MATLAB EXPO

Master Class: Driving into the Future: AI-Enabled Autonomous Systems

Dr Rishu Gupta, MathWorks

Peeyush Pankaj, MathWorks

MathWorks®

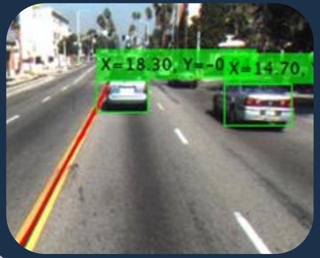
Agenda

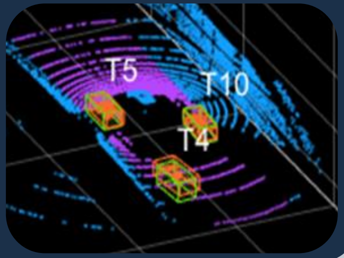
- Introduction to Autonomous systems
- Artificial Intelligence
  - **Deep Learning: Acceleration of motion planning using deep learning**
- Reinforcement Learning
  - **Developing controller for automated parking valet**
- Deployment of AI models to embedded devices

Key subsystems of an autonomous systems

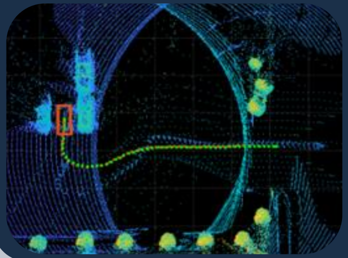
Algorithms

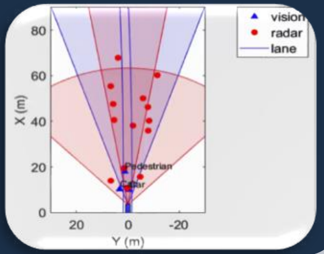
Perception



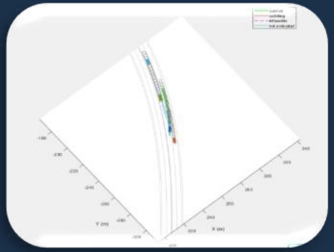


Sensor Fusion

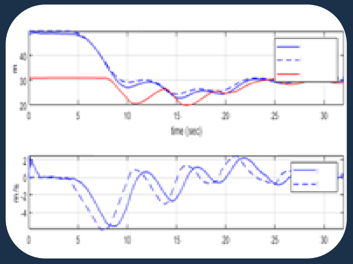


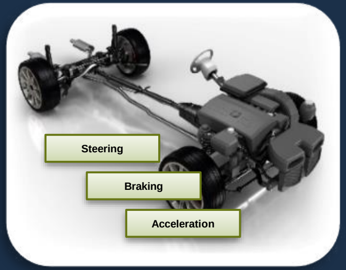


Planning



Decision & Controls





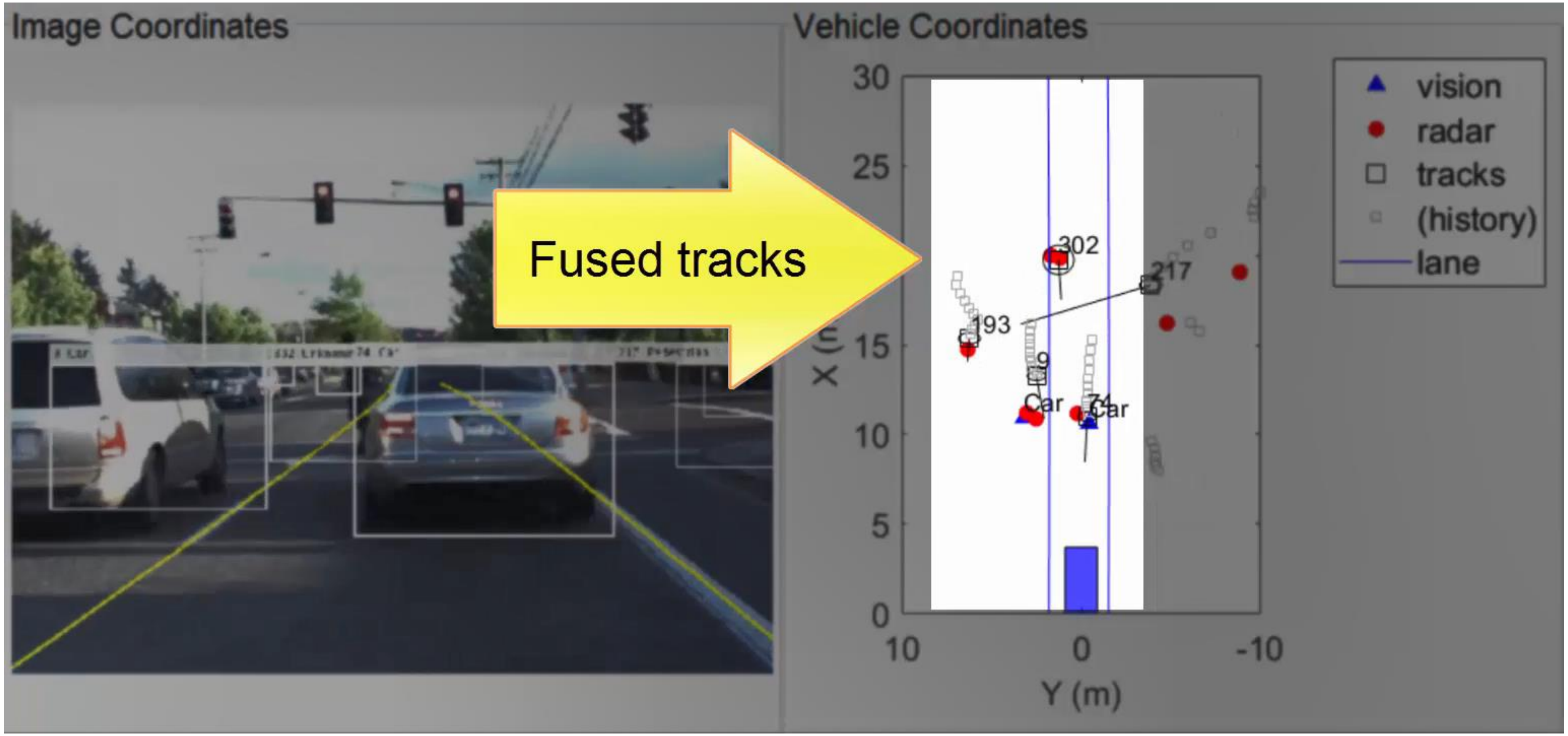
Key subsystems of an automated driving system

Perception

Key subsystems of an automated driving system

Perception

Sensor Fusion

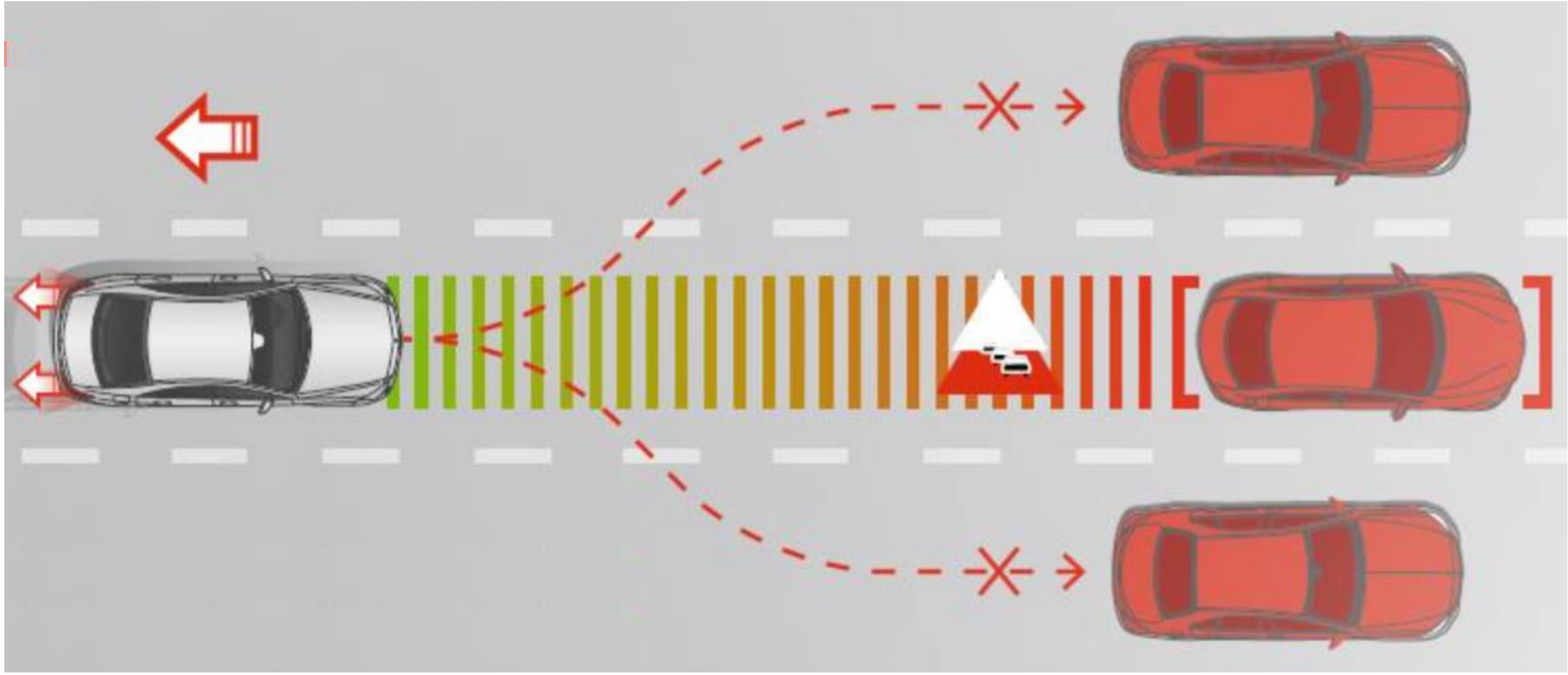


Key subsystems of an automated driving system

Perception

Sensor Fusion

Planning



Key subsystems of an automated driving system

Perception

↓

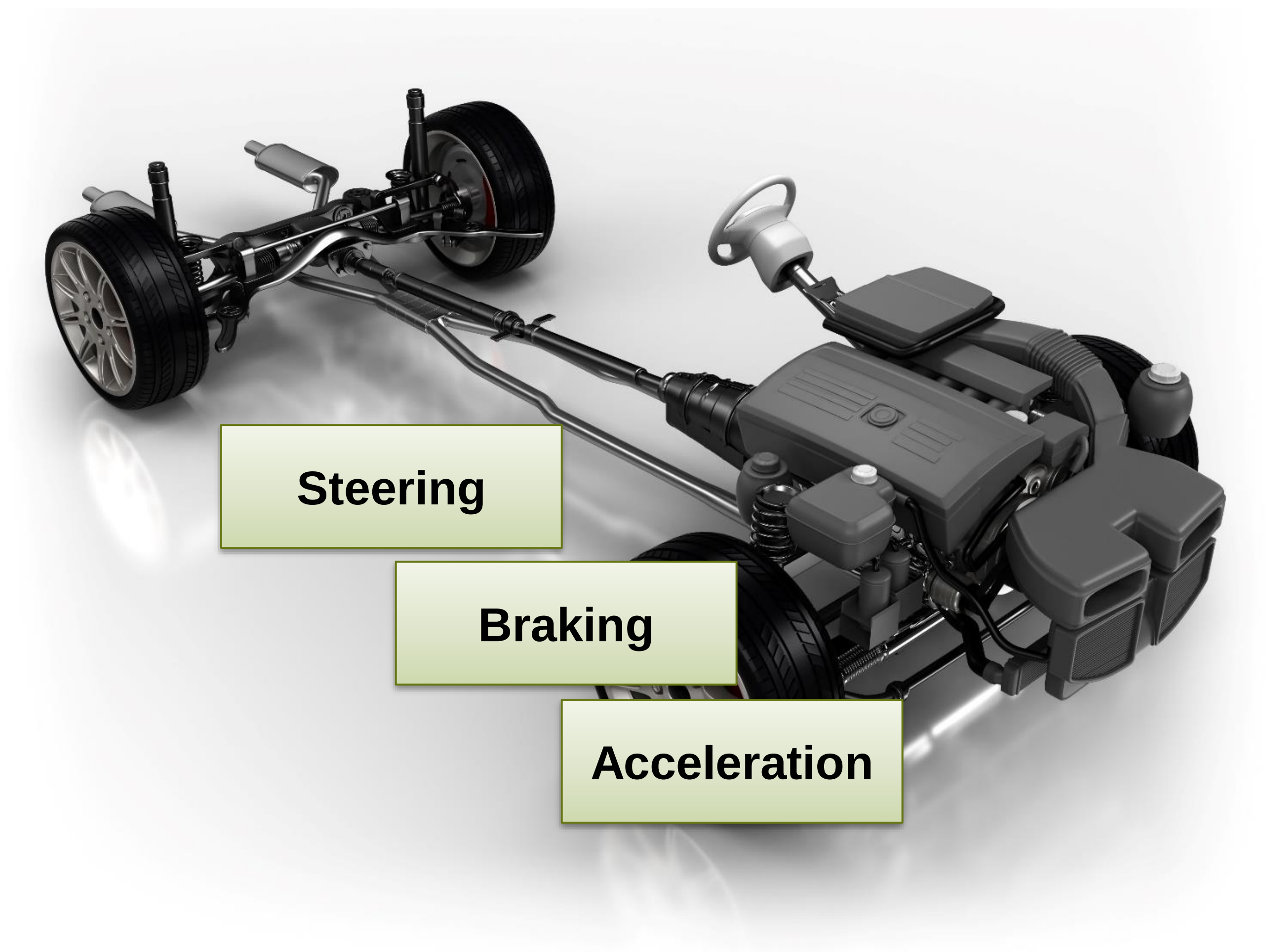
Sensor Fusion

↓

Planning

↓

Controls



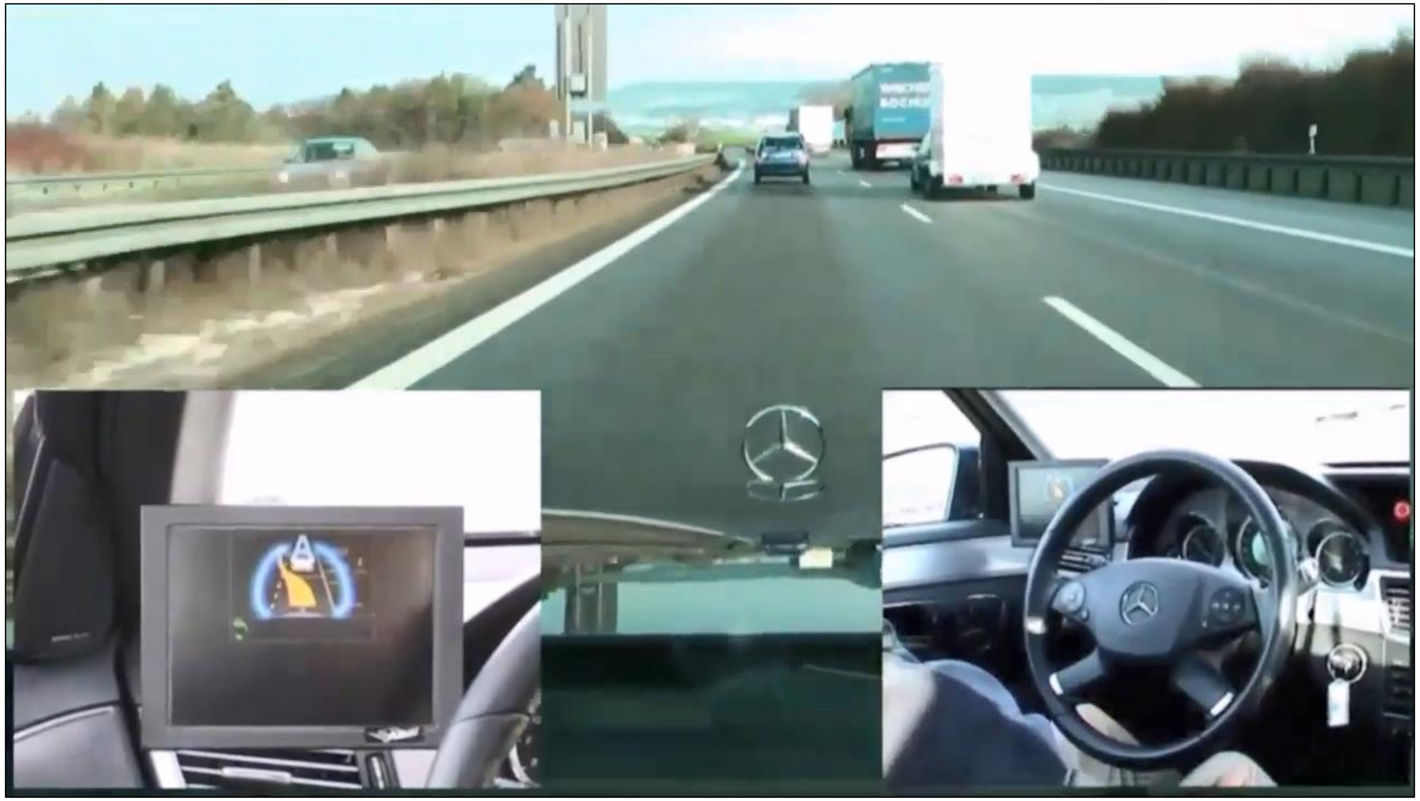
Key subsystems of an automated driving system

Perception

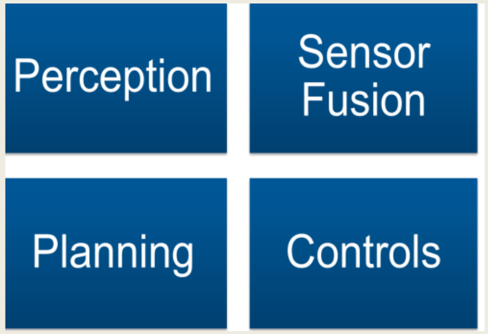
Sensor Fusion

Planning

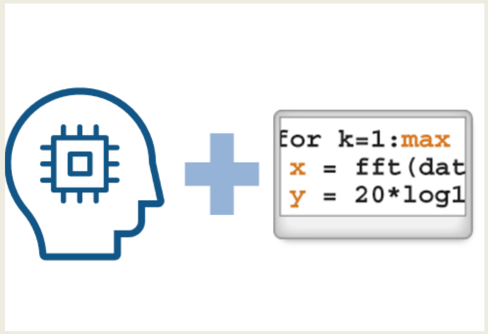
Controls



Challenges in developing and testing autonomous systems



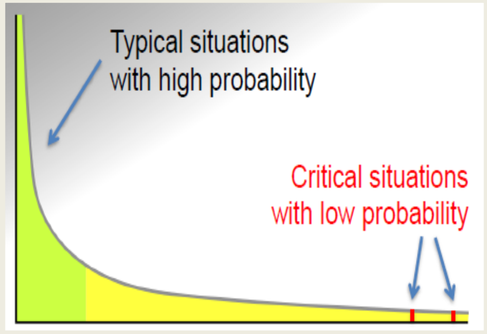
Complexity of subsystems



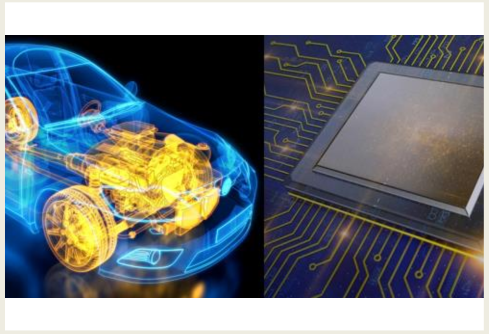
AI and traditional algorithms

Diverse, disconnected tools and workflows

Recreating difficult real-world scenarios



Long tail of edge cases*

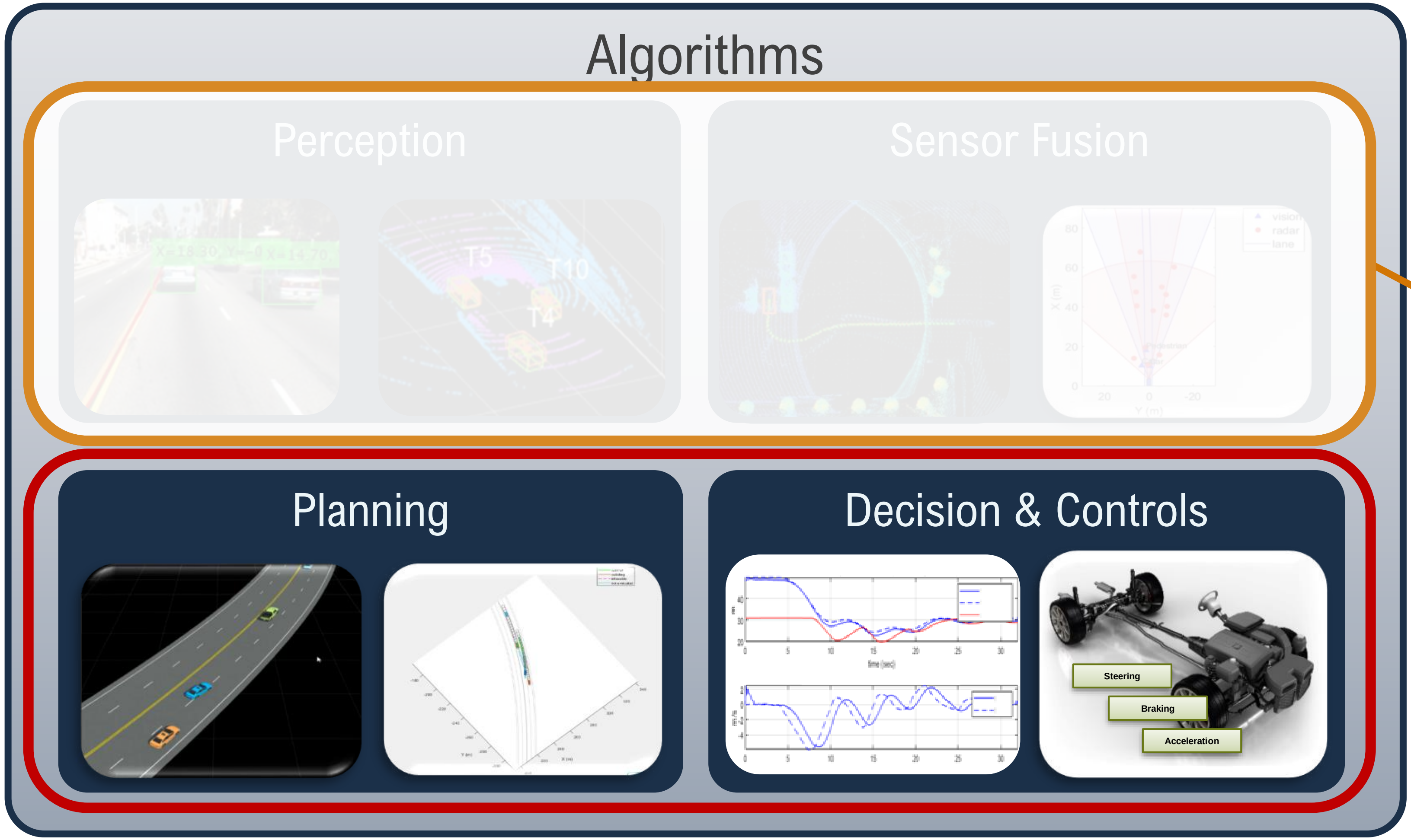


Billions of kms for reliability test**

* Saarland University

[RAND report](#)**

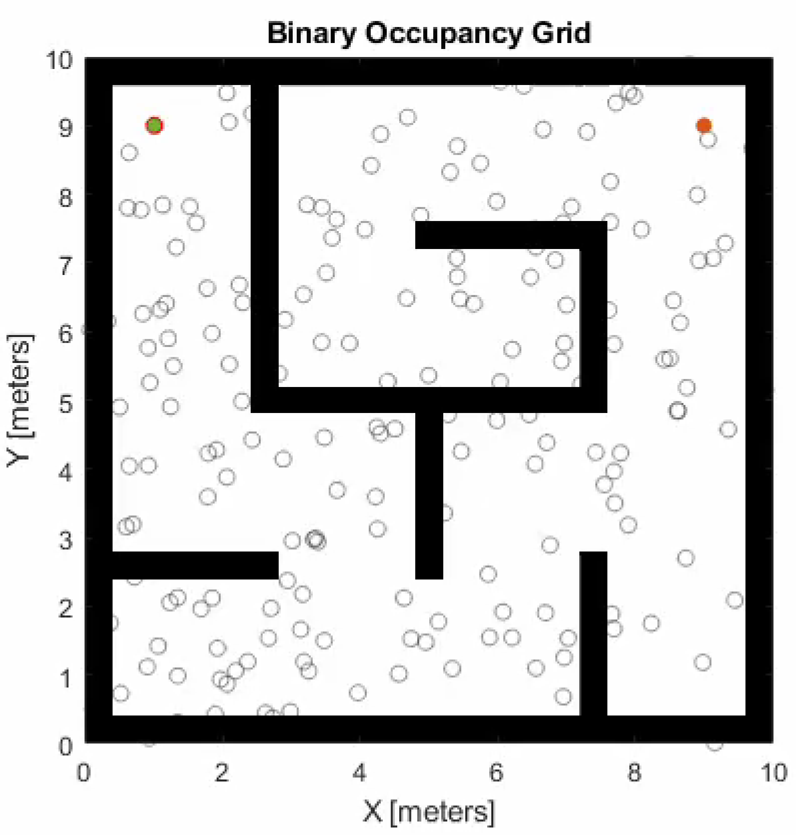
Artificial Intelligence as Key Enabler for Autonomous Systems



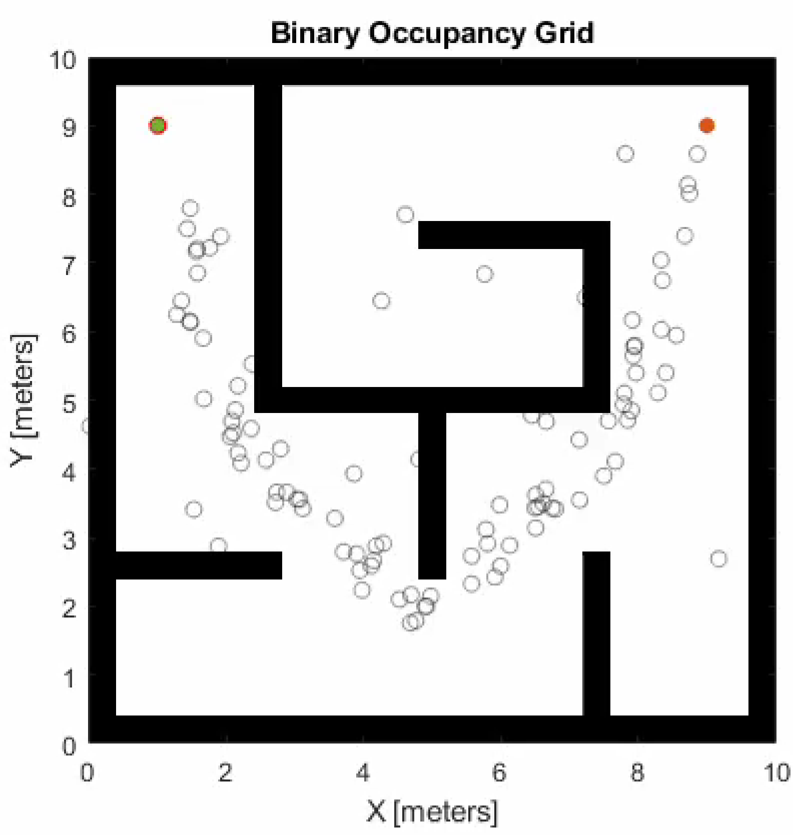
Object detection
Semantic segmentation
GANs
And so on …

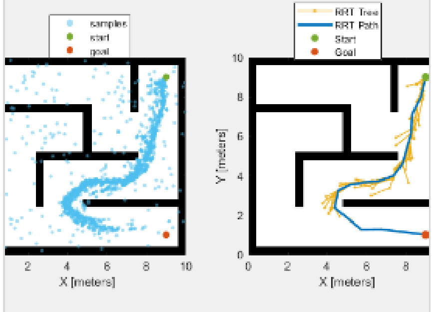
Accelerate Motion Planning with Deep Learning

RRT* Path
Uniform Sampling (λ=0)



RRT* Path
DL Based Sampling (λ = 0.9)



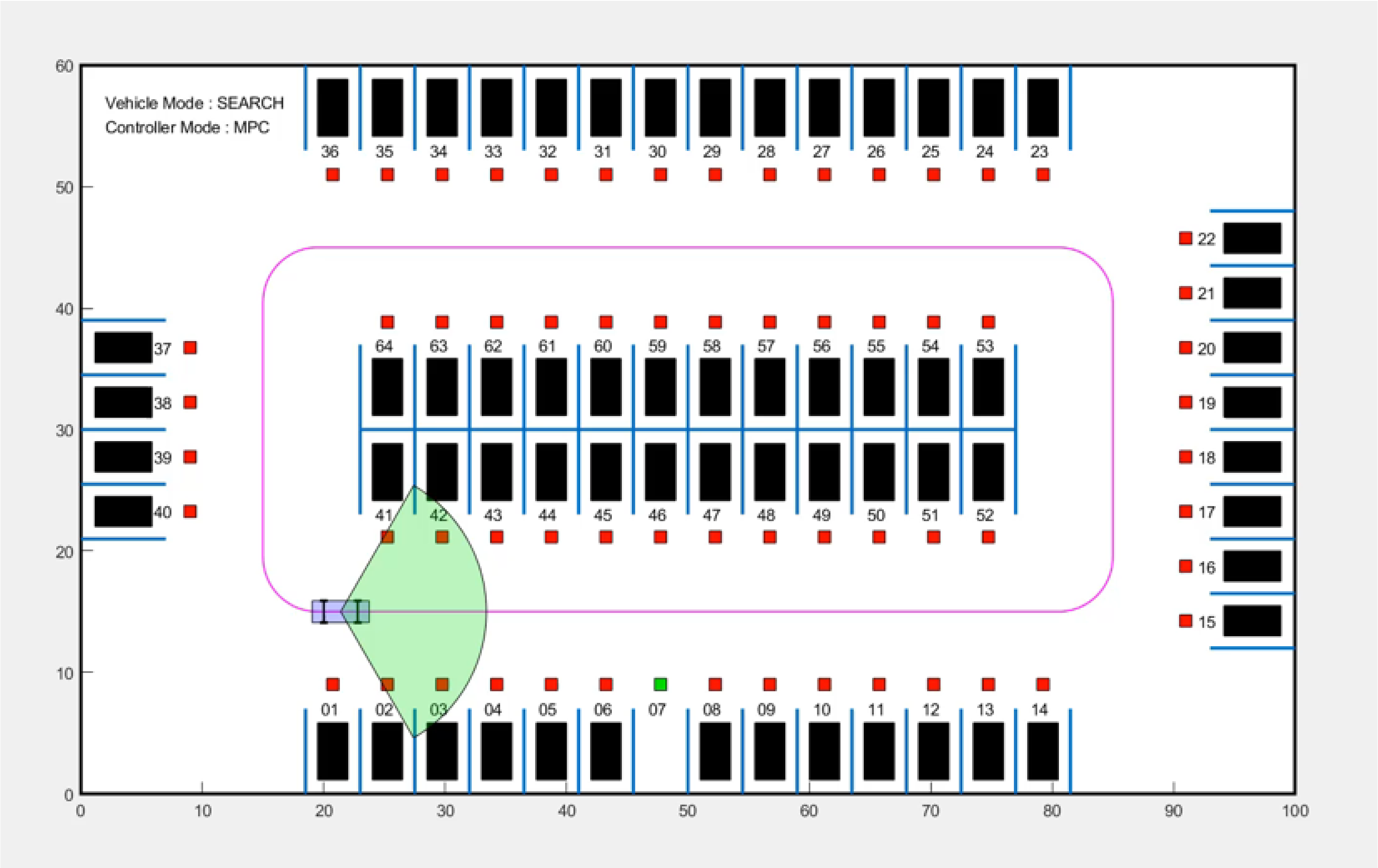


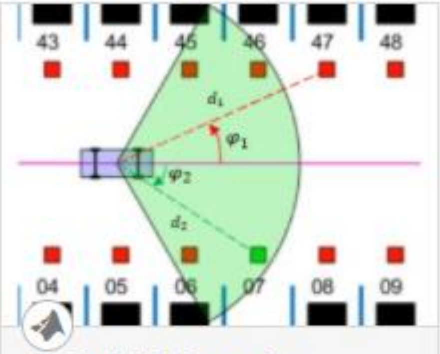
Accelerate Motion Planning with Deep-Learning-Based Sampler

The example demonstrates how to augment sampling-based planners such as RRT (rapidly-exploring random tree) and RRT* with a deep-

Open Live

Automated Parking Valet using Reinforcement Learning





Train PPO Agent for Automatic Parking Valet

Train a reinforcement learning agent to park a car in an open parking space.

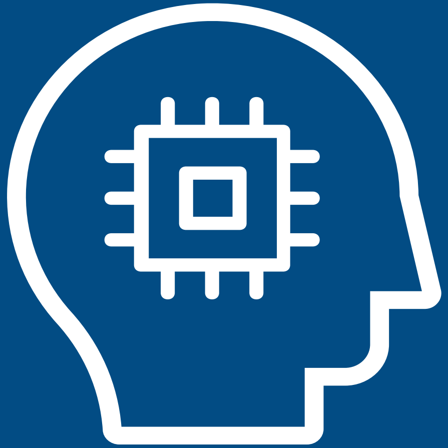
Agenda

- Introduction to Autonomous systems
- Artificial Intelligence
  - **Deep Learning: Acceleration of motion planning using deep learning**
- **Reinforcement Learning**
  - **Developing controller for automated parking valet**
- Deployment of AI models to embedded devices

Machine Learning is a key technology driving the AI megatrend

ARTIFICIAL INTELLIGENCE (AI)

Any technique that enables machines to mimic human intelligence

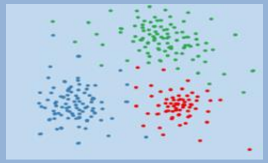


MACHINE LEARNING

Statistical methods that enable machines to “learn” tasks from data without explicitly programming

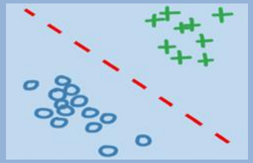
UNSUPERVISED LEARNING

(No Labeled Data)



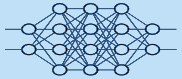
SUPERVISED LEARNING

(Labeled Data)



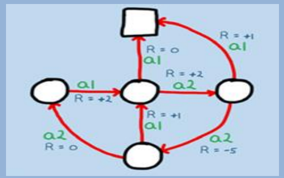
DEEP LEARNING

(Neural networks with many layers)



REINFORCEMENT LEARNING

(Interaction Data)



Brief Overview for AI-driven system design

Data Preparation

- Data cleansing and preparation
- Human insight
- Simulation-generated data

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Simulation & Test

- Integration with complex systems
- System simulation
- System verification and validation

Deployment

- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

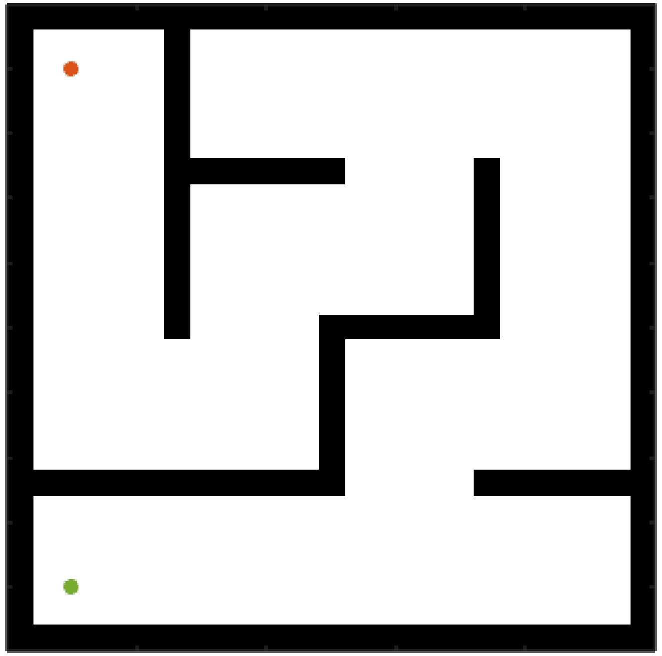
- **Labeller apps**
- **Unreal co-simulation**
- **Data generation- Virtual sensor modelling (Camera, LIDAR, RADAR)**
- **Simscape**

- Deep network designer
- Experiment manager/ Classification Learner
- Interoperability between DL toolbox and other frameworks

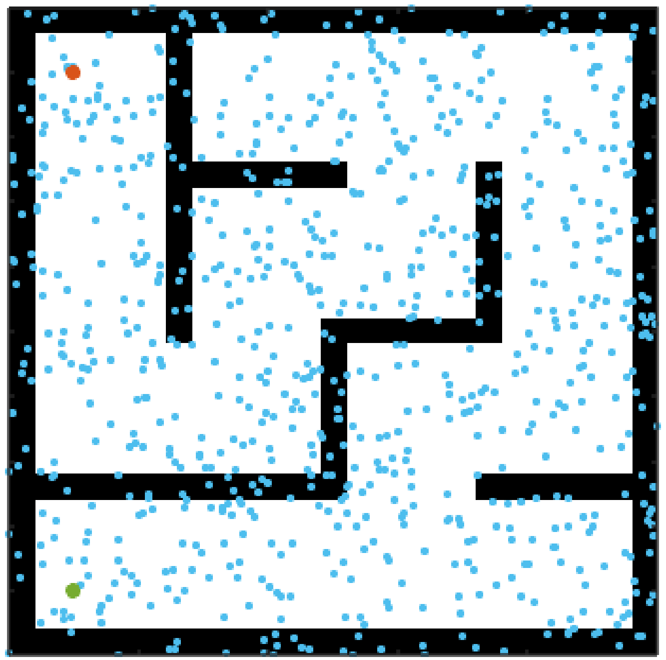
- Reference application for integration
- Integrating AI into Simulink

- CPUs, (ARM_ACL)
- Cloud (on-premise, service providers)
- Microservice Docker Containers
- Deploy Imported TensorFlow Model with MATLAB Compiler

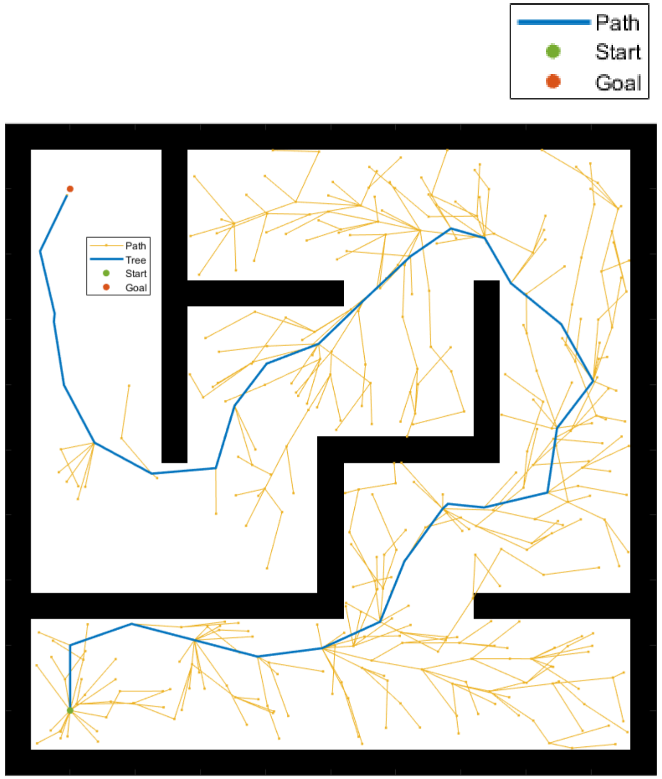
Accelerate Motion Planning with Deep Learning



Random Maze Dataset
representing occupancy map,
start & goal locations



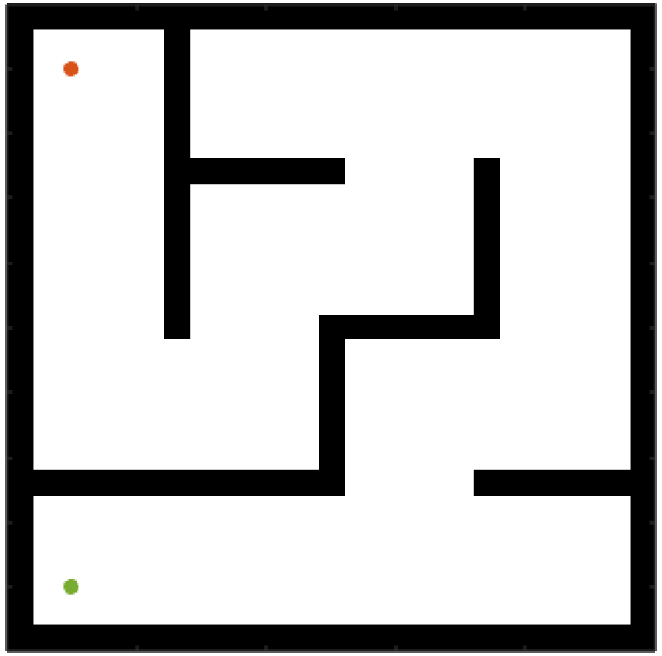
Uniformly Sampled space
(used by conventional motion
planning algorithms e.g., RRT/ RRT*)



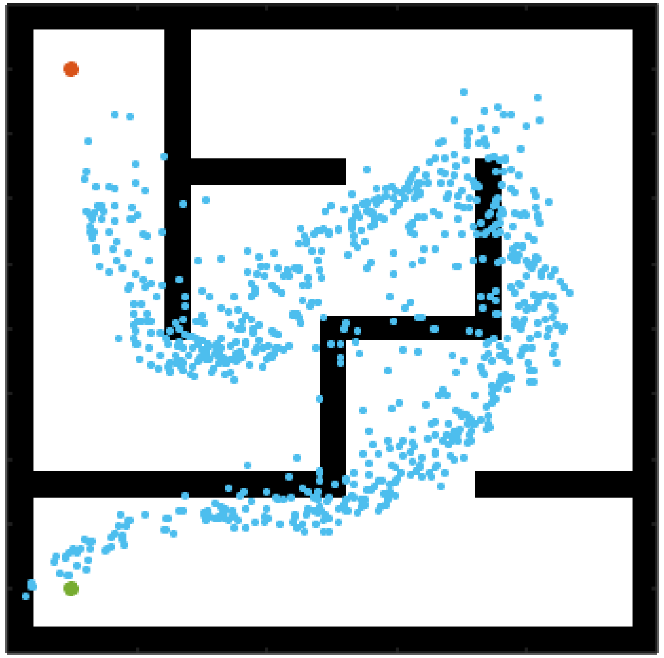
RRT*
(Rapidly Exploring Random Tree)

Can RRT* with Deep Learning based Sampler outperform the one with uniform sampling ?

Accelerate Motion Planning with Deep Learning



Random Maze Dataset
representing occupancy map,
start & goal locations



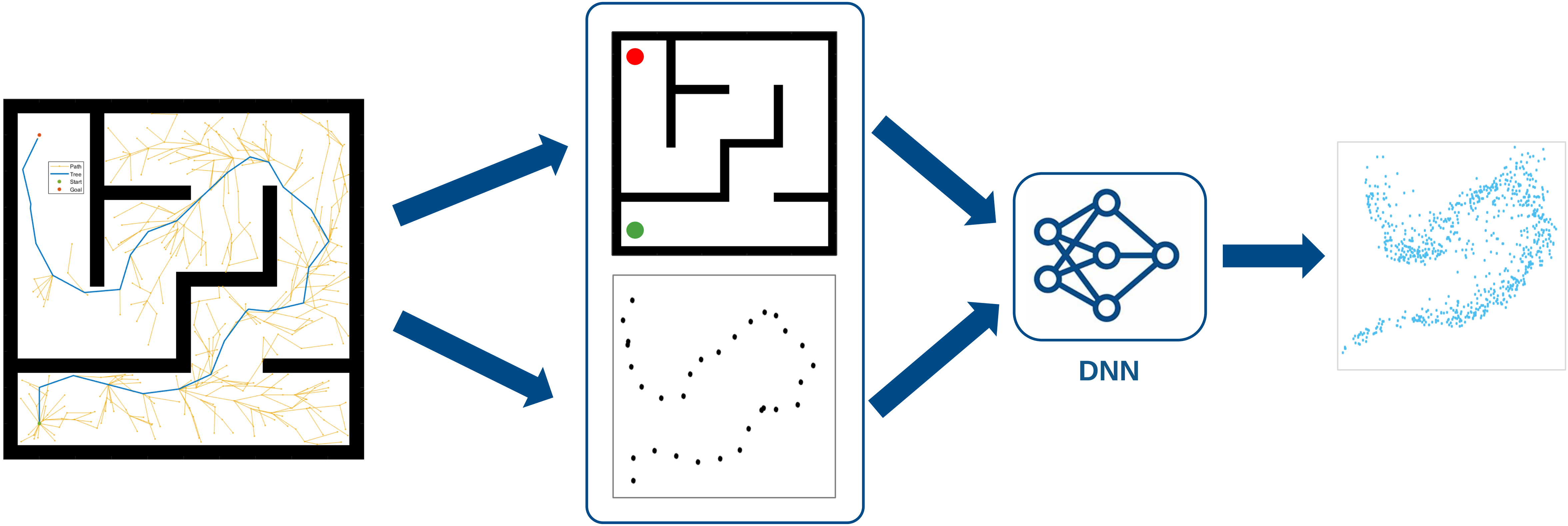
DL-Based Sampler

RRT*
(Rapidly Exploring Random Tree)

Can RRT* with Deep Learning based Sampler outperform the one with uniform sampling ?

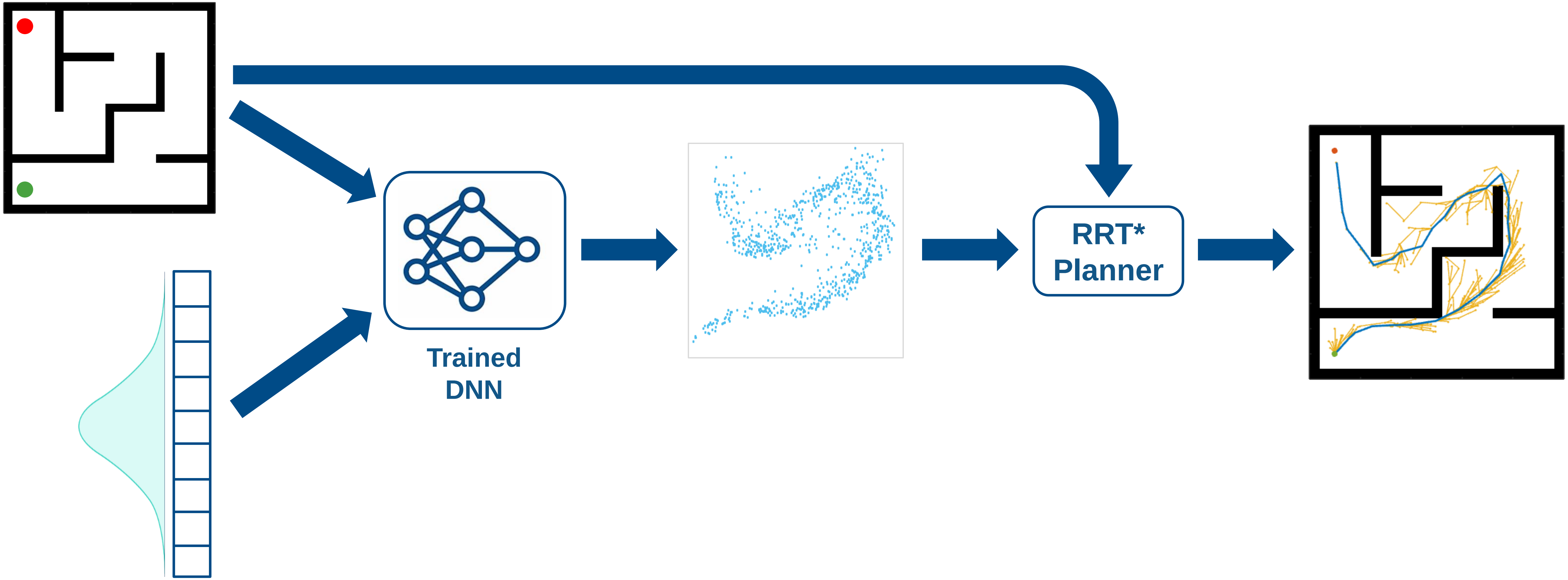
AI Workflow

Train: Iterate till you find the best model using historical data



AI Workflow

Predict: Integrate trained models into applications

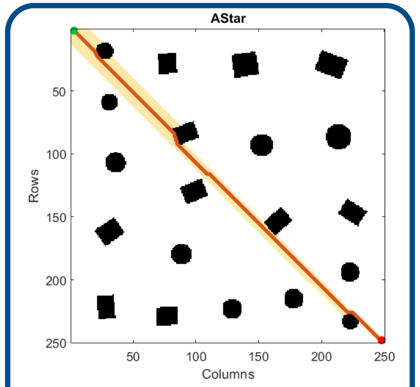


Path Planning

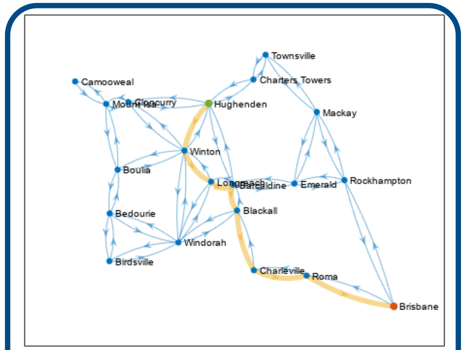
Navigation Toolbox
Design, simulate, and deploy algorithms for autonomous navigation

1 Data Preparation

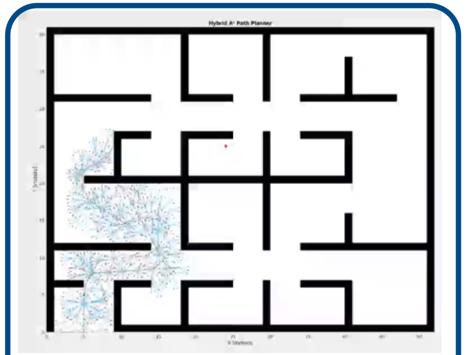
- Data cleansing and preparation
- Human insight
- Simulation-generated data



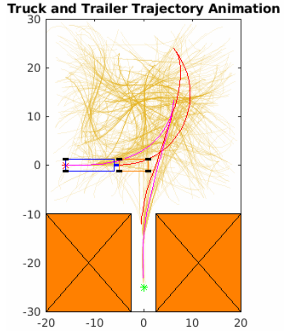
[Grid-based A*](#)
Plan the shortest collision-free path through an obstacle grid map



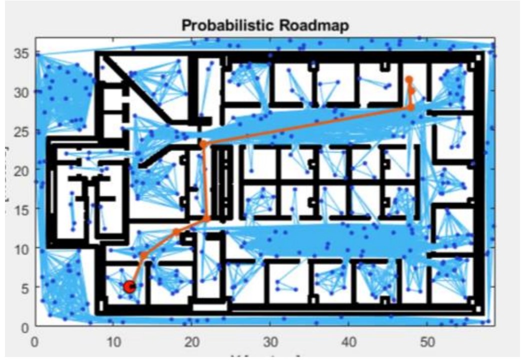
[Graph-based A*](#)
Plan shortest routes in a graph network



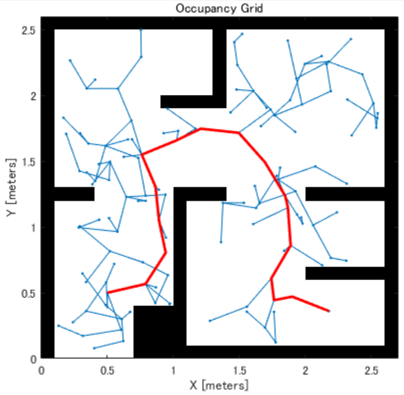
[Hybrid A*](#)
Grid search method to generate a smooth path in a given 2-D space for vehicles with nonholonomic constraints



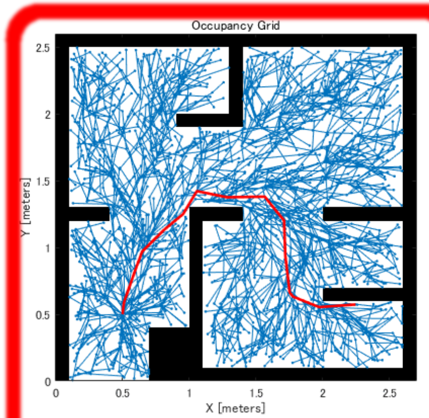
[Control-based RRT](#)
Plan kinematically and dynamically feasible paths with custom kinematics and control policies



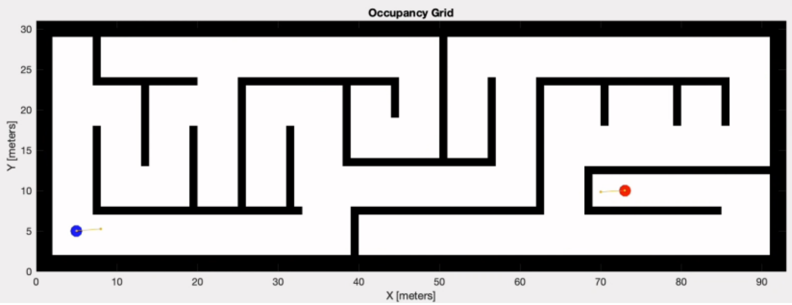
[PRM (Probabilistic Roadmap)](#)
Sampling-based planning. Search for shortest path in a densely constructed graph



[RRT (Rapidly-Exploring Random Tree)](#)
Nearest neighbor search tree incrementally from samples randomly drawn from a given state space



[RRT*](#)
Converges to an optimal solution in terms of the state space distance



[Bidirectional RRT](#)
Variant of RRT creating two search trees starting from both start and goal states simultaneously

Data Generation with RRT*

Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation

mapMaze
Generate random 2-D maze map
Since R2021a

stateSpaceSE2
SE(2) state space
Since R2019b

validatorOccupancyMap
State validator based on 2-D grid map
Since R2019b

binaryOccupancyMap
Create occupancy grid with binary values

plannerRRTStar
Create an optimal RRT path planner (RRT*)
Since R2019b

1 **Data Preparation**

- Data cleansing and preparation
- Human insight
- Simulation-generated data

```
% Number of maps
numMaps = 2000;
% Map size in metres (assume height = weight)
mapSize = 10;
% Number of states per map to exported
numStates = 100;
% Create stateSpace and stateValidator
stateSpace = stateSpaceSE2;
stateSpace.StateBounds = [maps{1}.XWorldLimits; maps{1}.YWorldLimits; [-pi, pi]];
stateValidato
stateValidato
% Run the plannerRRTStar
for i = 1:numMaps
    waitbar(i/numMaps,f,"Generating samples...");

    % Inflate obstacle to get safe paths
    map = binaryOccupancyMap(maps{i},10);
    inflate(map, 4, 'grid')
..
pRes);

% Create planner object
planner = plannerRRTStar(stateSpace, stateValidator);
planner.ContinueAfterGoalReached = true; % optimize
planner.MaxConnectionDistance = 1;
planner.GoalReachedFcn = @examplerHelperCheckIfGoalReached;
planner.MaxIterations = 2000;
```

Data Generation with RRT*

1 **Data Preparation**

- Data cleansing and preparation
- Human insight
- Simulation-generated data

Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation

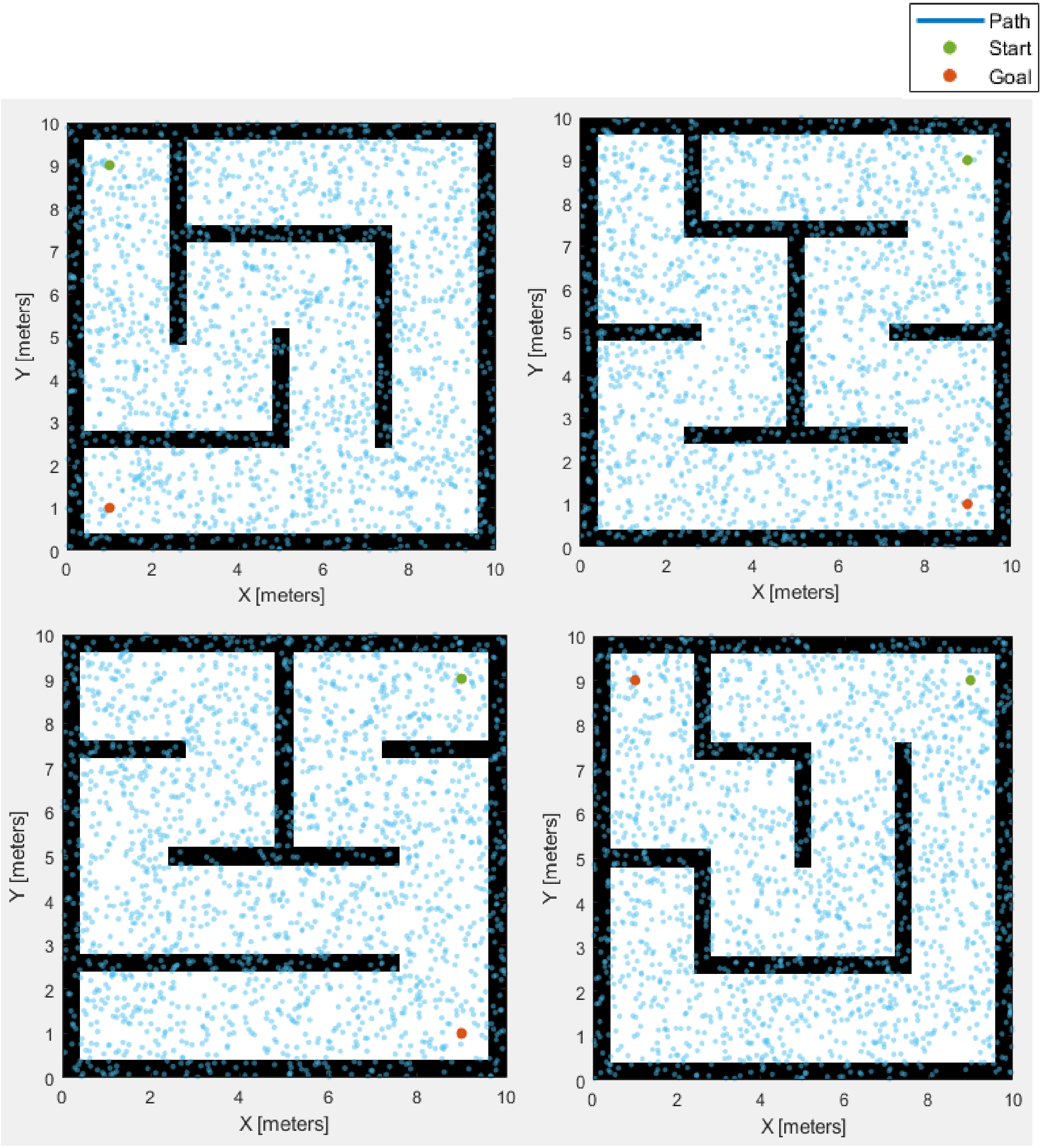
mapMaze
Generate random 2-D maze map
Since R2021a

stateSpaceSE2
SE(2) state space
Since R2019b

validatorOccupancyMap
State validator based on 2-D grid map
Since R2019b

binaryOccupancyMap
Create occupancy grid with binary values

plannerRRTStar
Create an optimal RRT path planner (RRT*)
Since R2019b



*No. of samples to be generated = 2000

Data Generation with RRT*

1 **Data Preparation**

- Data cleansing and preparation
- Human insight
- Simulation-generated data

Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation

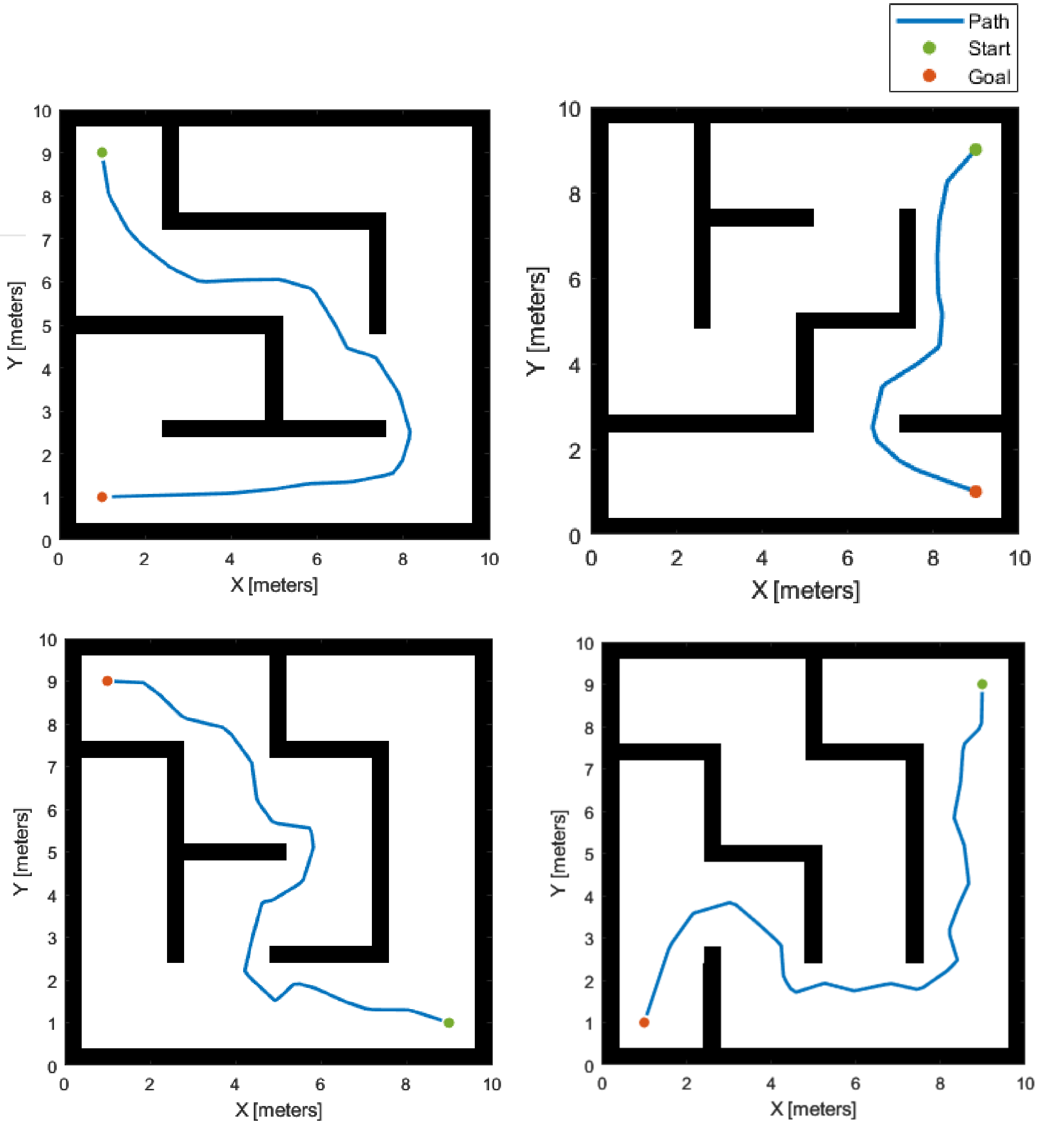
mapMaze
Generate random 2-D maze map
Since R2021a

stateSpaceSE2
SE(2) state space
Since R2019b

validatorOccupancyMap
State validator based on 2-D grid map
Since R2019b

binaryOccupancyMap
Create occupancy grid with binary values

plannerRRTStar
Create an optimal RRT path planner (RRT*)
Since R2019b



Start with a complete set of algorithms and pre-built models

2 AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Algorithms

Machine learning
Trees, Naïve Bayes, SVM…

Deep learning
CNNs, GANs, LSTM, MIMO…

Reinforcement learning
DQN, A2C, DDPG…

Regression
Linear, nonlinear, trees…

Unsupervised learning
K-means, PCA, GMM…

Predictive maintenance
RUL models, condition indicators…

Bayesian optimization

Pre-built models

Image classification models
AlexNet, GoogLeNet, VGG, SqueezeNet, ShuffleNet, ResNet, DenseNet, Inception…

Reference examples

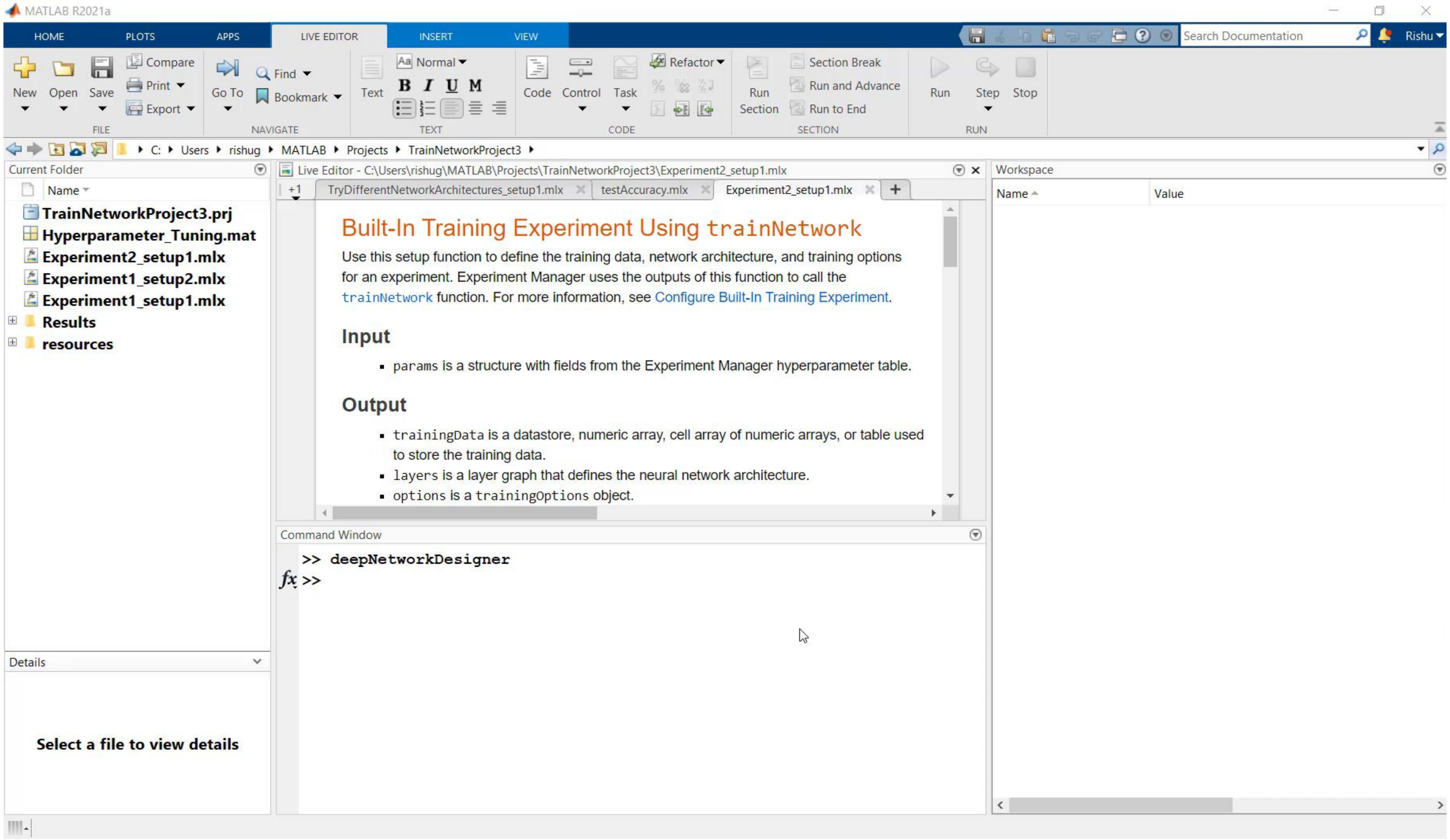
Object detection
Vehicles, pedestrians, faces…

Semantic segmentation
Roadway detection, land cover classification, tumor detection…

Signal and speech processing
Denoising, music genre recognition, keyword spotting, radar waveform classification…

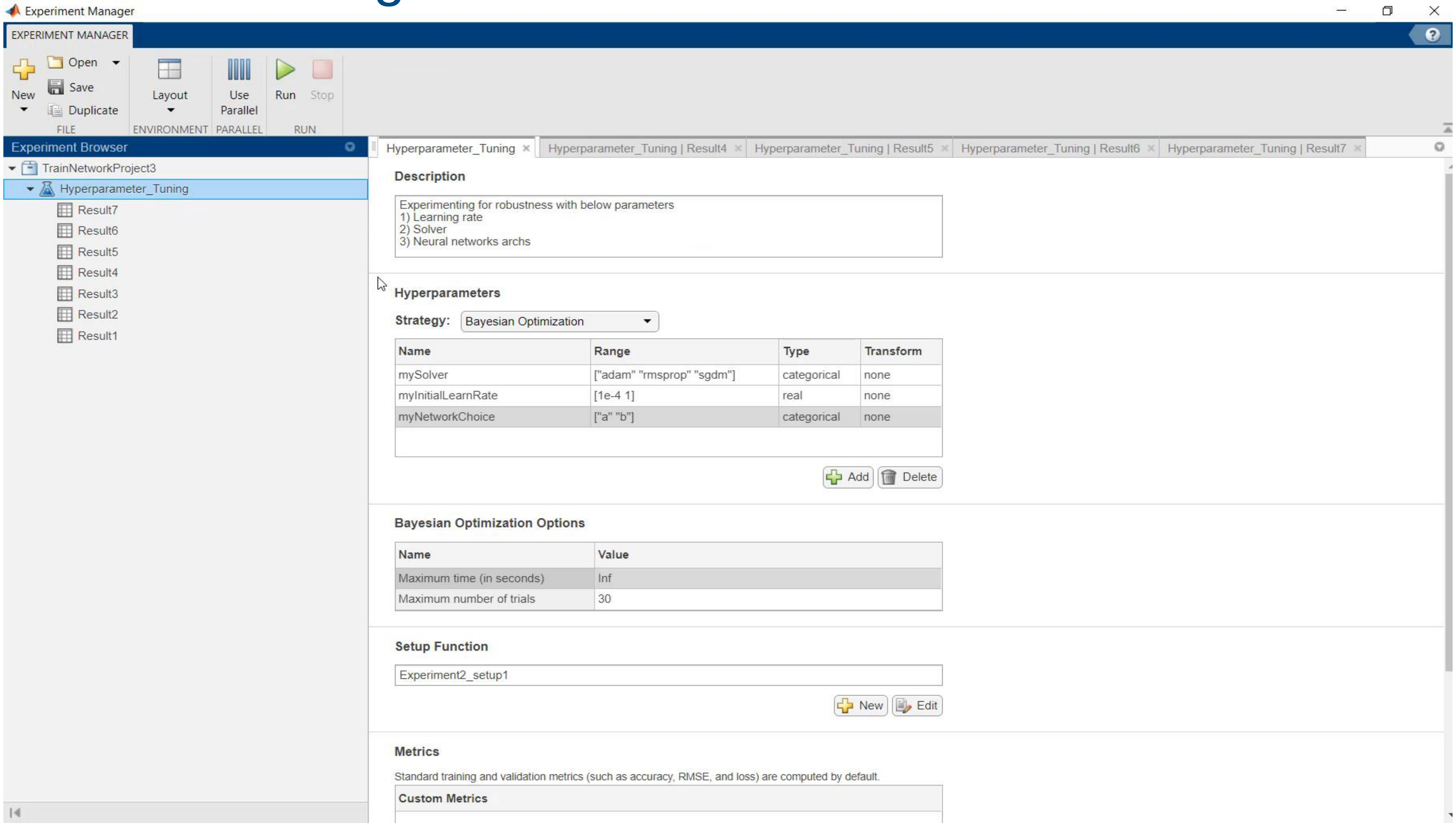
…and more…

Deep Neural Network Training



MATLAB EXPO

Experiment Manager

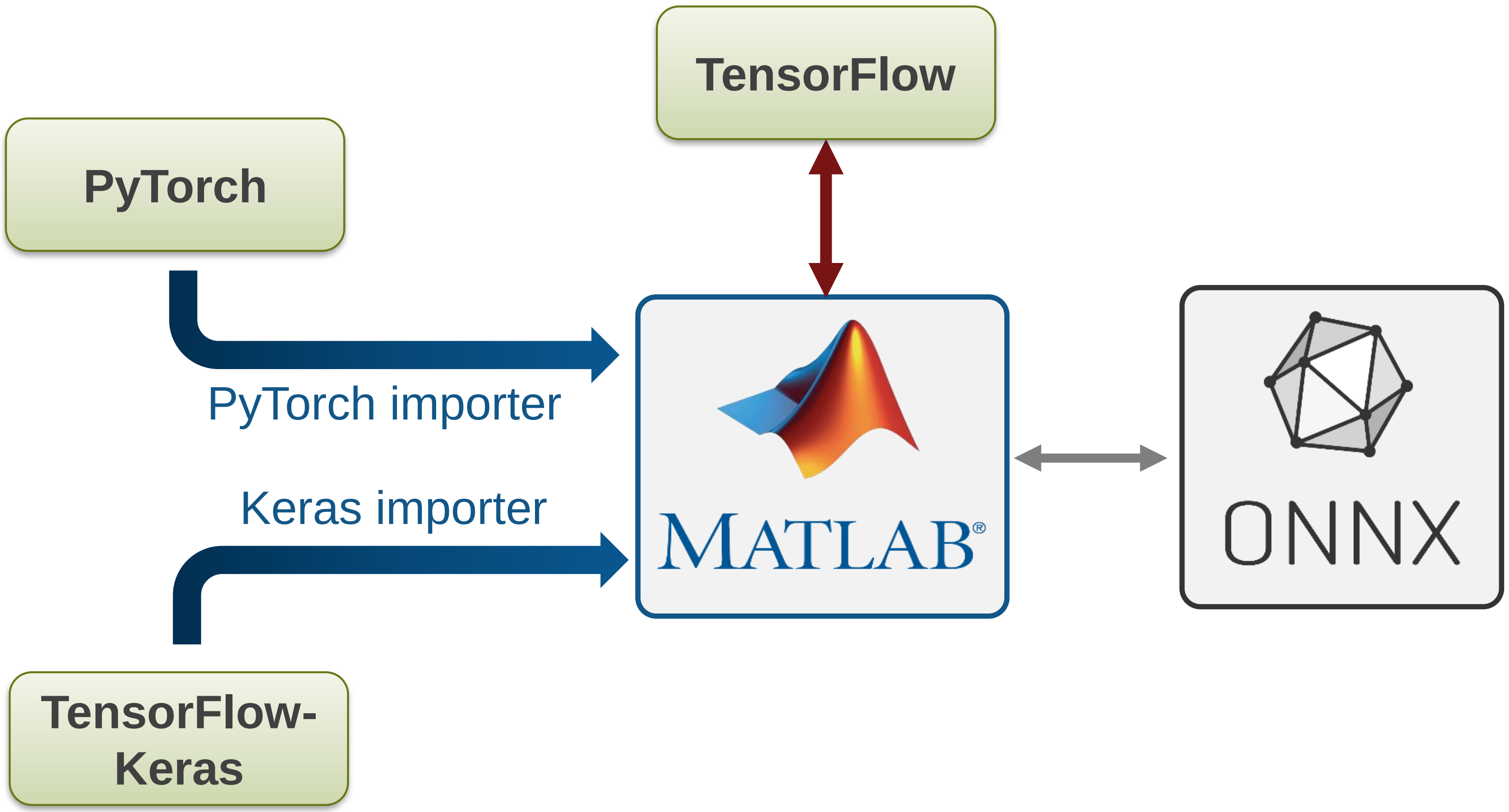


Importing Pretrained Network for Labelling Automation

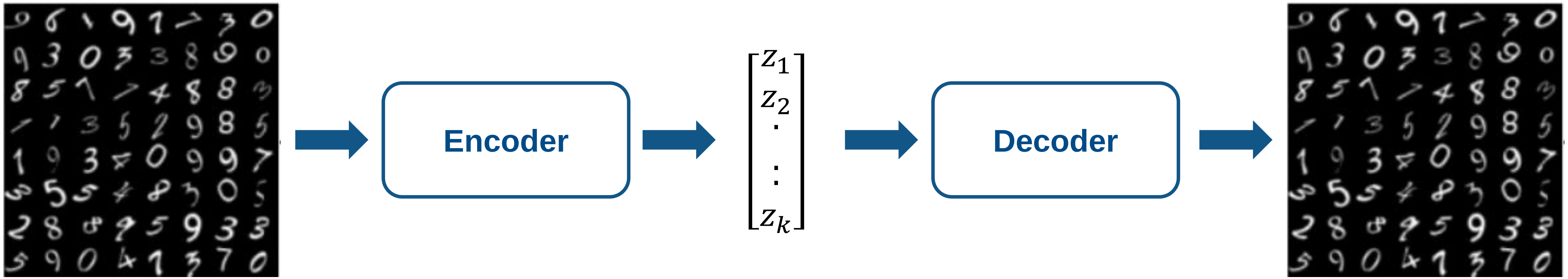
Framework Interoperability bridges the gap between data science, engineering and production

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability



Autoencoders

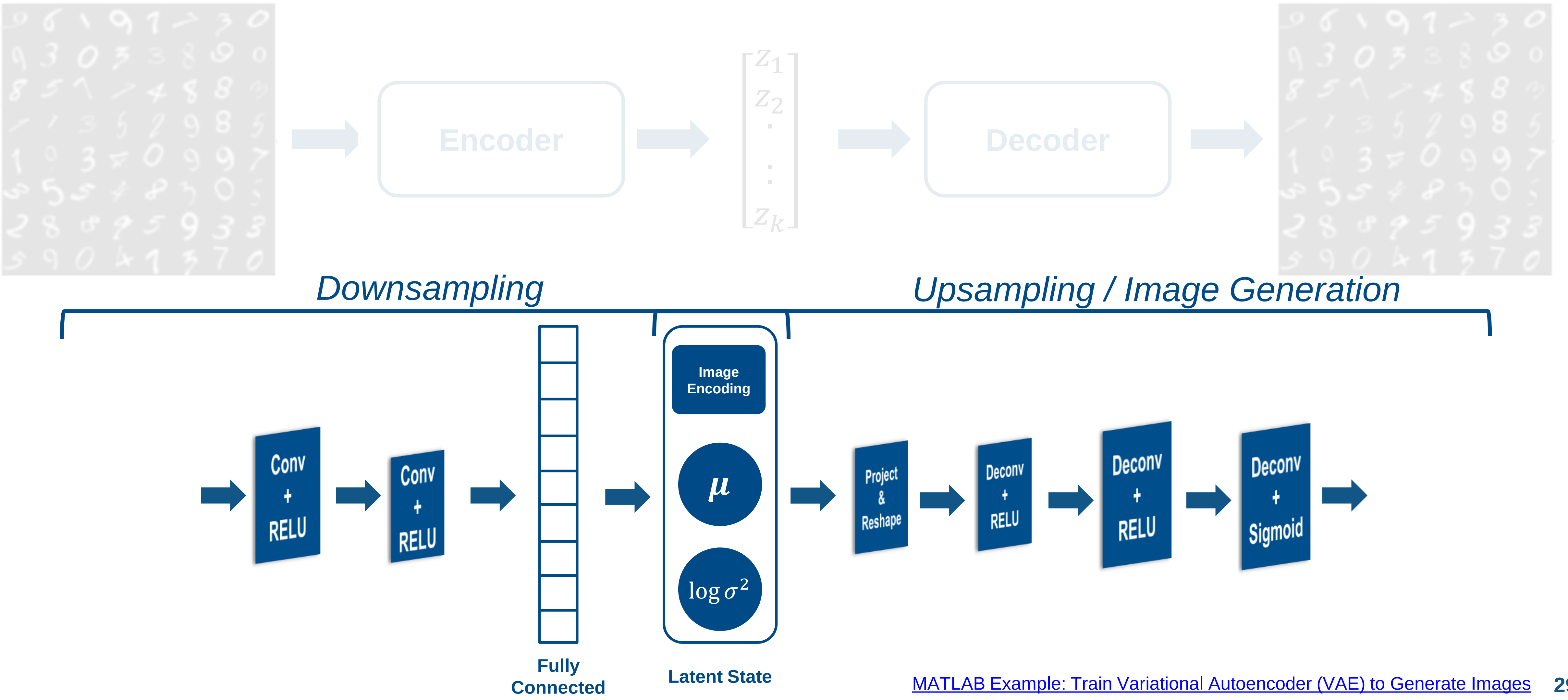


Input Images

Image Encodings /
Latent Representation

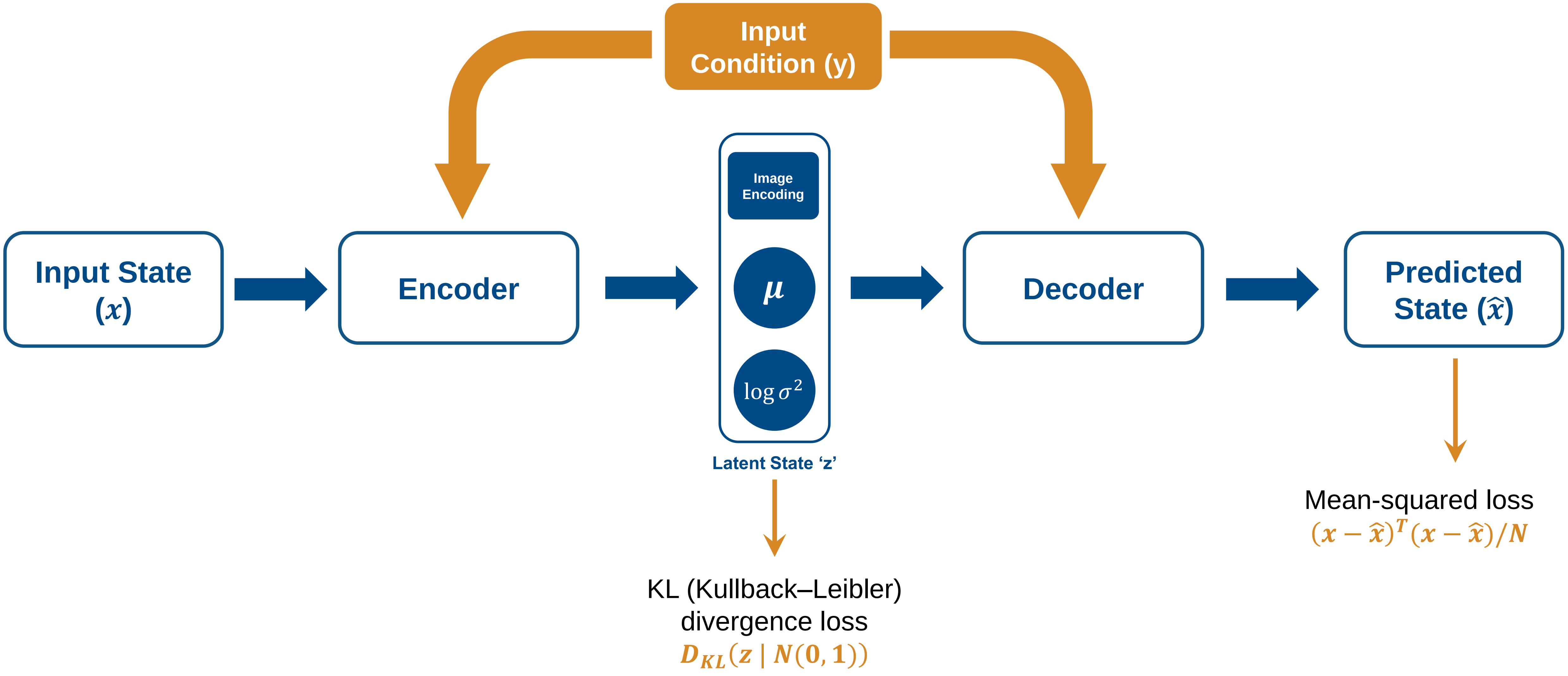
Reconstructed Images

Variational Autoencoders for Image Re-Generation

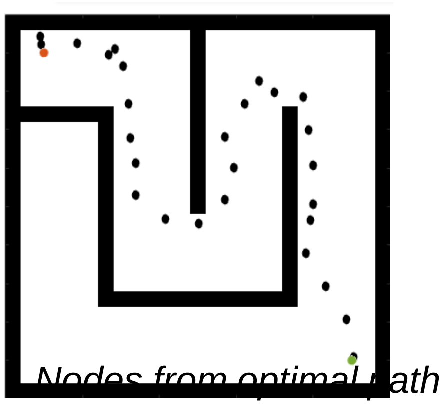


MATLAB Example: Train Variational Autoencoder (VAE) to Generate Images

Conditional Variational Autoencoders (CVAE)

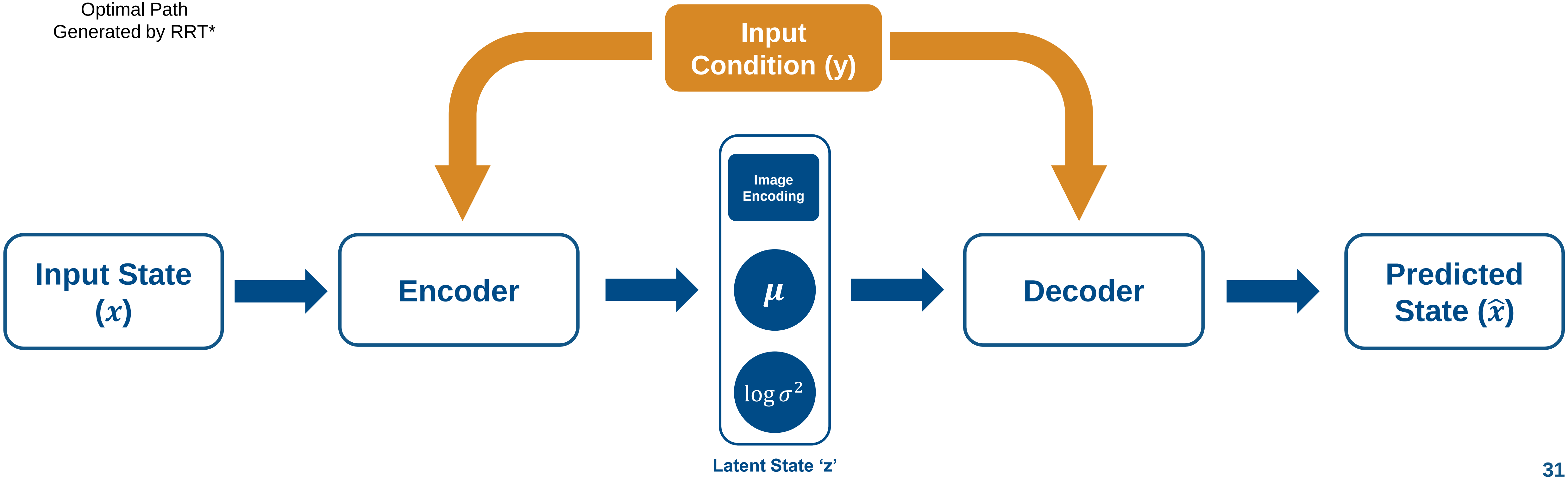


Conditional Variational Autoencoders (CVAE)

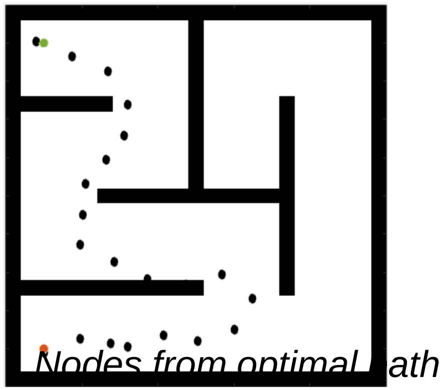


Optimal Path
Generated by RRT*

Occupancy Map
Start
Goal

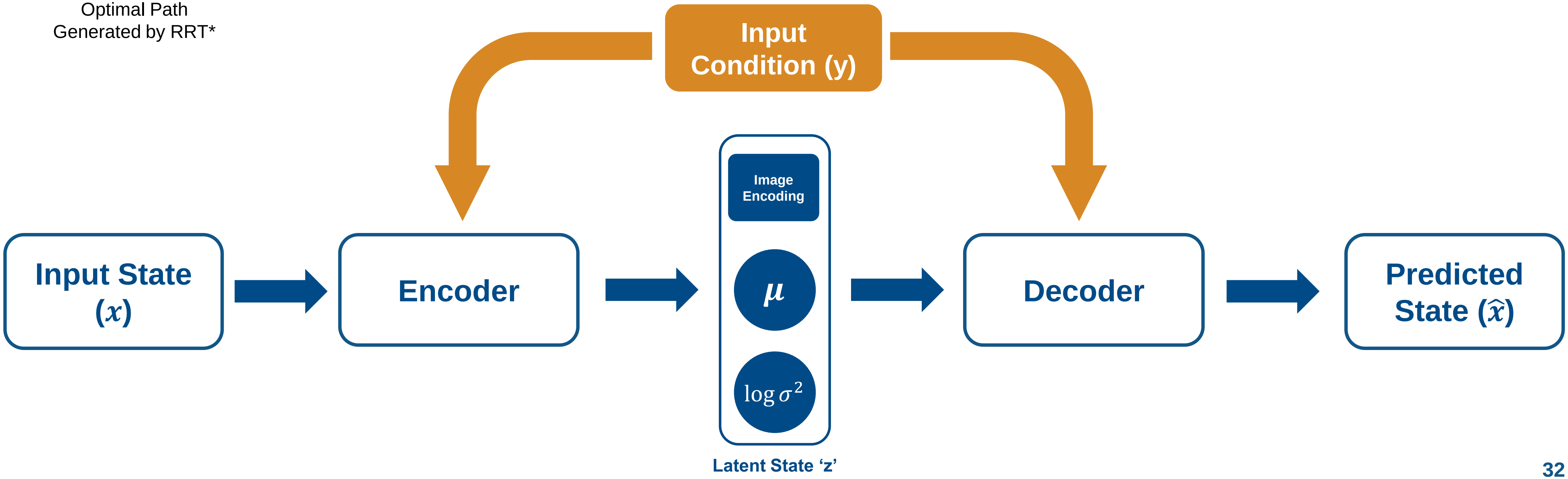


Conditional Variational Autoencoders (CVAE)

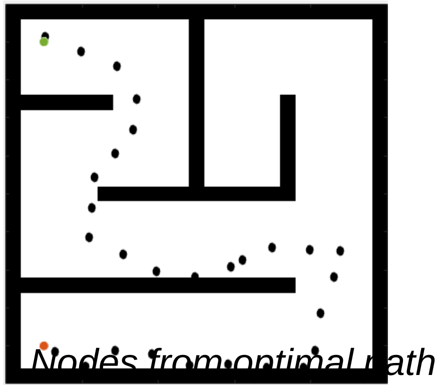


Optimal Path
Generated by RRT*

Occupancy Map
Start
Goal

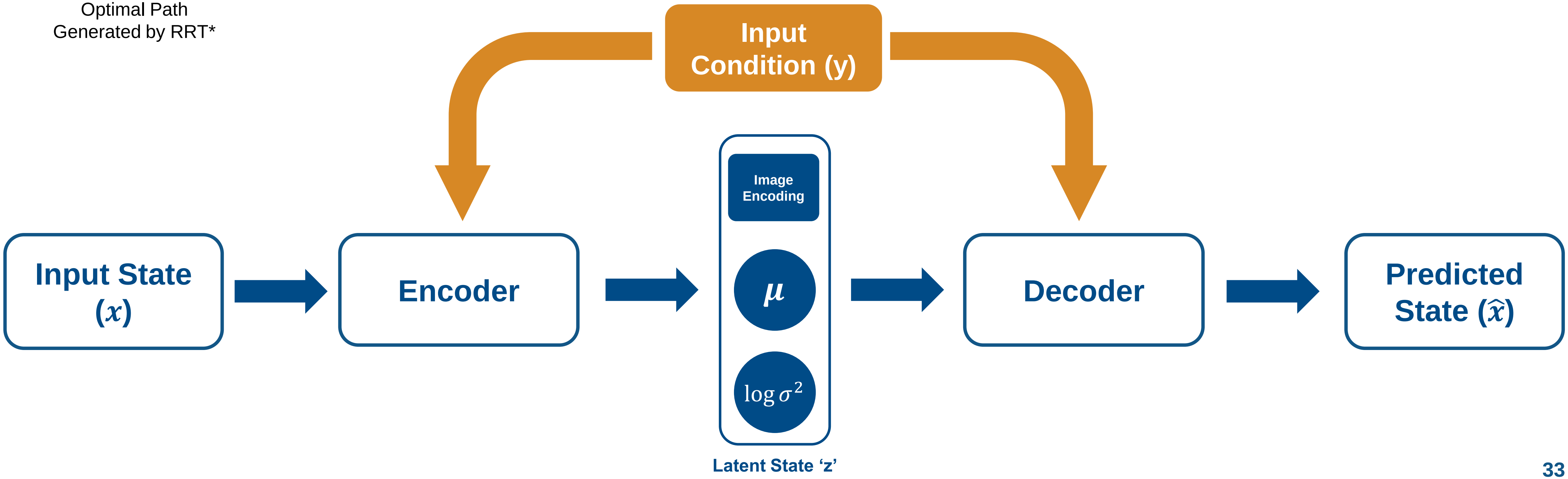


Conditional Variational Autoencoders (CVAE)

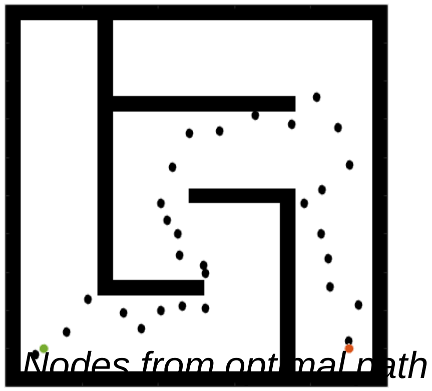


Optimal Path
Generated by RRT*

Occupancy Map
Start
Goal

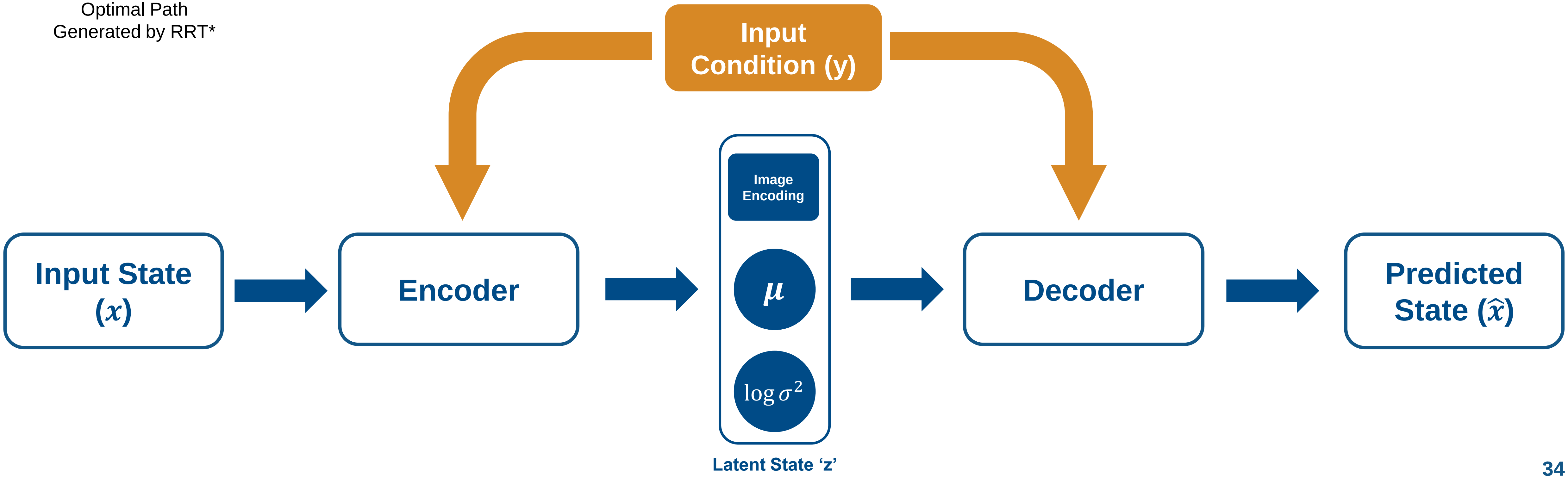


Conditional Variational Autoencoders (CVAE)



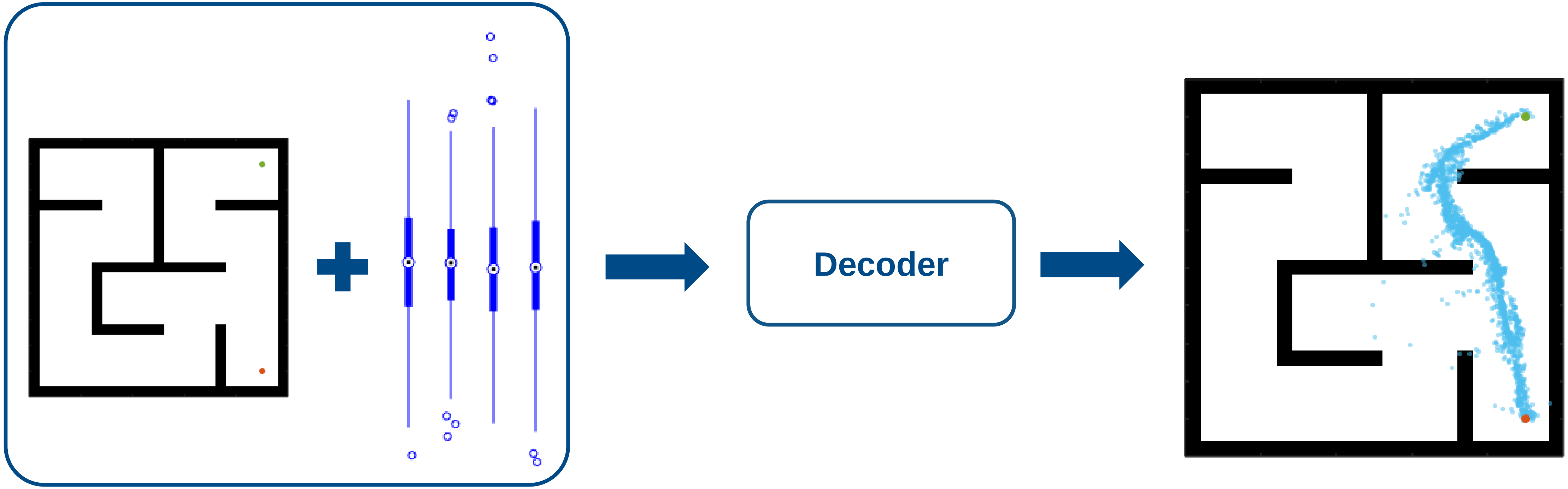
Optimal Path
Generated by RRT*

Occupancy Map
Start
Goal



Decoder Network for Generating Optimal States

```
% Predict states
statesLearned = predict(decoderNet, vertcat(condition, latentStates));
```

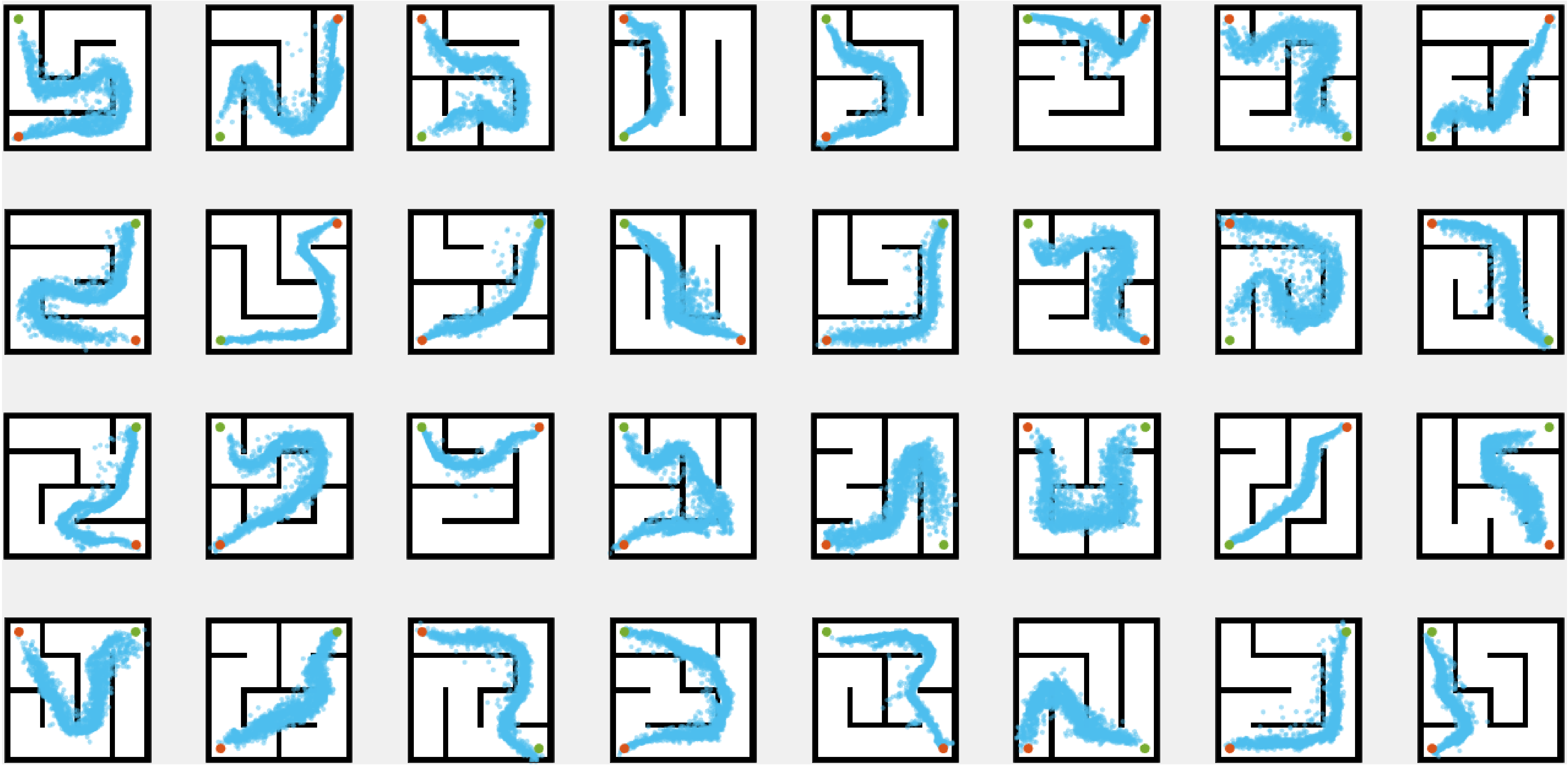


Occupancy Map
Start
Goal

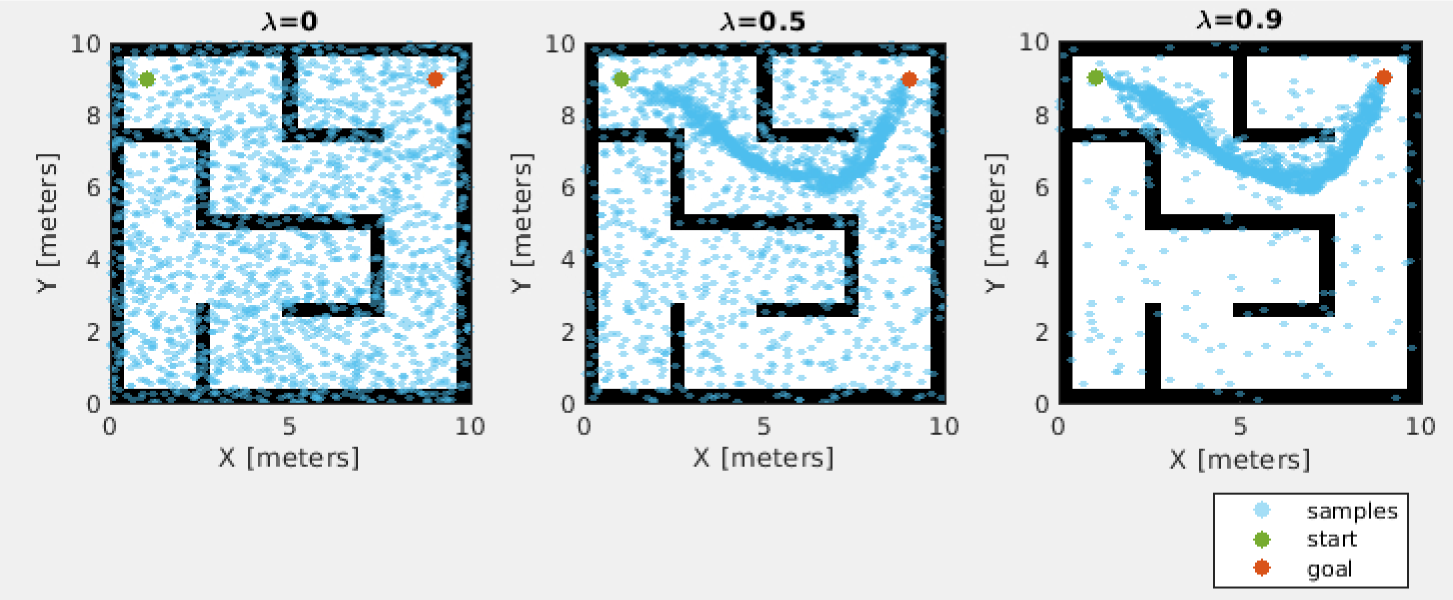
Latent state
Normally
Distributed
Random numbers
4X400

```
latentStates = dlarray(randn(latentStateSize, numSamplesPerSet), "CB");
```

More Examples on the Test Data

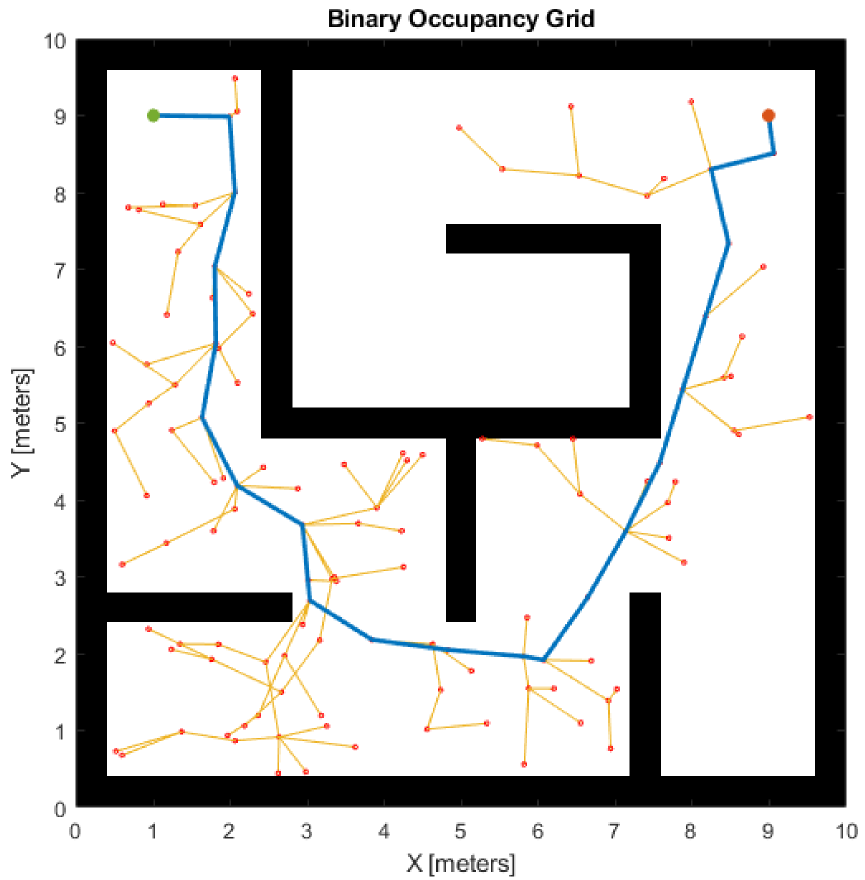


Choose the Learnable Sampling Factor

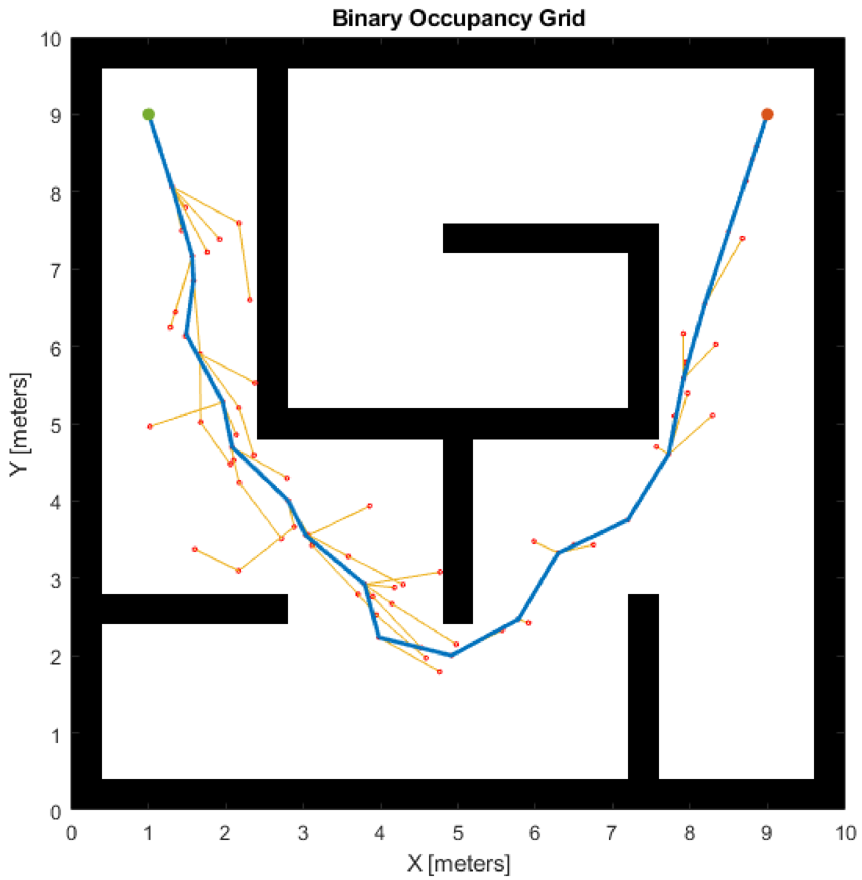


Mix both learned samples and uniform samples in a certain proportion λ, to bias the planner towards the optimal solution while also **guaranteeing to find a solution**

RRT* with Uniform & Learned Sampling



Uniform Sampling (λ = 0)



DL based Sampling (λ = 0.9)

90% from CVAE; 10% Uniform

Accelerate Motion Planning with Deep Learning

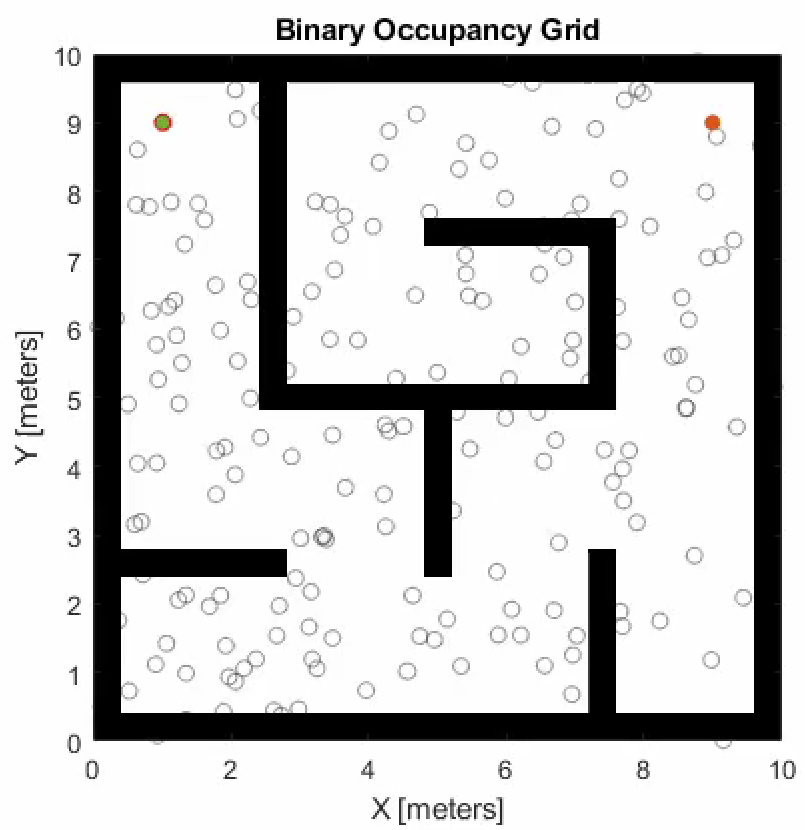
RRT* Path
Uniform Sampling (λ=0)

```
Elapsed time is 0.272630 seconds.
```

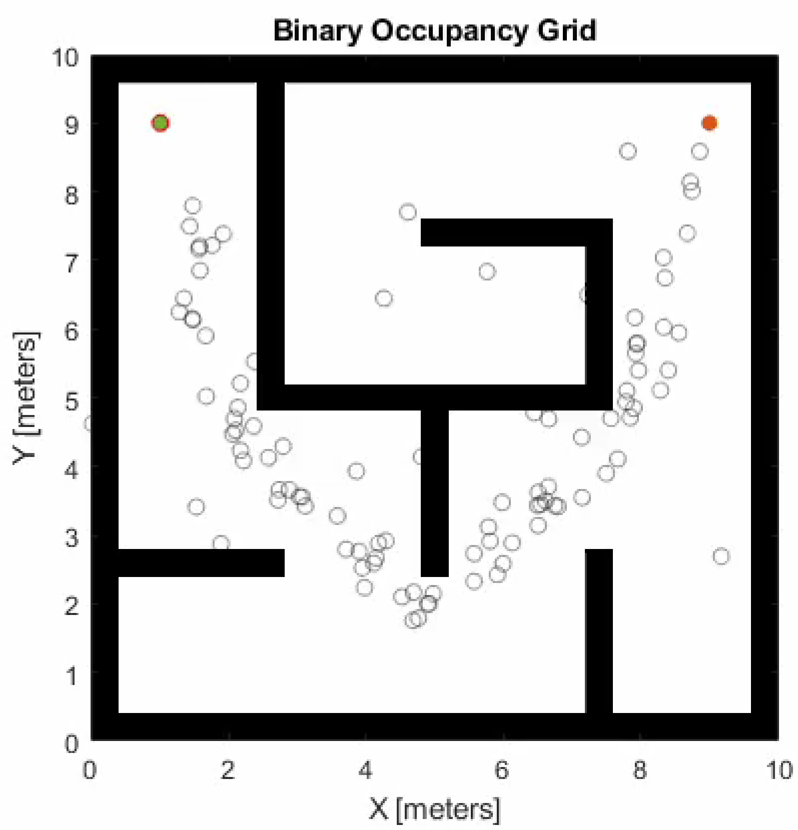
solnInfo

1x1 struct with 6 fields

Field	Value
IsPathFound	1
ExitFlag	1
NumNodes	122
NumIterations	182
TreeData	368x3 double
PathCosts	182x1 double



RRT* Path
DL Based Sampling (λ = 0.9)




```
Elapsed time is 0.094654 seconds.
```

solnInfo

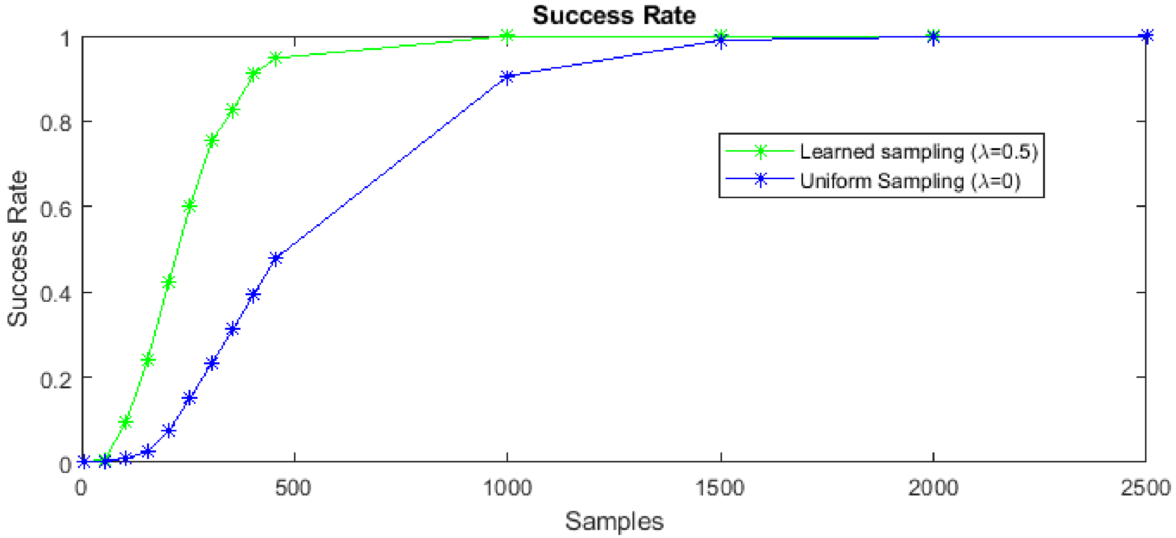
1x1 struct with 6 fields

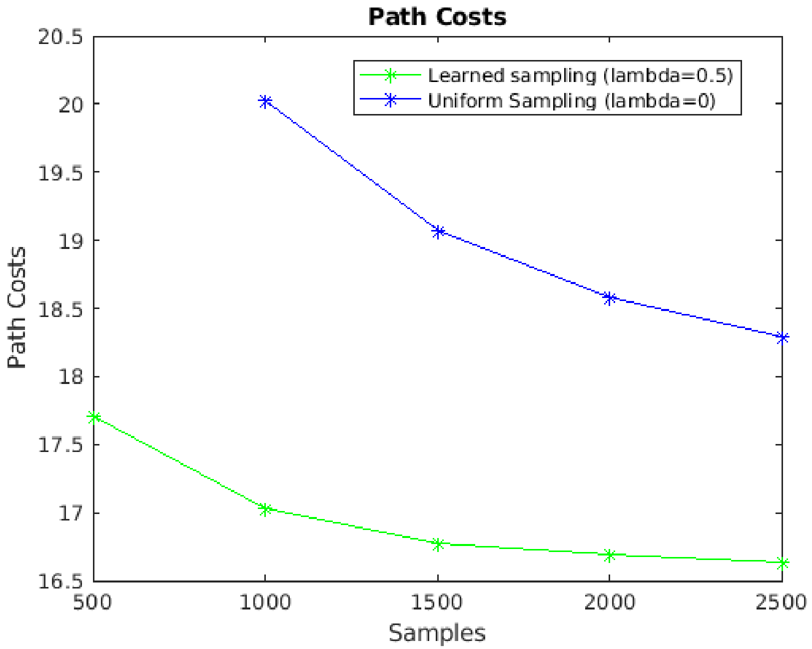
Field	Value
IsPathFound	1
ExitFlag	1
NumNodes	75
NumIterations	100
TreeData	227x3 double
PathCosts	100x1 double

2-3X Faster*
Less iterations
Less sampling states

* Animation runtime just reflects the ratio of treeData (as the animation was created during post-processing), whereas the elapsed time reflects the actual compute time

Accelerate Motion Planning with Deep Learning





- **Faster Convergence to finding a valid path with Deep Learning based sampling**
- For 500 samples, Uniform sampling can't find a path for each map & each run
- Learned sampling path **cost function much better** than uniform sampling

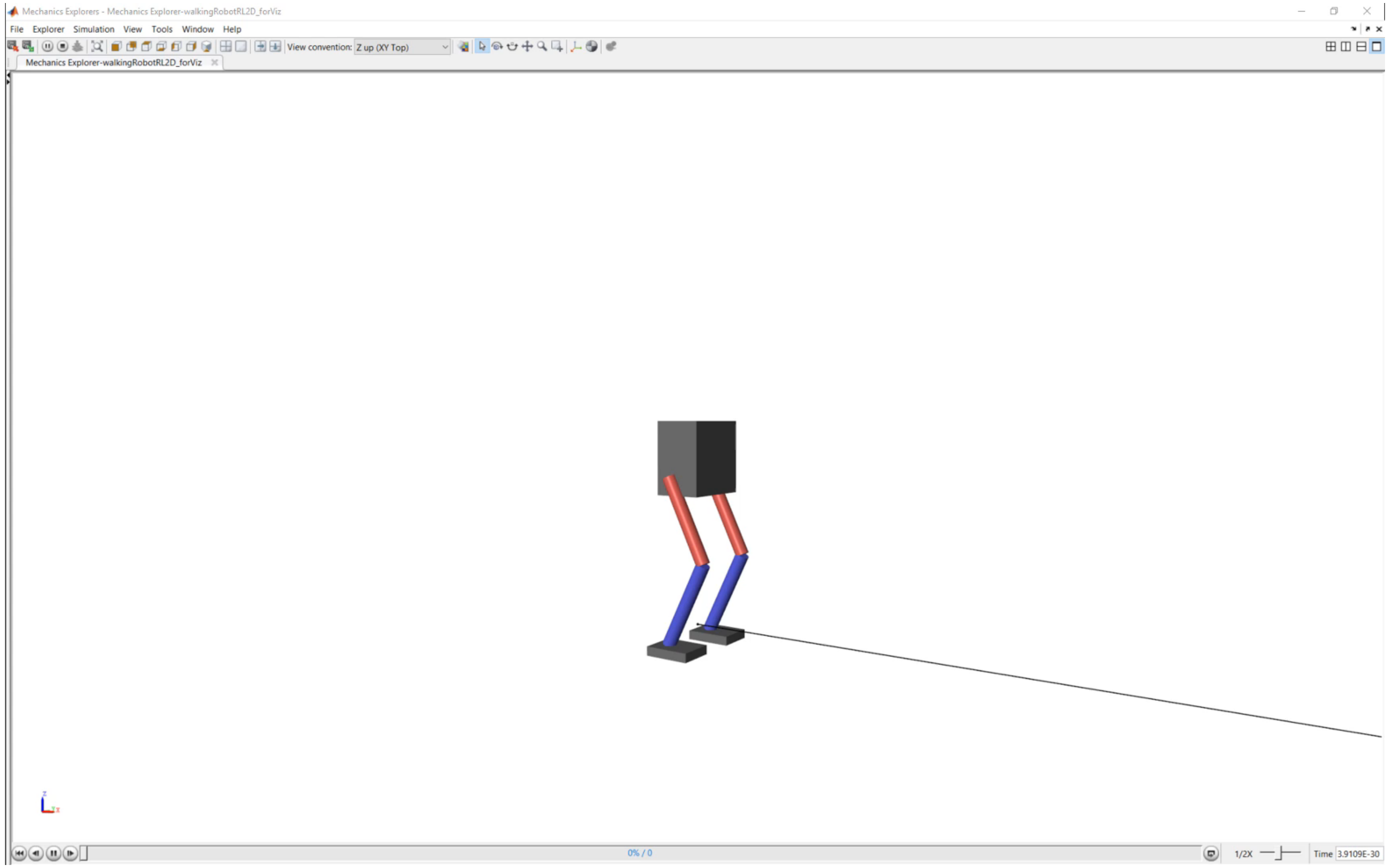
Agenda

- Introduction to Autonomous systems
- Artificial Intelligence
  - **Deep Learning: Acceleration of motion planning using deep learning**
- **Reinforcement Learning**
  - **Developing controller for automated parking valet**
- Deployment of AI models to embedded devices

What is Reinforcement Learning?

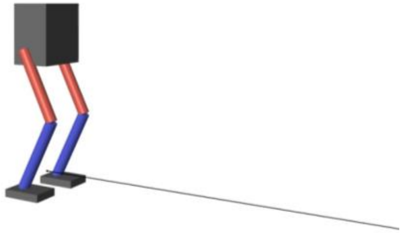
- What is Reinforcement Learning?
  - Type of machine learning that trains an ‘**agent**’ through repeated interactions with an environment

- How does it work?
  - Through a trial & error process that uses a reward system to maximize success

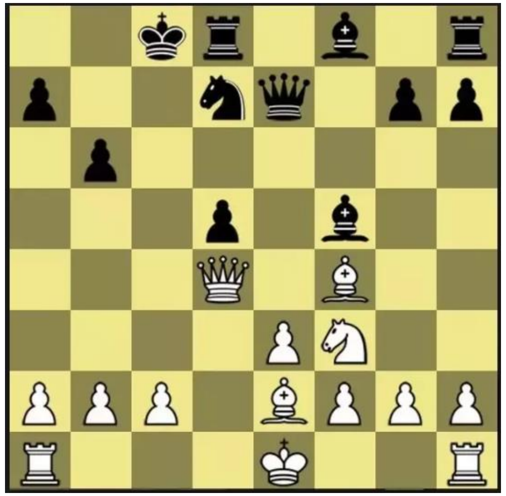


Reinforcement Learning enables the use of Deep Learning for Controls and Decision Making Applications

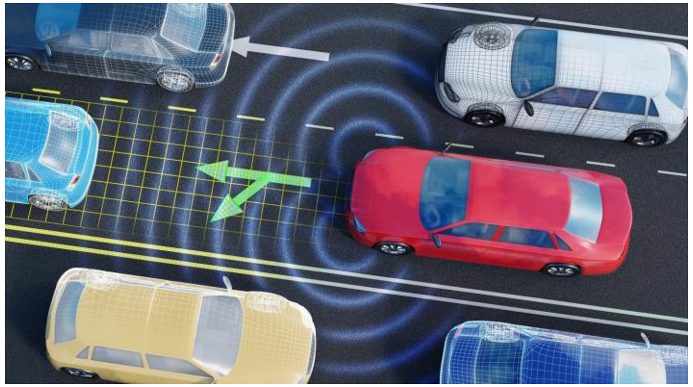
Controls



Robotics



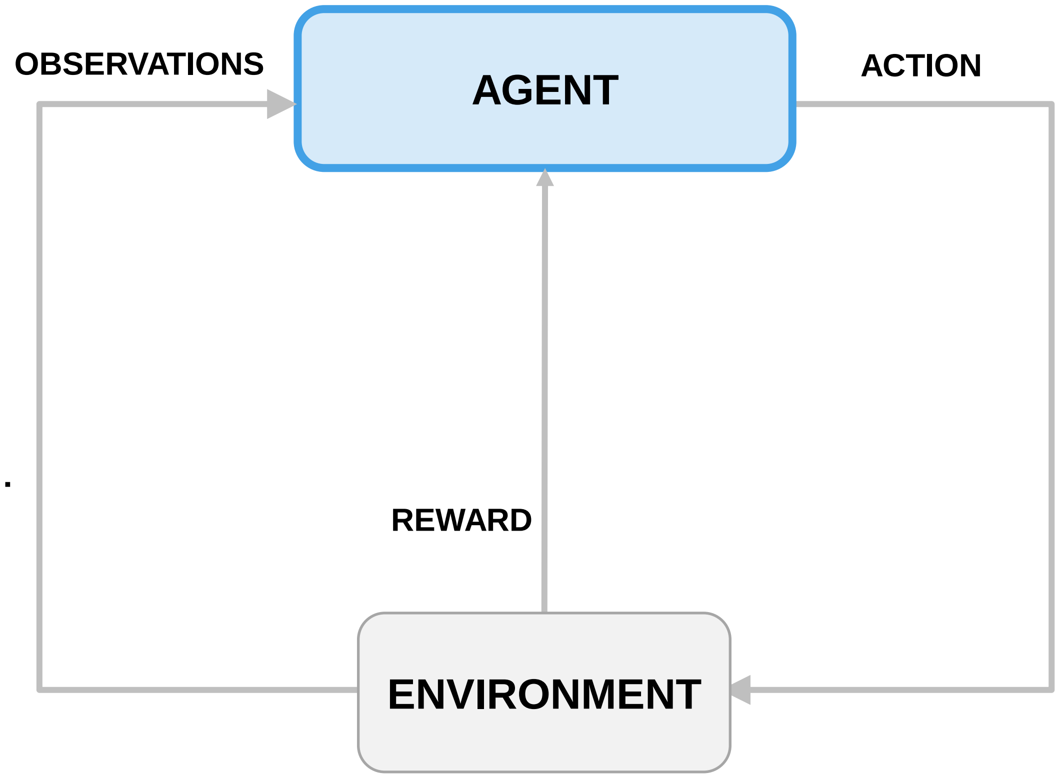
A.I. Gameplay



Autonomous driving

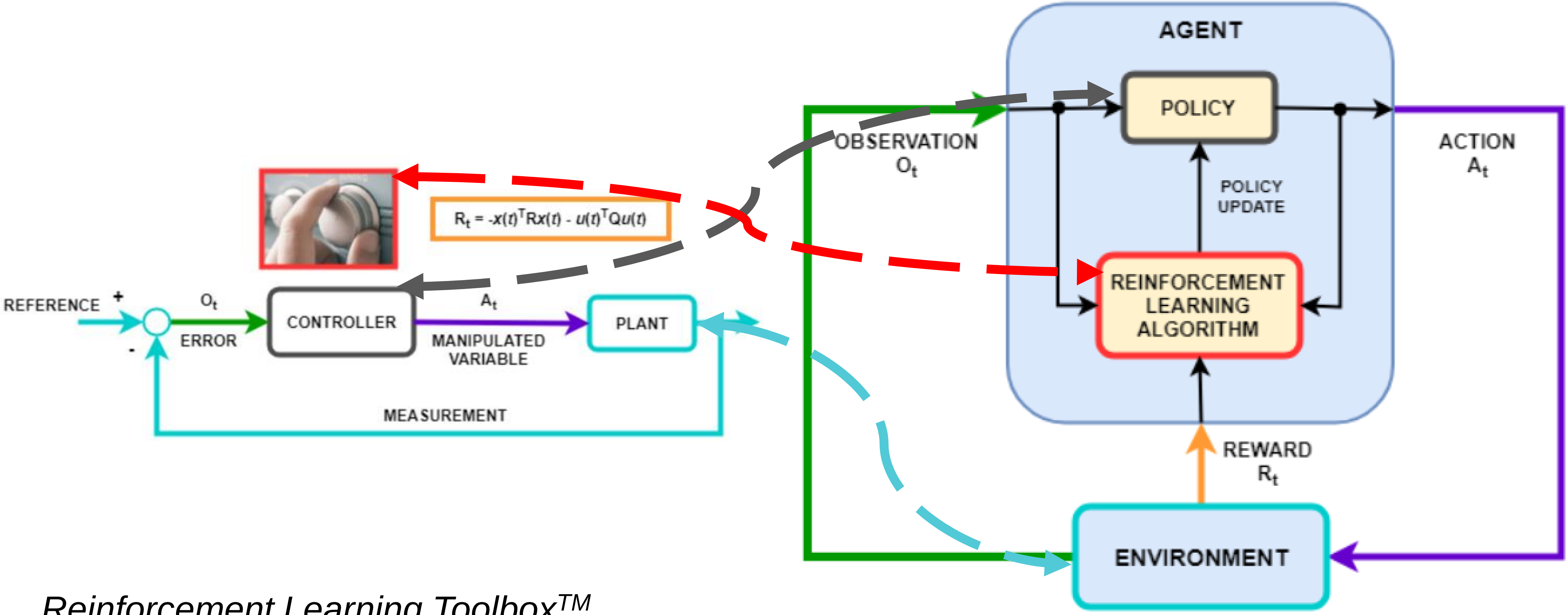
A Practical Example of Reinforcement Learning
Training an Automated Parking Valet Controller

- Vehicle's computer learns how to drive…
  (**agent**)
- using sensor readings from LIDAR, cameras,…
  (**observations**)
- that represent road conditions, vehicle position,…
  (**environment**)
- by generating steering, braking, throttle commands,…
  (**action**)
- to avoid collisions and lane deviation…
  (**reward**).



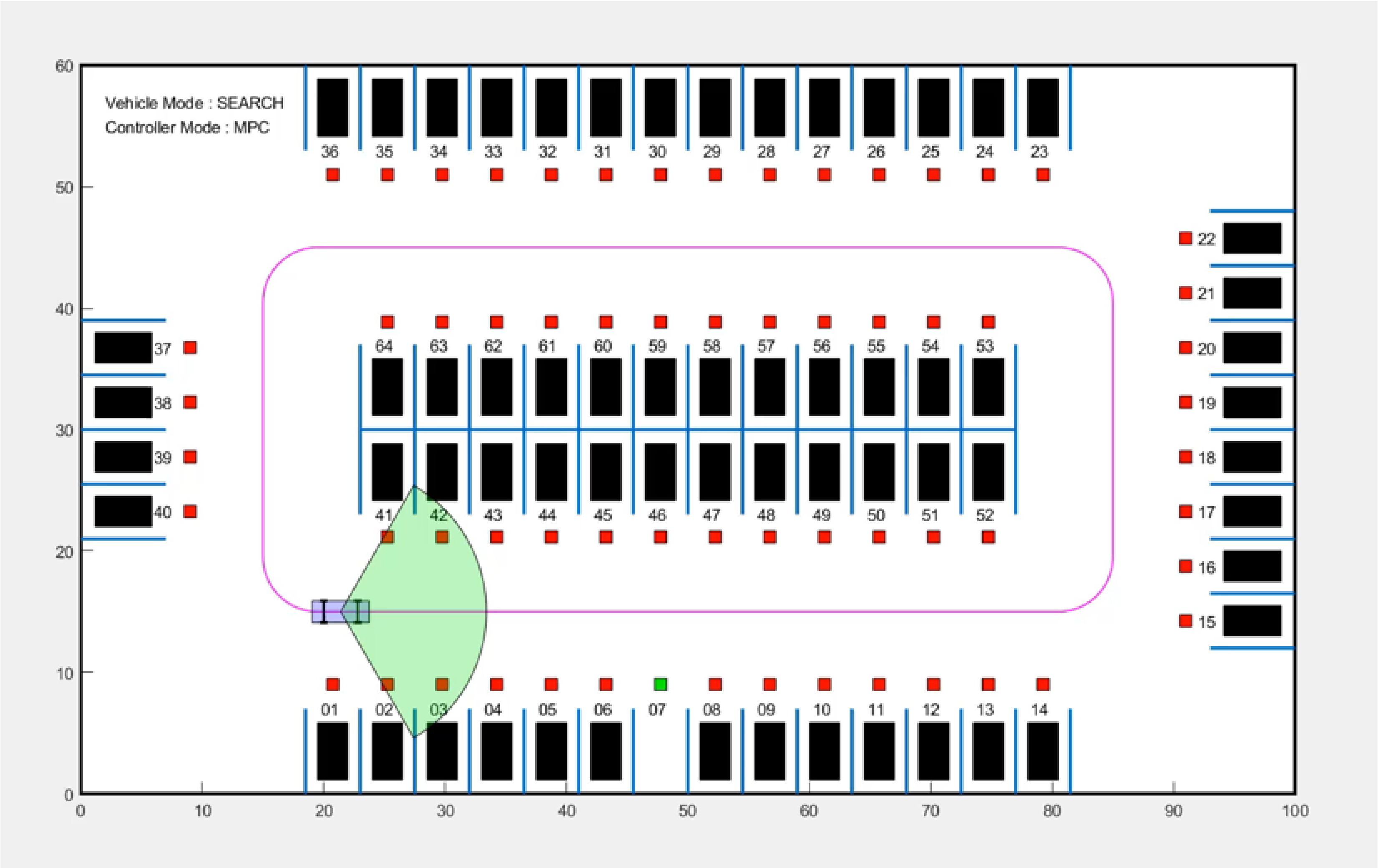
The goal of Reinforcement learning is for the agent to find an optimal algorithm for performing a task

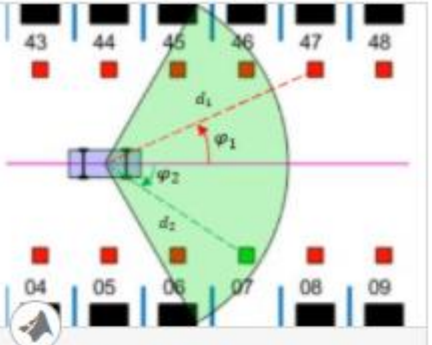
Drawing Parallels- RL and Controls



Reinforcement Learning Toolbox™

Simulate trained agent for automatic parking

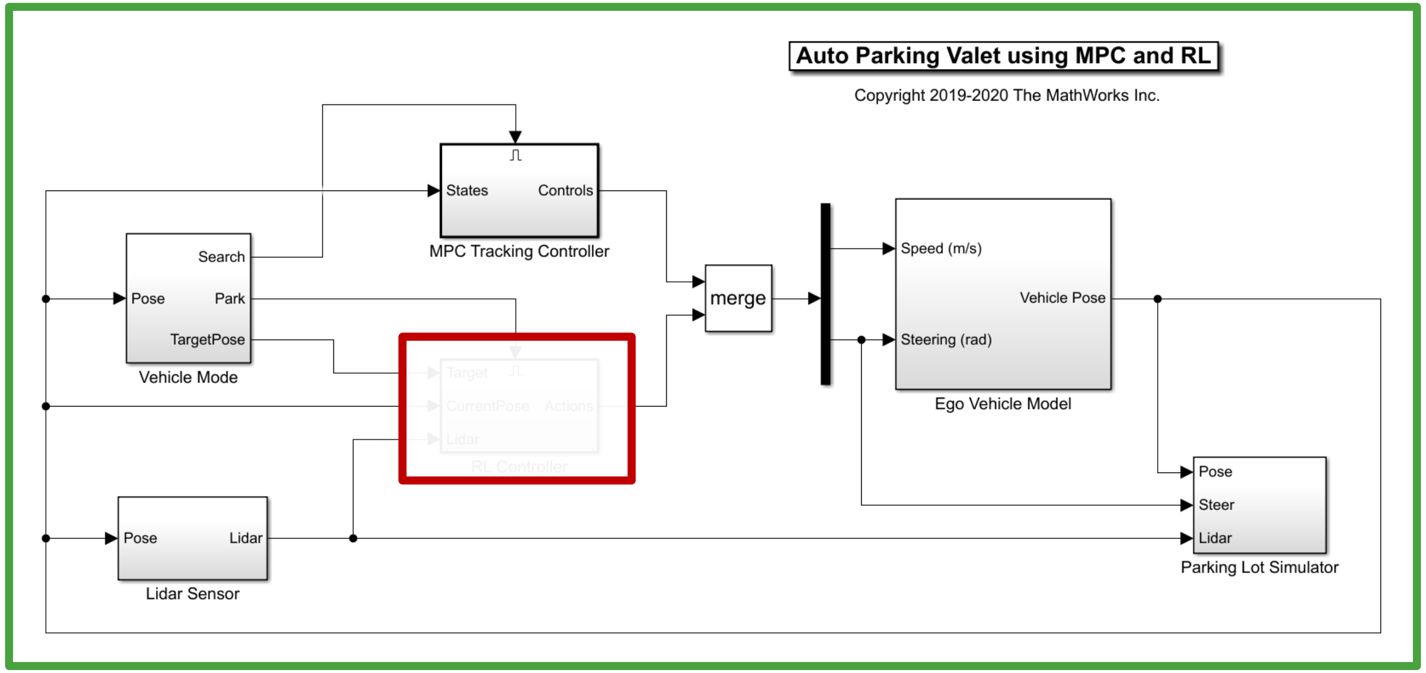


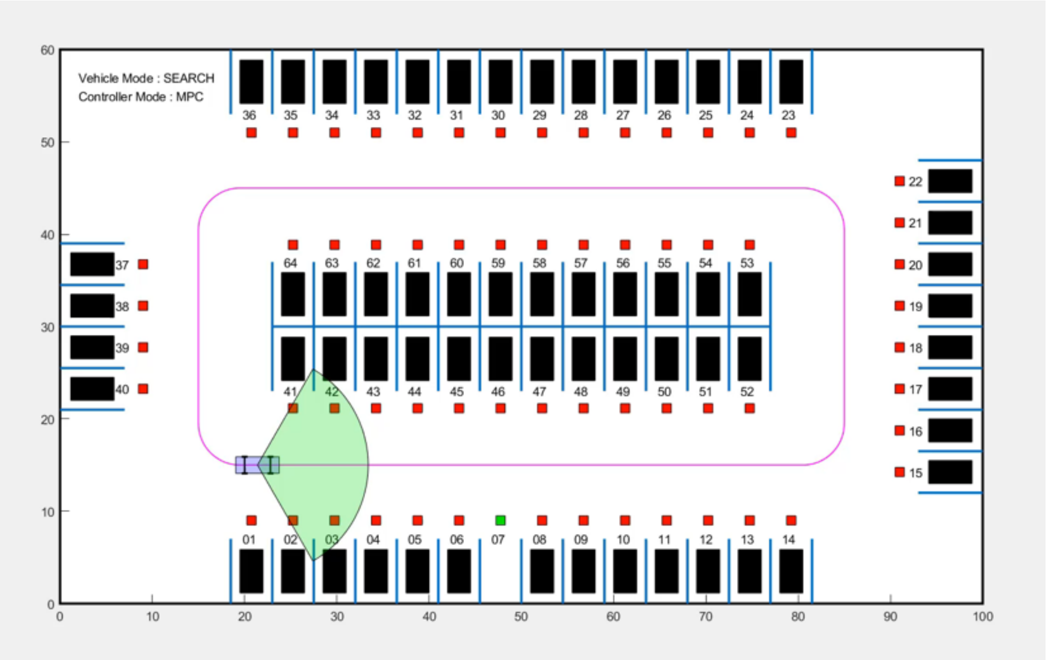


Train PPO Agent for Automatic Parking Valet

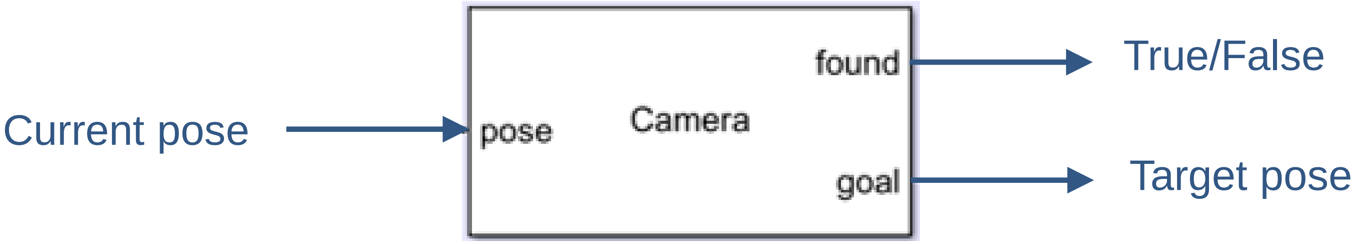
Train a reinforcement learning agent to park a car in an open parking space.

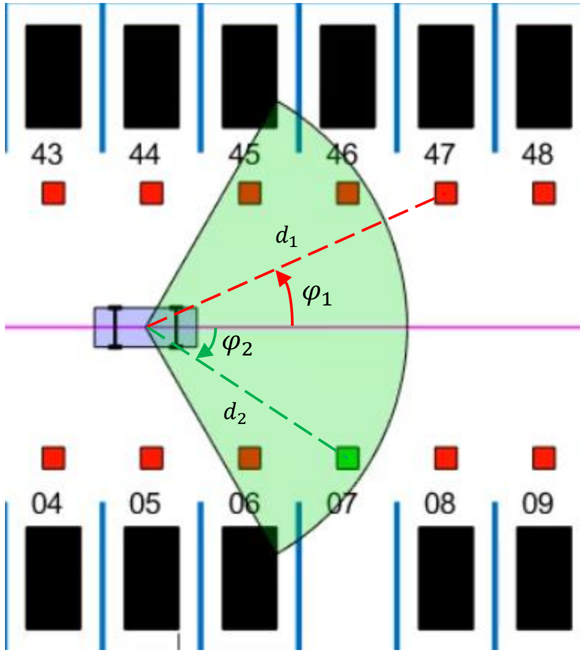
Simulink Model Bench for Parking Valet





Camera





Camera Parameters

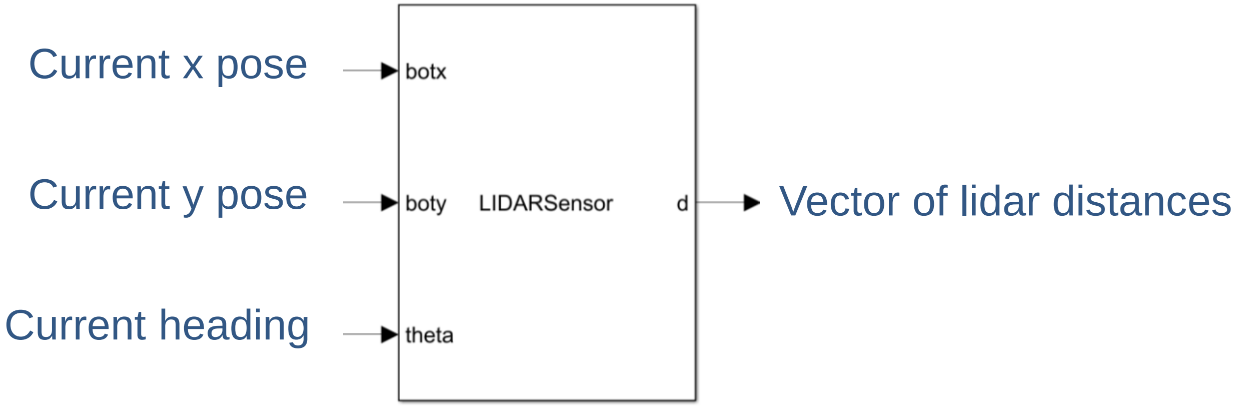
1. Depth (d_{max})
2. Field of view ($\varphi_{min}, \varphi_{max}$)

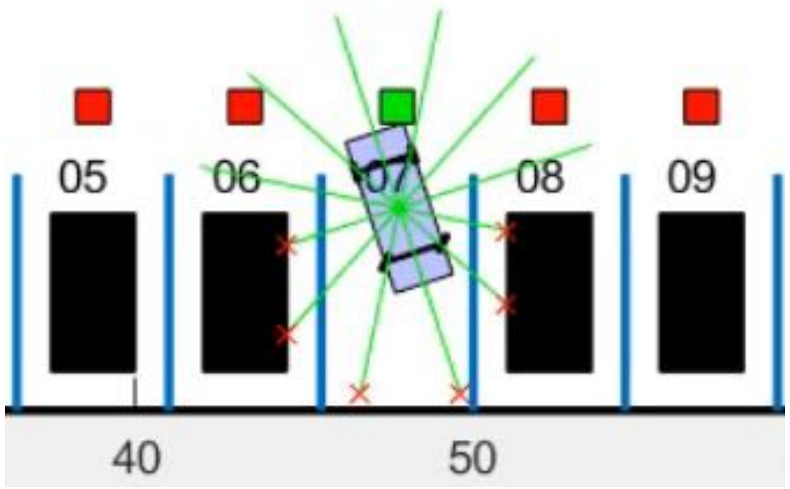
A spot in within range if

$$d_i \leq d_{max}$$

$$\varphi_{min} \leq \varphi_i \leq \varphi_{max}$$

Lidar

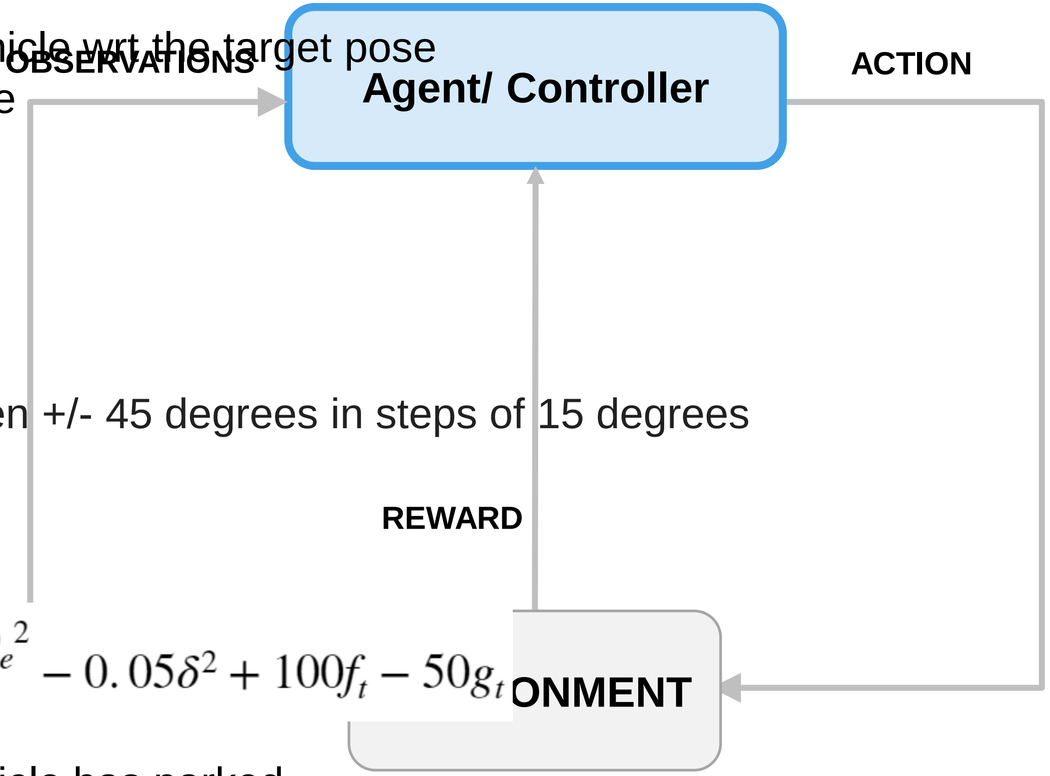




Lidar Parameters

1. Parking environment
2. No. of lidar readings
3. Maximum lidar distance
4. Geometry of the ego car
5. Geometry of obstacles

RL Controller



Observations:

- Position errors of the ego vehicle wrt the target pose
- True heading angle θ, and the
- Lidar sensor readings.

Actions:

- Constant Speed: 2 m/s
- Steering angle: range between +/- 45 degrees in steps of 15 degrees

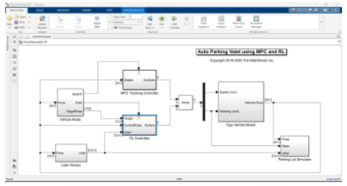
Reward:

$$r_t = 2e^{-\left(0.05X_e^{\ 2}+0.04Y_e^{\ 2}\right)} + 0.5e^{-40\theta_e^{\ 2}} - 0.05\delta^2 + 100f_t - 50g_t$$

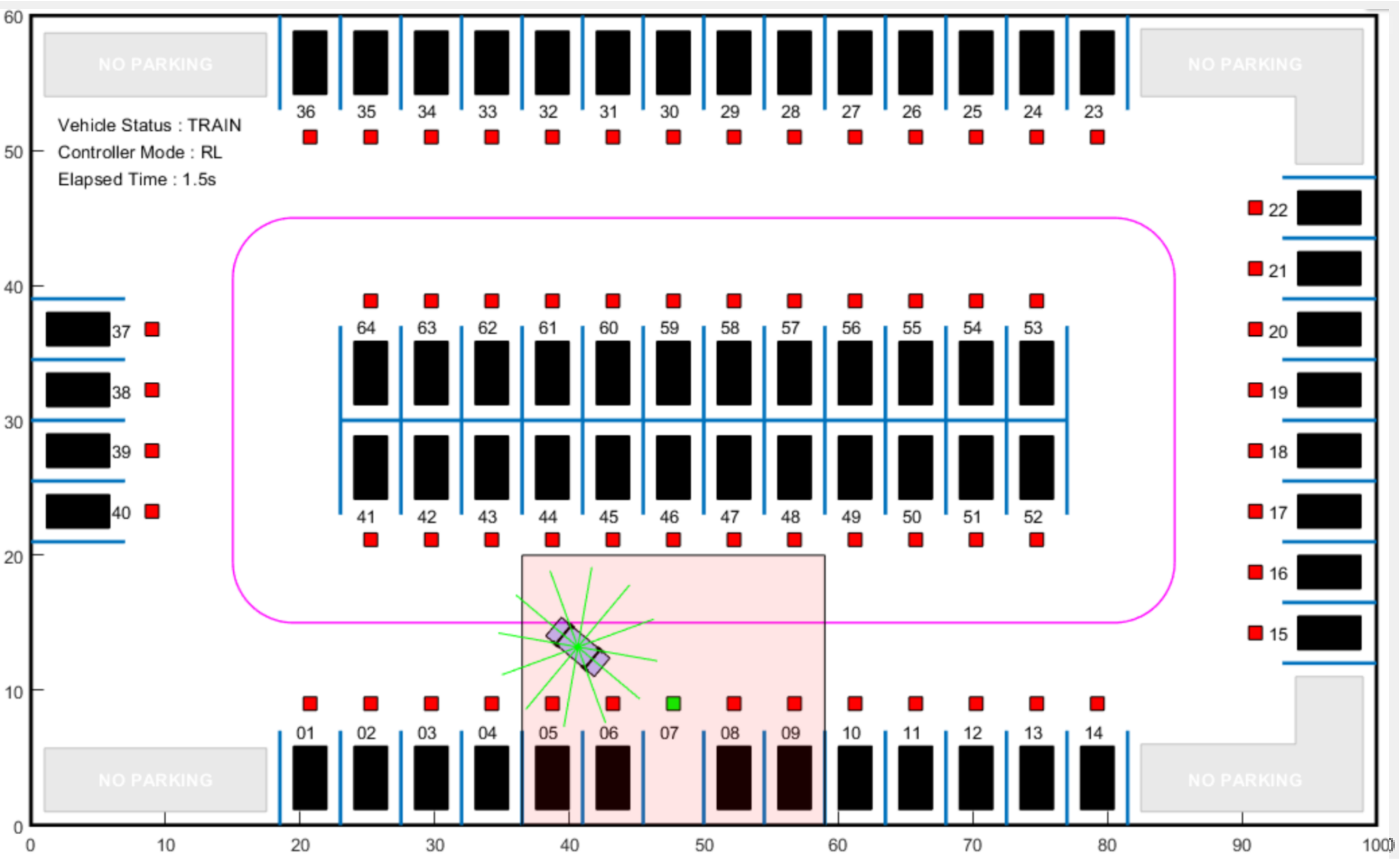
f_t (0 or 1) indicates whether the vehicle has parked

g_t Indicates collision

δ Steering angle



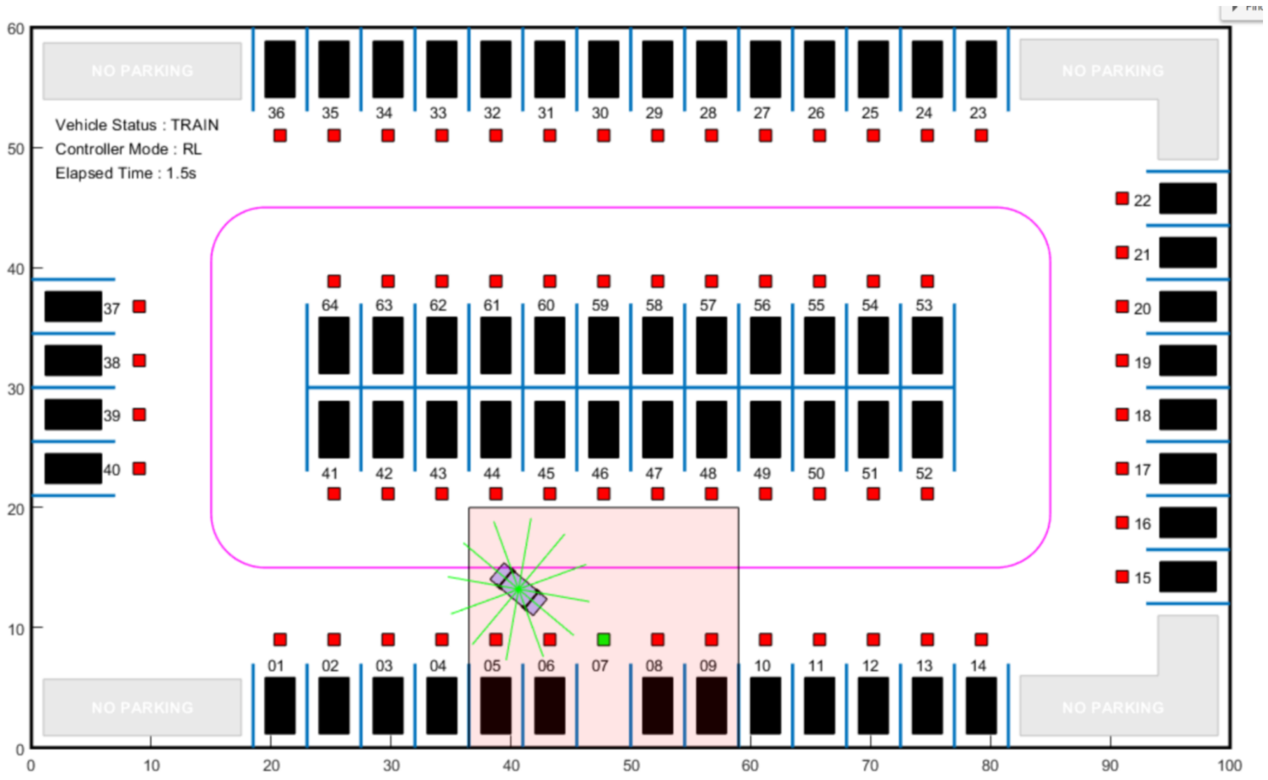
Environment



Mapping different parking locations

Observations for different parking spot locations could be coordinate transformations on vehicle pose

- 1-14: no transformation
- 15-22: $\overline{X} = Y, \overline{Y} = -X,\ \overline{\theta} = \theta - \pi/2$
- 23-36: $\overline{X} = 100 - X, \overline{Y} = 60 - Y,\ \overline{\theta} = \theta - \pi$
- 37-40: $\overline{X} = 60 - Y, \overline{Y} = X,\ \overline{\theta} = \theta - 3\pi/2$
- 41-52: $\overline{X} = 100 - X, \overline{Y} = 30 - Y,\ \overline{\theta} = \theta + \pi$
- 53-64: $\overline{X} = X, \overline{Y} = Y - 28,\ \overline{\theta} = \theta$



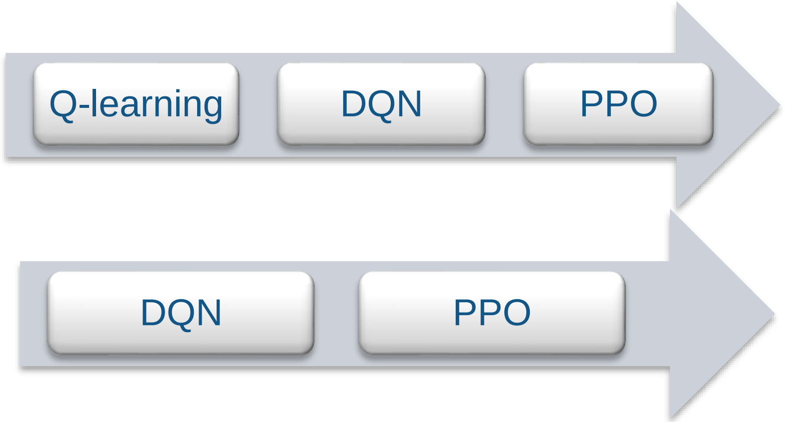
Choosing RL agent

Selection criteria:

1. Discrete or continuous spaces?
2. Complexity of algorithm
3. Algorithm-specific reasons

Discrete action space
Discrete observation space

Discrete action space
Continuous observation space



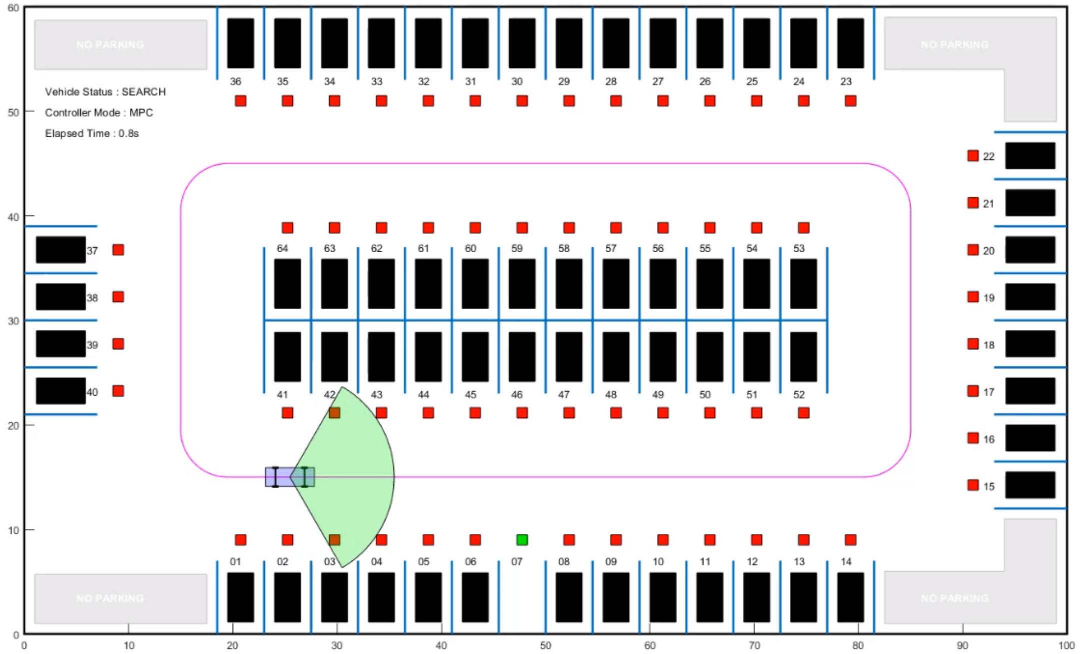
- PPO has more **stable** updates but requires **more training**
- TD3 is an **improved**, more **complex** version of DDPG
- SAC is an **improved**, more **complex** version of DDPG that generates **stochastic** policies

Built-in Reinforcement Learning Agents

Agents	Type	Observation Space	Action Space
Deep Q-Networks (DQN)	Value-based	Continuous/Discrete	Discrete
Q Learning	Value-based	Continuous/Discrete	Discrete
SARSA	Value-based	Continuous/Discrete	Discrete
Policy Gradient (REINFORCE)	Policy-based	Continuous/Discrete	Continuous/Discrete
Deep Deterministic Policy Gradient (DDPG)	Actor critic	Continuous/Discrete	Continuous
Actor Critic (A2C & A3C as well)	Actor critic	Continuous/Discrete	Continuous/Discrete
Proximal Policy Optimization (PPO)	Actor critic	Continuous/Discrete	Continuous/Discrete
Twin Delayed Deep Deterministic Policy Gradient (TD3)	Actor critic	Continuous/Discrete	Continuous
Soft Actor Critic (SAC)	Actor critic	Continuous/Discrete	Continuous

Tables can only be used with **discrete** observations and actions

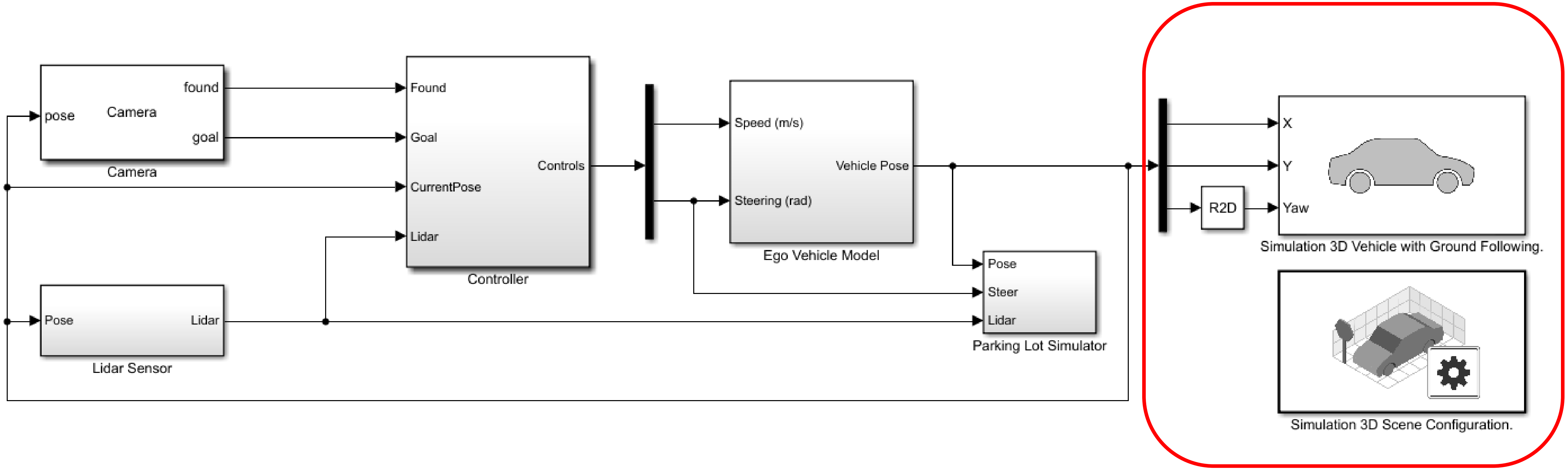
Scaling environment- Unreal Cosimulation



Train PPO Agent for Automatic Parking Valet Original Example

Unreal Engine – Large Parking Lot Scene

Incorporating 3D Simulation

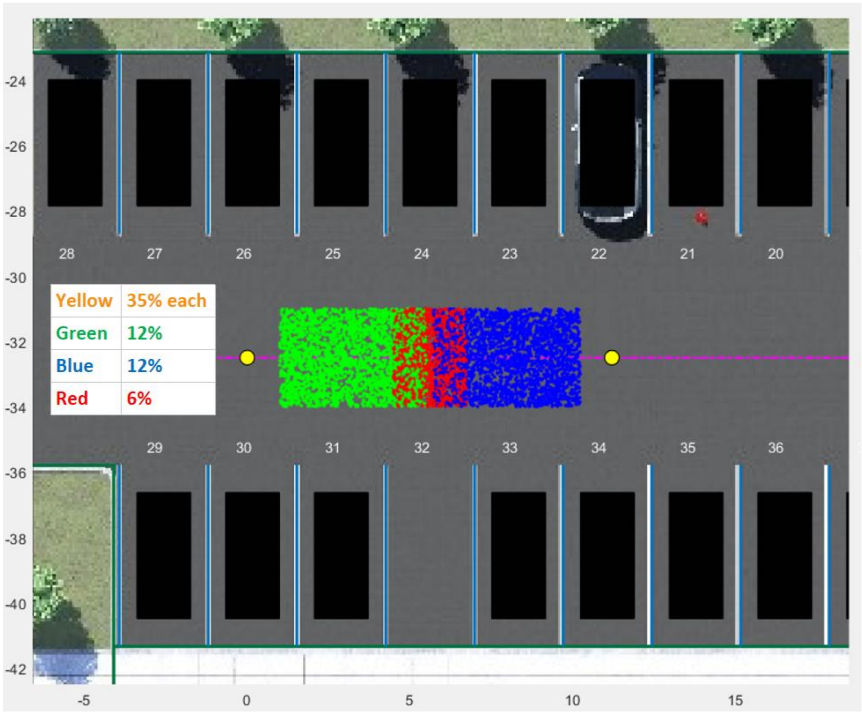


Reinforcement Learning Environment

$$r_t = 2e^{-\left(0.05X_e^{\,2}+0.04Y_e^{\,2}\right)} + 0.5e^{-40\theta_e^{\,2}} - 0.05\delta^2 + 100f_t - 50g_t$$

Reward Function

Training Environment

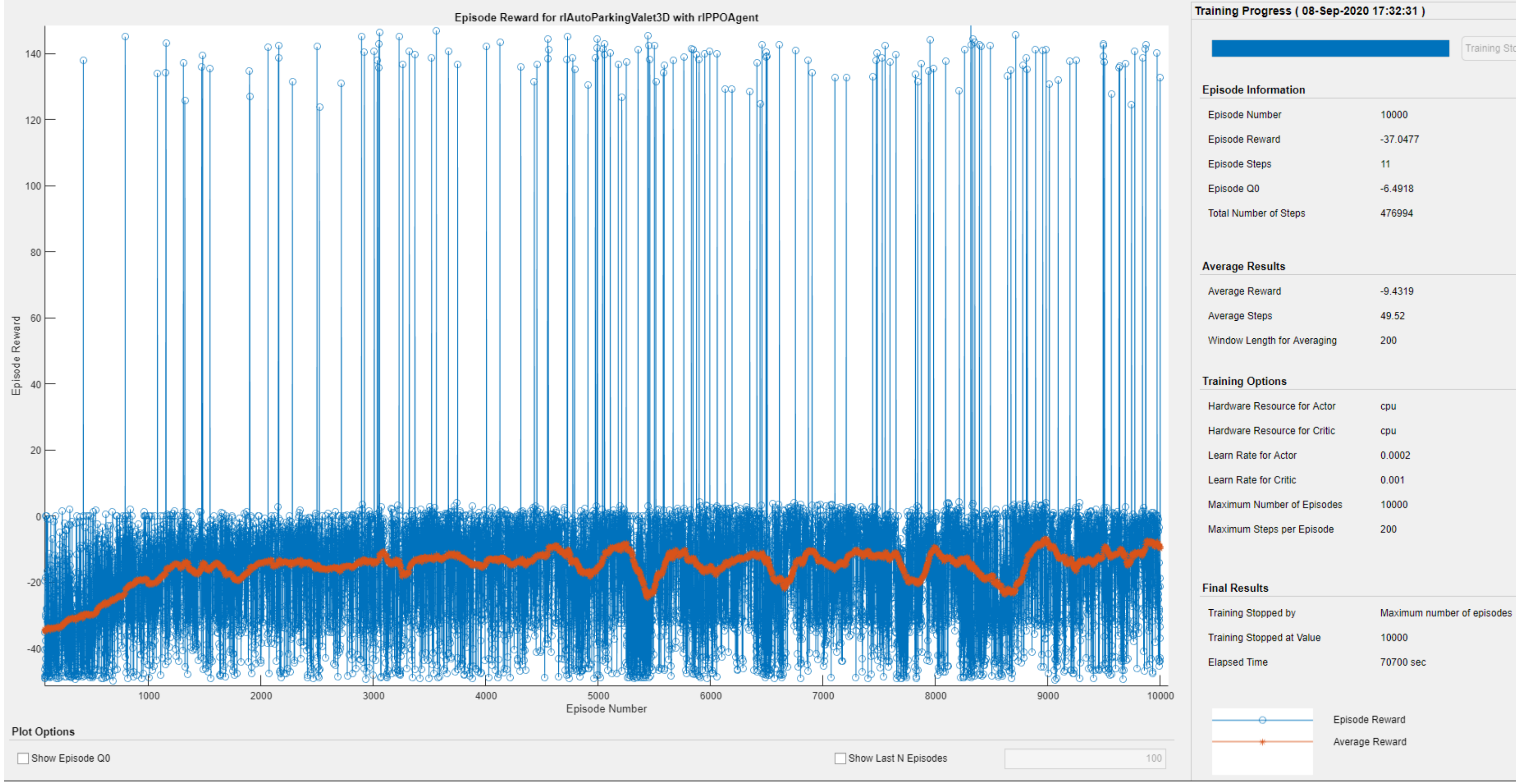


Vehicle Initial Pose during Training

29-37	NA	NA	NA
15-28	$\bar{X} = 41 - X$	$\bar{Y} = -64.485 - Y$	$\bar{\theta} = \theta - \pi$
1-14	$\bar{X} = X$	$\bar{Y} = Y + 20.41$	$\bar{\theta} = \theta$
38-46	$\bar{X} = 41 - X$	$\bar{Y} = -84.48 - Y$	$\bar{\theta} = \theta - \pi$

Coordinate Transformations on Vehicle Pose

Training the Agent



Training from Scratch, Hyperparameters from Original Example

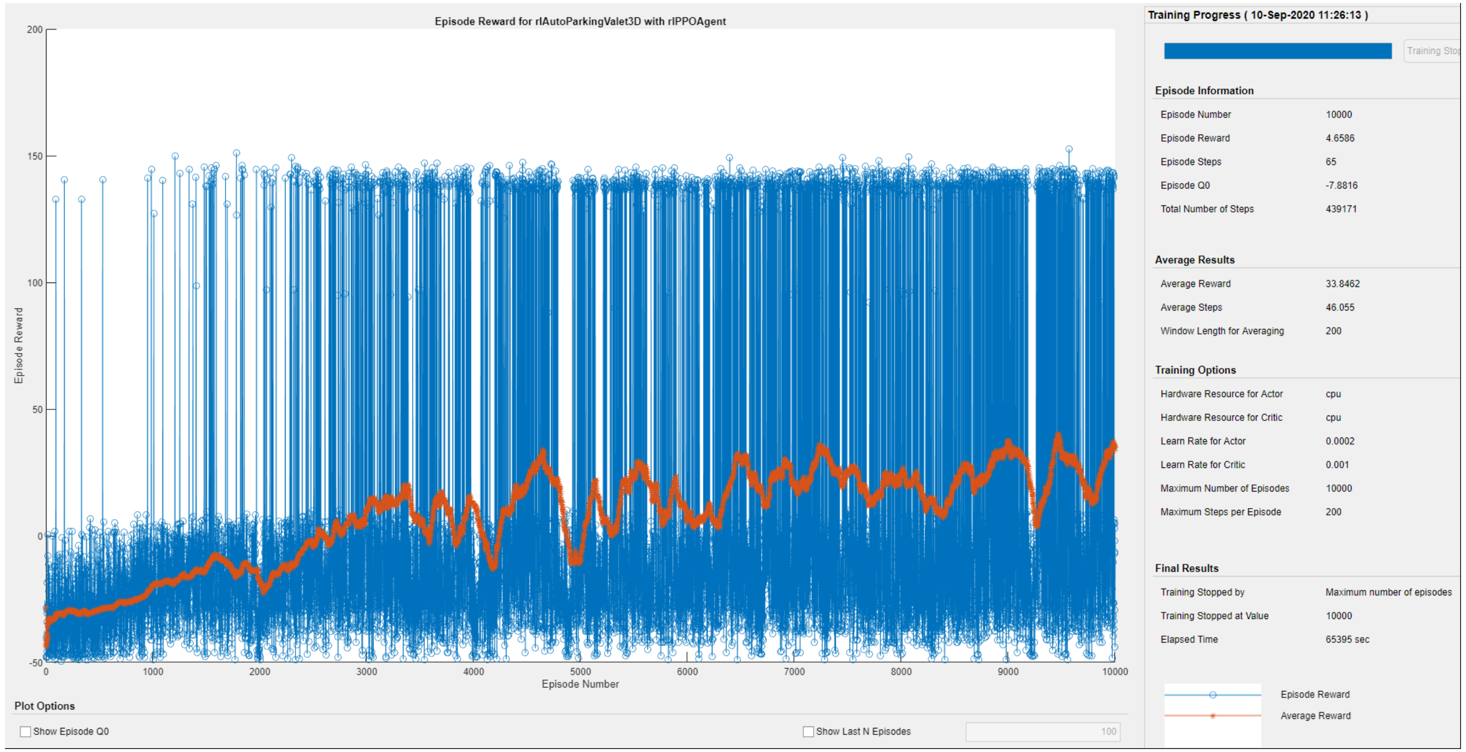
PPO Agent Options

SampleTime	Ts
ExperienceHorizon	200
ClipFactor	0.2
EntropyLossWeight	0.01
MiniBatchSize	64
NumEpoch	3
AdvantageEstimateMethod	gae
GAEFactor	0.95
DiscountFactor	0.998

Training Options

MaxEpisodes	10000
MaxStepsPerEpisode	200
ScoreAveragingWindowLength	200
Plots	training-progress
StopTrainingCriteria	AverageReward
StopTrainingValue	inf

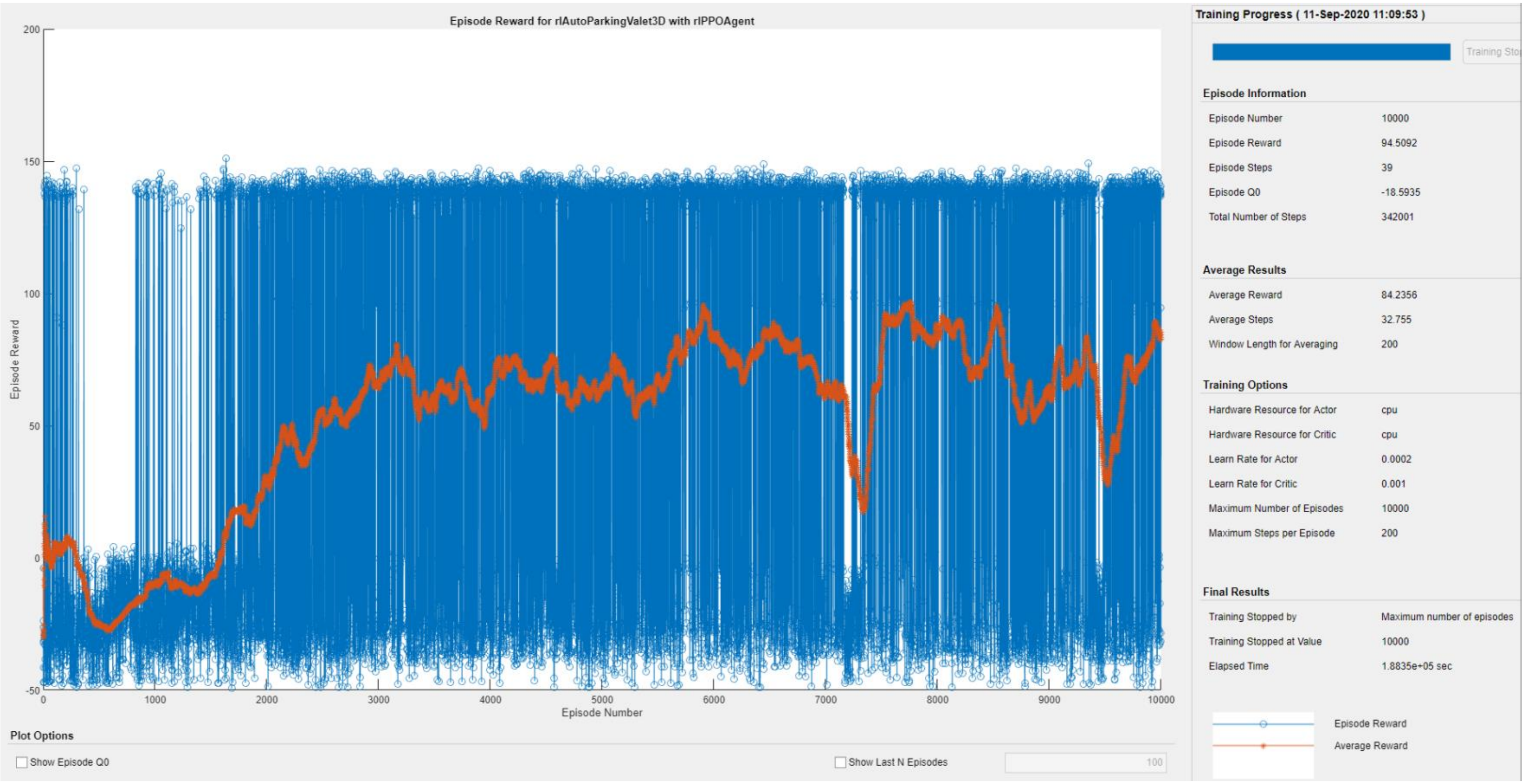
Training the Agent cont.



Training from Scratch, Changed Parked Vehicle Dimensions

Testing the Agent

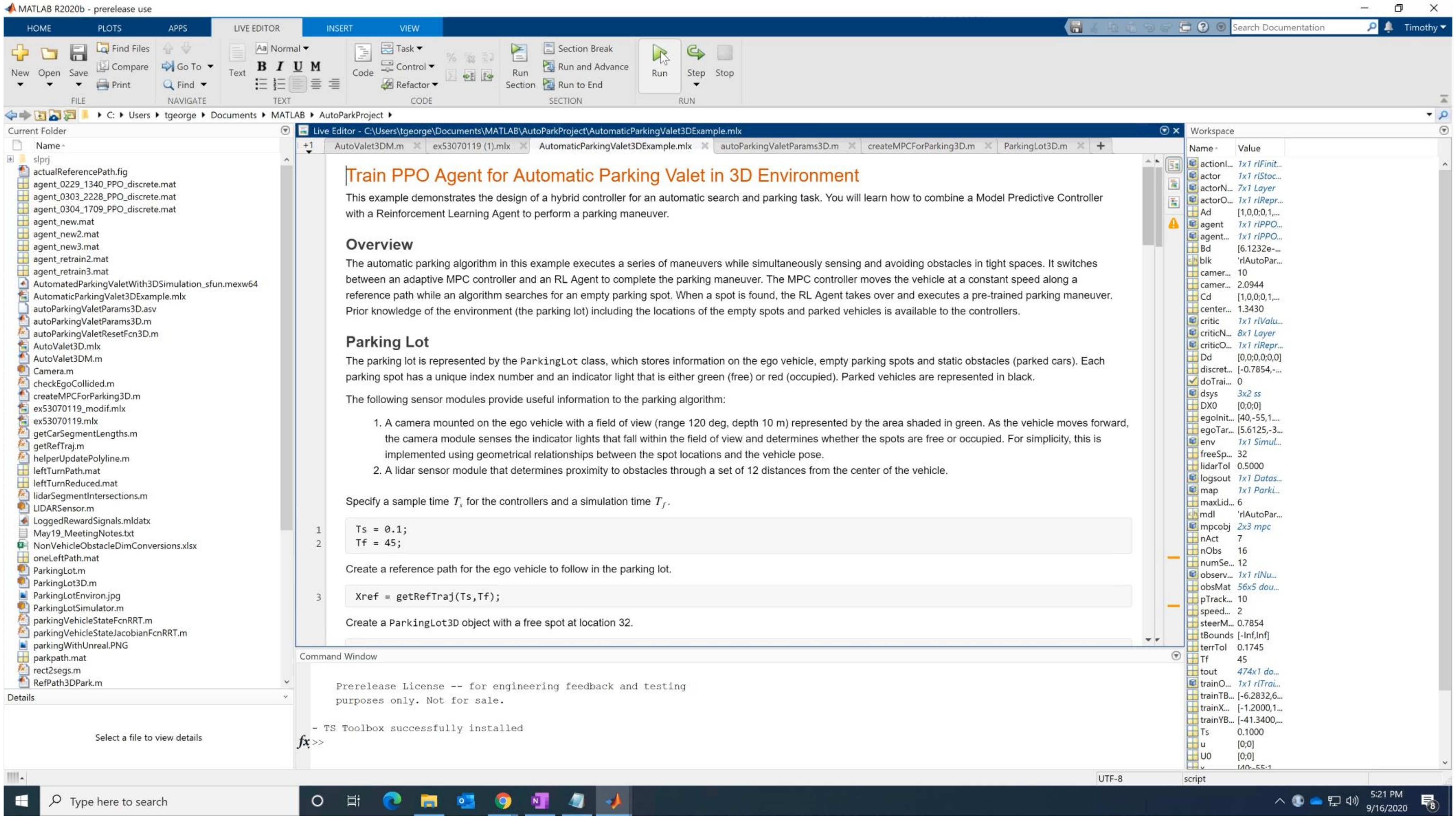
Final Agent Training



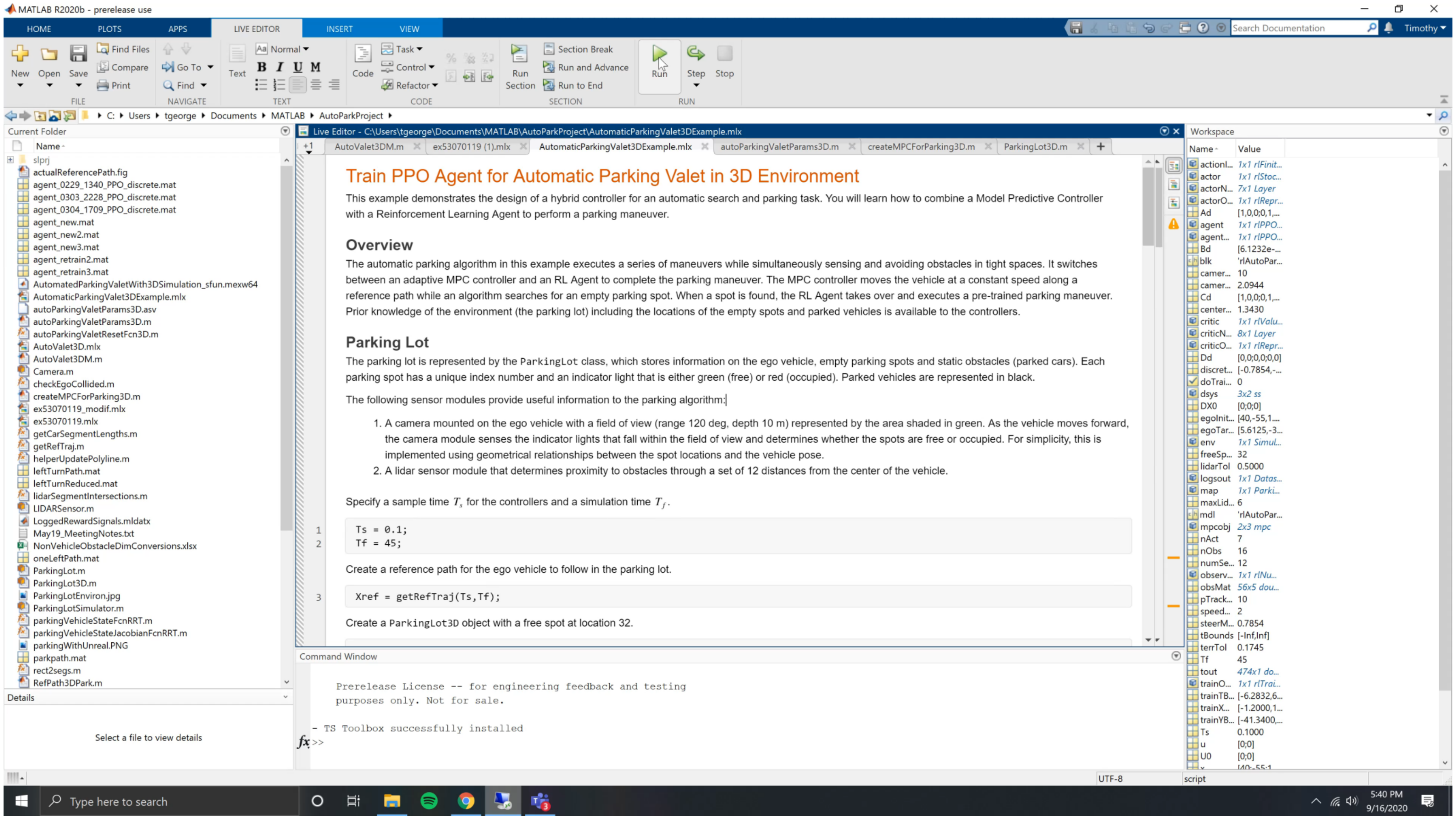
Retraining Agent from Original Example

- Retrained agent from original example
- Again used hatchback dimensions for parked cars
- Highest average reward
- Appeared to park successfully during test

Final Example Demo 1

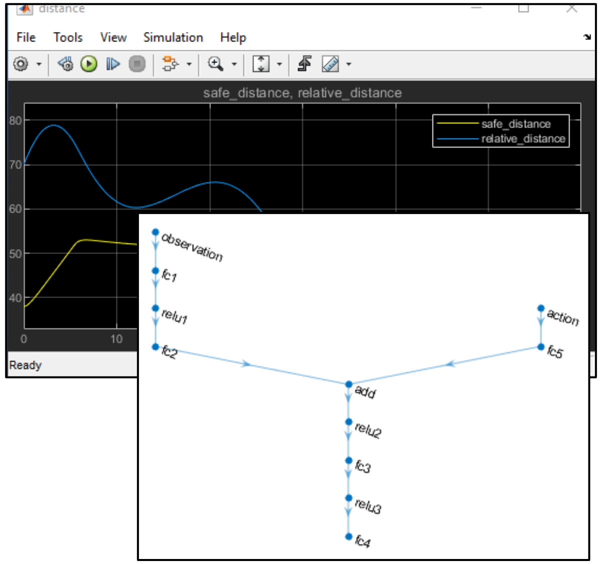


Final Example Demo 2



Design reinforcement learning agents for controls

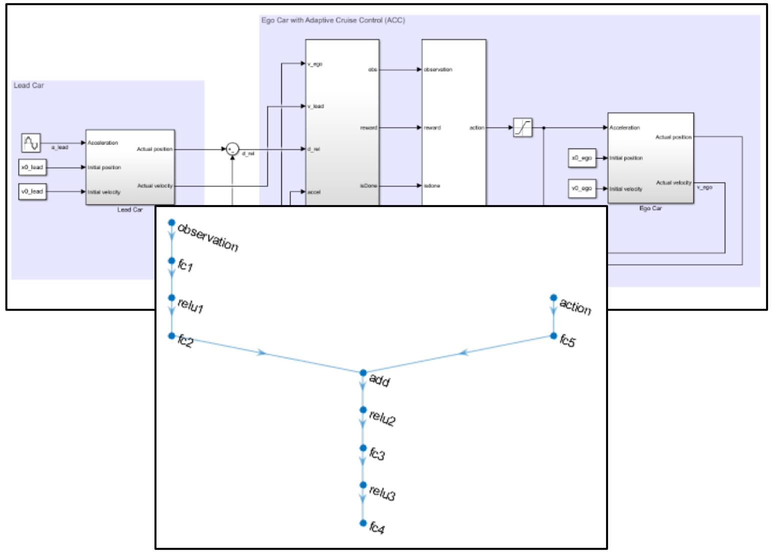
DDPG Agent



[Train Deep Deterministic Policy Gradient (DDPG) Agent for Adaptive Cruise Control]()

Reinforcement Learning Toolbox™

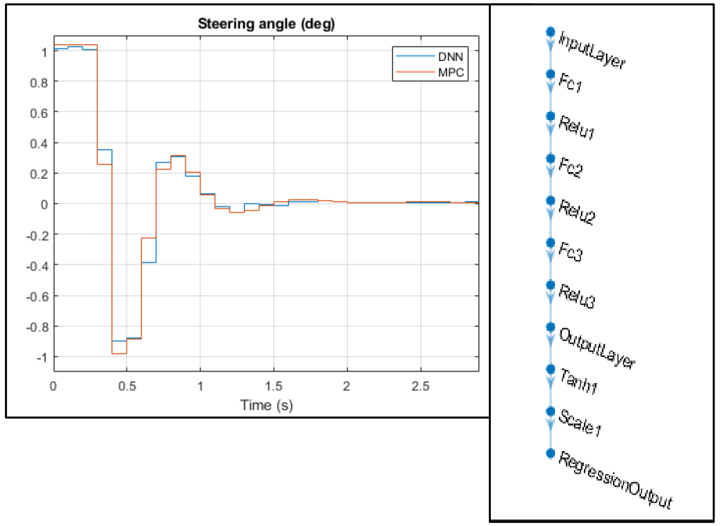
DDPG Agent



[Train DDPG Agent for Path Following Control]()

Reinforcement Learning Toolbox™

Neural Network

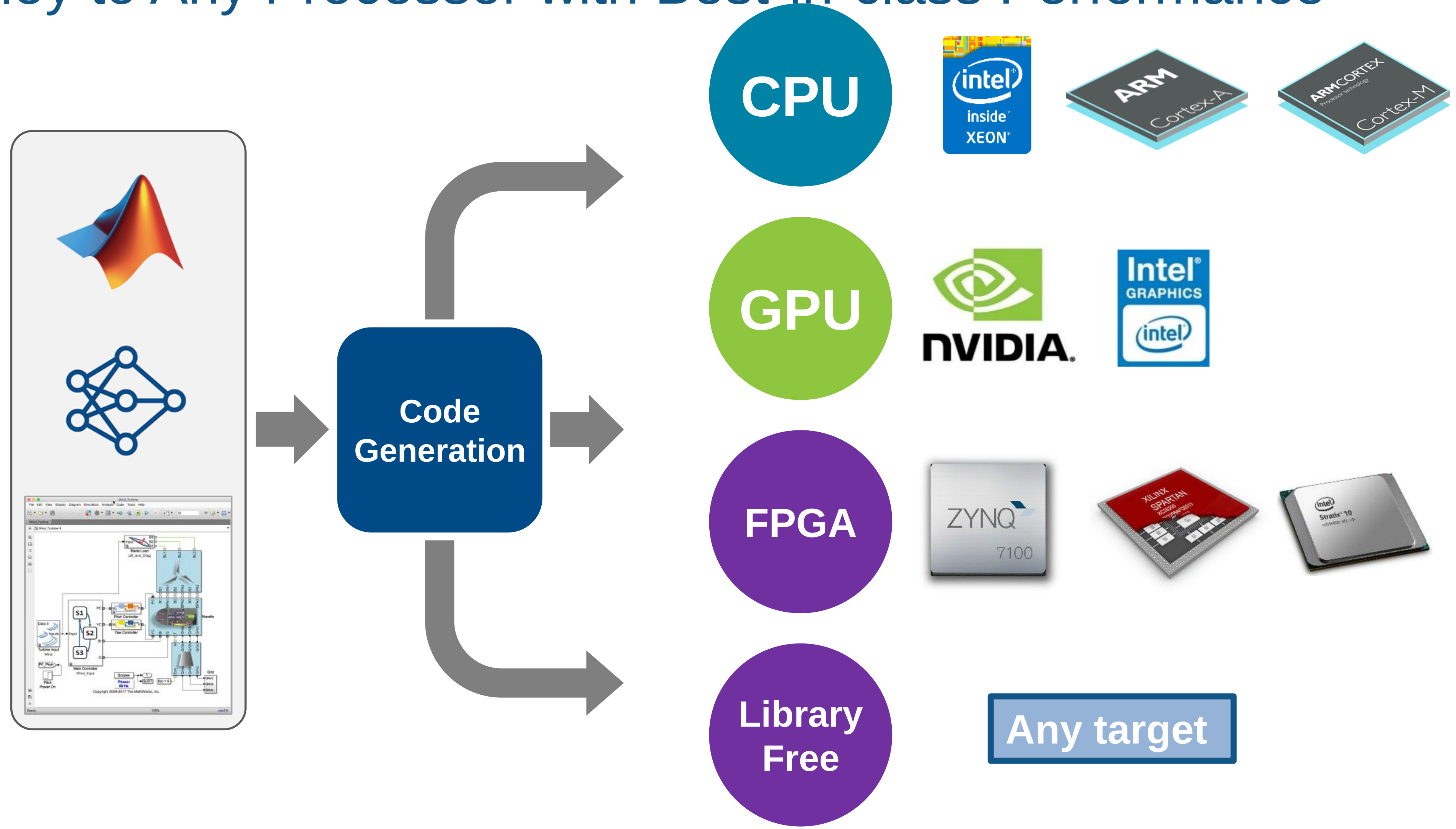


[Imitate MPC Controller for Lane Keep Assist using a Neural Network]()

Reinforcement Learning Toolbox™

Model Predictive Control Toolbox™

Deploy to Any Processor with Best-in-class Performance



AI models in MATLAB and Simulink can be deployed on embedded devices, edge devices, enterprise systems, the cloud, or the desktop

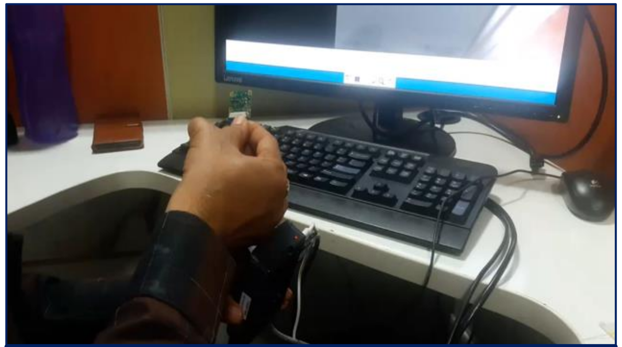
AI deployed on Embedded Devices

- Need code that takes advantage of:
  - NVIDIA® CUDA libraries, including cuDNN and TensorRT
  - Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN) for Intel processors
  - ARM® Compute library for ARM processors

NVIDIA Jetson TX1 board

Android Phone

Intel Xeon Desktop PC



Raspberry Pi Board

Technology Showcase Demo Booths

	Demo Booth Title	Demo Description
Robotics and Autonomous systems	**Virtual World and Algorithm Development for Automated Driving**	• Create virtual world (scene/scenarios) from specifications and recorded data • Interoperate with ASAM standards and build road networks from HD map services • Develop algorithms for perception, sensor fusion, planning and control systems • Test algorithms with a virtual testing environment
	HIL Testing for an ADAS ECU in a Virtual Environment	• Establish interfaces for ECU under test • Generate code and deploy subcomponents on HIL machines • Address synchronization in a closed loop setup with multiple machines
	Service-Oriented Architectures (SOA) for Designing and Deployment in Automated Driving	• Architect services for adaptive cruise control using System Composer • Design and simulate algorithm behavior for vehicle actuation (brakes, acceleration) • Use automatic code generation and deployment as service (Adaptive AUTOSAR, ROS 2, DDS, etc.)
Artificial Intelligence	**Surrogate AI Models for CAE Applications**	• Build a design of experiments (DOE) table for component design • Create surrogate AI models from FEA/CFD simulations • Run multiobjective design optimization studies using AI models
	Enabling Industry 4.0	• Secure data exchange with smart industrial plant sensors and servers • Develop predictive maintenance, smart manufacturing, and SCADA applications
	Hardware Deployment of AI Models	• Low-code aspect of AI workflow • Scope of hardware selection within the auto-code generation workflow • AI models to target hardware deployment

MATLAB EXPO

Thank you

Dr Rishu Gupta, MathWorks

Peeyush Pankaj, MathWorks

MathWorks®