Aerial Robotic Autonomy

> *Uncertainty-aware Navigation using Deep Neural Networks*

Huan Nguyen, Kostas Alexis
Autonomous Robots Lab

Norwegian University of Science and Technology

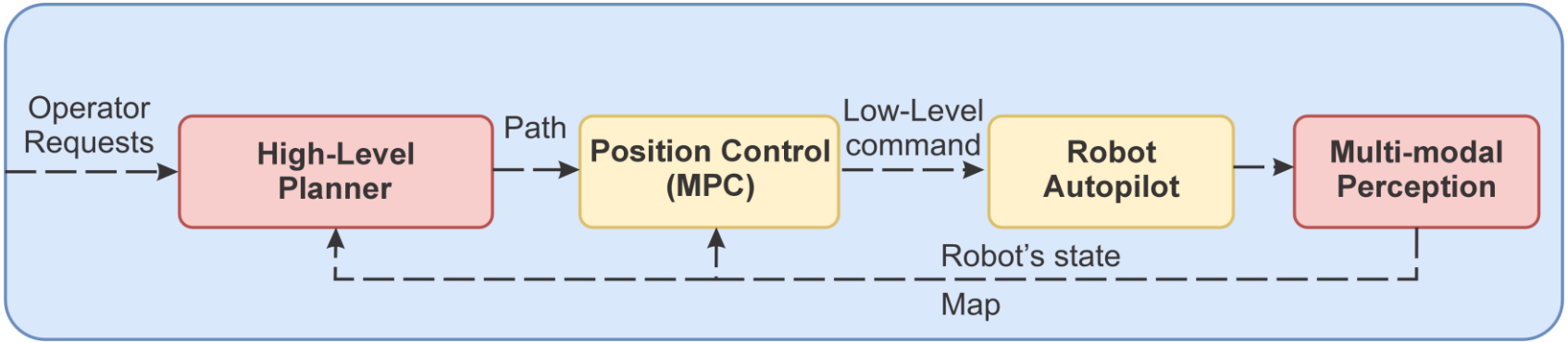
Introduction

Robotic operations take place in a diverse set of challenging environments.

Key challenges:

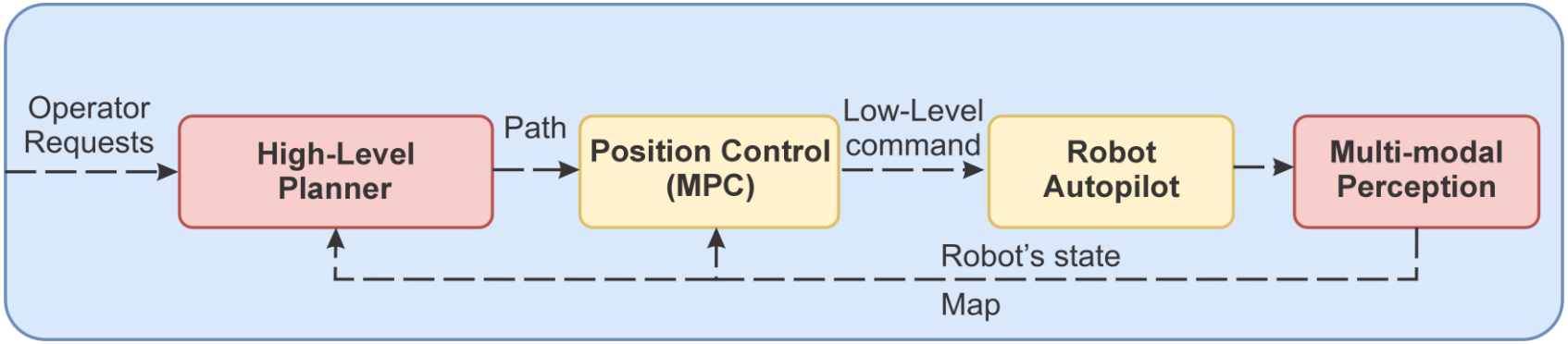
- Limited communication
- Perception degradation
- Noisy state estimate/sensor observations
- Hard-to-detect obstacles

Conventional Autonomy Stack



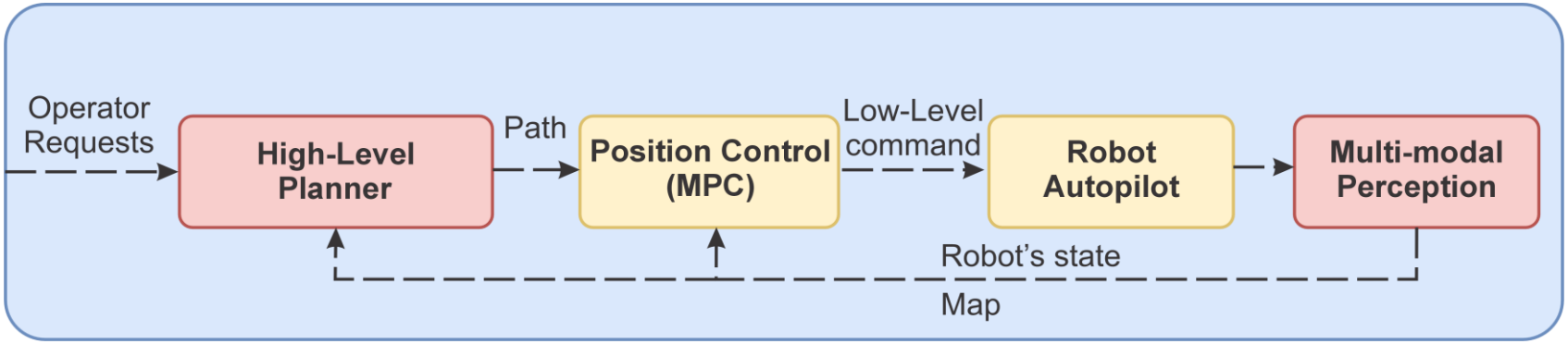
Requires position estimates and map to be accurate

Conventional Autonomy Stack



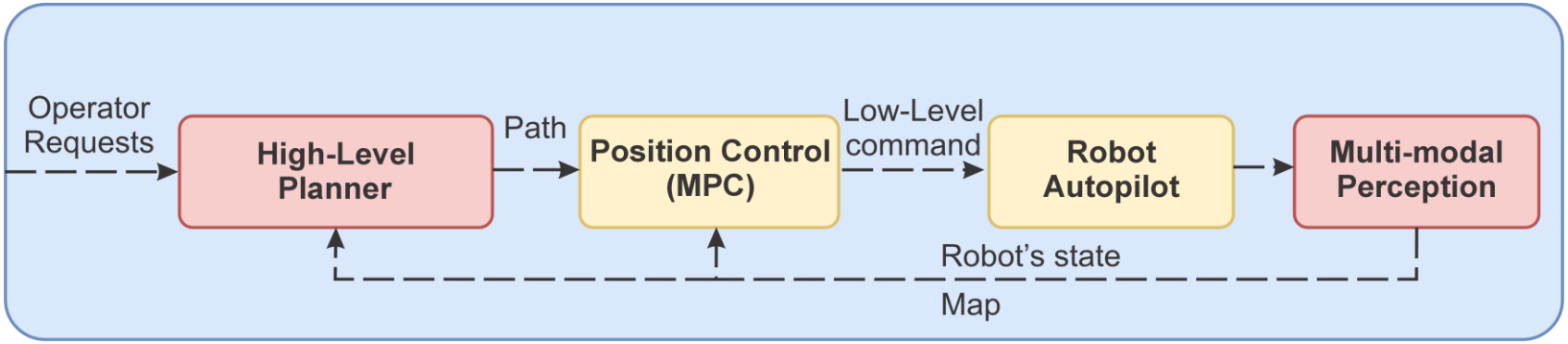
- SLAM can **fail/drift** significantly
- Mapping can require high **computational** cost

Conventional Autonomy Stack



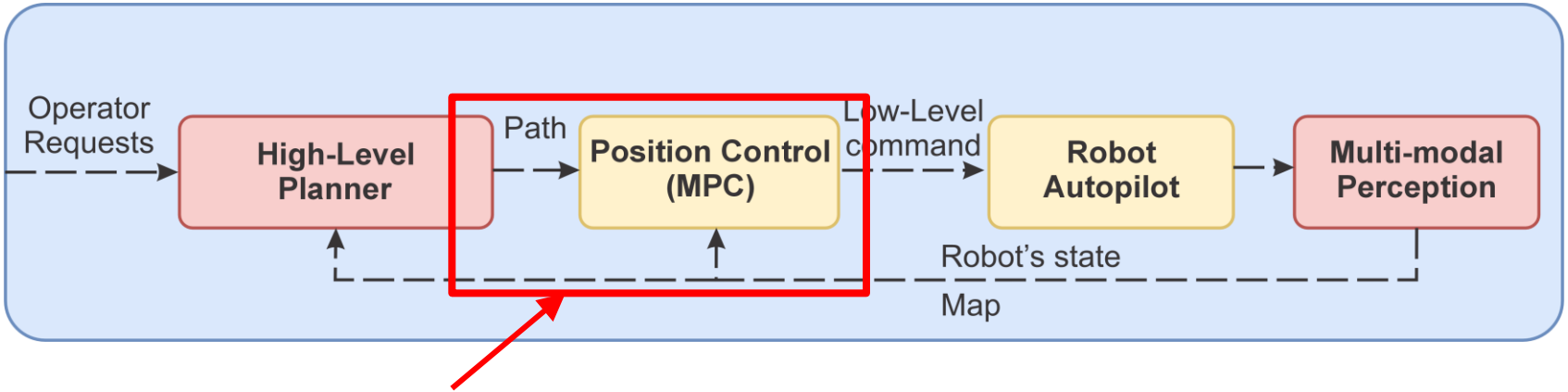
- SLAM can **fail/drift** significantly
- Mapping can require high **computational** cost → **speed limit**

Conventional Autonomy Stack



- SLAM can **fail/drift** significantly
- Mapping can require high **computational** cost → **speed limit**
- **Noisy + high-dimensional** exteroceptive data
- **Uncertain** robot's state estimate

Conventional Autonomy Stack



Resilient learning-based navigation methods

- "Last-resort" policy when SLAM fails/drifts **(Redundancy)**
- A safety layer – utilizing same inputs as mapping-planning modules **(Resourcefulness)**
- Resilient to noise **(Robustness)**

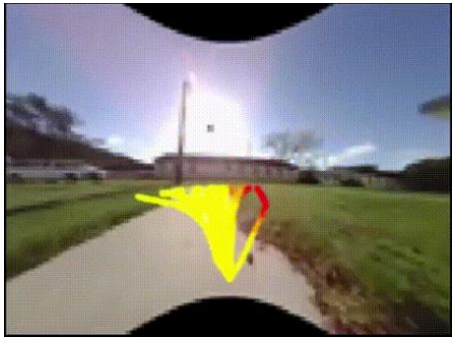
Related Work

Agile [2]

Dagger imitation learning

Require position tracking controller

Robustness to noise (up to some levels)

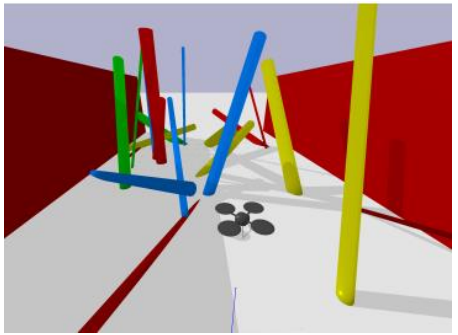


BADGR [3]

Score a Motion Primitive Libraries

No position feedback

No uncertainty consideration



PAC-Bayes [4]

Score a Motion Primitive Libraries

Require position tracking controller

No uncertainty consideration

Only in simulation

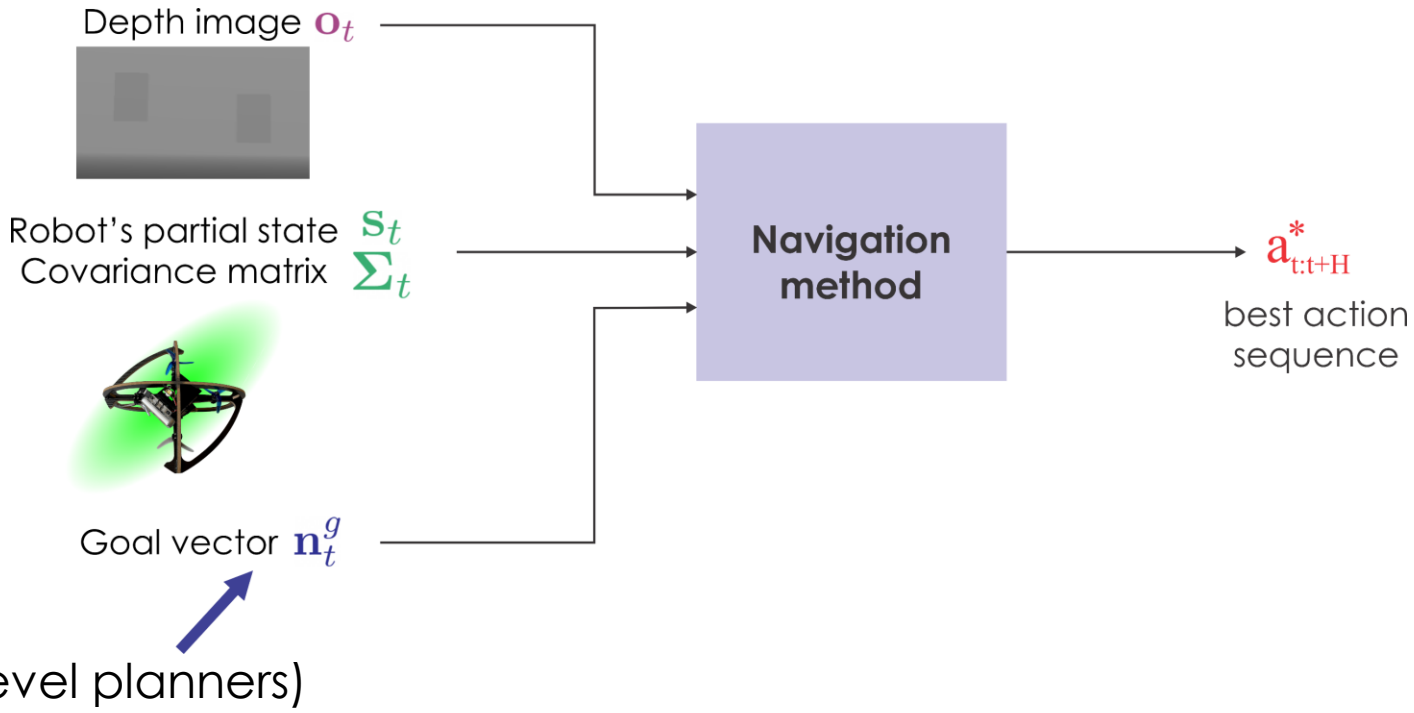
ORACLE (ours) [1]

Score a Motion Primitive Libraries

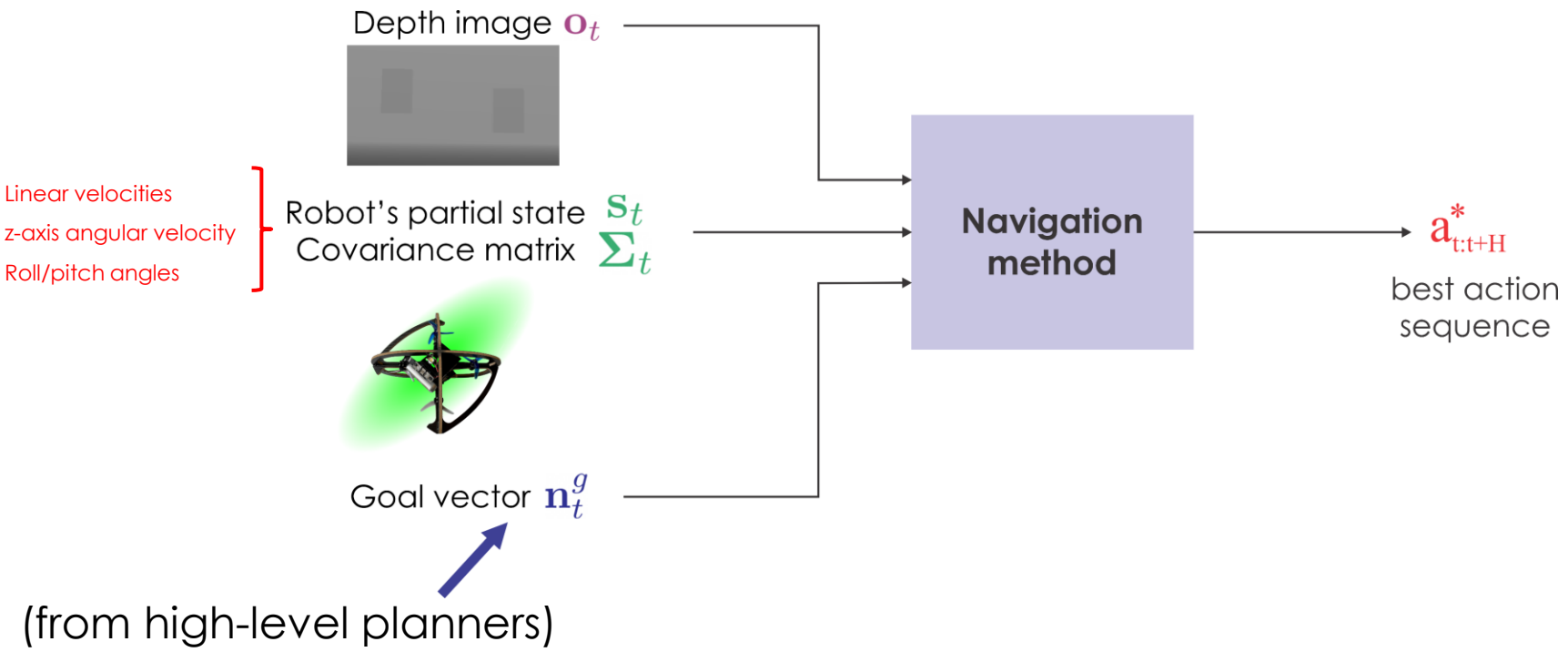
No position feedback

Uncertainty-aware

Problem Formulation



Problem Formulation



Approach

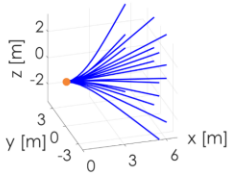
ORACLE

Depth image $\mathbf{o}_t$

Robot's partial state

$\mathbf{m}_1 = \mathbf{s}_t$

Motion primitives library (MPL)



$\mathbf{a}^1_{t:t+H}$

$\mathbf{a}^2_{t:t+H}$

$\vdots$

$\mathbf{a}^{N_{MP}}_{t:t+H}$

$\mathbf{n}^g_t$

Goal vector

$\mathbf{a}^*_{t:t+H}$

best action sequence

Approach

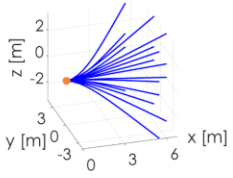
ORACLE

Depth image $\mathbf{o}_t$

Robot's partial state

$\mathbf{m}_1 = \mathbf{s}_t$

Motion primitives library (MPL)



$\mathbf{a}^1_{t:t+H}$

$\mathbf{a}^2_{t:t+H}$

$\mathbf{a}^{N_{MP}}_{t:t+H}$

$\mathbf{a}_{t:t+H} = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}]$

$\mathbf{a}_{t+i} = [(\mathbf{v}^r_{t+i})^T, \delta^r_{t+i}]^T$

3D velocity, steering angle

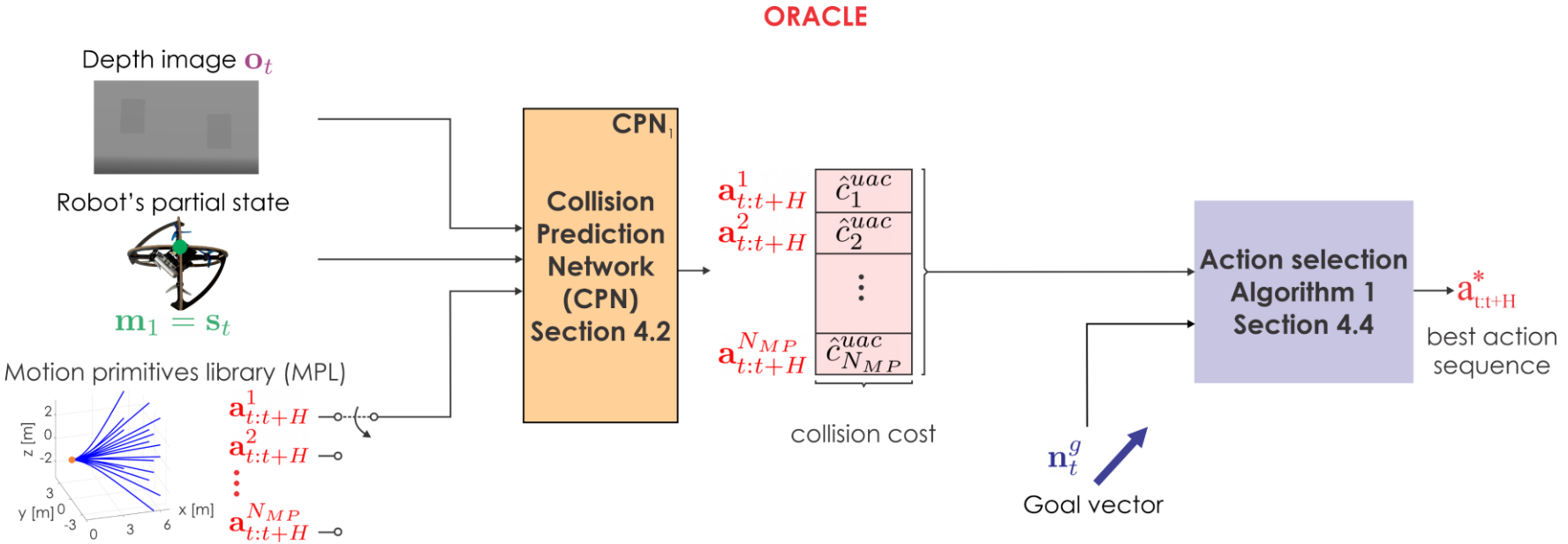
$\mathbf{n}^g_t$

Goal vector

$\mathbf{a}^*_{t:t+H}$

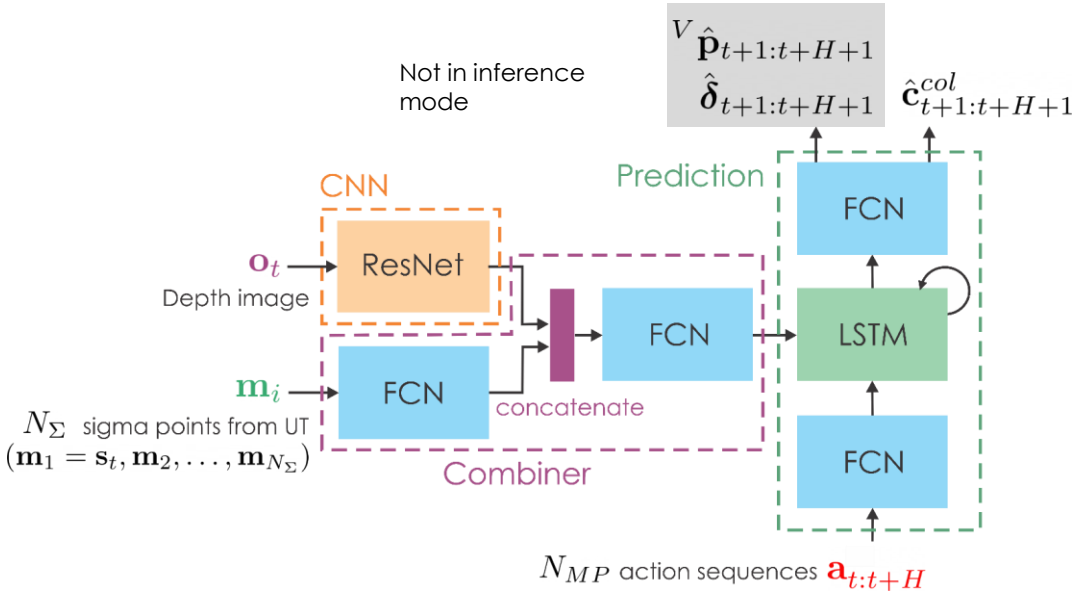
best action sequence

Approach

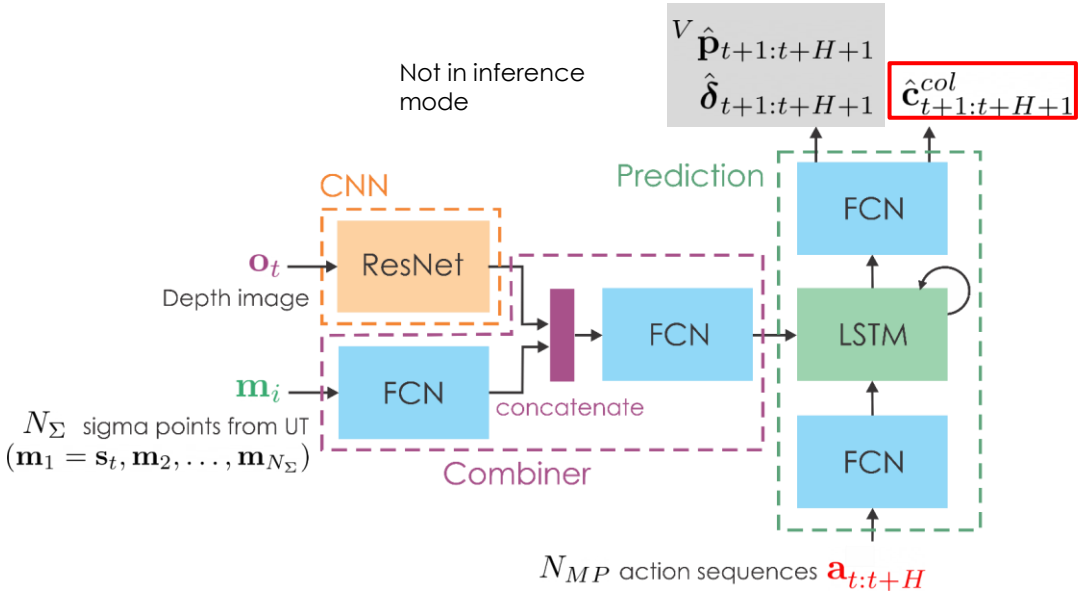


NTNU

CPN Architecture & Training Losses



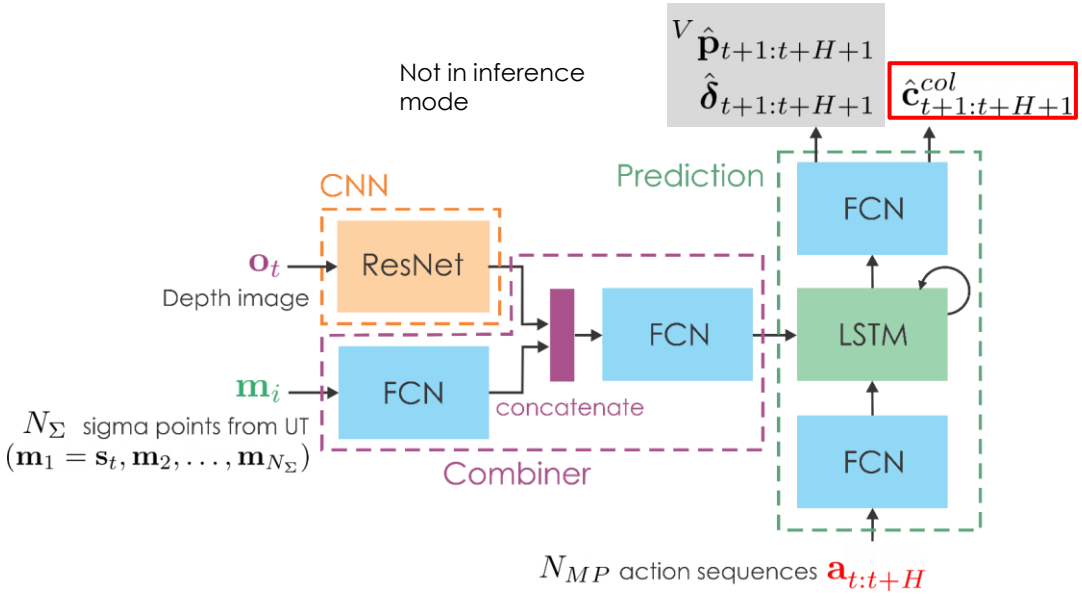
CPN Architecture & Training Losses



Binary Cross Entropy for collision prediction

$$L_{CPN} = \alpha_1 L_{BCE}(\mathbf{c}^{col}_{t+1:t+H+1}, \hat{\mathbf{c}}^{col}_{t+1:t+H+1}) + \alpha_2 L_{MSE}(^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, ^{\mathcal{V}}\hat{\mathbf{p}}_{t+1:t+H+1}) + \alpha_3 L_{MSE}(\boldsymbol{\delta}_{t+1:t+H+1}, \hat{\boldsymbol{\delta}}_{t+1:t+H+1})$$

CPN Architecture & Training Losses



$$L_{CPN} = \alpha_1 L_{BCE}(\mathbf{c}^{col}_{t+1:t+H+1}, \hat{\mathbf{c}}^{col}_{t+1:t+H+1}) + \alpha_2 L_{MSE}(^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, {}^{\mathcal{V}}\hat{\mathbf{p}}_{t+1:t+H+1}) + \alpha_3 L_{MSE}(\boldsymbol{\delta}_{t+1:t+H+1}, \hat{\boldsymbol{\delta}}_{t+1:t+H+1})$$

Mean Squared Error for position and yaw predictions

Data Collection

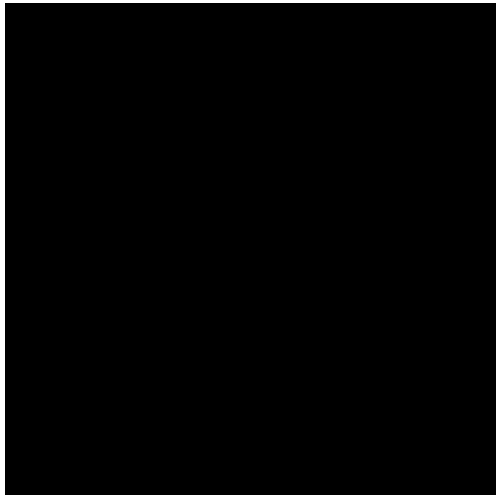
- Roll out action sequences in RotorS simulator
- Match dynamics with the real robot
- Format of one datapoint

$$\mathbf{d}_{in} = (\mathbf{o}_t, \mathbf{s}_t, \mathbf{a}_{t:t+H})$$
$$\mathbf{d}_{out} = (\mathbf{c}^{col}_{t+1:t+H+1}, {}^{V}\mathbf{p}_{t+1:t+H+1}, \boldsymbol{\delta}_{t+1:t+H+1})$$

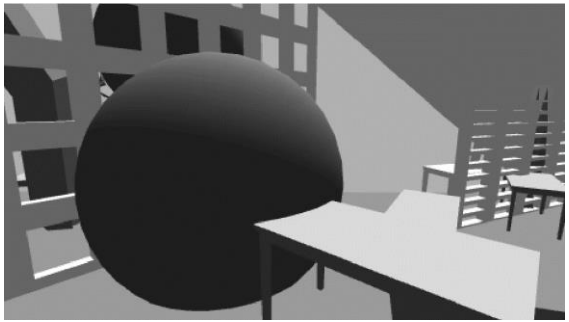
collision status
0: non-collision
1: collision

relative position

relative yaw

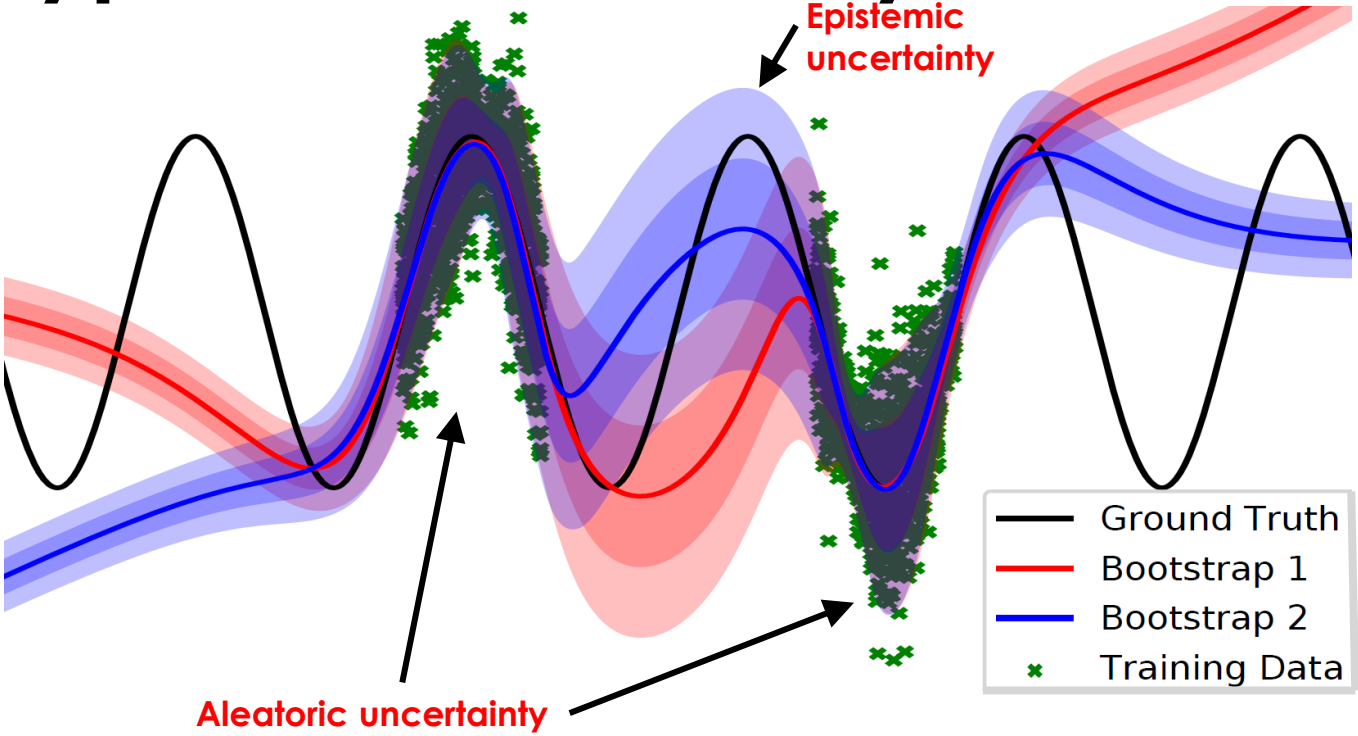


Randomized environments



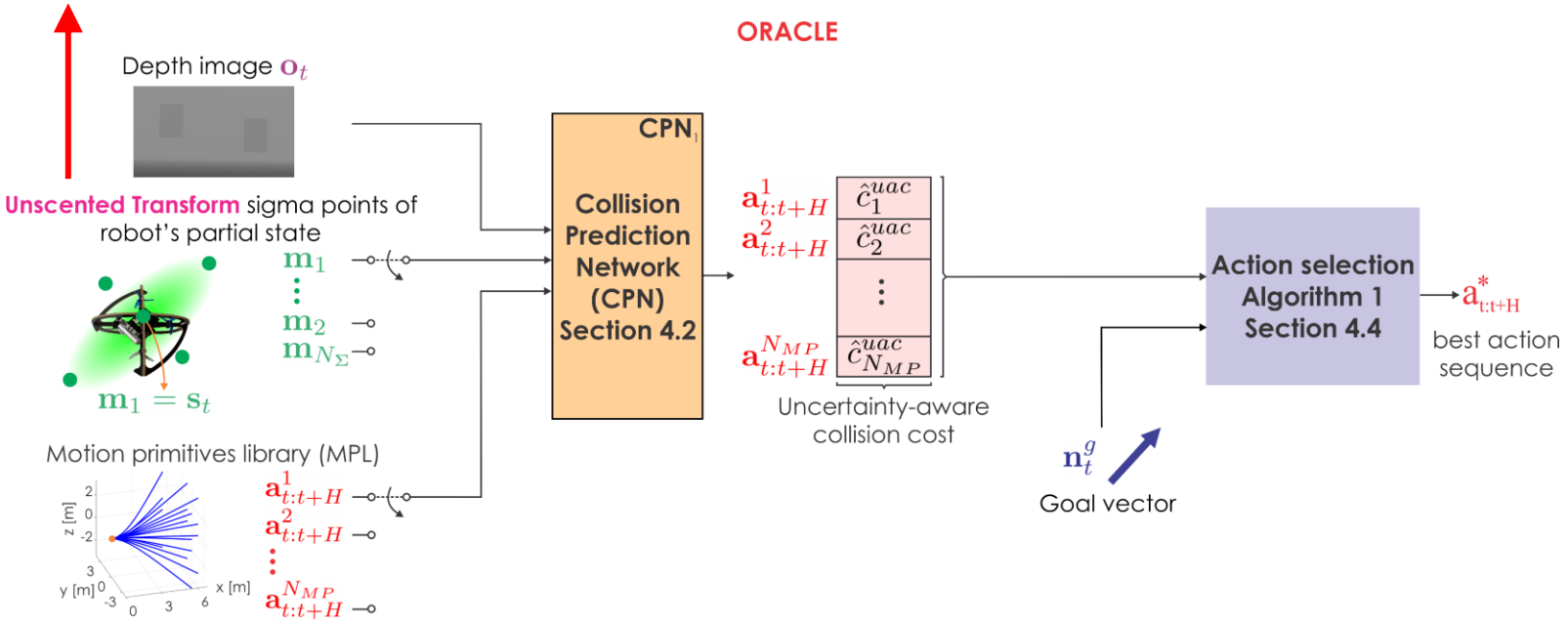
Onboard camera when random action sequences are rolled out

Two types of uncertainty in DL



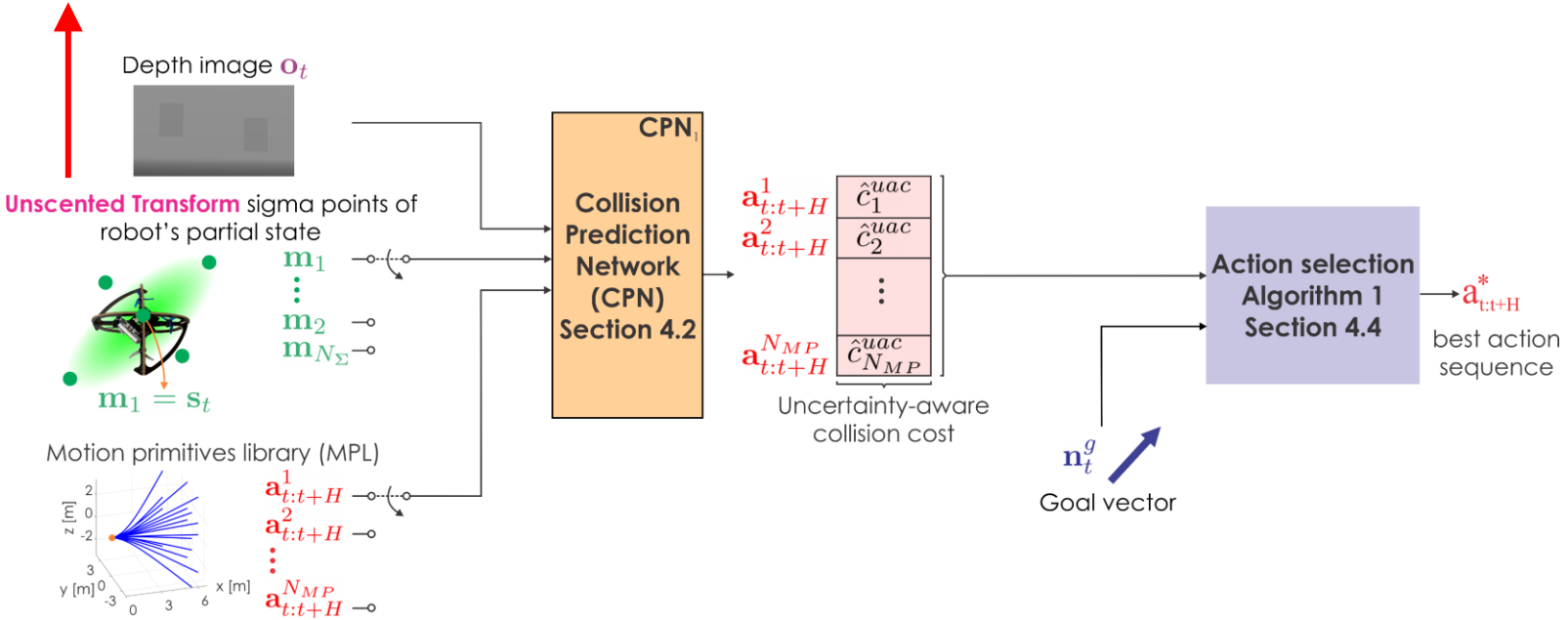
Uncertainty-Aware Safe Navigation

- **Aleatoric uncertainty:** prediction uncertainty due to the noise in input data
- Cannot be reduced with more training data



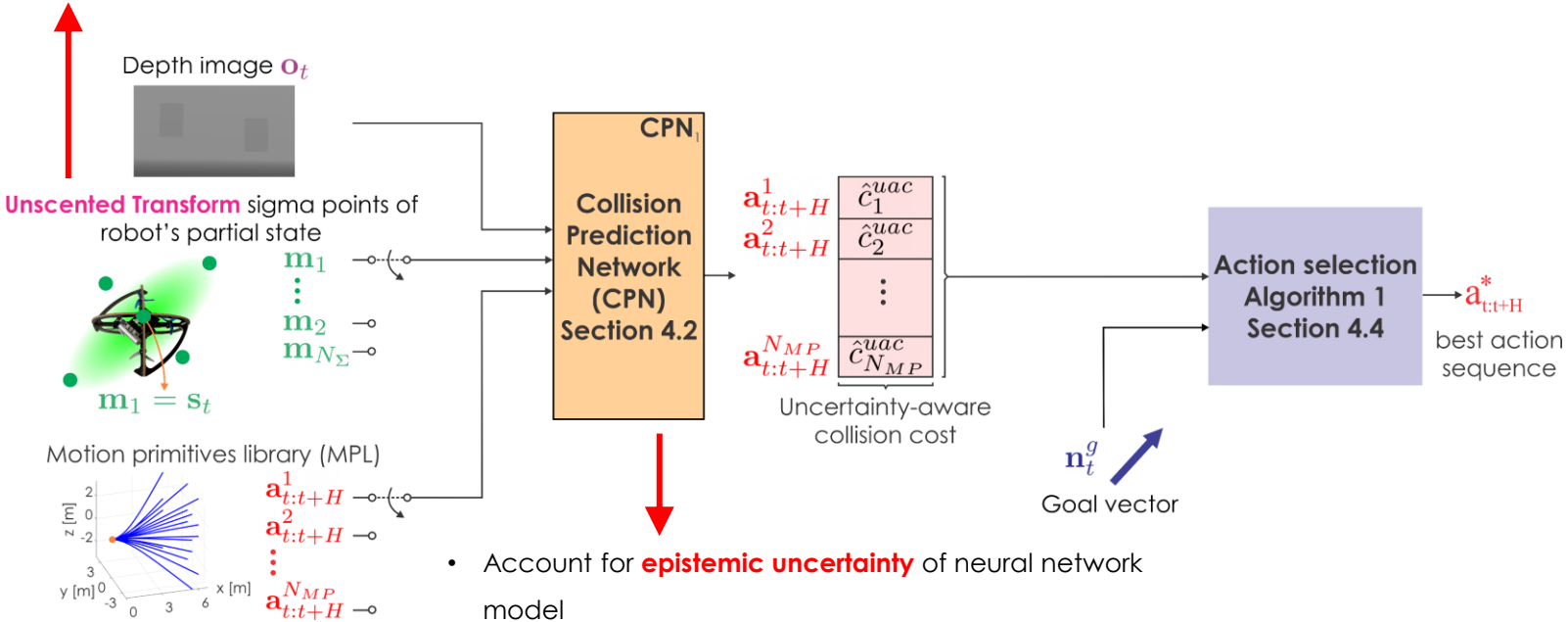
Uncertainty-Aware Safe Navigation

- **Aleatoric uncertainty:** prediction uncertainty due to the noise in input data
- Cannot be reduced with more training data
- Adapt to current estimation uncertainty (vs train with fixed noise)



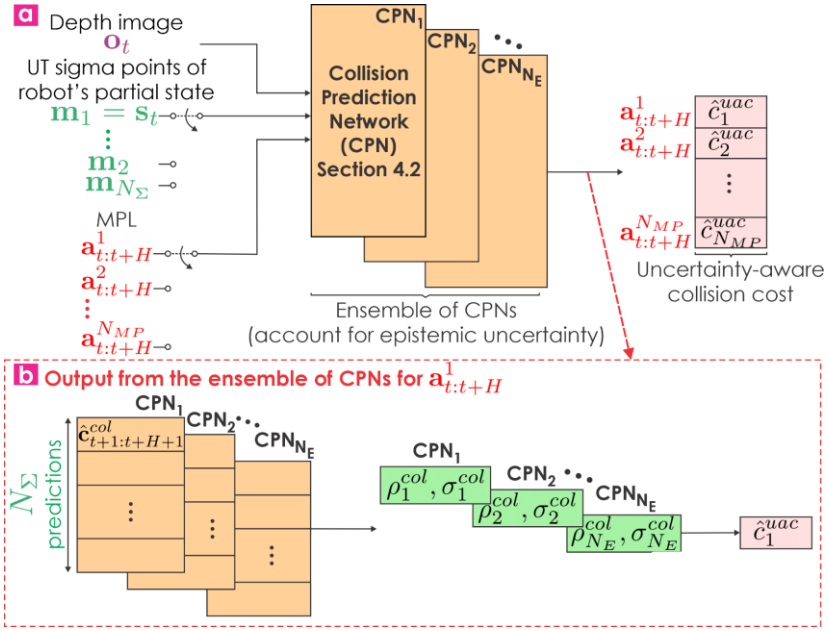
Uncertainty-Aware Safe Navigation

- **Aleatoric uncertainty:** prediction uncertainty due to the noise in input data
- Cannot be reduced with more training data
- Adapt to current estimation uncertainty (vs train with fixed noise)



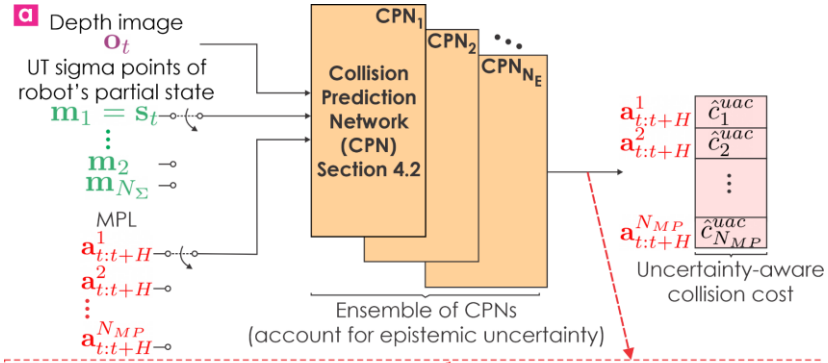
- Account for **epistemic uncertainty** of neural network model
- Can not be ignored for unseen noisy input data

Get Uncertainty-aware Collision Costs



Parallel evaluation on GPU

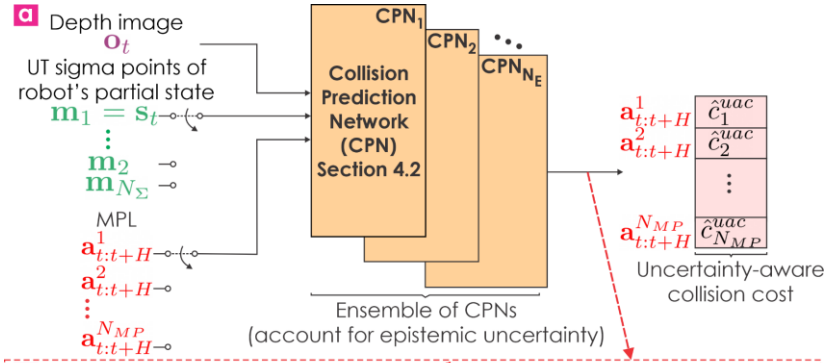
Get Uncertainty-aware Collision Costs



Parallel evaluation on GPU

$$\hat{c}^{col} = \sum_{i=1}^{H} \hat{c}^{col}_{t+i} e^{-\lambda(i-1)}, \ \lambda > 0$$ **Unscented Transform**

Get Uncertainty-aware Collision Costs

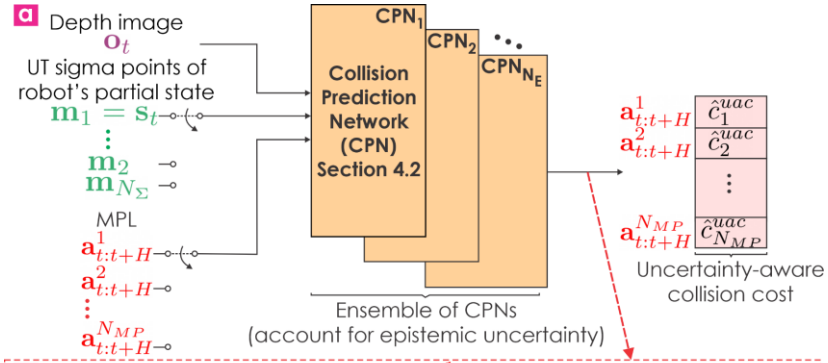


Parallel evaluation on GPU

$$\sigma^{col}_{tot} = \frac{1}{N_E}\sum_{n=1}^{N_E}\left[\sigma^{col}_n + (\rho^{col}_n - \bar{\rho}^{col})^2\right]$$

$$\hat{c}^{uac} = \bar{\rho}^{col} + \alpha\sqrt{\sigma^{col}_{tot}},\ \alpha > 0$$

Get Uncertainty-aware Collision Costs



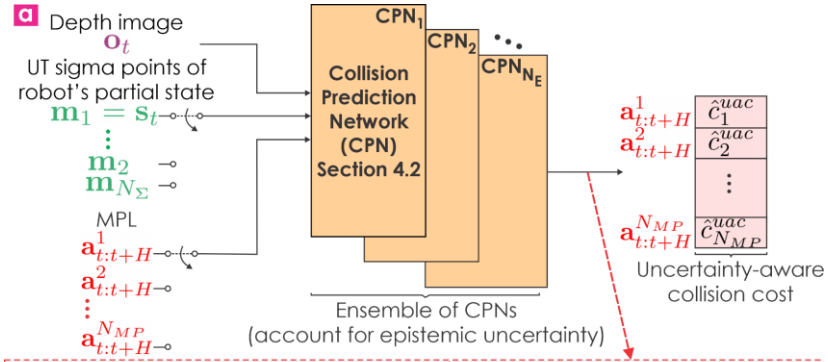
Parallel evaluation on GPU

Mean of variance

$$\sigma^{col}_{tot} = \frac{1}{N_E}\sum_{n=1}^{N_E}\left[\sigma^{col}_n + (\rho^{col}_n - \bar{\rho}^{col})^2\right]$$

$$\hat{c}^{uac} = \bar{\rho}^{col} + \alpha\sqrt{\sigma^{col}_{tot}},\ \alpha > 0$$

Get Uncertainty-aware Collision Costs



Parallel evaluation on GPU

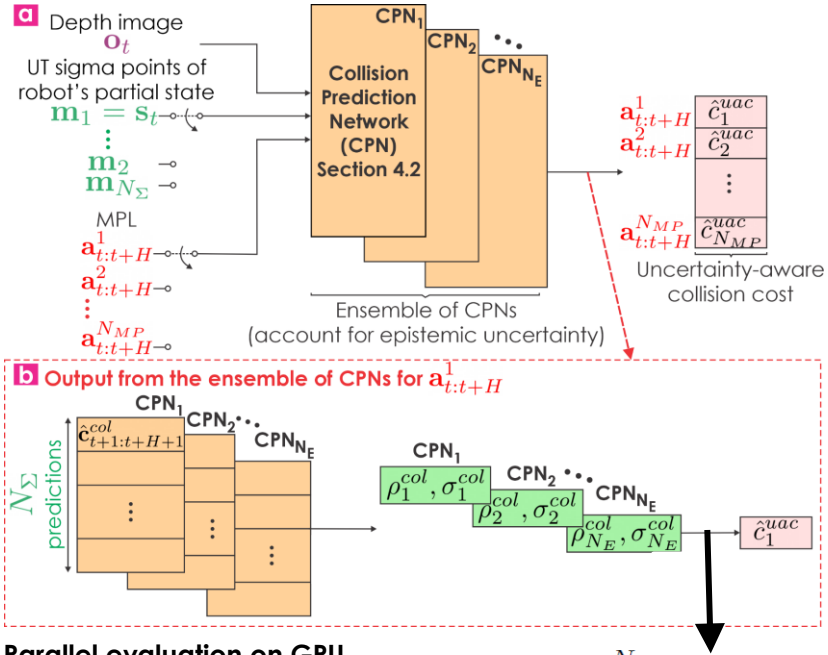
Variance of mean

$$\sigma^{col}_{tot} = \frac{1}{N_E}\sum_{n=1}^{N_E}\left[\sigma^{col}_n + (\rho^{col}_n - \bar{\rho}^{col})^2\right]$$

$$\hat{c}^{uac} = \bar{\rho}^{col} + \alpha\sqrt{\sigma^{col}_{tot}},\ \alpha > 0$$

NTNU

Get Uncertainty-aware Collision Costs



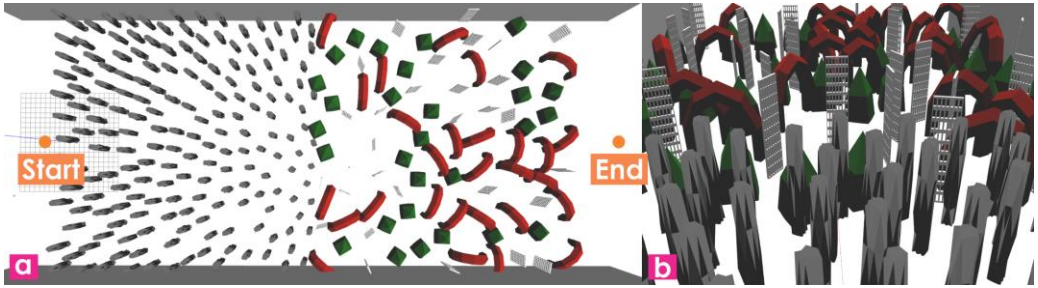
Parallel evaluation on GPU

$$\sigma^{col}_{tot} = \frac{1}{N_E}\sum_{n=1}^{N_E}\left[\sigma^{col}_n + (\rho^{col}_n - \bar{\rho}^{col})^2\right]$$

$$\hat{c}^{uac} = \bar{\rho}^{col} + \alpha\sqrt{\sigma^{col}_{tot}},\ \alpha > 0$$

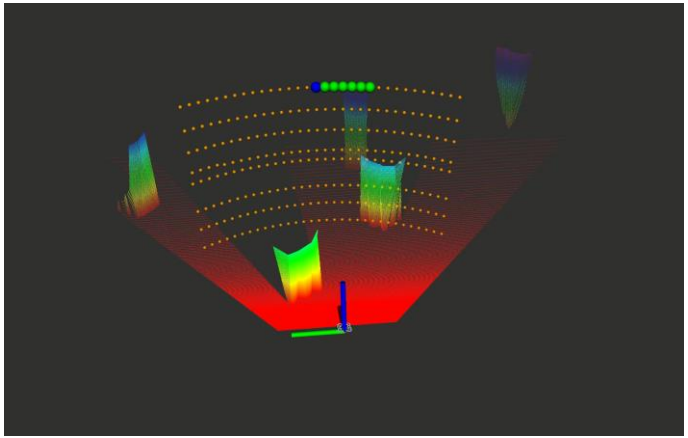
Upper Confidence Bound

Simulation Results (Gazebo)



Simulated environments:

- Noisy velocity and/or depth image
- Poisson disc sampling

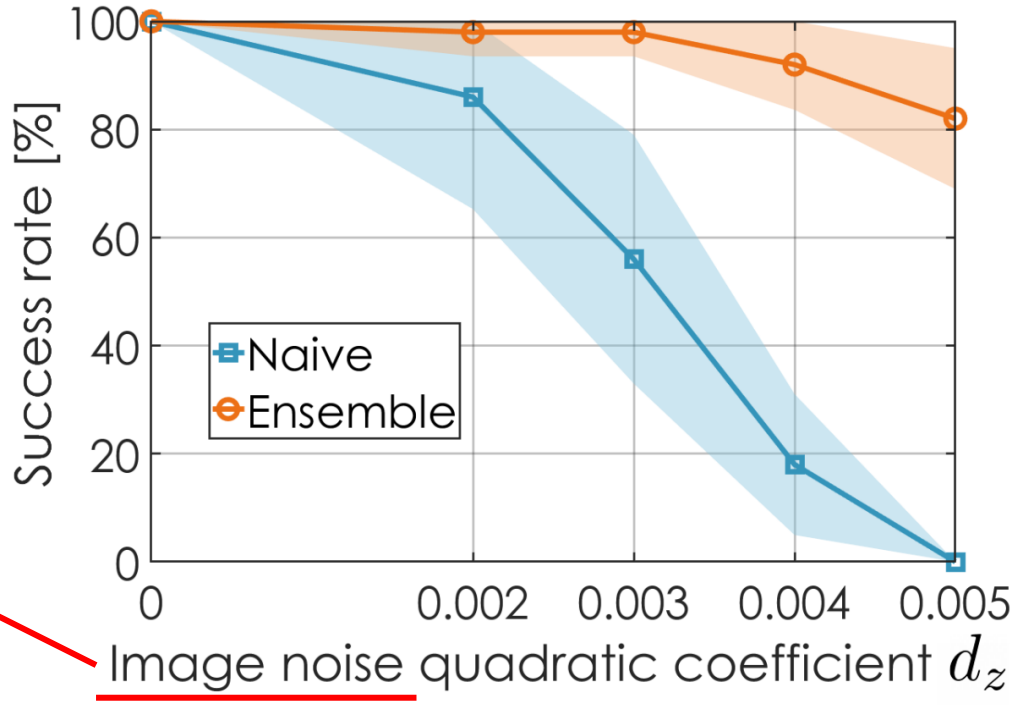


Green: safe action sequences

Blue: chosen action sequence

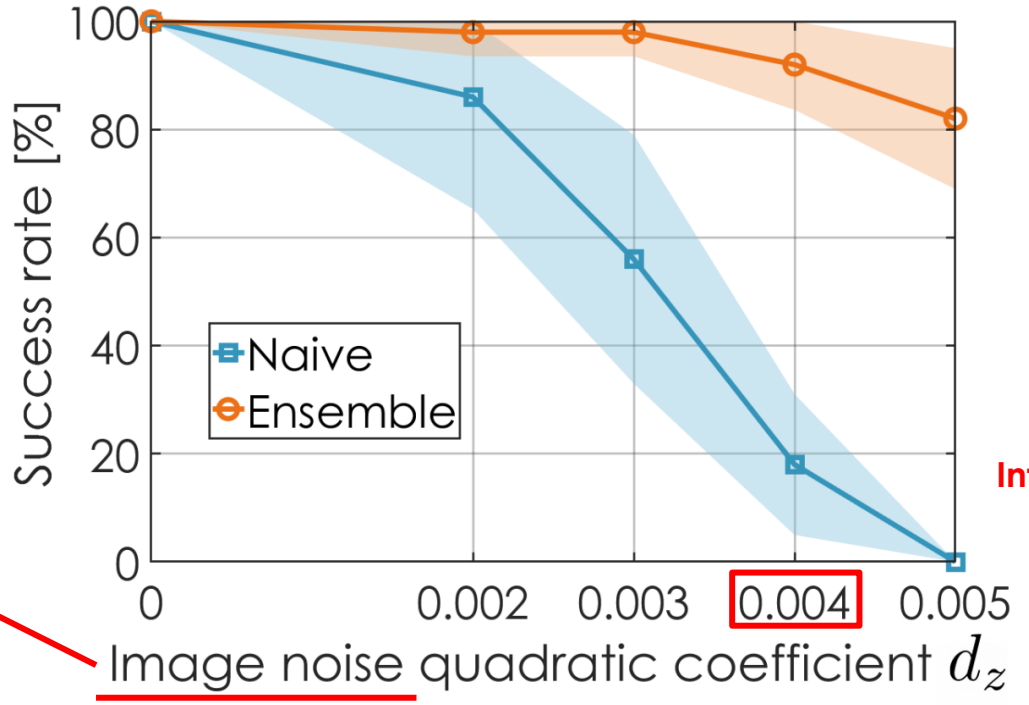
Methods	Naive	Ensemble	ORACLE
UT	No	No	Yes
Ensemble	No	Yes	Yes

Simulation Results (Gazebo)



$$z_{noisy} = z + \mathcal{N}(0, \sigma_z(z))$$
$$\sigma_z(z) = d_z z^2$$

Simulation Results (Gazebo)

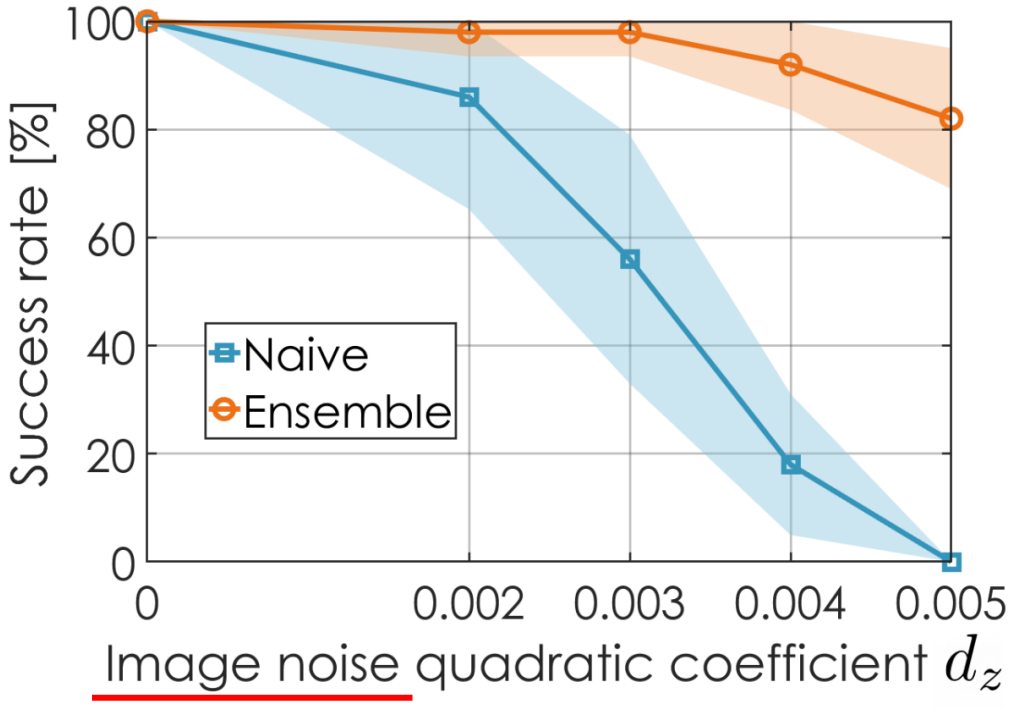


$$z_{noisy} = z + \mathcal{N}(0, \sigma_z(z))$$
$$\sigma_z(z) = d_z z^2$$

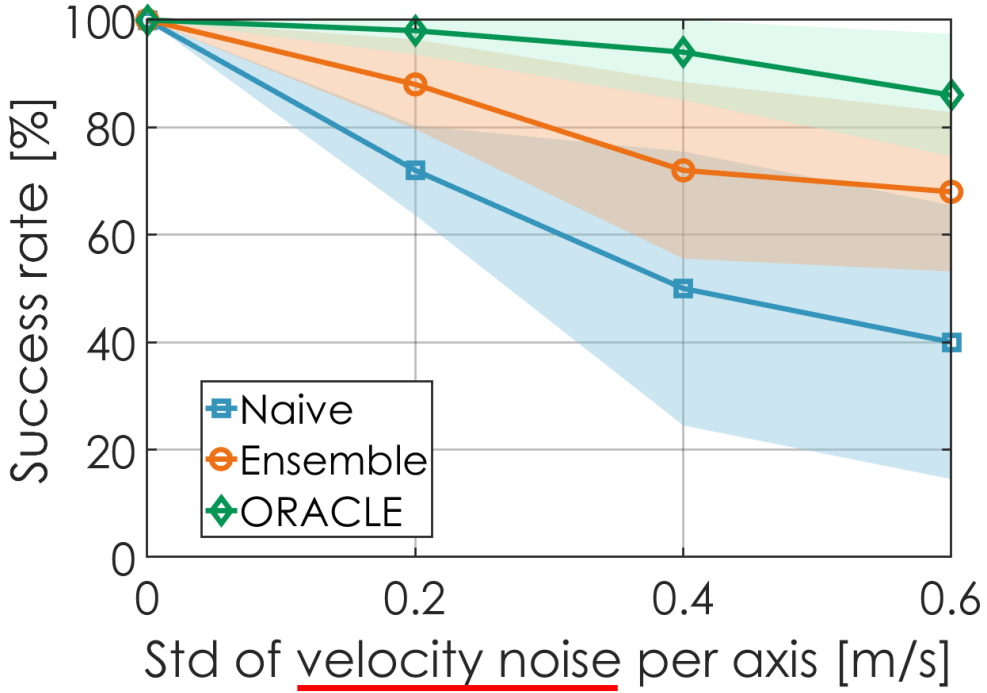
Intel Realsense D435

Simulation Results (Gazebo)

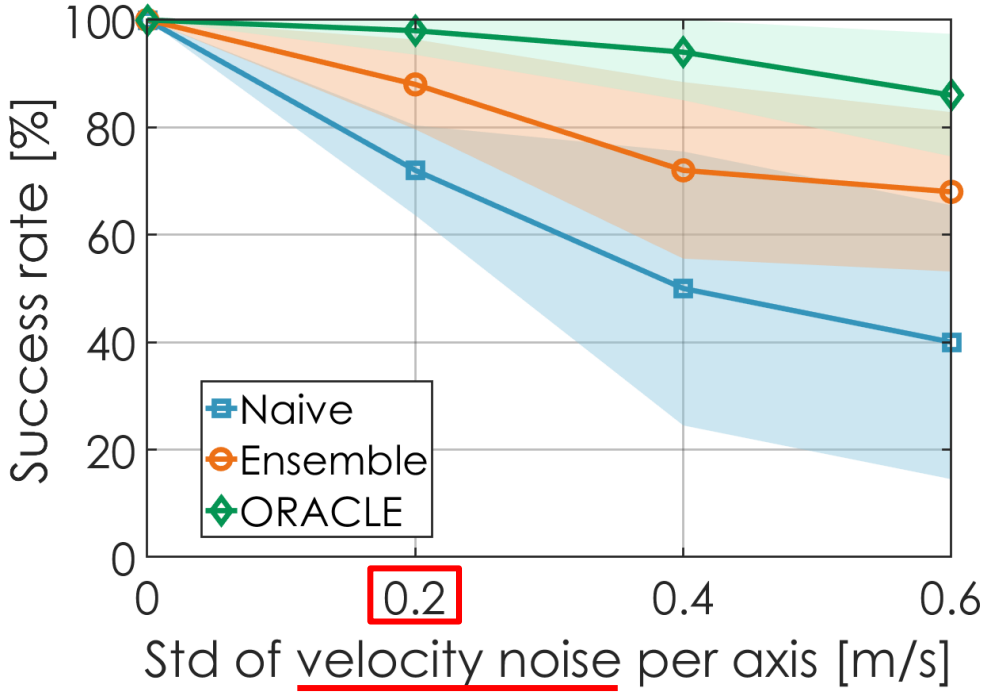
- **ORACLE = Ensemble** (no velocity noise) outperforms **Naïve**



Simulation Results (Gazebo)

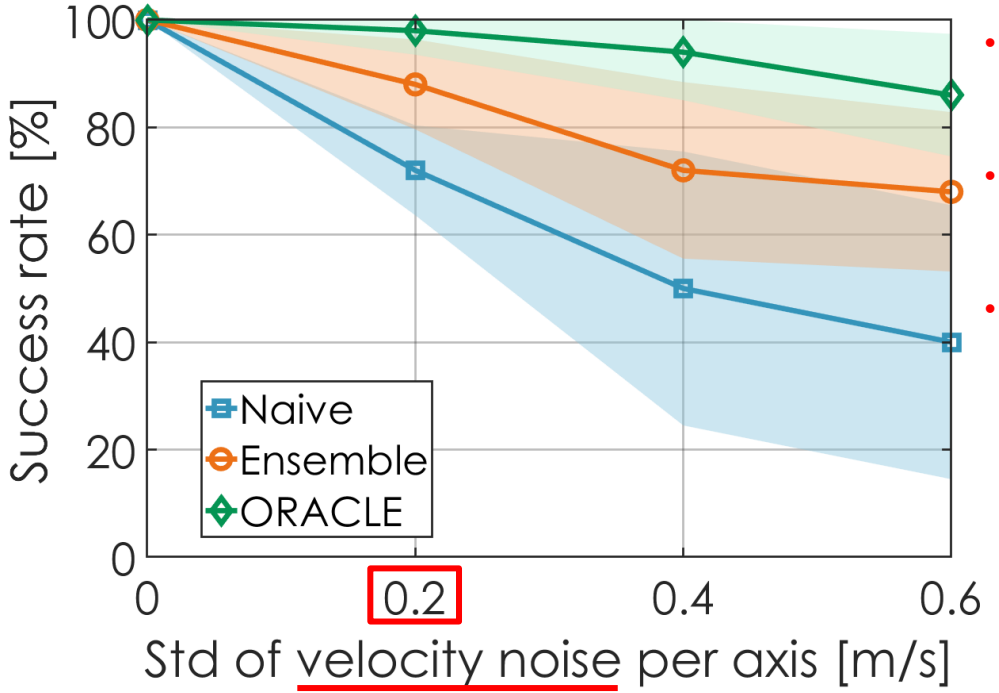


Simulation Results (Gazebo)



Velocity noise in y-axis (main reactive axis) simulated in Agile

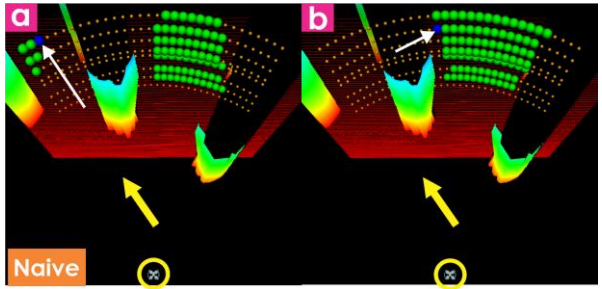
Simulation Results (Gazebo)



- **ORACLE** outperforms **Naïve** and **Ensemble** baselines
- **ORACLE** outperforms **Naïve** and **Ensemble** baselines
- **Ensemble** performs reasonably well without UT. **Why?**

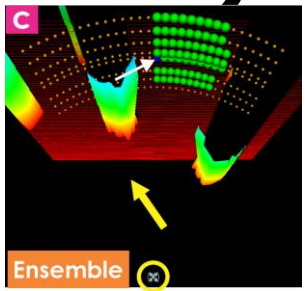
Velocity noise in y-axis (main reactive axis) simulated in Agile

Simulation Results (Gazebo)

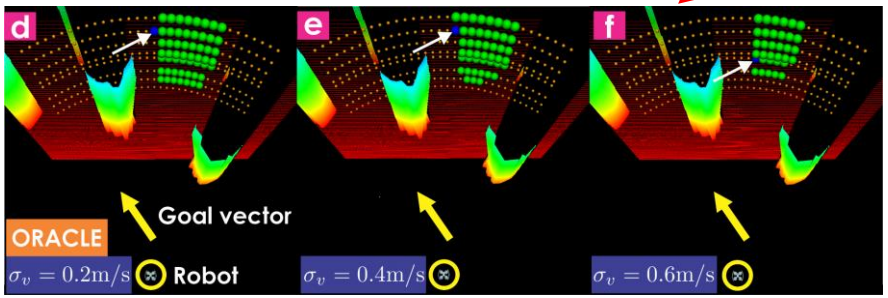


Predictions from 2 different CPNs in the Ensemble

+ Ensemble



+ UT



Velocity noise in UT increases

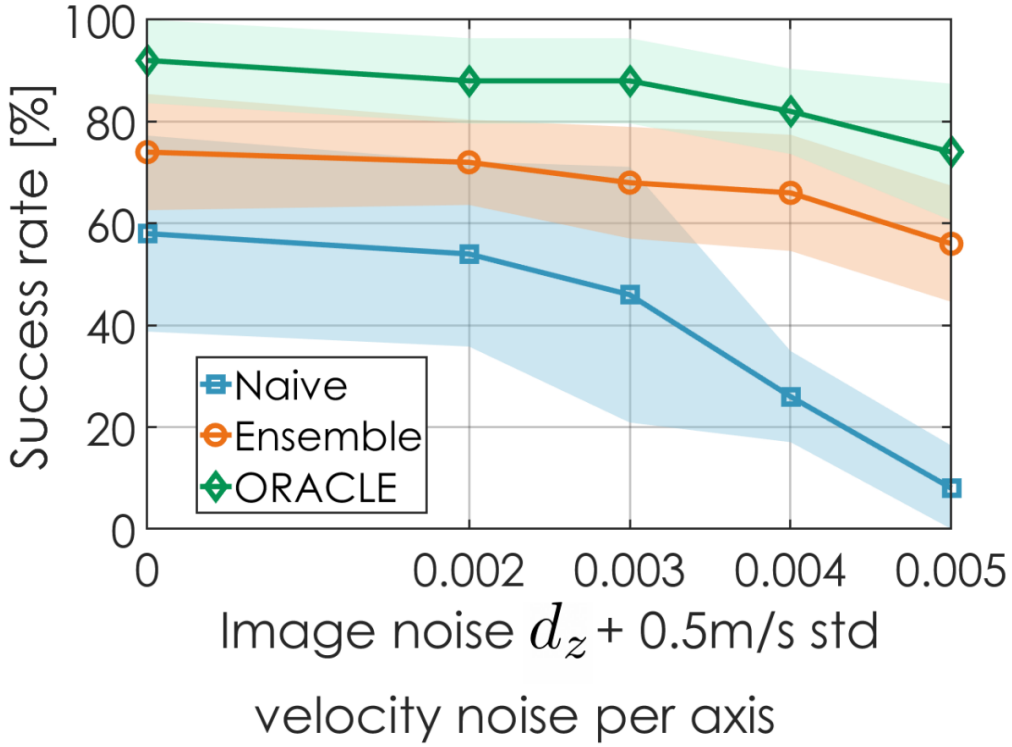
Green: safe action sequences

Blue with arrow: chosen action sequence

- Using the Ensemble leads to **more conservative** set of **safe** action sequences
- Using the Ensemble + UT **enhances conservativeness**

Simulation Results (Gazebo)

- **ORACLE** outperforms **Naïve** and **Ensemble** baselines



Simulation Results (Flightmare)

- **ORACLE** vs **Naïve, Ensemble, Agile** (pretrained weights)
- ORACLE and its variants haven't seen this type of environment in training

Onboard RGB images in an indicative mission

Simulation Results (Flightmare)

Noise levels	Metrics	Naive	Ensemble	ORACLE	Agile
No noise	Success rate [%]	65 (31)	80 (35)	**90** (10)	55 (29)
	Traveled distance of every run [m]	42.3 (14.8)	47.2 (10.5)	**50.2** (5.4)	38.4 (16.3)
	Traveled distance of successful runs [m]	51.2 (1.5)	51.4 (1.7)	51.4 (2.4)	**48.4** (0.4)
	Average acceleration [m/s^2]	0.7 (0.1)	**0.6** (0.1)	**0.6** (0.1)	1.5 (0.4)
	Average jerk [m/s^3]	5.5 (1.1)	4.6 (1.0)	**4.4** (0.8)	8.6 (2.4)
Velocity noise	Success rate [%]	55 (19)	60 (18)	**82** (15)	35 (25)
	Traveled distance of every run [m]	40.5 (15.3)	39.9 (16.5)	**49.0** (13.1)	33.4 (16.6)
	Traveled distance of successful runs [m]	51.6 (1.5)	51.7 (1.6)	53.9 (5.8)	**50.3** (0.6)
	Average acceleration [m/s^2]	1.0 (0.2)	**0.8** (0.2)	0.9 (0.2)	1.4 (0.2)
	Average jerk [m/s^3]	13.1 (2.2)	8.9 (1.6)	9.7 (1.3)	**8.0** (1.7)
Image noise	Success rate [%]	50 (25)	66 (32)	**73** (34)	32 (28)
	Traveled distance of every run [m]	46.3 (14.6)	45.6 (15.1)	**57.0** (18.1)	31.8 (15.0)
	Traveled distance of successful runs [m]	53.4 (4.2)	55.3 (4.0)	60.6 (16.6)	**48.4** (0.3)
	Average acceleration [m/s^2]	1.0 (0.1)	**0.8** (0.1)	**0.8** (0.1)	1.4 (0.3)
	Average jerk [m/s^3]	8.3 (1.0)	6.3 (1.0)	**5.4** (0.8)	8.5 (1.6)
Both noise	Success rate [%]	32 (16)	53 (16)	**64** (27)	24 (25)
	Traveled distance of every run [m]	42.9 (18.1)	44.1 (15.5)	**52.8** (22.1)	30.2 (15.5)
	Traveled distance of successful runs [m]	54.0 (5.4)	58.8 (9.7)	62.1 (10.8)	**50.2** (0.6)
	Average acceleration [m/s^2]	1.2 (0.1)	**0.9** (0.1)	1.0 (0.1)	1.5 (0.3)
	Average jerk [m/s^3]	15.3 (1.8)	9.9 (1.1)	10.4 (1.3)	**8.5** (1.8)

Simulation results in Flightmare with 10 forests x 10 runs/forest,
the metrics are given in mean (std) values

Simulation Results (Flightmare)

Noise levels	Metrics	Naive	Ensemble	ORACLE	Agile
No noise	Success rate [%]	65 (31)	80 (35)	**90** (10)	55 (29)
	Traveled distance of every run [m]	42.3 (14.8)	47.2 (10.5)	**50.2** (5.4)	38.4 (16.3)
	Traveled distance of successful runs [m]	51.2 (1.5)	51.4 (1.7)	51.4 (2.4)	**48.4** (0.4)
	Average acceleration [m/s^2]	0.7 (0.1)	**0.6** (0.1)	**0.6** (0.1)	1.5 (0.4)
	Average jerk [m/s^3]	5.5 (1.1)	4.6 (1.0)	**4.4** (0.8)	8.6 (2.4)
Velocity noise	Success rate [%]	55 (19)	60 (18)	**82** (15)	35 (25)
	Traveled distance of every run [m]	40.5 (15.3)	39.9 (16.5)	**49.0** (13.1)	33.4 (16.6)
	Traveled distance of successful runs [m]	51.6 (1.5)	51.7 (1.6)	53.9 (5.8)	**50.3** (0.6)
	Average acceleration [m/s^2]	1.0 (0.2)	**0.8** (0.2)	0.9 (0.2)	1.4 (0.2)
	Average jerk [m/s^3]	13.1 (2.2)	8.9 (1.6)	9.7 (1.3)	**8.0** (1.7)
Image noise	Success rate [%]	50 (25)	66 (32)	**73** (34)	32 (28)
	Traveled distance of every run [m]	46.3 (14.6)	45.6 (15.1)	**57.0** (18.1)	31.8 (15.0)
	Traveled distance of successful runs [m]	53.4 (4.2)	55.3 (4.0)	60.6 (16.6)	**48.4** (0.3)
	Average acceleration [m/s^2]	1.0 (0.1)	**0.8** (0.1)	**0.8** (0.1)	1.4 (0.3)
	Average jerk [m/s^3]	8.3 (1.0)	6.3 (1.0)	**5.4** (0.8)	8.5 (1.6)
Both noise	Success rate [%]	32 (16)	53 (16)	**64** (27)	24 (25)
	Traveled distance of every run [m]	42.9 (18.1)	44.1 (15.5)	**52.8** (22.1)	30.2 (15.5)
	Traveled distance of successful runs [m]	54.0 (5.4)	58.8 (9.7)	62.1 (10.8)	**50.2** (0.6)
	Average acceleration [m/s^2]	1.2 (0.1)	**0.9** (0.1)	1.0 (0.1)	1.5 (0.3)
	Average jerk [m/s^3]	15.3 (1.8)	9.9 (1.1)	10.4 (1.3)	**8.5** (1.8)

Simulation results in Flightmare with 10 forests x 10 runs/forest,
the metrics are given in mean (std) values

- **Naïve-Agile** similar performance

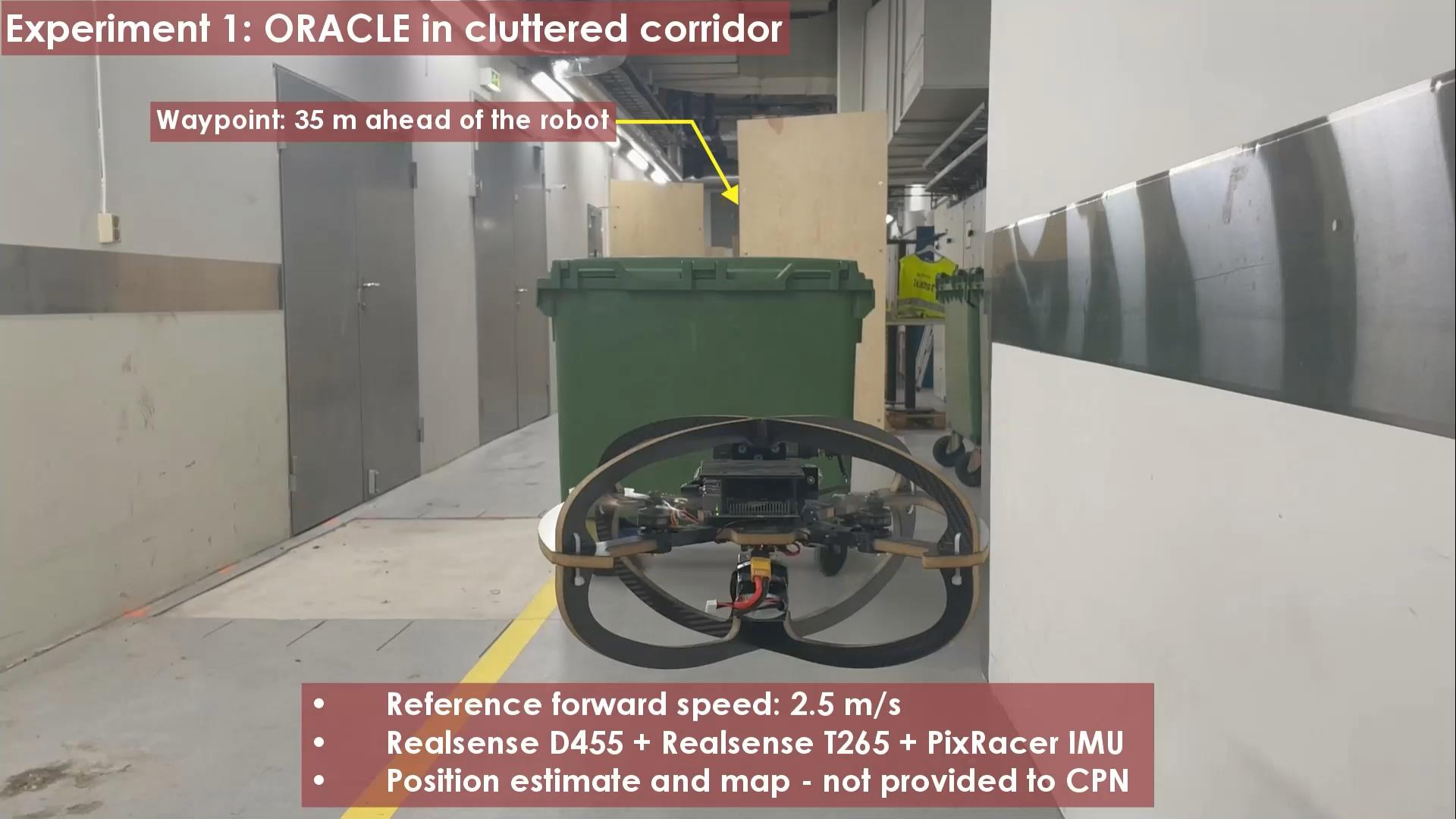
Simulation Results (Flightmare)

Noise levels	Metrics	Naive	Ensemble	ORACLE	Agile
No noise	Success rate [%]	65 (31)	80 (35)	**90** (10)	55 (29)
	Traveled distance of every run [m]	42.3 (14.8)	47.2 (10.5)	**50.2** (5.4)	38.4 (16.3)
	Traveled distance of successful runs [m]	51.2 (1.5)	51.4 (1.7)	51.4 (2.4)	**48.4** (0.4)
	Average acceleration [m/s^2]	0.7 (0.1)	**0.6** (0.1)	**0.6** (0.1)	1.5 (0.4)
	Average jerk [m/s^3]	5.5 (1.1)	4.6 (1.0)	**4.4** (0.8)	8.6 (2.4)
Velocity noise	Success rate [%]	55 (19)	60 (18)	**82** (15)	35 (25)
	Traveled distance of every run [m]	40.5 (15.3)	39.9 (16.5)	**49.0** (13.1)	33.4 (16.6)
	Traveled distance of successful runs [m]	51.6 (1.5)	51.7 (1.6)	53.9 (5.8)	**50.3** (0.6)
	Average acceleration [m/s^2]	1.0 (0.2)	**0.8** (0.2)	0.9 (0.2)	1.4 (0.2)
	Average jerk [m/s^3]	13.1 (2.2)	8.9 (1.6)	9.7 (1.3)	**8.0** (1.7)
Image noise	Success rate [%]	50 (25)	66 (32)	**73** (34)	32 (28)
	Traveled distance of every run [m]	46.3 (14.6)	45.6 (15.1)	**57.0** (18.1)	31.8 (15.0)
	Traveled distance of successful runs [m]	53.4 (4.2)	55.3 (4.0)	60.6 (16.6)	**48.4** (0.3)
	Average acceleration [m/s^2]	1.0 (0.1)	**0.8** (0.1)	**0.8** (0.1)	1.4 (0.3)
	Average jerk [m/s^3]	8.3 (1.0)	6.3 (1.0)	**5.4** (0.8)	8.5 (1.6)
Both noise	Success rate [%]	32 (16)	53 (16)	**64** (27)	24 (25)
	Traveled distance of every run [m]	42.9 (18.1)	44.1 (15.5)	**52.8** (22.1)	30.2 (15.5)
	Traveled distance of successful runs [m]	54.0 (5.4)	58.8 (9.7)	62.1 (10.8)	**50.2** (0.6)
	Average acceleration [m/s^2]	1.2 (0.1)	**0.9** (0.1)	1.0 (0.1)	1.5 (0.3)
	Average jerk [m/s^3]	15.3 (1.8)	9.9 (1.1)	10.4 (1.3)	**8.5** (1.8)

Simulation results in Flightmare with 10 forests x 10 runs/forest, the metrics are given in mean (std) values

- **Naïve-Agile** similar performance
- Simulation with noise: **ORACLE, Ensemble** outperform **Agile**

Experiment 1: ORACLE in cluttered corridor



- Reference forward speed: 2.5 m/s
- Realsense D455 + Realsense T265 + PixRacer IMU
- Position estimate and map - not provided to CPN

Code Examples and Tasks

- https://github.com/ntnu-arl/aerial_gym_simulator
- https://ntnu-arl.github.io/aerial_gym_simulator/
- https://github.com/ntnu-arl/ORACLE

Find out more

- Nguyen, H., Andersen, R., Boukas, E. and Alexis, K., 2024. Uncertainty-aware visually-attentive navigation using deep neural networks. The International Journal of Robotics Research, 43(6), pp.840-872.
- A. Loquercio et al., "Learning high-speed flight in the wild," Science Robotics, 2021
- G. Kahn et al., "Badgr: An autonomous self-supervised learning-based navigation system," RAL 2021
- S. Veer et al., "Probably approximately correct vision-based planning using motion primitives," CoRL 2020