

MATLAB EXPO

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

Dr Rishu Gupta, MathWorks



Peeyush Pankaj, MathWorks



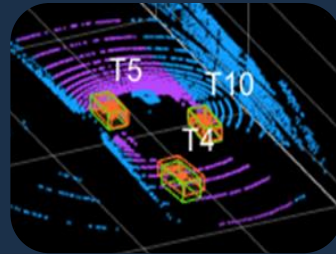
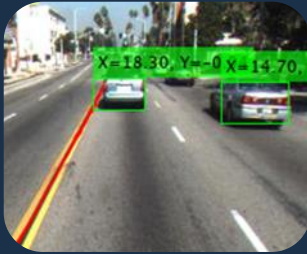
Agenda

- 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

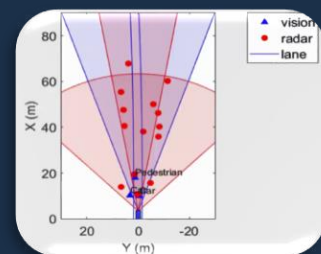
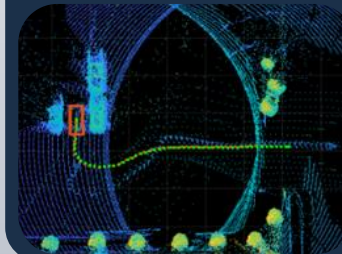
Key subsystems of an autonomous systems

Algorithms

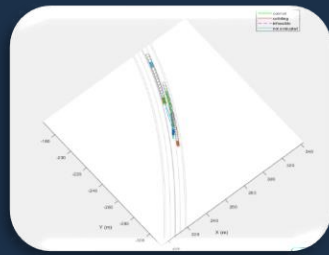
Perception



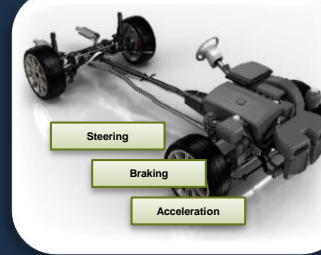
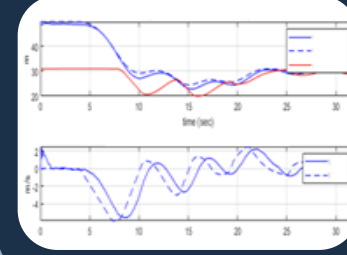
Sensor Fusion



Planning



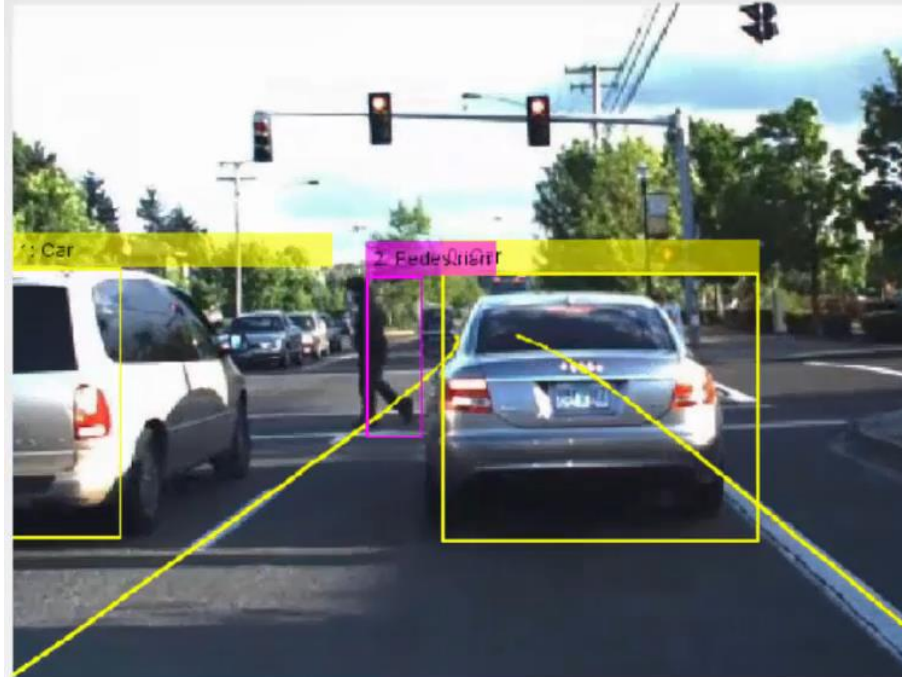
Decision & Controls



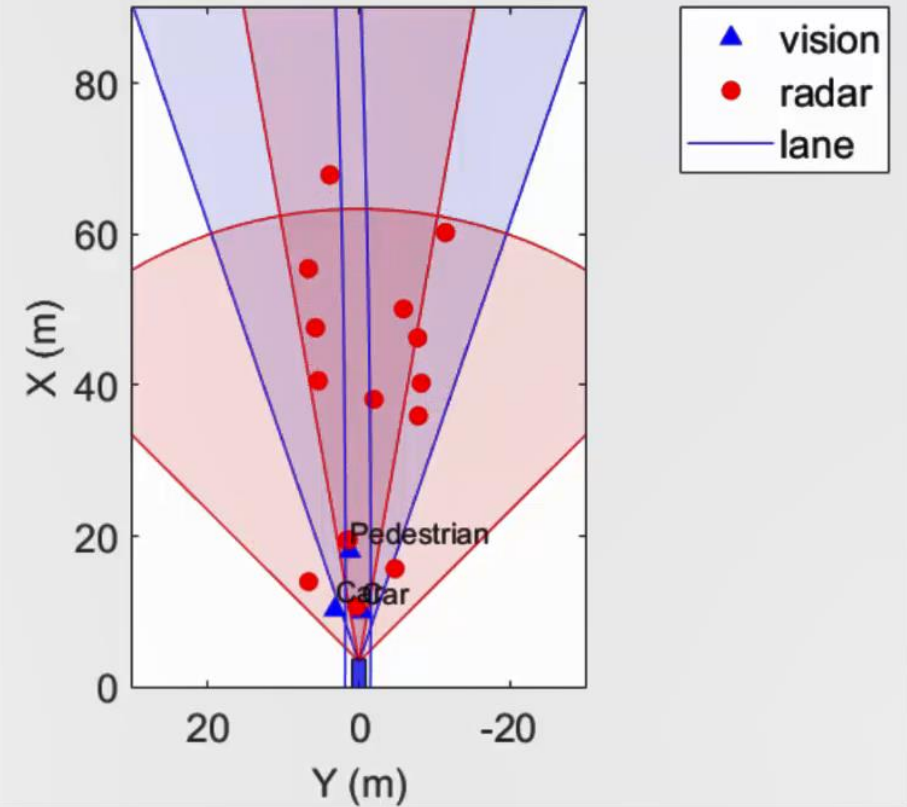
Key subsystems of an automated driving system

Perception

Image Coordinates



Vehicle Coordinates

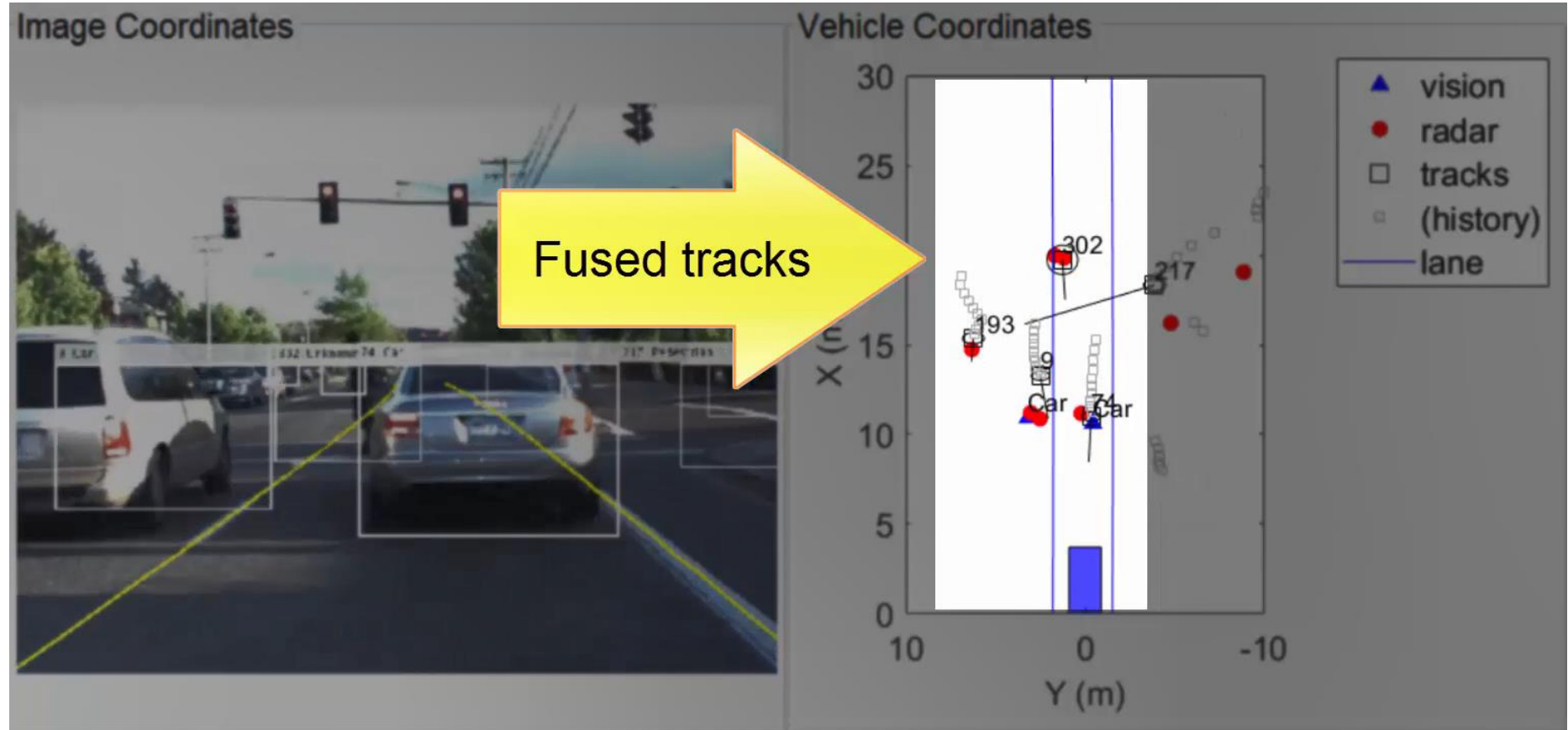


Key subsystems of an automated driving system

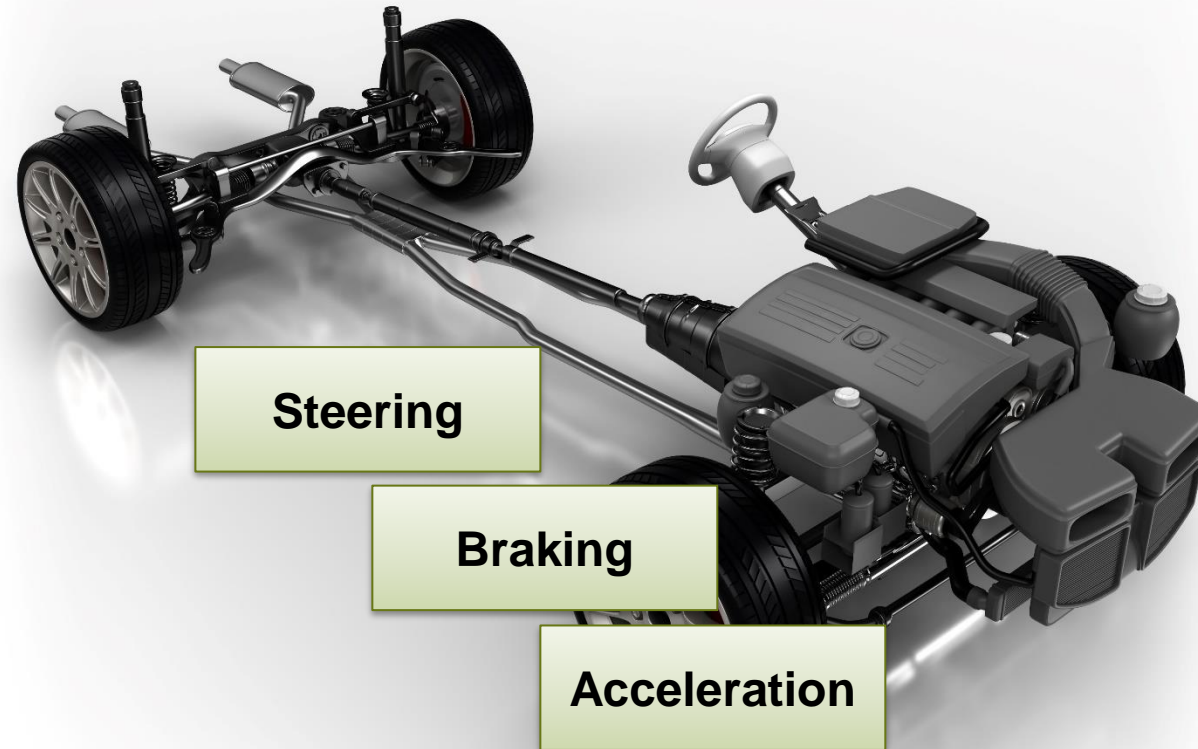
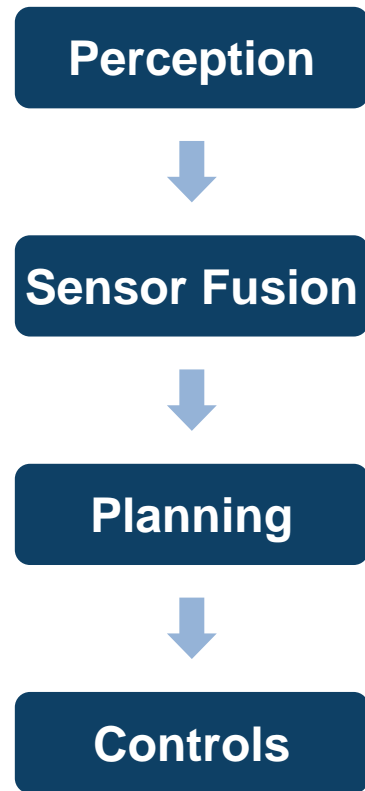
Perception



Sensor Fusion



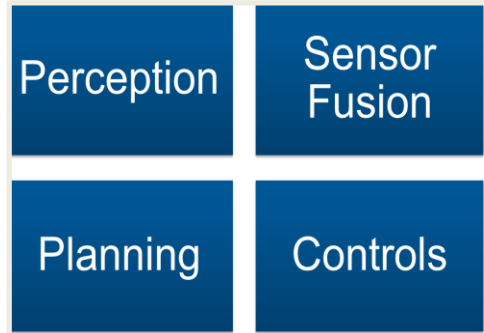
Key subsystems of an automated driving system



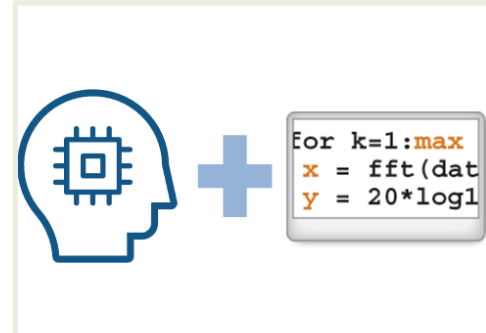
Key subsystems of an automated driving system



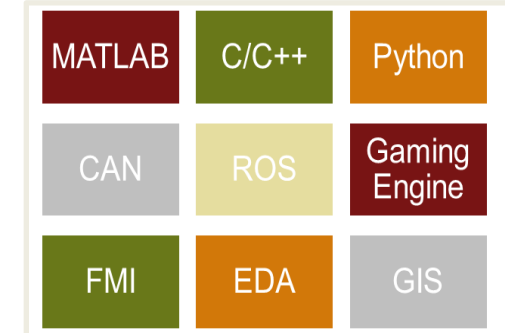
Challenges in developing and testing autonomous systems



Complexity of subsystems



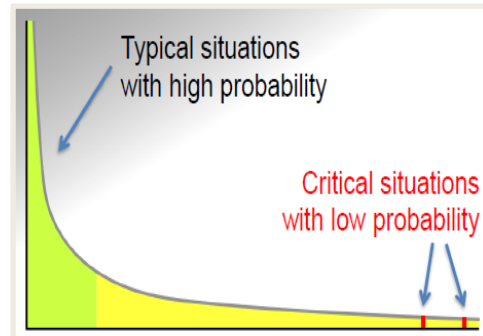
AI and traditional algorithms



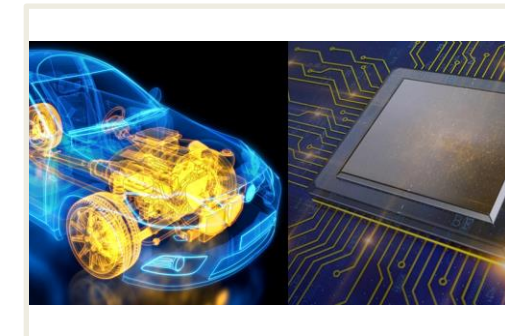
Diverse, disconnected tools and workflows



Recreating difficult real-world scenarios



Long tail of edge cases*



Billions of kms for reliability test**

* Saarland University

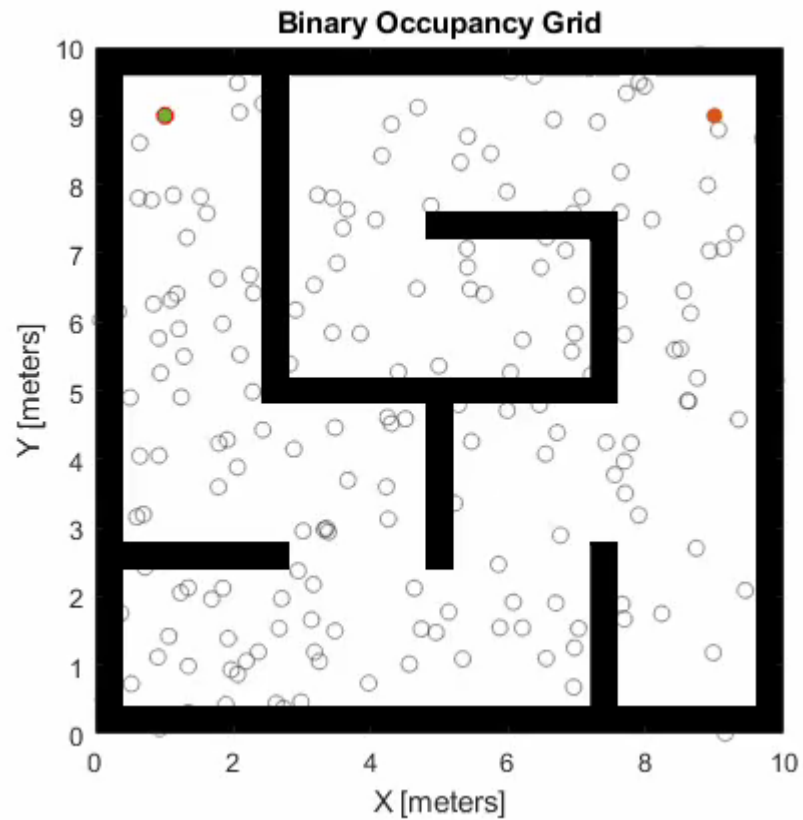
Artificial Intelligence as Key Enabler for Autonomous Systems



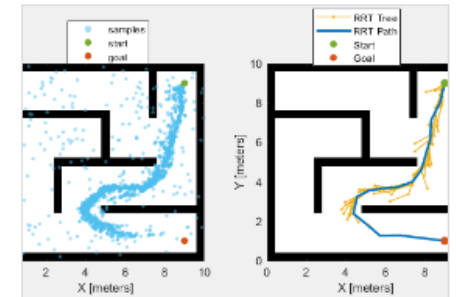
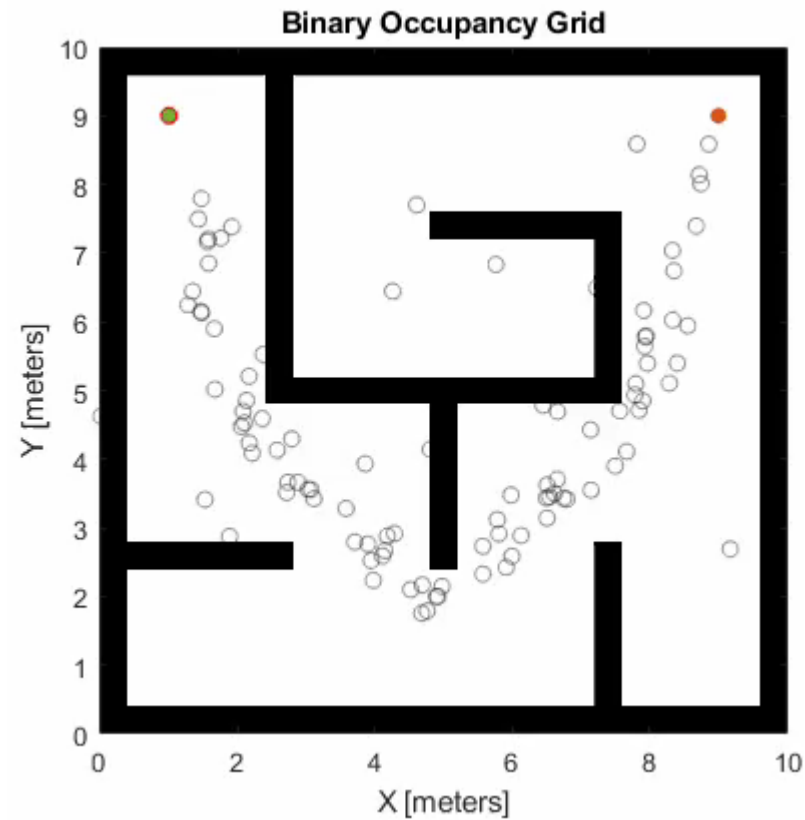
Object detection
Semantic segmentation
GANs
And so on ...

Accelerate Motion Planning with Deep Learning

**RRT* Path
Uniform Sampling ($\lambda=0$)**



**RRT* Path
DL Based Sampling ($\lambda = 0.9$)**



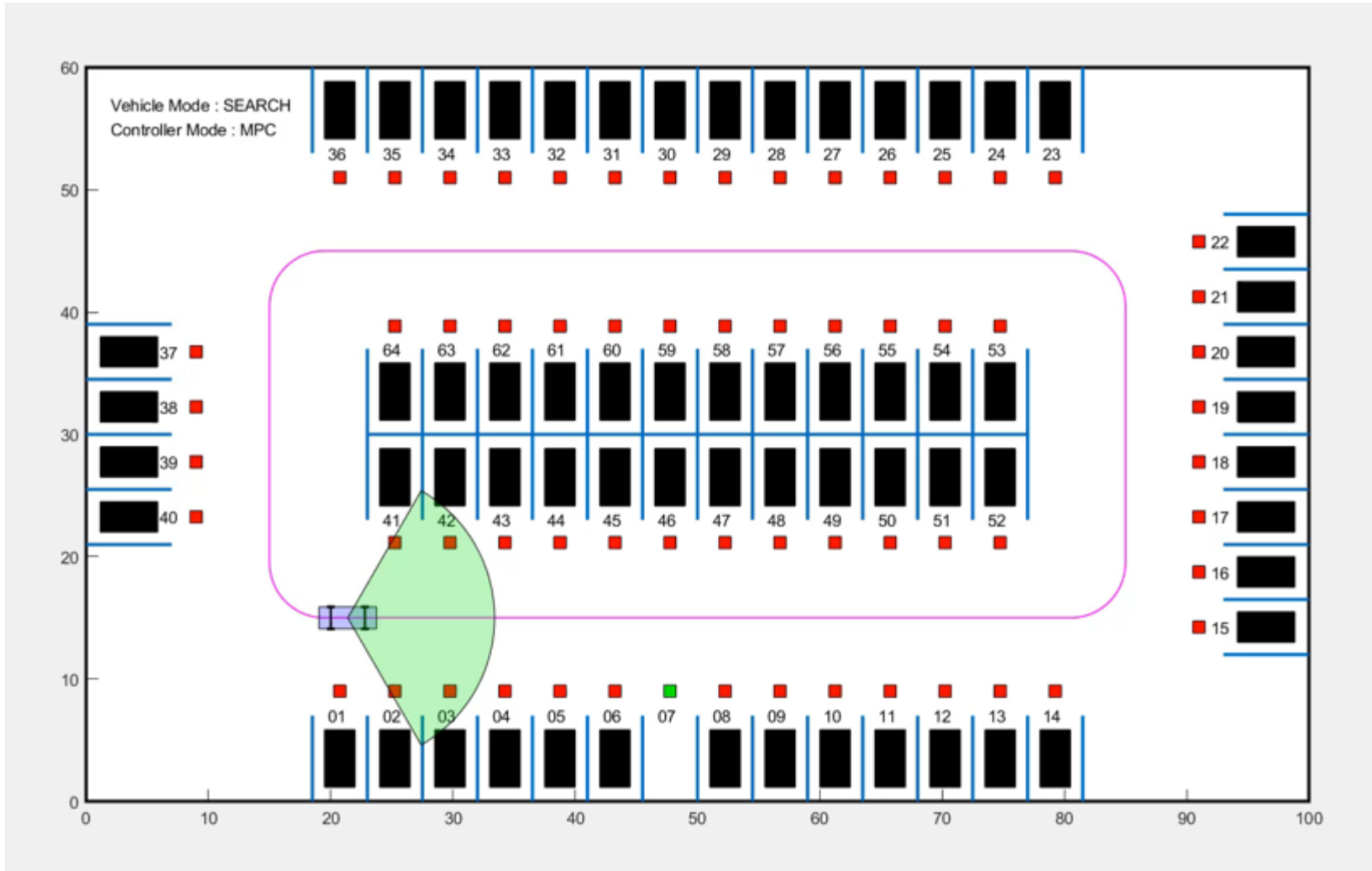
Accelerate Motion Planning with Deep-Learning-Based Sampler

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.



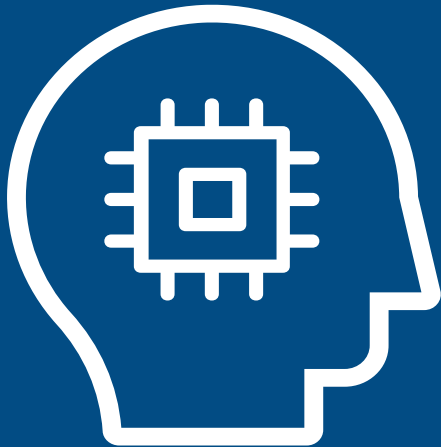
Agenda

- 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

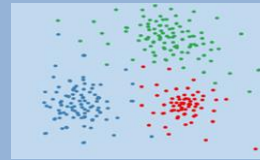


MACHINE LEARNING

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

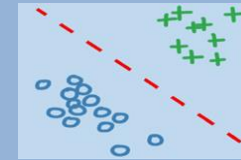
UNSUPERVISED LEARNING

(No Labeled Data)



SUPERVISED LEARNING

(Labeled Data)

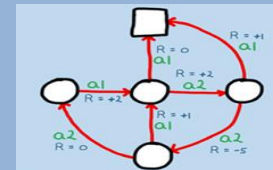


DEEP LEARNING
(Neural networks with many layers)

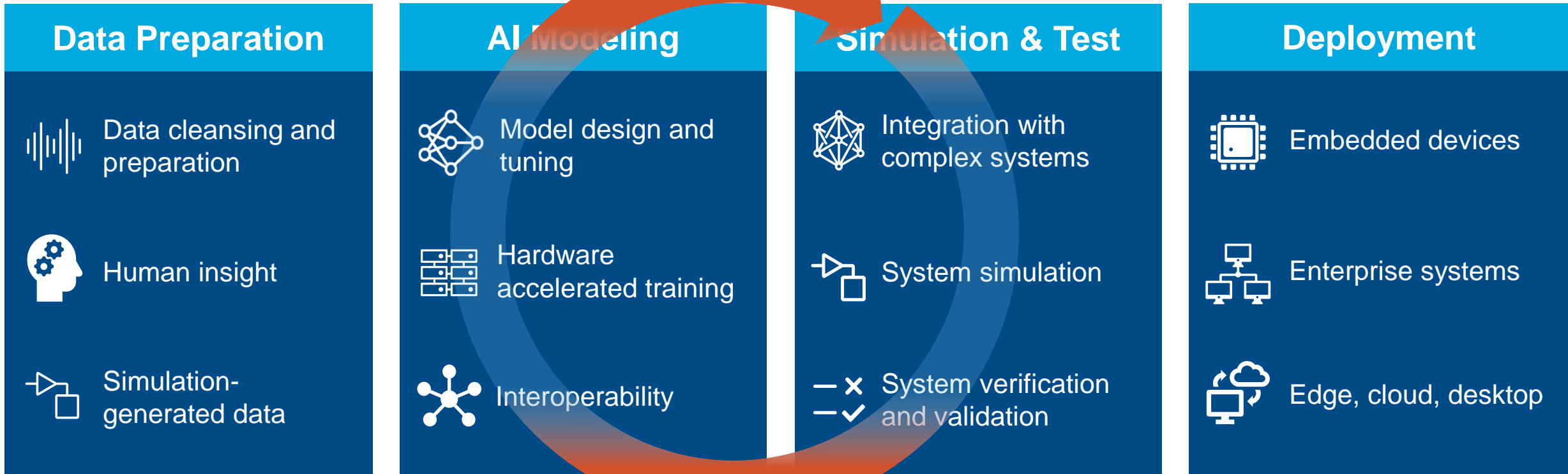


REINFORCEMENT LEARNING

(Interaction Data)



Brief Overview for AI-driven system design



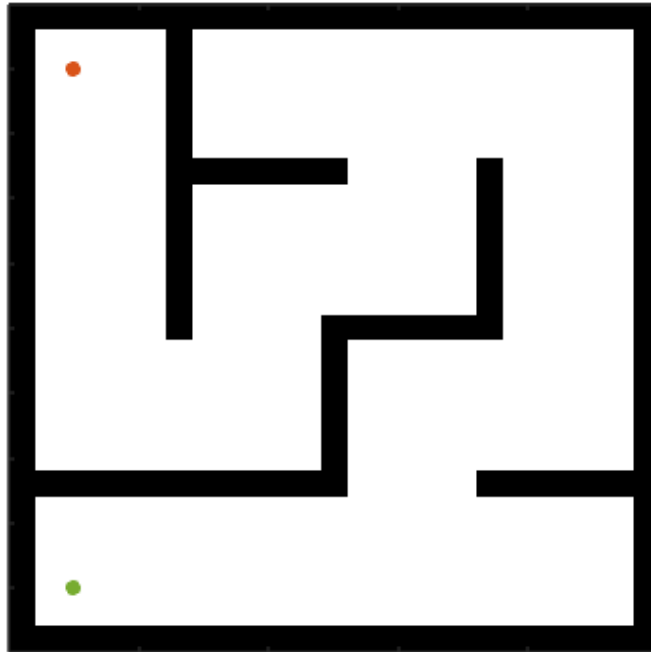
- [Labeller apps](#)
- [Unreal co-simulation](#)
- Data generation- Virtual sensor modelling ([Camera](#), [LIDAR](#), [RADAR](#))
- Simscape

- [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](#)

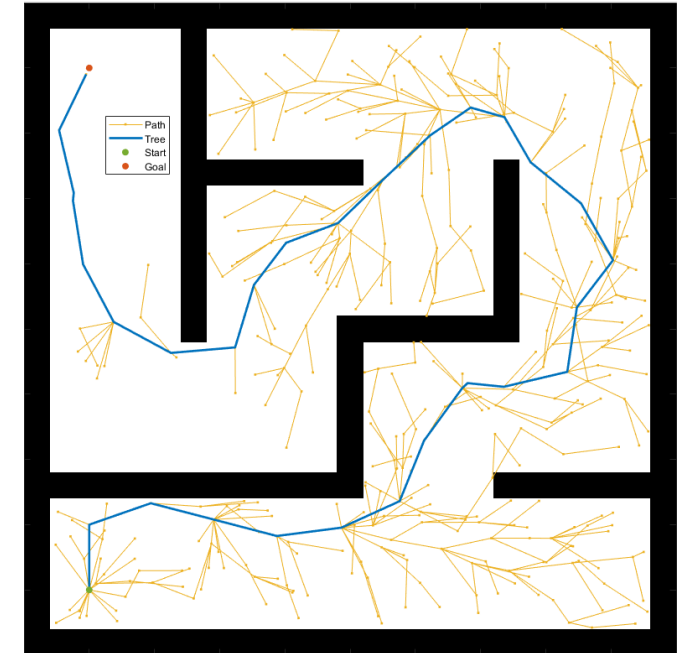
Accelerate Motion Planning with Deep Learning



Random Maze Dataset
representing occupancy map,
start & goal locations



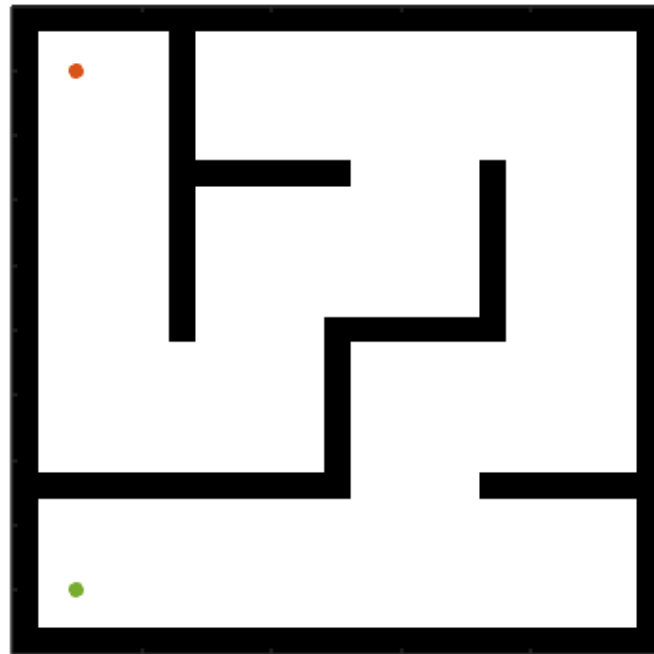
Uniformly Sampled space
(used by conventional motion
planning algorithms e.g., RRT/ RRT*)



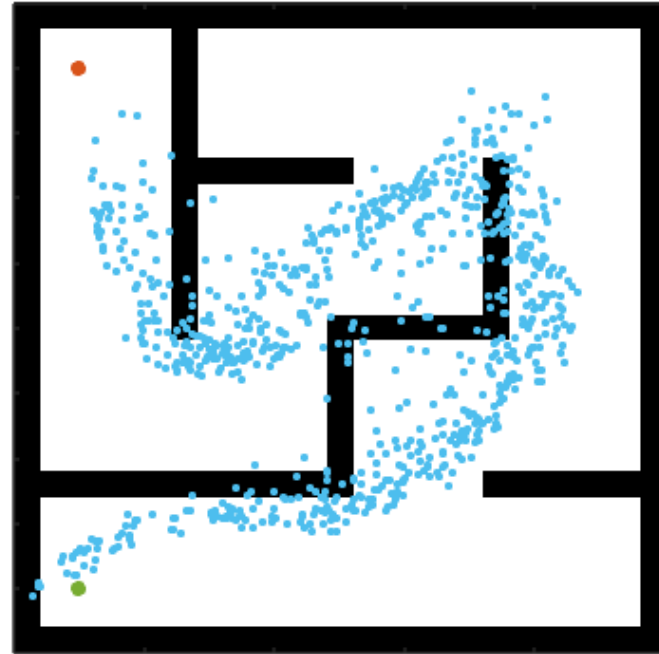
RRT*
(Rapidly Exploring Random Tree)

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

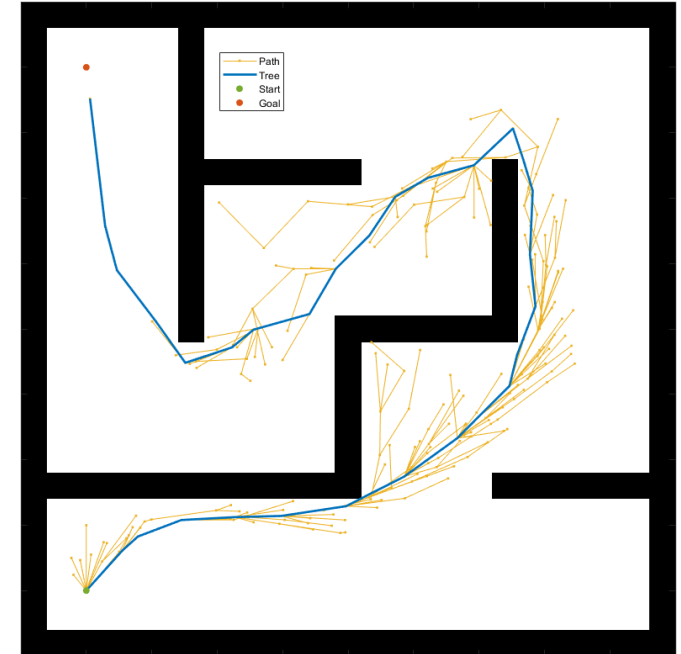
Accelerate Motion Planning with Deep Learning



Random Maze Dataset
representing occupancy map,
start & goal locations



DL-Based Sampler

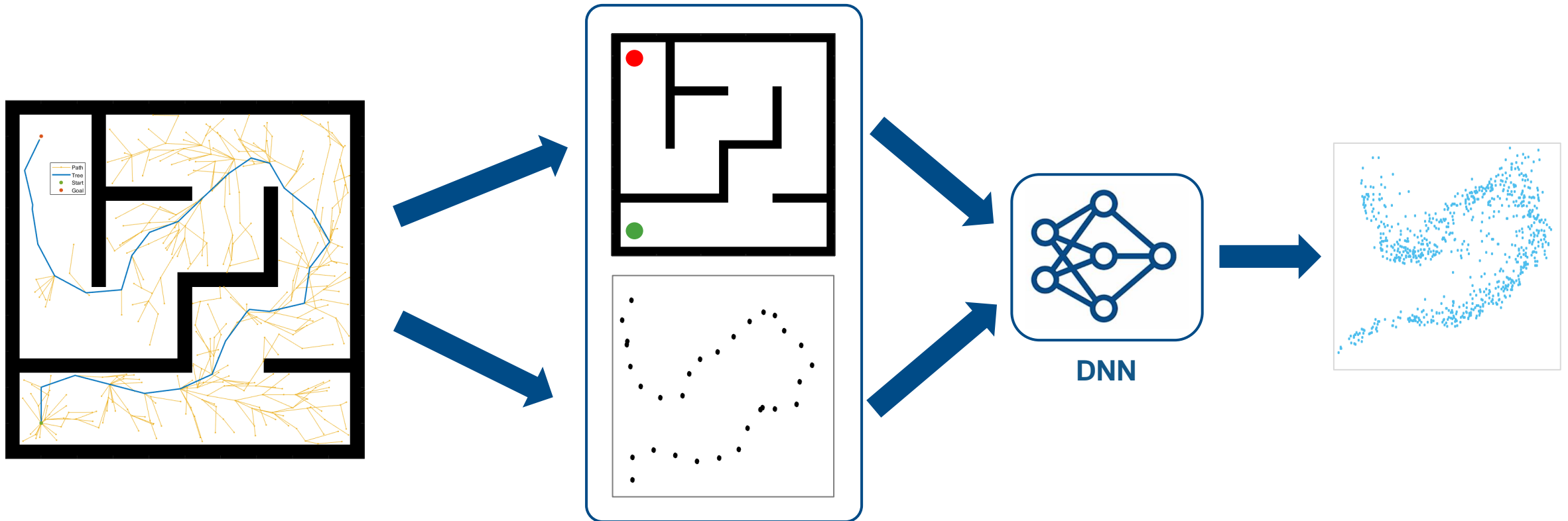


RRT*
(Rapidly Exploring Random Tree)

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

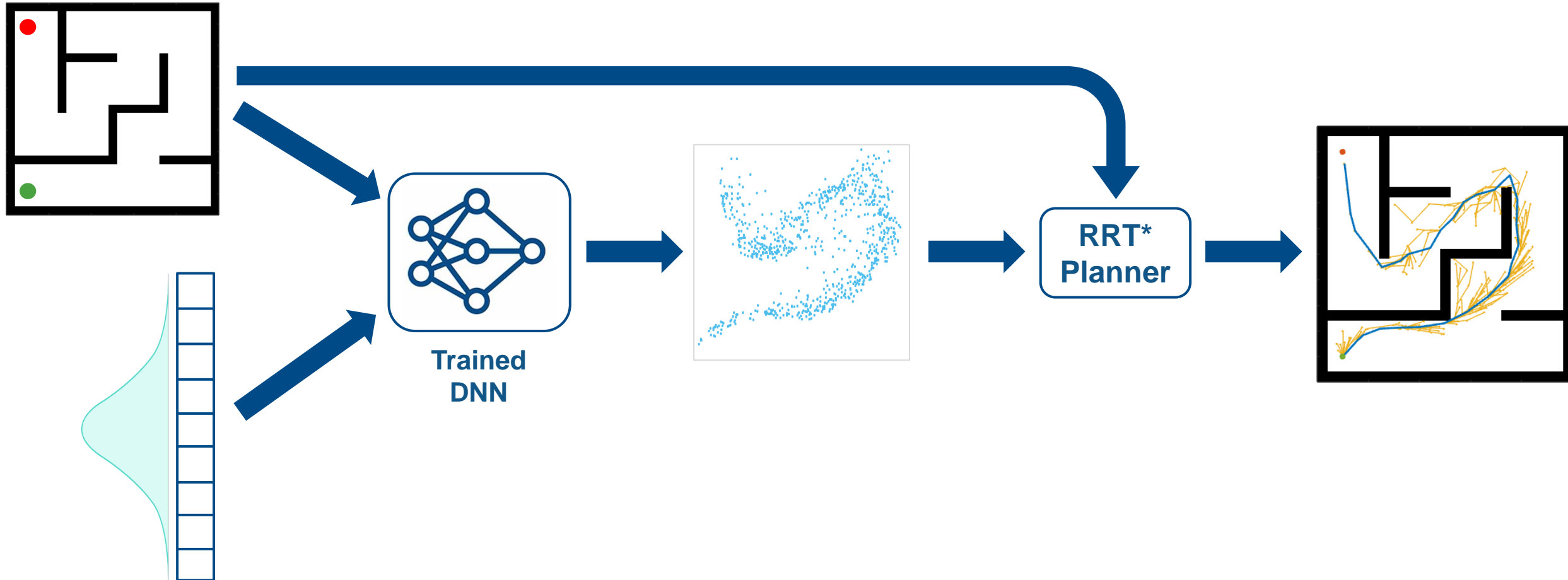
AI Workflow

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



AI Workflow

Predict: Integrate trained models into applications



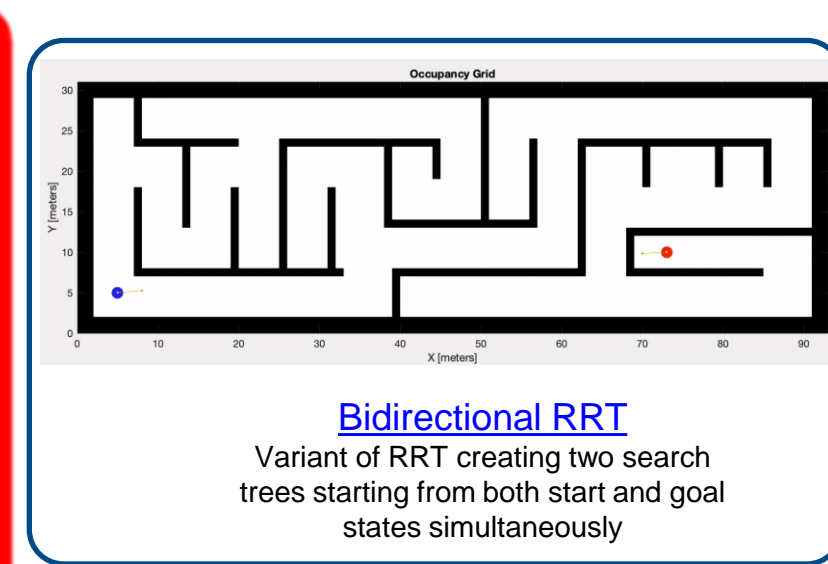
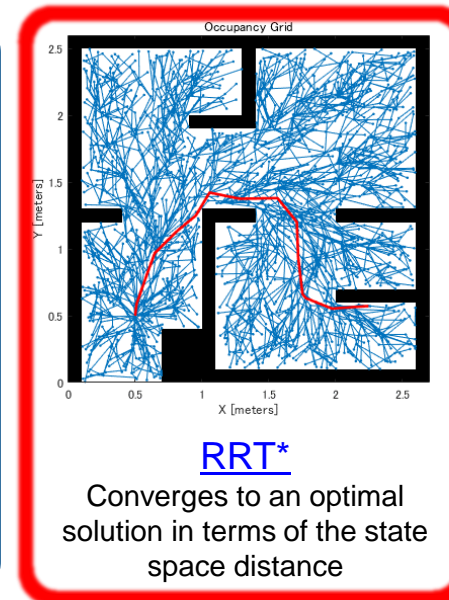
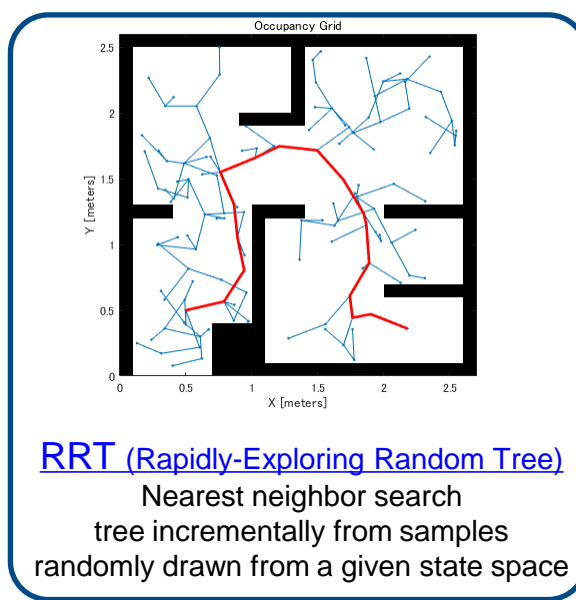
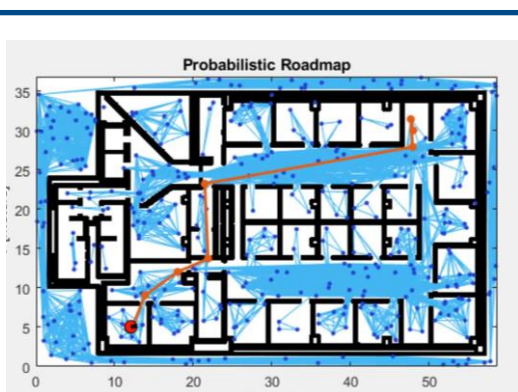
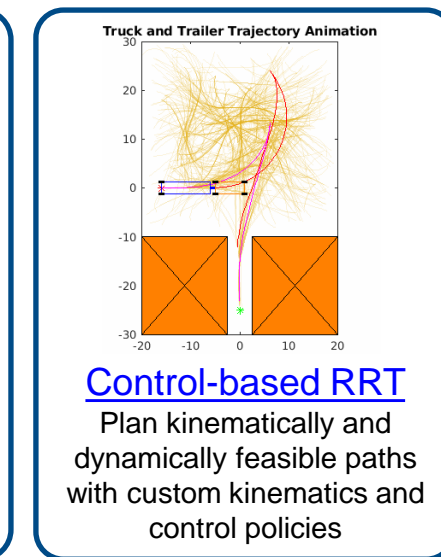
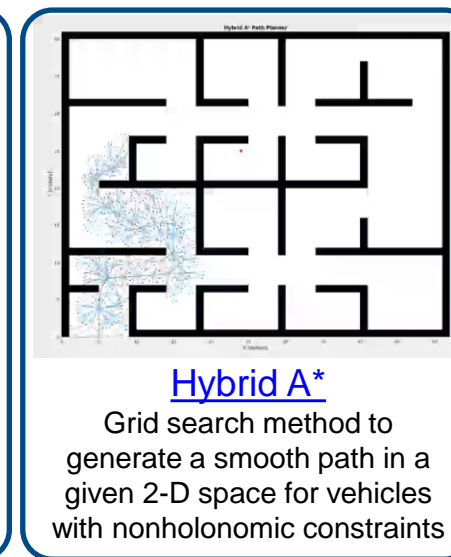
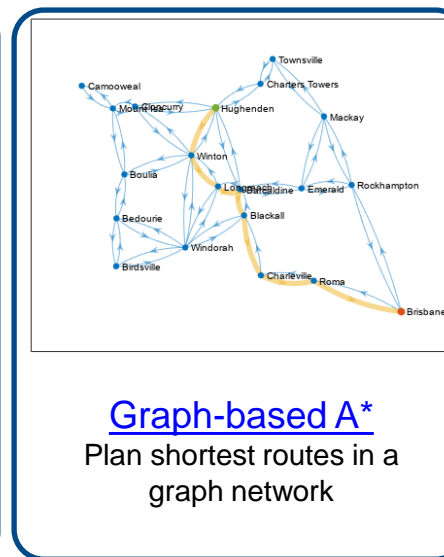
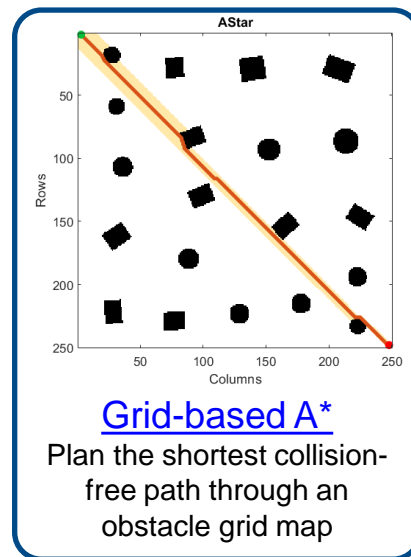
Path Planning

Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation

1 Data Preparation

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




Data Generation with RRT*


Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation

1 Data Preparation

 Data cleansing and preparation

 Human insight

 Simulation-generated data

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

```
% 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]];
stateValidator = validatorOccupancyMap(stateSpace, binaryOccupancyMap(maps{1}, 10));
% Run the plannerRRTStar
for i = 1:numMaps
    waitbar(i/numMaps, f, "Generating samples...");

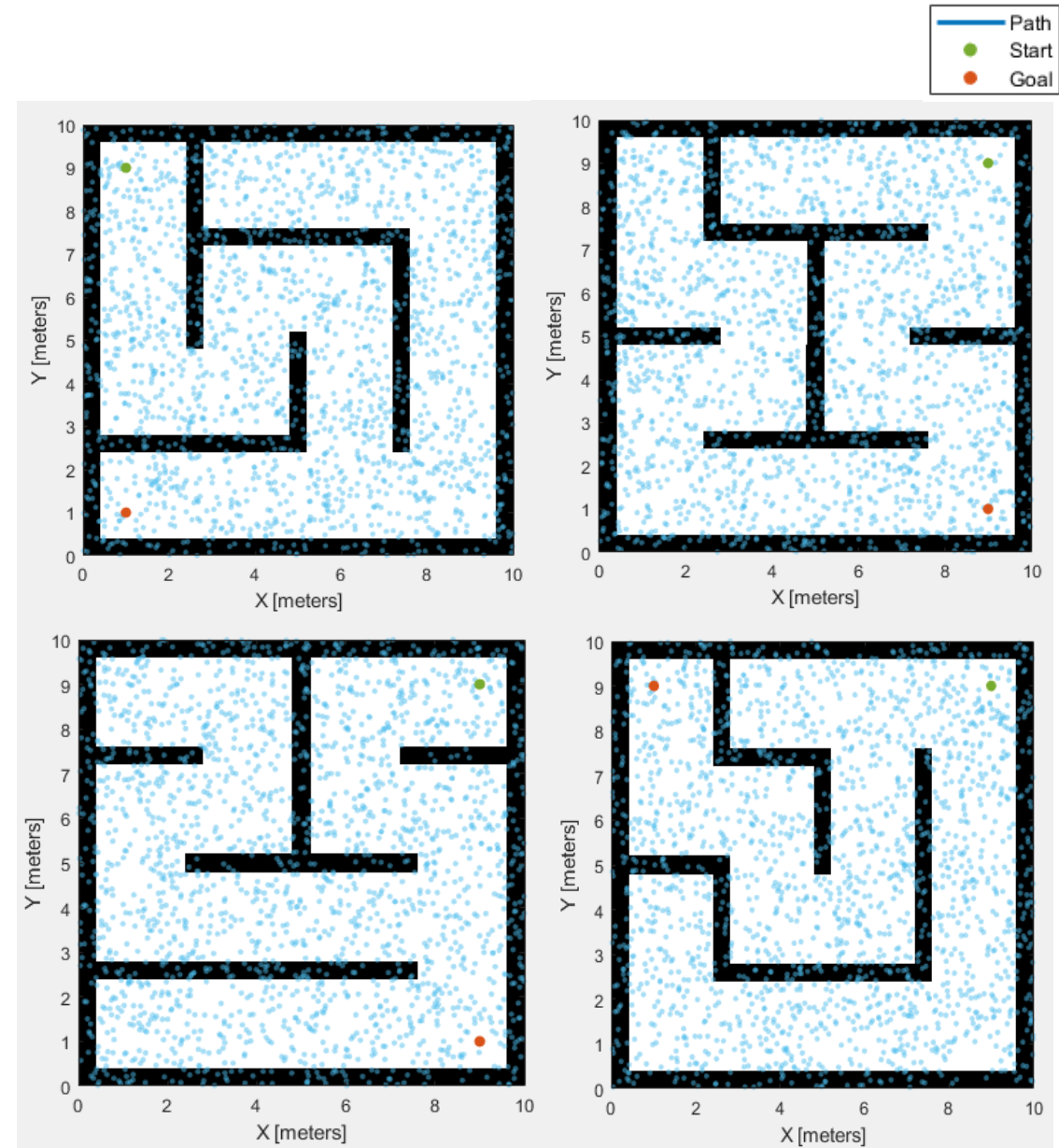
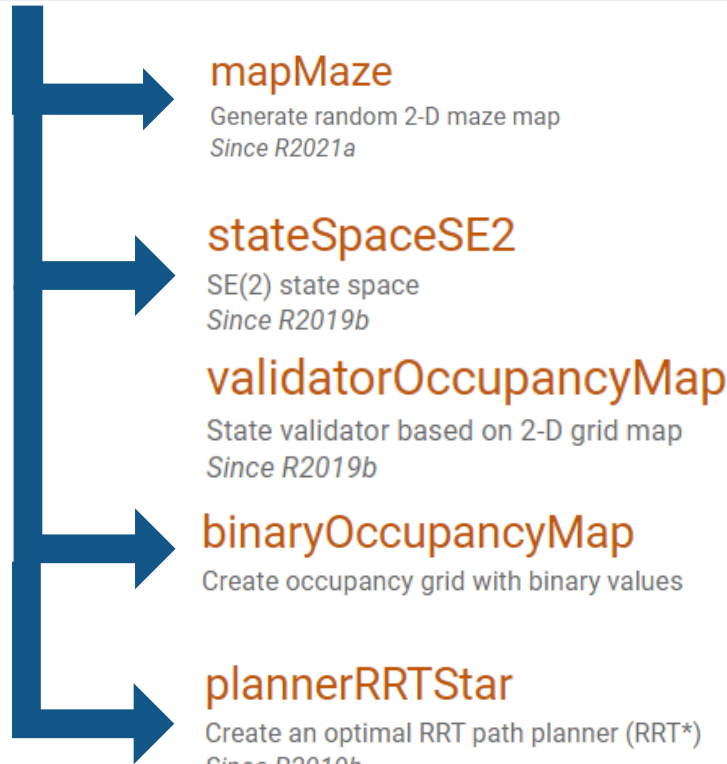
    % Inflate obstacle to get safe paths
    map = binaryOccupancyMap(maps{i}, 10);
    inflate(map, 4, 'grid');

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

Data Generation with RRT*

Navigation Toolbox

Design, simulate, and deploy algorithms for autonomous navigation



*No. of samples to be generated = 2000

Data Generation with RRT*

Navigation Toolbox






Design, simulate, and deploy algorithms for autonomous navigation

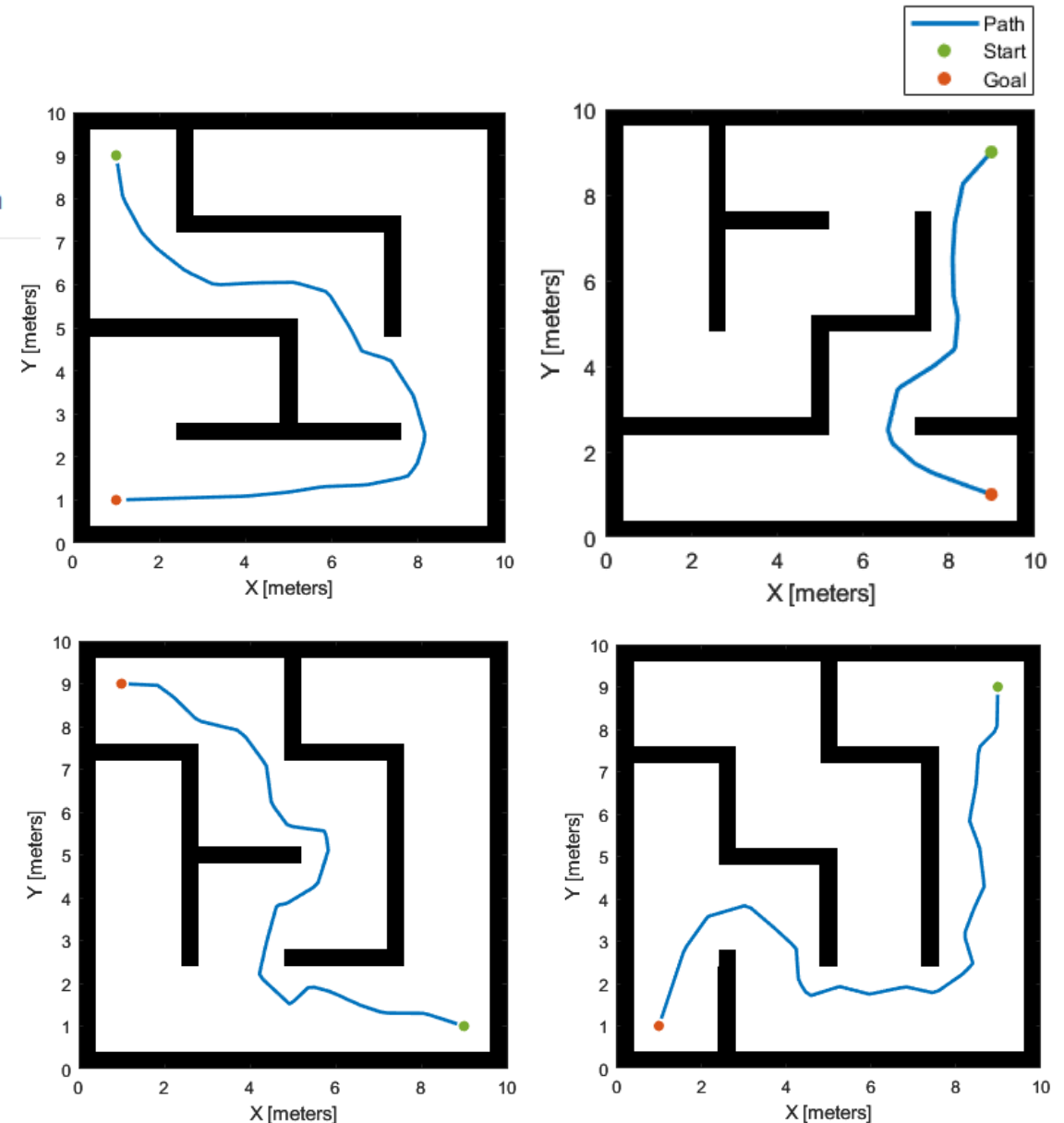
1 Data Preparation

Data cleansing and preparation

Human insight

Simulation-generated data

- 
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

The image shows the MATLAB R2021a interface. The top ribbon includes tabs for HOME, PLOTS, APPS, LIVE EDITOR, INSERT, and VIEW. The LIVE EDITOR tab is active, showing a toolbar with icons for file operations (New, Open, Save, Print, Export), navigation (Go To, Bookmark), text formatting (Normal, Bold, Italic, Underline, Monospace), code control (Code, Control, Task, Refactor), section management (Section Break, Run, Run and Advance, Run to End), and execution (Run, Step, Stop). The current folder is 'TrainNetworkProject3'. The Live Editor window displays the following content:

Built-In Training Experiment Using `trainNetwork`

Use this setup function to define the training data, network architecture, and training options for an experiment. Experiment Manager uses the outputs of this function to call the `trainNetwork` function. For more information, see [Configure Built-In Training Experiment](#).

Input

- `params` is a structure with fields from the Experiment Manager hyperparameter table.

Output

- `trainingData` is a datastore, numeric array, cell array of numeric arrays, or table used to store the training data.
- `layers` is a layer graph that defines the neural network architecture.
- `options` is a `trainingOptions` object.

The Command Window shows the following commands:

```
>> deepNetworkDesigner
fx >>
```

The Workspace window is empty, showing columns for Name and Value. The bottom of the interface has a 'Details' section with the text 'Select a file to view details'.

Experiment Manager

Experiment Manager

EXPERIMENT MANAGER

Open
 Save
 Duplicate
 FILE

Layout
 Use Parallel
 Run
 Stop
 ENVIRONMENT PARALLEL RUN

Experiment Browser

TrainNetworkProject3

- Hyperparameter_Tuning
 - Result7
 - Result6
 - Result5
 - Result4
 - Result3
 - Result2
 - Result1

Hyperparameter_Tuning | Hyperparameter_Tuning | Result4 | Hyperparameter_Tuning | Result5 | Hyperparameter_Tuning | Result6 | Hyperparameter_Tuning | Result7

Description

Experimenting for robustness with below parameters

- 1) Learning rate
- 2) Solver
- 3) Neural networks archs

Hyperparameters

Strategy: Bayesian Optimization

Name	Range	Type	Transform
mySolver	["adam" "rmsprop" "sgdm"]	categorical	none
myInitialLearnRate	[1e-4 1]	real	none
myNetworkChoice	["a" "b"]	categorical	none

+ Add - Delete

Bayesian Optimization Options

Name	Value
Maximum time (in seconds)	Inf
Maximum number of trials	30

Setup Function

Experiment2_setup1

+ New - Edit

Metrics

Standard training and validation metrics (such as accuracy, RMSE, and loss) are computed by default.

Custom Metrics

Importing Pretrained Network for Labelling Automation

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

AI Modeling



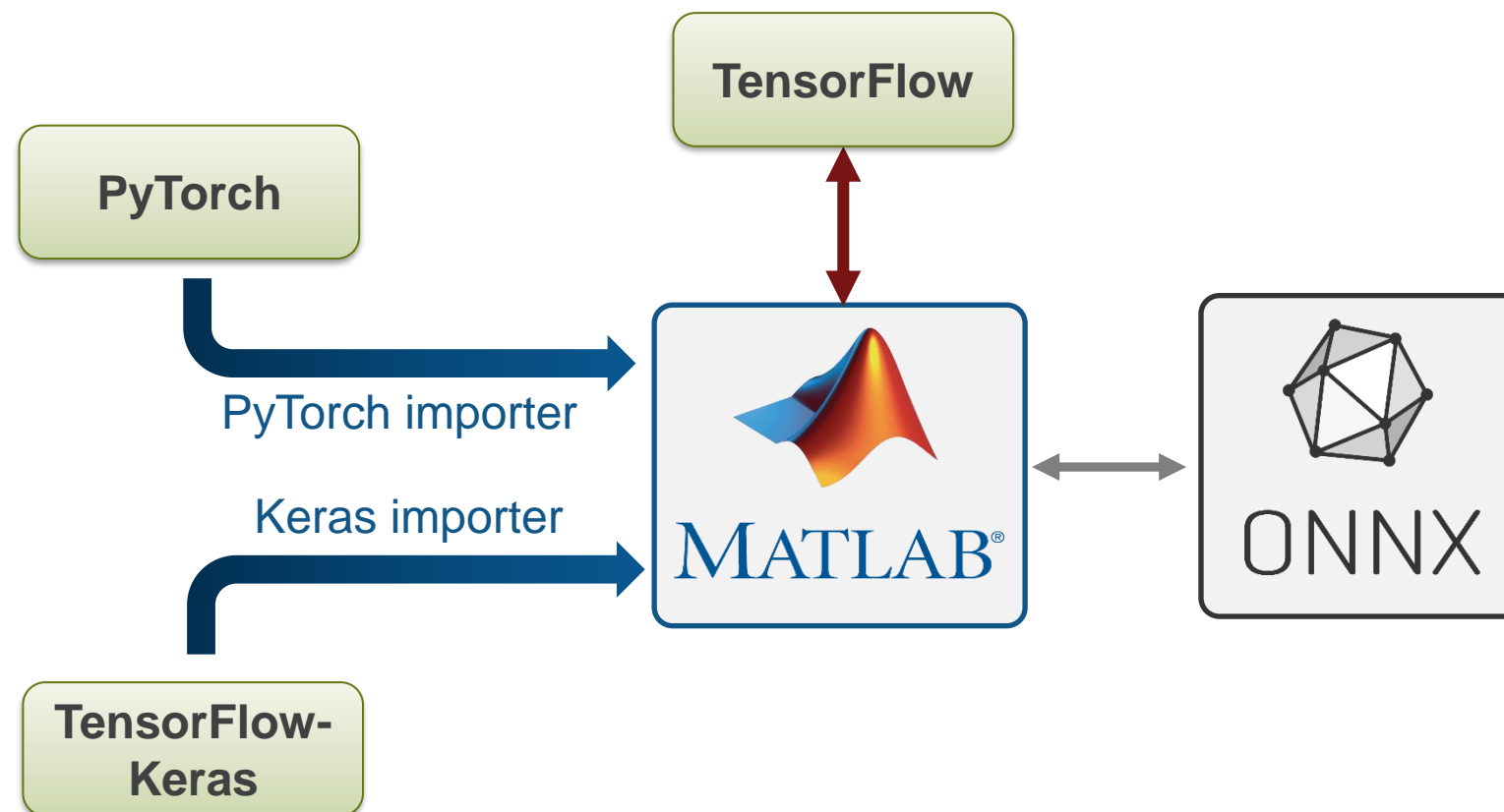
Model design and tuning



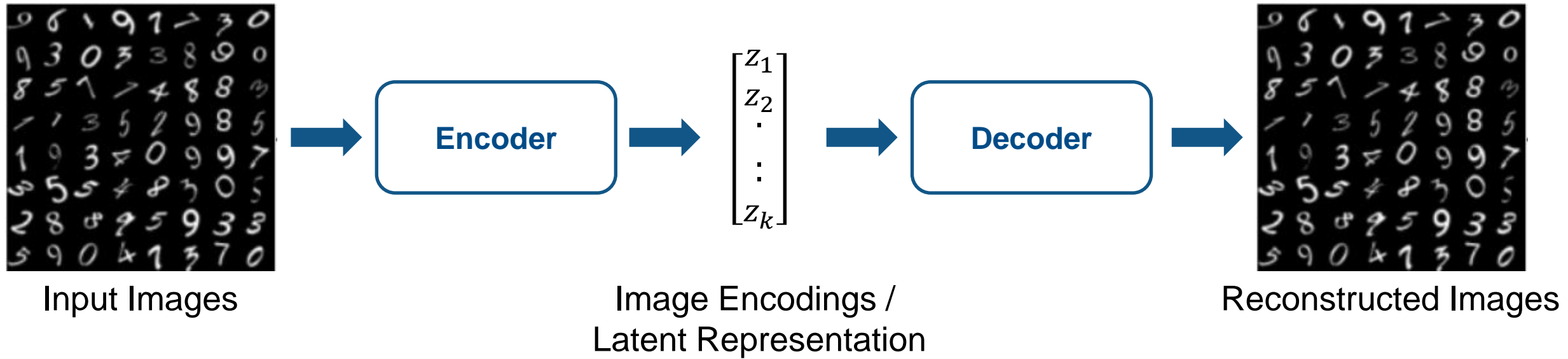
Hardware accelerated training



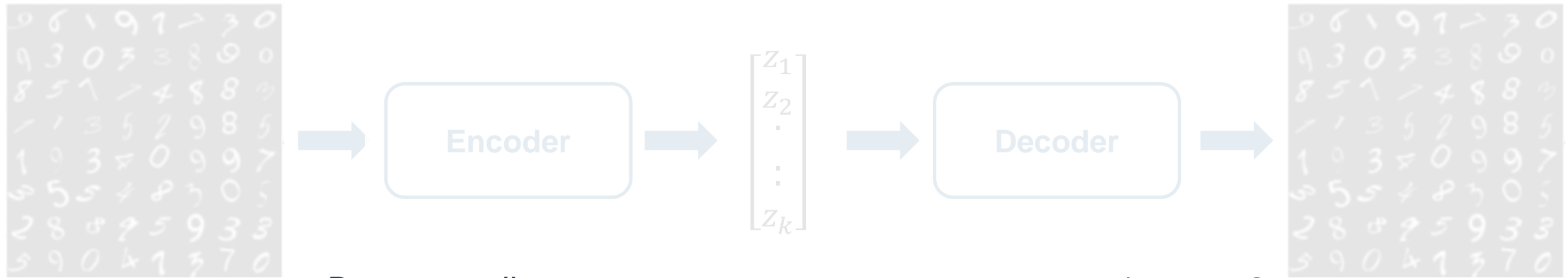
Interoperability



Autoencoders

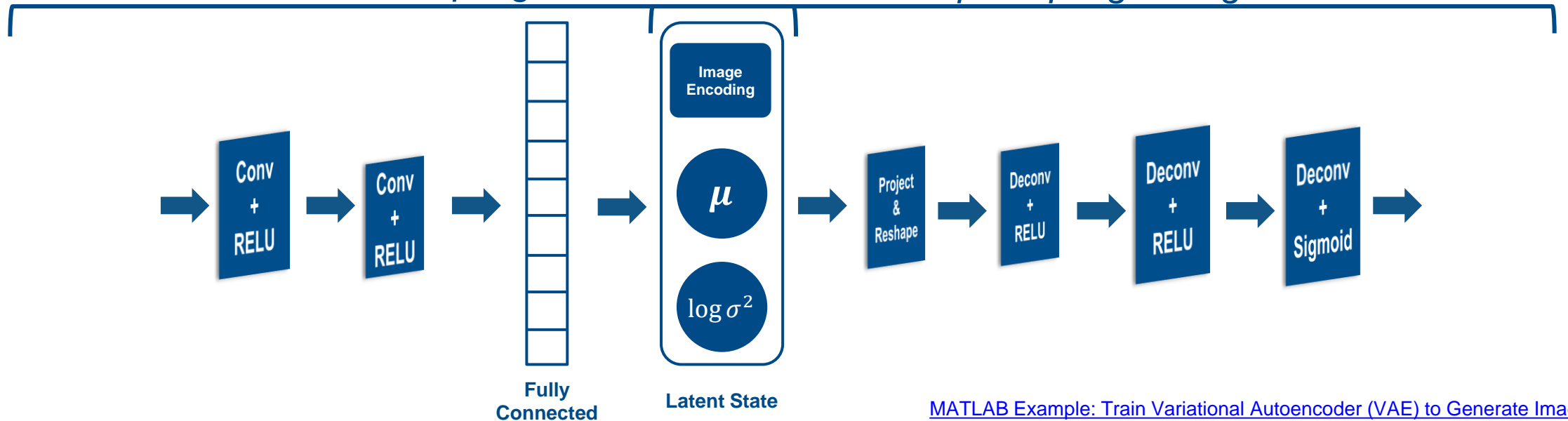


Variational Autoencoders for Image Re-Generation

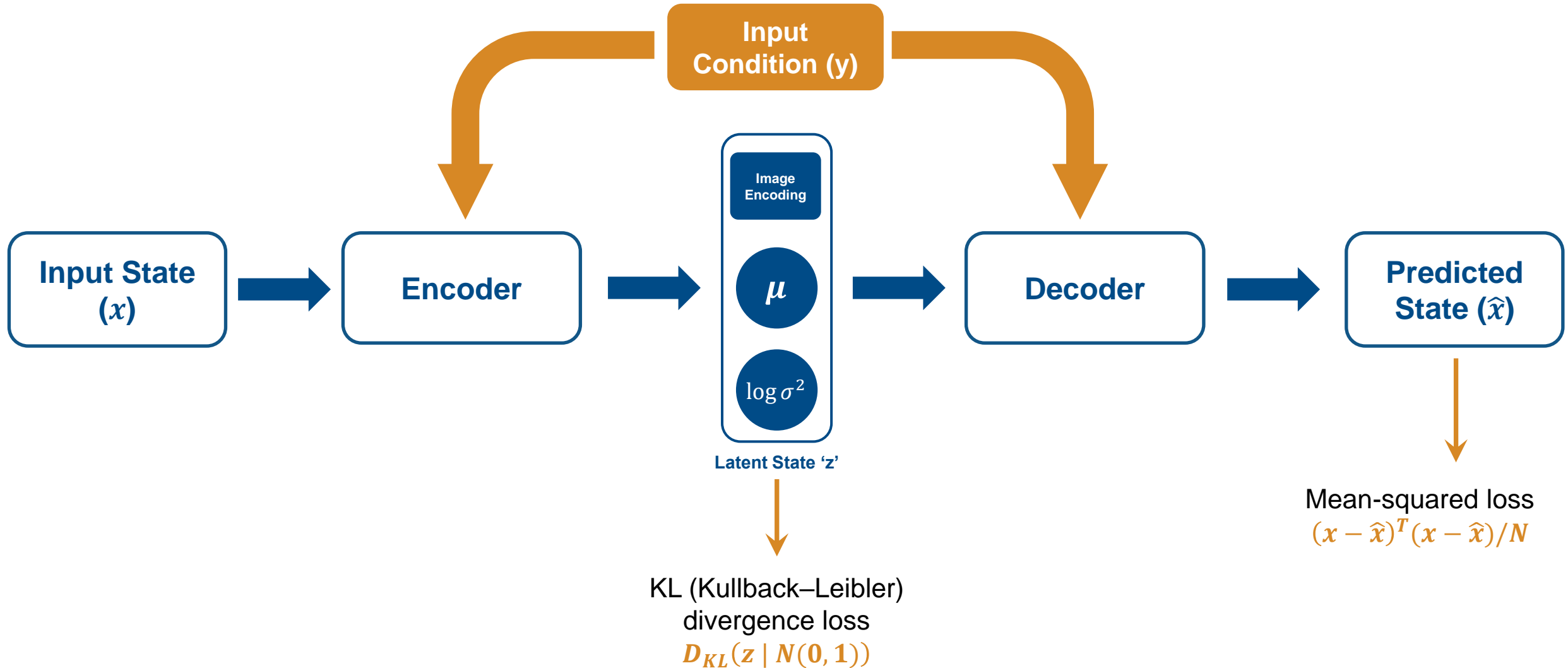


Downsampling

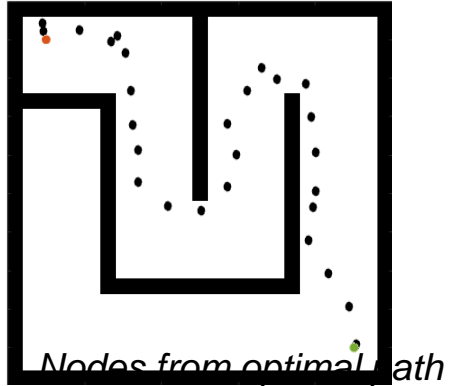
Upsampling / Image Generation



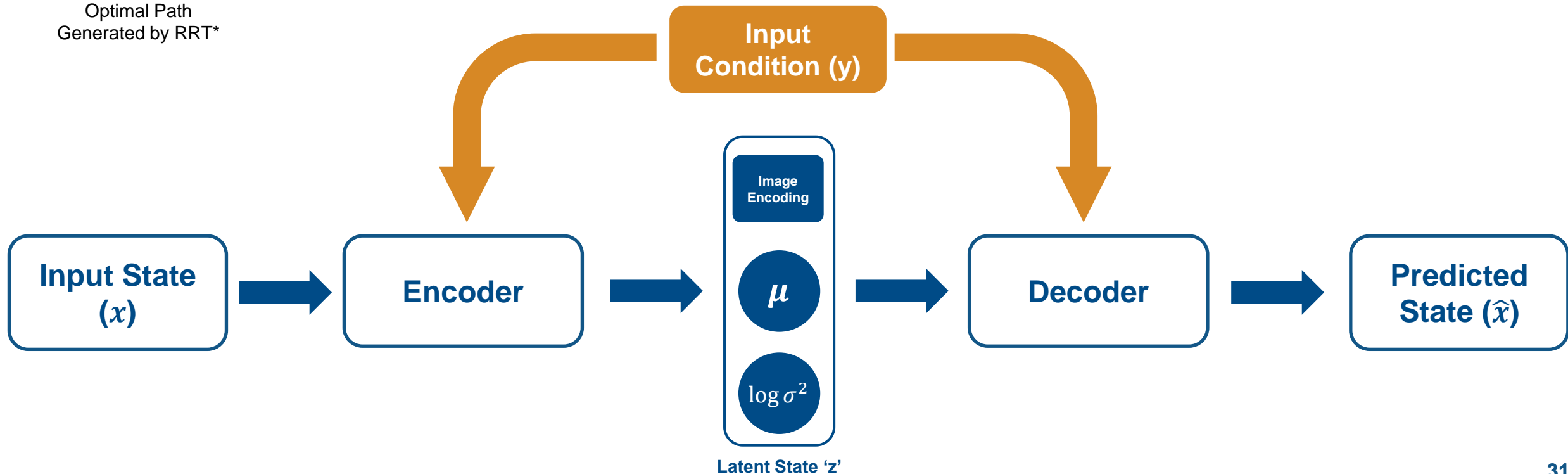
Conditional Variational Autoencoders (CVAE)



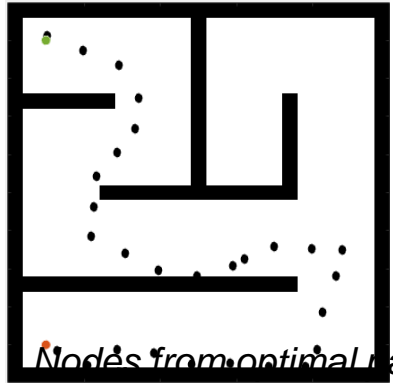
Conditional Variational Autoencoders (CVAE)



Occupancy Map
Start
Goal

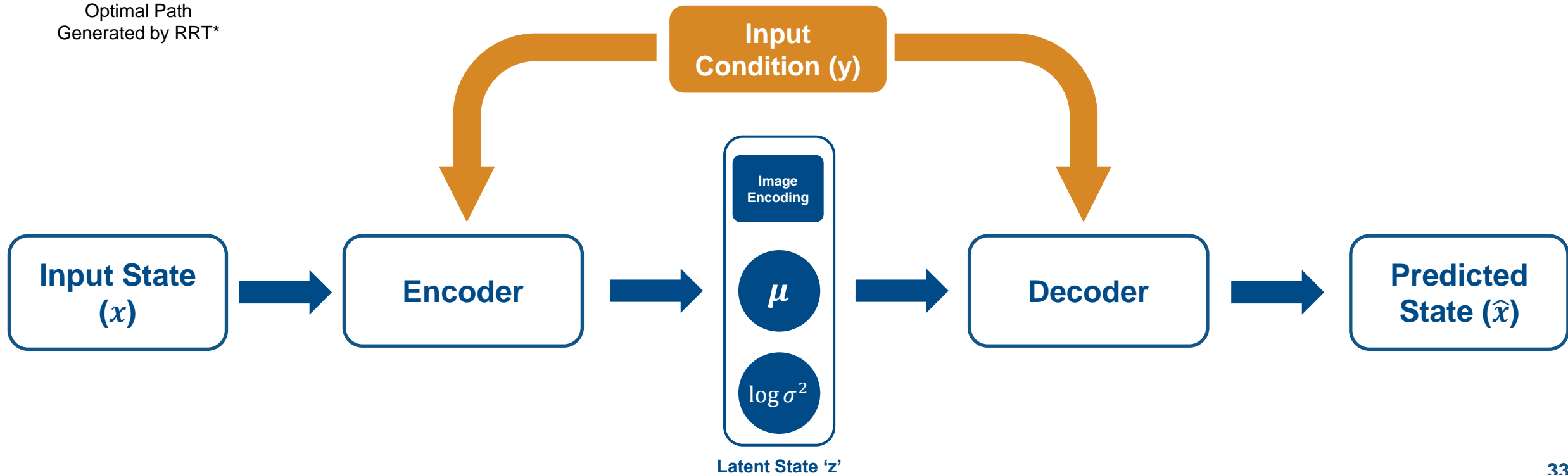


Conditional Variational Autoencoders (CVAE)

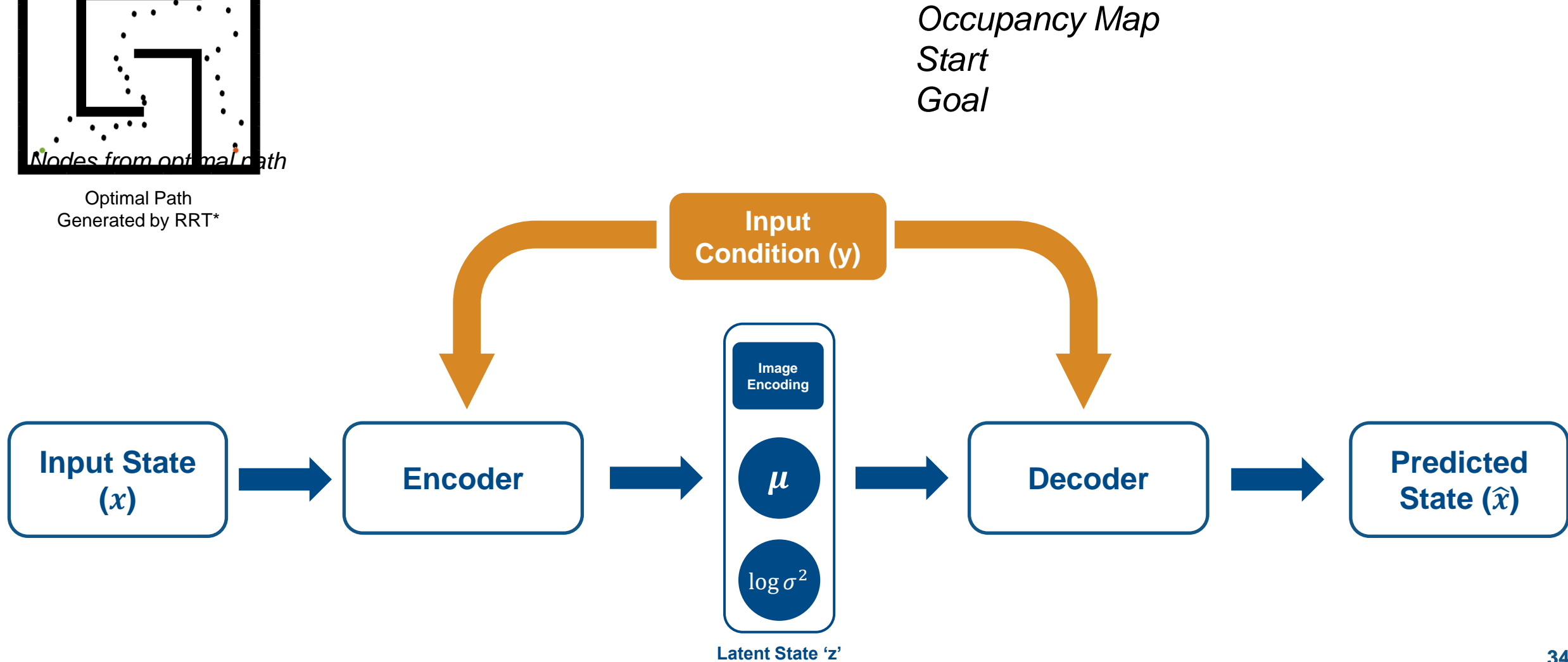
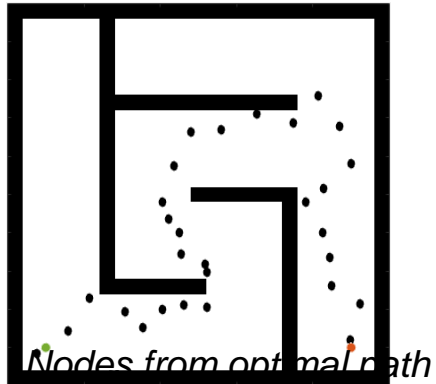


Optimal Path
Generated by RRT*

Occupancy Map
Start
Goal

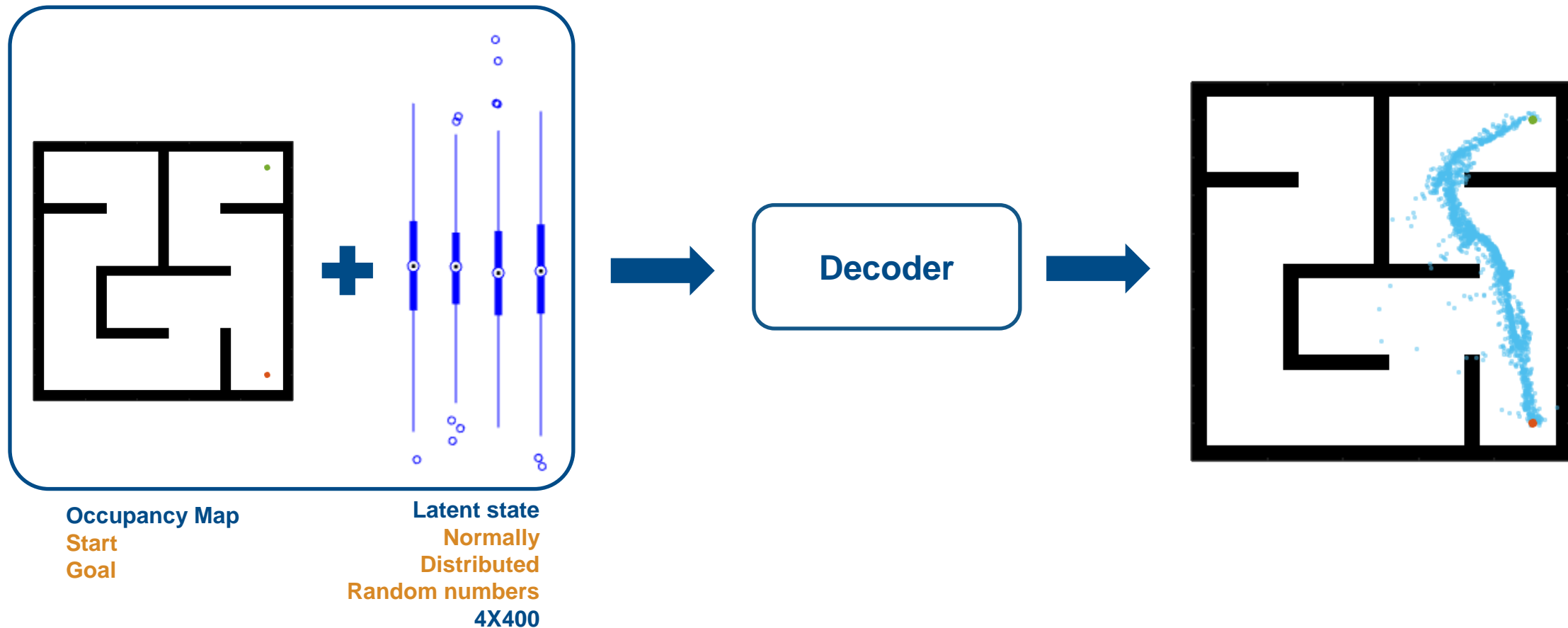


Conditional Variational Autoencoders (CVAE)



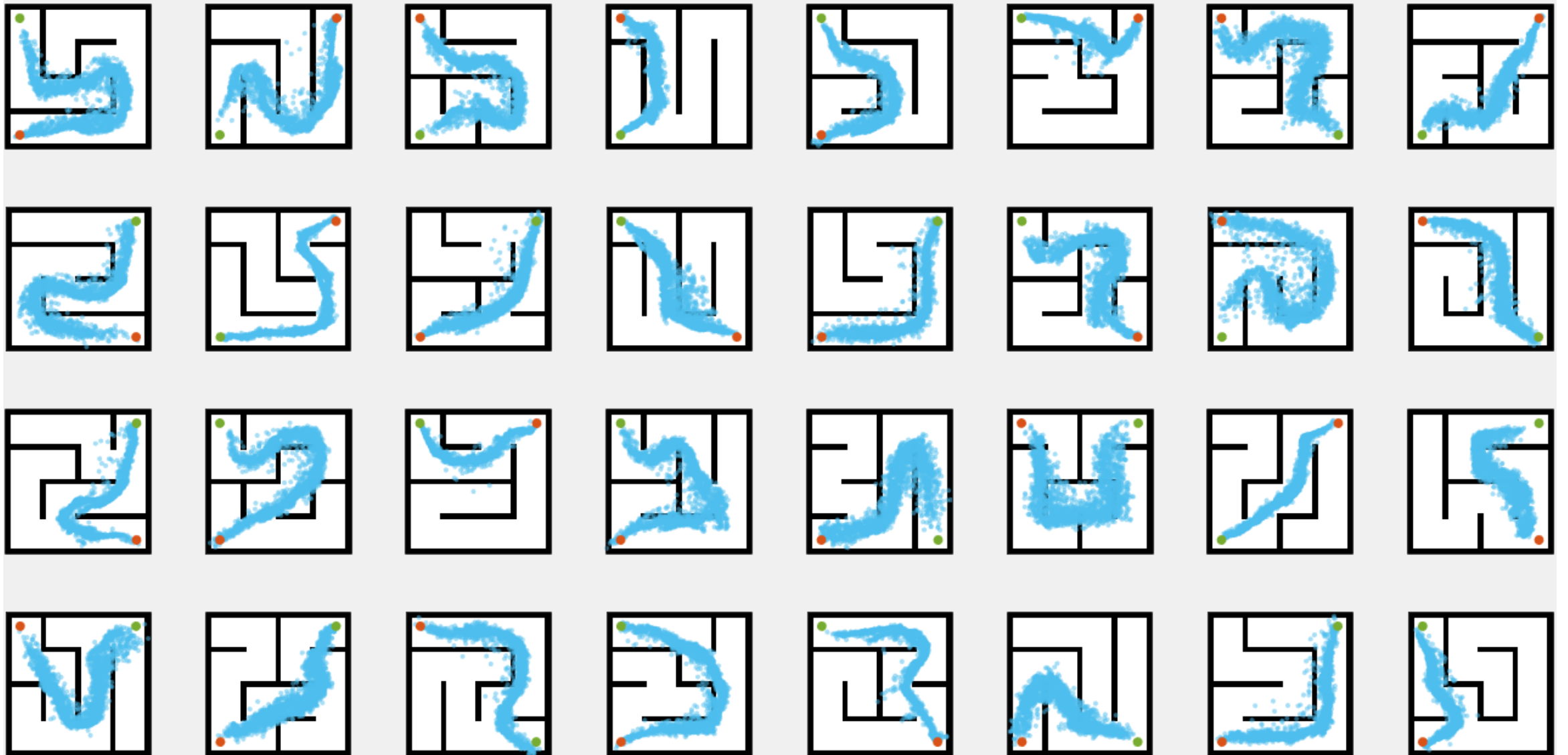
Decoder Network for Generating Optimal States

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

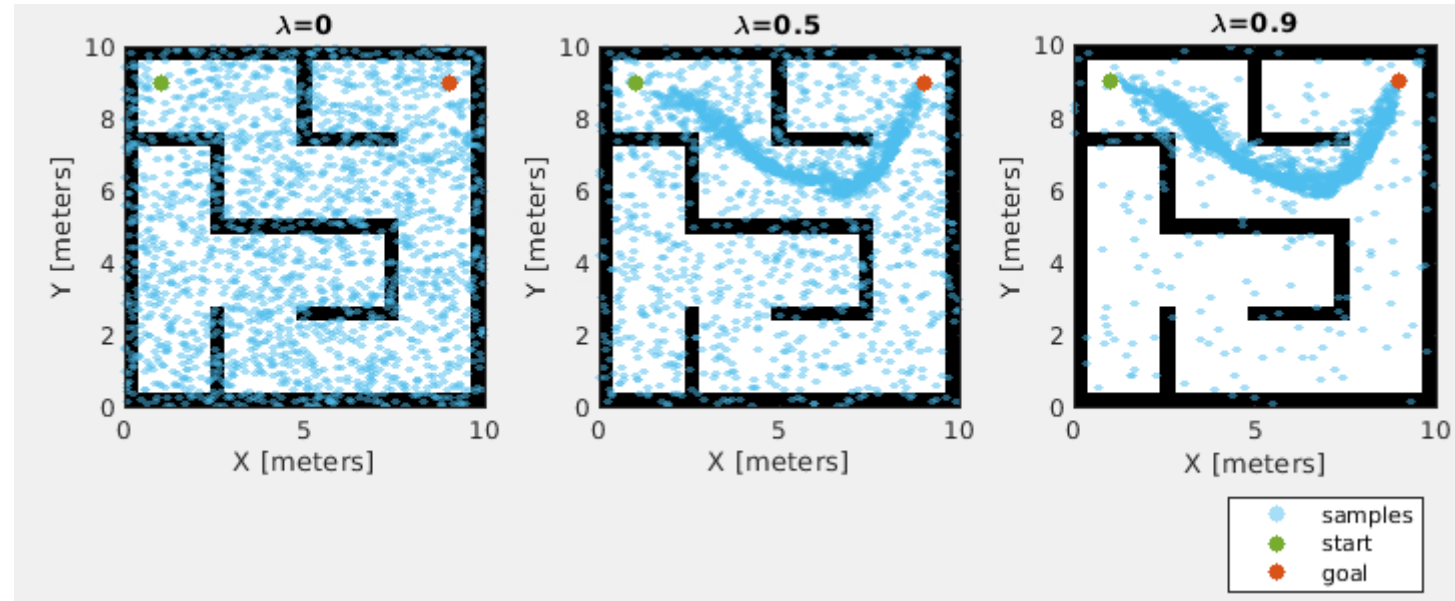


```
latentStates = dlarray(randn(latentStateSize, numSamplesPerSet), "CB");
```

More Examples on the Test Data

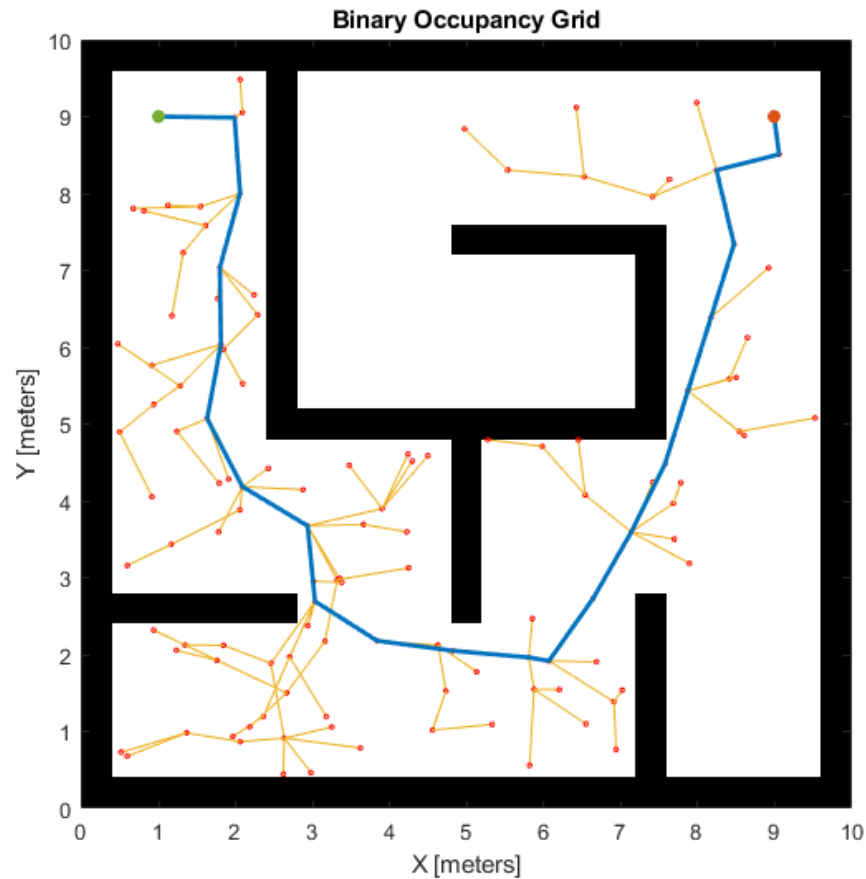


Choose the Learnable Sampling Factor

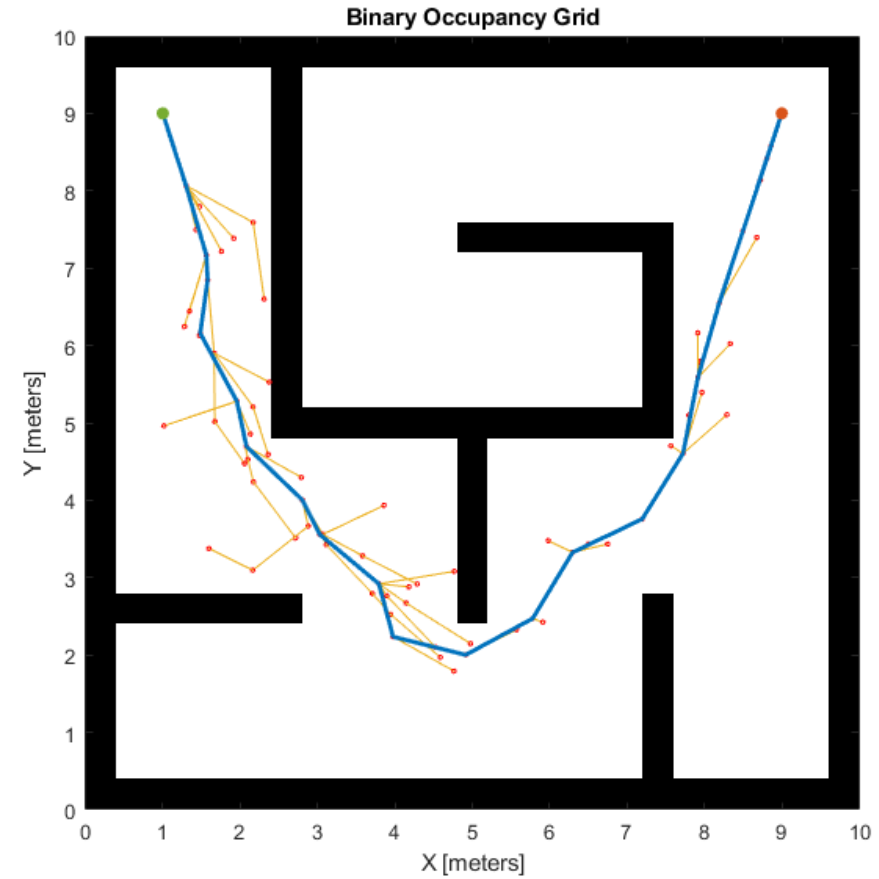


Mix both learned samples and uniform samples in a certain proportion λ , to bias the planner towards the optimal solution while also guaranteeing to find a solution

RRT* with Uniform & Learned Sampling



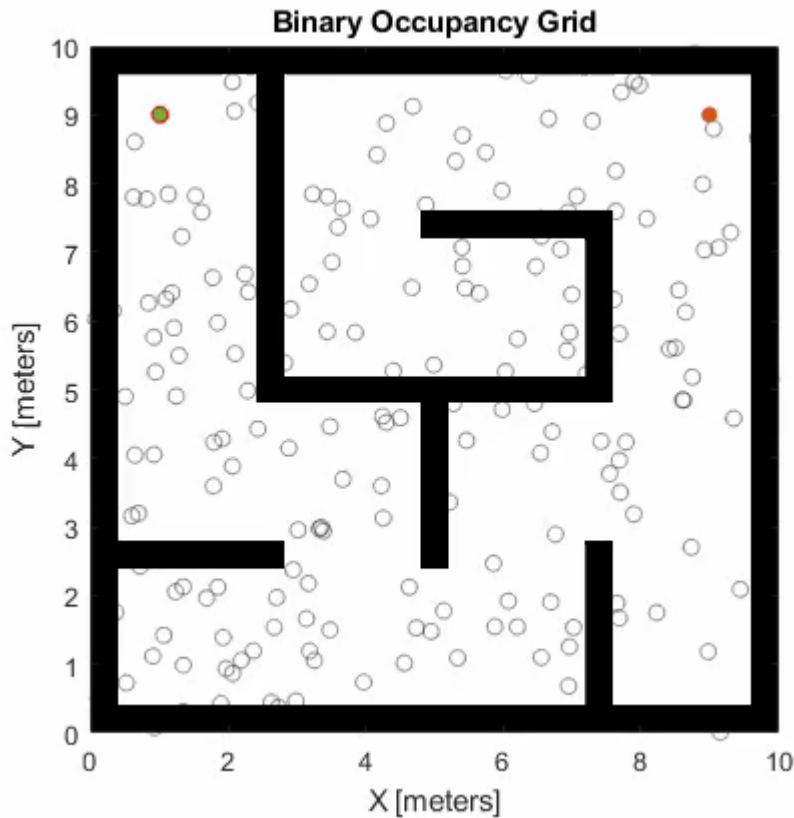
Uniform Sampling ($\lambda = 0$)



DL based Sampling ($\lambda = 0.9$)
90% from CVAE; 10% Uniform

Accelerate Motion Planning with Deep Learning

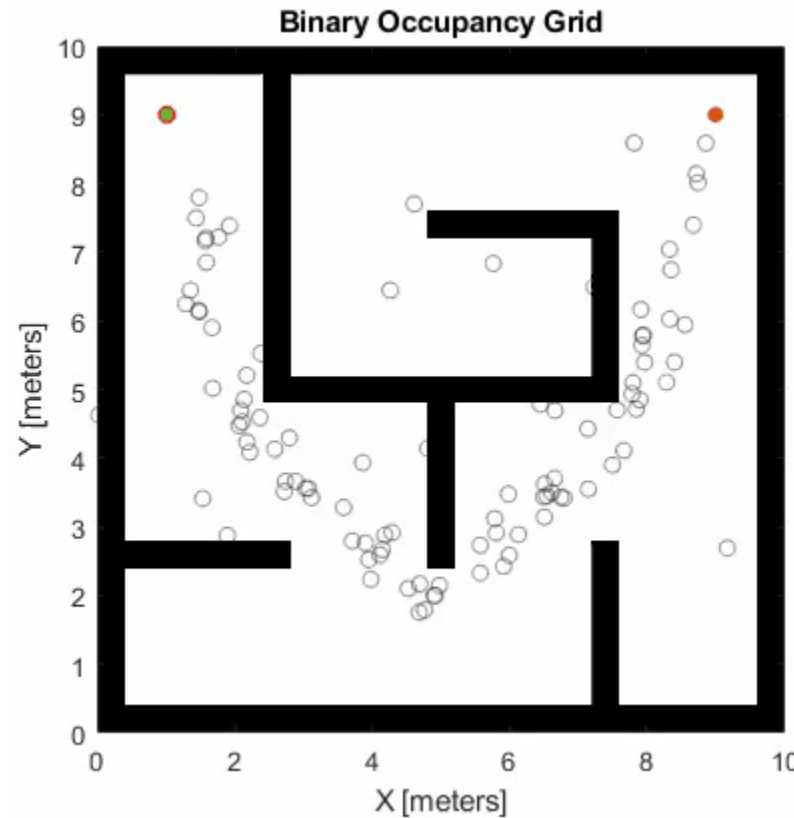
**RRT* Path
Uniform Sampling ($\lambda=0$)**



Elapsed time is 0.272630 seconds.

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

**RRT* Path
DL Based Sampling ($\lambda = 0.9$)**



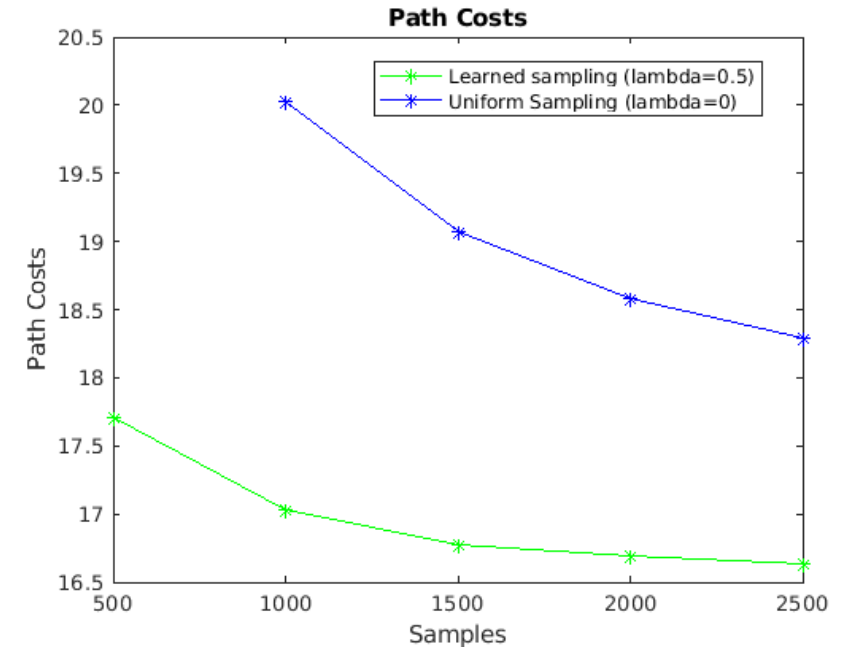
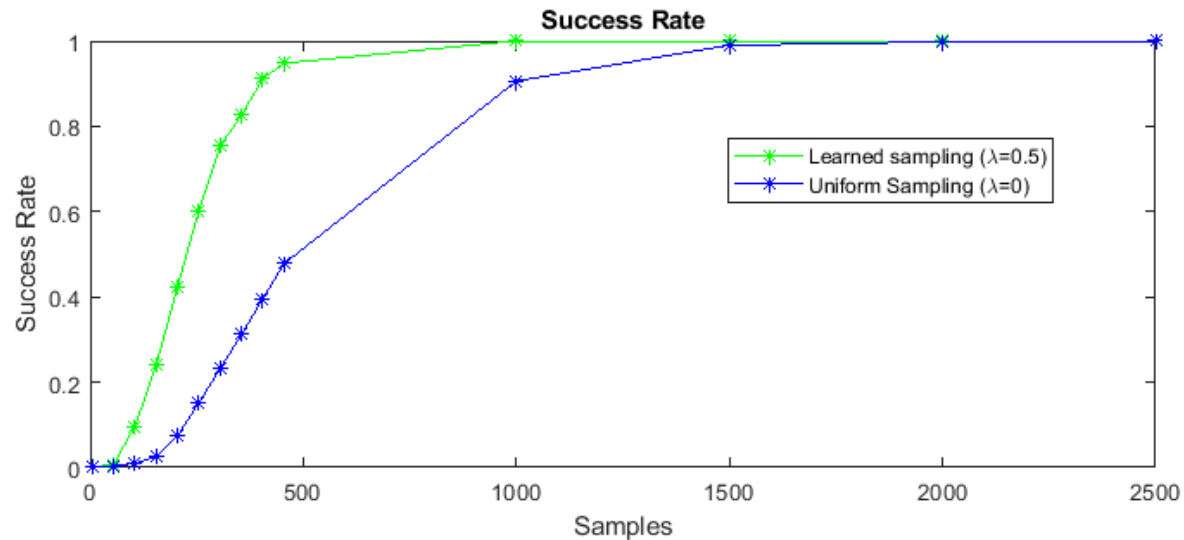
Elapsed time is 0.094654 seconds.

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



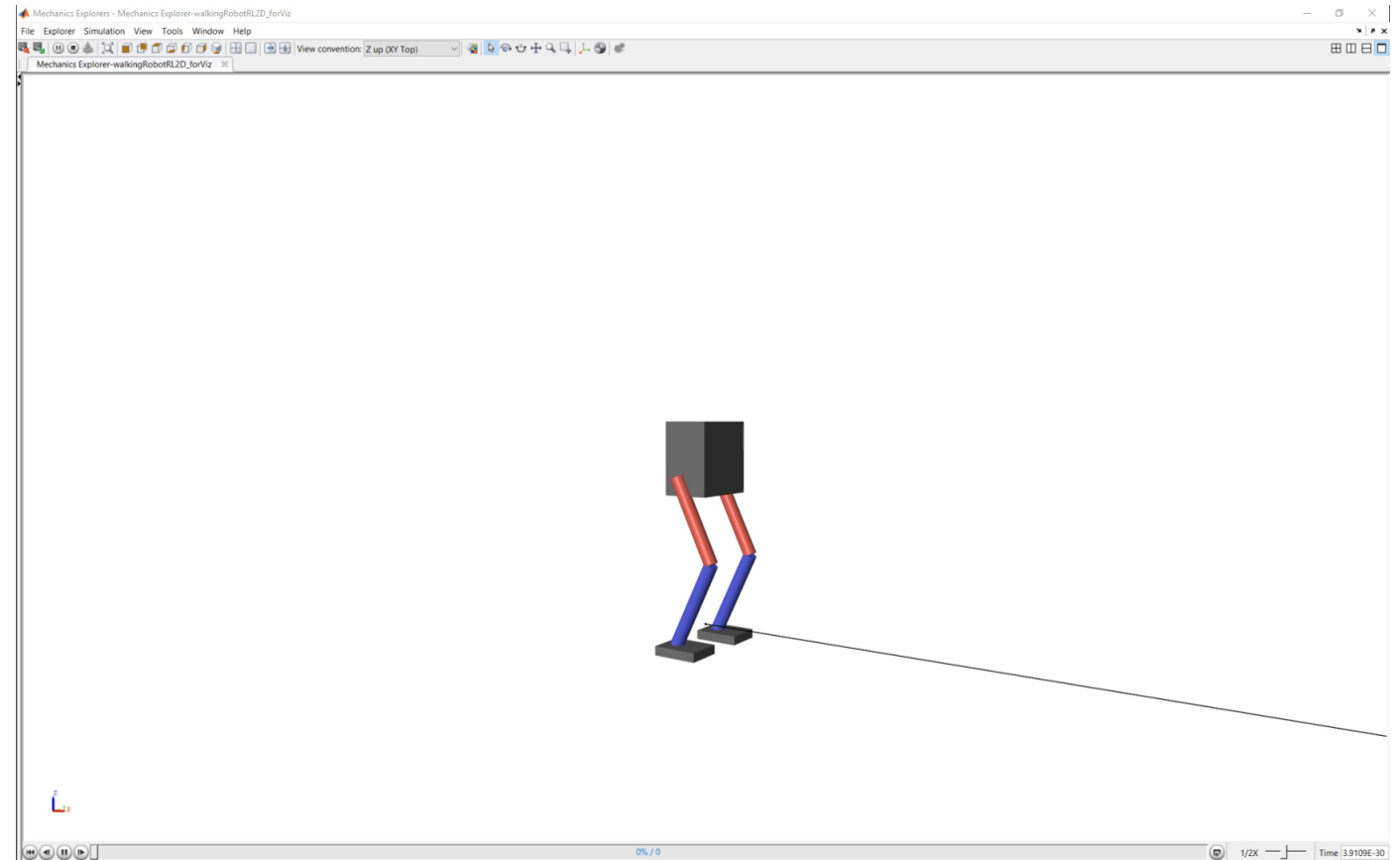
- **Faster Convergence to finding a valid path with Deep Learning based sampling**
- For 500 samples, Uniform sampling can't find a path for each map & each run
- Learned sampling path **cost function much better** than uniform sampling

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?

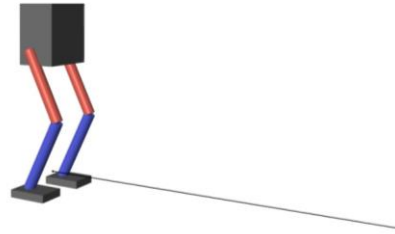
- What is Reinforcement Learning?
 - Type of machine learning that trains an **'agent'** through repeated interactions with an environment
- How does it work?
 - Through a trial & error process that uses a reward system to maximize success



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



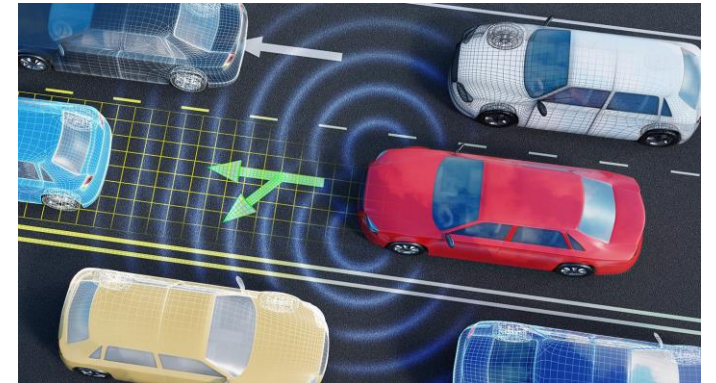
Controls



Robotics



A.I. Gameplay

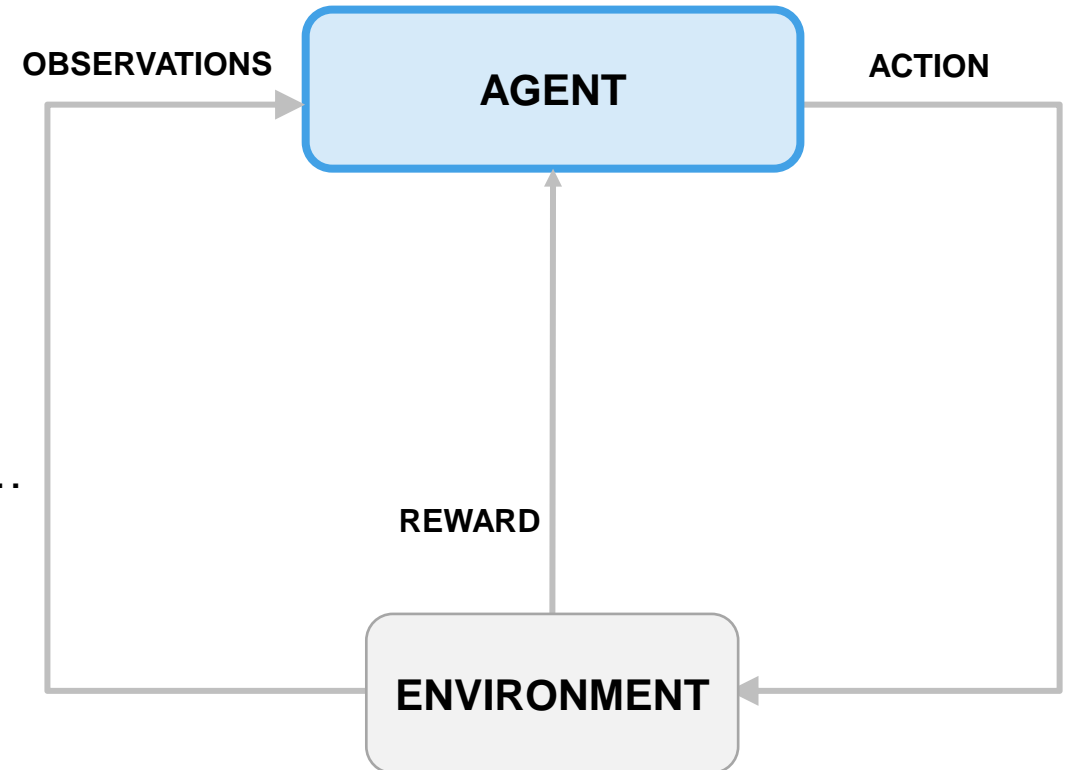


Autonomous driving

A Practical Example of Reinforcement Learning

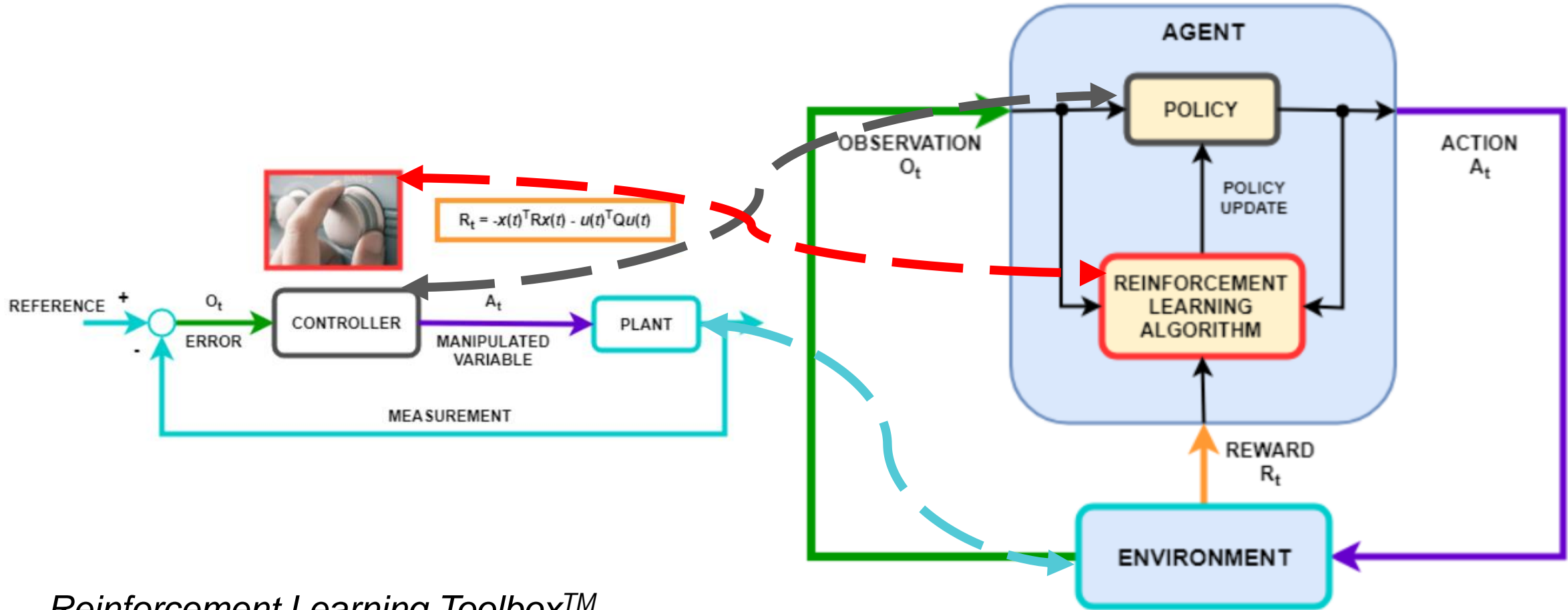
Training an Automated Parking Valet Controller

- 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**).



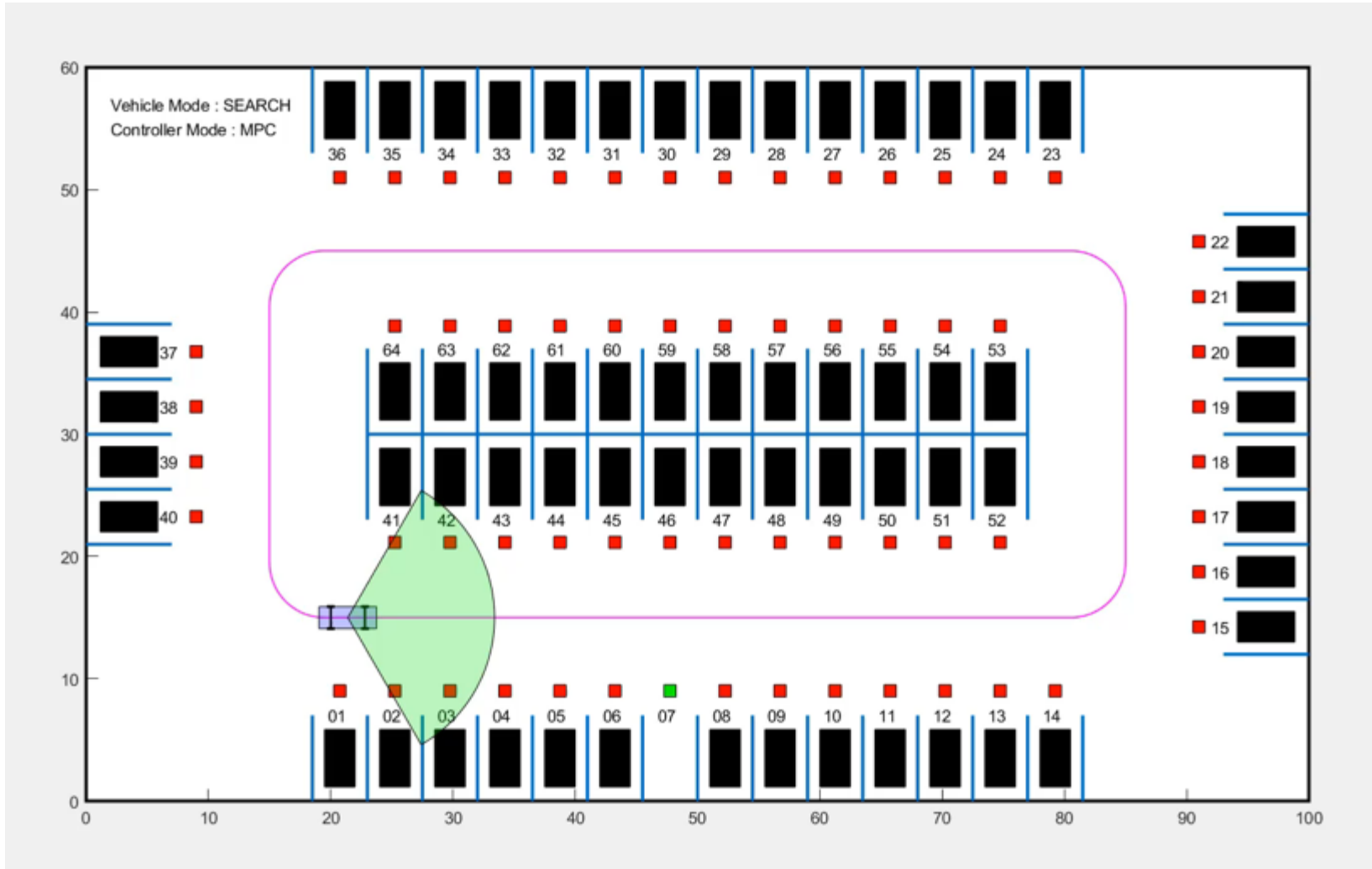
The goal of Reinforcement learning is for the agent to find an optimal algorithm for performing a task

Drawing Parallels- RL and Controls



Reinforcement Learning Toolbox™

Simulate trained agent for automatic parking

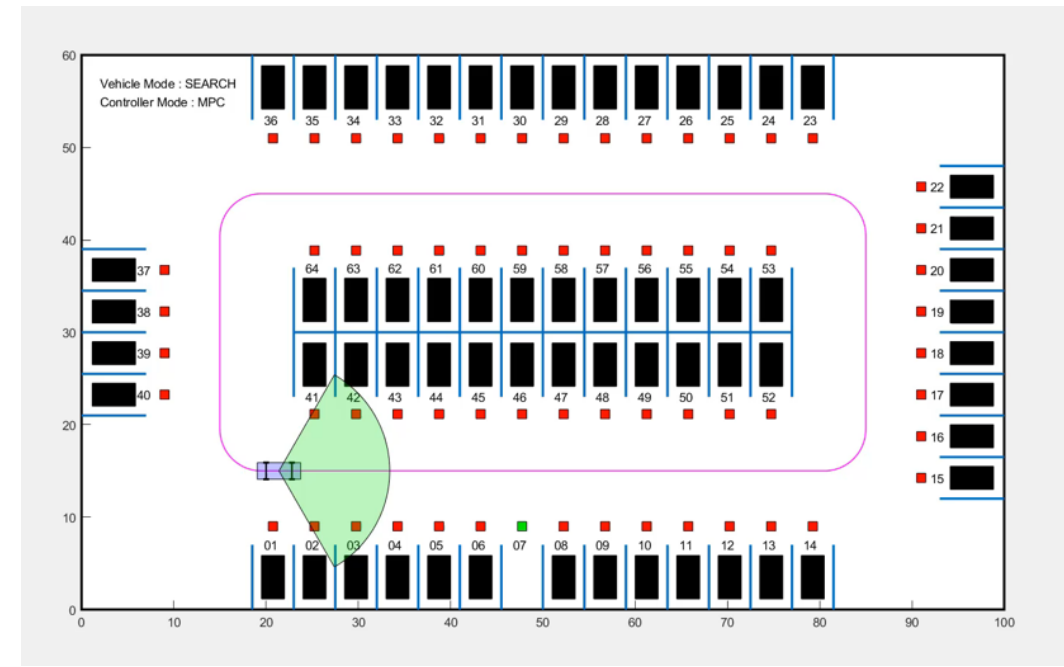
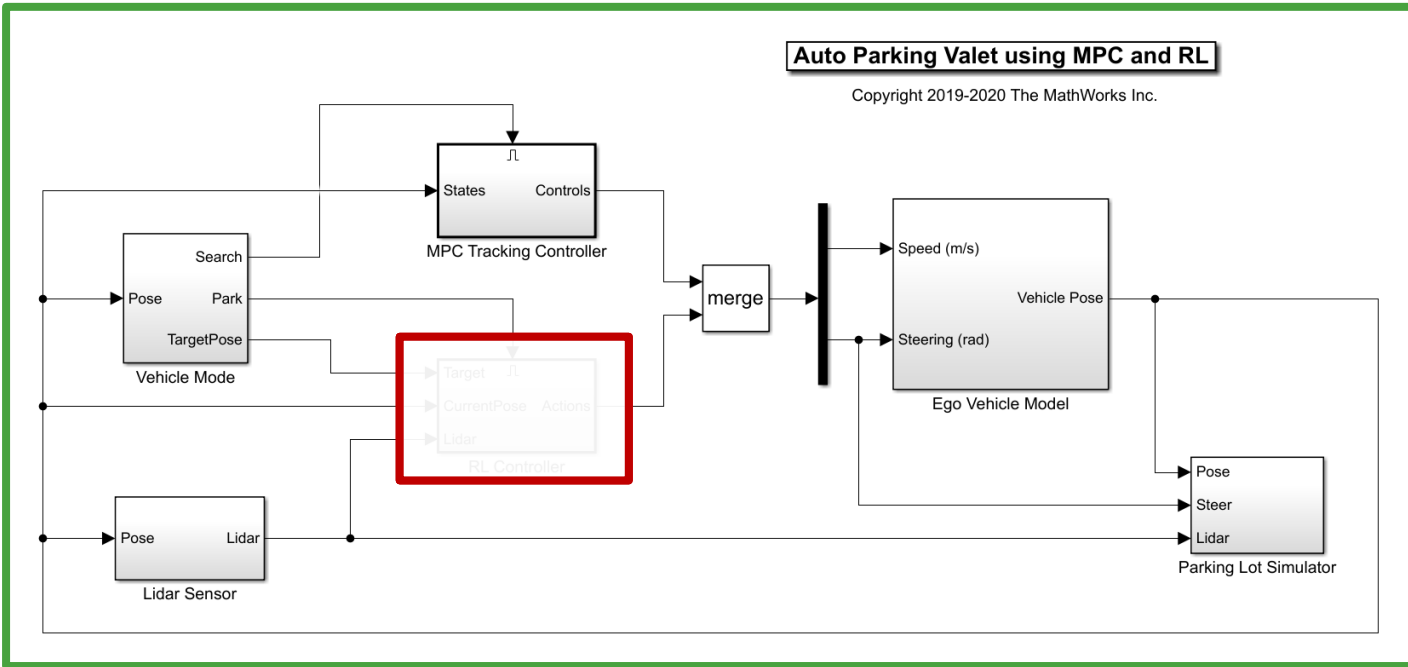


Train PPO Agent for Automatic Parking Valet

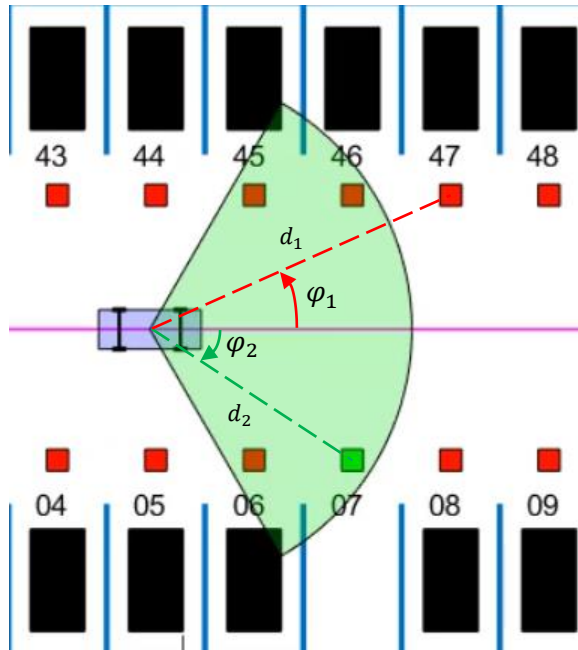
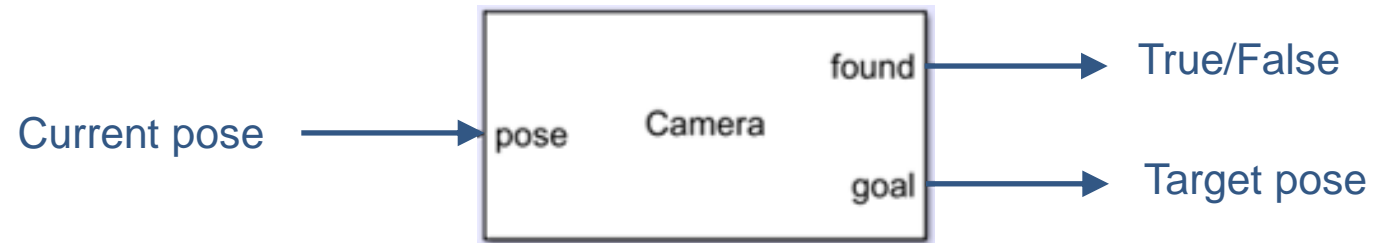
Train a reinforcement learning agent to park a car in an open parking space.



Simulink Model Bench for Parking Valet



Camera



Camera Parameters

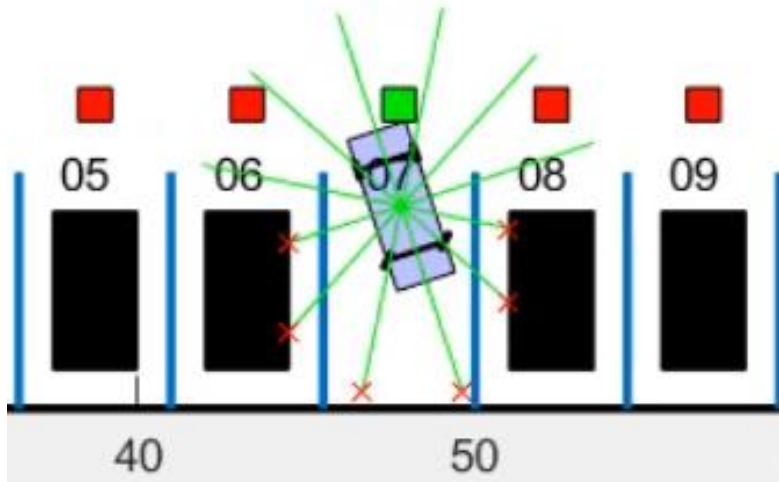
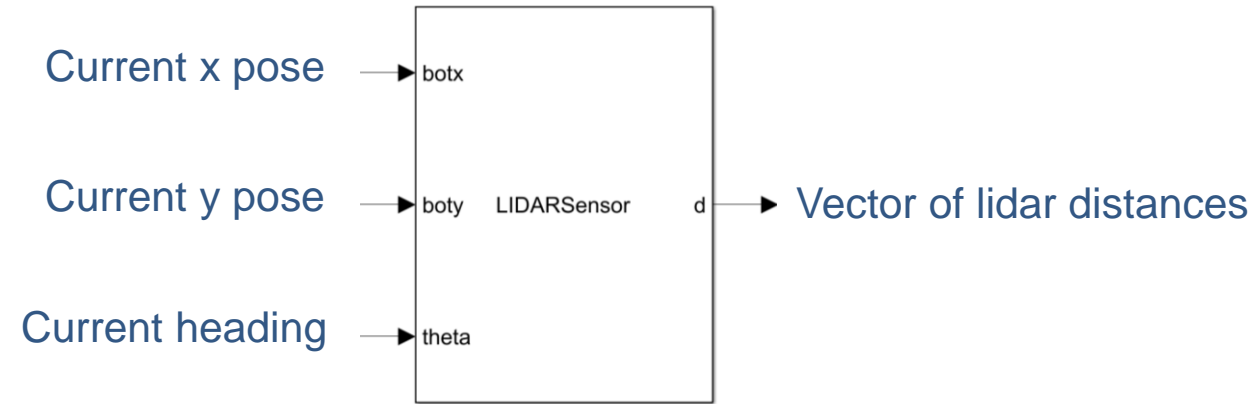
1. Depth (d_{max})
2. Field of view ($\varphi_{min}, \varphi_{max}$)

A spot is 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

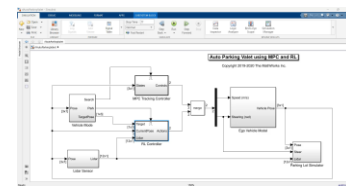
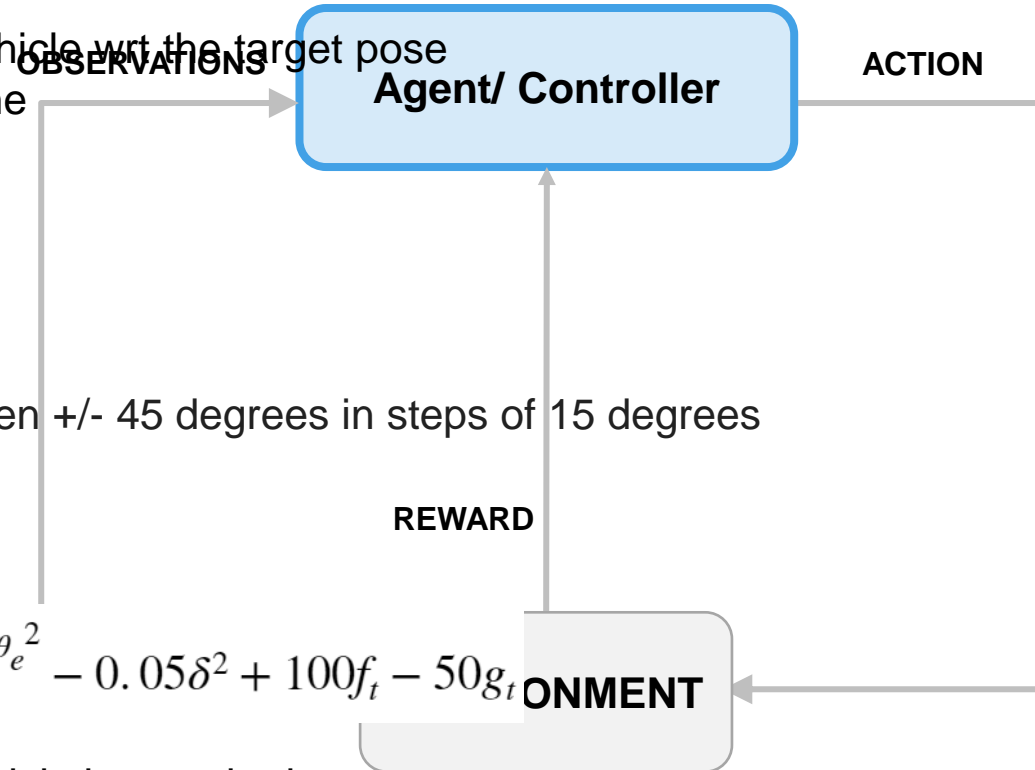
Reward:

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

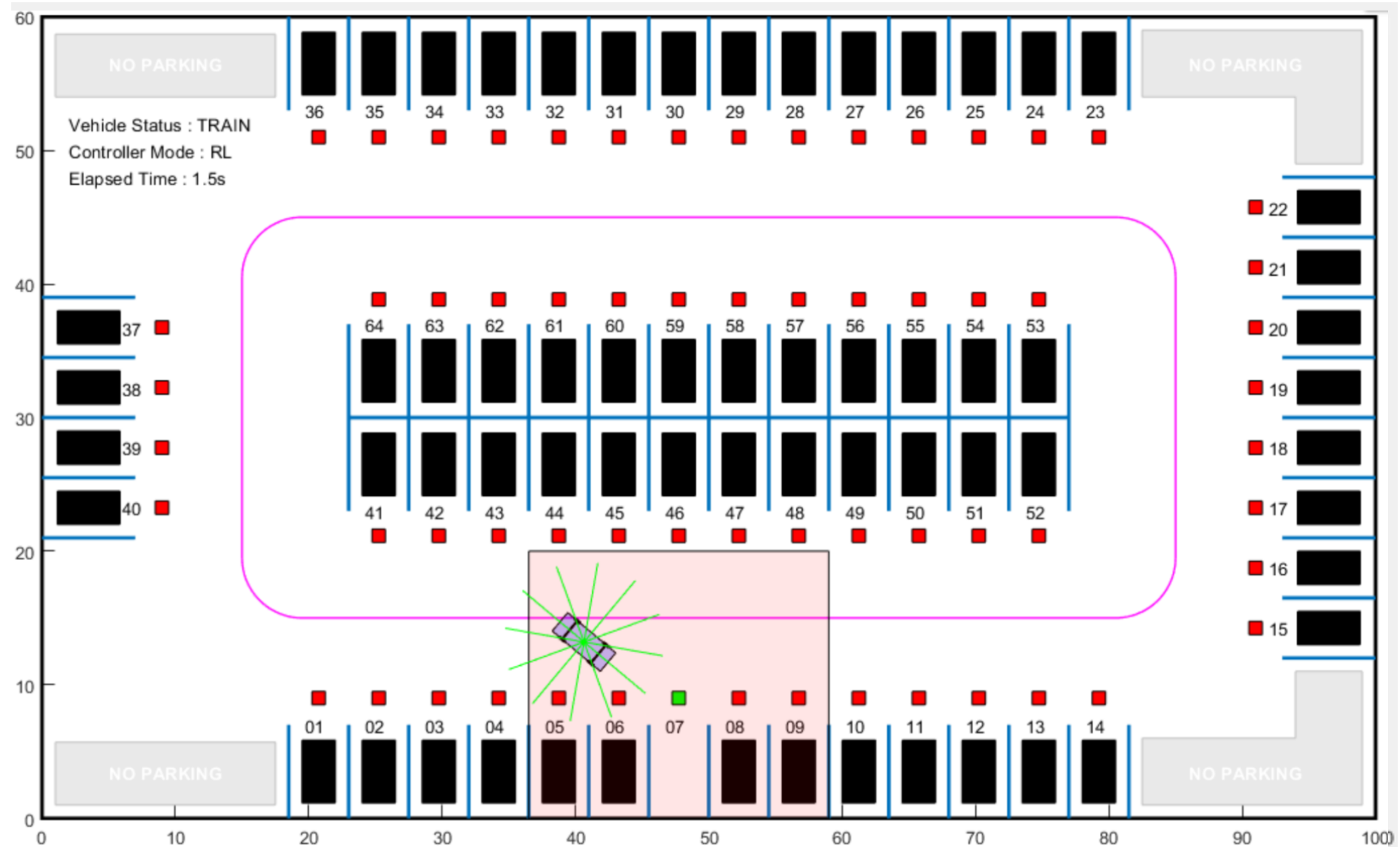
f_t (0 or 1) indicates whether the vehicle has parked

g_t Indicates collision

δ Steering angle



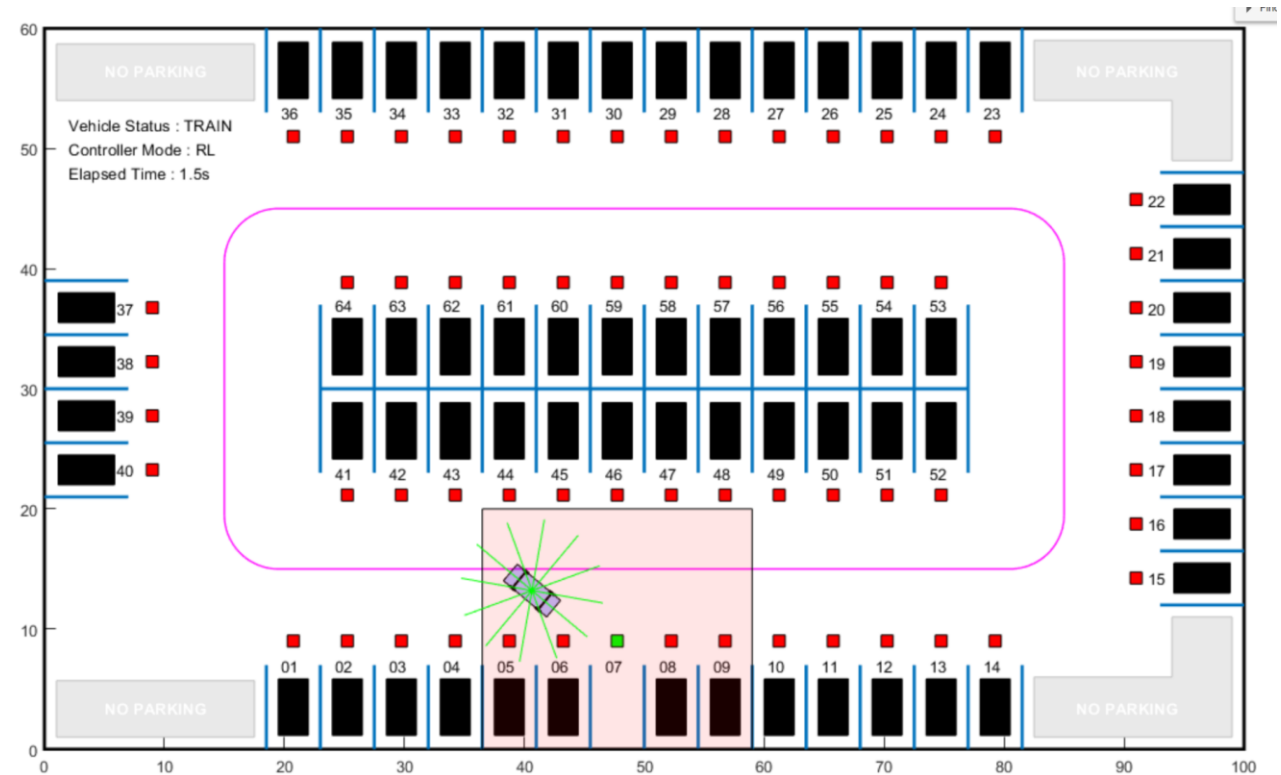
Environment



Mapping different parking locations

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

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



Choosing RL agent

Selection criteria:

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

Discrete action space
Discrete observation space

Q-learning

DQN

PPO

Discrete action space
Continuous observation space

DQN

PPO

- 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

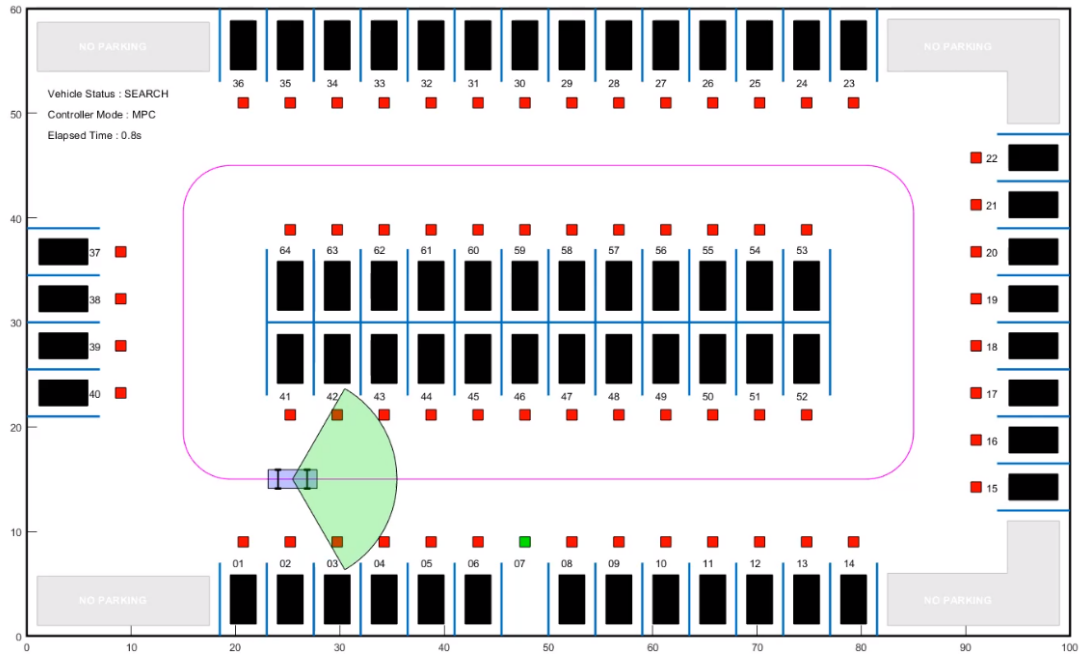
Built-in Reinforcement Learning Agents

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



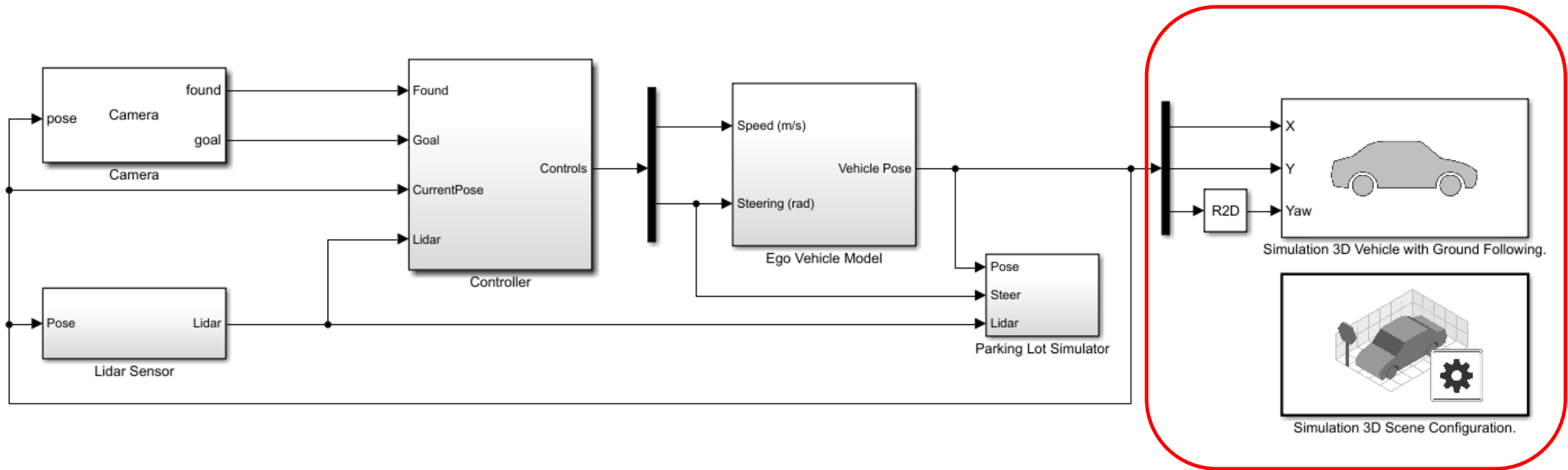
Scaling environment- Unreal Cosimulation



Train PPO Agent for Automatic Parking Valet Original Example

Unreal Engine – Large Parking Lot Scene

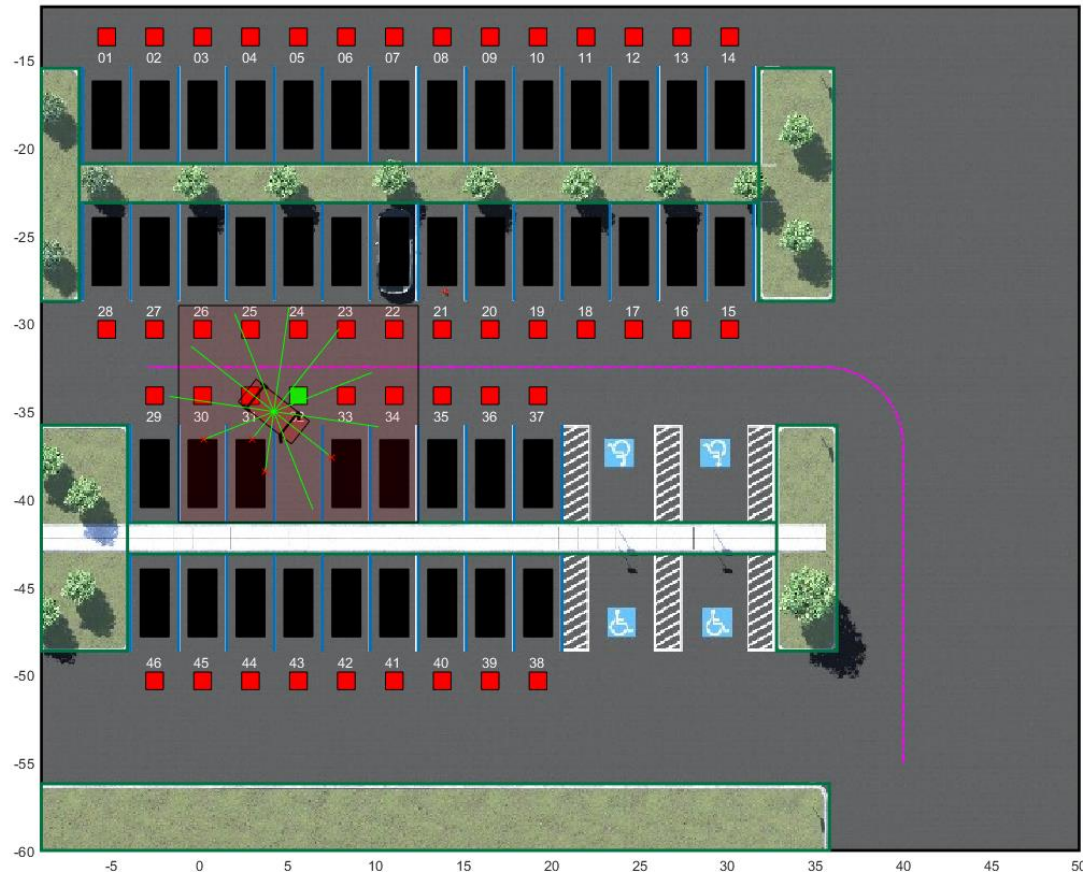
Incorporating 3D Simulation



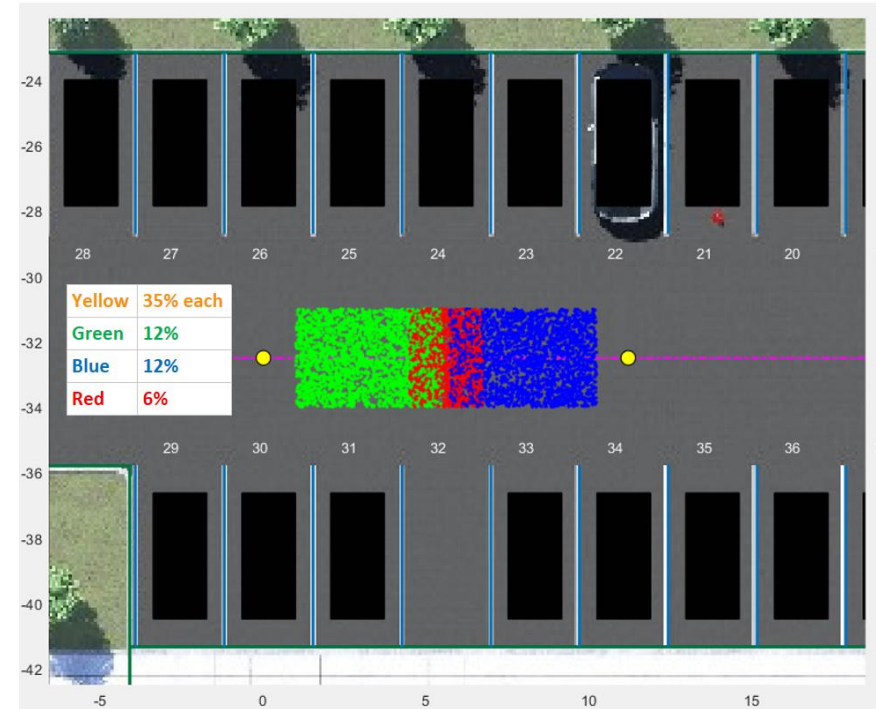
Reinforcement Learning Environment

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

Reward Function



Training Environment

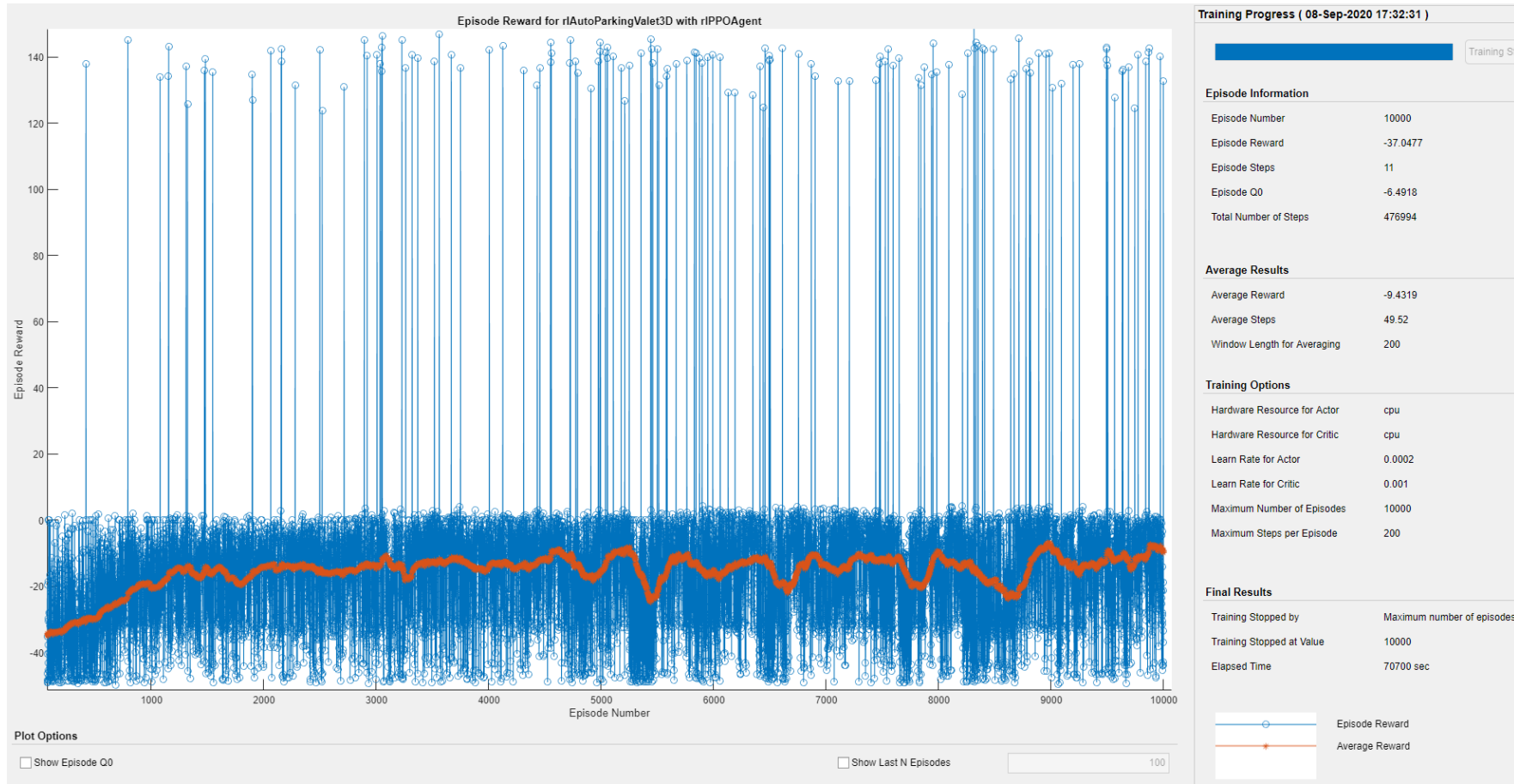


Vehicle Initial Pose during Training

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



PPO Agent Options

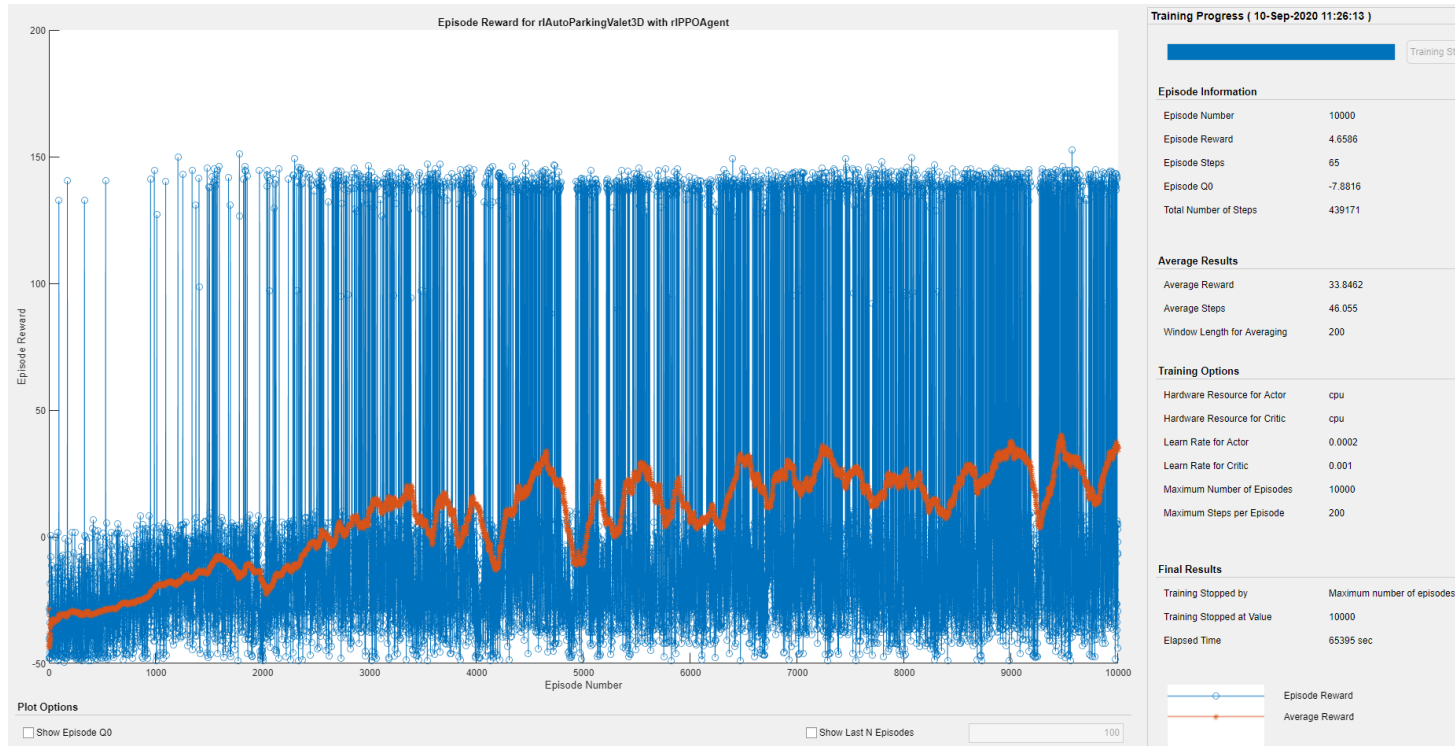
SampleTime	Ts
ExperienceHorizon	200
ClipFactor	0.2
EntropyLossWeight	0.01
MiniBatchSize	64
NumEpoch	3
AdvantageEstimateMethod	gae
GAEFactor	0.95
DiscountFactor	0.998

Training Options

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

Training from Scratch, Hyperparameters from Original Example

Training the Agent cont.

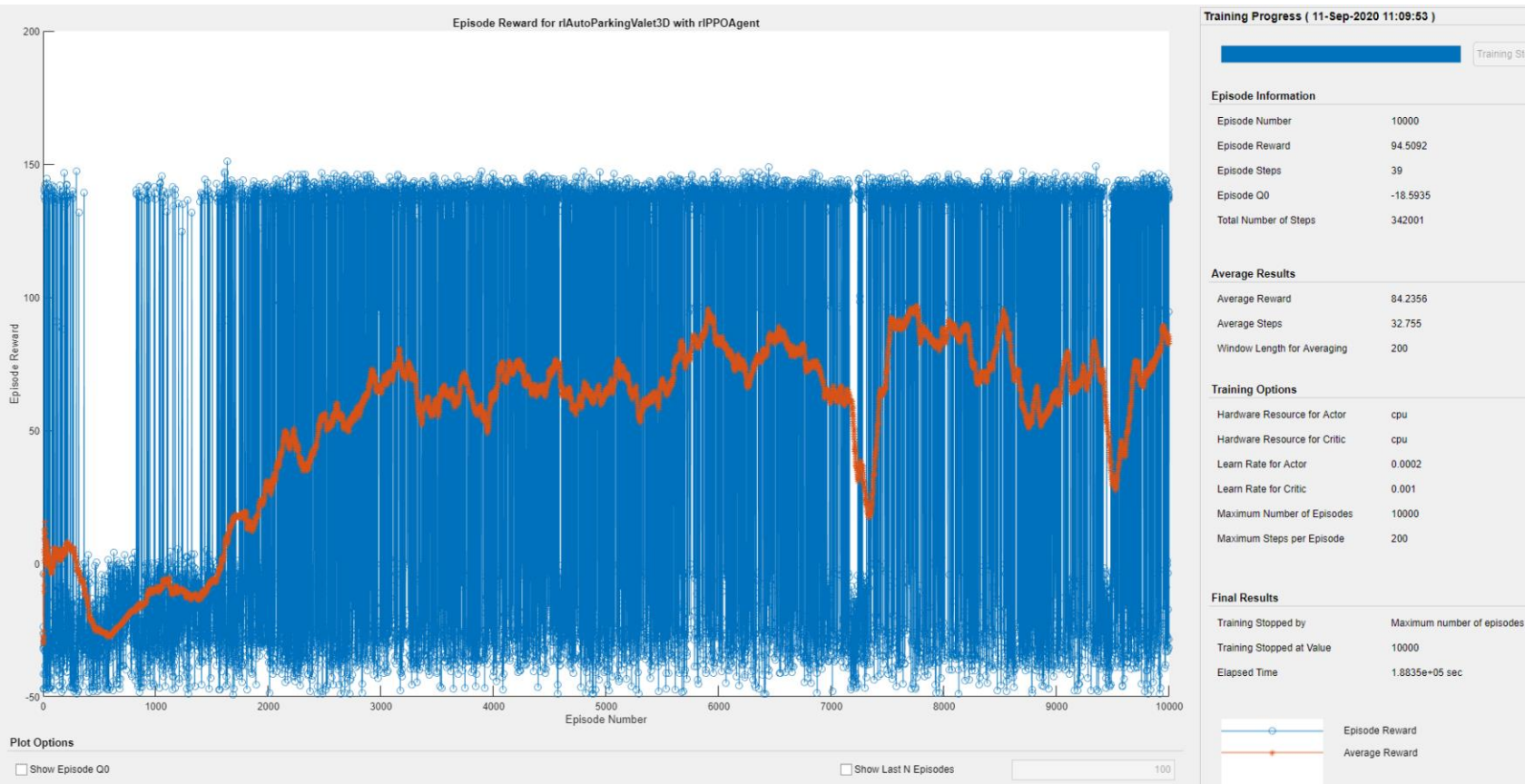


Training from Scratch, Changed Parked Vehicle Dimensions



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 2

Train PPO Agent for Automatic Parking Valet in 3D Environment

This example demonstrates the design of a hybrid controller for an automatic search and parking task. You will learn how to combine a Model Predictive Controller with a Reinforcement Learning Agent to perform a parking maneuver.

Overview

The automatic parking algorithm in this example executes a series of maneuvers while simultaneously sensing and avoiding obstacles in tight spaces. It switches between an adaptive MPC controller and an RL Agent to complete the parking maneuver. The MPC controller moves the vehicle at a constant speed along a reference path while an algorithm searches for an empty parking spot. When a spot is found, the RL Agent takes over and executes a pre-trained parking maneuver. Prior knowledge of the environment (the parking lot) including the locations of the empty spots and parked vehicles is available to the controllers.

Parking Lot

The parking lot is represented by the `ParkingLot` class, which stores information on the ego vehicle, empty parking spots and static obstacles (parked cars). Each parking spot has a unique index number and an indicator light that is either green (free) or red (occupied). Parked vehicles are represented in black.

The following sensor modules provide useful information to the parking algorithm:

1. A camera mounted on the ego vehicle with a field of view (range 120 deg, depth 10 m) represented by the area shaded in green. As the vehicle moves forward, the camera module senses the indicator lights that fall within the field of view and determines whether the spots are free or occupied. For simplicity, this is implemented using geometrical relationships between the spot locations and the vehicle pose.
2. A lidar sensor module that determines proximity to obstacles through a set of 12 distances from the center of the vehicle.

Specify a sample time T_s for the controllers and a simulation time T_f .

```

1 Ts = 0.1;
2 Tf = 45;

```

Create a reference path for the ego vehicle to follow in the parking lot.

```

3 Xref = getRefTraj(Ts,Tf);

```

Create a `ParkingLot3D` object with a free spot at location 32.

Command Window

```

Prerelease License -- for engineering feedback and testing
purposes only. Not for sale.

- TS Toolbox successfully installed
fx>>

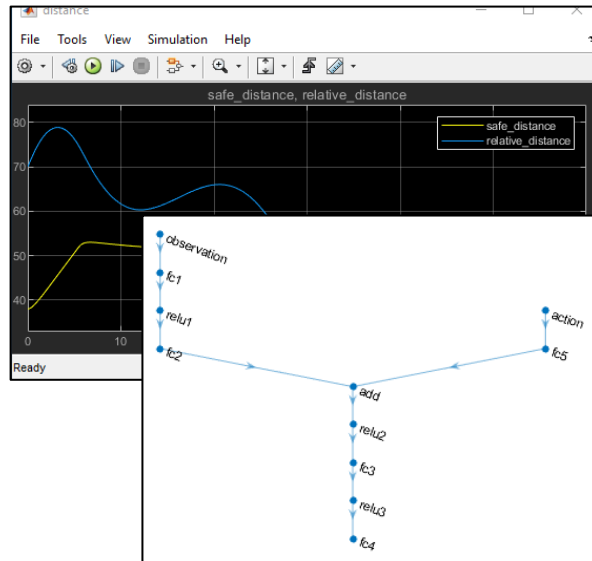
```

Workspace

Name	Value
action...	1x1 rIFinit...
actor...	1x1 rStoc...
actorN...	7x1 Layer
actorO...	1x1 rRepr...
Ad	[1,0,0,0,1,...
agent	1x1 rPPO...
agent...	1x1 rPPO...
Bd	[6.1232e-...
blk	'IAutoPar...
camer...	10
camer...	2.0944
Cd	[1,0,0,0,1,...
center...	1.3430
critic	1x1 rValu...
criticN...	8x1 Layer
criticO...	1x1 rRepr...
Dd	[0,0,0,0,0]
discret...	[-0.7854,-...
doTrai...	0
dsys	3x2 ss
DX0	[0,0,0]
egolnit...	[40,-55,1...
egoTar...	[5.6125,-3...
env	1x1 Simul...
freeSp...	32
lidarTol	0.5000
logout	1x1 Datas...
map	1x1 Parki...
maxLid...	6
mdl	'IAutoPar...
mpcobj	2x3 mpc
nAct	7
nObs	16
numSe...	12
observ...	1x1 rNu...
obsMat	56x5 dou...
pTrack...	10
speed...	2
steerM...	0.7854
tBounds	[-Inf,Inf]
terrTol	0.1745
Tf	45
tout	474x1 do...
trainO...	1x1 rTrai...
trainTB...	[-6.2832,6...
trainX...	[-1.2000,1...
trainYB...	[-41.3400...
Ts	0.1000
u	[0,0]
U0	[0,0]
x	[40,-55,1...

Design reinforcement learning agents for controls

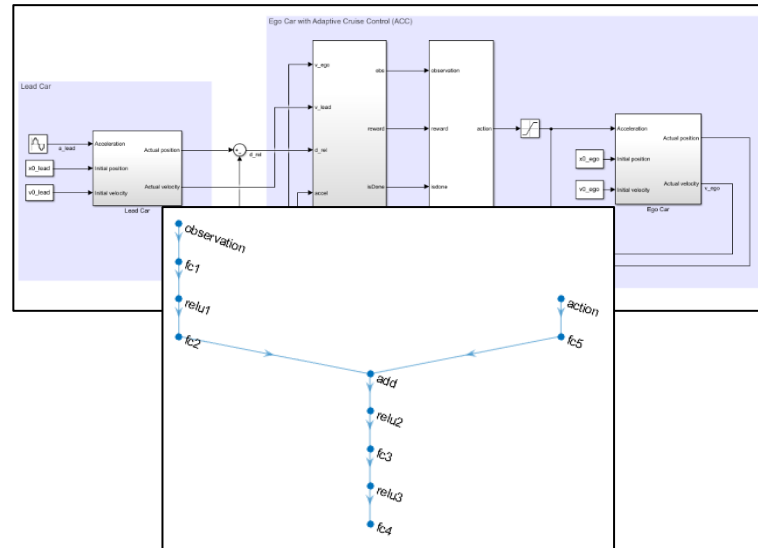
DDPG Agent



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

Reinforcement Learning Toolbox™

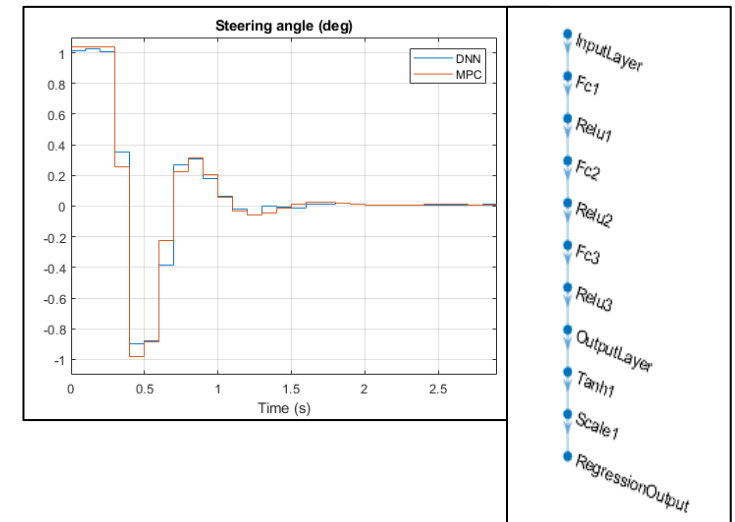
DDPG Agent



[Train DDPG Agent for Path Following Control](#)

Reinforcement Learning Toolbox™

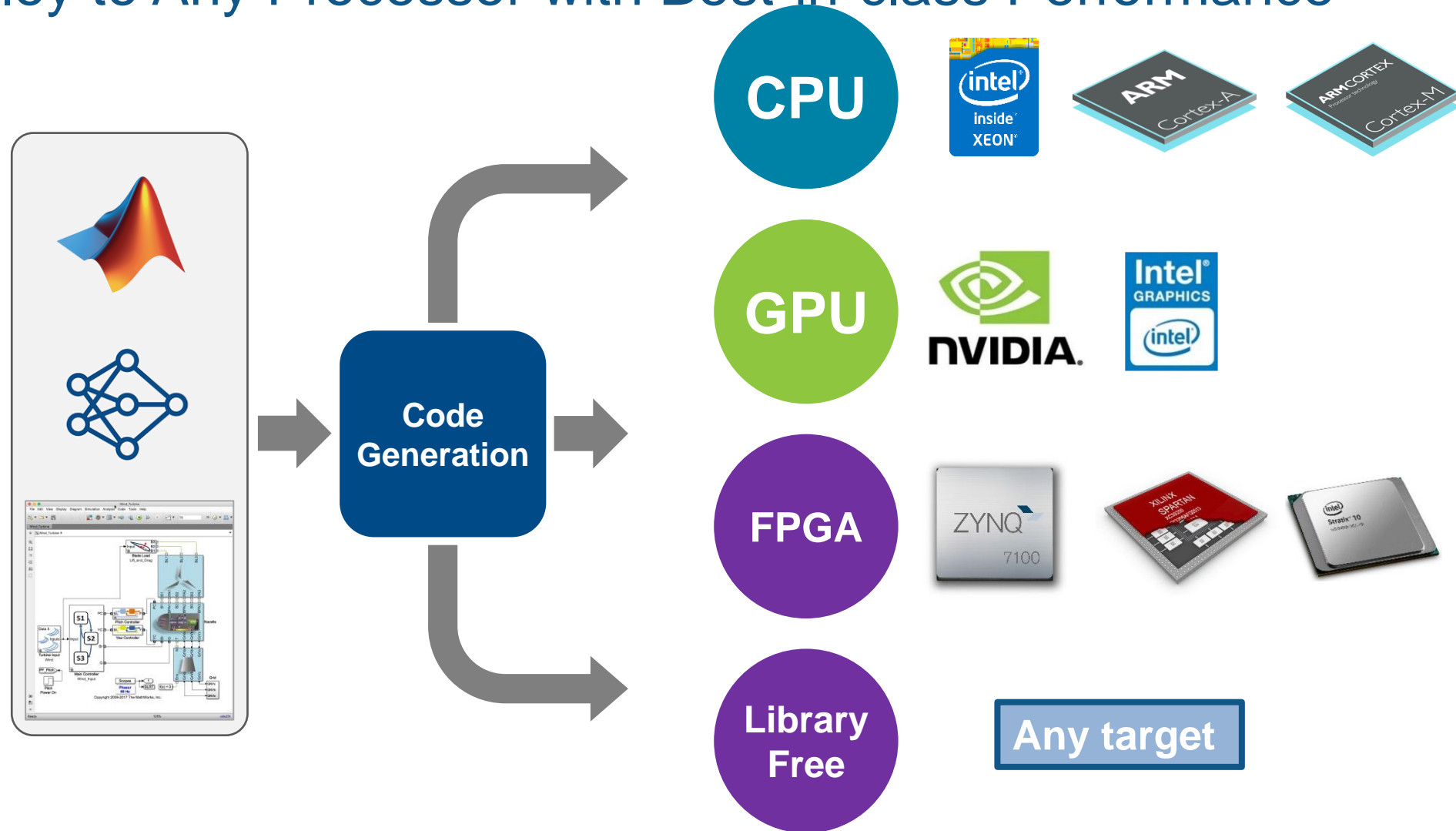
Neural Network



[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

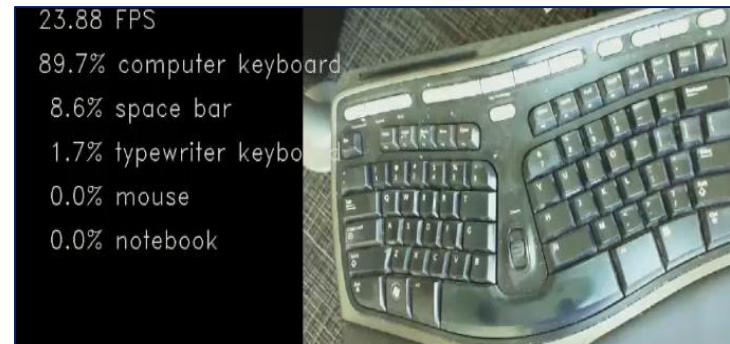
- Need code that takes advantage of:
 - NVIDIA® CUDA libraries, including cuDNN and TensorRT
 - Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN) for Intel processors
 - ARM® Compute library for ARM processors



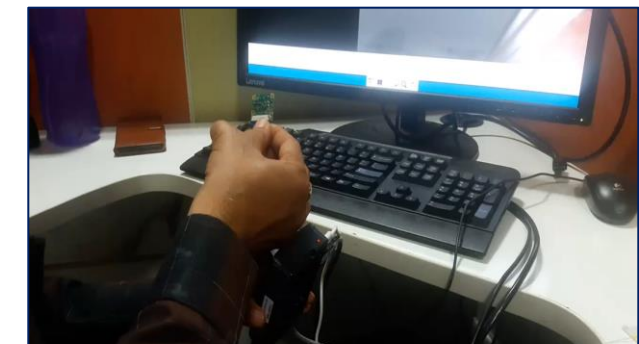
NVIDIA Jetson TX1 board



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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Secure data exchange with smart industrial plant sensors and servers • Develop predictive maintenance, smart manufacturing, and SCADA applications
	Hardware Deployment of AI Models	<ul style="list-style-type: none"> • 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



© 2023 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.