

Risk-Aware Hybrid LQR–MPC Navigation for Autonomous Systems

1 Introduction and Motivation

Autonomous robots must navigate safely and efficiently in environments that may vary significantly in complexity. In open, obstacle-free regions, simple feedback controllers are sufficient and desirable due to their low computational cost. However, in cluttered or dynamic environments, controllers must explicitly reason about constraints and future system behavior to avoid collisions.

Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) represent two fundamentally different approaches to this problem. LQR offers computational efficiency and strong local stability guarantees but ignores constraints. MPC, on the other hand, explicitly handles constraints and anticipates future states, but at the cost of significant computational overhead.

Running MPC continuously is often unnecessary and inefficient. Conversely, relying solely on LQR compromises safety. This motivates a **hybrid control strategy** that combines the strengths of both methods.

2 Final Problem Statement

The objective of this project is to design a **risk-aware hybrid navigation controller** such that:

- LQR is used for nominal navigation in low-risk environments
- MPC is activated only when navigation risk becomes significant
- Switching between controllers is governed by explicit, measurable risk metrics

Risk is quantified using:

- Geometric proximity to obstacles
- Predicted constraint violations over a finite horizon
- Estimation or sensing uncertainty

The overarching goal is to achieve safety comparable to always-on MPC while maintaining efficiency close to LQR and significantly reducing average computational cost.

3 Existing Work and Current State of the Field

3.1 LQR-Based Navigation

LQR has been widely used for trajectory tracking and stabilization due to its simplicity and optimality for linear systems. It performs exceptionally well when system dynamics are well modeled and constraints are inactive. However, LQR fundamentally lacks a mechanism to enforce state or input constraints, making it unsuitable for direct obstacle avoidance.

3.2 MPC-Based Navigation

MPC has become the standard tool for constrained navigation and obstacle avoidance. By solving an optimization problem at each time step, MPC explicitly enforces constraints and predicts future system behavior. Numerous works demonstrate MPC's effectiveness in robotic navigation, autonomous driving, and UAV control.

Despite its strengths, MPC suffers from high computational cost, sensitivity to horizon length, and solver dependence. These issues are particularly problematic for embedded systems and multi-agent scenarios.

3.3 Hybrid and Compositional Control Approaches

Recent works have explored combining local optimal controllers with receding-horizon planners. Examples include hybrid iLQR–MPC frameworks and controller composition methods. However, most existing hybrid approaches:

- Are domain-specific (e.g., legged locomotion, contact tasks)
- Use simple distance-based switching
- Do not quantify computational savings versus safety loss

4 Detailed Research Gap

Although the idea of combining LQR and MPC is not new, a **clear and general navigation-oriented framework** is missing.

Specifically:

- Switching is often reactive rather than predictive
- Risk is rarely quantified beyond raw distance thresholds
- Stability during switching is insufficiently addressed in practical implementations
- There is little empirical guidance on how to tune switching thresholds, horizons, or solver parameters

This project addresses these gaps by introducing predictive, risk-aware switching and systematic experimental evaluation.

5 Core Technical Architecture

The proposed system follows a layered control architecture that cleanly separates low-level control, high-level planning, and supervisory decision-making. This separation is intentional and critical for both stability and extensibility.

5.1 Overall Architecture

The control stack consists of four layers:

1. System Modeling Layer
2. Nominal Control Layer (LQR)
3. Safety-Critical Control Layer (MPC)

4. Risk-Aware Supervisory Layer

Each layer has a clearly defined responsibility and communicates through well-defined interfaces.

5.2 System Modeling Layer

The system is modeled using a discrete-time state-space representation. For ground robots, this typically includes planar position, orientation, and velocity states. For UAVs, translational and rotational dynamics are included, with linearization around a nominal operating point.

The purpose of this layer is not perfect modeling, but **sufficient accuracy for short-horizon prediction**. This choice enables real-time feasibility and robustness.

5.3 Nominal Control Layer (LQR)

The LQR controller serves as the default controller and operates continuously unless overridden. Its role is to provide:

- Fast response
- Low computational overhead
- Strong local stability

LQR does not explicitly consider obstacles or constraints. This is a conscious design decision: LQR is trusted only in regions where risk is low.

5.4 Safety-Critical Control Layer (MPC)

The MPC controller is activated only when risk is predicted to exceed acceptable limits. It solves a constrained optimization problem over a short horizon to:

- Enforce obstacle avoidance
- Respect actuator and state constraints
- Anticipate future system behavior

To ensure real-time feasibility:

- The horizon length is kept short
- The optimization is warm-started using LQR predictions
- Conservative constraints are avoided unless necessary

5.5 Risk-Aware Supervisory Layer

The supervisory layer evaluates risk at each control step and decides which controller should be active. This layer is the key contribution of the project.

Risk is assessed using:

- Current geometric proximity to obstacles
- Predicted constraint violations under LQR rollout
- Growth of estimation or sensing uncertainty

Switching is governed by hysteresis and minimum dwell-time constraints to ensure stability and prevent chattering.

6 Platform Choice: Detailed Analysis

The hybrid LQR–MPC framework is platform-agnostic, but execution complexity varies significantly across different autonomous systems. This section explains **if, why, and how** each platform fits the project.

6.1 Single Ground Robot

If used: A single ground robot serves as the primary experimental platform.

Why:

- Dynamics are slow and well-behaved
- Linearization remains valid for longer durations
- Obstacle avoidance is predominantly planar
- Simulation and hardware risks are minimal

How implemented:

- Differential-drive or car-like kinematic model
- MPC enforces position and velocity constraints
- LQR handles nominal path tracking

This platform provides the cleanest validation of the hybrid control concept.

6.2 UAV / Drone

If considered: UAVs are considered as an extension rather than the primary platform.

Why relevant:

- Tight actuator constraints
- High risk associated with collisions
- Strong need for predictive safety control

How it differs:

- Faster dynamics require shorter MPC horizons
- Switching must be more conservative
- Latency and solver speed become critical

UAVs increase complexity and are therefore addressed analytically or as future work.

6.3 Multi-Robot and Swarm Systems

If extended: Hybrid control naturally extends to multi-robot systems.

Why attractive:

- LQR enables efficient formation control
- MPC resolves local conflicts only when needed
- Scalability improves compared to centralized MPC

Engineering challenges:

- Communication delays
- Distributed decision-making
- Increased experiment orchestration complexity

7 Comparative View of Control Strategies

7.1 Pure LQR

Strengths:

- Extremely low computation
- Strong local stability

Weaknesses:

- Cannot enforce constraints
- Unsafe near obstacles

7.2 Pure MPC

Strengths:

- Explicit constraint handling
- Predictive planning

Weaknesses:

- High computational cost
- Often unnecessary in low-risk regions

7.3 Hybrid LQR–MPC

Strengths:

- Safety comparable to MPC
- Efficiency close to LQR
- Scales better to complex environments

Trade-offs:

- Requires careful switching design
- Additional supervisory logic

8 Detailed Engineering To-Do List

Phase 1: Foundations

1. Select robot model and simulation environment
2. Derive linear or linearized dynamics
3. Implement LQR baseline

Phase 2: Safety Controller

1. Formulate MPC optimization problem
2. Select solver and horizon
3. Implement warm-start strategy

Phase 3: Risk Supervisor

1. Implement distance-based risk metric
2. Implement predictive constraint violation check
3. Add uncertainty-aware risk metric (optional)

Phase 4: Hybrid Integration

1. Implement switching logic
2. Add hysteresis and dwell-time
3. Validate stability in long simulations

Phase 5: Experiments

1. Open-space navigation
2. Static cluttered environment
3. Dynamic obstacle scenarios
4. Noise and uncertainty injection

Phase 6: Evaluation and Analysis

1. Compare LQR, MPC, and hybrid controllers
2. Measure collision rate, tracking error, control effort
3. Analyze computation time and switching frequency

9 Performance Metrics

Performance evaluation is a central contribution of this project. Metrics are chosen to quantify not only safety and accuracy, but also efficiency and practicality.

9.1 Collision Rate

Collision rate measures the frequency of collisions or near-collisions during navigation tasks. It directly reflects safety performance.

Interpretation:

- Low collision rate indicates effective risk handling
- Hybrid controller should match pure MPC

9.2 Tracking Error

Tracking error measures deviation from the reference trajectory.

Interpretation:

- LQR typically yields low tracking error
- Hybrid controller should preserve LQR-level accuracy

9.3 Control and Energy Effort

Control effort quantifies actuator usage and indirectly reflects energy consumption.

Interpretation:

- Excessive MPC usage increases energy cost
- Hybrid control should reduce unnecessary actuation

9.4 Computation Time

Computation time per control step is critical for real-time feasibility.

Interpretation:

- Pure MPC serves as an upper bound
- Hybrid controller should significantly reduce average computation

9.5 Switching Statistics

Additional metrics include:

- Frequency of MPC activation
- Duration of MPC usage
- Correlation between risk metrics and switching events

These metrics provide insight into the effectiveness of the risk-aware supervisor.

10 Implementation Details

The goal is to ensure that the approach is not only theoretically sound, but also directly reproducible in simulation and transferable to real robotic systems.

10.1 Implementation Environment and Toolchain

The primary implementation target is a **single ground mobile robot** in a simulated environment.

Simulation environment:

- ROS 2 (Foxy/Humble) for system integration
- Gazebo for physics-based simulation
- Python or C++ for controller implementation

Alternative rapid-prototyping environment:

- MATLAB/Simulink with MPC Toolbox

ROS–Gazebo is chosen for the main implementation because it:

- Closely matches real robotic deployment
- Supports modular node-based architectures
- Allows easy extension to multi-robot scenarios

10.2 High-Level Software Architecture

The system is implemented as a collection of ROS nodes communicating through topics and services. Each node has a clearly defined responsibility.

- **State Estimation Node**
- **Environment Perception Node**
- **LQR Controller Node**
- **MPC Controller Node**
- **Risk Evaluation and Supervisor Node**
- **Actuation Interface Node**

This modular design enables independent testing and debugging of each component.

10.3 State Estimation

The state estimation node provides the system state required by both controllers.

Inputs:

- Odometry (wheel encoders or simulated ground truth)
- IMU (optional)

Outputs:

- Estimated state vector (position, velocity, orientation)
- Optional state covariance

For initial implementation:

- Perfect state feedback or simulated odometry is used

For extended experiments:

- An Extended Kalman Filter (EKF) can be integrated
- Covariance estimates are forwarded to the risk supervisor

10.4 Environment Perception and Obstacle Representation

The perception node processes sensor data to identify obstacles.

Inputs:

- Simulated LiDAR or range sensors

Processing:

- Convert raw range data into obstacle positions
- Inflate obstacles based on safety margin and uncertainty
- Maintain a local obstacle map

Outputs:

- Obstacle list or occupancy grid

Only local obstacles within a predefined radius are considered to keep computation bounded.

10.5 System Model and Linearization

A discrete-time state-space model is used for both LQR and MPC.

Steps:

1. Define continuous-time kinematic or dynamic model
2. Linearize around the current operating point
3. Discretize using zero-order hold

The same model is shared by both controllers to ensure consistent behavior.

10.6 LQR Controller Implementation

The LQR controller runs continuously and serves as the default controller.

Implementation details:

- Solve the discrete-time algebraic Riccati equation offline
- Store the feedback gain matrix
- Compute control input at every control cycle

Control rate:

- Typically 50–100 Hz

The LQR controller does not consider obstacles explicitly and assumes operation in low-risk regions.

10.7 MPC Controller Implementation

The MPC controller is designed for short-horizon, safety-critical operation.

Formulation:

- Quadratic cost function (state tracking and control effort)
- Linear state and input constraints
- Obstacle avoidance constraints

Solver choices:

- OSQP for quadratic programming
- CasADi with IPOPT for nonlinear extensions

Design choices:

- Short horizon (5–10 steps)
- Warm-start using LQR rollout
- Soft constraints for numerical robustness

Control rate:

- 10–20 Hz

10.8 Risk Evaluation and Switching Supervisor

The supervisor node evaluates risk and decides which controller is active.

Risk metrics implemented:

- Minimum distance to obstacles
- Predicted constraint violation under LQR rollout
- Growth of estimation uncertainty (optional)

Switching logic:

- If risk exceeds threshold → activate MPC
- If risk remains low for sufficient time → revert to LQR

Stability safeguards:

- Hysteresis thresholds
- Minimum dwell-time in each mode

10.9 Control Arbitration and Actuation

The actuation node receives commands from the supervisor.

Logic:

- If LQR active, forward LQR control
- If MPC active, forward MPC control

Smooth transitions are ensured by:

- Initializing MPC with LQR predictions
- Clipping control inputs within safe bounds

10.10 Execution Timeline and Data Flow

At each control cycle:

1. State estimator updates system state
2. Perception node updates obstacle information
3. Risk supervisor evaluates risk
4. Appropriate controller is selected
5. Control input is computed and applied

This loop runs continuously throughout the navigation task.

10.11 Logging and Experiment Management

All relevant data is logged for offline analysis:

- State trajectories
- Control inputs
- Risk metrics
- Switching events
- Computation time per step

Logs are used to compute performance metrics and generate comparative plots.

10.12 Extension to Real Hardware

The same architecture can be deployed on a physical robot by:

- Replacing simulated sensors with real sensor drivers
- Ensuring real-time solver performance
- Adding safety stop mechanisms

No architectural changes are required, only interface substitutions.

10.13 Implementation Summary

The implementation emphasizes:

- Modularity
- Real-time feasibility
- Clear separation of concerns

This design ensures that the hybrid LQR–MPC framework is robust, extensible, and suitable for both academic evaluation and real-world deployment.

11 Recent Key Literature

This project builds upon recent advances in hybrid and compositional control methods that combine local optimal control with receding-horizon optimization. The most relevant and recent works are summarized below.

- **Kong, N., Li, C., Council, G., Johnson, A. M.,** *Hybrid iLQR Model Predictive Control for Contact-Implicit Stabilization on Legged Robots,* **IEEE Transactions on Robotics**, 2023.
IEEE Xplore: <https://ieeexplore.ieee.org/document/10252162>
arXiv: <https://arxiv.org/pdf/2207.04591>
Why relevant: This paper demonstrates that local optimal control methods (iLQR, closely related to LQR) can be effectively combined with receding-horizon MPC for real-time control of complex robotic systems. It provides strong evidence that hybrid local–global control architectures are feasible on hardware, motivating the use of LQR-like controllers as efficient default modes within an MPC-based safety framework.
- **Le Cleac'h, S., Howell, T. A., Schwager, M., Manchester, Z.,** *Fast Contact-Implicit Model Predictive Control,* **IEEE Transactions on Robotics**, 2024.
IEEE Xplore: <https://ieeexplore.ieee.org/document/10384795>
arXiv: <https://arxiv.org/abs/2107.05616>
Why relevant: This work focuses on reducing the computational burden of MPC for constrained and contact-rich robotic tasks. The techniques for improving solver efficiency and real-time performance are directly applicable to designing a lightweight MPC mode that is activated only during high-risk navigation phases in a hybrid LQR–MPC framework.
- **Wu, F., et al.,** *Composing MPC with LQR and Neural Networks for Real-Time Hybrid Control,* arXiv preprint, 2021.
arXiv: <https://arxiv.org/pdf/2112.07238>
Why relevant: This paper explicitly studies controller composition, where MPC is combined with computationally cheaper controllers such as LQR and learned surrogates. It supports the idea that MPC need not be run continuously, and that structured fallback controllers can significantly reduce computational cost while maintaining stability.
- **Awad, N., Lasheen, A., Elnaggar, M., et al.,** *Model Predictive Control with Fuzzy Logic Switching for Path Tracking of Autonomous Vehicles,* 2022.
ScienceDirect: <https://www.sciencedirect.com/science/article/pii/S001905782100639X>
Why relevant: This work demonstrates a practical and interpretable switching mechanism between controllers using fuzzy logic. While the switching is heuristic and reactive, it provides a useful baseline for comparison with more structured, predictive, and risk-aware switching strategies proposed in this project.

12 Conclusion

This project proposes a principled, risk-aware hybrid LQR–MPC navigation framework that balances safety and efficiency. By explicitly quantifying risk and systematically evaluating performance trade-offs, the work provides both practical insights and research contributions suitable for academic dissemination.