Reinforcement Learning with MATLAB & Simulink

Christoph Stockhammer

MathWorks Application Engineering

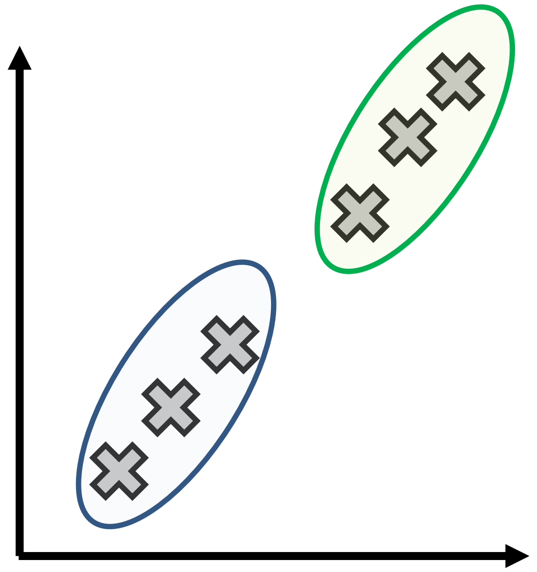
© 2019 The MathWorks, Inc.

Machine Learning, Deep Learning, and Reinforcement Learning

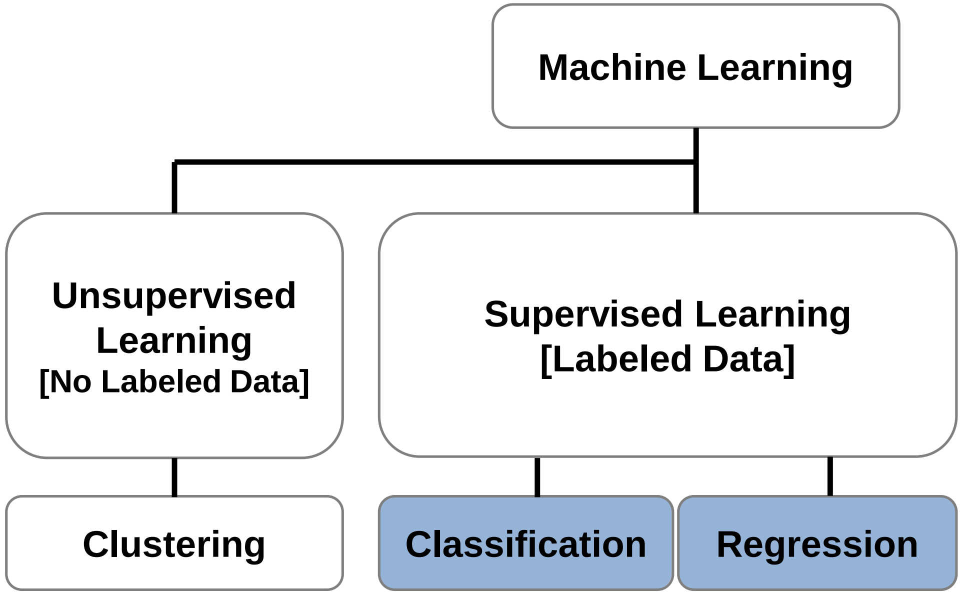
Machine Learning

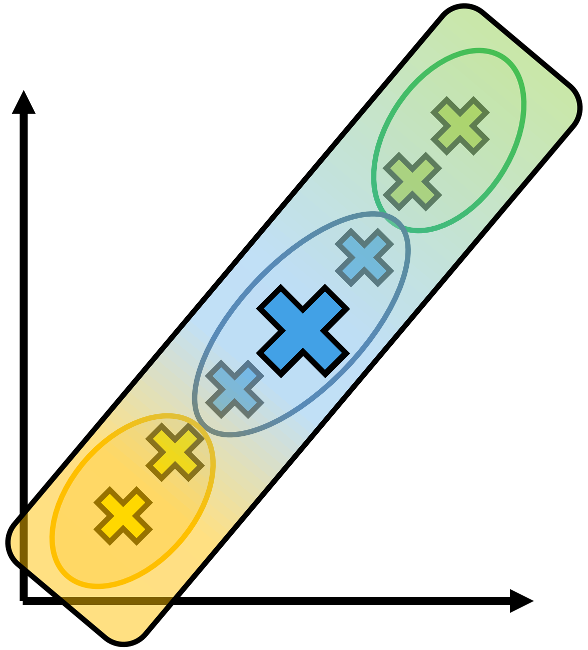
Unsupervised Learning
[No Labeled Data]

Clustering

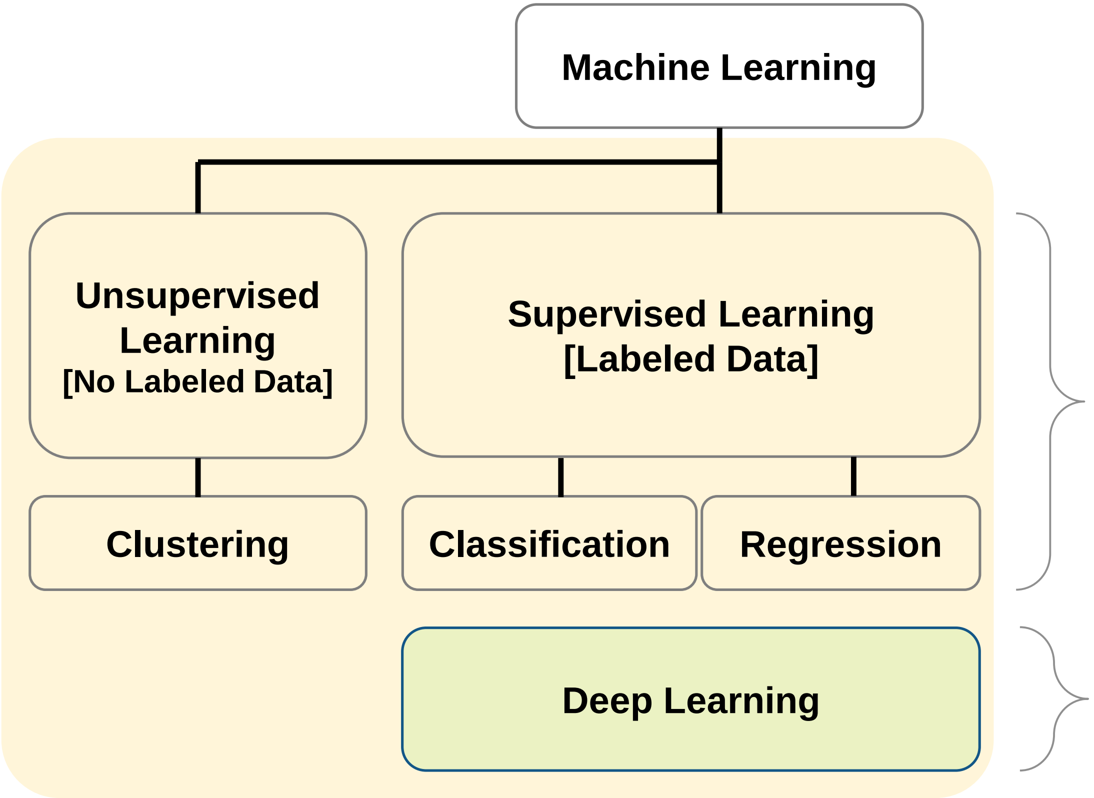


Machine Learning, Deep Learning, and Reinforcement Learning





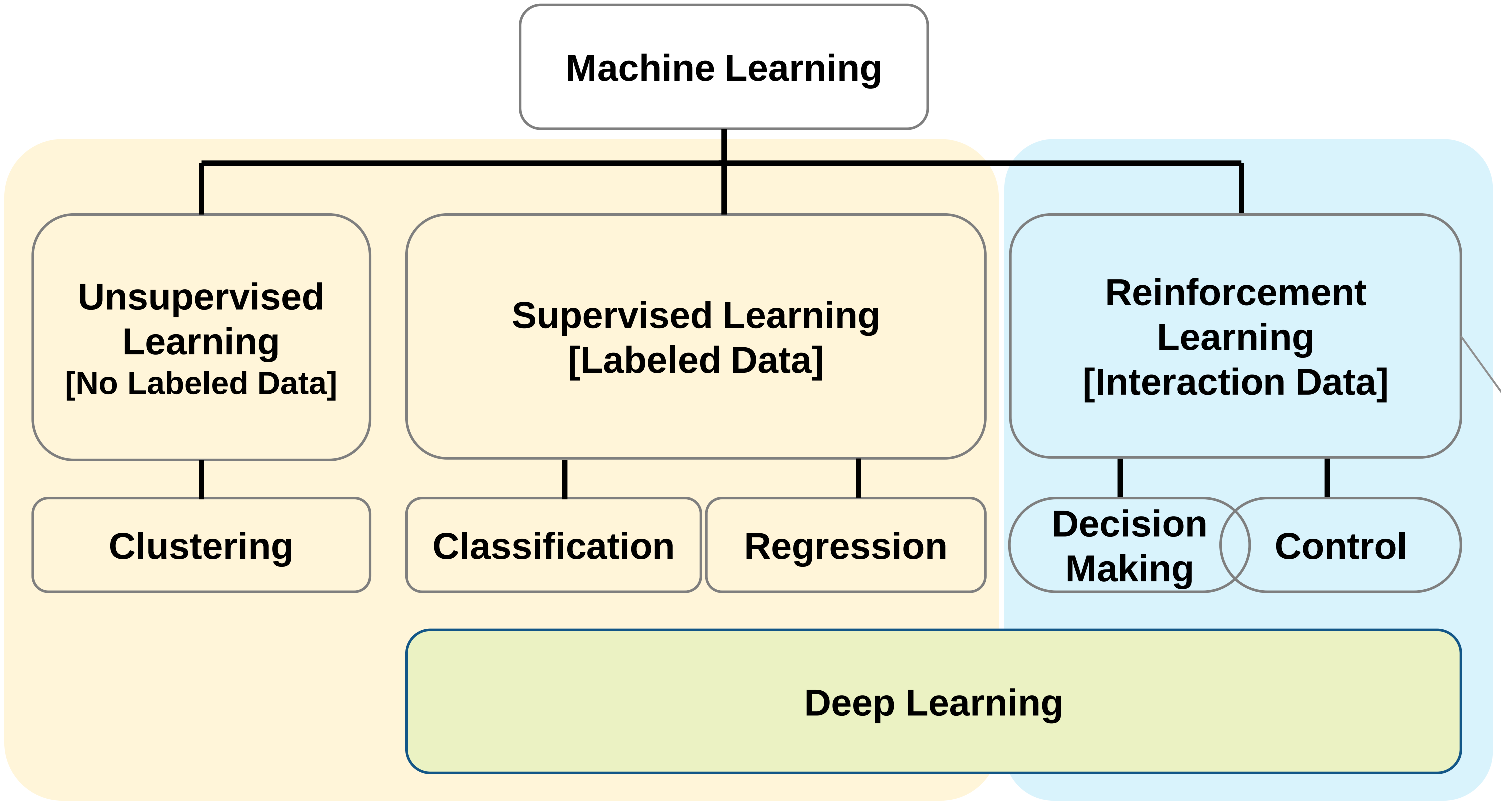
Machine Learning, Deep Learning, and Reinforcement Learning



Supervised learning typically involves manual feature extraction

Deep learning typically does not involve feature extraction

Reinforcement Learning vs Machine Learning

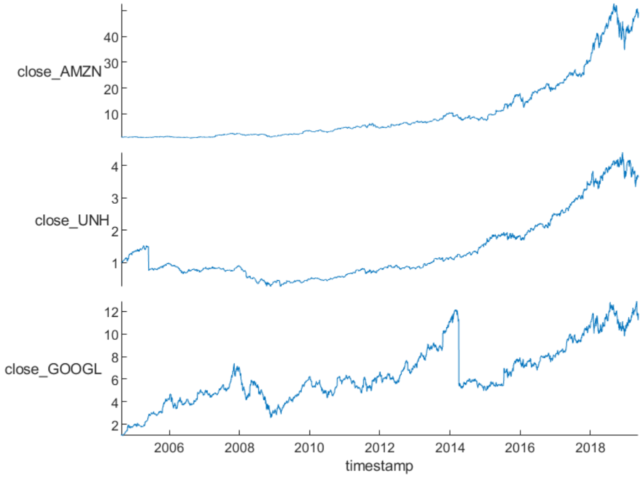


Reinforcement learning:

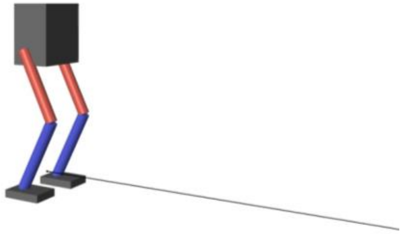
- Learning a **behavior** or accomplishing a **task** through trial & error
  [*interaction*]
- Complex problems typically need deep models
  [*Deep Reinforcement Learning*]

Reinforcement Learning Toolbox
New in R2019a

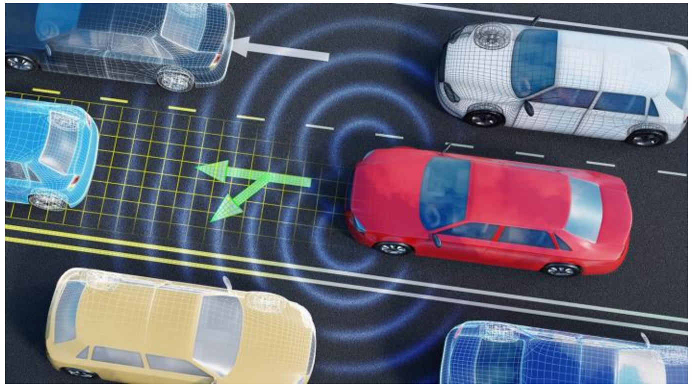
Reinforcement Learning enables the use of Deep Learning for Controls and Decision Making Applications



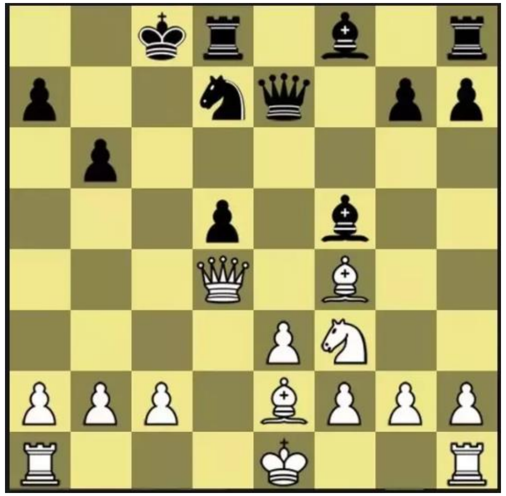
Finance



Robotics

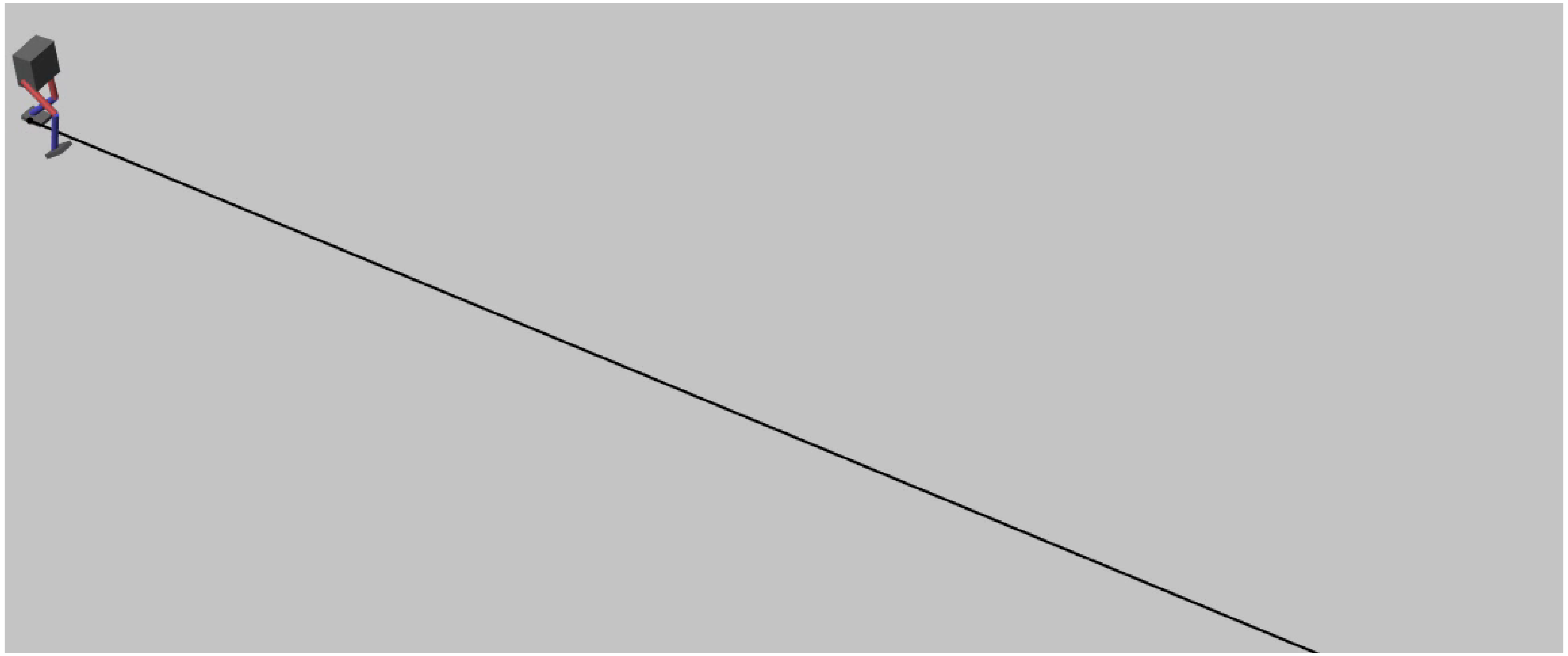


Autonomous driving



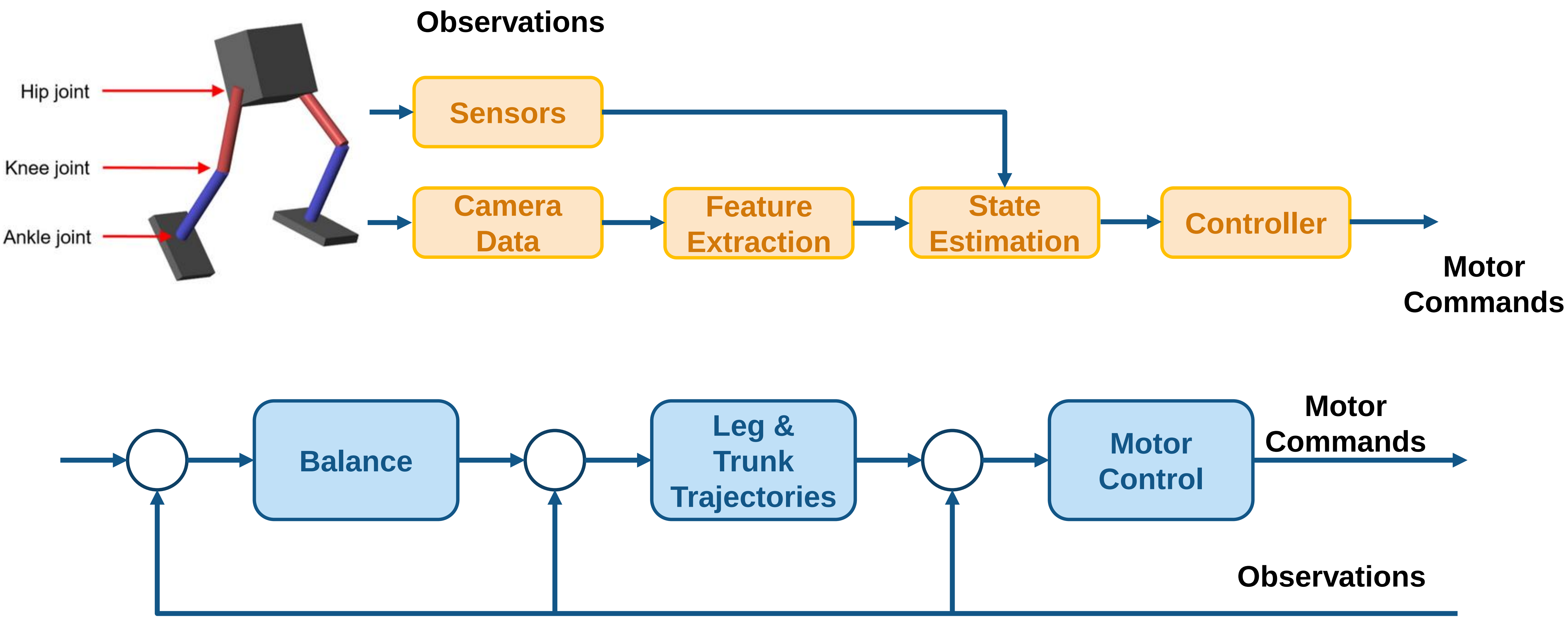
A.I. Gameplay

Why is reinforcement learning appealing?

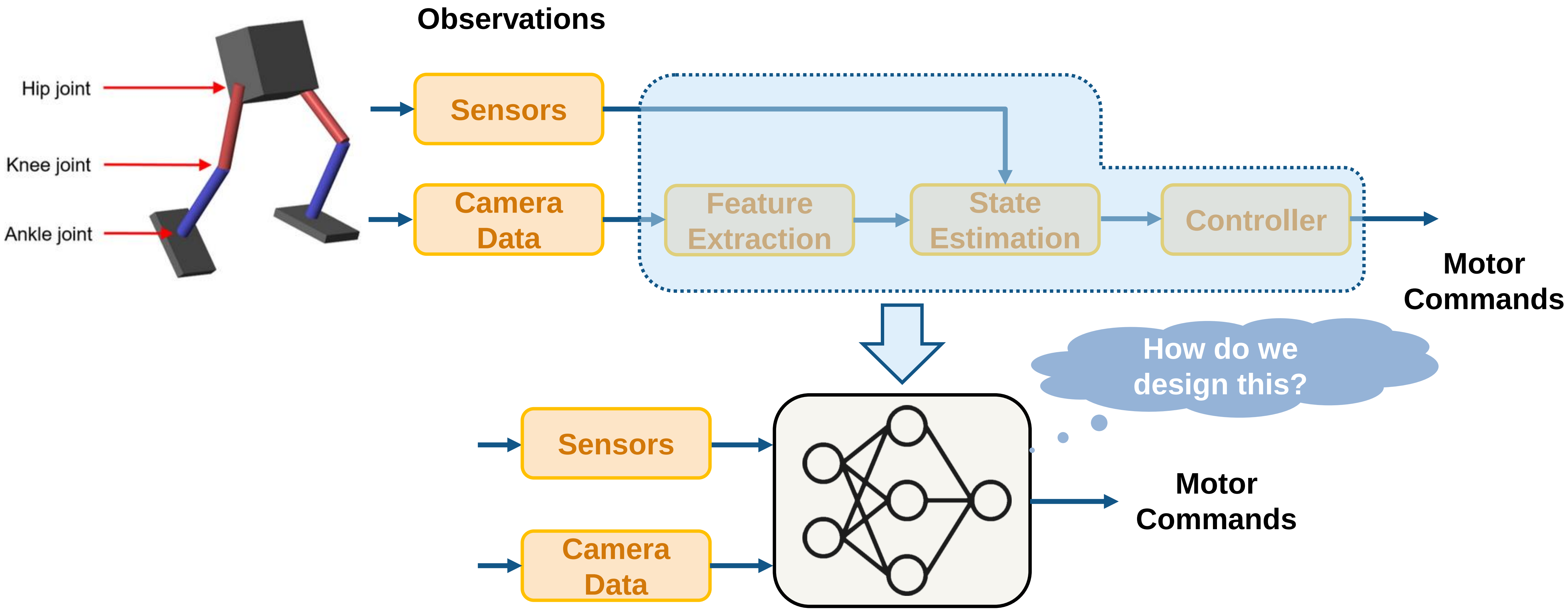


Teach a robot to follow a straight line using camera data

Let's try to solve this problem the traditional way



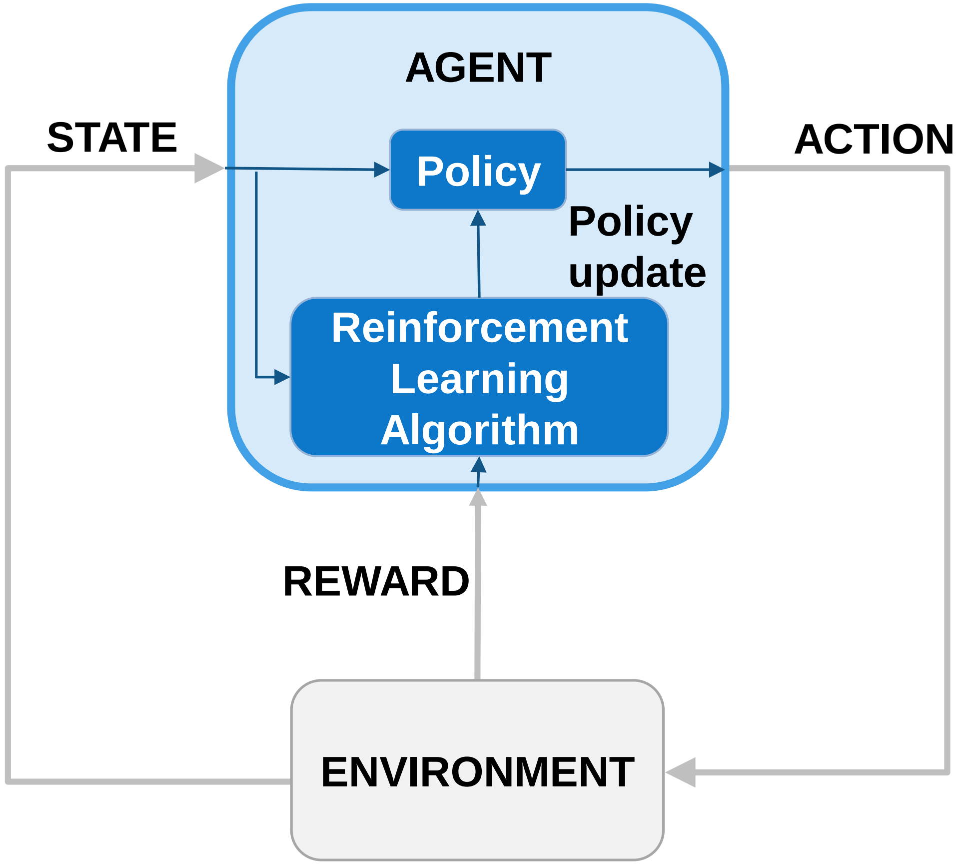
What is the alternative approach?



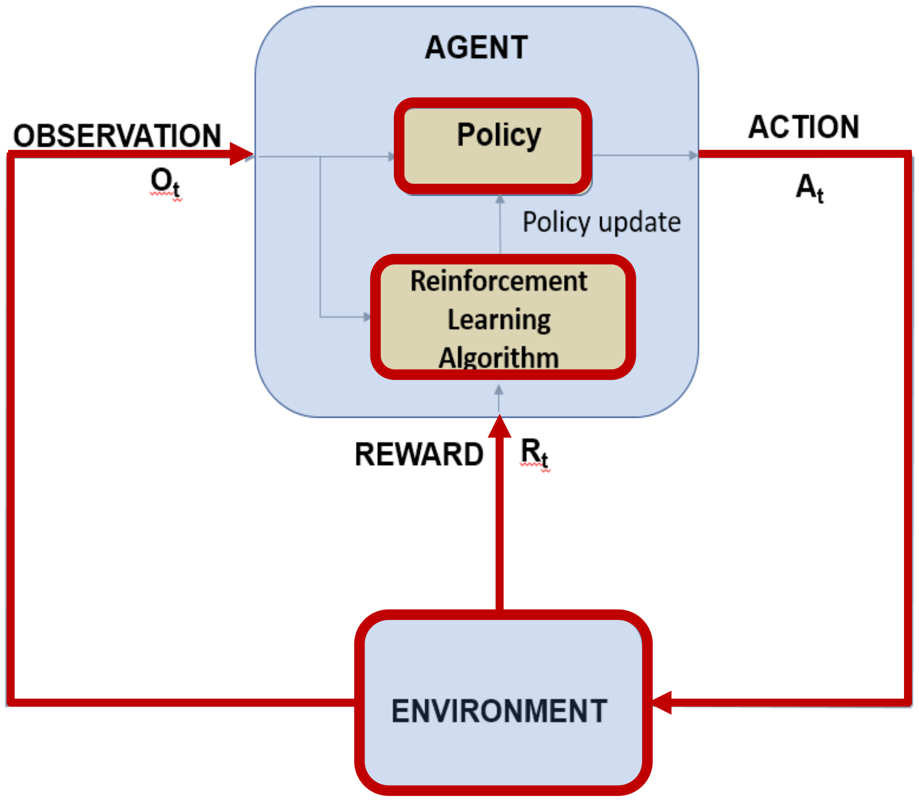
A Practical Example of Reinforcement Learning
Training a Robot to Walk

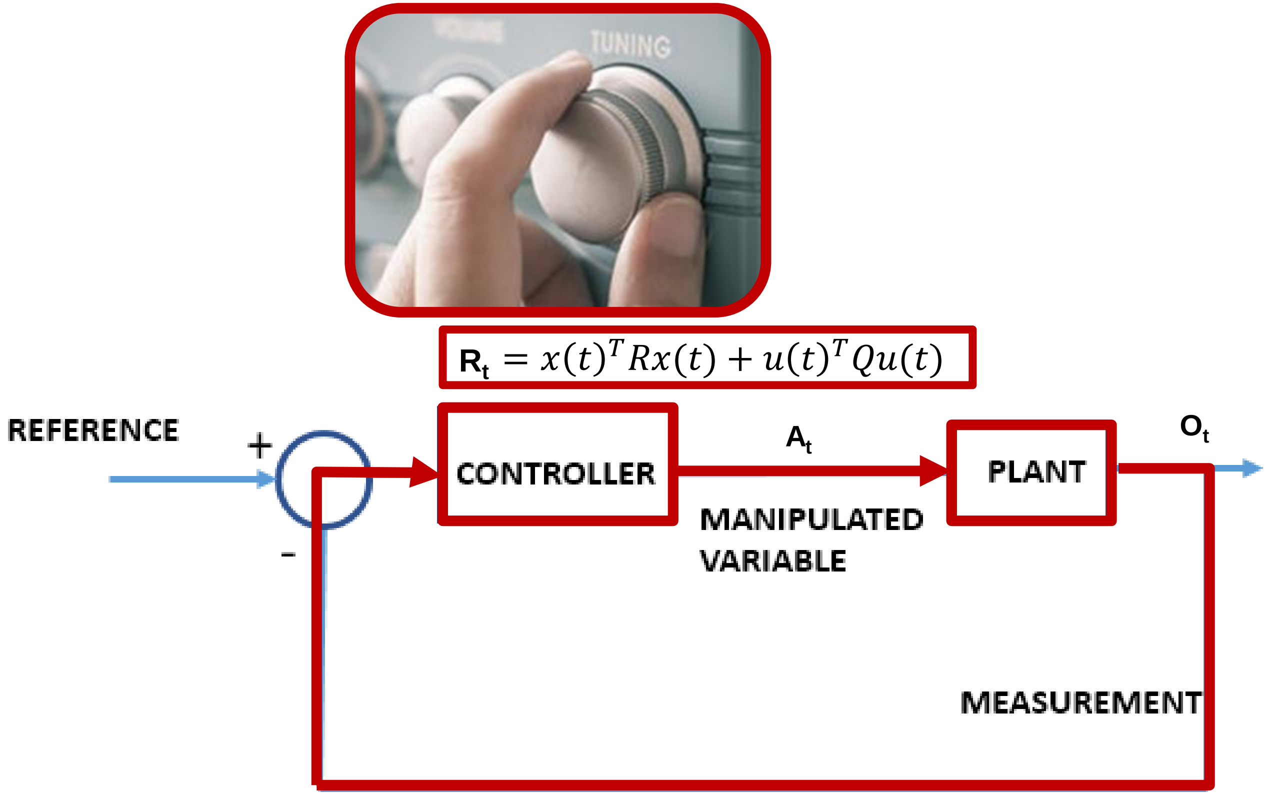
- Robot's computer learns how to walk… (**agent**)
- using sensor readings from joints, torso,… (**state**)
- that represent robot's pose and orientation,… (**environment**)
- by generating joint torque commands,… (**action**)
- based on an internal state-to-action mapping… (**policy**)
- that tries to optimize forward locomotion, … (**reward**).

- The policy is updated through repeated trial-and-error by a **reinforcement learning algorithm**

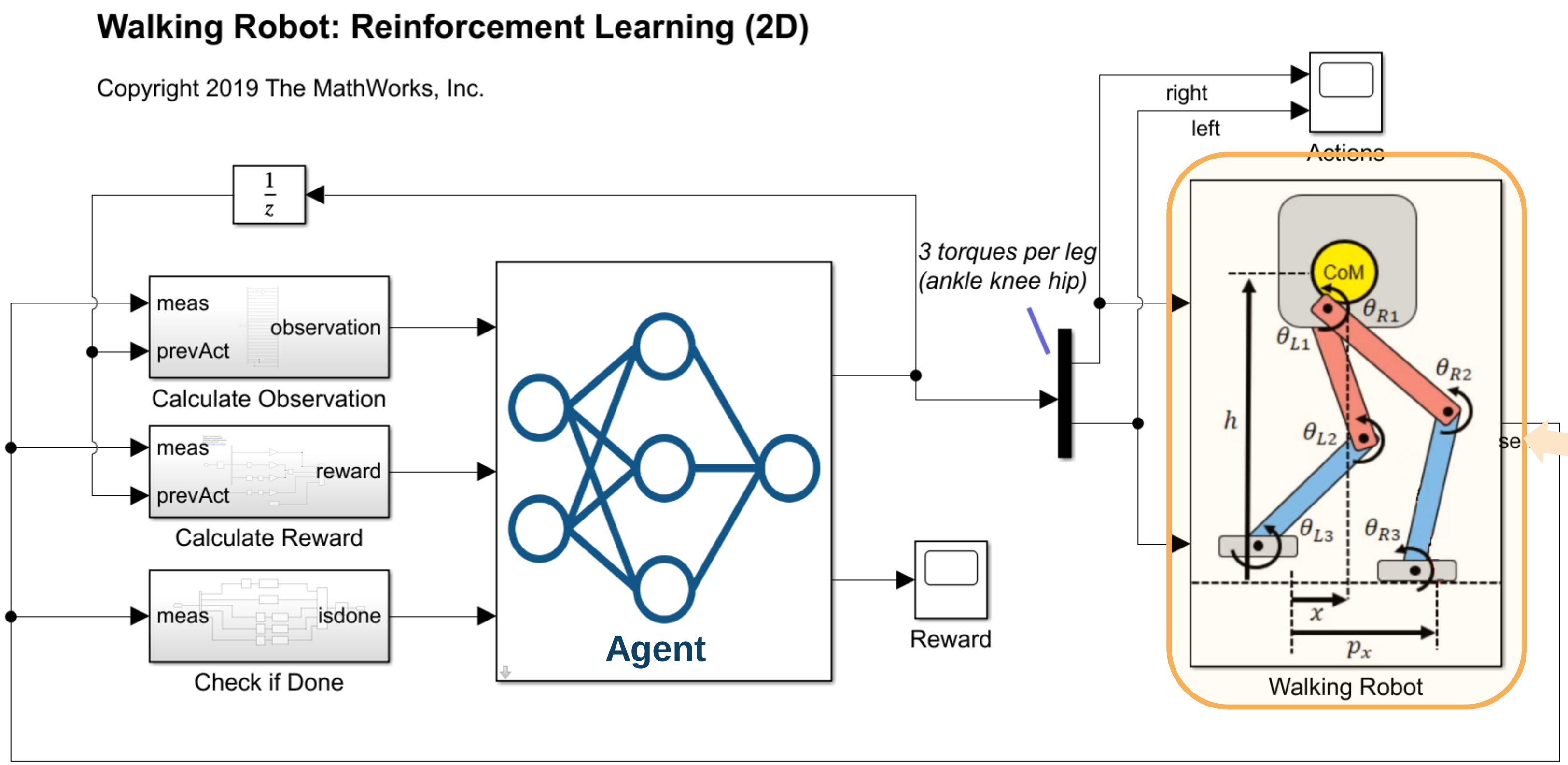


Connections with Controls





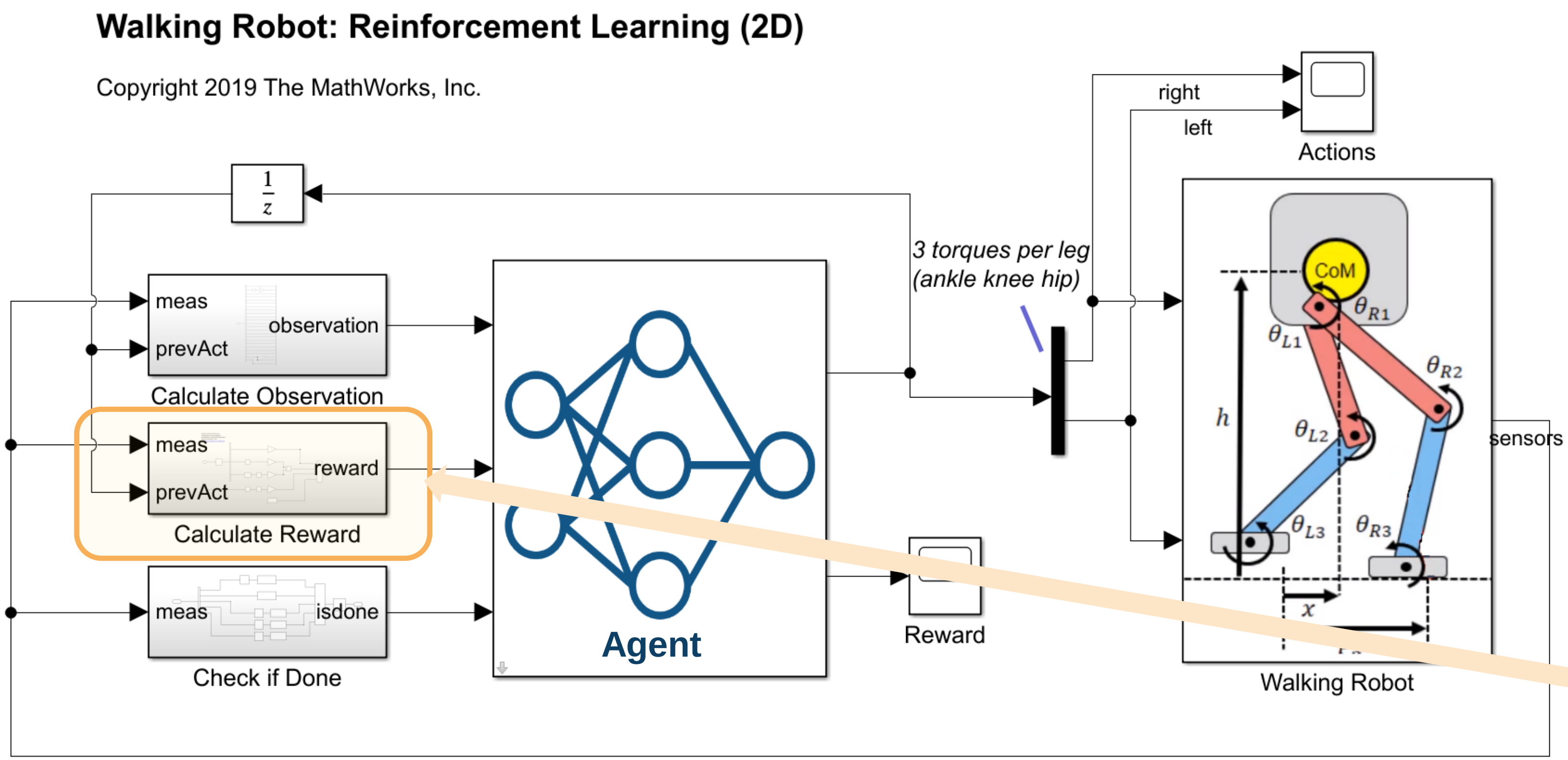
Define Environment to Generate Data



Physical modeling of robot dynamics and contact forces using Simscape

The environment provides 29 observations to the agent.
The observations are: Y (lateral) and Z (vertical) translations of the torso center of mass; X (forward), Y (lateral), and Z (vertical) translation velocities; yaw, pitch, and roll angles of the torso; yaw, pitch, and roll angular velocities; angular position and velocity of 3 joints (ankle, knee, hip) on both legs; and previous actions from the agent. The translation in the Z direction is normalized to a similar range as the other observations.

Define Environment to Generate Data



Reward defines task to learn

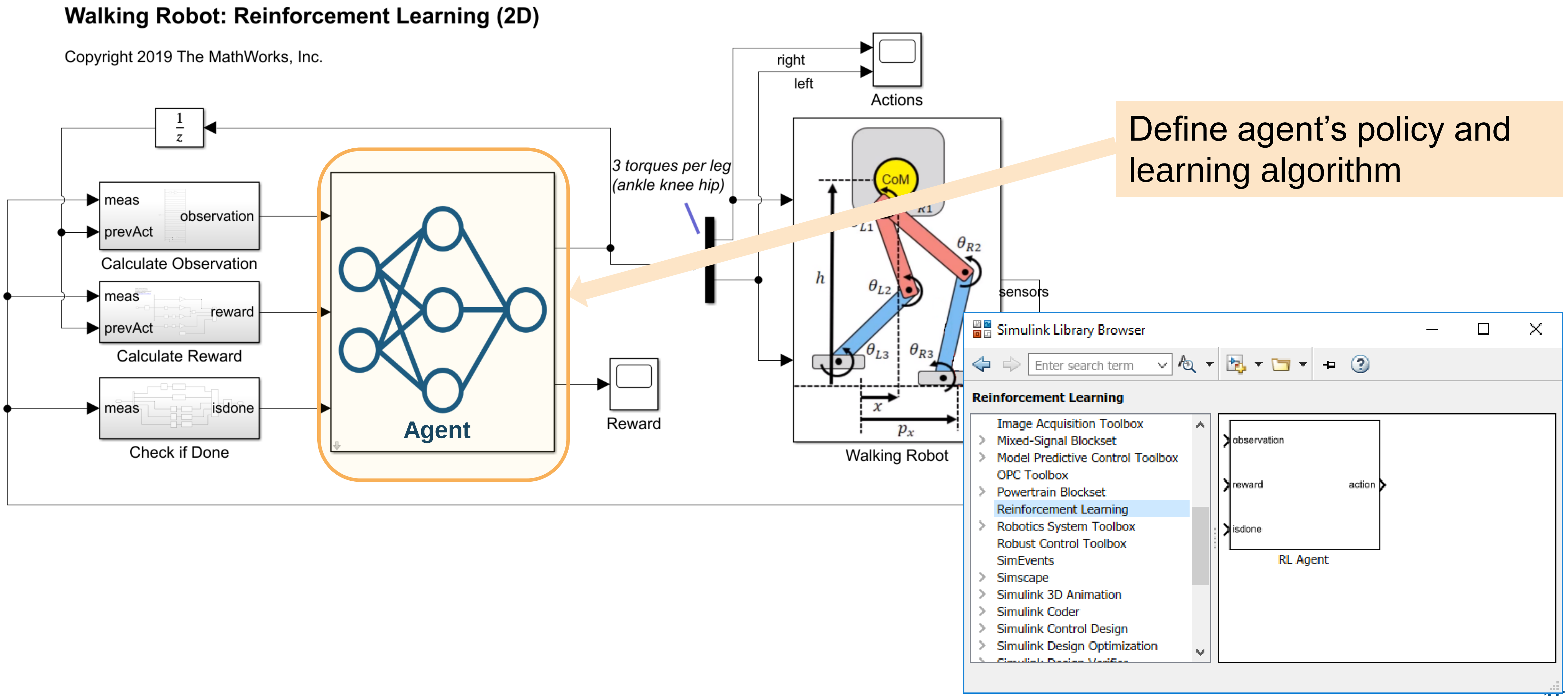
Define Environment to Generate Data

$$r_t = v_x - 3y^2 - 50\hat{z}^2 + 25\frac{\mathrm{Ts}}{\mathrm{Tf}} - 0.02\sum_i {u_{t-1}^i}^2$$

- v_x is the translation velocity in X direction (forward toward goal) of the robot.
- y is the lateral translation displacement of the robot from the target straight line trajectory.
- $\hat{z}$ is the normalized horizontal translation displacement of the robot center of mass.
- u_{t-1}^i is the torque from joint i from the previous time step.
- Ts is the sample time of the environment.
- Tf is the final simulation time of the environment.

*Reward function inspired by: N. Heess et al, "Emergence of Locomotion Behaviours in Rich Environments," Technical Report, *ArXiv*, 2017.

Define Policy and Learning Algorithm



Define agent’s policy and learning algorithm

Code for Configuring Agent and Training

Create Environment Interface

Create the observation specification.

```
numObs = 29;
obsInfo = rlNumericSpec([numObs 1]);
obsInfo.Name = 'observations';
```

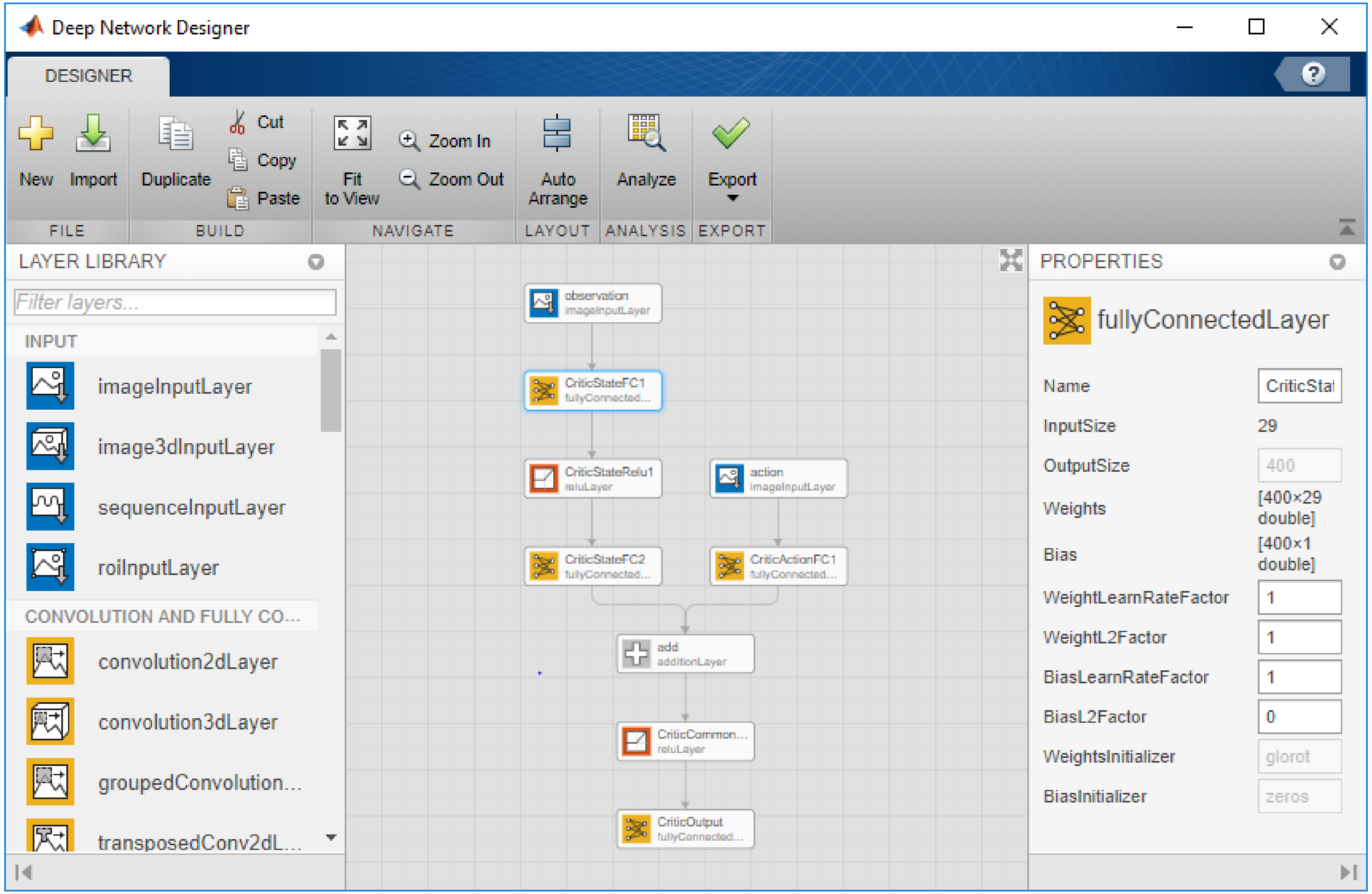
Create the action specification.

```
numAct = 6;
actInfo = rlNumericSpec([numAct 1],'LowerLimit',-1,'UpperLimit', 1);
actInfo.Name = 'foot_torque';
```

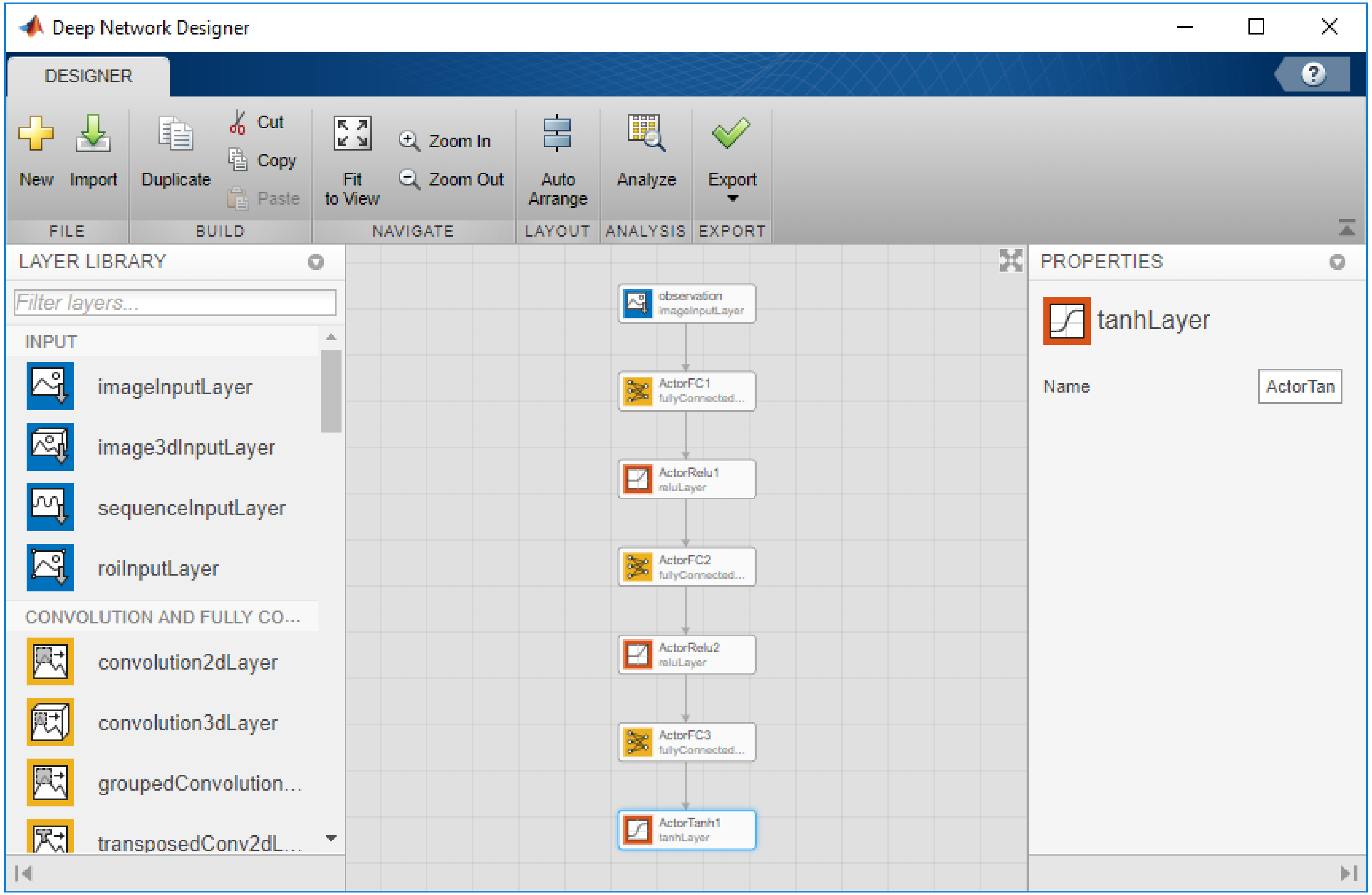
Create the environment interface for the walking robot model.

```
blk = [mdl,'/RL Agent'];
env = rlSimulinkEnv(mdl,blk,obsInfo,actInfo);
env.ResetFcn = @(in) walkerResetFcn(in,upper_leg_length/100,lower_leg_length/100,h/100);
```

Create Critic Network



Create Actor Network



Create DDPG Agent

Specify options for the critic and actor representations using `rlRepresentationOptions`.

```
criticOptions = rlRepresentationOptions('Optimizer','adam','LearnRate',1e-3, ...
                                        'GradientThreshold',1,'L2RegularizationFactor',1e-5);
actorOptions = rlRepresentationOptions('Optimizer','adam','LearnRate',1e-4, ...
                                       'GradientThreshold',1,'L2RegularizationFactor',1e-5);
```

Create the critic and actor representations using the specified deep neural networks and options. You must also specify the action and observation information for each representation, which you already obtained from the environment interface. For more information, see `rlRepresentation`.

```
critic = rlRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'observation'},'Action',{'action'},criticOptions);
actor  = rlRepresentation(actorNetwork,obsInfo,actInfo,'Observation',{'observation'},'Action',{'ActorTanh1'},actorOptions);
```

To create the DDPG agent, first specify the DDPG agent options using `rlDDPGAgentOptions`.

```
agentOptions = rlDDPGAgentOptions;
agentOptions.SampleTime = Ts;
agentOptions.DiscountFactor = 0.99;
agentOptions.MiniBatchSize = 128;
agentOptions.ExperienceBufferLength = 1e6;
agentOptions.TargetSmoothFactor = 1e-3;
agentOptions.NoiseOptions.MeanAttractionConstant = 1;
agentOptions.NoiseOptions.Variance = 0.1;
```

Then, create the DDPG agent using the specified actor representation, critic representation, and agent options. For more information, see `rlDDPGAgent`.

```
agent = rlDDPGAgent(actor,critic,agentOptions);
```

Training the Agent

```
maxEpisodes = 20000;
maxSteps = Tf/Ts;
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxEpisodes,...
    'MaxStepsPerEpisode',maxSteps,...
    'ScoreAveragingWindowLength',250,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward',...
    'StopTrainingValue',100,...
    'SaveAgentCriteria','EpisodeReward',...
    'SaveAgentValue',150);
```

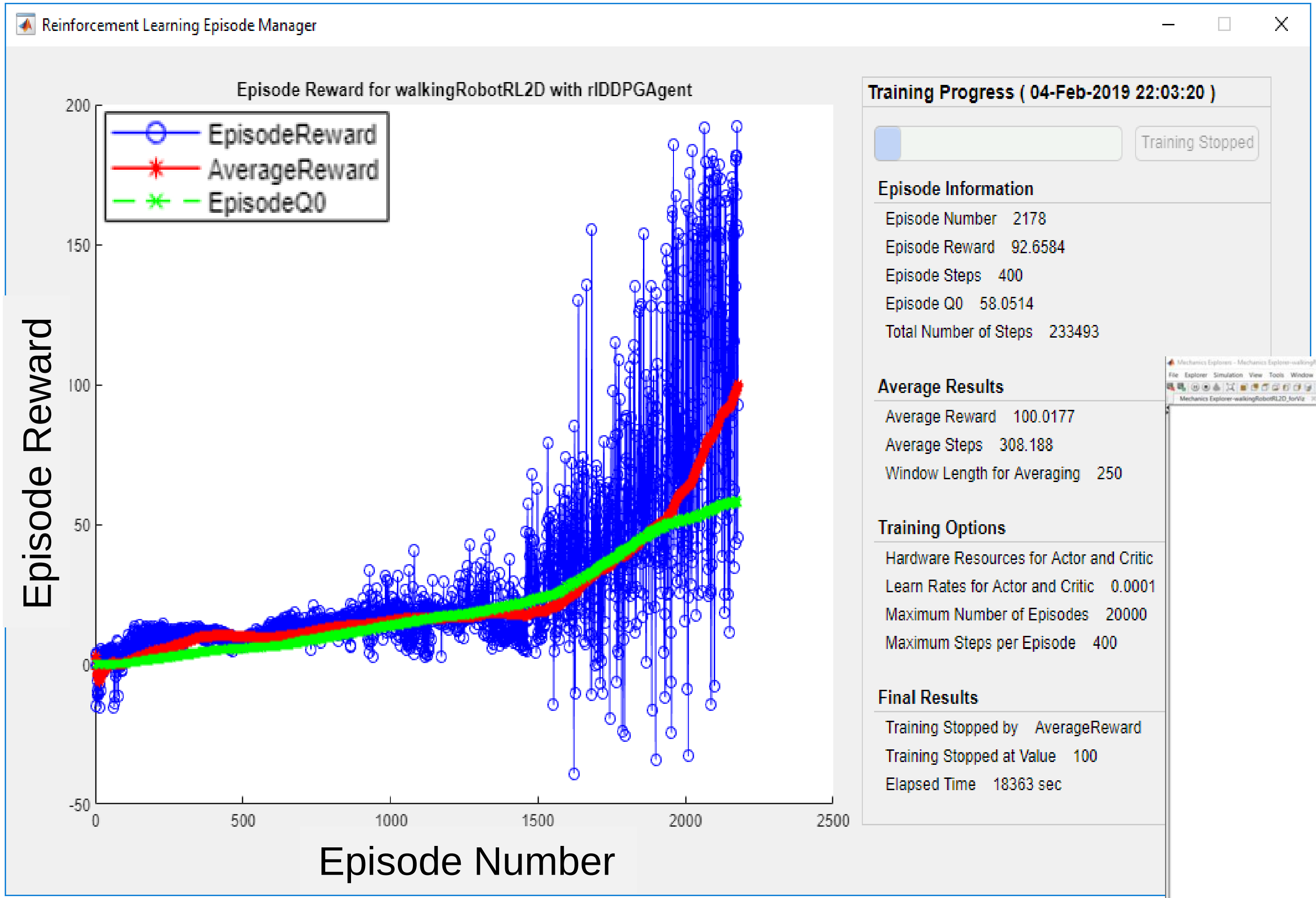
To train the agent in parallel, specify the following training options.

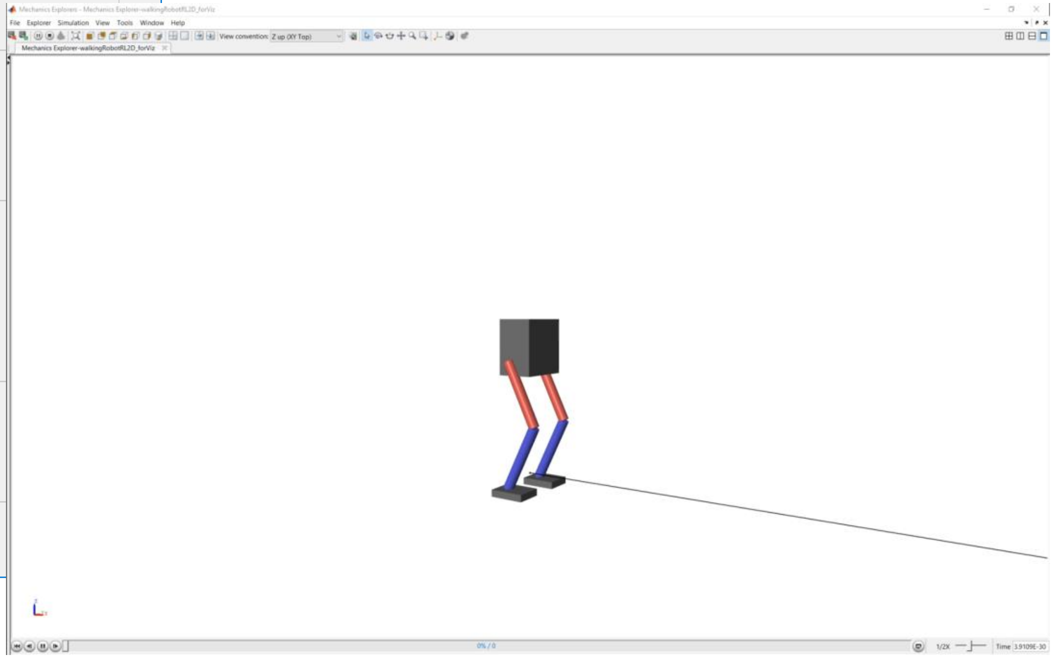
- Set the `UseParallel` option to `true`.
- Train the agent in parallel asynchronously.
- After every 32 steps, each worker sends experiences to the host.
- DDPG agents require workers to send `'Experiences'` to the host.

```
trainOpts.UseParallel = true;
trainOpts.ParallelizationOptions.Mode = 'async';
trainOpts.ParallelizationOptions.StepsUntilDataIsSent = 32;
trainOpts.ParallelizationOptions.DataToSendFromWorkers = 'Experiences';
```

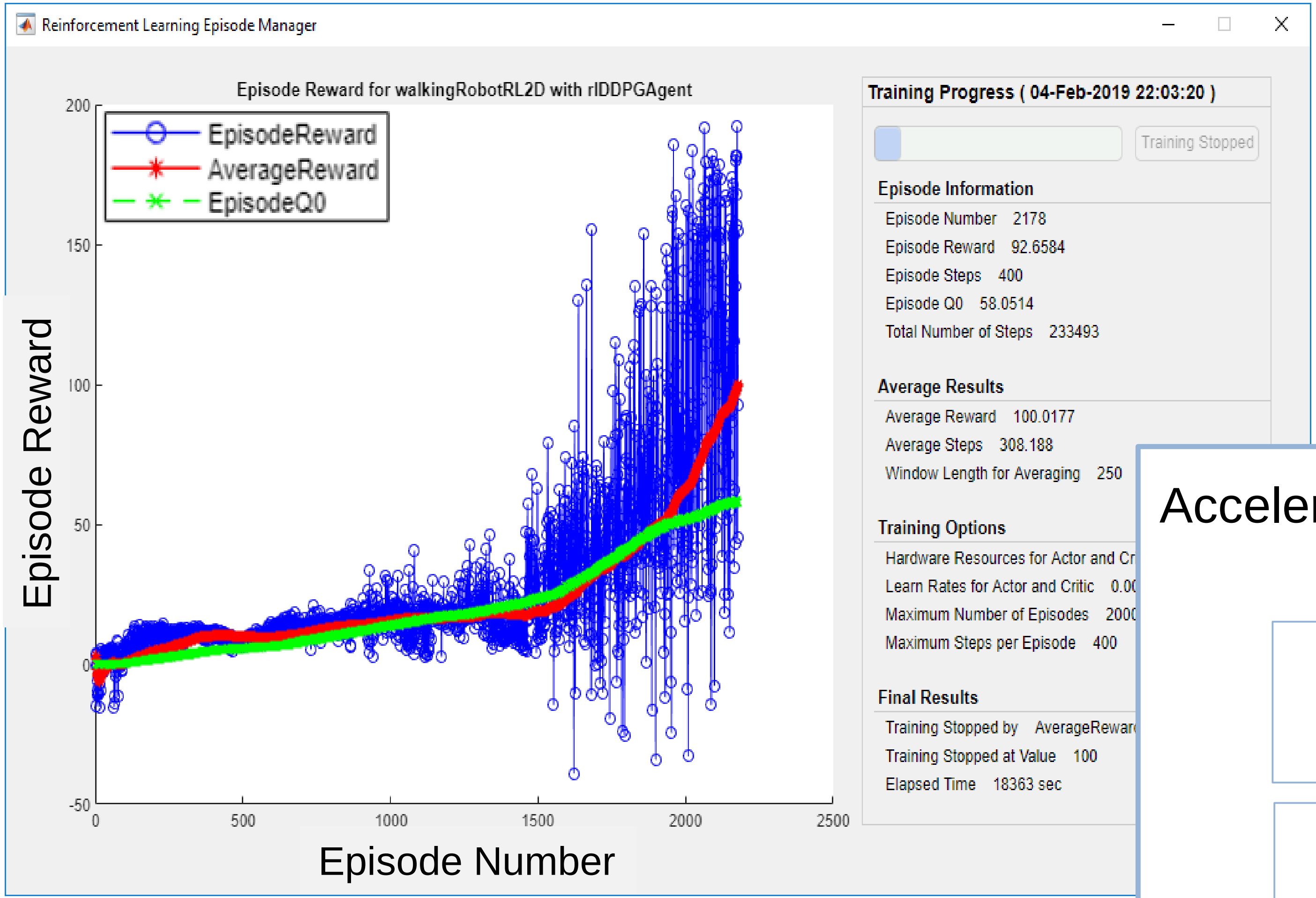
```
trainingStats = train(agent,env,trainOpts);
```

Train Robot to Walk and Track Progress

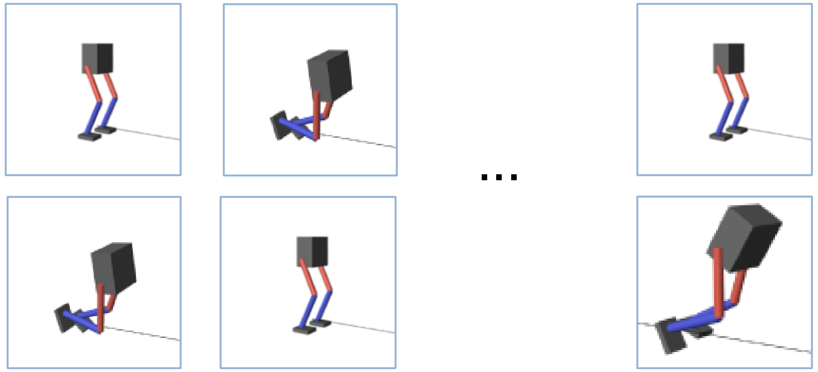




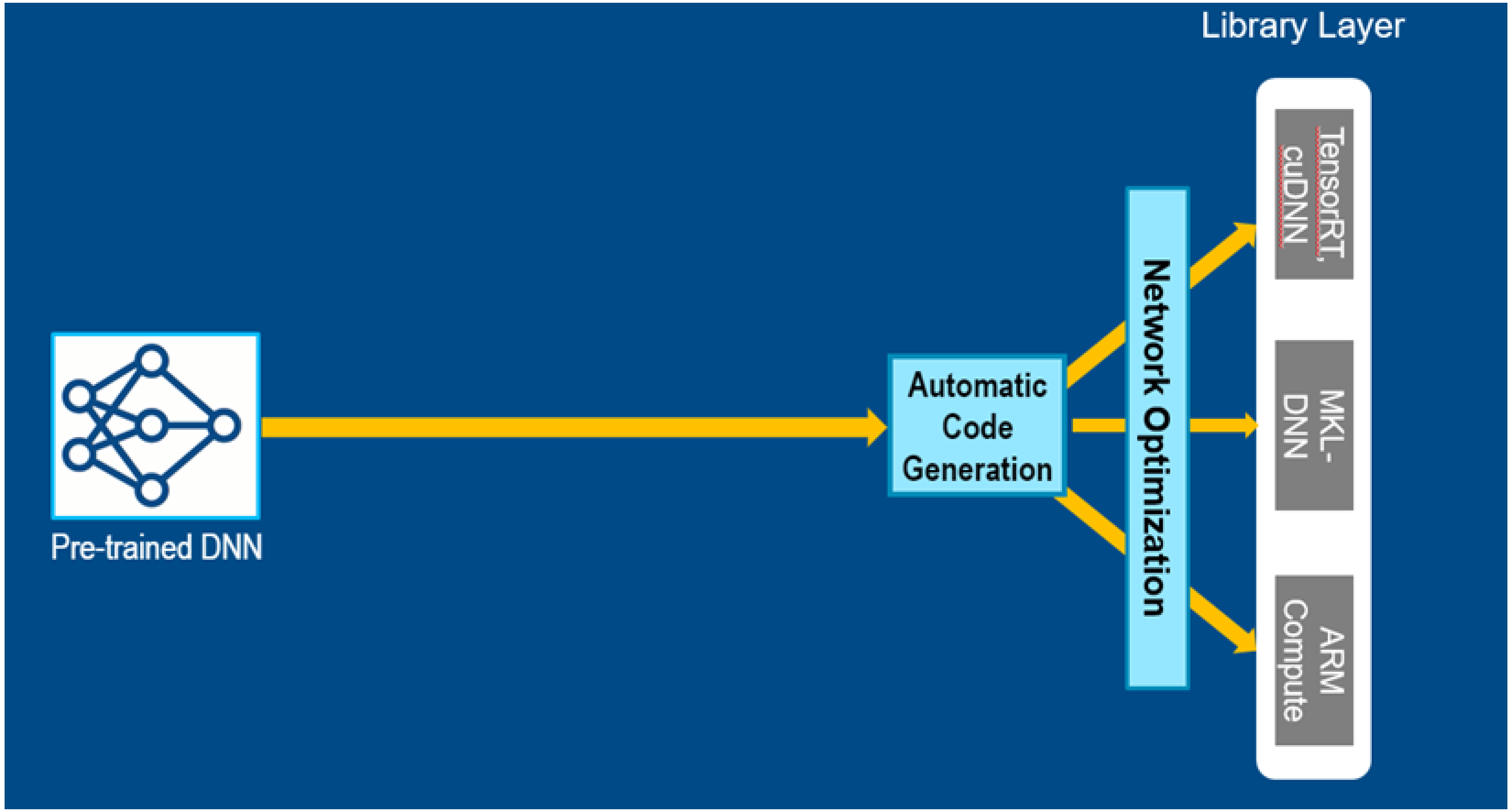
Train Robot to Walk and Track Progress



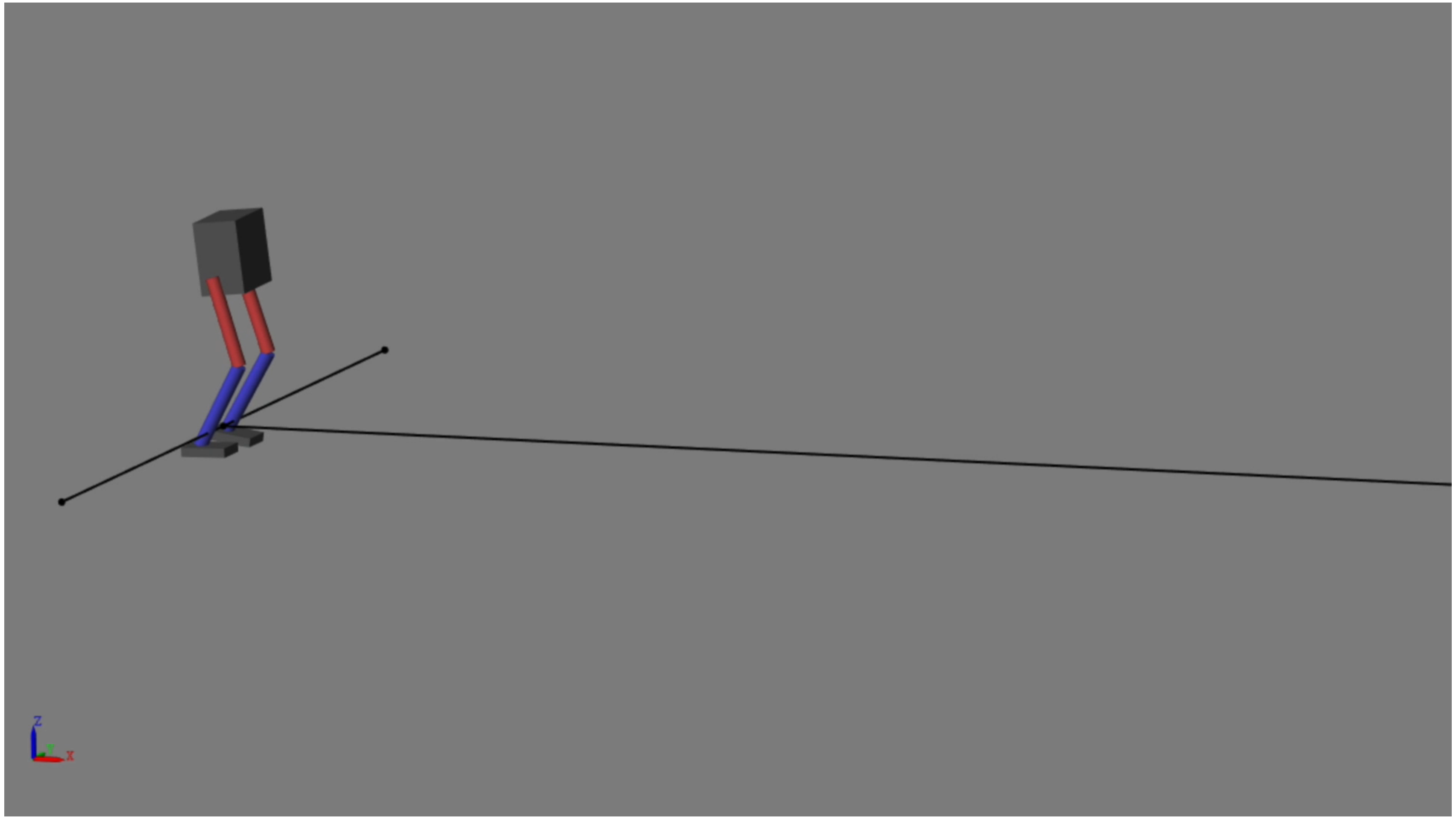
Accelerate Training with Parallel Computing



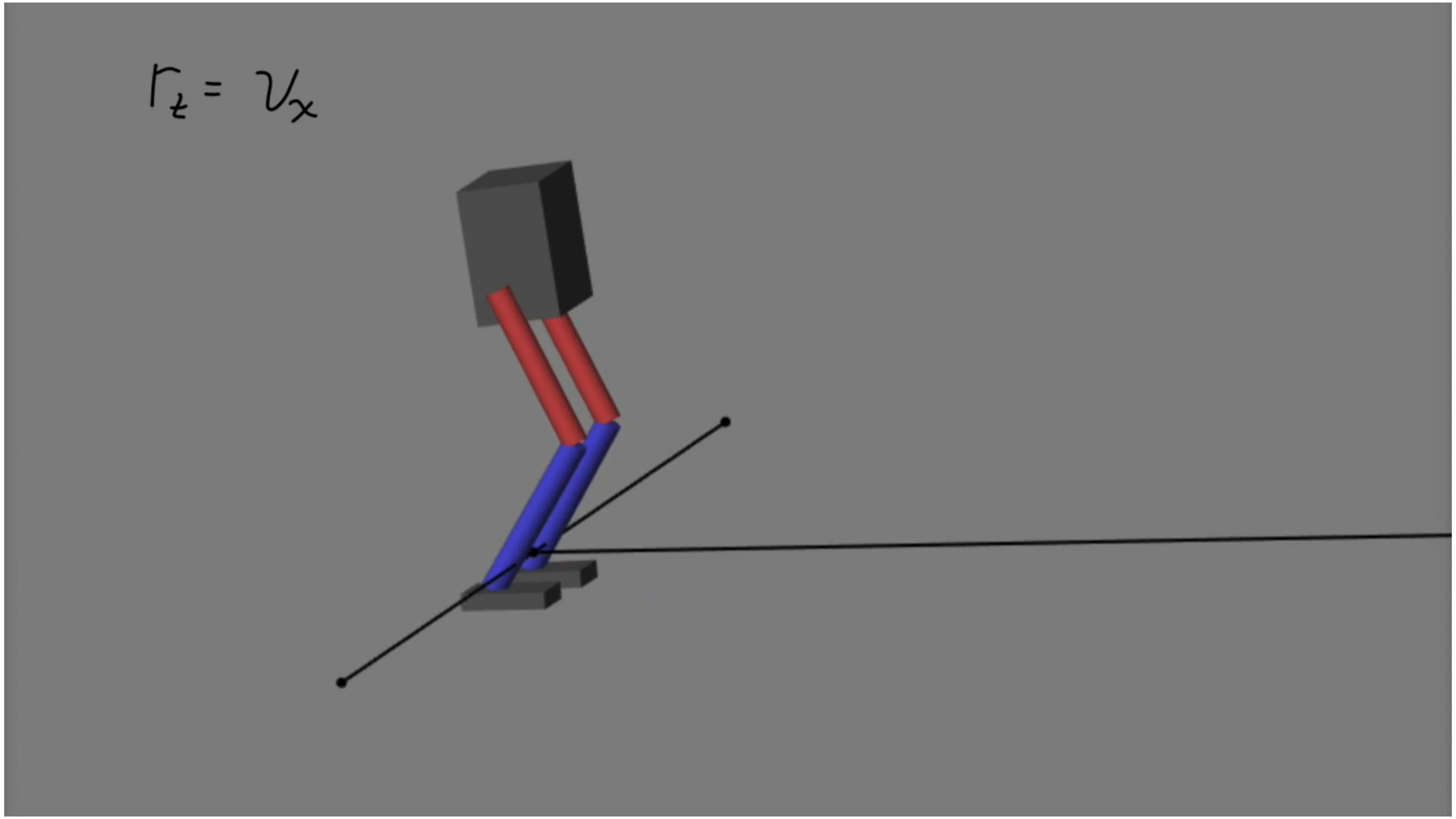
Deploy Policy to Embedded Device



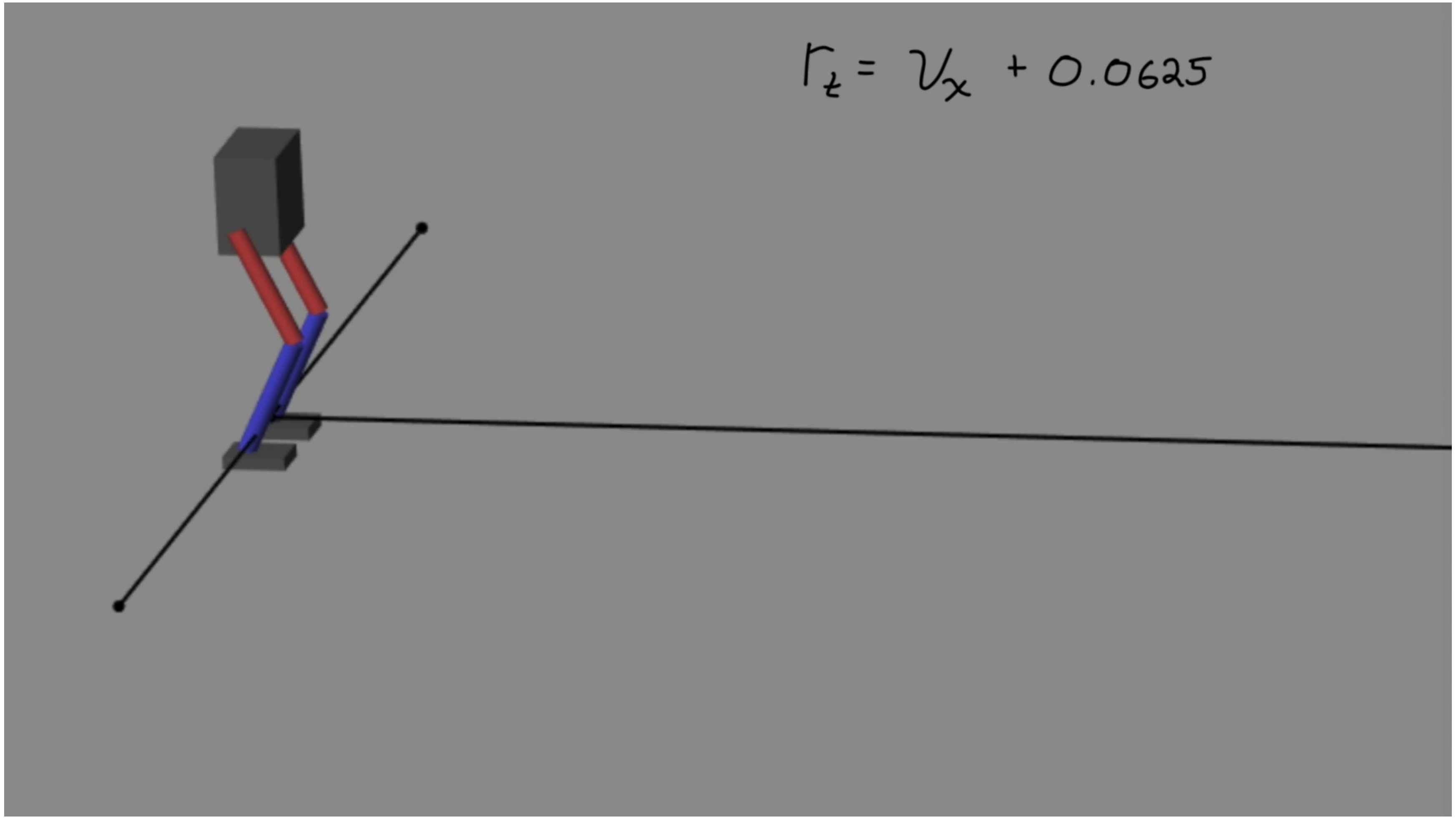
Everything is Great, Right?



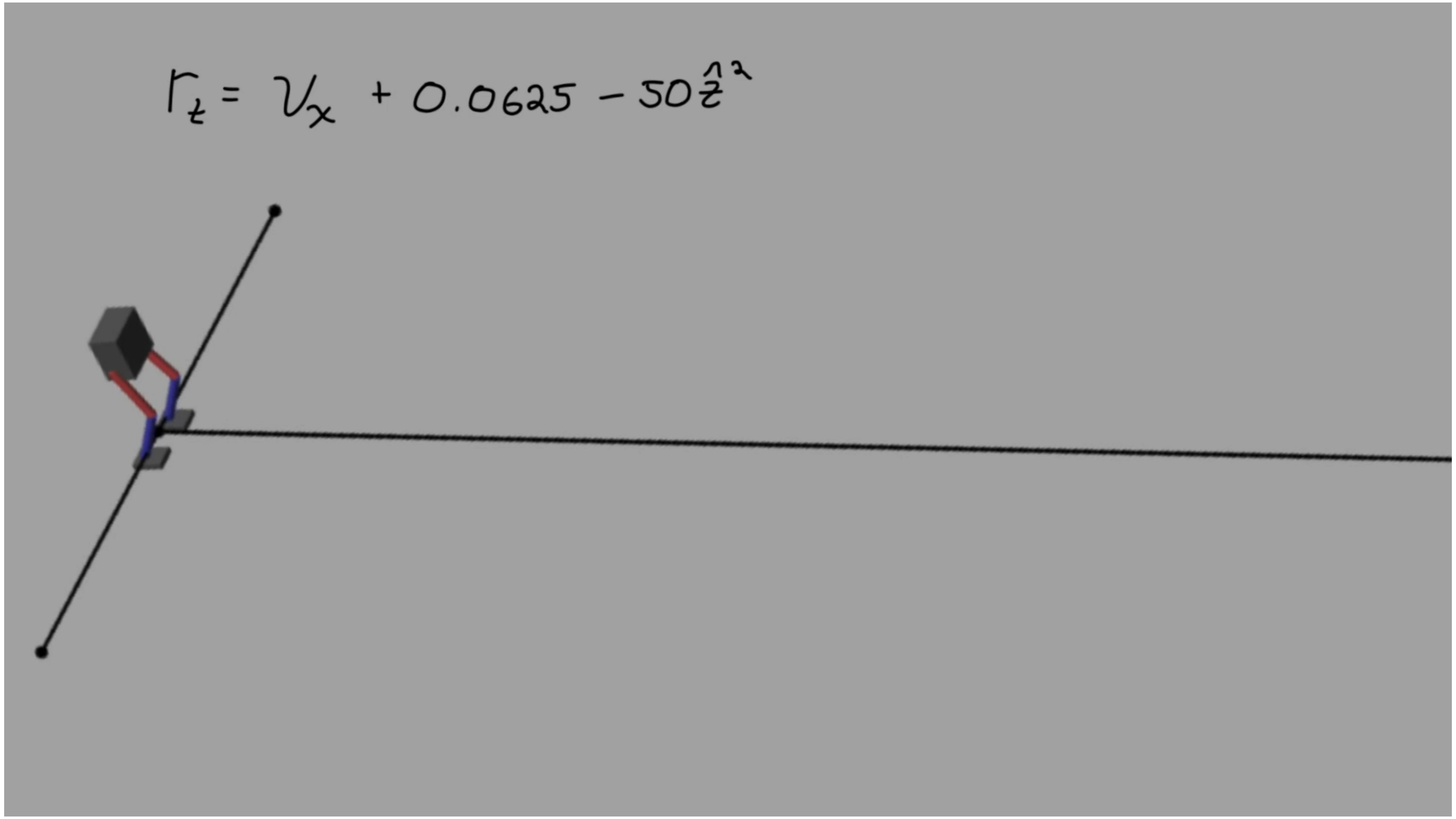
Reward Function Design Matters



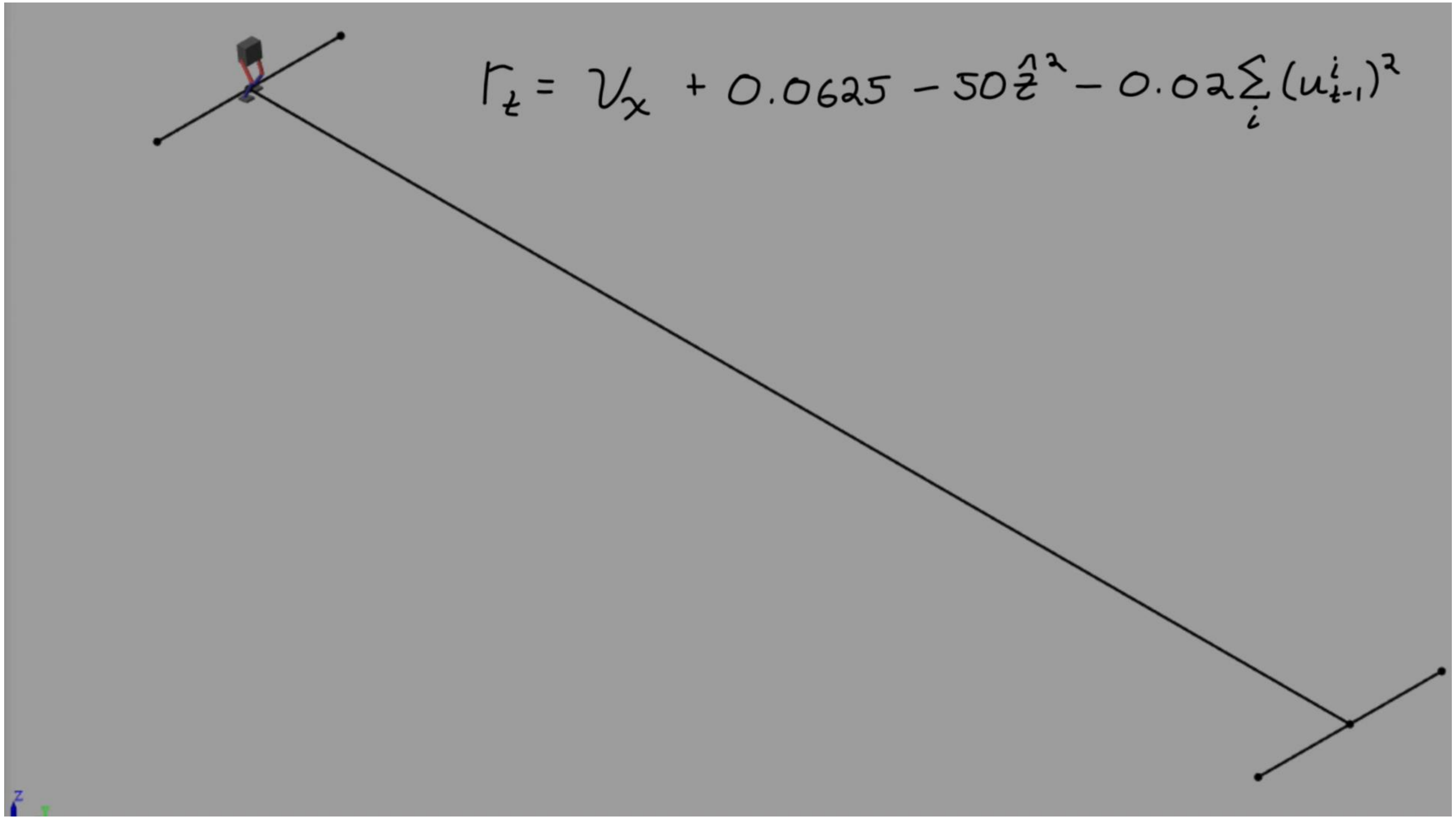
Reward Function Design Matters



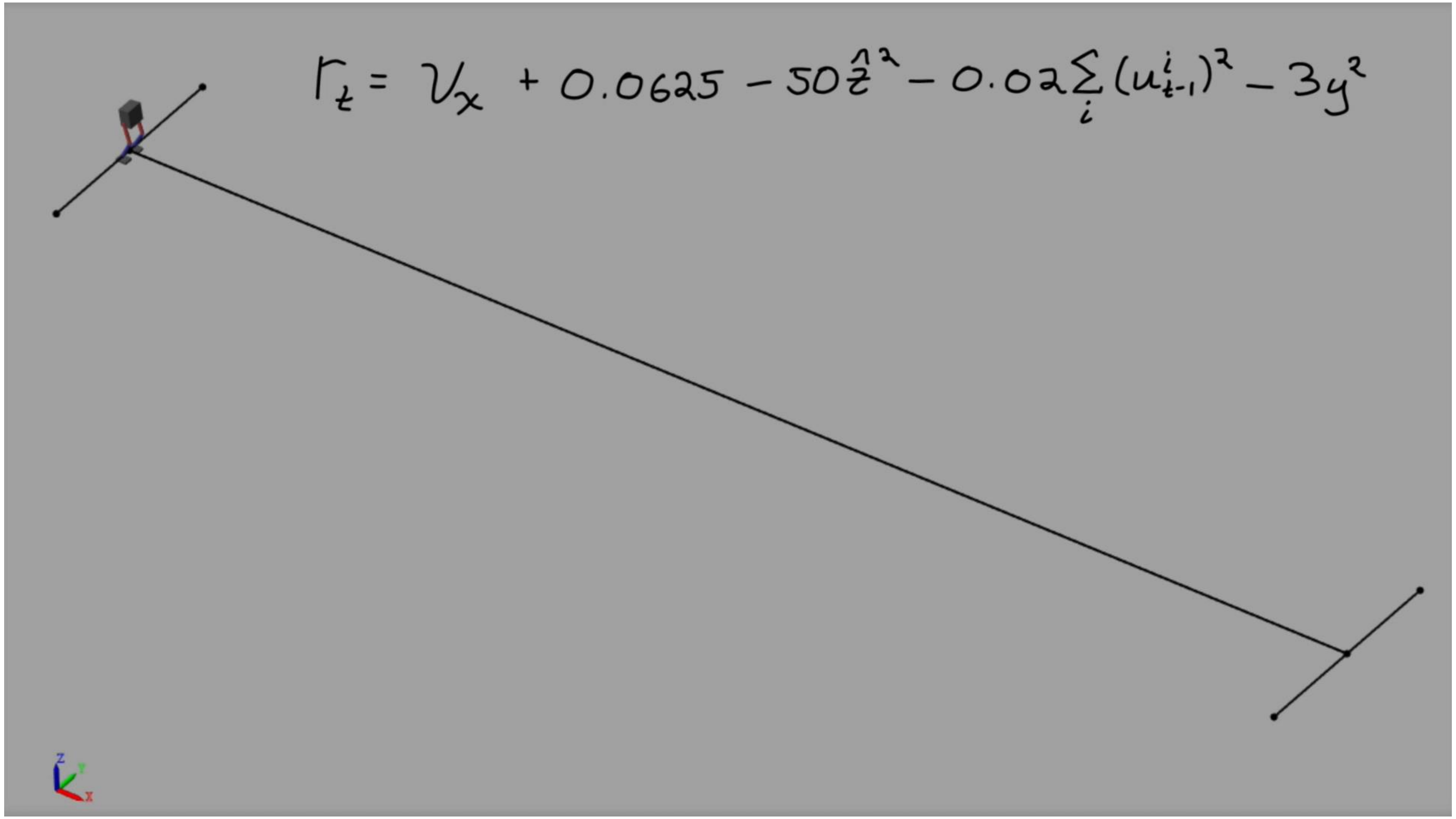
Reward Function Design Matters



Reward Function Design Matters



Reward Function Design Matters

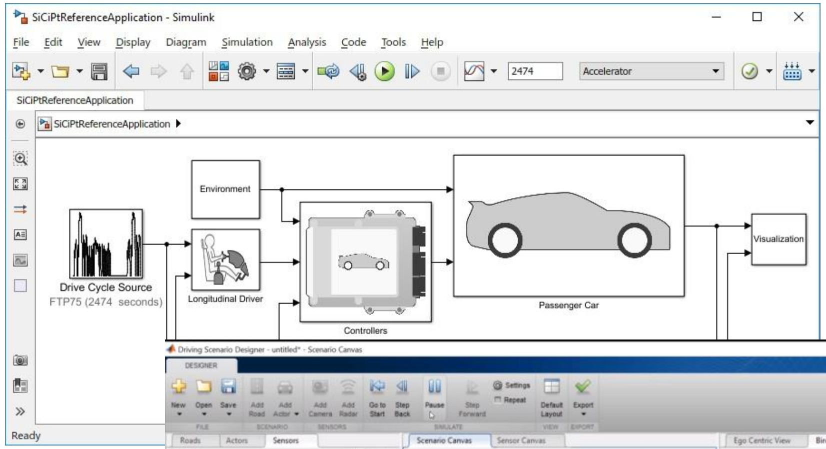


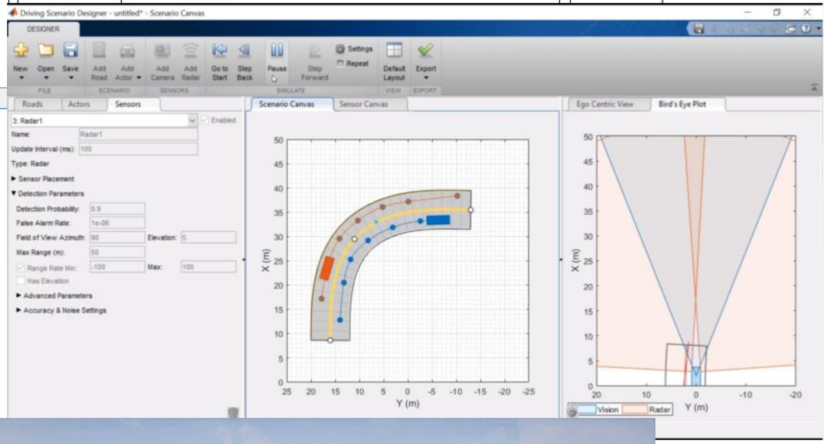
Simulation and Virtual Models are a Key Aspect of Reinforcement Learning

- Reinforcement learning needs **a lot** of data (*sample inefficient*)
  - Training on hardware can be prohibitively expensive and dangerous

- Virtual models allow you to simulate conditions hard to emulate in the real world
  - This can help develop a more robust solution

- Many of you have already developed MATLAB and Simulink models that can be reused



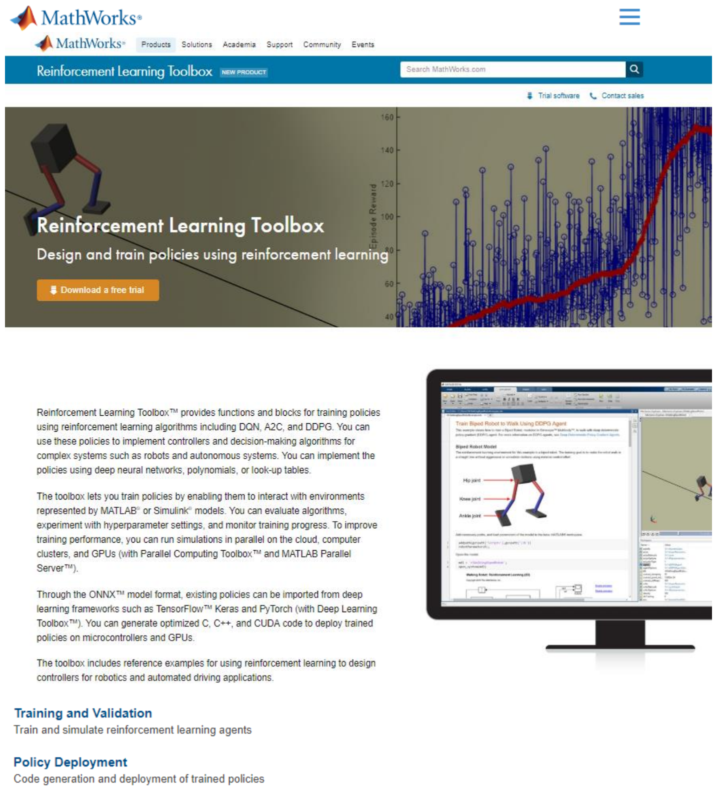


Pros & Cons of Reinforcement Learning

Pros	Cons
No need to label data before training	A lot of simulation trials required
Complex end-to-end solutions can be developed (e.g. camera input→ car steering wheel)	Reward signal design, network layer structure & hyperparameter tuning can be challenging
Can be applied to uncertain, nonlinear environments	No performance guarantees, Training may not converge
Virtual models allow simulations of varying conditions and training parallelization	Further training might be necessary after deployment on real hardware

Reinforcement Learning Toolbox
New in R2019a

- Built-in and custom algorithms for reinforcement learning
- Environment modeling in MATLAB and Simulink
- Deep Learning Toolbox support for designing policies
- Training acceleration through GPUs and cloud resources
- Deployment to embedded devices and production systems
- Reference examples for getting started



Predefined Environments and Many Examples

- MATLAB Environment
  - 'BasicGridWorld'
  - 'CartPole-Discrete'
  - 'CartPole-Continuous'
  - 'DoubleIntegrator-Discrete'
  - 'DoubleIntegrator-Continuous'
  - 'SimplePendulumWithImage-Discrete'
  - 'SimplePendulumWithImage-Continuous'
  - 'WaterFallGridWorld-Stochastic'
  - 'WaterFallGridWorld-Deterministic'

- Simulink Environment
  - 'SimplePendulumModel-Discrete'
  - 'SimplePendulumModel-Continuous'
  - 'CartPoleSimscapeModel-Discrete'
  - 'CartPoleSimscapeModel-Continuous'

- Examples
  - Grid World, MDP
  - Classical Control Benchmarks
  - Automotive
  - Robotics
  - Custom LQR Agent

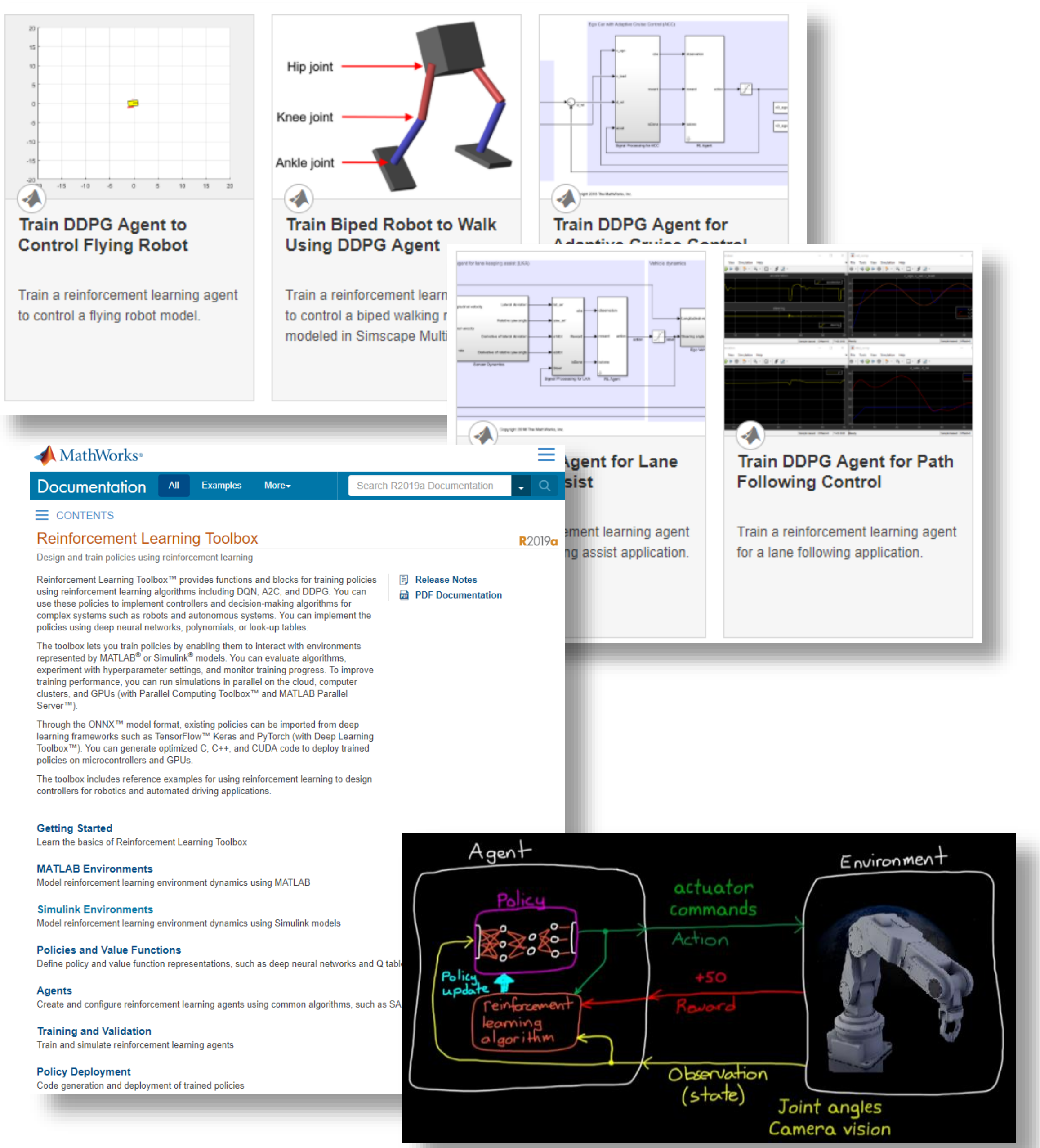
Extensible Environment Interface

- env = rlFunctionEnv(obsInfo,actInfo,stepfcn,resetfcn)
  - obsInfo: Observation Specification
  - actInfo: Action Specification
  - stepfcn: Function handle for stepping the environment
  - resetfcn: Function handle for resetting the environment

- Subclassing from rl.env.MATLABEnvironment
  - Custom MATLAB Environments
  - Interfacing with 3rd party simulators (e.g. OpenAI Gym)

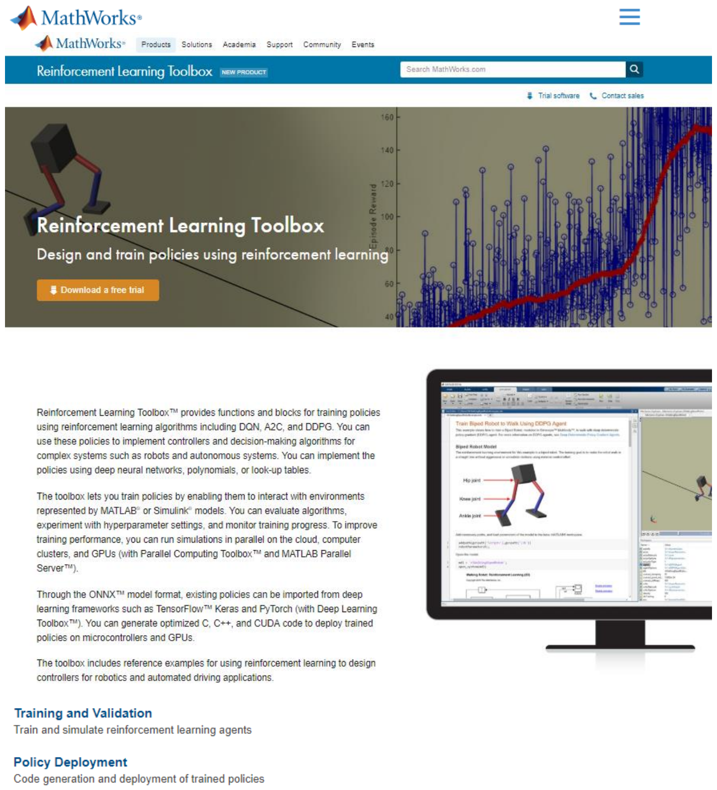
Resources

- Reference examples for controls, robotics, and autonomous system applications
- Documentation written for engineers and domain experts
- Tech Talk video series on reinforcement learning concepts for engineers



Reinforcement Learning Toolbox
New in R2019a

- Built-in and custom algorithms for reinforcement learning
- Environment modeling in MATLAB and Simulink
- Deep Learning Toolbox support for designing policies
- Training acceleration through GPUs and cloud resources
- Deployment to embedded devices and production systems
- Reference examples for getting started



Questions?