CSC 59866-E: Senior Project I
AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

March 11, 2026 CSC 59866

Model Predictive Control (MPC): Real-time constraints, mistake correction and daily control tasks by AI Agents

Logistics and Motivation for Control Theory

Recall Lecture 12: We explored Offline RL and Inverse RL as ways to safely learn from data without dangerous trial-and-error in the real world.

The New Paradigm: What if we actually *know* the physics of the environment? Do we still need to blindly learn a policy, or can we just plan the best path mathematically?

Today’s Agenda

What is MPC? The theory of receding horizon control.

The Mathematics of Planning: State-space dynamics and cost functions.

Interactive Problem: Solving a 1-step prediction horizon.

Real-time Constraints: Hardware limits and mistake correction.

Modern AI & MPC: How Deep RL and MPC are merging.

Multimodal Agentic Model Predictive Control: How MPC can leverage modern AI models.

MPC Core Concepts

What is Model Predictive Control

The Intuition: Imagine playing chess. You don't just memorize a rulebook; you look at the board, predict your opponent's next 5 moves, decide on a sequence of actions, make *only your first move*, and then evaluate the board again.

The Definition: MPC uses a mathematical model of a system to optimize a sequence of future control actions over a finite time window.

The Shift: Unlike RL which learns a reactive "reflex" (Policy $\pi(a|s)$) ahead of time, MPC solves a complex optimization problem *live*, on the fly, at every single time step.

MPC vs. RL

Reinforcement Learning:

- **Pros:** Extremely fast at runtime (just a forward pass through a neural network). Can learn unknown environments.
- **Cons:** Hard to guarantee absolute safety or mathematically prove it won't crash. Takes weeks to train.

Model Predictive Control:

- **Pros:** Perfect safety guarantees. Hard constraints (e.g., "NEVER steer past 30 degrees") are strictly enforced.
- **Cons:** Computationally expensive. Requires a highly accurate physical model of the environment.

Mathematics of MPC

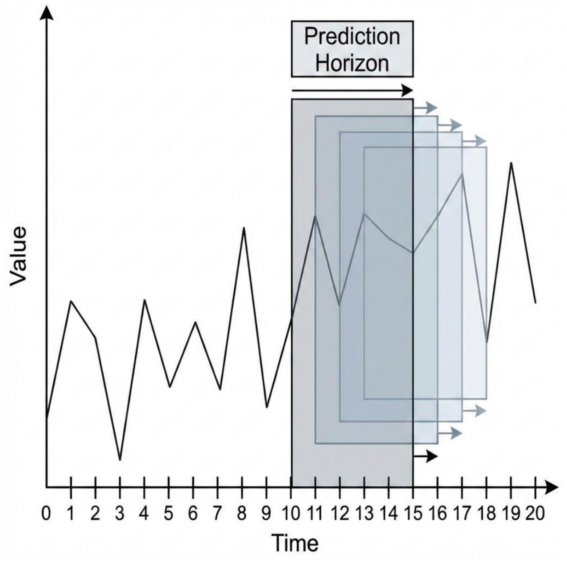
The "Receding Horizon" Strategy

Step 1: Predict: At time t , look ahead for N steps (the Prediction Horizon).

Step 2: Optimize: Calculate the mathematically perfect sequence of actions for those N steps to reach the goal.

Step 3: Execute ONE Step: Apply *only* the first action u_t from that sequence to the physical system. Throw the rest of the plan away!

Step 4: Shift & Repeat: Move to time $t+1$, read the new sensors, and plan the next N steps all over again.



System Dynamics

To predict the future, the agent needs a model of the universe. In control theory, we use **State-Space Equations**.

For a discrete-time linear system, the future state is a function of the current state and the chosen action:

$$x_{t+1} = Ax_t + Bu_t$$

- x_t **(State Vector):** Where the agent is (e.g., position, velocity, angle).
- u_t **(Control Input Vector):** What the agent does (e.g., steering, throttle).

A **&** B **(System Matrices):** The physics of the world. How does momentum carry the agent forward (A)? How does the throttle affect velocity (B)?

The Cost Function

- How does the agent know which path is "best"? It minimizes a **Cost Function** (J), which is the inverse of RL's Reward Function.
- A standard quadratic cost function looks like this:

$$J = \sum_{k=0}^{N-1} \left(x_{t+k}^T Q x_{t+k} + u_{t+k}^T R u_{t+k} \right) + x_{t+N}^T P x_{t+N}$$

- **State Penalty (Q):** Punishes the agent for being far from the goal.
- **Control Penalty (R):** Punishes the agent for using too much energy (e.g., slamming the gas and brakes wastes battery).
- **Terminal Cost (P):** An estimate of the remaining cost after the horizon N ends (similar to a Value function in RL!).

The Optimization Problem

At every single millisecond, the AI agent must solve this constrained problem:

$$\min_{u_t,\ldots,u_{t+N-1}} \quad J$$

Subject to:

1. $x_{k+1} = f(x_k, u_k)$ *(Physics cannot be broken)*
2. $x_{\min} \leq x_k \leq x_{\max}$ *(State constraints: Stay inside the lane!)*
3. $u_{\min} \leq u_k \leq u_{\max}$ *(Actuator constraints: Steering wheel cannot turn more than 40 degrees!)*

MPC Toy Example

Let's solve an MPC step by hand!

- **The Agent:** A 1D sliding robot trying to reach the origin ($x = 0$).
- **Current State:** $x_t = 5$
- **The Physics (Dynamics):** $x_{t+1} = x_t + u_t$
- **The Cost Function (1-step horizon):** We want to reach the origin, but save energy.

$$J = (x_{t+1})^2 + (u_t)^2$$

- **Your Task:** Using calculus, we will find the optimal control action u_t that minimizes J

Solution

Step 1: Substitute the physics into the cost function.

$$J = (x_t + u_t)^2 + (u_t)^2$$

$$J = (5 + u_t)^2 + u_t^2 = 25 + 10u_t + 2u_t^2$$

Step 2: Take the derivative with respect to u_t and set to 0 (to find the minimum).

$$\frac{dJ}{du_t} = 10 + 4u_t = 0$$

Step 3: Solve for u_t .

$$4u_t = -10 \implies u_t = -2.5$$

The Result: The optimal action is to move -2.5 units. The robot's next state will be $x_{t+1} = 5 - 2.5 = 2.5$ It moves halfway to the goal to balance speed and energy!

Real-Time Constraints & Mistake Correction

Mistake Correction

If our plan on step t gives us a perfect 10-step path, why do we throw away the last 9 steps and recalculate at $t+1$?

1. **Disturbances:** Wind gusts, road friction, or a bumped sensor change the state unexpectedly.
2. **Model Mismatch:** Our matrix A is just an approximation. The real world is non-linear and chaotic.

By re-reading the sensors at $t+1$ and re-optimizing, MPC acts as a **Feedback Controller**. It instantly notices the drone was blown off course and corrects the mistake.

Real-Time Constraints

Solving a constrained quadratic optimization problem is computationally heavy.

Recall Lecture 7 (Latency): If your autonomous vehicle is moving at 65 mph (30 meters/second), and your MPC solver takes $200\mathrm{ms}$ to find the optimal path, the car has already moved 6 meters before the brakes are applied!

The Engineering Challenge: We must carefully tune the Prediction Horizon (N). A longer horizon allows smarter planning (seeing the sharp turn early) but increases computation time exponentially.

Real-World Control Tasks

HVAC / Smart Thermostats: Predicting weather and thermal inertia of a building to optimize heating hours in advance to save electricity costs.

Chemical Plants: Controlling fluid levels and temperatures safely across hours-long reaction times.

Autonomous Vehicles: Self-driving companies (like Waymo) use MPC for trajectory generation. The agent calculates smooth, comfortable steering and braking paths while strictly enforcing hard safety constraints like "stay within the lane" and "do not hit the pedestrian."

Integrating MPC with RL

How do we combine the fast adaptability of RL with the strict safety of MPC?

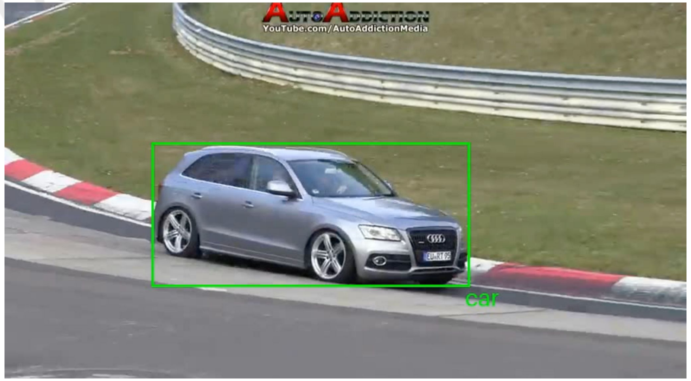
Learned Dynamics Models: For complex systems (like a quadruped robot walking on ice), the physics (A and B) are too hard to program. We use Deep Learning to *learn* the dynamics model $f_\theta(x, u)$, and use MPC to plan through the neural network!

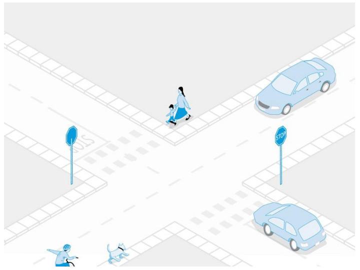
Value Function as Terminal Cost: We can use RL to train a highly accurate Value Function, and use that as the Terminal Cost (P) in our MPC equation, allowing the MPC to use a very short (fast) horizon but still act optimally.

Multimodal Agentic Model Predictive Control

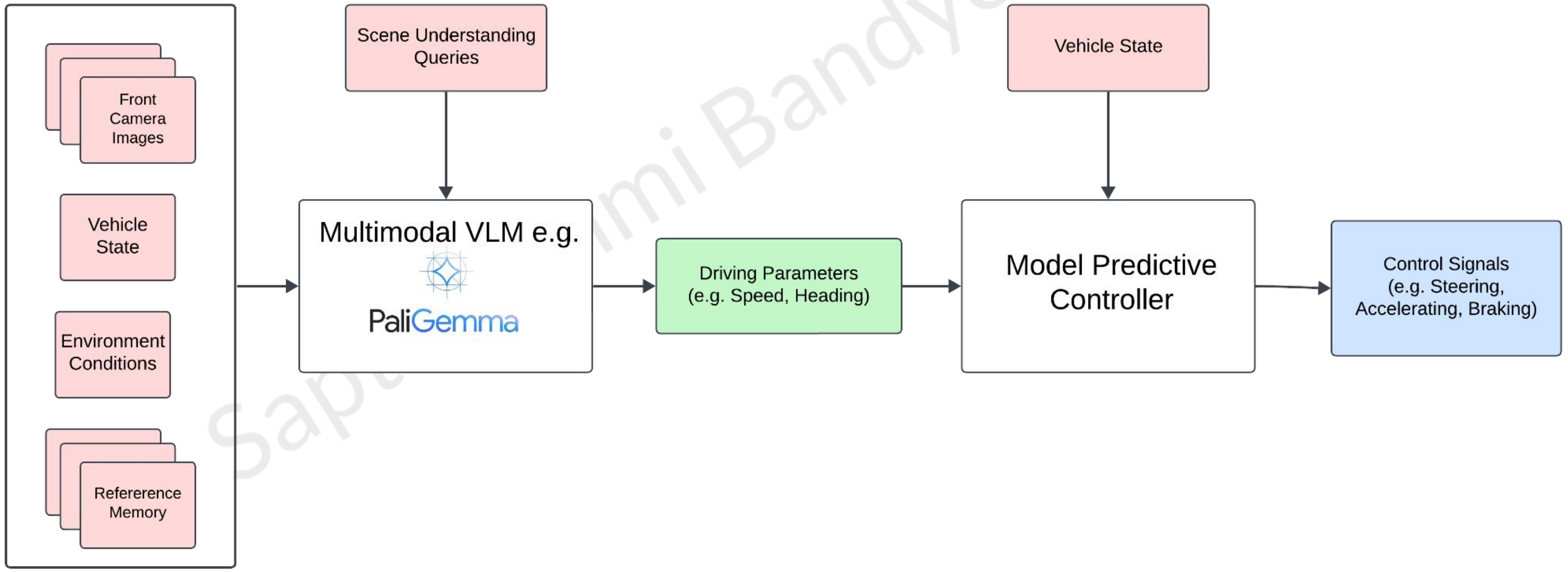
Research Question for Self-Driving Cars

- How can we train Autonomous Vehicle (e.g. Drone, Car) agents better with fine-grained model-predictive control (MPC)?
    - Can we improve fine-grained control of demonstrations on which vehicles train with Model Predictive Control?
    - Can we improve semantic understanding with VLMs?
    - How do we combine VLM technology with traditional control?
    - How can we use the “world knowledge” of multimodal AI Agents?

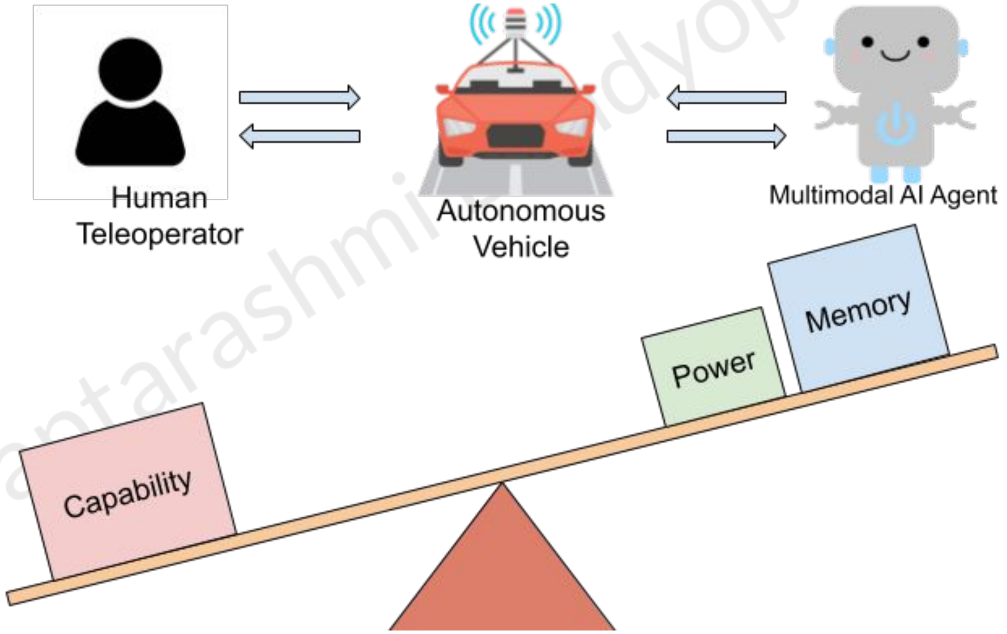




VLMs with Model Predictive Control for Self-Driving Cars


Front Camera Images
Vehicle State
Environment Conditions
Refererence Memory
Scene Understanding Queries
Multimodal VLM e.g.
PaliGemma
Driving Parameters (e.g. Speed, Heading)
Vehicle State
Model Predictive Controller
Control Signals (e.g. Steering, Accelerating, Braking)

Performance vs Efficiency of Embedded AI Agents



Integration of VLM Insights into Model Predictive Control

- With Vision-Language Models, we have access to a new domain of contextual information through semantic querying
  - “Are there any pedestrians in front”?
  - “How fast is the car to the left going”?
- These insights can be turned into actionable driving parameters for the MPC component, including driving angle and acceleration
- MPC safety constraints provide guarantees that VLMs alone do not

Why Combine VLMs with MPC?

- Have more **context-aware decision-making** and **improved safety** in complex and dynamic situations
- Be able to address **real-time decision making problems** with powerful AI agency
- Leverage **pre-trained knowledge to handle unforeseen circumstances** similar to driving experience
- Improving representations of interactions of agents with humans by using language-based signals

Finding Opportunities from Current Challenges

- Addressing Ethical Considerations: To engage in interdisciplinary research to tackle the ethical implications of using AI for critical decision-making in autonomous vehicles.
- Standardizing Evaluation Metrics: To develop relevant benchmarks and evaluation frameworks that can assess the performance of integrated VLM-MPC systems across diverse scenarios.
- Fostering Collaboration: To encourage partnerships between academia, industry, and regulatory bodies to accelerate the development and deployment of these advanced autonomous systems.
- The fusion of VLMs with MPC:
    - Promises to advance the field of autonomous vehicle control
    - Opens up new possibilities for intelligent autonomous agents across various domains

Questions?