Starting on the Right Foot with Reinforcement Learning

bostondynamics.com/blog/starting-on-the-right-foot-with-reinforcement-learning

mrodin March 19, 2024

Spot's legged mobility enables it to bring advanced sensing and manipulation capabilities across a wide range of structured and unstructured environments. In our customer's hands, Spot performs multi-floor factory inspections, overcomes dust and debris during nuclear plant decommissioning, and navigates frozen arctic pit mines, among many other applications. Spot robots deployed across the world have cumulatively walked over 250,000km and continue operations day and night at a current rate of nearly the circumference of Earth every three months.

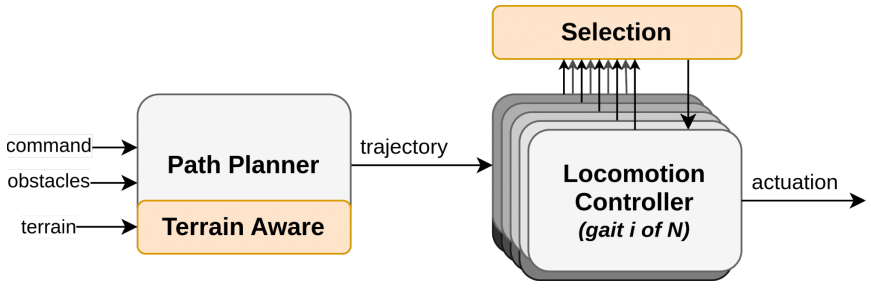
But how does Spot's walking control system actually work? The legged robots we've built at Boston Dynamics have historically leveraged Model Predictive Control (MPC), a control strategy that predicts and optimizes the future states of the robot in order to decide what action to take in the moment. These strategies work well when the controller's model behaves similarly to the physical system and when the robot's state, environment, and goals can be well represented as objectives and constraints in an optimization problem. These systems are typically intuitive, debuggable, and they allow us to tune robot behaviors to achieve both practical and aesthetic goals.

However in real-world conditions, there are situations that can be difficult to observe and accurately model. For example, to perform routine inspections in one customer facility, Spot has to traverse over the edge of a container used for liquid containment. Here, the robot must move over the obstacle without tripping or slipping on the potentially wet or greasy floor. There are many other examples like this, where the particular arrangements and conditions of real-world environments stresses the robot's control system. This raises the question: *how do we design control software that can be extended over time to handle more and more of these cases while not degrading performance in all other situations*?

Locomotion Control on Spot

Control systems for legged robots make decisions at multiple time scales. They decide what path the robot should take, what pattern of steps it should use to move along that path, and how to make high speed adjustments to posture and step timing in order to maintain balance. For example, carefully sequencing steps on stairs allows the robot to execute smooth motion while maintaining a sufficient margin from the stair edges to recover from uncertainty. Or when the robot slips, fast step recovery motions are critical to returning to stable locomotion.

Spot's production controller achieves this multi-scale decision making by evaluating many MPC controllers simultaneously. In less than a millisecond, Spot evaluates dozens of individual predictive horizons, each with its own unique step trajectory reference that considers both the robot and environment state (e.g., terrain roughness, presence of obstacles). To choose from among these many solutions, we've developed a system that can "score" the output of each of these controllers, selecting the one with the maximum value and using its output to control the robot.



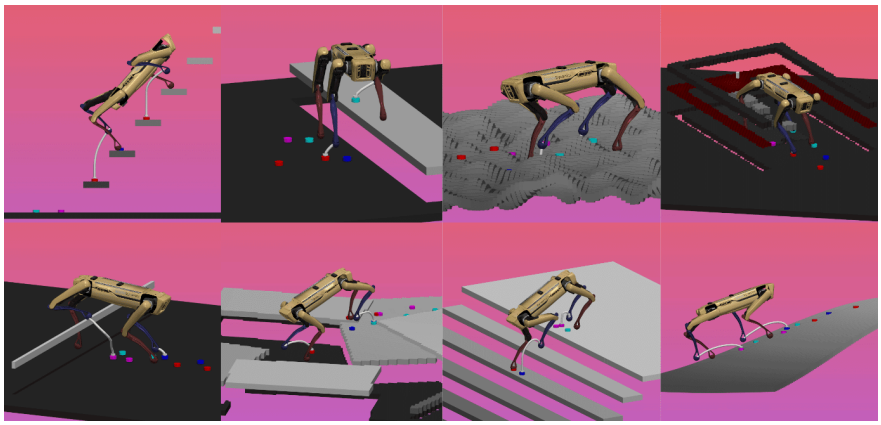
Legacy locomotion control system

This approach has yielded a mean time between falls of hundreds of hours, but it has a couple drawbacks. First, having to evaluate multiple MPC instances in parallel consumes significant onboard compute, which takes additional time and energy. Second, making changes to the selection function in order to address new failure modes can be nontrivial and risks inadvertently reducing performance in other situations. To address both of these challenges, we recently employed a data-driven approach to robot control design called reinforcement learning.

Reinforcement Learning for Locomotion

Traditional approaches to creating robot controllers require engineers to think carefully about the problem they want to solve, devise a detailed strategy for the robot to solve the problem in source code (e.g., C++), then run the compiled code on the robot to measure performance. The primary challenge with this approach is that the human programmers have to come up with the strategy to solve the problem, which can be hard to do for complex problems (like walking reliably all over the world).

Reinforcement learning (RL) is an alternative approach to programming robots that instead optimizes the strategy (i.e. the controller or policy) through trial and error experience in a simulator. In this setting, the policy is no longer written in human-readable source code, it is instead implemented in a neural network whose parameters are optimized by the RL algorithm. This changes the process of programming controllers significantly because engineers need only be able to simulate scenarios of interest and define a performance objective to be optimized (called a reward function). This approach can be advantageous in situations where the problem conditions are easy to simulate, but good strategies are difficult to describe. For example, the below image illustrates simulation scenarios we can programmatically generate to capture a wide range of locomotion situations Spot robots experience on a daily basis.



Sample scenarios experience in simulation during training

To apply RL to Spot's locomotion control stack, we defined the inputs and outputs of the policy in the following way:

Locomotion control system with integrated RL policy

Architecting things this way has multiple benefits. First, it allows us to take advantage of our existing model-based locomotion controller to solve the part of the problem it handles well, while focusing learning on the part of the strategy that is harder to program. Second, it reduces the computational complexity of the production controller by removing the need to run multiple MPC instances in parallel.

We train the policy by running over a million simulations on a compute cluster and using the data from those simulations to update the policy parameters. Simulation environments are generated randomly with varying physical properties (e.g., stair dimensions, terrain roughness, ground friction) and the objective maximized by the RL algorithm includes different terms that reflect the robot's ability to follow navigation commands while not falling or bumping its body into parts of the environment. The result of this process is a policy that works better on average across the distribution of simulation environments it experiences during learning, but how do we know if the policy will lead to an improvement on the real robot fleet?

Evaluating Learned Policies with Robustness Testing

We learned years ago that in order to be confident in robot software changes, it is important to be able to do testing at a representative scale. We achieve this through a combination of simulation and hardware testing to evaluate all software changes before they are shipped to customers. We applied the same process to evaluating learned locomotion policies.

We used many parallel simulations to benchmark the robot performance on a set of scenarios with increasing difficulty. For example, decrease the coefficient of friction to characterize the slipperiest environment that can be reliably traversed. We collected statistics to validate that the learned policies improved performance in specific scenarios while maintaining performance in hundreds of others where the current production system already performs well.

Once simulation results met our standards, we deployed the policy to our internal robustness fleet of Spot robots. These robots operate 24/7 for a cumulative runtime of over 2,000 hours a week indoors, outdoors, on various stairs, floor materials, and simulated weather conditions. We leverage systems that track performance metrics of this fleet in a digital dashboard and upload logs for engineers to review.

To robustify our learned policy given the data we collect, falls and general mobility issues that are reproducible within a physics simulation are recreated in simulation where they become either part of the training or evaluation set. Retraining a policy then effectively robustifies to failure modes we've already seen.

After thousands of hours of simulated and on-robot runtime across varied configurations and terrains is the policy ready for deployment to customer robots.

Results with RL

In both our reliability fleet and in customer fleets, we have seen the benefits of these upgrades. Spot is now less likely to fall—even on extremely slick or irregular surfaces. This improved locomotion means customers can more easily deploy Spot in areas of their facilities that would have otherwise been challenging to access, as well as receive additional downstream benefits from improved reliability. For example, reducing the likelihood of falls also reduces the likelihood of operator interventions or damage resulting from a fall.

Current Spot users will automatically benefit from the performance improvements of this RL policy following our most recent software update. And we're continuing to iterate with the data we're collecting to further refine the policy and expand the capabilities.

Before and after performance using Spot Version 4.0

What's Next?

While a hybrid control strategy that leverages both a model predictive controller and a learned policy is proving effective at expanding and robustifying robotic capabilities, it largely imposes the limitations of the modeling on the full control system. To unlock totally new capabilities or be tolerant in scenarios that excessively violate the assumptions of the models, we are actively exploring and testing totally different architectures.

A learned policy in combination with a model-based controller that is not limited to the periodic gait assumptions of the stepping controllers is capable of climbing boxes >70cm in height

In addition to our own R&D initiatives, we are also excited to enable the broader robotics research community. RL in the real world requires three essential components: robot hardware with joint level controls, a high-performance AI computer, and a simulator where the robot can be trained. At NVIDIA GTC, we announced our new Spot RL Researcher Kit in collaboration with NVIDIA and the AI Institute to provide these components in a single package. The kit includes a license for joint level control API, a payload based on the NVIDIA Jetson AGX Orin for deploying the RL policy, and a GPU accelerated Spot simulation environment based on NVIDIA Isaac Lab.

The new research kit enables developers to create advanced skills for Spot. For example, the AI Institute, our first customer using these tools, developed new gaits for Spot that allow the robot to walk much faster than the standard version. Interested developers can contact us to get started with the Spot RL Researcher Kit.

As our robots continue to expand into new environments and perform more types of work, we expect that machine learning will play an increasing role in our robot behavior development and we will continue to build on the processes, experience, and unique data we generate to make our robots go further and do more.