Last Time

☐ latent & semantic safety

This Time

☐ runtime monitoring & recovery via VLMs

lecture 11

EAIS SP'26

Andrea Bajcsy

On Runtime Monitoring & Fallbacks

Safety filtering is one strategy for doing runtime monitoring & for computing a fallback policy:

$V^{safe}(x)$ ← this is our <u>runtime monitor</u> b/c it tells us if we should be applying our fallback policy

$\pi^{safe}(x)$ ← this is our <u>fallback policy</u> b/c it keeps our robot safe when we get close to danger.

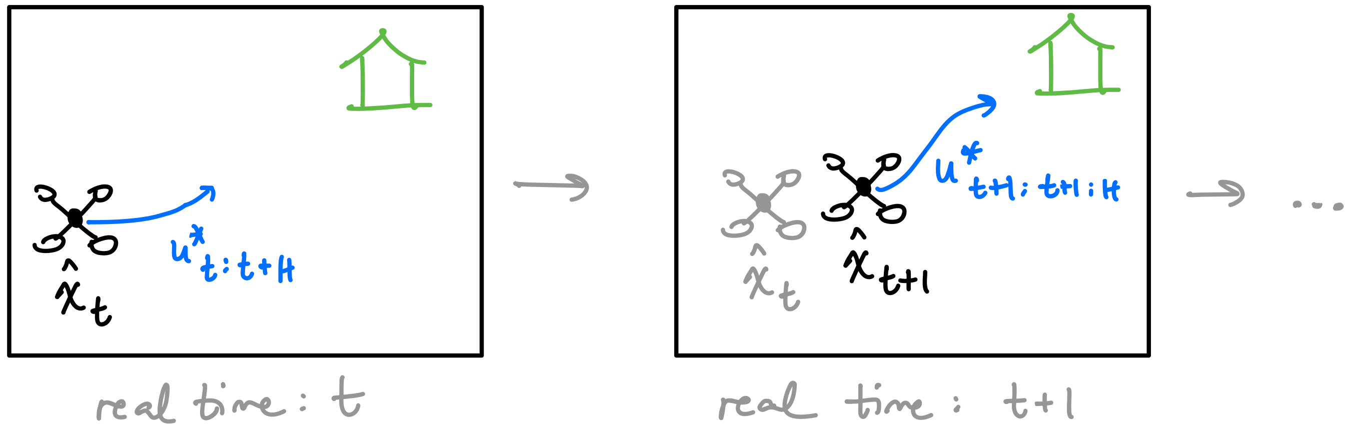
<u>BUT</u> so far we have assumed that the monitor & fallback are pre-computed offline; but, as we saw in the online updates lecture, if the conditions or safety spec. changes online, $V(x)$ and $\pi(x)$ may not be valid anymore.

$\boxed{Q}$ What is another strategy we could take for designing a runtime monitor + a fallback policy?

$\boxed{A}$ Let's take an <u>online planning / control approach</u> - specifically

<u>Model Predictive Control (MPC)</u>: Idea is simple

1. measure the current state $\hat{x}_t$,
2. optimize a (locally optimal) trajectory from $\hat{x}_t$ → get $u^*_{t:t+H}$
3. execute just the first action, u^*_t
4. let the real system evolve for one step to x_{t+1}
5. repeat!



Under the hood, we are still solving an opt. problem:

$$\begin{aligned} \underset{\text{traj.}}{\text{minimize}} \quad & \text{Cost}(\text{trajectory}) \\ \text{s.t.} \quad & \text{dynamics} \\ & \text{initial state} = \hat{x} \end{aligned}$$

But we are typically re-solving it online starting only from current $\hat{x}$, solving it locally, and doing it for shorter time horizon $(H \ll T)$

$\uparrow$ MPC horizon

$\leftarrow$ task horizon

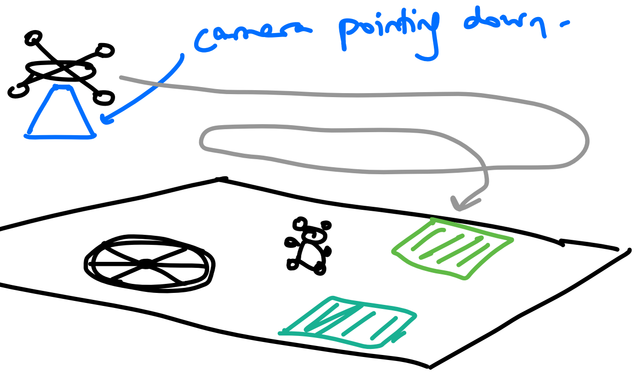
TODAY Two case studies on how MPC can be used to do runtime monitoring + fallback planning

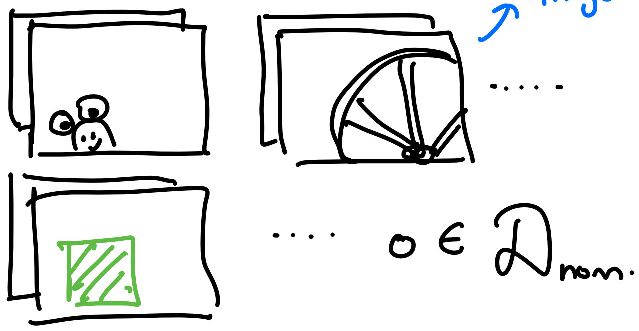
$\Rightarrow$ BONUS: we will see VLMs/LLMs pop up to help us "generalize" MPC ingredients.

CASE STUDY 1: Sinha et. al. "Real-Time Anomaly Detection & Reactive Planning w/ LLMs!" Robotics: Science & Systems, 2024.

Best Paper Award! $\leftarrow$

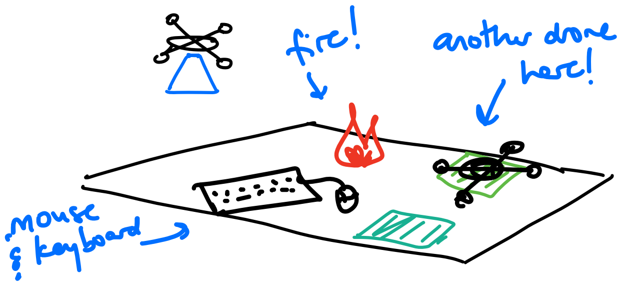
We will use a running example throughout of an autonomous drone navigating through environment:





Now, you want to let your robot operate/fly when conditions are nominal, but if anything goes "wrong" we want to have a fallback -- ex. landing in grassy patch; hovering, etc.

↳ maybe you trained your system here, maybe its previous deployment

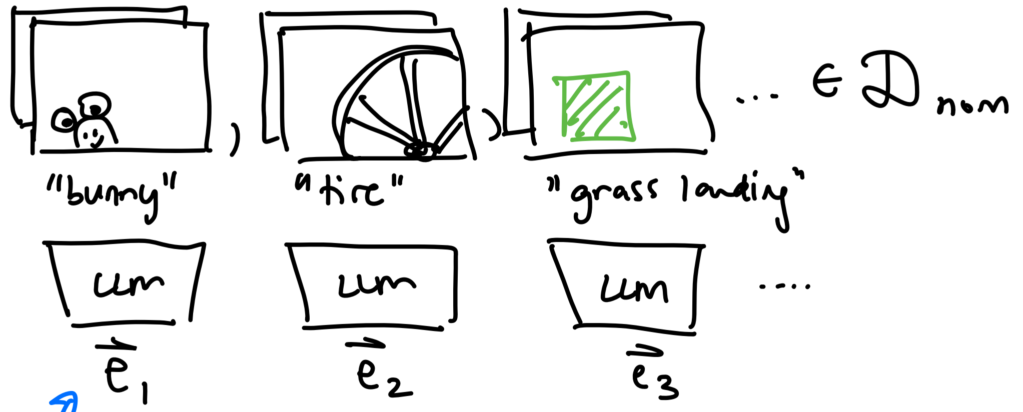


↳ ① How to detect situation isn't nominal (i.e. monitor)?

↳ ② How to plan a recovery (fallback)

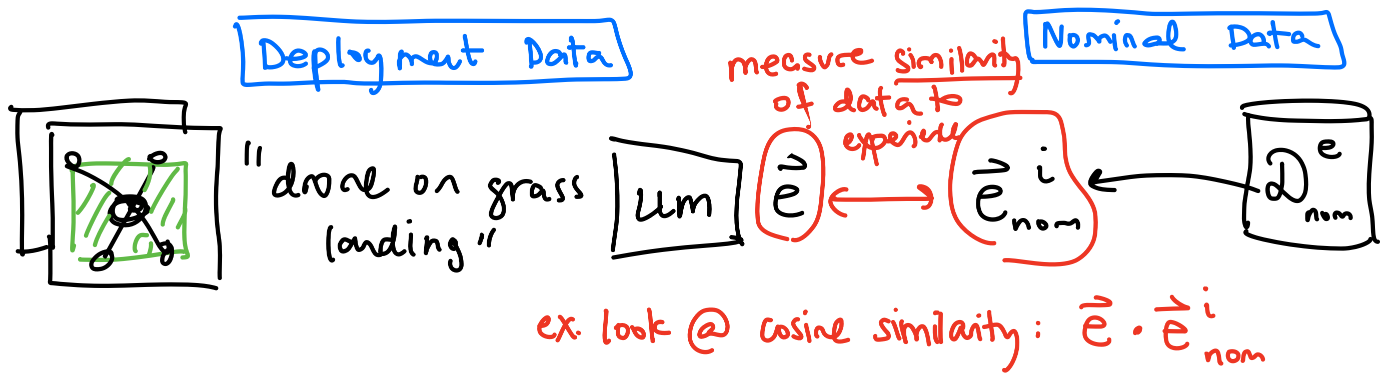
① monitoring as Semantic Anomaly Detection:

Concider example above - how can we detect such weird open-ended scenarios? Here, let's take the notion of semantic anomaly detection + heart & use commonsense knowledge of LLMs to detect these hazards!



textual embedding vectors of the robot's prior experiences!

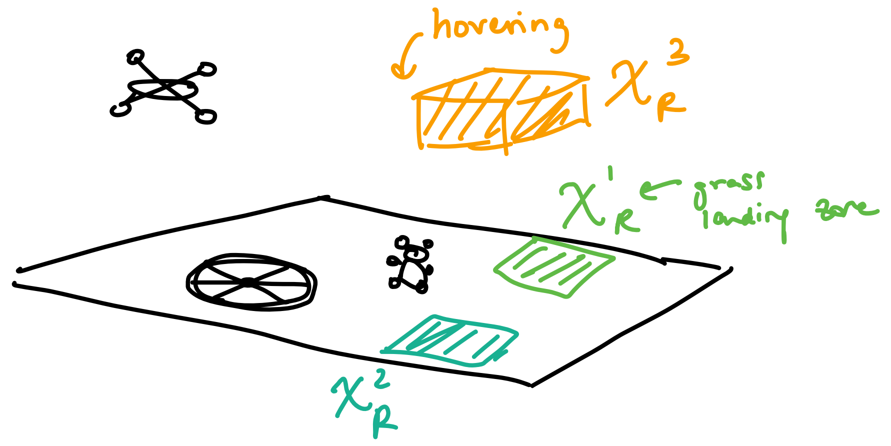
Now, construct a runtime monitor which checks how similar the textual description of a deployment scene is with a nominal scene!


Deployment Data
"drone on grass landing"
LLM
$\vec{e}$
measure similarity of data to experience
$\vec{e}^{\,i}_{nom}$
Nominal Data
$\mathcal{D}^e_{nom}$
ex. look @ cosine similarity: $\vec{e} \cdot \vec{e}^{\,i}_{nom}$

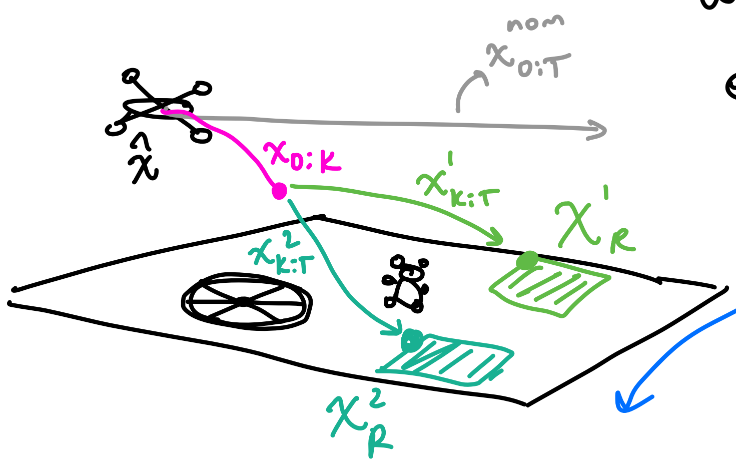
② Fallback Planning as an MPC Problem:

What should we do if anomalous? Consider N recovery regions

note: technically these should be control invariant: i.e. once you get here, you can stay here for all time.



Our fallback planner will solve a special MPC problem where we keep multiple branches for each recovery behavior!



$$\underset{\vec{x}}{\text{minimize}} \quad \text{Cost}(x^{nom}_{0:T})$$

s.t. dynamics

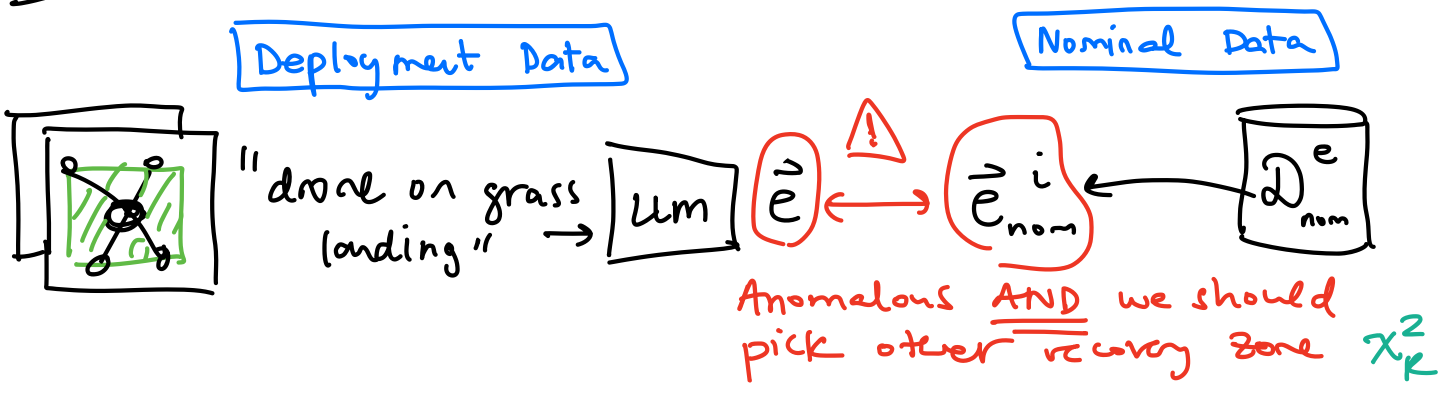
$x^1_T \in \mathcal{X}^1_R$ // end recovery 1 in zone 1

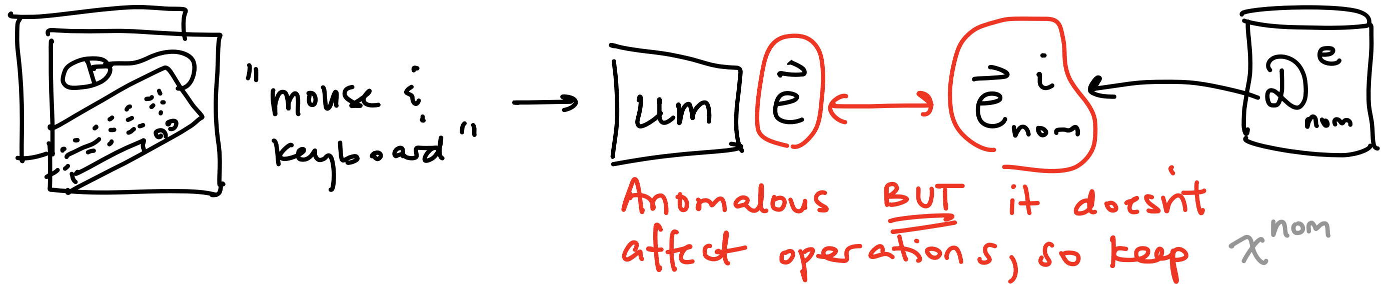
$x^2_T \in \mathcal{X}^2_R$ // end recovery 2 in zone 2

$$\vec{x} = \{ x^{nom}_{0:T}, x_{0:K}, x^1_{K:T}, x^2_{K:T} \}$$

The pink traj. $x_{0:k}$ is shared by all recovery policies & allows the method to call the LLM one more time for slower reasoning about the anomalies, figuring out if it really should activate any recovery behavior, or if its ok & it should return to nominal operation.

ex.



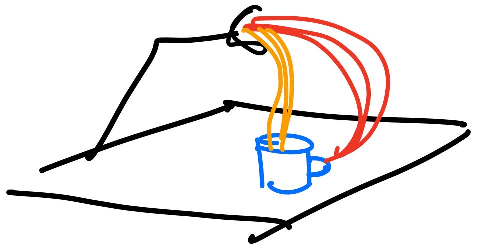


CASE STUDY 2: Wu et al. "VLM-in-the-loop Policy Steering" Robotics: Science & Systems, 2025.

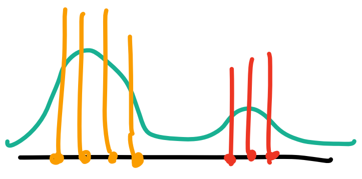
Recovery zones made sense in the drone nav. example, but it's a bit strange in domains like manipulation...

BUT the notion of monitoring, fallback, & MPC is still relevant!

We will use running example of manipulator which has learned some behaviors via a diffusion policy.



Here, the diffusion policy has learned to approximate a multimodal distribution over actions:



$a_{t:t+H}$

learned distribution & each visualized traj. is one sample

$$\mathbb{P}(\underline{a_{t:t+H}} \mid o_t)$$

learned from expert demos

RGB imgs.

future sequence of end-effector poses

Usually, we just sample $\vec{a} \sim \mathbb{P}(\vec{a} \mid o_t)$ & execute — this is SOTA! But, not all samples are created equal: some fail, some are inappropriate for the context (ex. putting gripper inside mug).

Let's pose a type of mpc problem which will let us do monitoring + fallback:

final action sample is verified before exec

cost function does our "monitoring"

$$a^* = \underset{a^i \in \{a^1, a^2, a^3, a^4, a^5, a^6\}}{\text{minimize}} \quad C(o_{t:t+H}; \mathcal{L}) \quad \circledast$$

language description of task.

set of samples I take from $\mathbb{P}(a_{t:t+H} \mid o_t)$

$$\text{s.t.} \quad o_{t:t+H} \sim \mathbb{P}(o_{t:t+H} \mid a^i, o_t)$$

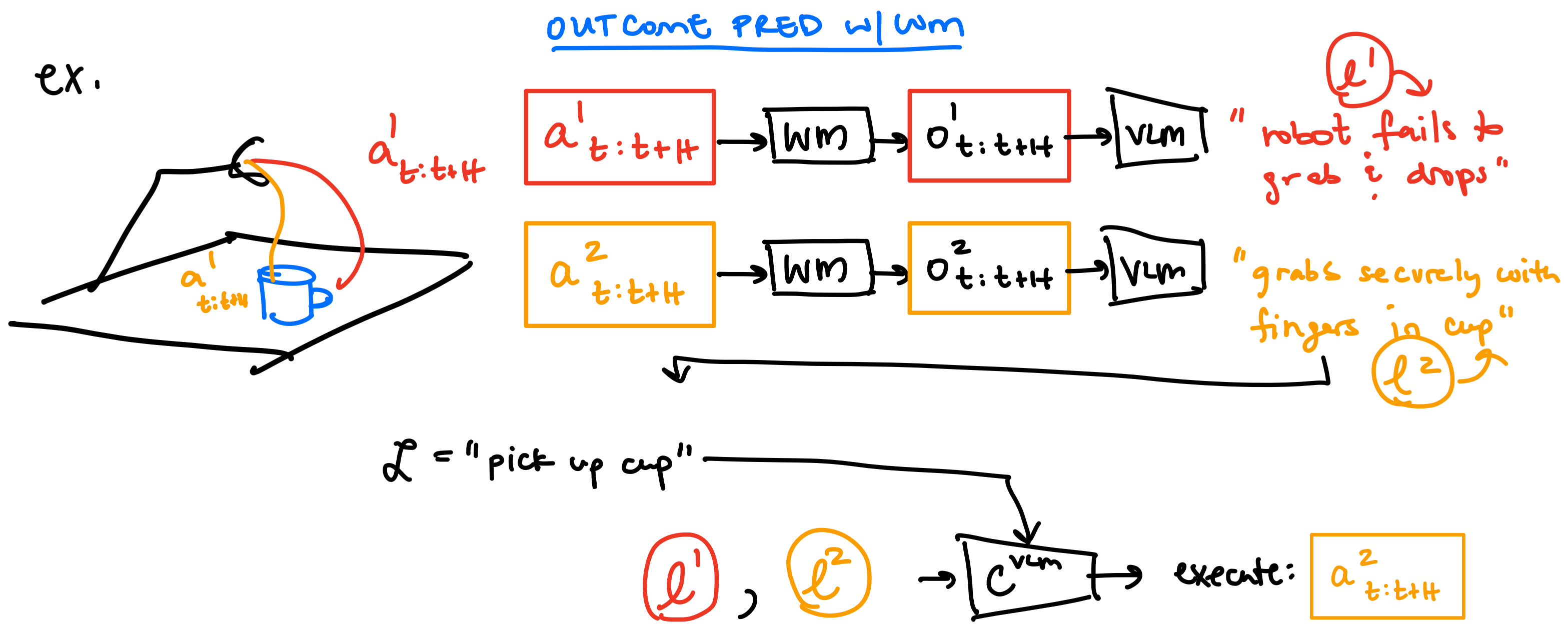
initial obs

future obs.

⚠ maybe we would want to use vlm as our cost function, but we need to translate each action to (observ.) outcomes for evaluation! This is the "model" in model predictive control (MPC).

In this work, we have similar philosophy to prior work: if we can translate everything into text represent. then we can use the VLM's reasoning as an adaptable, "semantically-aware" cost function.

ex.



If you think about it, the VLM is now responsible for implicitly solving this MPC problem b/c it not only evaluates but also selects the minimizer of ⊛ b/c it selects the textual description that corresponds to the "best" action sample.