AuSRoS Hands-on Workshop: High-level Planning to Low-level control

Part 1: Planning with learnt locomotion in Isaac Sim

Part 2: Deep Dive - Training locomotion policies in Isaac Lab

Goal: Connect planning algorithms to robots running in simulation

Part 1:

Planning with Learned Locomotion in Isaac Sim

- Connecting RRT/MCTS planners to learned locomotion policies on bipedal/quadruped robots
- Bridging the gap between high-level geometric path planning and low-level dynamic execution.

The Simulation Ecosystem

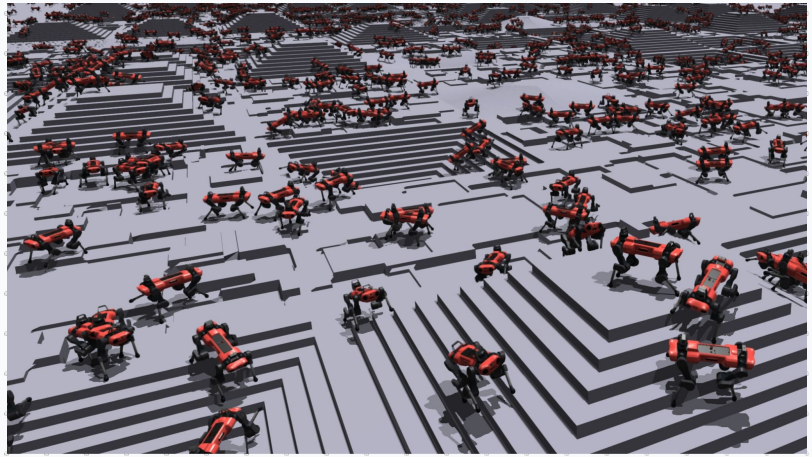
Isaac Sim vs. Isaac Lab

Isaac Sim (Physics & Deployment)

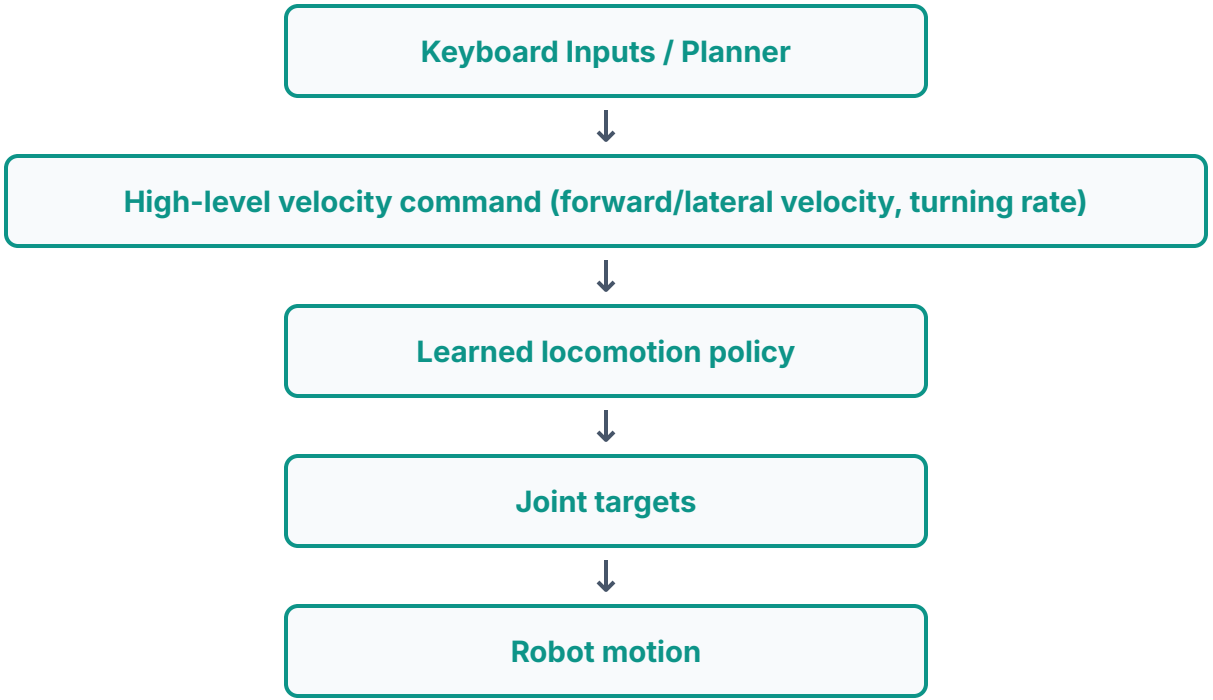
Handles physics simulation, deployment testing, and complex sensor simulation.

Isaac Lab (RL Training)

Provides high-throughput reinforcement learning environments, utilizing PPO for massive parallel policy training.



Isaac Sim Quick Look + Learned Locomotion Policy Example

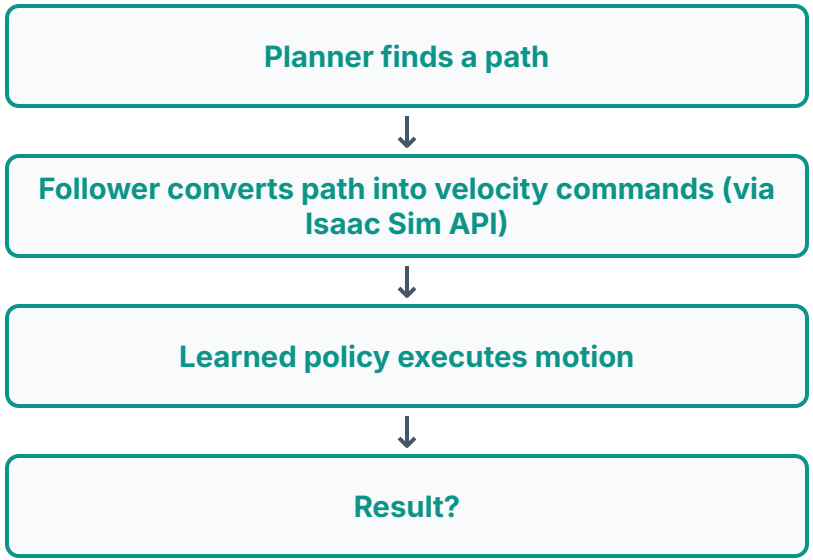


Demo: Step 1

We have a robot which knows how to walk.

How can we utilize what we have learned to make this robot reach a goal?

Geometric RRT - When a Planned Path Meets a Robot



Observation

A geometric path may be valid.

But, execution also depends on tracking error, obstacle clearance, learned policy behaviour, etc.

Demo: Step 2, 3

Geometric RRT - Recap

Sparse Waypoint Following

Naive Approach

Take the next RRT waypoint and drive directly toward it.

Key Problem

Poor guidance results in erratic motion; the robot may cut corners or overshoot heavily.

Denser Waypoint Following

The Solution

Densify the planned path. Choose a dynamic lookahead target and steer smoothly towards it.

Result

Generates far smoother velocity commands and successfully reduces corner cutting.

How does RRT become Robot Commands?

1. Path Input

The RRT planner generates a geometric path composed of purely spatial coordinates:

`[(x1, y1), (x2, y2), ...]`

2. The Follower

Acts as the bridge.

It translates static waypoint positions into real-time physical actions by constantly computing target directions.

3. Robot Commands

The lower-level robot policy executes motion based on dynamic velocity requirements:

`[v_x, v_y, yaw_rate]`

Demo: Step 4

What is Kinodynamic Planning?

Geometric Planning

Core Question

"Can I draw a path?"

Focus

Solely asks if a collision-free connection exists in the configuration space, ignoring physical dynamics.

Kinodynamic Planning

Core Question

"Can the robot move this way?"

Key Considerations

- Position & orientation
- Velocity commands
- Short future motion

Geometric RRT vs Kinodynamic RRT

What does an RRT Edge mean?

Geometric RRT Edge

Definition

point A → straight line → point B

Focus

Connects configurations directly in space. Assumes the robot can instantly transition along a perfectly straight line segment with no physical constraints.

Kinodynamic RRT Edge

Definition

state + command + short rollout

Example:

```
state = (x, y, yaw)
command = [v_x, v_y, yaw_rate]
```

The edge is the result of applying a command for a short time, ensuring the path respects the robot's physical movement limits.

https://www.aorthey.com/assets/lectures/motion-planning/07_lecture.pdf

Implementation - Kinodynamic RRT Lite

Policy Rollouts Are Faithful but Slow

High-Fidelity Policy Rollout

Process

`command → policy → physics → motion`

Computational Bottleneck

Too slow to use for every RRT edge.

Time Estimates ≈ hours

Proposed Compromise

Approximation

`command → kinematic rollout`

Key Benefits

- Bypasses heavy physics simulation and policy evaluation steps.
- Reduces rollout calculation times from hours to milliseconds.
- Enables rapid exploration of the search space in real-time.

Demo: Step 5

Kinodynamic RRT Lite: Compounding Execution Errors

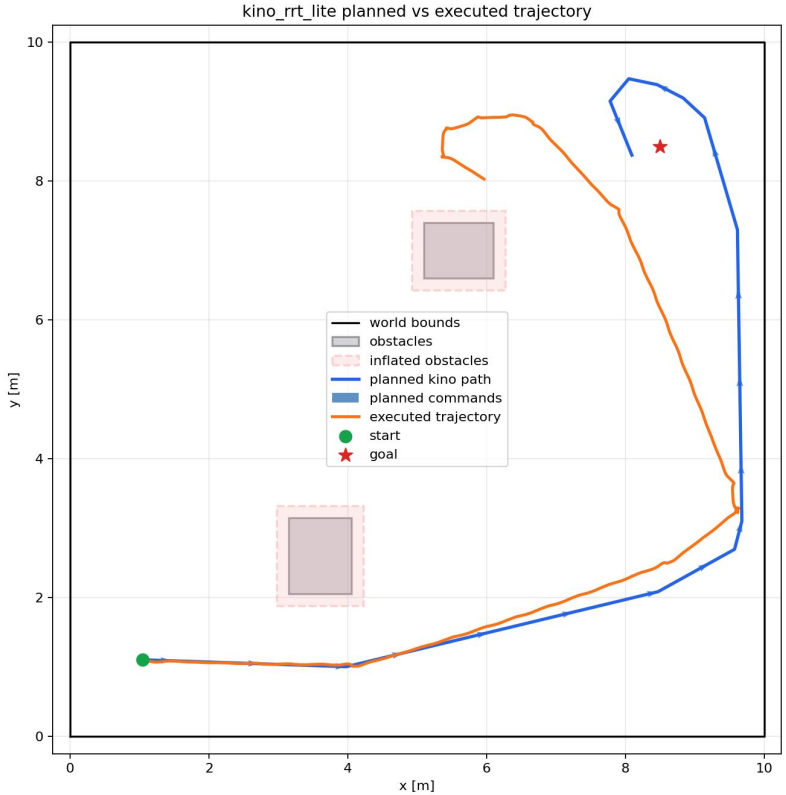
Why Divergence Occurs

THE DRIFT PHENOMENON

Bypassing full physics simulation during offline path planning results in kinematically feasible but dynamically unachievable trajectories.

KEY OBSERVATIONS

- **Planned Path (Blue):** Successfully finds a route to the goal state using simplified kinematic rollouts.
- **Executed Trajectory (Orange):** Progressively drifts due to unmodeled dynamics, failing to reach the red star goal.
- **Compounding Nature:** Small initial controller discrepancies accumulate over time, leading to noticeable tracking failure.



Offline planning to Online planning - MCTS

Traditional Approach

Offline: RRT Planning

RRT gives a plan once.

- Calculates the entire trajectory beforehand.
- Assumes perfect execution with no disturbances.
- Fails to adapt to dynamic drift or unexpected obstacle changes.

Why Online Planning?

Online: MCTS Replanning

MCTS replans repeatedly in a closed loop:

01 Get current robot pose

02 Try short command sequences

03 Score possible futures

04 Execute best command

The MDP behind MCTS

State

Robot pose, goal location, and local obstacle information.

Action

Sampled velocity command:

`[v_x, v_y,yaw_rate]`

Transition

Rollout model dynamically predicts the next state based on current action.

Reward

- Progress to goal
- Obstacle clearance
- Smooth commands
- No-go penalty

What does the reward encourage?

A good rollout should:

- move closer to the goal
- stay away from no-go regions
- avoid unnecessary turning
- avoid command jitter
- reach the goal if possible

Rollouts: Kinematic vs. Policy-in-the-Loop

Fast Approximation

Kinematic

```
command
→ simple kinematic model
```

Quickly approximates robot motion using direct mathematical models without simulating complex forces or physical friction.

High-Fidelity Rollout

Policy-in-the-Loop

```
command
→ learned policy
→ Isaac physics
```

Evaluates actions through a neural network policy operating directly inside a high-fidelity physical simulator like Isaac Sim.

Comparison

The Trade-off

Kinematic
Fast but approximate. Ideal for high-speed planning loops.

Policy-in-the-Loop
Faithful but slow. Essential for realistic contact dynamics.

Demo: Step 6

Main Takeaways

- A path is not yet robot behaviour.
- RRT gives geometry; the robot needs commands.
- Pure pursuit improves path-to-command tracking.
- MCTS searches short command futures.
- Policy rollouts are faithful but slower.

Transitioning to Part 2 of Workshop (After Break)

- **In this part:** path planning → velocity commands → learned locomotion
- **Next part:** Deep dive - training low-level locomotion policies