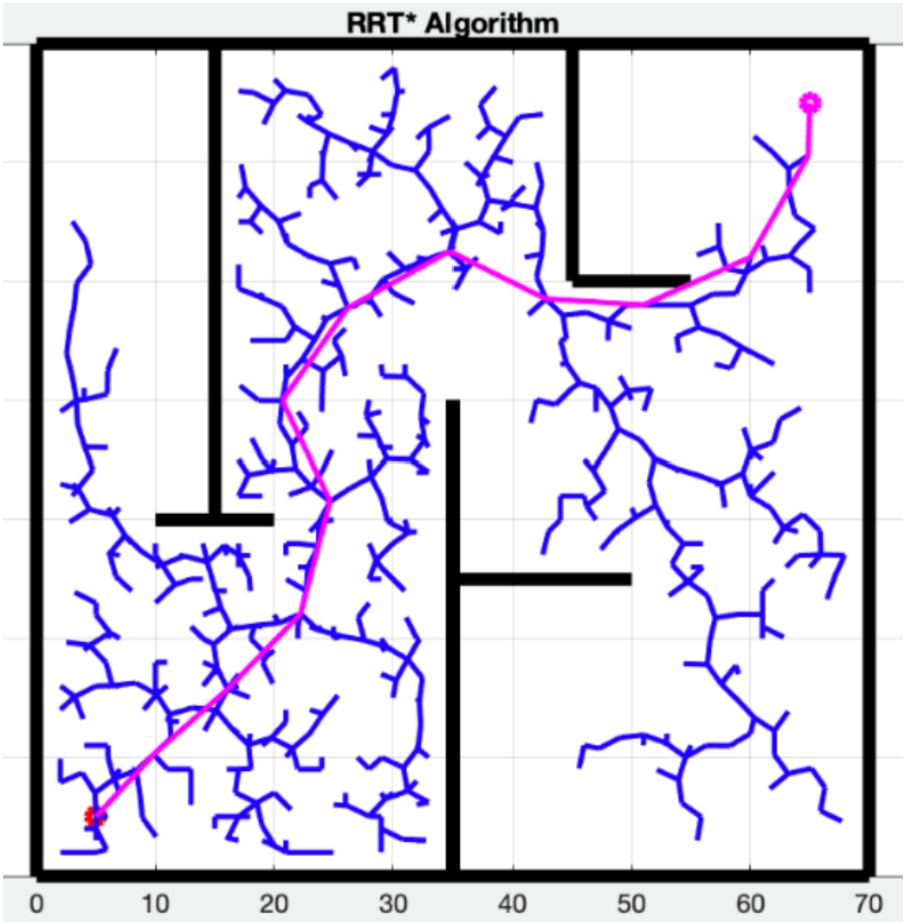

Fig. 2: Example of an RRT* search in a 2D maze environment. The tree (blue) grows outward from the start position (red) through collision-free space, progressively filling the environment. The magenta line shows the current lowest-cost path from start to goal, obtained through incremental rewiring. Source: Reproduced from Suwoyo et al. (2023, p. 279).

Earlier sampling-based planners such as probabilistic roadmap (PRM) and RRT were appreciated for their probabilistic completeness, yet Karaman and Frazzoli showed that, under mild technical conditions, the cost of the solution returned by such broadly used algorithms can converge almost surely to a non-optimal value as the number of samples grows (Karaman & Frazzoli, 2011, pp. 846–847). To address this limitation, they introduced the asymptotically optimal PRM* and RRT* algorithms, in which the connection radius or k-nearest neighbour count is chosen as a function of the number of vertices, with the corresponding neighbourhoods shrinking on the order of $\left(\frac{\log n}{n}\right)^{1/d}$, and the edge costs are induced by the same cost function as the continuous problem, and proved that under these choices the cost of the best path converges almost surely to the optimal value (Karaman & Frazzoli, 2011, pp. 846, 854–855).

Subsequent work addressed the computational cost of maintaining optimality guarantees. The fast marching tree star (FMT*) algorithm grows a tree by moving outward in cost-to-come space, repeat-

edly extracting the lowest-cost node from an open set and connecting it to nearby unvisited samples. It performs a forward dynamic-programming recursion over a disk-connected neighbourhood graph and uses lazy collision checking to reduce the number of collision checks while retaining asymptotic optimality (Janson et al., 2015, pp. 884, 886–887, 890). batch informed tree star (BIT*) extend batch-based optimal planning by combining batched sampling with heuristic ordering and incremental refinement. BIT* uses batches of samples to perform an ordered search over an implicit random geometric graph and reuses information between batches. After an initial solution, later batches restrict new samples to the part of the state space that can still contain a better path, for path-length minimization an ellipse with start and goal as foci. This informed batching and ordered evaluation of candidate edges improve the rate at which solution quality increases, especially in higher-dimensional problems, while BIT* remains probabilistically complete and asymptotically optimal (Gammell et al., 2015, pp. 3067–3071).

Dynamic RRT (RRTX) is an asymptotically optimal single-query replanning algorithm for dynamic environments with unpredictably changing obstacles (Otte & Frazzoli, 2016, pp. 797, 818). It reuses a single search graph as obstacles appear, disappear, or move and uses priority-queue-driven rewiring cascades to remodel the shortest-path tree after each detected change, enabling real-time replanning while inheriting probabilistic completeness and asymptotic optimality from RRT* (Otte & Frazzoli, 2016, pp. 797–798, 803, 817). These and other informed-sampling variants bias sampling toward regions that are more likely to improve the current solution, which can improve early solution quality and reduce exploration of irrelevant parts of the state space (Kyaw et al., 2024, pp. 1–2). However, when the current solution path is tortuous or the informed heuristic provides little useful information, the resulting restricted region can remain unnecessarily large, which lowers the probability of drawing samples that improve the path and slows convergence (Kyaw et al., 2024, pp. 1–3).

In parallel, sparse kinodynamic planners target bottlenecks in memory usage and nearest-neighbor search. stable sparse RRT (SST) instantiates the `SPARSE_BEST_FIRST_TREE` framework with best-first selection, fully random propagation, and pruning of nodes that are locally dominated in path cost, so that only representatives of the best-cost trajectories remain and the tree stays sparse (Y. Li et al., 2015, pp. 537, 541). Under delta-robustness assumptions this pruning maintains probabilistic completeness and asymptotically delta-robust near-optimal costs (Y. Li et al., 2015, p. 548). Because the active set of nodes and their witnesses remains finite, SST bounds near-neighbor query complexity and achieves per-iteration time smaller than RRT-BestNear, with running time and space requirements that can even improve on RRT (Y. Li et al., 2015, pp. 548, 555–556). While sparsity reduces graph size and near-neighbor cost, kinodynamic problems introduce additional requirements.

In its standard form, RRT* assumes access to an exact optimal steering function that can join any two

states by solving a two-point boundary value problem, which in practice restricts its use to systems with relatively simple dynamics (Webb & van den Berg, 2013, p. 5054). For differentially constrained or kinodynamic systems, computing such point-to-point optimal motions is generally difficult, so existing extensions of RRT* have either been tailored to particular simple models or have relied on approximate boundary-value solvers that only drive the system into a small neighbourhood of the target or guarantee only a bounded degree of suboptimality in cost (Webb & van den Berg, 2013, p. 5054). When the dynamics are linear, the associated optimal value function can be obtained analytically with low computational effort, and a similar LQR construction can be used for nonlinear systems by first linearizing the process dynamics around a chosen operating point (Perez et al., 2012, p. 2538).

Taken together, these variants adjust three main levers: memory usage, nearest-neighbor effort, and the cost of repairing the tree in changing environments. Foundational planners like PRM* and RRT* provide asymptotic optimality. Informed planners such as FMT* and BIT* bias samples towards promising regions to speed up convergence, but can converge more slowly when the heuristic offers little useful guidance. Sparse planners like SST prune non-essential nodes to keep the tree small and near-neighbor queries bounded, while dynamic variants such as RRTX repair the tree through lazy rewiring when obstacles or costs change.

Despite these structural differences, all such planners inherit a common requirement from RRT*: a local steering method that connects nearby states with dynamically feasible, cost-consistent trajectories. In kinodynamic settings, steering-based two-point boundary value solvers form a major computational bottleneck in asymptotically optimal planners (Arslan et al., 2017, p. 4991). In practice, the design of this steering or prediction step heavily influences both convergence behaviour and the feasibility of the resulting trajectories. This thesis therefore follows the line of work that improves the shared steering primitive rather than proposing a new tree structure. The next subsection examines the LQR as an efficient steering method for RRT* and its extensions.

2.3 LQR and LQR-derived local Methods

The empirical lesson of the past decade is that the quality of a sampling-based planner hinges on the local model it uses to advance the tree. When the steering routine can only produce approximate trajectories, RRT* must repeatedly re-propagate large portions of the tree, inflating the vertex count. By contrast, for systems with controllable linear dynamics, Kinodynamic RRT* uses a fixed-final-state, free-final-time optimal controller that connects any pair of states exactly, and for the subclass with a nilpotent dynamics matrix these optimal trajectories admit closed-form expressions that

keep the additional computational cost compared to holonomic RRT* small (Webb & van den Berg, 2013, pp. 5054–5055). To handle nonlinear systems, the planner linearizes the dynamics about the sampled state, creating a first-order Taylor approximation. The fixed-final-state, free-final-time controller then generates trajectories based on these linearized dynamics (Webb & van den Berg, 2013, p. 5059).

Alternatively, LQR-RRT* linearises the dynamics to automatically derive a distance metric and a feedback extension controller. This approach achieves asymptotic optimality on torque-limited and underactuated systems without domain-specific parameter tuning (Perez et al., 2012, p. 2537). Solving the algebraic Riccati equation yields the matrix P, which determines the feedback gain (Hanks & Skelton, 1991, p. 3)

$$K = -R^{-1}B^{\top}P,$$

the closed-loop matrix

$$A_{\text{cl}} = A - BR^{-1}B^{\top}P,$$

and the value

$$V(x_0) = x_0^{\top}Px_0.$$

Standard RRT* uses Euclidean balls to find nearby nodes, assuming geometric proximity implies dynamic reachability. This breaks down for torque-limited systems: a pendulum state might lie millimeters away yet require impossible torques to reach (Perez et al., 2012, p. 2538). Fig. 3 shows this on a torque-limited pendulum. Green nodes sit close to x_{new} in state space but cannot reach it, the required control would exceed torque limits or demand physically impossible trajectories. Blue nodes, selected by LQRNear, can actually steer to the target. The curved paths (color-coded by cost) follow the systems' natural dynamics rather than fighting against them. LQR-RRT* instead ranks nodes by control effort, the actual cost to connect them under the systems' constraints.

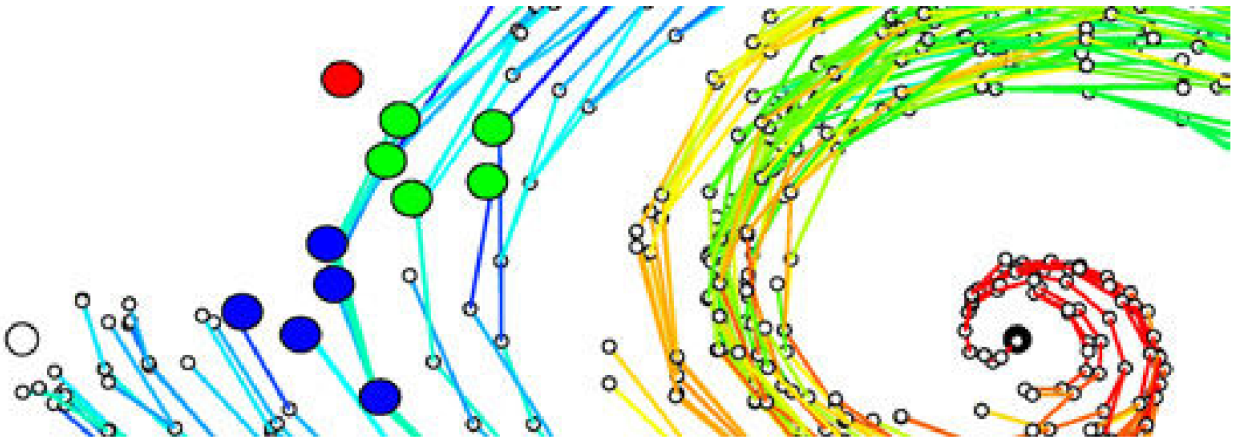

Fig. 3: LQR vs. Euclidean neighborhood on the torque-limited pendulum. Euclidean “near” nodes (green) are closer in $\mathbb{R}^d$ but often cannot reach x_{new}. LQRNear (blue) selects nodes that can reach x_{new} with minimum LQR effort; policies are false-colored by cost. Source: Reproduced from (Perez et al., 2012, p. 2538).

In the Kinodynamic RRT* framework, experiments on a double integrator and a quadrotor confirm that gradually tightening the neighbor radius accelerates planning while preserving asymptotic optimality. This shrinking neighborhood reduces the computational burden of edge evaluation without degrading the final solution cost (Webb & van den Berg, 2013, p. 5060). Experiments demonstrate that replacing the Euclidean metric with the infinite-horizon LQR cost-to-go and extending edges via the associated feedback policy accelerates convergence to the optimal solution. This approach successfully solves torque-limited and underactuated tasks where Euclidean heuristics fail. By ranking neighbours using the LQR value function, the planner directs growth along dynamically efficient paths (Perez et al., 2012, pp. 2539–2541). Realising these benefits in real time, however, requires mitigating the per-edge Riccati solve cost. Solving a continuous-time algebraic Riccati equation for every candidate edge entails a computational complexity of $O\left(n^3\right)$ (Laub, 1979, p. 916), (Bartels & Stewart, 1972, p. 821). This cubic scaling imposes a significant computational burden for RRT* as the state dimension increases. Three optimisation strategies can mitigate that burden.

- A low-rank modification algorithm reuses the previous Riccati factorisation, reducing the computational cost of a rank-k update to $O\left(kn^2\right)$, significantly below the $O\left(n^3\right)$ complexity of a fresh solution (Nielsen et al., 2013, p. 3689).
- Off-loading the matrix-inverse computations to a hybrid CPU-graphics processing unit (GPU) platform yields a 21-fold speed-up on a $n = 5177$ benchmark Riccati problem compared to a sequential CPU implementation (Ezzatti et al., 2011, pp. 44–45).

- Greedy RRT* restricts sampling to an L^2 greedy informed set defined by the state on the current path with the maximum heuristic cost, thereby reducing the search volume and accelerating convergence (Kyaw et al., 2024, pp. 1–4).
- The Policy Algebraic Equation solves directly for the feedback policy in nm variables, reducing the number of unknowns compared to the growth of the standard Algebraic Riccati Equation (Sassano, 2024, p. 370).

After cutting the computational cost, we must still determine whether the linearised model keeps its collision avoidance guarantees once the robot strays from the nominal trajectory. The following paragraphs tackles that question by analysing regions of attraction and their safety properties.

Tedrake et al. (2010, p. 1041) use the quadratic cost-to-go from infinite-horizon LQR synthesis as a Lyapunov candidate for the nonlinear system. Sums-of-squares verification identifies a sub-level set of this function where the trajectory remains asymptotically stable under the full nonlinear dynamics. The LQR-trees algorithm assembles a sparse library of these stabilized trajectories to probabilistically cover the controllable state space (Tedrake et al., 2010, p. 1038). Majumdar (2013, p. 3) extends this framework to robust motion planning by computing ”funnels” that bound the reachable set of the system under disturbances and modeling uncertainty. While these funnels explicitly certify safety within their validity regions, the reliance on offline precomputation limits their flexibility in changing environments. The following paragraph surveys second-order local optimal-control methods that adjust to new conditions online: iterative LQR, differential dynamic programming, and distributionally robust LQR extensions.

Iterative LQR and differential dynamic programming expand the dynamics to first and second order, respectively, while approximating the cost to second order (W. Li & Todorov, 2004, p. 223). Differential dynamic programming uses Newton steps to achieve quadratic convergence, while iterative LQR relies on a Gauss-Newton approximation (Tassa et al., 2014, pp. 1168–1169). These methods yield locally optimal policies valid near a nominal trajectory rather than a global value function (W. Li & Todorov, 2004, p. 228). Distributionally robust extensions of LQR minimise the expected cost for the worst-case distribution within a data-driven ambiguity set. The resulting controller guarantees mean-square stability with high probability (Coppens et al., 2020, p. 521). Robust LQR formulations for linear time-invariant systems minimise a worst-case discounted quadratic cost under parametric uncertainty. However, this formulation does not enforce explicit state constraints, preventing the certification of collision avoidance in dynamic workspaces (Jongeneel et al., 2019, pp. 6742–6743).

Despite these advances, three obstacles remain. First, Berkenkamp et al. show that a controller derived from a linearized model with parameter errors stabilizes the system only within a limited neigh-

bourhood of the equilibrium. Model mismatch causes the true region of attraction to remain small, leading to instability for large deviations (Berkenkamp et al., 2017, pp. 8–9). Second, a quadratic, time-agnostic LQR distance treats the workspace as static. Without a time stamp, edges may be certified safe in a frozen map yet collide with moving obstacles, a limitation addressed by time-augmented RRT variants for dynamic environments (Guo et al., 2023, p. 7136). Third, even with parallelisation, Jallet et al. report that a single constrained Riccati recursion for a 37-state system takes 4.4 ms to 10.3 ms (Jallet et al., 2024, p. 11). Given that sampling-based planners often require thousands of connections per cycle, this cost dominates the real-time budget for high-dimensional systems. These limitations motivate CBFs, which provide forward invariance under time-varying constraints and enable fast per-step safety checks. When integrated into sampling-based planners with consistent cost and feasibility models, asymptotic optimality can be preserved.

2.4 Control-Barrier Functions for safety-critical Control

CBFs formulate safety as constraints within a real-time optimization framework. A continuously differentiable function $h(x)$ defines the safe set $\mathcal{C}$ as the set of states where $h(x) \geq 0$ (Ames et al., 2017, pp. 3861–3863). At each control cycle, the system ensures safety by enforcing a condition on the time derivative of h. Specifically, a Zeroing Control Barrier Function requires the control input u to satisfy a linear inequality involving the system dynamics and the value of h (Ames et al., 2017, p. 3867).

This condition is implemented via a QP that mediates between safety and performance. The controller finds the input u that minimizes the deviation from a desired nominal input subject to the affine barrier constraint (Ames et al., 2017, p. 3869). By satisfying this constraint, the controller renders the set $\mathcal{C}$ forward invariant, ensuring the system never leaves the safe set. This approach directly addresses the limitations of purely discrete planners: it accounts for the true nonlinear dynamics and imposes safety as a computationally efficient constraint manageable at high update rates (Ames et al., 2017, pp. 3866–3868).

The runtime architecture is shown in Fig. 4. The nominal controller produces u_{des}, which the Active Set Invariance Filter filters to produce u_{act} that satisfies the safety constraint, and u_{act} is applied to the system.

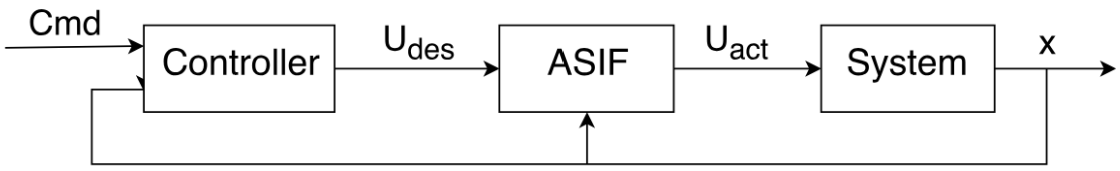

Fig. 4: Active Set Invariance Filter: the desired input u_{des} is filtered by a CBF-QP to produce the applied input u_{act} that enforces safety. Source: Reproduced from (Ames et al., 2019, p. 3428).

For systems controlled in discrete time, Agrawal and Sreenath (2017, p. 1) extend the barrier framework to the discrete domain. They introduce Discrete-Time Control Barrier Functions and exponential variants (Agrawal & Sreenath, 2017, p. 1). These formulations replace the continuous derivative condition with a stepwise difference constraint on the barrier function, guaranteeing that the safety set remains forward invariant at every sampling instant (Agrawal & Sreenath, 2017, pp. 3–4). CBF-QP formulations encode safety specifications as inequality constraints on the control input (Xu et al., 2018, p. 1218). Because the system dynamics are affine, these constraints are linear in the control variable, keeping the optimization problem convex and allowing independently specified safety and performance rules to be enforced simultaneously within a single QP framework. Adding further CBF constraints preserves convexity, so multiple rules can be enforced simultaneously (Xu et al., 2018, pp. 1218, 1225). Beyond simple conjunctions, recent constructions encode unions and mixed Boolean logic into a single continuously differentiable barrier (Molnar & Ames, 2023, p. 3615). These methods use smooth approximations, such as log-sum-exp functions, to capture complex safety specifications within the standard CBF framework (Molnar & Ames, 2023, pp. 3617, 3619).

To unify performance and safety, Ames et al. (2017) formulate a QP that mediates between these potentially conflicting specifications. In this framework, the control Lyapunov function (CLF) stability objective is treated as a soft constraint using a relaxation variable, whereas the CBF operates as a hard constraint. This structure ensures that safety is strictly maintained even if the stability objective must be violated (Ames et al., 2017, p. 3862). Additionally, defining the barrier function on a domain larger than the immediate safe set allows the system to account for model perturbations while preserving forward invariance (Ames et al., 2017, p. 3865). Ames et al. (2017) discuss this connection and illustrate it via prior work. In addition, they report real-time CLF-CBF QP implementations at $200\,\text{Hz}$ to $1000\,\text{Hz}$ on embedded systems, supporting online feasibility.

Building on the same formulation, time-varying CBFs define safe sets that change explicitly with time, enabling dynamic obstacle avoidance (Wu & Sreenath, 2016, p. 2); (Dai et al., 2025, p. 1).

These methods treat safety as time-dependent, allowing the safe region to move or deform. Wu and Sreenath (2016, p. 5) extend this concept to systems evolving on Riemannian manifolds, providing a geometric formulation for safety on nonlinear configuration spaces. Recent applications demonstrate these techniques on whole-body collision avoidance for mobile robots (Huang et al., 2024, p. 5149) and dynamic avoidance for high-degree-of-freedom manipulators (Dai et al., 2025, p. 7).

Exponential CBFs extend the method to high relative-degree constraints, such as position-based safety requirements in mechanical systems (Nguyen & Sreenath, 2016, p. 325). By enforcing conditions on higher-order derivatives of the barrier function, the resulting safety conditions remain affine in the control input (Nguyen & Sreenath, 2016, p. 323). This structure allows the online filter to be solved as a convex QP at each time step, modifying a nominal controller only when necessary to maintain safety (Ames et al., 2019, p. 3423). However, with input bounds and multiple active safety constraints, the resulting CLF-CBF QPs may become infeasible; adding a single feasibility constraint enforced as a CBF maintains compatibility with the existing constraints and guarantees feasibility of each QP in the sequence (Xiao et al., 2022, p. 1). A complementary framework provides a locally Lipschitz controller that handles multiple CBFs while respecting input constraints and offering robustness properties (Shaw Cortez et al., 2022, p. 1742). CLF-CBF QPs can also induce undesired equilibrium points in the interior of the safe set or at parts of its boundary. This effect is analyzed and can be eliminated via a small modification of the QP (Tan & Dimarogonas, 2024, p. 1). In practice, aggressive switching schemes can yield jerky behavior (Choi et al., 2021, p. 6816). Furthermore, sampled-data implementations often require margin terms to maintain safety, which can lead to conservativeness under coarse sampling (Taylor et al., 2022, p. 7127).

Axis-aligned rectangular obstacles are considered. For systems with decoupled second-order position dynamics in x and y, the axis-aligned bounds reduce to per-coordinate linear inequalities. Discrete-time CBFs can enforce these bounds at each control step (Agrawal & Sreenath, 2017, p. 1). Additionally, sampled-data CBF conditions can guarantee safety over the inter-sample interval for piecewise-constant controllers (Breeden et al., 2022, p. 367). The minimum distance function between polytopes is non-differentiable. Duality-based nonsmooth CBF formulations resolve this issue and enable real-time safety enforcement via QPs (Thirugnanam et al., 2022, pp. 2239–2240). When a continuously differentiable surrogate is preferred, log-sum-exp compositions yield a single smooth CBF that approximates max/min set operations (Molnar & Ames, 2023, pp. 3617–3618). Differentiable optimization based TV-CBFs demonstrate hardware-validated dynamic obstacle avoidance with faster computation times than model predictive control (MPC) baselines (Dai et al., 2025, pp. 1, 5).

CBF filters have been integrated with sampling-based planners, including RRT and RRT*, to enforce safety constraints directly within the steering controller. This integration enables the synthesis of

collision-free trajectories without relying on explicit collision-checking functions (Peng et al., 2023, p. 3650); (Chu et al., 2025, p. 9). While standard sampling-based algorithms offer probabilistic completeness, geometric approaches often generate non-smooth paths that fail to comply with complex system dynamics (Peng et al., 2023, p. 3649). To address this, variants such as LQR-RRT* utilize the steering procedure to ensure dynamic feasibility and define edge costs based on an optimal control policy rather than Euclidean distance (Chu et al., 2025, p. 9). The following subsection details hybrid LQR-CBF planners that combine these steering mechanics.

2.5 Hybrid LQR-CBF Planners

Hybrid planners that combine sampling-based search with optimal local steering and online safety certificates have evolved along three primary design axes: enforcing barrier conditions directly within the steering controller (Yang et al., 2019, p. 24), computing optimal local steering via linear quadratic regulation (Perez et al., 2012, p. 2537), and adapting sampling distributions based on search progress (Ahmad et al., 2022, p. 4513). One approach enforces safety constraints directly within the local steering controller. Yang et al. (2019, p. 24) introduced CBF-RRT, which solves a sequence of CBF-constrained QPs during each edge extension. This algorithm eliminates explicit collision checking and nearest-neighbour search; instead, it samples existing vertices directly and ensures separation via the forward invariance of the CBF safety set (Yang et al., 2019, pp. 24–25). The method handles dynamic obstacles in continuous time by shifting the computational burden to the sequence of QPs solved inside the steering function (Yang et al., 2019, p. 21).

A second stream investigates optimal planning using CBF constraints. Ahmad et al. (2022, pp. 4513, 4516) proposed CBF-RRT*, an algorithm that preserves probabilistic completeness and bypasses explicit collision checking by employing CBF-based local planners. The method utilizes exploratory CBF-QP rollouts for tree expansion and an exact CLF-CBF-QP steering controller for parent selection and rewiring (Ahmad et al., 2022, pp. 4515–4516). To enhance sampling efficiency, the authors introduce an importance sampling scheme that adapts the density function using the cross-entropy method on elite trajectories (Ahmad et al., 2022, p. 4517). Although this design unifies the safety models of planning and execution, it requires solving sequences of QPs, which creates significant computational overhead (Yang et al., 2025, p. 3700).

Recent work investigates two complementary questions: whether CLF and CBF constraints can be certified as jointly feasible along planned edges before simulation, and whether safety enforcement can be amortized by learning a safe steering law. The C-CLF-CBF-RRT algorithm introduces

optimization-based compatibility checks to verify if CLF and CBF constraints are simultaneously feasible along the regions connecting candidate nodes (Mestres et al., 2025, p. 6442). For linear systems subject to polytopic or ellipsoidal constraints, this verification reduces to a quadratically constrained quadratic program (QCQP), allowing the planner to generate dynamically feasible paths without performing closed-loop rollouts at every sampling step (Mestres et al., 2025, p. 6442). In parallel, Yu et al. (2024, p. 14348) learn a CBF-induced neural controller and integrate it into the local steering function. The controller accepts either signed distance states or LiDAR point clouds to generate safe control inputs, reducing the number of explored nodes and increasing success rates for 4-7 degree of freedom manipulators in both static and dynamic environments (Yu et al., 2024, pp. 14349–14352). The method preserves probabilistic completeness by optionally discarding unsafe nominal actions rather than modifying them during expansion (Yu et al., 2024, p. 14351). These approaches decouple sampling strategy from safety enforcement: one method screens candidate edges via a fast CLF-CBF compatibility check (Mestres et al., 2025, p. 6442), while the other employs a CBF-induced neural steering controller to improve connectivity and reduce the search tree size (Yu et al., 2024, p. 14348).

Hybridization also appears in architectures that pair RRT with MPC, where a CBF-constrained controller refines edges during search and an MPC tracker executes the path (Liu et al., 2024, pp. 1–4). This framework has been demonstrated in scenarios involving moving obstacles and multi-robot interactions (Liu et al., 2024, p. 9). Such systems prioritize safety over strict optimality, utilizing the barrier layer as the primary safety mechanism during both the planning and execution phases (Liu et al., 2024, pp. 1–2). A related approach in maritime navigation employs a hierarchical architecture that combines MPC in the navigation layer with CBF-based online optimization in the control layer to certify safety during port maneuvers (Otsuki et al., 2023, pp. 3138, 3140).

The above strands suggest several practical patterns. First, enforcing safety filters pointwise along the steering rollout reduces false feasibility but makes each edge extension expensive, which is especially relevant in RRT* with its many local connections during parent selection and rewiring. Second, planners that align feasibility checks with the same structures used for cost evaluation keep the local steering model and the execution controller consistent. Third, some methods adapt sampling based on information gathered from the current tree, biasing samples toward regions that appear promising. Fourth, dynamic obstacles are handled explicitly by embedding continuous-time safety filters and by pairing the planner with tracking controllers that can react to other moving agents.

Within this landscape, LQR-CBF hybrids pursue a sharper division of labor. LQR provides a cost-consistent local steerer and a value-function distance metric (Perez et al., 2012, pp. 2539–2540). The CBF layer certifies safety without forcing a QP in the inner loop. LQR-CBF-RRT* operationalizes that idea (Yang et al., 2025). An example outcome is shown in Fig. 5: the tree grown by LQR-CBF-RRT*

(green) with the final collision-free trajectory (red) in a cluttered workspace.

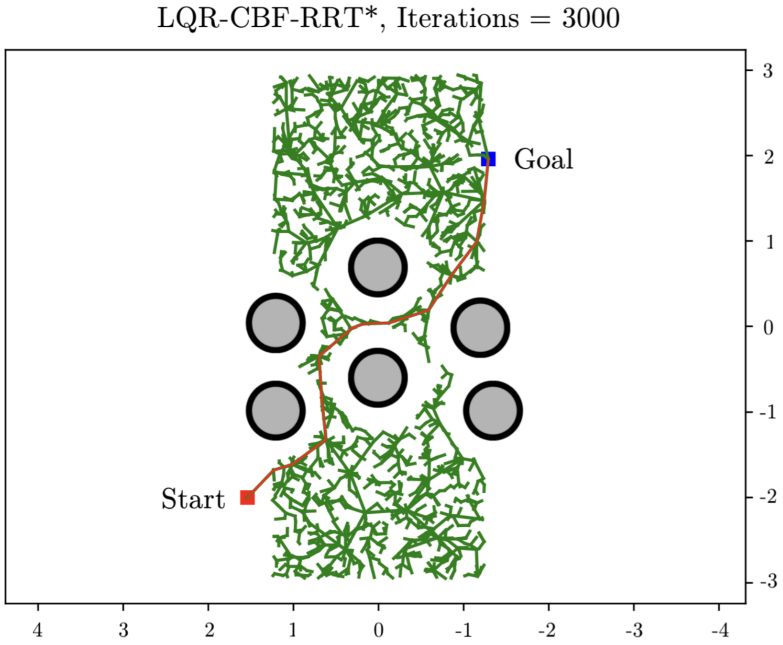

Fig. 5: LQR-CBF-RRT* simulation result: sampled trajectories (green) and the final collision-free path (red) in a cluttered workspace. Source: Reproduced from (Yang et al., 2023, p. 7).

The algorithm performs edge extension using an LQR steerer and directly verifies CBF constraints on the resulting trajectory segments, avoiding the need to solve CLF-CBF QPs at every step (Yang et al., 2025, pp. 3700–3702). If a barrier violation occurs, the extension terminates immediately, preserving the valid trajectory prefix (Yang et al., 2025, p. 3702). To improve efficiency, the planner caches optimal LQR gains in a hash table to prevent redundant linearizations during rewiring and employs cross-entropy importance sampling to accelerate convergence (Yang et al., 2025, pp. 3700, 3703). LQR maintains local consistency with the optimal control objective, while CBF checks serve as safety certificates that reject unsafe trajectories (Yang et al., 2025, p. 3700). Experiments demonstrate reduced computation time compared to QP-based baselines, and theoretical results prove asymptotic optimality within a conservative safety set (Yang et al., 2025, pp. 3703, 3704). However, dynamic obstacles and online planning were explicitly excluded from the study's scope (Yang et al., 2025, p. 3700). The next subsection reviews path planning in 2D dynamic environments to set up this case.

2.6 Path Planning in 2D Dynamic Environments

Empirical research on dynamic-obstacle avoidance leans on a small set of public trajectory corpora augmented by purpose-built synthetic worlds. Crowd-navigation work frequently relies on the ETH and UCY pedestrian datasets. These corpora contain trajectories from scenes such as university campuses and hotels, representing environments with varying pedestrian densities and trajectory linearities (Mavrogiannis et al., 2023, p. 36:15). UCY is denser and displays more non-linear motion than ETH, complicating trajectory prediction (Alahi et al., 2016, p. 966). The Stanford Drone Dataset extends this paradigm to mixed traffic, pedestrians, bicycles, skateboarders, cars, buses, and golf carts, with top-view aerial videos on a university campus (Robicquet et al., 2016a, pp. 550, 553). An example Stanford Drone Dataset aerial frame is shown in Fig. 6, illustrating mixed pedestrian-vehicle traffic from a top view. Across these corpora, trajectories are typically encoded as time-stamped positions (x_t, y_t) (Dong et al., 2025, p. 2191). World-referenced tracks are used directly in datasets like ETH-UCY and NBA, while the Stanford Drone Dataset is documented in camera coordinates (pixels) (Dong et al., 2025, p. 2194). The SiT dataset includes 3D trajectory data (x, y, z) and 3D bounding box annotations (Bae et al., 2023, pp. 3, 6).

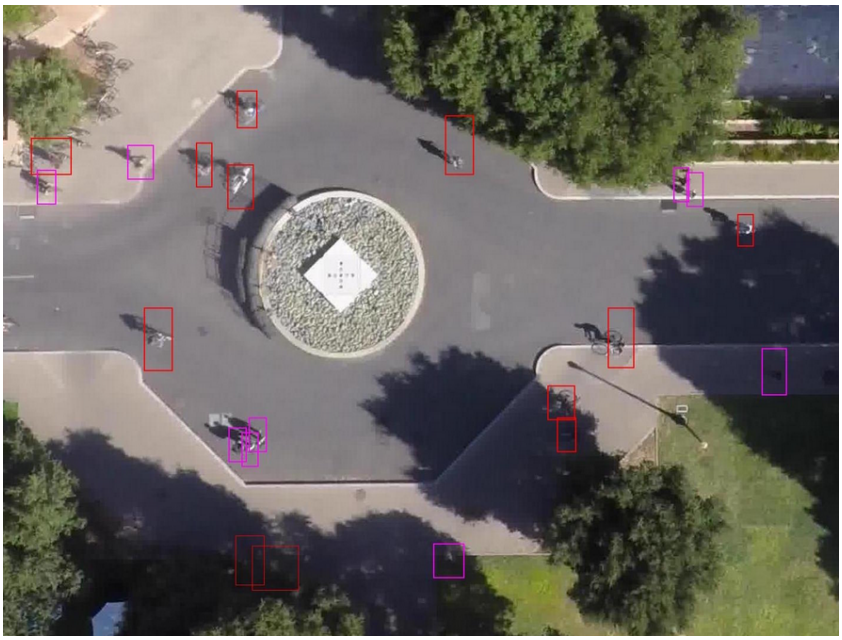

Fig. 6: Stanford Drone Dataset: example aerial frame depicting mixed agents (pedestrians, cyclists, vehicles) on a university campus. Source: Reproduced from (Robicquet et al., 2016b).

Autonomous-driving research complements these crowd datasets with recordings of traffic at intersections captured from drones. The inD dataset provides naturalistic trajectories collected at four German unsignalised intersections and is available for non-commercial research. It contains vehi-

cles, cyclists, and pedestrians recorded from a bird's-eye view with high positional accuracy (Bock et al., 2020, pp. 1929, 1933–1934). The rounD dataset focuses on roundabouts and provides georeferenced trajectories alongside Lanelet2 maps (Krajewski et al., 2020). The openDD dataset contains over 62 hours of data and high-definition maps, available under a license permitting both commercial and non-commercial use (Breuer et al., 2020, pp. 1–2). The INTERACTION dataset provides semantic lane-level maps and includes scenarios covering intersections, roundabouts, and merging maneuvers across multiple countries (Zhan et al., 2019). Fig. 7 highlights INTERACTION's scenario diversity (merges, roundabouts, unsignalized intersections).

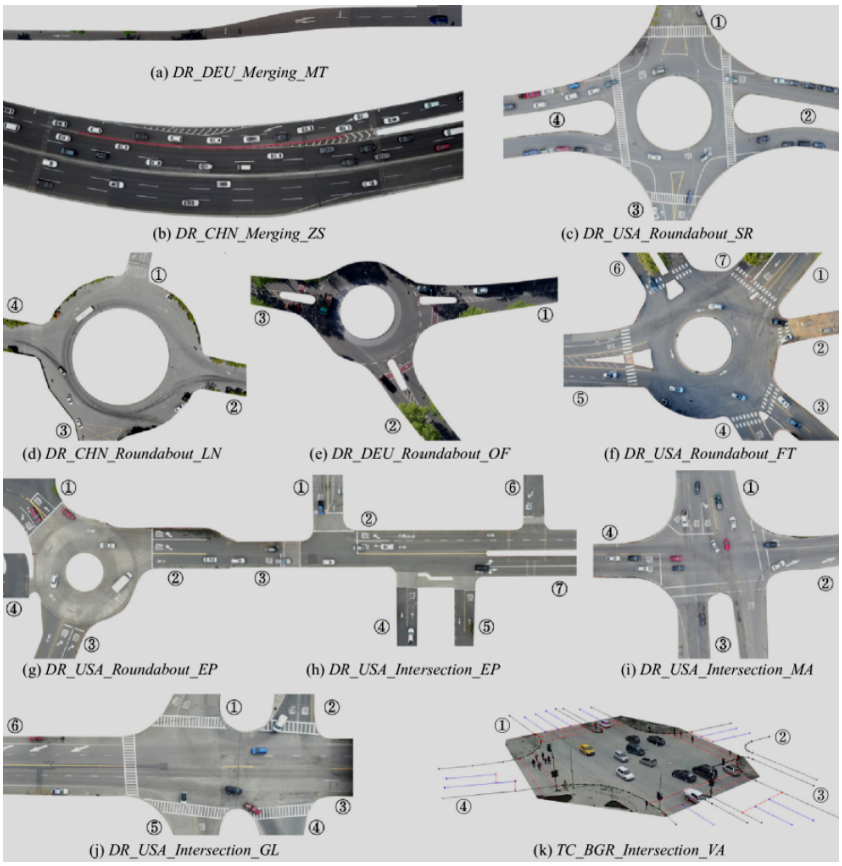

Fig. 7: INTERACTION dataset: diverse interactive driving scenarios recorded by drones/traffic cameras (merges, roundabouts, unsignalized intersections). Source: Reproduced from (Zhan et al., 2019, p. 4).

In practice, these drone sets are often accessed via open-source Python loaders for importing and visualising trajectories, which simplifies replay and batch Monte Carlo evaluation (ika RWTH Aachen University, 2024).

Alongside replayed recordings, synthetic 2D dynamic worlds are used to isolate planner behaviour under controllable traffic patterns. Two design choices recur. First, time-augmented configuration spaces are used so that planning occurs in space-time. For instance, the open motion planning li-

brary (OMPL) provides an implementation of space-time-RRT* (Grothe et al., 2022, pp. 3314–3315). Second, safe-interval scheduling is used for grid or roadmap baselines: Safe-interval path planning represents motion by pairing configurations with safe time intervals (Phillips & Likhachev, 2011, p. 5628). This interval-based abstraction extends to recent multi-objective (Ren et al., 2022, p. 8154) and real-time (Wild Thomas et al., 2024, p. 161) variants. When dense crowds are required beyond recorded data, reciprocal-velocity simulators like RVO2 generate collision-avoiding multi-agent flows. The Python-RVO2 library makes this functionality accessible from Python, allowing simulation runs to be logged as time-stamped agent positions for planner reuse (Stüvel, 2020).

Within this ecosystem, CBF studies targeting moving agents typically use small 2D scenes with explicitly time-varying safety sets. TV-CBFs combined with differentiable optimisation have been demonstrated for dynamic-obstacle avoidance and compared against an MPC-based method in planar tasks (Dai et al., 2025, pp. 1, 5). Exponential and high-order CBFs address the higher relative-degree constraints inherent in vehicle models, maintaining an affine safety constraint on the control input that can be enforced via a per-step QP (Nguyen & Sreenath, 2016, pp. 1, 5); (Ames et al., 2019, p. 3425). For multi-vehicle coordination, representative studies model unsignalised four-way intersections in the plane and apply barrier functions for safety. Notable examples include future-focused CBF controllers validated with ground rovers at a four-way crossing (Black et al., 2023) and barrier-certified optimal coordination frameworks for connected vehicles at signal-free intersections (Chalaki & Malikopoulos, 2022). These works specify planar dynamics and controller parameters, reporting metrics such as success rates, QP feasibility, and computation time.

Hybrid sampling-and-barrier planners can use the same planar scaffolding. The hybrid LQR-CBF planners from Section 2.5, together with CBF- and CLF-CBF-augmented RRT* variants in the literature, can be evaluated in 2D, time-indexed scenes where the barrier constraints used for tree expansion match those used for feedback control; this alignment makes roll-outs reproducible over recorded or synthetic (x, y, t) trajectories. The LQR-CBF-RRT* study by Yang et al. (2025) fits this pattern: it operates in planar, static environments and reports offline planning together with a hardware validation, placing the hybrid in the same abstract 2D setting as the datasets and synthetic benchmarks described above, but without time-varying safety sets (Yang et al., 2025). Taken together, these hybrids and the surrounding ecosystem of time-indexed trajectory datasets, space-time sampling planners, safe-interval path-planning baselines, and CBF controllers in dynamic scenes define a shared 2D testbed with established metrics such as time-indexed collision checks, wall-clock planning time, and path-cost comparisons.

2.7 Gap Analysis

Sections above established three ingredients:

1. asymptotically optimal sampling (RRT* and informed/sparse variants),
2. LQR steering that supplies a cost-consistent local model,
3. and CBFs that enforce forward invariance in sampled time.

Hybrids that combine these elements exist, including CBF-RRT* with per-step QPs or compatibility checks, and an LQR-CBF-RRT* that verifies barrier conditions along LQR rollouts in static scenes. What is missing is not another variant but evidence that the specific hybrid LQR-CBF-RRT* holds up in dynamic environments.

In the PRISMA-screened corpus for this review (see Fig. 1), no study that couples LQR steering with CBF safety inside an asymptotically optimal sampler under moving obstacles advanced to the final inclusion stage. The eligibility assessment left the corpus empty. Published work on LQR-CBF-RRT* is confined to static, offline planning; dynamic obstacles and online planning are out of scope, and evaluation follows an offline pipeline with static clutter and a separate tracker (Yang et al., 2025, p. 3700). Secondary sources reproduce or compare against that algorithm only in static layouts, reporting cost and runtime trade-offs but no moving-obstacle trials (Chu et al., 2025). Parallel CBF-RRT* studies demonstrate feasibility and hardware validation in stationary environments (for example, bipedal robots in rooms with randomly placed obstacles) but do not couple LQR steering with RRT* optimality under motion (Peng et al., 2023). Taken together, the literature reports no instance of LQR-CBF-RRT* applied and tested in dynamic environments.

The surrounding literature on 2D, time-indexed datasets, synthetic space-time benchmarks, safe-interval baselines, and CBF controllers in planar scenes shows that simple planar testbeds already serve as a common reference for dynamic-obstacle studies. The original LQR-CBF-RRT* evaluation fits into this same abstraction, using a planar static workspace with LQR steering and barrier checks along local rollouts. Extending that setup to moving obstacles in time-indexed (x, y, t) scenes keeps the cost function, steering law, and modelling assumptions aligned with the published work, while exposing the missing dynamic case instead of introducing a new scenario with extra design choices.

This gap matters because safety certificates become time-indexed when obstacles move. A path that is admissible in a frozen map can still intersect a moving agent unless barrier conditions are checked along each edge in time. RRT* concentrates compute in parent selection and tree rewiring; adding barrier checks during those stages can shift median planning time and path cost. Barrier tuning that is

benign in static scenes can alter success rates and minimum signed distance once obstacle velocity and sampling period interact. The next section addresses this gap by extending LQR-CBF-RRT* to moving obstacles in 2D, time-indexed environments and reporting the resulting safety, performance, and parameter-sensitivity outcomes.

3 Research Methodology

This chapter specifies the methodological ground on which the evidence in Section 4 rests. It fixes notation and modeling assumptions, states the safety-critical planning problem, and describes the components of the LQR-CBF-RRT* planner with state and input constraints, obstacle kinematics, the time-varying safe set, the steering law used for local connections, the per-step CBF filter used to enforce safety during expansion, and the search/rewiring rules that define the tree. Design choices are tied to three research questions that ask whether the planner maintains a collision frequency below a prespecified target under bounded obstacle speeds, what additional wall clock planning time and geometric path cost are incurred relative to an LQR-RRT* baseline evaluated on identical scenes, and how sensitive the outcomes are to the control barrier gain, the nominal expansion speed, and the extension length. The chapter also fixes the experimental design by specifying scenario definitions, measurement procedures, and statistical conventions established a priori so that subsequent results can be interpreted without further methodological assumptions.

The guiding principles are:

1. Formal safety constraints are evaluated at discrete planning instants with absolute timestamps.
2. Feasibility is enforced lexicographically before optimality.
3. Comparisons to the baseline are paired whenever possible to remove scene-induced variance.
4. Statistical conventions are fixed in advance and applied consistently.

Assumptions are explicit and minimal: a single-integrator motion model for steering and propagation, disc-shaped dynamic obstacles with constant-velocity predictions over short horizons, and box-bounded inputs. Where such simplifications create mismatch to obstacles or higher-order dynamics, limitations are flagged and carried into Section 4.

The structure proceeds as follows. Section 3.1 lays out definitions and sets used notations. Section 3.2 formulates the problem by specifying the workspace, the discrete-time dynamics with input bounds, the time-varying safe set, and the feasibility-first objective. Section 3.3 then states the dynamic-obstacle model, including disc kinematics, the short prediction horizon tied to absolute time, and the barrier functions used to define safety. Section 3.4 describes the algorithmic composition of nominal LQR steering, the per-step CBF safety filter, and RRT* expansion with rewiring. Finally, Section 3.5 details the experimental design, including scenarios, metrics, decision rules, the pairing protocol for baseline comparisons, and the statistical conventions referenced in Section 4.

3.1 Definitions and Notation

All symbols follow the List of Symbols (Section b). The table serves as the single authoritative reference for symbols in this chapter. Only conventions not listed there are fixed here. Vectors are treated as column vectors in $\mathbb{R}^2$, and $\|\cdot\|_2$ denotes the Euclidean norm. Inequalities between vectors, such as $u_{\min} \le u \le u_{\max}$, are interpreted componentwise. The transpose is written $(\cdot)^\mathsf{T}$, and the symbols $\succeq$ and $\succ$ indicate positive semidefiniteness and positive definiteness, respectively. The identity matrix in two dimensions is I_2. Unless stated otherwise, all quantities are defined on the discrete time grid $t_k = t_0 + k\Delta t$ with sampling period $\Delta t > 0$.

Conventions specific to edge construction, timestamps, acceptance, and rewiring are defined where first used in Section 3.4. Statistical notation and decision rules are defined once in Section 3.5 and applied in Section 4. Having established the notation, the next subsection formalizes the safety-critical planning problem. It specifies the workspace and goal set, the discrete-time dynamics and input bounds.

3.2 Problem Formulation

Planning takes place in a bounded planar workspace $\mathcal{W} \subset \mathbb{R}^2$. The robot state is the position $p_k \in \mathcal{W}$ at discrete time $t_k = t_0 + k\Delta t$ with sample-and-hold period $\Delta t > 0$. The motion model is the discrete-time single integrator

$$p_{k+1} = p_k + \Delta t\, u_k, \qquad u_k \in \mathcal{U} := \{u \in \mathbb{R}^2 \mid u_{\min} \le u \le u_{\max}\}, \tag{1}$$

where $u_{\min}, u_{\max} \in \mathbb{R}^2$ are applied componentwise.

The initial condition is $p_0 \in \mathcal{W}$. The goal set is the closed ball

$$\mathcal{G} = \{p \in \mathcal{W} \mid \|p - p_g\|_2 \le \varepsilon_{\text{goal}}\}, \tag{2}$$

with center $p_g \in \mathcal{W}$ and radius $\varepsilon_{\text{goal}} > 0$.

Let $\mathcal{O}(t)$ denote the obstacle set at time t. For each obstacle $i \in \{1, \dots, M\}$ with center $c_i(t)$ and radius

r_i, define the per-obstacle barrier

$$h_i(p,t) := \|p - c_i(t)\|_2^2 - r_i^2, \tag{3}$$

$$h_{\min}(p,t) := \min_{1\le i\le M} h_i(p,t). \tag{4}$$

The admissible (safe) set is

$$\mathcal{S}(t) = \{p \in \mathcal{W} \mid h_{\min}(p,t) \ge 0\}, \tag{5}$$

and a collision occurs whenever $h_{\min}(p,t) < 0$ as shown by Ames et al. (2017).

The objective is to find a finite horizon $N \in \mathbb{N}$, a state sequence $p_{0:N}$, and an input sequence $u_{0:N-1}$ that reach the goal while respecting the system constraints at every step. Feasibility is treated lexicographically before optimality. Only trajectories that satisfy the dynamics, the input bounds, and the safety condition are admissible. Among feasible trajectories, the tree minimizes a geometric-control surrogate, given by the edge length plus $\|u\|$.

The quadratic expression

$$J = \sum_{k=0}^{N-1} \left(x_k^\top Q\, x_k + u_k^\top R\, u_k\right), \qquad x_k := p_k - p_k^\star, \tag{6}$$

with $Q \succeq 0$ and $R \succ 0$ following Perez et al. (2012) is used to design the LQR steering in Section 3.4.3. Subject to

$$\begin{aligned} &p_{k+1} = p_k + \Delta t\, u_k, \quad u_k \in \mathcal{U}, \quad p_k \in \mathcal{S}(t_k) \quad \forall k \in \{0,\dots,N-1\}, \\ &p_0 \text{ given}, \quad p_N \in \mathcal{G}. \end{aligned} \tag{7}$$

The planner returns an RRT* path as a sequence of time-stamped nodes with parent-child edges realizing the feasible state-control pairs along the selected route, together with a control log obtained by concatenating the per-edge control sequences u_k for every accepted edge segment. To render the safety constraint evaluable for moving obstacles, the next subsection specifies the dynamic-obstacle model and the short-horizon predictions used during planning.

3.3 Dynamic-obstacle Model and Assumptions

Dynamic obstacles are modeled as moving discs to reduce computational cost during online prediction and safety evaluation. Each dynamic obstacle i is characterized by its initial center $c_i^0 \in \mathbb{R}^2$, a radius

$r_i > 0$, and a constant-velocity vector $v_i \in \mathbb{R}^2$ satisfying $\|v_i\|_2 \le v_{\max}$. Under this model the obstacle center evolves according to

$$c_i(t) = c_i^0 + v_i t. \tag{8}$$

The time-varying obstacle set is the union of discs

$$\mathcal{O}(t) = \bigcup_{i=1}^{M} \{\, p \in \mathcal{W} \mid \|p - c_i(t)\|_2 \le r_i + \delta_i \},\tag{9}$$

where $\delta_i \ge 0$ is a fixed isotropic inflation margin that aggregates model error and inter-sample motion. Runtime CBF constraints use the un-inflated barrier h_i below. The inflation δ_i is used only in conservative prechecks.

Predictions use a short per-edge horizon that is tied to absolute time. For a candidate edge that starts at time t and has length ℓ_{edge}, obstacle centers are evaluated for times in $[t, t + \ell_{\text{edge}}/v_{\text{nom}}]$. The predictor is recomputed at every planning tick. The model does not forecast beyond the duration of the candidate edge. Obstacle motion is defined in $\mathbb{R}^2$, and obstacles may leave $\mathcal{W}$. The robot is treated as a point. Safety compares p with the disc obstacles via the barrier functions $h_i(p,t)$. A collision is declared whenever $h_{\min}(p,t) < 0$.

With obstacle kinematics and barrier functions defined, the following subsection details how LQR steering, CBF filtering, and RRT* search are combined into a single planning procedure.

3.4 Algorithm Design: LQR-CBF-RRT*

The algorithm design follows the implementation of Yang et al. (2025). The planner couples nominal LQR steering with online safety filtering and RRT* graph search with rewiring. For a sampled target in $\mathcal{W}$, the nearest tree node proposes a nominal edge via LQR steering. Along that edge, a CBF safety filter solves a quadratic program at each discrete step to minimally adjust the nominal input while enforcing safety and input bounds. The edge is accepted only if each per-step CBF check is feasible and the predicted samples remain in $\mathcal{S}(t)$. Standard RRT* rewiring then reduces accumulated cost. The barriers $h_i(p,t)$ are as defined above. Numerical settings appear in Section 3.4.4.

3.4.1 CBF Safety Filter

For each dynamic obstacle, use the time-varying barrier in Eq. (4). Here $h_i(p,t) \geq 0$ encodes disc clearance, and the safe set $\mathcal{S}(t)$ follows from Eq. (5). The CBF imposes the relative-degree-one constraint

$$\dot{h}_i(p,t) + \alpha(h_i(p,t)) \geq 0, \qquad \alpha(h) = k_{\text{cbf}}\, h^{p_{\text{cbf}}}, \tag{10}$$

with α strictly increasing, $k_{\text{cbf}} > 0$, and $p_{\text{cbf}} \geq 1$. Increasing k_{cbf} tightens the decay requirement uniformly across the safe set. Choosing $p_{\text{cbf}} > 1$ weakens the constraint in a neighborhood of the boundary $h = 0$ and strengthens it away from the boundary. Even exponents should be avoided because $h^{p_{\text{cbf}}} > 0$ also for $h < 0$. The implementation therefore uses $p_{\text{cbf}} = 1$ (linear α) in all experiments.

In implementation, $\dot{h}_i$ is evaluated with predicted obstacle motion $c_i(t)$ and velocities v_i, and the inequality is enforced via a small QP when enabled.

Differentiating at step k gives

$$\begin{aligned} \dot{h}_i(p_k,t_k) &= \nabla_p h_i(p_k,t_k)^\top \dot{p}_k + \partial_t h_i(p_k,t_k) \\ &= 2\left[p_k - c_i(t_k)\right]^\top (u_k - v_i), \end{aligned} \tag{11}$$

where ∇_p denotes the gradient with respect to p, ∂_t the partial derivative with respect to t (holding p fixed), and an overdot denotes the total derivative w.r.t. time along the candidate trajectory. Specifically,

$$\nabla_p h_i(p_k,t_k) = 2\left[p_k - c_i(t_k)\right], \tag{12}$$

$$\partial_t h_i(p_k,t_k) = -2\left[p_k - c_i(t_k)\right]^\top v_i, \tag{13}$$

$$\dot{p}_k = u_k. \tag{14}$$

At step k along a proposed edge, solve

$$\min_{u_k} \|u_k - u_{\text{nom},k}\|_2^2 \tag{15}$$

$$\text{such that } 2\left[p_k - c_i(t_k)\right]^\top (u_k - v_i) + k_{\text{cbf}}\, h_i(p_k,t_k)^{p_{\text{cbf}}} \geq 0, \tag{16}$$

$$\forall i,\ u_k \in \mathcal{U}.$$

The corresponding implementation of the predicted obstacle position and relative-velocity CBF constraint used in the QP is shown in Listing 1.

```
                # Predict obstacle position at current time
                x_obs_t = x_obs_0 + vx * current_time
                y_obs_t = y_obs_0 + vy * current_time

                h = (x1 - x_obs_t) ** 2 + (x2 - y_obs_t) ** 2 - r ** 2

                # CBF constraint accounting for relative motion
                # The derivative includes both robot motion and obstacle motion
                lgh = (2 * (x1 - x_obs_t) * (self.u1 - vx) +
                    2 * (x2 - y_obs_t) * (self.u2 - vy))

                self.m.addConstr((lgh + self.k_cbf * h**self.p_cbf) >= 0)
```

Listing 1: CBF QP: dynamic obstacle prediction and relative-velocity constraint

This quadratic program is convex because the objective is quadratic and each CBF constraint is affine in u. It is solved once per planner step at time t_k for all obstacles, which provides a per-step enforcement of the continuous-time condition. For small Δt this approximates forward invariance of $\mathcal{S}(t)$. The edge is accepted only if all per-step tests succeed and the QP is feasible at every step; otherwise it is rejected. Having established per-step safety via the CBF filter, the next subsection details how candidate edges are incorporated into the tree through RRT* target sampling and cost-reducing rewiring.

3.4.2 Sampling and Rewiring

Tree expansion follows the standard RRT* procedure, where the planner samples a target, applies LQR steering, assigns a timestamp to the proposed edge, and rewires only if the change reduces cost while preserving safety. Sampling draws the local target $p_k^\star$ uniformly from $\mathcal{W}$ with a fixed goal bias. Although LQR-based tracking energy

$$V_k = x_k^\top P\, x_k \tag{17}$$

is often used as the nearest-neighbor metric in LQR-RRT*, here the Euclidean distance is used for nearest-neighbor queries for simplicity, while retaining LQR for steering toward $p_k^\star$. The planner connects p_k toward $p_k^\star$ with an LQR step whose geometric extension is capped by `step_len`, producing samples $\{p_j\}_{j=-1}^{m}$ with $p_0 = p_k$ and $p_m = p_{k+1}$. These samples follow the single-integrator propagation

$p_{j+1} = p_j + \Delta t\, u_{\text{nom},j}$. The nominal arc length is

$$\ell_{\text{edge}} = \sum_{j=0}^{m-1} \left\| p_{j+1} - p_j \right\|_2, \tag{18}$$

and since

$$p_{j+1} - p_j = \Delta t\, u_{\text{nom},j}, \tag{19}$$

it can be written equivalently as

$$\ell_{\text{edge}} = \sum_{j=0}^{m-1} \Delta t \left\| u_{\text{nom},j} \right\|_2, \tag{20}$$

i.e., a Riemann sum of the nominal speed. The candidate child's timestamp uses the nominal traversal time at speed v_{nom},

$$t_{\text{new}} = t_{\text{near}} + \frac{\ell_{\text{edge}}}{v_{\text{nom}}}, \tag{21}$$

which aligns absolute times for evaluating predicted obstacle centers and the CBF constraints. The core expansion with the time update and the predicted-collision guard is given in Listing 2.

```
            dist_to_new = math.hypot(node_rand.x - node_near.x, node_rand.y - node_near.y)
            time_to_reach = min(dist_to_new, self.step_len) / self.nominal_velocity
            node_rand.time = node_near.time + time_to_reach

            node_new = self.LQR_steer(node_near, node_rand)

            if k % 100 == 0:
                print(f"Iteration: {k}, Time: {self.planning_time:.2f}s")

            if node_new and not self.utils.is_collision_with_dynamic_predicted(node_near,
                ↪ node_new, self.initial_dynamic_obs):
                neighbor_index = self.find_near_neighbor(node_new)
                self.vertex.append(node_new)

                if neighbor_index:
                    self.LQR_choose_parent(node_new, neighbor_index)
```

Listing 2: Expansion step with time update and predicted-collision guard

Rewiring reassigns any neighbor whose total cost strictly decreases via p_{new}, provided the new edge passes the same per-step CBF filter. With these sampling and acceptance rules in place, the next subsection specifies the LQR steering metric.

3.4.3 LQR Steering Metric

Since the sampling step selects a local connect target and timestamps rely on nominal propagation, the steering metric specifies the nominal control used to generate candidate edges. For nearest-neighbor queries the Euclidean distance is used, while rewiring uses a path cost built from geometric increments and control effort. The local model is the discrete single integrator

$$A = I_2, \quad B = \Delta t\, I_2 \tag{22}$$

with sample period Δt. Let

$$x_k := p_k - p_k^\star \tag{23}$$

for the connect target $p_k^\star$. With $Q \succeq 0$ and $R \succ 0$, the discrete LQR gain K is computed from

$$K = (R + B^\mathsf{T} P B)^{-1} B^\mathsf{T} P A, \tag{24}$$
$$\text{with } P = A^\mathsf{T} P A - A^\mathsf{T} P B\,(R + B^\mathsf{T} P B)^{-1} B^\mathsf{T} P A + Q.$$

following Perez et al. (2012).

Here P is the stabilizing solution of the discrete algebraic Riccati equation, adhering to Laub (1979), and the quadratic function from Eq. (17) serves as a local Lyapunov metric. The nominal input is

$$u_{\text{nom},k} = -K\, x_k, \tag{25}$$

which yields the closed-loop decrease

$$V_{k+1} - V_k = -x_k^\mathsf{T}\big(Q + K^\mathsf{T} R K\big)\, x_k \le 0 \tag{26}$$

under the unconstrained model with

$$p_{k+1} = p_k + \Delta t\, u_{\text{nom},k}. \tag{27}$$

The solver implementation used to compute K in the code (`dlqr`) is given in Listing 3.

```
    def dlqr(self, A, B, Q, R):
        """
␣␣␣␣␣␣␣␣Solve␣the␣discrete␣time␣lqr␣controller.
␣␣␣␣␣␣␣␣x[k+1]␣=␣A␣x[k]␣+␣B␣u[k]
```

```
        cost = sum x[k].T*Q*x[k] + u[k].T*R*u[k]
        """
        # first, solve the ricatti equation
        P = np.matrix(scipy.linalg.solve_discrete_are(A, B, Q, R))
        # compute the LQR gain
        K = np.matrix(scipy.linalg.inv(B.T * P * B + R) * (B.T * P * A))

        eigVals, eigVecs = scipy.linalg.eig(A - B * K)

        return -K
```

Listing 3: Discrete LQR solver `dlqr` used by the steering metric

In the planner, this nominal propagation generates a finite sequence toward an intermediate target placed at distance $\min(\texttt{step\_len}, \text{dist})$ and stops when the distance to that target is below the tolerance. Parent selection and rewiring then use a path cost composed of geometric increments and the sum of control magnitudes.

3.4.4 Implementation Details

This implementation builds upon Yang et al. (2025) open-source baseline `LQR_CBF_rrtStar`[2]. The repository used here is the thesis author's fork (branch `dyn-tvcbf`)[3], which extends the baseline from static to dynamic environments with a predictive CBF QP, animation, and structured result extraction. The implementation separates the code into explicit modules and the data flow follows that decomposition.

The following modules were implemented or adjusted from the baseline code to function in dynamic environments:

- environment description
- nominal steering
- safety filtering
- graph search
- experiment drivers

[2] `https://github.com/mingyucai/LQR_CBF_rrtStar`
[3] `https://github.com/Leg0shii/LQR_CBF_rrtStar/tree/dyn-tvcbf`

- visualization

The planner core `LQR_CBF_rrtStar_linear.py` defines the tree structure (`Node`), the RRT* expansion and rewiring loop, and holds references to the environment, LQR steering, and utility routines. It returns the final path and the concatenated open-loop control sequence after a run. The nominal steering module `LQR_planning.py` implements the discrete single-integrator LQR, generates the reference input sequence, and optionally applies the safety layer at each integration step. The sampling period and the stopping tests are set in this module. The safety layer `CBFsteer.py` constructs a small quadratic program that minimally perturbs the nominal input while enforcing per-obstacle barrier inequalities, and it includes a variant that predicts dynamic obstacle positions at the requested time index. The experiment harness `experiments.py` creates randomized dynamic-obstacle fields, runs multiple roll-outs, logs progress, and persists collision trajectories and summaries.
The `replay_experiment.py` script reloads and animates any stored collision case. The plotting utilities render the workspace, the tree, and the paths, and they provide an animation utility for dynamic scenes. When launched, `replay_experiment.py` enumerates saved collision roll-outs and prompts for selection. Fig. 8 shows the command line interface. After a file is chosen, the tool prints a structured summary with positions, obstacle parameters, and environment metadata. This can be found in Appendix A.2 Fig. 15.

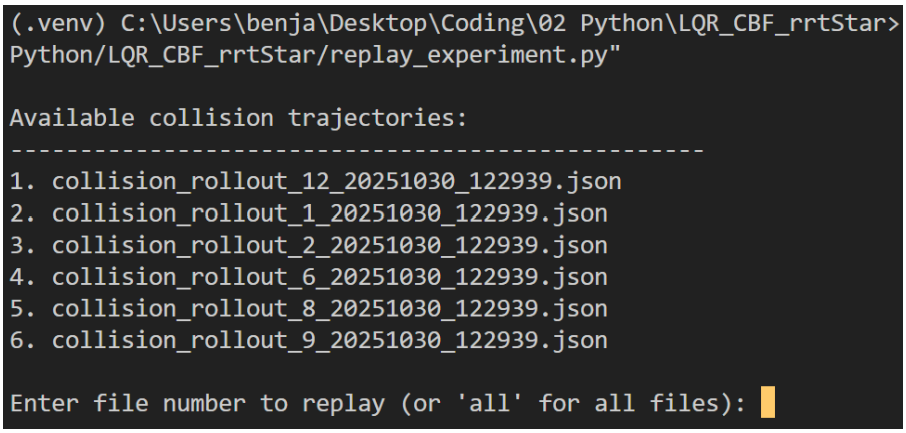

Fig. 8: Replay utility: Command line interface file selection for saved collision roll-outs under `collision_trajectories/`. Source: Own illustration.

Linear algebra uses NumPy and SciPy in the LQR routines, which solve the discrete algebraic Riccati equation once for the fixed single-integrator model and then step the closed loop at Δt. The CBF safety filter formulates a convex quadratic objective with affine barrier constraints and solves it with Gurobi. Console output is suppressed in the experiment harness to avoid input/output (I/O) overhead. If the dynamic-constraints QP is not optimal or infeasible, the planner terminates the nominal rollout

and rejects the edge rather than falling back to $u_{\text{nom},k}$. Random sampling for targets and obstacle generation uses Python's `random` and NumPy's random number generators (RNGs). Seeds are not fixed by default in the driver, so bit-for-bit reproduction requires setting both RNG seeds prior to roll-outs. All computations use default double precision. No mixed-precision or GPU kernels are employed.

Tab. 1 lists the parameters used by the experiment drivers. The sampling period Δt and the geometric extension cap `step_len` bound the number of nominal and CBF updates along each attempted edge. The nearest-neighbour choice only selects the source node. The nominal speed v_{nom} timestamps edges for time-indexed barrier evaluation and is swept in the parameter study. The discrete LQR weights Q and R remain fixed for the single-integrator steering. The experiment harness varies obstacle count, sizes, and velocities within the workspace bounds before each roll-out, and it rejects goals that would be covered by a moving obstacle.

Tab. 1: Parameters used in experiments. Source: Own illustration.

Quantity	Value(s)	Source
Sampling period Δt	$0.05\,\text{s}$	Section 3.4.3 and code
Nearest-neighbour metric	Euclidean distance	Section 3.4.3 and planner code
Geometric extension cap `step_len`	$\{6, 12\}\,\text{m}$	Experiment driver (arguments)
Nominal speed v_{nom}	$\{1.5, 3.0, 5.0\}\,\text{m s}^{-1}$	Experiment driver (arguments)
Goal-sample rate	0.10	Planner init `goal_sample_rate`
Search radius	$20\,\text{m}$	Planner init `search_radius`
Max iterations	1500	Experiment driver default
Workspace bounds	$x \in [5, 45]\text{m}$, $y \in [5, 25]\,\text{m}$	Experiment checks
Dynamic obstacles	$3 - 10$ discs, $r \in [1.5, 2.5]\,\text{m}$	Experiment driver
Obstacle speeds	$\|v_i\|_2 \leq v_{\text{max}}$, $v_{\text{max}} \leq 1.0\,\text{m s}^{-1}$	Driver argument and model
QP solver	Gurobi, quiet mode	Driver setup and CBF module

Performance engineering focuses on keeping per-step costs proportional to the number of nearby obstacles and to the discretization used within each extension. The planner assembles all active obstacle constraints into one QP per step along an edge, which avoids per-obstacle solver calls. Logging is configured once at process start and writes a timestamped log file. Gurobi output is disabled globally to prevent per-solve console traffic and I/O overhead. The dynamic obstacle predictor is algebraic and limited to the current edge duration, so no horizon-dependent memory is accumulated

across steps in a single extension. The main cost drivers are nearest-neighbor searches, LQR propagation with collision checks, and CBF QP solves. These paths are implemented with NumPy and the external QP solver without additional layers.

Reproducibility relies on file-backed artifacts produced by the driver. Each collision roll-out is serialized to a JSON file under `collision_trajectories/` with the start and goal states, all dynamic obstacles, the time-stamped robot path, and fields for the collision time, the obstacle index, and the signed distance. A run-level summary JSON records the list of collisions. The replay utility enumerates these files, prints human-readable diagnostics, and can optionally export an animation for any selected case, which supports qualitative auditing of failures. For the performance-cost study, the driver writes a comma-separated values (CSV) file of per-trial timings and costs so that downstream analysis scripts can compute medians and overheads without re-running the planner. Package and solver versions depend on the executing environment. Listing 4 shows a minimal segment of the saved payload. See Appendix A.1 Listing 9 for the complete JSON schema.

```
      "dynamic_obstacles": [
          {
              "initial_x": obs[0],
              "initial_y": obs[1],
              "radius": obs[2],
              "velocity_x": obs[3],
              "velocity_y": obs[4],
              "speed": math.sqrt(obs[3]**2 + obs[4]**2)
          } for obs in dynamic_obstacles
      ],
      "robot_trajectory": [
          {
              "x": point[0],
              "y": point[1],
              "time": point[2] if len(point) >= 3 else 0,
              "waypoint_idx": i
          } for i, point in enumerate(path[::-1]) # Reverse to get start-to-goal order
      ],
```

Listing 4: Minimal JSON payload saved for collision replays

3.5 Experimental Design

This subsection specifies scenarios, metrics, and analysis procedures used to answer RQ 1, RQ 2 and RQ 3. RQ 1 tests whether, under bounded obstacle speeds, the planner's failure rate (no path or on-path collision) stays below the $1\,\%$ target, judged by the $95\,\%$ Wilson upper endpoint. RQ 2 collects per-rollout wall-clock planning time and path cost for the CBF-enabled planner and a baseline LQR-RRT* with the same steering and sampling, and reports median overheads. RQ 3 assesses how the CBF gain k_{cbf}, the nominal velocity v_{nom}, and the extension cap `step_len` affect success (path found) and minimum signed clearance with the planner and workspace bounds held fixed, identifying hyperparameter regions that maintain clearance and reachability.

3.5.1 Benchmark Scenarios

All scenarios use the workspace, obstacle ranges, and planner constants in Tab. 2. The start and goal are placed in the workspace $\mathcal{W} = [5, 45] \times [5, 25]\,\text{m}$ inside a $50 \times 30\,\text{m}$ simulator box with $1\,\text{m}$ boundary walls. At each roll-out, $M \sim \text{Unif}\{3, \ldots, 10\}$ disc obstacles are placed with initial centers $c_i^0 \in [15, 35] \times [12, 18]\,\text{m}$ and radii $r_i \sim \text{Unif}[1.5, 2.5]\,\text{m}$, enforcing pairwise spacing $\|c_i^0 - c_j^0\|_2 \geq r_i + r_j + 2.0\,\text{m}$. Each obstacle receives a velocity vector v_i with random direction and speed $\|v_i\|_2 \sim \text{Unif}[0.2, v_{\text{max}}]\,\text{m s}^{-1}$. Prediction uses constant-velocity kinematics $c_i(t) = c_i^0 + v_i t$.

Start and goal are sampled independent and identically distributed from $\mathcal{W}$ subject to (i) a static feasibility margin of $2.0\,\text{m}$ from any initial disc, (ii) a minimum start-goal distance of $15\,\text{m}$, and (iii) a dynamic goal-accessibility check: for $t \in [0, 100)\,\text{s}$ on a $1\,\text{s}$ grid, no predicted obstacle disc of radius $r_i + 3.0\,\text{m}$ may cover the goal.[4] The robot is modeled as a point. The returned path is validated against dynamic discs by interpolation at approximately $0.05\,\text{m}$ arc-length resolution and a collision is declared when the center distance falls below r_i.

Planner constants are held fixed across scenarios unless stated:

[4] If a valid configuration is not found within 50 attempts, the roll-out is skipped and excluded from the number of completed roll-outs used in the Wilson interval.

Tab. 2: Fixed planner constants. Source: Own illustration.

Quantity	Symbol / Key	Value
Goal-sample rate	`goal_sample_rate`	0.10
Search radius	`search_radius`	20 m
Max iterations	`iter_max`	1500
LQR integrator step	Δt	0.05 s
Goal tolerance	$\varepsilon_{\text{goal}}$	0.1 m

The extension cap `step_len` is varied only for RQ 3. RQ 1 uses a low-speed regime with $v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$ and 3000 attempted roll-outs. RQ 2 uses paired roll-outs with and without the CBF QP controller on identical stochastic scenes by resetting Python and NumPy RNGs to seed 42 before each run so that obstacle sets, starts, and goals remain identical. RQ 3 sweeps the CBF gain k_{cbf}, the nominal speed v_{nom} used by the steering, and `step_len` while the workspace $\mathcal{W}$ remains fixed.

The outputs recorded in these benchmark scenarios form the basis for quantitative evaluation. Each roll-out records planning time, path cost, success flag, and collision status.

3.5.2 Evaluation Metrics

Safety for RQ 1 is quantified by dynamic collisions and by clearance, using the per-obstacle barrier in Eq. (4) and its minimum aggregator in Eq. (5). A collision is recorded when $h_{\min}(p,t) < 0$ along the time-stamped path under the predicted obstacle motion $c_i(t) = c_i^0 + v_i t$. Trials that fail to produce a path within the iteration budget are labeled no-path and reported separately. A composite safety-or-reachability failure rate counts either collision or no-path as a failure and uses all completed roll-outs. For both the collision proportion and the composite failure proportion, the study reports the two-sided 95 % Wilson interval and, when comparing against a safety budget, uses the upper endpoint of that interval as the reported upper confidence limit. For RQ 1, the decision rule uses 3000 roll-outs and the Wilson interval defined below: the 1 % safety target is judged satisfied when the upper endpoint of the two-sided 95 % Wilson interval on the collision proportion does not exceed 0.01. With 3000 completed roll-outs this requirement is met for up to 19 observed collisions; the corresponding upper endpoints are approximately 0.0013 for 0 collisions, 0.00613 for 10 collisions, 0.00987 for 19 collisions, and 0.01028 for 20 collisions. The distribution of the minimum signed distance along collision-free

paths is also reported to expose near misses.

The Wilson interval is used for binomial proportions because coverage probability remains reliable for rare events and for moderate sample sizes (Brown et al., 2001, pp. 101–102). The undercoverage associated with the normal Wald interval near zero, where the collision rate is expected to lie, is avoided (Brown et al., 2001, p. 106).

Performance for RQ 2 is quantified by wall-clock planning time and by geometric path length. Wall-clock time T is measured from the start of tree expansion to the end of the planning loop under a fixed iteration budget `iter_max`. Scenes are paired by seed so that each sampled environment is solved by both planners. Reported summaries include the median planning time per planner on successful roll-outs and the median paired difference across matched seeds, computed from the saved per-rollout logs, in order to remove variance induced by the scene. Trajectory quality is measured by the geometric length (see Eq. (18)) computed on the returned path. The primary trajectory summary is the median paired difference in ℓ_{edge} between the CBF-filtered planner and the baseline on successful pairs; pairs in which either planner returns no path are excluded from the paired summaries and reported separately.

Sensitivity for RQ 3 is evaluated on a grid over $(k_{\text{cbf}}, v_{\text{nom}}, \texttt{step\_len})$. For each grid point, the success rate is reported, where success means that a path is returned. The distribution of the minimum signed distance on successful paths is summarized to quantify dynamic clearance. Each grid point is treated as a group and these success rates, collision proportions, and clearance summaries are interpreted descriptively to identify parameter regions that maintain safety margins and practical path quality. No additional formal hypothesis tests are imposed.

4 Results and Discussion

This section first reports the empirical outcomes for RQ 1 – RQ 3 in a question-by-question format, restricting the presentation in Section 4.1 – Section 4.3 to descriptive statistics, figures, and tables without interpreting the results. Definitions, counting rules, and statistical conventions are fixed in the List of Symbols (Section b) and Section 3 and are applied here without re-derivation. Where useful, figures summarise central quantities with confidence intervals, and tables list exact counts so that the reported numbers can be checked from first principles. After the per-question results, Section 4.4 integrates the findings with a focus on safety-critical implications, Section 4.5 states explicit threats to validity, and Section 4.6 closes with a short summary of the main findings.

4.1 Safety

The safety evaluation uses the low-speed generator defined in Section 3.5, with dynamic-obstacle speeds bounded by $1.0\,\mathrm{m\,s^{-1}}$ and obstacle counts sampled as specified there. Denominators count only completed roll-outs, and numerators count runs with at least one dynamic collision along the time-stamped path under the stated prediction model. The evaluation produced 3000 completed roll-outs, 0 detected collisions, an observed collision proportion $\hat{p} = 0.0000$, and a Wilson $95\,\%$ confidence interval $[0.0000, 0.0013]$. The pre-registered decision rule accepts the low-speed generator ($v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$) when the upper endpoint of the two-sided $95\,\%$ Wilson interval does not exceed 0.01. Since the observed upper endpoint equals $0.13\,\%$, the requirement is satisfied with margin and the decision is Pass. This gives a direct answer to RQ 1: in the low-speed setting with $v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$ the dynamic-obstacle LQR-CBF-RRT* meets the stated safety target on the sampled evidence.

The accounting reported here is based on the actual pre-registered decision rule execution logs. The full logs and run artifacts are available in the companion repository `https://github.com/Leg0shii/LQR_CBF_rrtStar/tree/dyn-tvcbf` in the folder results. Listing 5 reproduces the console trace for roll-out $1/3000$ and shows three dynamic obstacles with positions, radii, velocity vectors, and speeds, followed by the start and goal states and the straight-line distance between them. Planning diagnostics then report tree growth and waypoint count. In this instance the planner added 1317 nodes, reached a tree size of 1318, returned a path with 54 waypoints, and finished with the status `SUCCESS`.

```
2025-09-21 13:42:25,019 - INFO - Experiment timestamp: 20250921_134225
2025-09-21 13:42:25,019 - INFO -
--- ROLLOUT 1/3000 ---
```

```
2025-09-21 13:42:25,019 - INFO - Obstacles (3):
2025-09-21 13:42:25,019 - INFO - Obs 0: pos=(24.5, 17.9), r=2.2, v=(0.95, 0.15), speed=0.96
2025-09-21 13:42:25,019 - INFO - Obs 1: pos=(30.1, 13.1), r=1.8, v=(-0.34, -0.31), speed
    ↪ =0.46
2025-09-21 13:42:25,019 - INFO - Obs 2: pos=(18.6, 12.8), r=1.8, v=(-0.32, 0.10), speed=0.34
2025-09-21 13:42:25,019 - INFO - Start: (22.2, 8.1)
2025-09-21 13:42:25,019 - INFO - Goal: (41.8, 11.1)
2025-09-21 13:42:25,019 - INFO - Distance: 19.8
2025-09-21 13:42:40,758 - INFO - Planning: 1317 nodes added, tree size: 1318
2025-09-21 13:42:40,791 - INFO - Path: 54 waypoints
2025-09-21 13:42:40,796 - INFO - Result: SUCCESS
2025-09-21 13:42:40,830 - INFO -
```

Listing 5: Roll-out 1/3000 log for the low-speed generator ($v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$). Three dynamic obstacles with positions, radii, velocity vectors, and speeds are listed. Start and goal, straight-line distance, tree growth (nodes added = 1317, tree size = 1318), path length in waypoints (54), and the final status `SUCCESS` are shown.

Listing 6 aggregates all 3000 roll-outs at $v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$ and lists attempted, skipped, and completed counts, confirms that no runs were skipped for infeasible start or goal, records 3000 successful paths and 0 collisions, computes the collision rate as $0.00\,\%$, and displays the Wilson $95\,\%$ interval $[0.00\,\%, 0.13\,\%]$ together with the verdict `PASSED` under the upper-endpoint $1\,\%$ acceptance rule.

```
=== COLLISION RATE ANALYSIS ===
Total rollouts attempted: 3000
Rollouts skipped (infeasible start/goal): 0
Actual rollouts completed: 3000
Max obstacle speed: 1.0 m/s

Results:
- Successful paths: 3000
- Collisions detected: 0
  * No path found: 0
  * Collision on path: 0
- Collision rate: 0.0000 (0.00%)
- 95% Wilson confidence interval: [0.0000, 0.0013]
- 95% Wilson upper bound: 0.0013 (0.13%)

Collision trajectories saved: 0 files
Directory: collision_trajectories/
```

```
18
19 Verdict: PASSED: 95% upper bound is below 1%
```

Listing 6: Collision-rate audit aggregated over 3000 roll-outs at $v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$. Completed $= 3000$, collisions $= 0$, $\hat{p} = 0.0000$, Wilson 95 % interval $[0.0000, 0.0013]$. The upper endpoint of the two-sided 95 % Wilson interval 0.13 % lies below the 1 % requirement, hence PASSED.

Fig. 9 complements these logs with a trajectory visualization produced by a sample execution: the green tree shows the RRT* exploration, orange discs denote dynamic obstacles at the plotted time stamp, the red polyline is the returned solution path, the green square marks the start, the red square marks the goal, and the blue dot highlights the robot which is following the path. The annotation "Time = 6.45 s" indicates the simulation time for the rendered frame. This figure makes visible how the steering and the barrier checks interact with the sampled environment to yield a collision-free path.

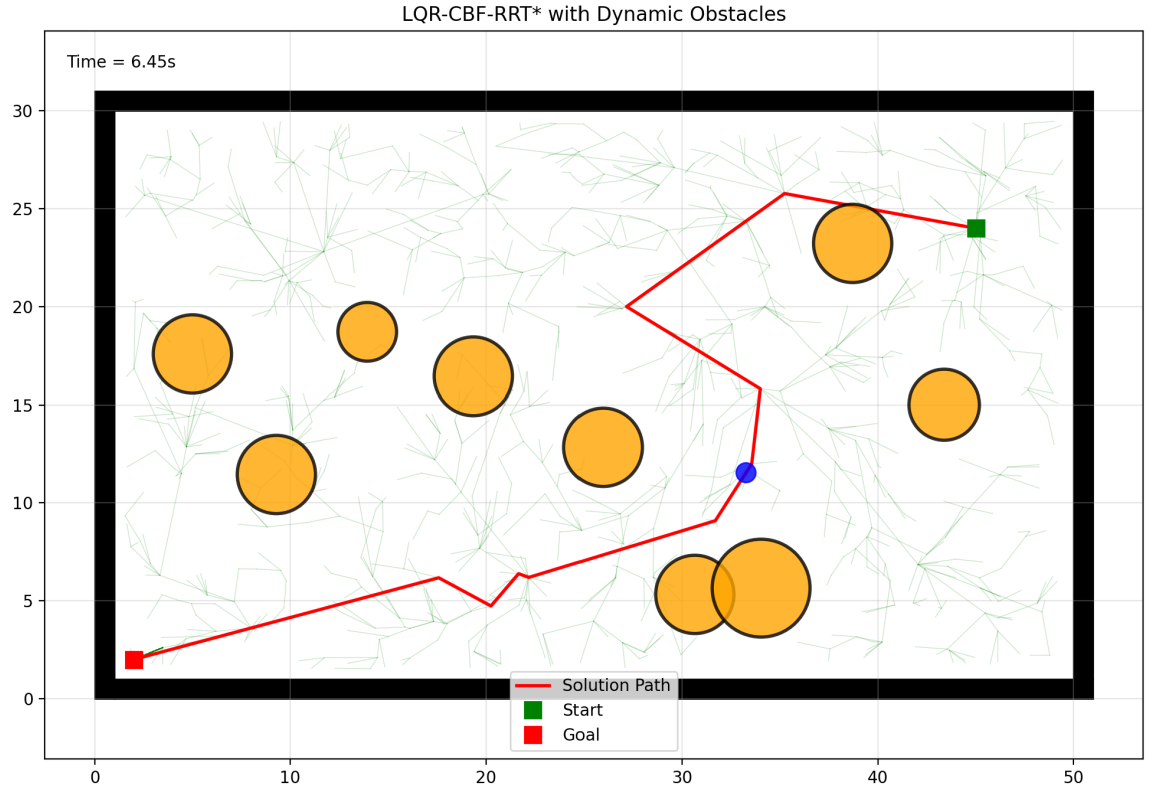

Fig. 9: Trajectory visualization for a representative low-speed-generator scene ($v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$). The green tree shows sampled edges of LQR-RRT* exploration, orange discs depict dynamic obstacles at the shown time, the red polyline is the final solution path from the red start marker to the green goal marker, and the blue dot marks a steering update. The timestamp indicates the simulated time for the rendered frame. Source: Own illustration.

During large-scale execution a computational trade-off had to be made to keep wall-clock time reasonable. The QP controller with prediction was disabled for the planner stepping loop because calls

to the online QP solver with time-varying constraints substantially increased per-step latency at scale. The disabled path corresponds to

```
if solve_QP:
    try:
        u = self.cbf_rrt_simulation.QP_controller_with_prediction(
            x, u, model="linear", current_time=current_time + time
        )
    except:
        print("infeasible")
        break
```

Listing 7: QP-based CBF correction with prediction. Abort on infeasibility.

so `solve_QP` was set to `False` for RQ 1 runs. Safety was still enforced by retaining the predictive CBF feasibility check before committing a step. The active path was

```
if cbf_check and not test_LQR and not solve_QP:
    current_pos = [rx[-1], ry[-1]]
    if not self.cbf_rrt_simulation.QP_constraint_with_prediction(
        current_pos, u, dt_horizon=current_time + time
    ):
        break
```

Listing 8: Predictive CBF feasibility check without QP. Terminate edge on violation.

which rejects any step that violates the forward-looking barrier constraint under the same prediction horizon. This choice preserves the formal guard on the executed trajectory while avoiding the cost of solving the QP at every integration step. The zero-collision outcome over 3000 roll-outs, together with the Wilson upper bound of 0.0013, indicates that this enforcement mechanism sufficed for the low-speed generator setting evaluated here.

4.2 Overhead

This subsection quantifies the additional computational and geometric cost introduced by the safety layer relative to the LQR-RRT* baseline under paired scenes. Pairing uses identical seeds and scene realizations as specified in Section 3.5 so that each pair differs only by the presence or absence of the CBF filter during expansion. Contrary to RQ 1, the quadratic-program controller branch was enabled. This increased runtime, so roll-outs were limited to 100 to stay within reasonable time. Safety was

also enforced by a predictive CBF feasibility check at each extension that rejects steps violating the barrier constraint. The overhead measured here therefore includes repeated QP solves in addition to barrier checks and altered exploration dynamics.

All 100 scene seeds produced plans in both configurations, so 100 matched pairs are available for time and for length comparisons and no exclusions occur. Median planning time for the baseline is $10.86\,\mathrm{s}$ with interquartile range $[10.28, 11.45]\,\mathrm{s}$. Median planning time for the safety-enabled variant is $92.92\,\mathrm{s}$ with interquartile range $[88.86, 98.05]\,\mathrm{s}$. The paired time-difference distribution has median $82.25\,\mathrm{s}$ with interquartile range $[78.46, 86.45]\,\mathrm{s}$. This magnitude implies a single-core planning rate near $0.092\,\mathrm{Hz}$ for the baseline and about $0.012\,\mathrm{Hz}$ for the safety-enabled configuration, consistent with an order-of-magnitude increase in wall-clock time. A scale-free summary gives a median runtime ratio $T_{\mathrm{cbf}}/T_{\mathrm{base}} = 8.59$ with interquartile range $[8.31, 8.97]$. Percentile checks provide the 90th-percentile planning times of $12.12\,\mathrm{s}$ (baseline) and $103.62\,\mathrm{s}$ (safety-enabled). Fig. 10 shows the two method medians with interquartile range (IQR) whiskers.

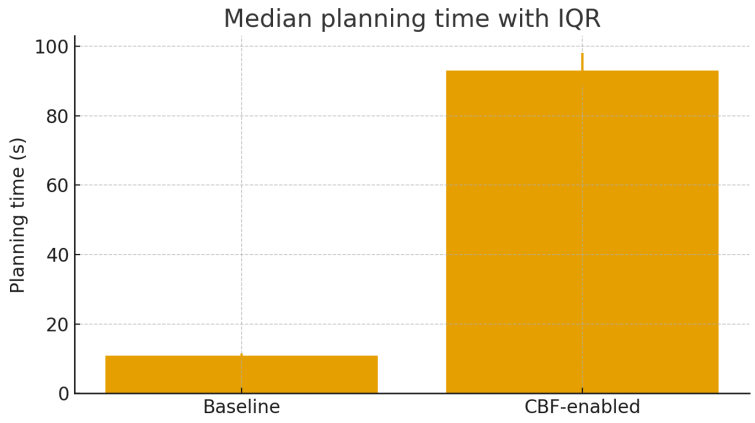

Fig. 10: Median planning time with IQR for baseline and CBF-enabled planners on 100 paired scenes. Source: Own illustration.

Path-length comparisons indicate no systematic inflation under the safety layer. Median geometric length is $27.02\,\mathrm{m}$ for the baseline with interquartile range $[19.56, 34.55]\,\mathrm{m}$ and $25.88\,\mathrm{m}$ for the safety-enabled planner with interquartile range $[18.46, 35.92]\,\mathrm{m}$. The paired median length difference is $1.72\,\mathrm{m}$ with interquartile range $[-3.05, 5.06]\,\mathrm{m}$. Quantiles of the paired length differences $(J_{\mathrm{cbf}} - J_{\mathrm{base}})$ are $-17.38\,\mathrm{m}$ at the 10th percentile and $9.41\,\mathrm{m}$ at the 90th percentile. Tree-size statistics for the same 100 pairs are: baseline median 1349.5 nodes $[1333.75, 1362.00]$, safety-enabled median 1352.0 nodes $[1331.75, 1363.25]$, and a paired median difference of 0.5 nodes with interquartile range $[-6.0, 6.25]$. Fig. 11 displays the medians with IQR whiskers.

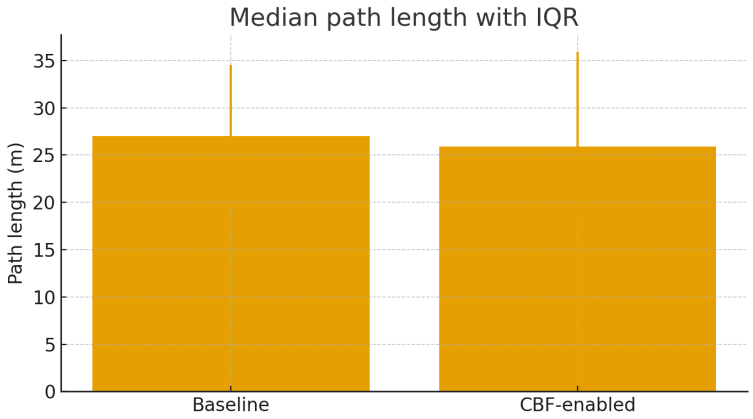

Fig. 11: Median geometric path length with IQR for baseline and CBF-enabled planners on 100 paired scenes. Source: Own illustration.

4.3 Sensitivity

This subsection examines how outcomes vary with the CBF gain α, the nominal expansion speed v_{nom}, and the geometric extension cap `step_len`. The reporting focuses on two quantities: (i) success rate (path found; proportion of completed roll-outs per cell) and (ii) the minimum signed obstacle distance d_{min} computed along collision-free paths. Results are stated as cell-wise summaries with figures and a numeric table for auditability.

Across all 18 factor cells there are $N = 1800$ roll-outs in total (100 per cell). The aggregate success count is 1799 (overall proportion 0.9994). Factor-wise aggregation yields the following summaries. For the CBF gain, the mean success rates aggregated over all v_{nom} and `step_len` are 1.000 at $\alpha = 0.25$, 1.000 at $\alpha = 0.50$, and 0.998 at $\alpha = 1.00$; the minimum cell-wise success within these levels is 1.00, 1.00, and 0.99, respectively. The corresponding mean cell-wise d_{min} values are $3.258\,\text{m}$, $3.174\,\text{m}$, and $3.254\,\text{m}$. For the nominal expansion speed, the mean success rates are 1.000 at $v_{\text{nom}} = 1.5\,\text{m}\,\text{s}^{-1}$, 0.998 at $3.0\,\text{m}\,\text{s}^{-1}$, and 1.000 at $5.0\,\text{m}\,\text{s}^{-1}$; the mean d_{min} values are $3.563\,\text{m}$, $3.224\,\text{m}$, and $2.898\,\text{m}$, respectively. For the extension cap, the mean success rates are 0.999 for `step_len` $= 6$ and 1.000 for `step_len` $= 12$; the mean d_{min} values are $3.134\,\text{m}$ and $3.322\,\text{m}$, respectively. The single failure occurs in the cell ($\alpha = 1.00, v_{\text{nom}} = 3.0\,\text{m}\,\text{s}^{-1}$, `step_len` $= 6\,\text{m}$). Fig. 12 shows that single failure case (0-based rollout index 90: the 91st trial), with the planned path, dynamic obstacles, and the collision point.

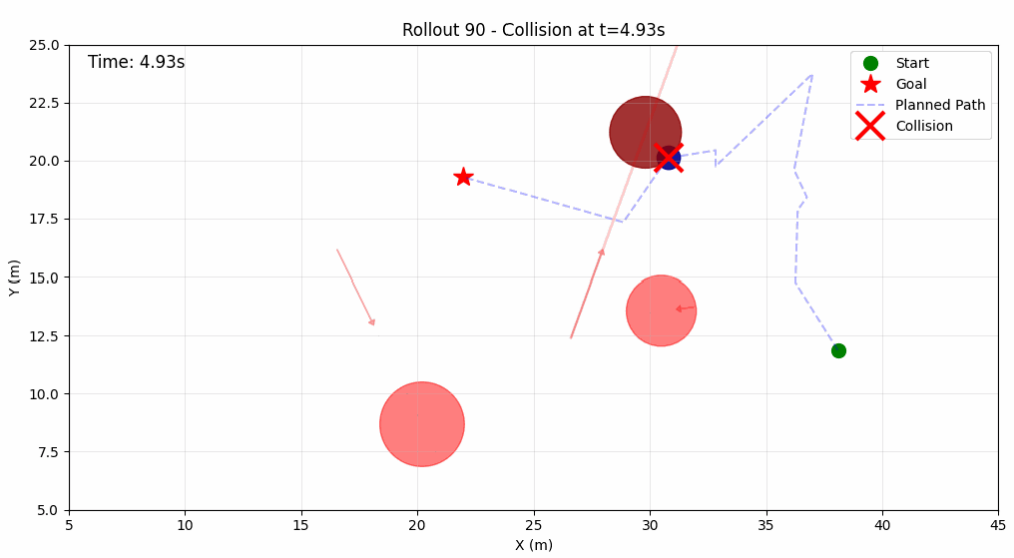

Fig. 12: Sensitivity study single failure instance for the cell $\alpha = 1.00$, $v_{\text{nom}} = 3.0\,\text{m}\,\text{s}^{-1}$, `step_len`=6 m. Dashed blue: planned path; red cross: collision; green dot: start, red star: goal; shaded discs with arrows: dynamic obstacles and velocities. The title shows Rollout 90, which is the 0-based index used in logging. Source: Own illustration.

Fig. 13 and Fig. 14 visualizes d_{min} across the factor grid using interaction plots. Each panel fixes `step_len` and varies $v_{\text{nom}} \in \{1.5, 3.0, 5.0\}\,\text{m}\,\text{s}^{-1}$ on the horizontal axis; lines correspond to α levels and points show cell medians with dispersion markers. This figure provides the graphical summary for RQ 3.

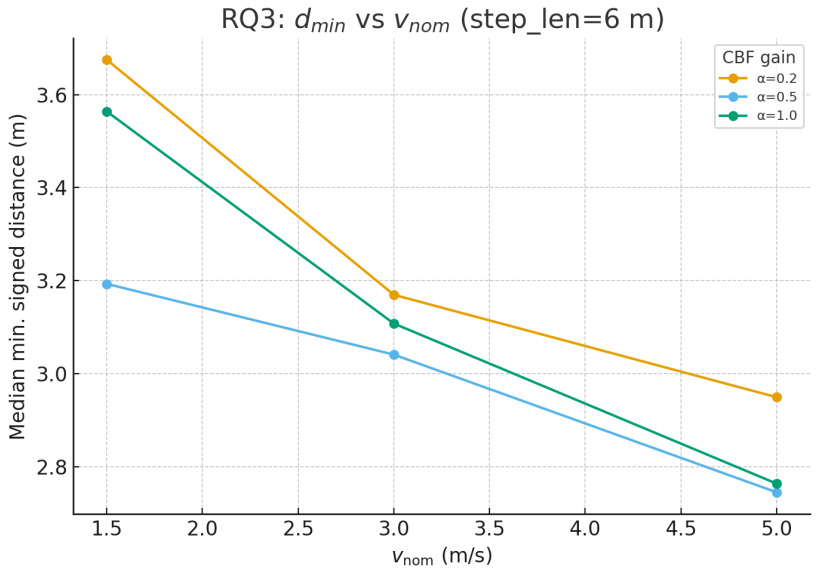

Fig. 13: Minimum signed obstacle distance d_{min} (m) by nominal velocity $v_{\text{nom}} \in \{1.5, 3.0, 5.0\}\,\text{m}\,\text{s}^{-1}$ with lines for $\alpha \in \{0.25, 0.5, 1.0\}$ at `step_len` $= 6\,\text{m}$. Points are cell medians; whiskers show dispersion where available. Source: Own illustration.

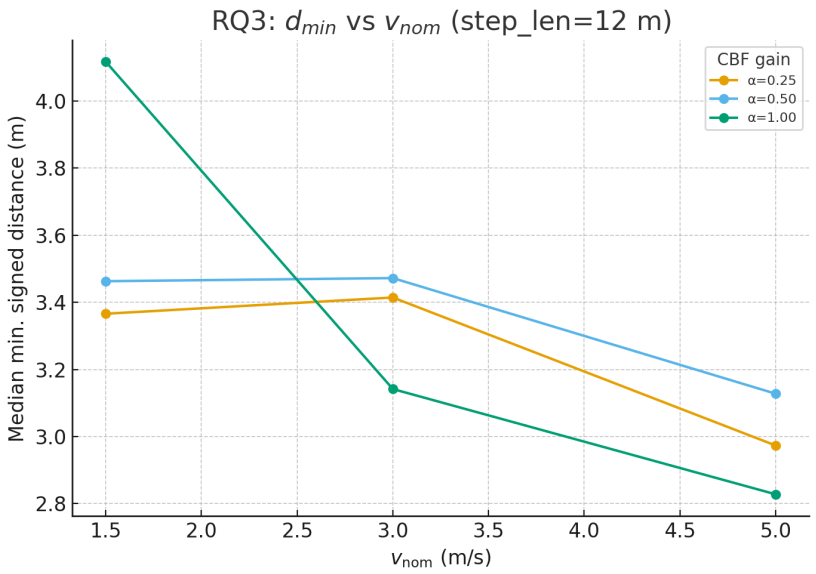

Fig. 14: Minimum signed obstacle distance d_{min} (m) by nominal velocity $v_{\text{nom}} \in \{1.5, 3.0, 5.0\}\,\text{m}\,\text{s}^{-1}$ with lines for $\alpha \in \{0.25, 0.5, 1.0\}$ at $\texttt{step\_len} = 12\,\text{m}$. Points are cell medians; whiskers show dispersion where available. Source: Own illustration.

The numeric detail per cell, including the number of trials and the number of successes, appears in Tab. 3. Success proportions are reported to three decimals and distances to three decimals (meters). When success proportions round to 0.99 in a cell, the counts in the last two columns remain authoritative.

Tab. 3: Sensitivity summary per factor cell. Success proportion and median minimum signed distance (averaged per cell). Source: Own illustration.

α	v_{nom} (m s^{-1})	`step_len` (m)	Success	d_{min} (m)	Trials	Successes
0.25	1.5	6	1.000	3.675	100	100
0.25	1.5	12	1.000	3.366	100	100
0.25	3.0	6	1.000	3.170	100	100
0.25	3.0	12	1.000	3.414	100	100
0.25	5.0	6	1.000	2.950	100	100
0.25	5.0	12	1.000	2.973	100	100
0.50	1.5	6	1.000	3.193	100	100
0.50	1.5	12	1.000	3.463	100	100
0.50	3.0	6	1.000	3.041	100	100
0.50	3.0	12	1.000	3.472	100	100
0.50	5.0	6	1.000	2.745	100	100
0.50	5.0	12	1.000	3.128	100	100
1.00	1.5	6	1.000	3.564	100	100
1.00	1.5	12	1.000	4.116	100	100
1.00	3.0	6	0.990	3.108	100	99
1.00	3.0	12	1.000	3.141	100	100
1.00	5.0	6	1.000	2.764	100	100
1.00	5.0	12	1.000	2.828	100	100

All numbers and the interaction figure in this subsection are computed from the RQ 3 statistics file and the execution logs produced by the factor sweep. The per-cell counts in Tab. 3 derive from the `num_total` and `num_success` fields in the experiment outputs. No additional processing beyond numeric aggregation and formatting is applied.

4.4 Discussion

The acceptance result for RQ 1 shows that, under the low-speed generator with obstacle speeds up to $1.0\,\text{m}\,\text{s}^{-1}$, completed trajectories have an estimated collision probability well below the $1\,\%$ budget when safety is enforced through predictive control-barrier checks and completion-based accounting (see Section 4.1). For applications that fit this envelope, planar motion, walking-speed traffic, and access to a reasonably accurate prediction model, this supports using the planner as a safety filter for route proposals: trajectories that pass the acceptance test can be treated as rare-collision candidates, while trajectories that fail call for re-planning or a fall-back strategy. The same result also clarifies what

is not covered. The test says nothing about behaviour at higher obstacle speeds, in denser fields than the generator, or under sensing and model errors, because these factors are held fixed in the experiments. In practice, this means that any attempt to transfer the acceptance statement to a new scenario should start by checking whether the new scenario can be mapped back to the generator assumptions (speed, clutter, dimensionality, prediction oracle); if not, the safe option is to treat the current safety bound as inapplicable and to rerun an acceptance campaign with adjusted parameters.

For RQ 2, the roughly order-of-magnitude increase in single-core planning time (Section 4.2) points to clear usage patterns. With the safety layer active, median runtimes correspond to an effective replanning rate around 0.01 Hz, which fits offline batch tasks: generating training data, stress-testing static maps, or producing route libraries that a faster online controller can track. The same timings make continuous replanning at automotive update rates unrealistic without additional engineering effort. A vehicle control stack that updates every 50 ms cannot wait tens of seconds for a new plan, so this planner would need parallelisation, hardware acceleration, or a reduced safety workload before it could serve as the front-line online planner. Geometric overhead is modest in the same paired design, with similar median path lengths for baseline and safety-enabled runs. This indicates that rejecting unsafe expansions through predictive feasibility checks does not, on these scenes, force the planner into clearly longer or more contorted routes. From an application perspective, this is useful: most of the cost of enforcing safety appears as computation, not as extra distance, so comfort and energy use are less affected than CPU time. If future optimisations can reduce the cost of the barrier checks and prediction, there is a plausible path to a configuration that retains current path quality while improving responsiveness.

For RQ 3, the parameter sweep shows near-saturated success and stable d_{min} across the tested grid, with trends dominated by nominal speed and `step_len` and weaker dependence on α (Section 4.3). This behaviour reduces the calibration burden for practitioners: within the evaluated ranges, it is hard to choose a clearly “bad” setting by accident, and parameter choices can be driven by resource constraints. A team that can afford higher compute budgets might run with slower nominal speeds to gain some clearance margin; a team that needs faster turn-around can push towards $v_{\text{nom}} = 5\,\text{m}\,\text{s}^{-1}$ and `step_len` = 12 m, as suggested by the results, without paying a noticeable penalty in success rate or minimum distance.

The single recorded failure marks an important boundary. It appears at an aggressive combination of parameters and involves a collision that occurs only after several seconds of motion along a path that has passed all expansion-time feasibility checks. This illustrates a general limitation of using predictive CBF tests solely as an accept-reject gate in the tree: they do not generate corrective inputs along the executed trajectory. For safety-critical use, this suggests two complementary levers. One

is to introduce an inner-loop CBF-QP or other tracking controller that can respond to late-emerging conflicts along the path. The other is to treat configurations near the observed failure cell as outside the certified operating region, reserving them for exploratory use rather than for any setting that must carry a quantitative safety claim.

Combining the three research questions yields a specific engineering picture. The current LQR-CBF-RRT* implementation is well suited to generating and auditing trajectories for low-speed planar motion under the tested generator and prediction model, it carries a substantial computational cost that currently pushes it towards offline or background roles, and it behaves robustly across a range of tuning parameters without strong geometric penalties. To extend these properties to higher speeds, perception-driven state estimates, or richer vehicle models, future work would need to accelerate the safety computations, model sensing and dynamics uncertainty explicitly, and re-run an acceptance test analogous to RQ 1 for the new operating envelope. Methodologically, the combination of a Wilson-bound acceptance test for RQ 1 and paired-scene medians with interquartile ranges for RQ 2 and RQ 3 provides an evaluation template that other planners can reuse. Any method tested on the same generator and under the same accounting rules can be compared against the current results on equal terms, which turns this study into a baseline for safety-critical path planners rather than an isolated case.

4.5 Threats to Validity

Internal validity is most affected by the decision to disable the online QP controller and to enforce feasibility through predictive control-barrier checks only. This choice alters exploration because candidate edges are either accepted or rejected without synthesizing corrective inputs that might rescue near-violations during execution. The acceptance statistics therefore reflect a stricter gate at expansion time rather than the behavior of a planner that continuously solves for control inputs along edges. This is acceptable for the stated research questions because RQ 1 evaluates an acceptance rule over completed roll-outs, RQ 2 quantifies the overhead of the implemented safety mechanism, and RQ 3 studies parameter effects under the same mechanism. None of these require equivalence to a controller-enabled variant. Nevertheless, conclusions should be read as properties of the predictive-feasibility design rather than of any CBF-QP controller in general.

Statistical validity hinges on rare-event inference with near-zero observed collisions. Wilson intervals are used in the analysis because they provide better coverage near the boundaries of the probability space compared to Wald approximations and perform reliably even for small sample sizes (Brown

et al., 2001, pp. 102, 110). Although the requirement is one-sided (an upper bound), the reported acceptance bound uses the upper endpoint of the two-sided $95\,\%$ Wilson interval, which is conservative relative to a one-sided $95\,\%$ Wilson limit. Denominators include only completed roll-outs so that the probability being bounded corresponds to the event of a collision along a produced trajectory, which is the relevant quantity for an acceptance decision. Residual uncertainty remains despite 3000 trials, and the interval width reflects that uncertainty. The upper endpoint could move if a small number of additional collisions were observed. All success proportions reported for RQ 3 use binomial denominators of 100 per cell and are therefore subject to rounding at the third decimal place.

Construct validity is limited by the modeling choices. The experiments use a 2D kinematic setting, a fixed dynamic-obstacle generator with stated speed bounds and counts, and a particular prediction model used for forward auditing. Sensing noise, state-estimation delay, and model mismatch are not injected, and the control stack is not closed with a dynamics-level tracker during planning. As a result, the measured safety and overhead characterize an idealized planning layer operating on perfect state with a specific prediction oracle. Claims about behavior under perception noise, nonholonomic constraints, or strongly nonlinear vehicle models are outside the scope of this study.

External validity is constrained by the tested envelope and environment randomness. Obstacle speeds are bounded by $1.0\,\mathrm{m\,s^{-1}}$, obstacle counts follow the generator in Section 3.5, and scenes lie in 2D rectangular workspaces. Generalization to higher speeds, denser fields, different obstacle policies, or different workspace topologies is not guaranteed. Likewise, extension to higher-dimensional robots or kinodynamic planners would require fresh evidence because tree growth, feasibility checks, and overhead scale differently with dimension. To extend claims, additional experiments should vary obstacle speed and density beyond the current bounds, alter motion models for dynamic obstacles, and test on vehicles with more complex state and control spaces under the same acceptance protocol.

Ablation choices also shape interpretation. Hyperparameters unrelated to the factors under study are held fixed across variants, paired seeds ensure that baseline and safety-enabled runs face identical scene realizations, and hardware and linear-algebra library settings are constant. These controls isolate differences to the presence of the safety mechanism and to the specified factor levels, but they also mean that conclusions pertain to this configuration. Different nearest-neighbor data structures, steering horizons, or collision-check tolerances could shift both compute and clearance metrics even if the qualitative mechanism remains the same.

Reproducibility is supported by storing the CSV summaries used for figures and tables and the generated images with filenames referenced in the text. The logs contain per-rollout metadata, including obstacle parameters, start and goal, planning diagnostics, and result status, as well as explicit records

for the single failure case. With these artifacts, the acceptance numbers, overhead medians, and sensitivity summaries can be regenerated from raw outputs, and the plotted values can be audited against their source tables.

4.6 Summary of Findings

The evidence supports the stated safety target under the evaluated low-speed generator. The acceptance decision is positive when assessed using the upper endpoint of the two-sided $95\,\%$ Wilson interval. See Section 4.1 for the formal test and artefacts. The safety layer introduces a substantial runtime overhead relative to the LQR-RRT* baseline, while geometric path cost remains comparable in paired scenes. The overhead arises from predictive feasibility checks, dynamic-obstacle propagation on the audit horizon, and resampling after rejected expansions. See Section 4.2 and Fig. 10 – Fig. 11. Parameter sensitivity is benign over the tested grid. Success rates are near-saturated, and minimum signed distance trends are dominated by nominal speed, with a secondary effect from the extension cap and only minor influence from the CBF gain. See Section 4.3, Fig. 13 – Fig. 14, and Tab. 3.

A practical default consistent with these findings is to use a moderate CBF gain, a higher nominal expansion speed, and the larger extension cap from the grid. This choice balances clearance with compute and matches the overhead profile observed in RQ 2. All conclusions are scoped to the generator, prediction model, and completion-based accounting used here. Reproducibility is ensured by the released logs, CSV summaries, and figure artefacts referenced across Section 4.1 – Section 4.3.

5 Conclusion

5.1 Summary

This thesis addressed safe and cost-efficient motion planning with moving obstacles in a two-dimensional dynamic environment. It investigated whether a hybrid LQR-CBF-RRT* planner can generate collision-free paths that meet a pre-specified safety acceptance rule while keeping computational and geometric costs practical. The planner uses linear-quadratic regulation for local steering and enforces execution-time safety via a predictive control-barrier feasibility gate, targeting scenes with bounded obstacle speeds.

Implementation-wise, LQR-directed, sample-based expansions are time-stamped and audited by the CBF gate before candidate edges enter the tree. The inner-loop QP controller was disabled in the safety and sensitivity studies (RQ 1, RQ 3) for scalability and enabled in the paired overhead study (RQ 2) to assess its cost. Safety is thus enforced via accept-reject filtering, while the underlying geometric layer retains the asymptotic optimality properties of RRT*.

The evaluation aligned with the three research questions and used paired designs and pre-registered counting rules. For safety, a low-speed scenario generated 3000 completed roll-outs with zero observed collisions. For overhead, 100 scene seeds were run with and without the safety layer under identical random realizations. Median planning time increased from $10.86\,\mathrm{s}$ to $92.92\,\mathrm{s}$. Geometric path length remained similar between variants. For parameter sensitivity, a $3 \times 3 \times 2$ grid over $\alpha \in \{0.25, 0.5, 1.0\}$, $v_{\text{nom}} \in \{1.5, 3, 5\}\mathrm{m\,s^{-1}}$, and $\texttt{step\_len} \in \{6, 12\}\mathrm{m}$ yielded 1799 successes in 1800 trials with one failure at $(\alpha = 1.0, v_{\text{nom}} = 3.0\,\mathrm{m\,s^{-1}}, \texttt{step\_len} = 6\,\mathrm{m})$. Median minimum signed obstacle distance decreased with higher nominal speed and was slightly larger for the $12\,\mathrm{m}$ extension cap. The dependence on the gain was secondary under the predictive feasibility gate.

Taken together, these outcomes support a concise restatement of the main findings. First, under the specified generator with obstacle speeds up to $1.0\,\mathrm{m\,s^{-1}}$, the planner met the safety acceptance criterion with substantial margin. Second, the safety layer imposed a large computational cost in the tested single-core implementation while leaving geometric optimality essentially unchanged in median terms. Third, sensitivity analysis showed saturated success across almost all cells and identified nominal expansion speed as the dominant factor for clearance, with a modest benefit from a larger extension cap. A practical operating point within the tested grid that balances clearance and compute is $(\alpha = 0.5, v_{\text{nom}} = 5\,\mathrm{m\,s^{-1}}, \texttt{step\_len} = 12\,\mathrm{m})$.

5.2 Future Work

A direct extension is to reintroduce an inner-loop controller while keeping the predictive feasibility gate. Warm-started QP, factorization caching, and batched solves could reduce per-step latency enough to make online CBF control practical in the loop. This would enable matched experiments, pure accept-reject filtering versus corrective control along the same accepted edges under identical scenes and logging, to separate the effects of missing feedback from those of prediction and gating. In parallel, the safety layer should account for model uncertainty: distributional forecasts of obstacle motion and explicit bounds on state-estimation error would support chance-constrained checks or robust tubes around nominal predictions. TV CBFs with Lipschitz certificates on prediction error, paired with discretization-aware relaxations, would turn nominal guarantees into statements that tolerate bounded mismatch between model and execution. Concretely, integrating receding-horizon replanning and prediction uncertainty is the most direct path to extend guarantees beyond the bounded-velocity generator used in this thesis.

Planner-level improvements should target the audit horizon that drives compute. State-dependent policies for the extension cap and nominal speed can keep the per-edge time window within a desired band, shorten edges or increase speed in dense traffic, relax those constraints in open space. Sampling and rewiring biased by barrier values can reduce rejected expansions, and nearest-neighbor structures with incremental pruning can limit candidate sets where the barrier is tight. The goal is to lower the number of predictive checks per solution without eroding clearance.

Systems work is needed to approach real-time operation. Parallelization across candidate edges, vectorized distance and prediction kernels, and GPU offload for batch CBF evaluations address the main hotspots. Profiling memory and cache behavior on representative scene mixes, and running hardware-in-the-loop tests, should expose latency spikes and jitter that single-threaded offline runs hide.

The evaluation should be broadened on three axes. First, push to higher obstacle speeds and densities under the same acceptance protocol to quantify how safety margins decay. Second, inject controlled perception noise, latency, and model misspecification to measure robustness gaps between nominal auditing and execution. Third, extend to higher-dimensional vehicles and kinodynamic steering in 3D workspaces, keeping the paired designs and counting rules so results remain comparable. Multi-agent and interaction-rich scenarios form another strand within this broadening: cooperative or adversarial obstacle policies, and multi-robot settings with reciprocal barriers, test whether the current predictive audits and gains carry over when agents respond to each other. Both acceptance and

overhead should be reported, since mutual constraints typically increase rejection rates and planning time.

Finally, reproducibility and benchmarking can be systematized. A compact public suite with fixed seeds and reference numbers for acceptance, clearance, and compute would enable independent verification and fair comparison. Ablations that toggle prediction fidelity, barrier gains, and controller availability, under a shared logging format, would provide a controlled path to identify which components trade compute for safety most effectively in dynamic environments.

References

Agrawal, A., & Sreenath, K. (2017). Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. *Robotics: Science and Systems*, *13*, 1–10.

Ahmad, A., Belta, C., & Tron, R. (2022). Adaptive sampling-based motion planning with control barrier functions. *2022 IEEE 61st Conference on Decision and Control (CDC)*, 4513–4518. https://doi.org/10.1109/CDC51059.2022.9993278

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., & Tabuada, P. (2019). Control barrier functions: Theory and applications. *2019 18th European Control Conference (ECC)*, 3420–3431. https://doi.org/10.23919/ECC.2019.8796030

Ames, A. D., Grizzle, J. W., & Tabuada, P. (2014). Control barrier function based quadratic programs with application to adaptive cruise control. *53rd IEEE Conference on Decision and Control*, 6271–6278. https://doi.org/10.1109/CDC.2014.7040372

Ames, A. D., Xu, X., Grizzle, J. W., & Tabuada, P. (2017). Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, *62*(8), 3861–3876. https://doi.org/10.1109/TAC.2016.2638961

Arslan, O., Berntorp, K., & Tsiotras, P. (2017). Sampling-based algorithms for optimal motion planning using closed-loop prediction. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4991–4996. https://doi.org/10.1109/ICRA.2017.7989581

Bae, J. W., Kim, J., Yun, J., Kang, C., Choi, J., Kim, C., Lee, J., Choi, J., & Choi, J. W. (2023). Sit dataset: Socially interactive pedestrian trajectory dataset for social navigation robots. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2023/file/4d6a000c216974f59e597bc878cd6325-Paper-Datasets_and_Benchmarks.pdf

Bartels, R. H., & Stewart, G. W. (1972). Algorithm 432 [c2]: Solution of the matrix equation ax + xb = c [f4]. *Commun. ACM*, *15*(9), 820–826. https://doi.org/10.1145/361573.361582

Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/766ebcd59621e305170616ba3d3dac32-Paper.pdf

Black, M., Jankovic, M., Sharma, A., & Panagou, D. (2023). Future-focused control barrier functions for autonomous vehicle control. *2023 American Control Conference (ACC)*, 3324–3331. https://doi.org/10.23919/ACC55779.2023.10156163

Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., & Eckstein, L. (2020). The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. *2020 IEEE Intelligent Vehicles Symposium (IV)*, 1929–1934. https://doi.org/10.1109/IV47402.2020.9304839

Breeden, J., Garg, K., & Panagou, D. (2022). Control barrier functions in sampled-data systems. *IEEE Control Systems Letters*, *6*, 367–372. https://doi.org/10.1109/LCSYS.2021.3076127

Breuer, A., Termöhlen, J.-A., Homoceanu, S., & Fingscheidt, T. (2020). Opendd: A large-scale roundabout drone dataset. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294301

Brown, L. D., Cai, T. T., & DasGupta, A. (2001). Interval estimation for a binomial proportion. *Statistical Science*, *16*(2), 101–133. https://doi.org/10.1214/ss/1009213286

Chalaki, B., & Malikopoulos, A. A. (2022). A barrier-certified optimal coordination framework for connected and automated vehicles. *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2264–2269. https://doi.org/10.1109/CDC51059.2022.9992572

Choi, J. J., Lee, D., Sreenath, K., Tomlin, C. J., & Herbert, S. L. (2021). Robust control barrier–value functions for safety-critical control. *2021 60th IEEE Conference on Decision and Control (CDC)*, 6814–6821. https://doi.org/10.1109/CDC45484.2021.9683085

Chu, Y., Chen, Q., & Yan, X. (2025). An overview and comparison of traditional motion planning based on rapidly exploring random trees. *Sensors*, *25*(7). https://doi.org/10.3390/s25072067

Coppens, P., Schuurmans, M., & Patrinos, P. (2020, October). Data-driven distributionally robust lqr with multiplicative noise. In A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, & M. Zeilinger (Eds.), *Proceedings of the 2nd conference on learning for dynamics and control* (pp. 521–530, Vol. 120). PMLR.

Dai, B., Khorrambakht, R., Krishnamurthy, P., & Khorrami, F. (2025). Differentiable optimization based time-varying control barrier functions for dynamic obstacle avoidance. *Robotics and Autonomous Systems*, *194*. https://doi.org/10.1016/j.robot.2025.105182

Dong, Y., Wang, L., Zhou, S., Hua, G., & Sun, C. (2025). Recurrent aligned network for generalized pedestrian trajectory prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, *35*(3), 2188–2201. https://doi.org/10.1109/TCSVT.2024.3493966

Ezzatti, P., Quintana-Ortí, E. S., & Remón, A. (2011). Solving algebraic riccati equations on hybrid cpu-gpu platforms. *IV High-Performance Computing Symposium (HPC 2011)(XL JAIIO, Córdoba, 31 de agosto y 1º de septiembre de 2011)*.

Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2015). Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 3067–3074. https://doi.org/10.1109/ICRA.2015.7139620

Grothe, F., Hartmann, V. N., Orthey, A., & Toussaint, M. (2022). St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time. *2022 International Conference on Robotics and Automation (ICRA)*, 3314–3320. https://doi.org/10.1109/ICRA46639.2022.9811814

Guo, Y., Liu, X., Jia, Q., Liu, X., & Zhang, W. (2023). Hpo-rrt*: A sampling-based algorithm for uav real-time path planning in a dynamic environment. *Complex & Intelligent Systems*, *9*(6), 7133–7153.

Haddaway, N. R., Page, M. J., Pritchard, C. C., & McGuinness, L. A. (2022). Prisma2020: An r package and shiny app for producing prisma 2020-compliant flow diagrams, with interactivity for optimised digital transparency and open synthesis. *Campbell Systematic Reviews*, *18*(2), e1230. https://doi.org/10.1002/cl2.1230

Hanks, B., & Skelton, R. (1991). Closed-form solutions for linear regulator-design of mechanical systems including optimal weighting matrix selection. *32nd Structures, Structural Dynamics, and Materials Conference*.

Huang, J., Chi, X., Liu, Z., & Su, H. (2024). Whole-body dynamic collision avoidance with time-varying control barrier functions. *2024 36th Chinese Control and Decision Conference (CCDC)*, 5149–5154. https://doi.org/10.1109/CCDC62350.2024.10588112

ika RWTH Aachen University. (2024). Drone-dataset tools: Python utilities for ind/highd/round [GitHub repository]. https://github.com/ika-rwth-aachen/drone-dataset-tools

Jallet, W., Dantec, E., Arlaud, E., Mansard, N., & Carpentier, J. (2024). Parallel and Proximal Constrained Linear-Quadratic Methods for Real-Time Nonlinear MPC. *Robotics: Science and Systems*. https://inria.hal.science/hal-04575334

Janson, L., Schmerling, E., Clark, A., & Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, *34*(7), 883–921. https://doi.org/10.1177/0278364915577958

Jongeneel, W., Summers, T., & Esfahani, P. M. (2019). Robust linear quadratic regulator: Exact tractable reformulation. *2019 IEEE 58th Conference on Decision and Control (CDC)*, 6742–6747. https://doi.org/10.1109/CDC40024.2019.9028884

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*(7), 846–894. https://doi.org/10.1177/0278364911406761

Krajewski, R., Moers, T., Bock, J., Vater, L., & Eckstein, L. (2020). The round dataset: A drone dataset of road user trajectories at roundabouts in germany. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294728

Kusano, K. D., Scanlon, J. M., Chen, Y.-H., McMurry, T. L., Chen, R., Gode, T., & Victor, T. (2024). Comparison of waymo rider-only crash data to human benchmarks at 7.1 million miles. *Traffic Injury Prevention*, *25*(sup1), S66–S77. https://doi.org/10.1080/15389588.2024.2380786

Kyaw, P. T., Le, A. V., Yi, L., Veerajagadheswar, P., Vu, M. B., & Elara, M. R. (2024). Greedy heuristics for sampling-based motion planning in high-dimensional state spaces. *arXiv preprint arXiv:2405.03411*.

Laub, A. (1979). A schur method for solving algebraic riccati equations. *IEEE Transactions on Automatic Control*, *24*(6), 913–921. https://doi.org/10.1109/TAC.1979.1102178

Li, W., & Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In: In *Proceedings of the first international conference on informatics in control, automation and robotics - volume 1: Icinco,* INSTICC. SciTePress, 2004, 222–229. ISBN: 972-8865-12-0. https://doi.org/10.5220/0001143902220229

Li, Y., Littlefield, Z., & Bekris, K. E. (2015). Sparse methods for efficient asymptotically optimal kinodynamic planning. In H. L. Akin, N. M. Amato, V. Isler, & A. F. van der Stappen (Eds.), *Algorithmic foundations of robotics xi: Selected contributions of the eleventh international workshop on the algorithmic foundations of robotics* (pp. 528–564). Springer International Publishing. https://doi.org/10.1007/978-3-319-16595-0_16

Lindemann, L., Cleaveland, M., Shim, G., & Pappas, G. J. (2023). Safe planning in dynamic environments using conformal prediction. *IEEE Robotics and Automation Letters*, *8*(8), 5116–5123. https://doi.org/10.1109/LRA.2023.3292071

Liu, L., Zhang, Y., Zhang, L., & Kermanshabi, M. (2024). RRT-CBF based motion planning. *CoRR*, *abs/2410.00343*. https://doi.org/10.48550/ARXIV.2410.00343

Majumdar, A. (2013, June). *Robust online motion planning with reachable sets* [Master's thesis]. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/82404

Mavrogiannis, C., Baldini, F., Wang, A., Zhao, D., Trautman, P., Steinfeld, A., & Oh, J. (2023). Core challenges of social robot navigation: A survey. *J. Hum.-Robot Interact.*, *12*(3). https://doi.org/10.1145/3583741

Mestres, P., Nieto-Granda, C., & Cortés, J. (2025). Safe and dynamically feasible motion planning using control lyapunov and barrier functions. *IEEE Transactions on Robotics*, *41*, 6440–6459. https://doi.org/10.1109/TRO.2025.3626614

Molnar, T. G., & Ames, A. D. (2023). Composing control barrier functions for complex safety specifications. *IEEE Control Systems Letters*, *7*, 3615–3620. https://doi.org/10.1109/LCSYS.2023.3339719

Nguyen, Q., & Sreenath, K. (2016). Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. *2016 American Control Conference (ACC)*, 322–328. https://doi.org/10.1109/ACC.2016.7524935

Nielsen, I., Ankelhed, D., & Axehill, D. (2013). Low-rank modifications of riccati factorizations with applications to model predictive control. *52nd IEEE Conference on Decision and Control*, 3684–3690. https://doi.org/10.1109/CDC.2013.6760450

Otsuki, S., Hatta, N., Hanif, M., Hatanaka, T., & Nakashima, K. (2023). Hierarchical vessel autonomous operation in a port with safety certificates: Combined mpc and cbf approach [22nd IFAC World Congress]. *IFAC-PapersOnLine*, *56*(2), 3138–3145. https://doi.org/10.1016/j.ifacol.2023.10.1447

Otte, M., & Frazzoli, E. (2016). Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, *35*(7), 797–822. https://doi.org/10.1177/0278364915594679

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The prisma 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, *372*. https://doi.org/10.1136/bmj.n71

Peng, C., Donca, O., Castillo, G., & Hereid, A. (2023). Safe bipedal path planning via control barrier functions for polynomial shape obstacles estimated using logistic regression. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 3649–3655. https://doi.org/10.1109/ICRA48891.2023.10160671

Perez, A., Platt, R., Konidaris, G., Kaelbling, L., & Lozano-Perez, T. (2012). Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. *2012 IEEE International Conference on Robotics and Automation*, 2537–2542. https://doi.org/10.1109/ICRA.2012.6225177

Phillips, M., & Likhachev, M. (2011). Sipp: Safe interval path planning for dynamic environments. *2011 IEEE International Conference on Robotics and Automation*, 5628–5635. https://doi.org/10.1109/ICRA.2011.5980306

Ren, Z., Rathinam, S., Likhachev, M., & Choset, H. (2022). Multi-objective safe-interval path planning with dynamic obstacles. *IEEE Robotics and Automation Letters*, *7*(3), 8154–8161. https://doi.org/10.1109/LRA.2022.3187270

Robicquet, A., Sadeghian, A., Alahi, A., & Savarese, S. (2016a). Learning social etiquette: Human trajectory understanding in crowded scenes. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – eccv 2016* (pp. 549–565). Springer International Publishing.

Robicquet, A., Sadeghian, A., Alahi, A., & Savarese, S. (2016b). Stanford drone dataset [Retrieved September 11, 2025, from https://cvgl.stanford.edu/projects/uav_data/].

Sassano, M. (2024). Policy algebraic equation for the lqr and the h∞ control problems. *IEEE Control Systems Letters*, *8*, 370–375. https://doi.org/10.1109/LCSYS.2024.3382439

Shaw Cortez, W., Tan, X., & Dimarogonas, D. V. (2022). A robust, multiple control barrier function framework for input constrained systems. *IEEE Control Systems Letters*, *6*, 1742–1747. https://doi.org/10.1109/LCSYS.2021.3133418

Stagecoach UK Bus. (2023, May). *Launch of uk's first autonomous bus service*. Stagecoach. Retrieved December 6, 2025, from https://www.stagecoachbus.com/news/national/2023/may/launch-of-uks-first-autonomous-bus-service

Stüvel, S. (2020, August 7). Python-rvo2: Optimal reciprocal collision avoidance, python bindings. Retrieved September 11, 2025, from https://github.com/sybrenstuvel/Python-RVO2

Suwoyo, H., Adriansyah, A., Andika, J., Ubaidillah, A., & Zakaria, M. F. (2023). An integrated rrt*smart-a* algorithm for solving the global path planning problem in a static environment. *IIUM Engineering Journal*, *24*(1), 269–284. https://doi.org/10.31436/iiumej.v24i1.2529

Tan, X., & Dimarogonas, D. V. (2024). On the undesired equilibria induced by control barrier function based quadratic programs. *Automatica*, *159*. https://www.sciencedirect.com/science/article/pii/S0005109823005253

Tassa, Y., Mansard, N., & Todorov, E. (2014). Control-limited differential dynamic programming. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 1168–1175. https://doi.org/10.1109/ICRA.2014.6907001

Taylor, A. J., Dorobantu, V. D., Cosner, R. K., Yue, Y., & Ames, A. D. (2022). Safety of sampled-data systems with control barrier functions via approximate discrete time models. *2022 IEEE 61st Conference on Decision and Control (CDC)*, 7127–7134. https://doi.org/10.1109/CDC51059.2022.9993226

Tedrake, R., Manchester, I. R., Tobenkin, M., & Roberts, J. W. (2010). Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, *29*(8), 1038–1052. https://doi.org/10.1177/0278364910369189

Thirugnanam, A., Zeng, J., & Sreenath, K. (2022). Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions. *2022 American Control Conference (ACC)*, 2239–2246. https://doi.org/10.23919/ACC53348.2022.9867246

Webb, D. J., & van den Berg, J. (2013). Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. *2013 IEEE International Conference on Robotics and Automation*, 5054–5061. https://doi.org/10.1109/ICRA.2013.6631299

Wild Thomas, D., Ruml, W., & Shimony, S. E. (2024). Real-time safe interval path planning. *Proceedings of the International Symposium on Combinatorial Search*, *17*(1), 161–169. https://doi.org/10.1609/socs.v17i1.31554

Wu, G., & Sreenath, K. (2016). Safety-critical geometric control for systems on manifolds subject to time-varying constraints. *IEEE Transactions on Automatic Control (TAC), in review*.

Xiao, W., Belta, C. A., & Cassandras, C. G. (2022). Sufficient conditions for feasibility of optimal control problems using control barrier functions. *Automatica*, *135*, 109960. https://www.sciencedirect.com/science/article/pii/S0005109821004866

Xu, X., Grizzle, J. W., Tabuada, P., & Ames, A. D. (2018). Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, *15*(3), 1216–1229. https://doi.org/10.1109/TASE.2017.2760863

Yang, G., Cai, M., Ahmad, A., Prorok, A., Tron, R., & Belta, C. (2023). Lqr-cbf-rrt*: Safe and optimal motion planning. https://arxiv.org/abs/2304.00790

Yang, G., Cai, M., Ahmad, A., Prorok, A., Tron, R., & Belta, C. (2025). Lqr-cbf-rrt*: Safe and optimal motion planning. *2025 American Control Conference (ACC)*, 3700–3705. https://doi.org/10.23919/ACC63710.2025.11108073

Yang, G., Vang, B., Serlin, Z., Belta, C., & Tron, R. (2019). Sampling-based motion planning via control barrier functions, 22–29. https://doi.org/10.1145/3365265.3365282

Yu, M., Yu, C., Naddaf-Sh, M.-M., Upadhyay, D., Gao, S., & Fan, C. (2024). Efficient motion planning for manipulators with control barrier function-induced neural controller. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 14348–14355. https://doi.org/10.1109/ICRA57147.2024.10610785

Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kummerle, J., Konigshof, H., Stiller, C., de La Fortelle, A., & Tomizuka, M. (2019). Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. https://arxiv.org/abs/1910.03088

Zhang, L., Cai, K., Sun, Z., Bing, Z., Wang, C., Figueredo, L., Haddadin, S., & Knoll, A. (2025). Motion planning for robotics: A review for sampling-based planners. *Biomimetic Intelligence and Robotics*, *5*(1), 100207. https://doi.org/10.1016/j.birob.2024.100207

A Appendix

A.1 Code

```
        "dynamic_obstacles": [
            {
                "initial_x": obs[0],
                "initial_y": obs[1],
                "radius": obs[2],
                "velocity_x": obs[3],
                "velocity_y": obs[4],
                "speed": math.sqrt(obs[3]**2 + obs[4]**2)
            } for obs in dynamic_obstacles
        ],
        "robot_trajectory": [
            {
                "x": point[0],
                "y": point[1],
                "time": point[2] if len(point) >= 3 else 0,
                "waypoint_idx": i
            } for i, point in enumerate(path[::-1]) # Reverse to get start-to-goal order
        ],
        "collision_details": {
            "robot_x_at_collision": collision_info['robot_pos'][0],
            "robot_y_at_collision": collision_info['robot_pos'][1],
            "obstacle_x_at_collision": collision_info['obstacle_pos'][0],
            "obstacle_y_at_collision": collision_info['obstacle_pos'][1],
            "obstacle_initial_position": {
                "x": dynamic_obstacles[collision_info['obstacle_idx']][0],
                "y": dynamic_obstacles[collision_info['obstacle_idx']][1],
                "radius": dynamic_obstacles[collision_info['obstacle_idx']][2]
            },
            "obstacle_velocity": {
                "vx": dynamic_obstacles[collision_info['obstacle_idx']][3],
                "vy": dynamic_obstacles[collision_info['obstacle_idx']][4]
            }
        },
        "environment_bounds": {
```

```
            "x_min": 5,
            "x_max": 45,
            "y_min": 5,
            "y_max": 25
        }
    }
```

Listing 9: Complete JSON payload saved for collision replays

A.2 Images

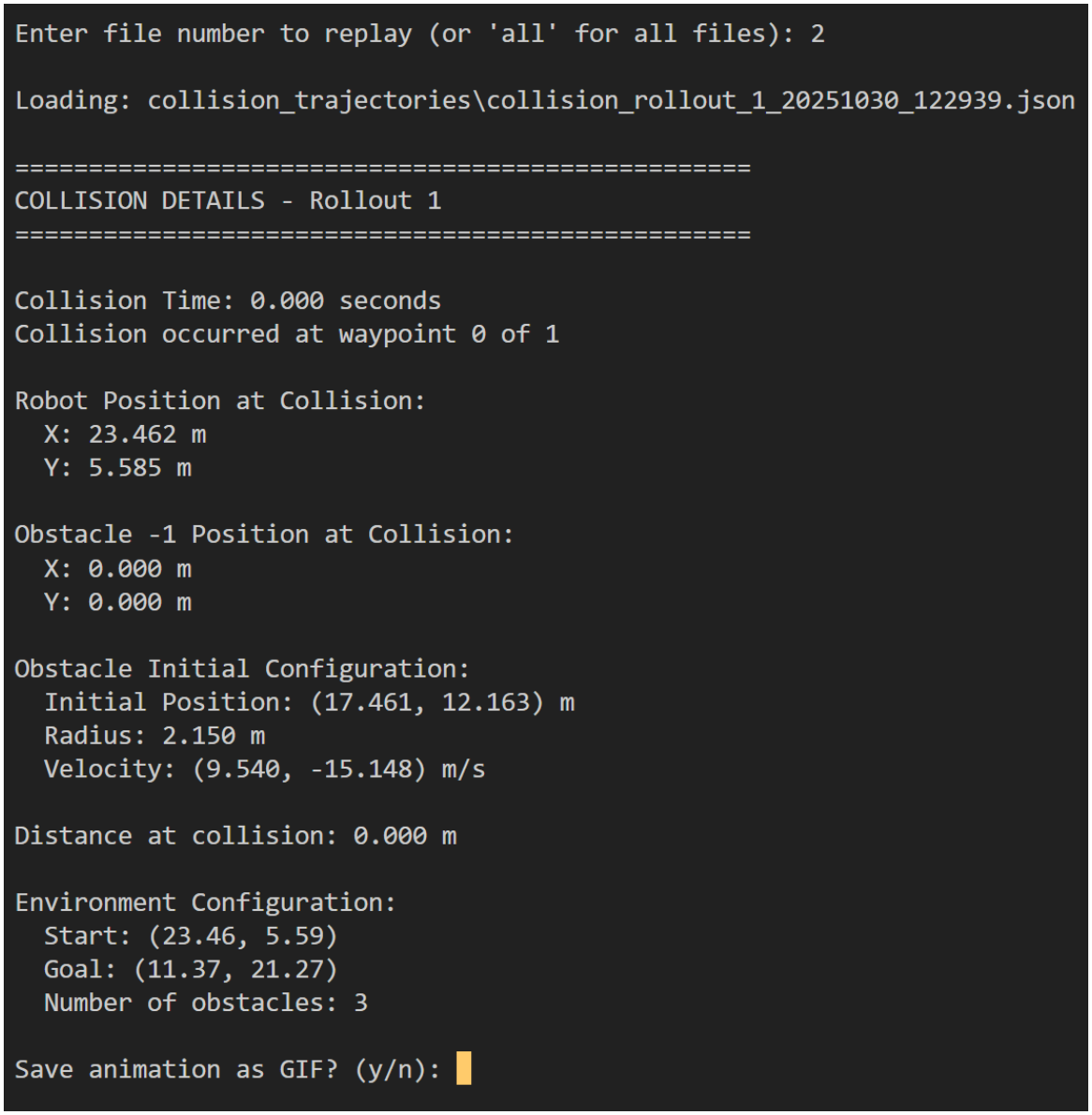

Fig. 15: Replay utility: per-case diagnostics including collision time, robot/obstacle states, environment configuration, and an option to export an animation. Note: To generate a collision case for illustrative purposes, the obstacle velocity was set significantly beyond the experimental bounds, creating an unavoidable collision scenario.

B Declaration of Authenticity

I hereby declare that I have completed this Master's thesis on my own and without any additional external assistance. I have made use of only those sources and aids specified and I have listed all the sources from which I have extracted text and content. This thesis or parts thereof have never been presented to another examination board. I agree to a plagiarism check of my thesis via a plagiarism detection service.

Place, date: Schwerin, December 12, 2025 Signature: B. Müller