# Evaluating Robot designs in Generated Environments

Anwesha Chattoraj, Eric Vin, Daniel Fremont, Ankur Mehta

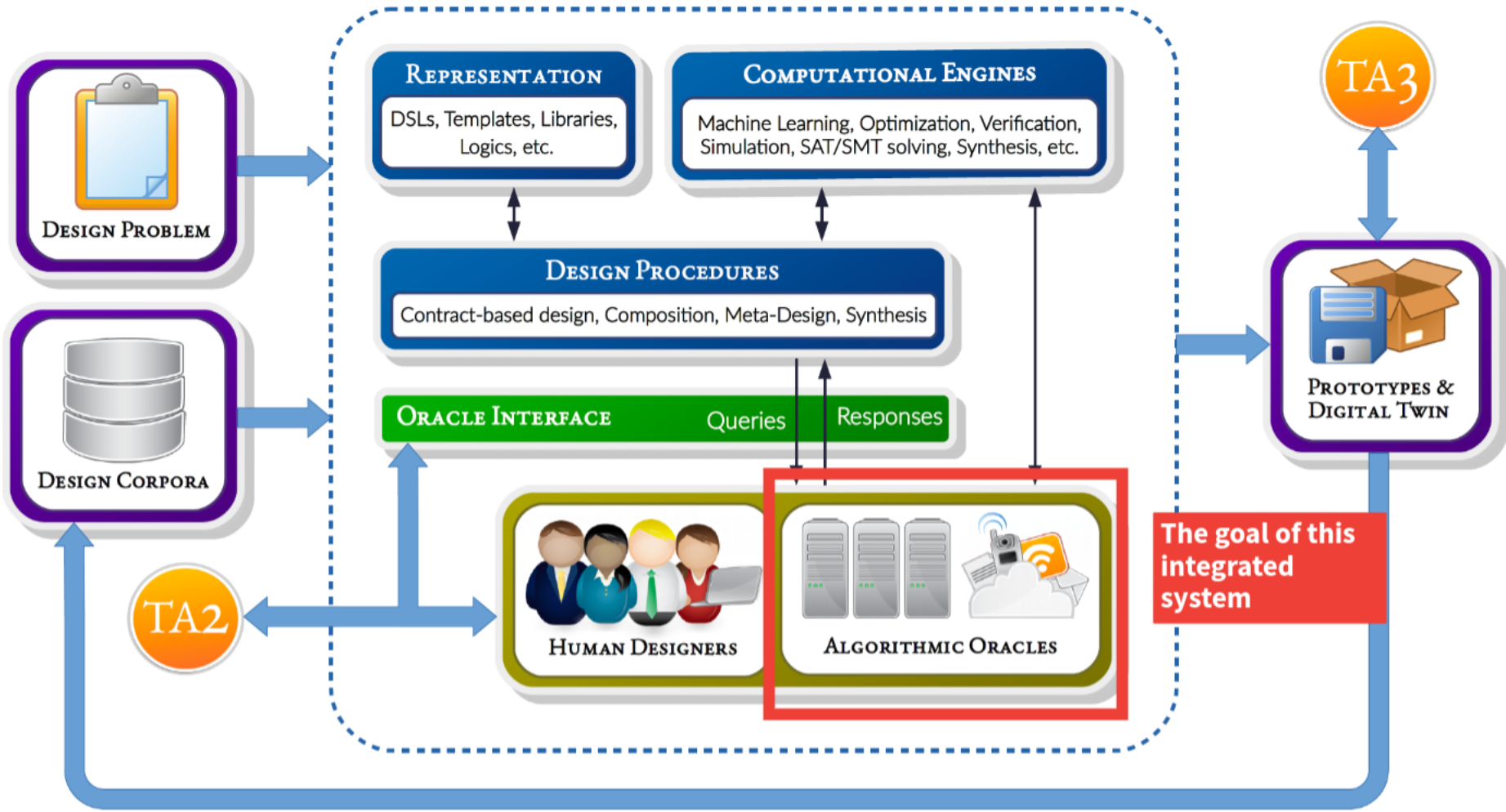
LEMUR, UCSC Formal Methods,
UCLA,UCSC

January 28, 2022

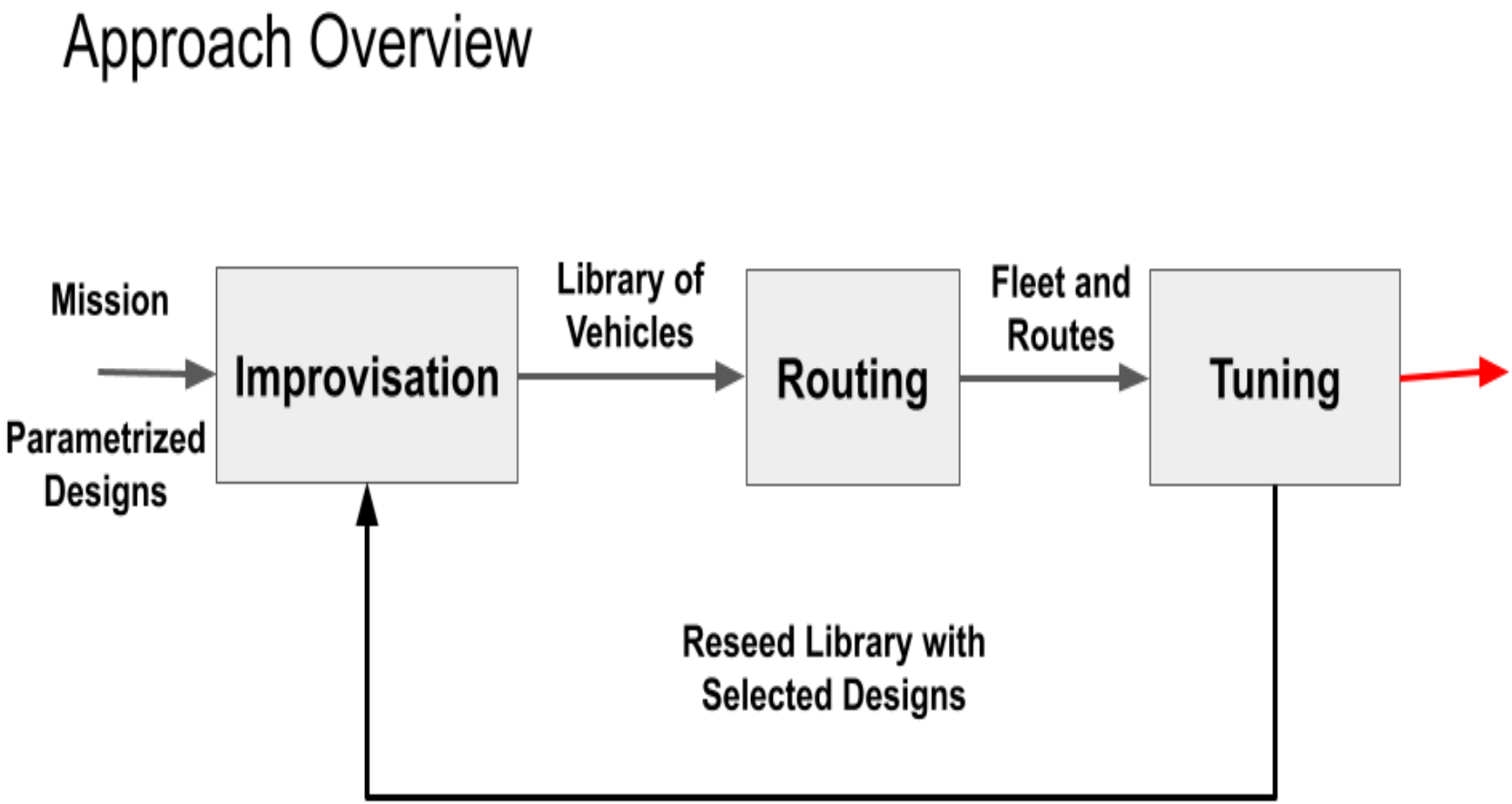
# Table of Contents


1. Motivation
2. Robot Compiler (RoCo)
3. Scenic
4. Webots Simulations
5. Results
6. Future Work
7. References

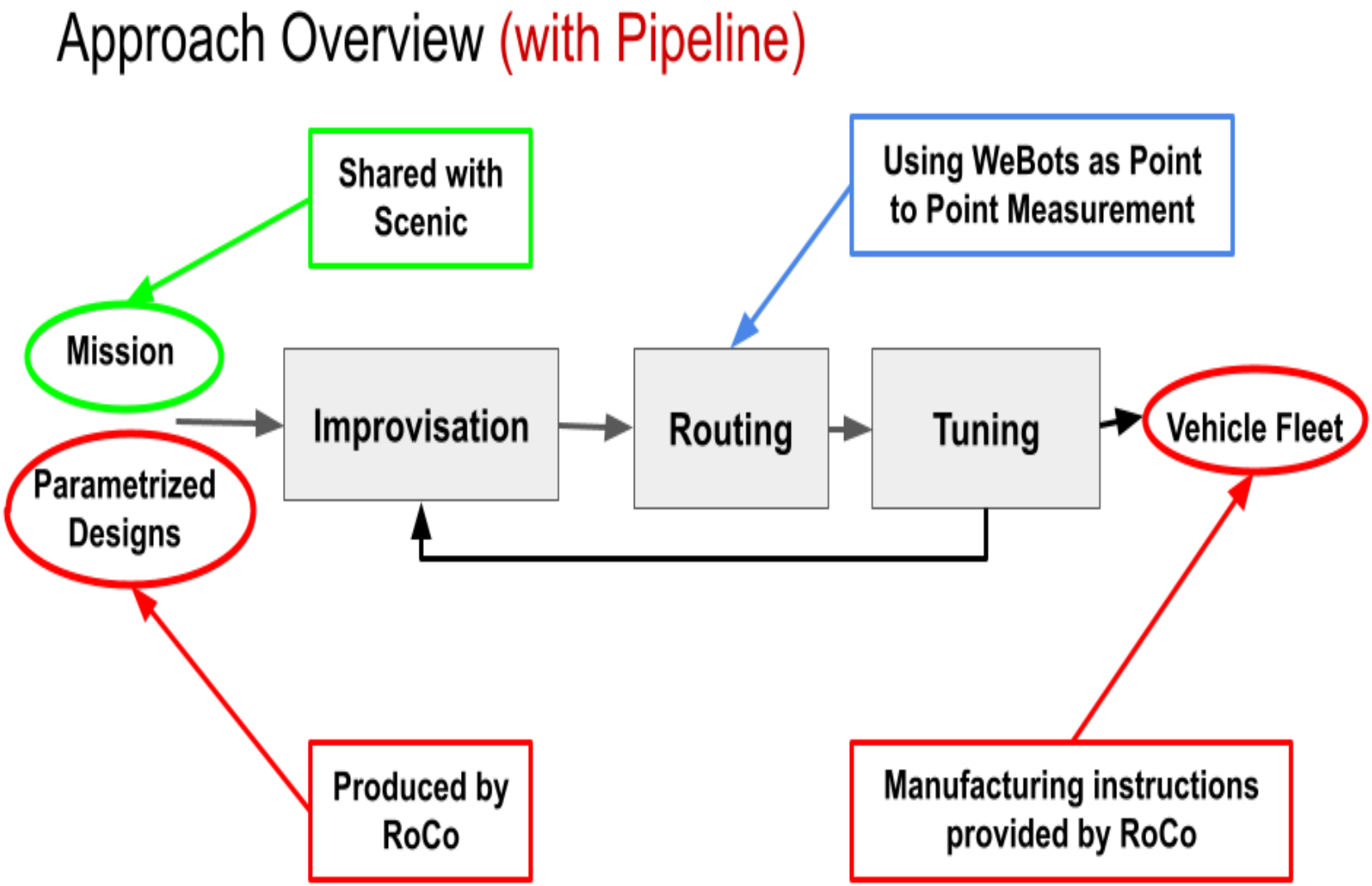
# Niche in larger LOGiCs space



# Downstream Integration: Symbiotic Fleet Configuration  Planning Co-Design

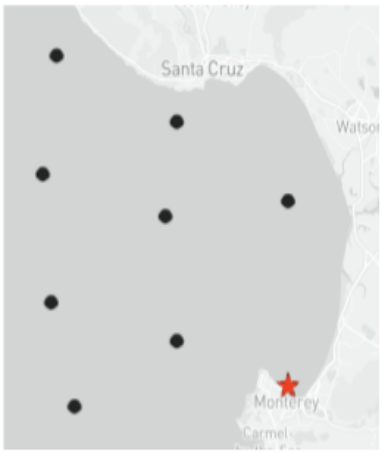


# Downstream Integration: Symbiotic Fleet Configuration  Planning Co-Design + Pipeline



# Symbiotically Learning **Heterogeneous** Fleets

Connects to *STR Challenge Mission #2*




| DRIVING VARIABLES | | | |
|---|---|---|---|
| Vehicle fairing displacement | 833.33 | kg | |
| Diameter | 0.5644 | m | Overall diameter - reference vehicle is 0.3048m |
| Length | 4.1034 | m | Overall length - reference vehicle is 2.2m |
| PV Material | Aluminum Oxide | | CHOOSE FROM MATERIAL LIST TO RIGHT |
| Depth Rating | 1,000.00 | m | Determines thickness and weight of pressure vessel |
| Safety Factor for Depth Rating | 1.20 | multiplier | Safety factor of 1.5 fine for unmanned syste |
| Battery Specific Energy | 180.00 | W-hr/kg | Batteries used to power vehicle. |
| Battery Fraction | 0.50 | fraction | Fraction of Pressure Vessel (P ancy (total buoyancy - PV weight) use |
| Non-scalable payload weight | 10.00 | kg | Dry weight of systems t cale in size with vehicle size (e.g. an inertial |
| Vehicle Hotel Power Draw includin | 50.00 | W | Power draw by p pulsion systems (e.g. power draw that does not chang |
| Payload Power Draw | 81.54 | W | All non-p ower, including sonar. |
| Cd - Drag coefficient | 0.01 | Nondimensional | D (Cd*wetted surface area*(1+appendage fraction)*water density*vel |
| Appendage added area | 0.10 | fraction | count for control fin drag, we just suppliment the vehicle area (a cheat) |
| Propulsion efficiency | 0.50 | fraction | This combines efficiencies of propellor and motor, and includes losses for gearb |
| Density of seawater | 1,027.00 | kg | An average for the ocean is 1027 kg/m^3 |
| Fineness Ratio (L/D) | | This should really be in the vicinity of 6-7 | |
| Vehicle fairing dispacement | 4880 | kg | |
| Wetted Surface of Fairing | 6.36 | m^2 | Area used in the drag calculation |
| Midbody length | 1.68 | m | |
| Midbody volume | 0.51 | m^3 | |
| PV Weight | 172.95 | kg | |
| Pressure Vessel | 527.23 | kg | |
| Battery Weight | 154.07 | kg | |
| Battery Capacity | 27,732.72 | W-hr | 99,837,780.12 J |

| WARNING PANEL | |
|---|---|
| 4.10 | **PV Length.** If red, |
| 154.07 | **Battery weight** m |
| 7.27 | Length/Diameter - |
| 1,000.00 | Ocean isn't deeper |

Note: Reference vehicle is 33.5 MJ for

**Replace with our pipeline**

**Challenge for heterogeneous fleets:**

**No mapping between mission objectives and vehicle constraints**

## Research Focus

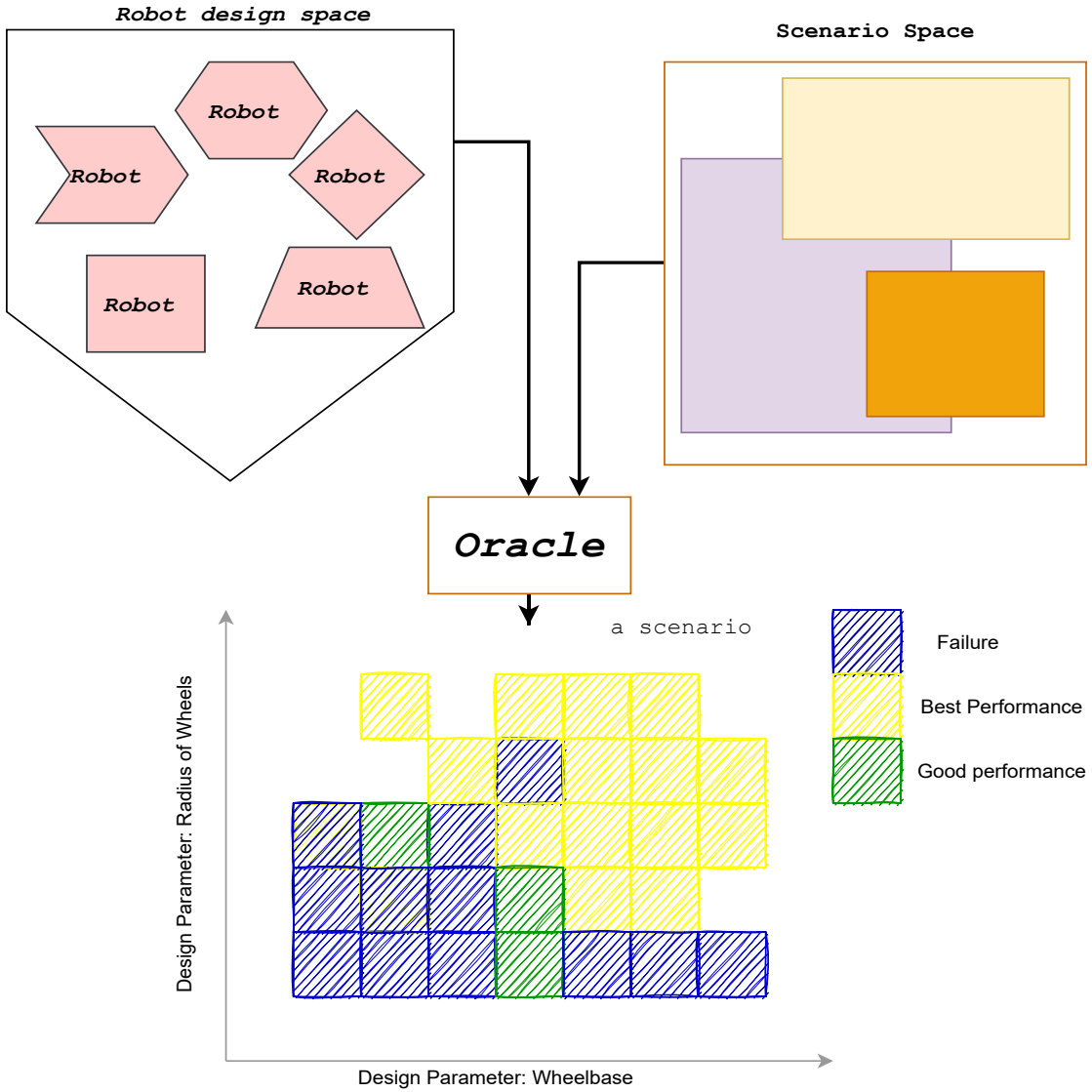
- Framework for learning fleet compositions from a **spatial** mission
- Algorithmic challenges in a variant of the multi-agent TSP
- Symbioses with human developers to guide the search

2

# Motivation

- Implement an algorithmic oracle to effectively evaluate robot designs over multiple environments

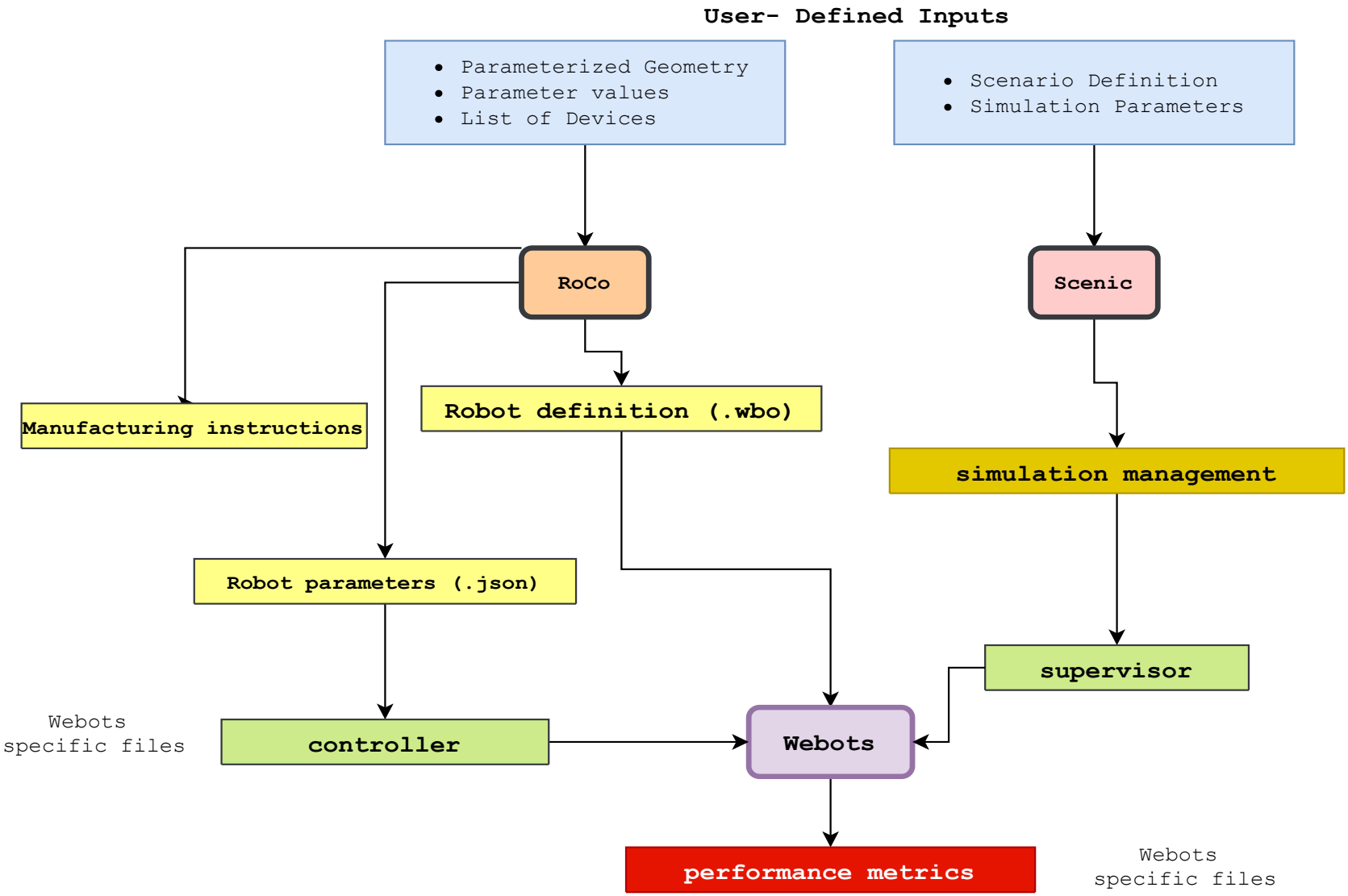
# Finding designs with desired performance characteristics



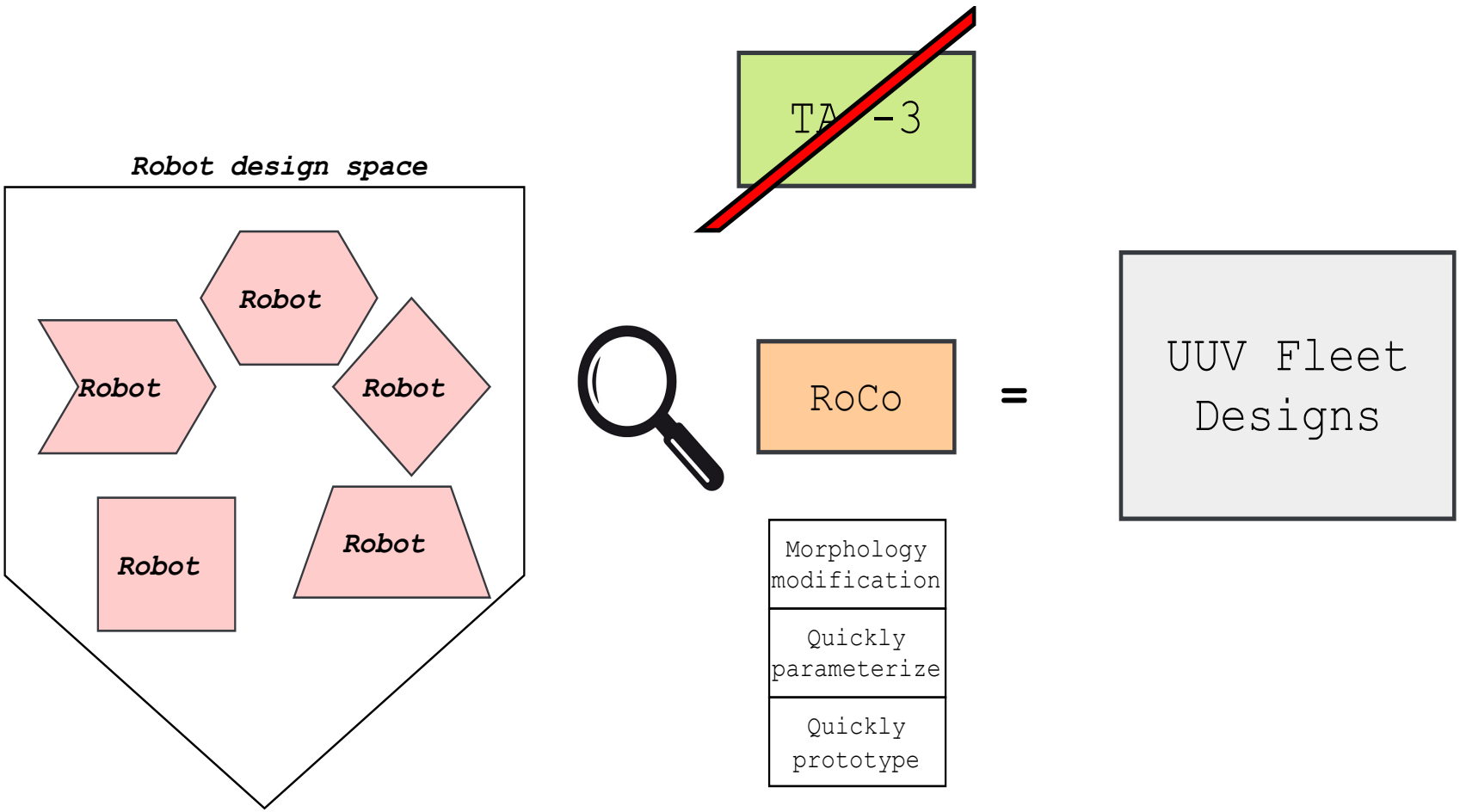
# Finding designs with desired performance characteristics

- Iterate multiple designs over a wide space of scenarios
- Identify parameter combinations with desired performance characteristics

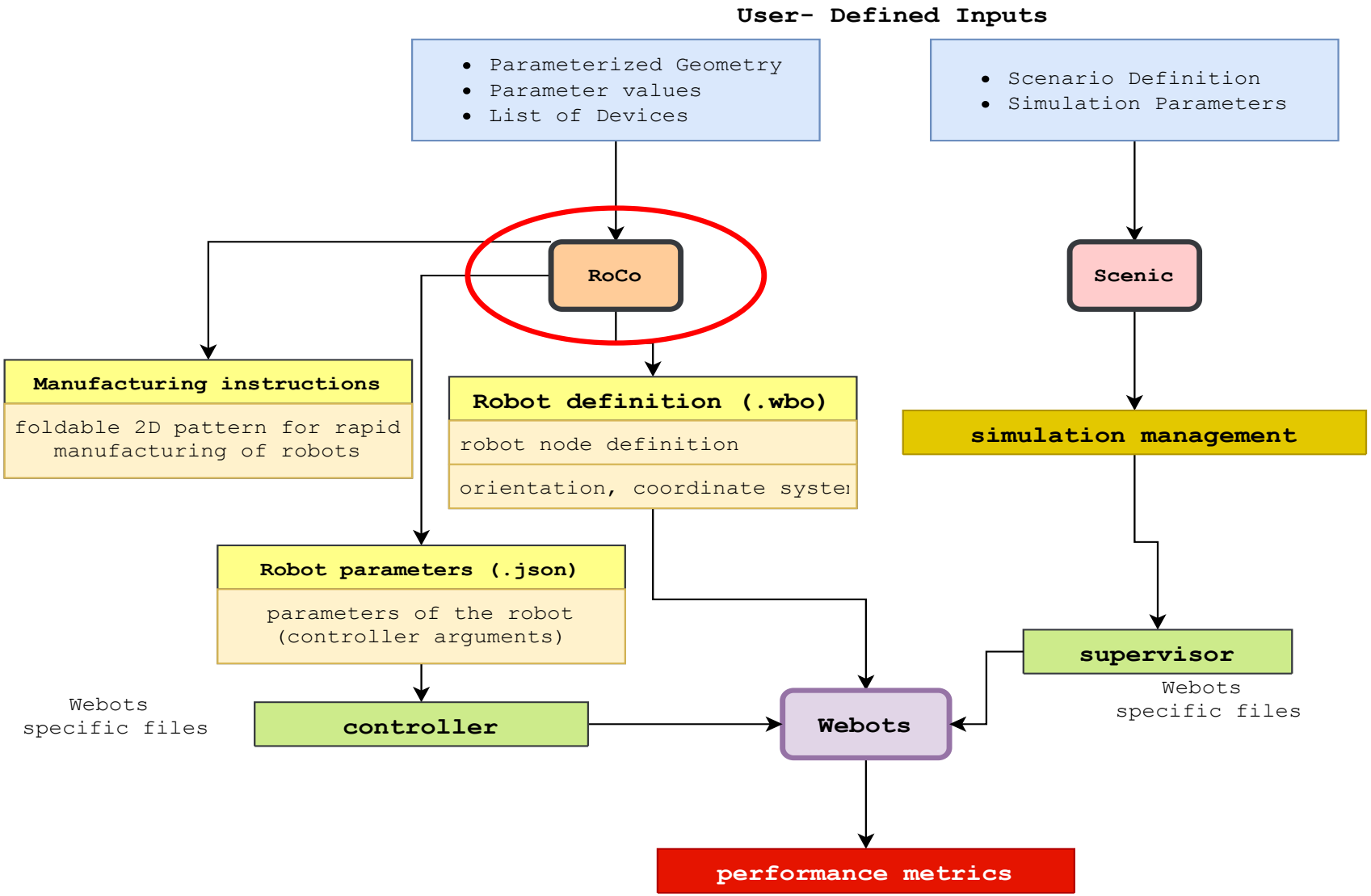
# Overview


User- Defined Inputs
Parameterized Geometry
Parameter values
List of Devices
Scenario Definition
Simulation Parameters
RoCo
Scenic
Manufacturing instructions
Robot definition (.wbo)
simulation management
Robot parameters (.json)
supervisor
Webots
specific files
controller
Webots
performance metrics
Webots
specific files

# Design space exploration motivation

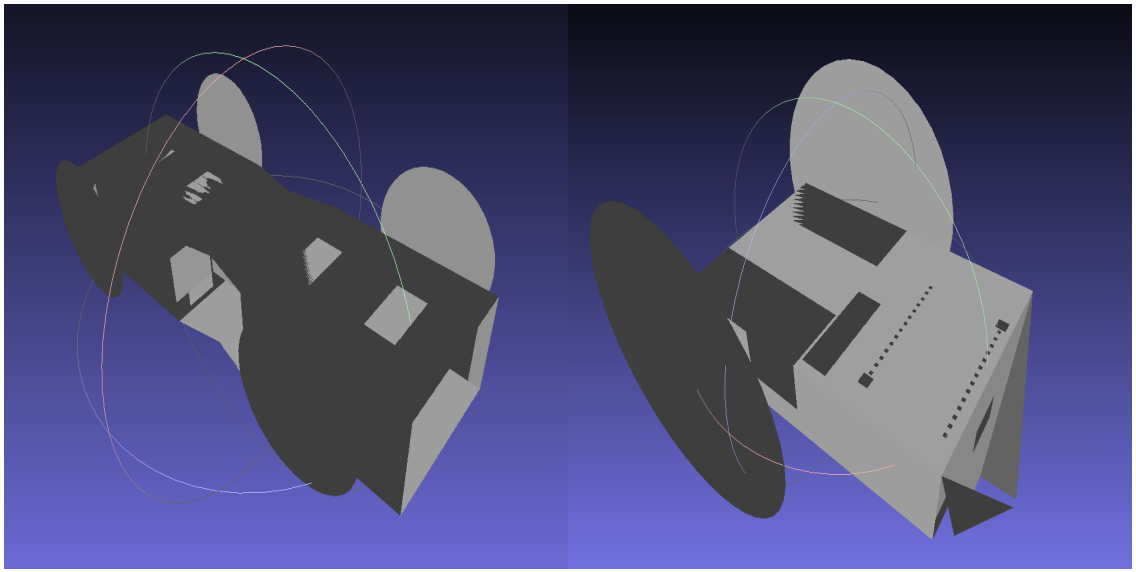


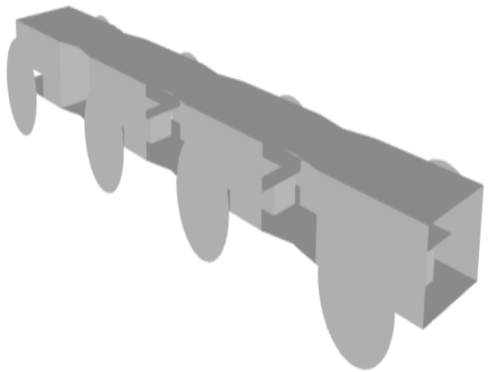
# RoCo [1] in the pipeline



# Robot Morphologies

STL files - Visualizing the robot in a CAD software



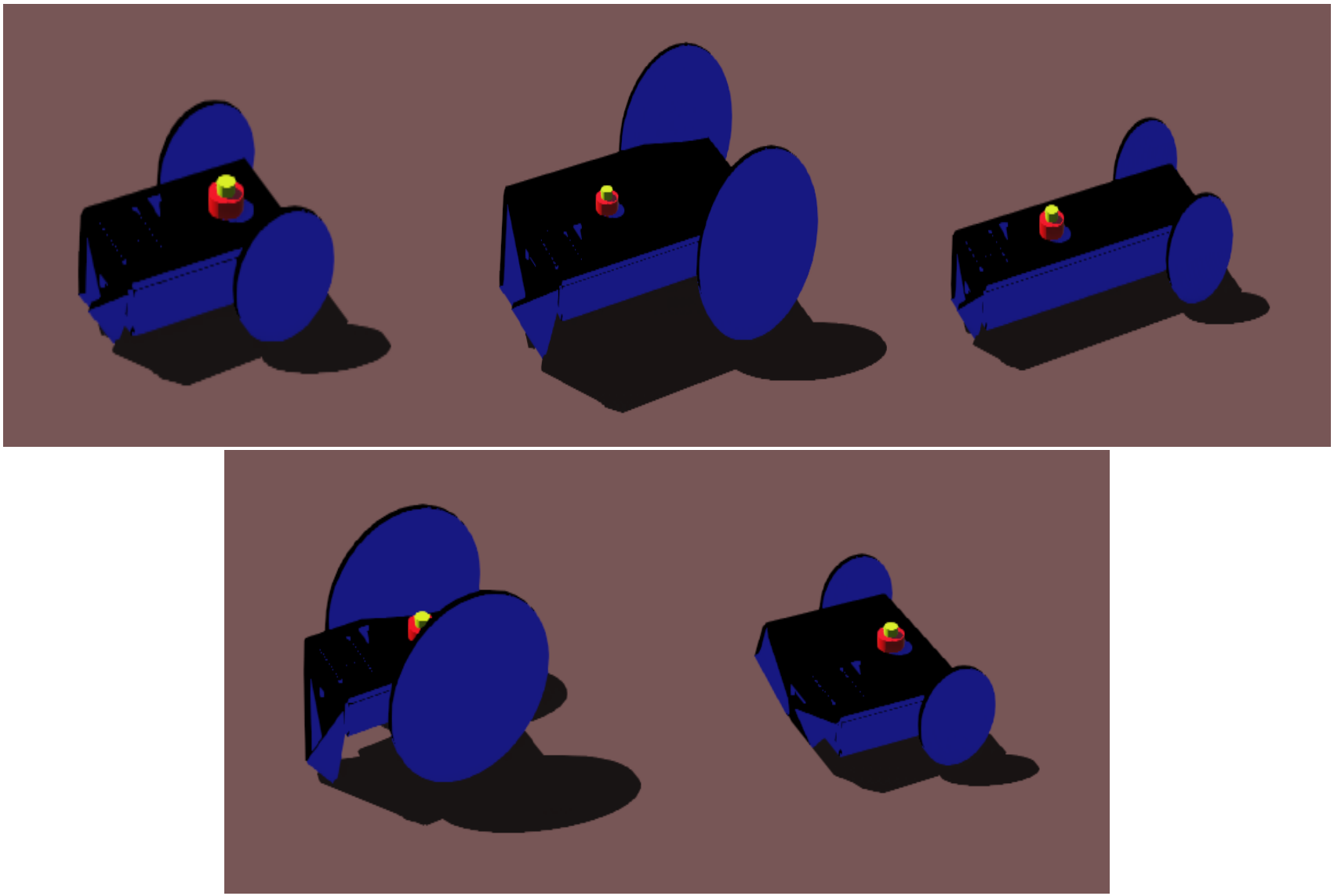


# Add Devices

- Compass
- GPS
- Motors
- Microcontroller

# Parameterized robots

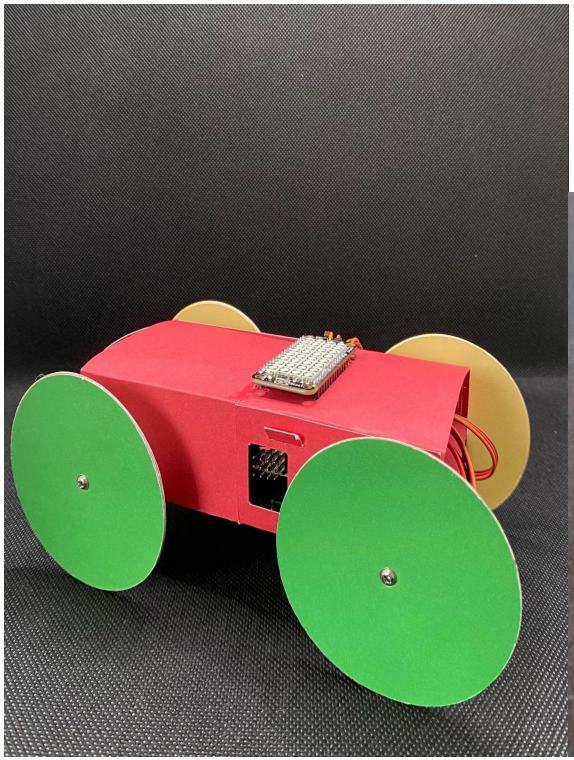
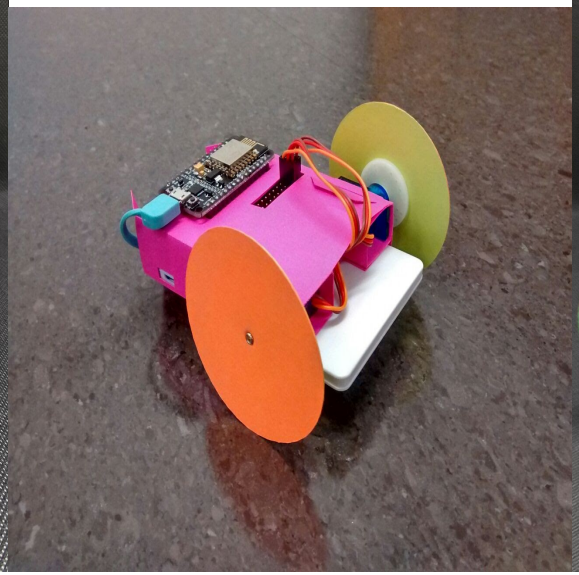
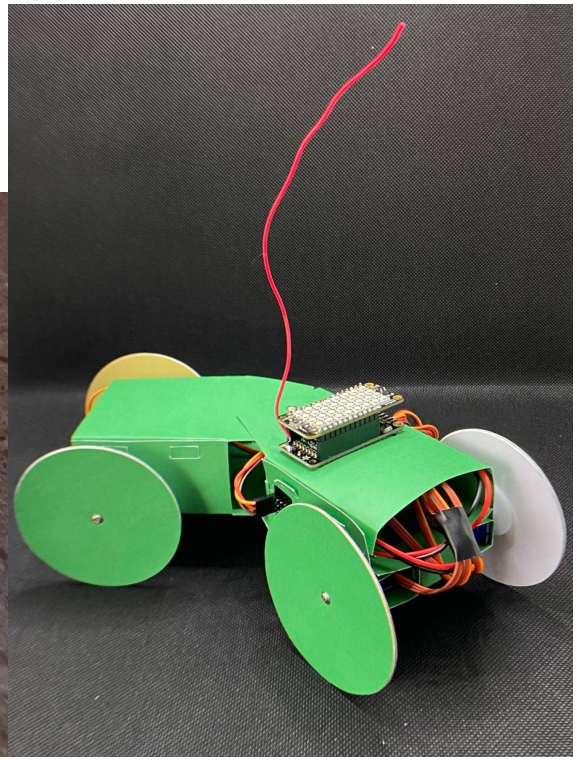
Parametrize a given robot morphology



# Design Parameters

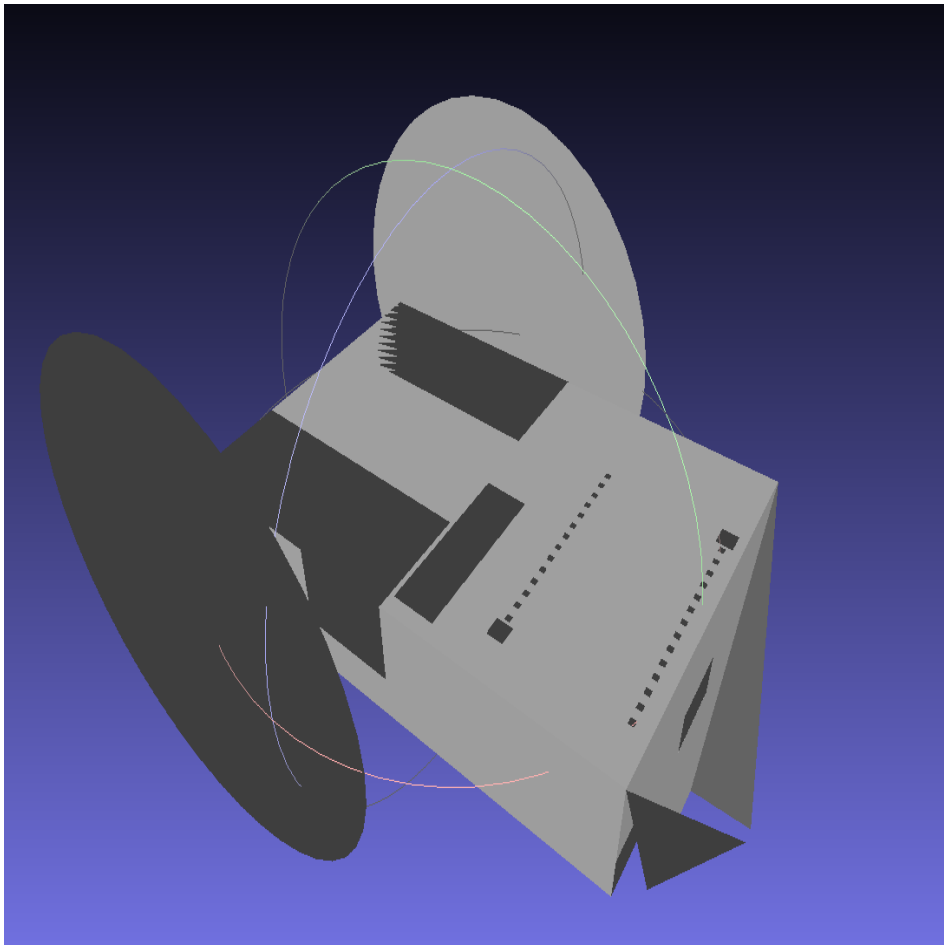
Change dimensions for any module on the robot - the change cascades down the design tree

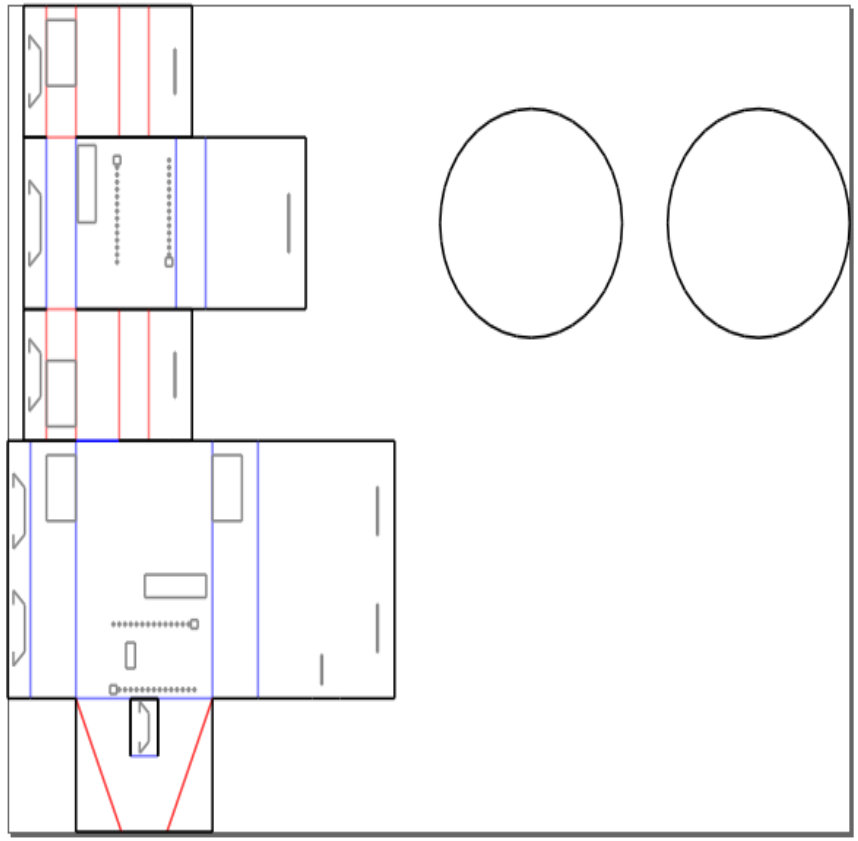
# Robot Morphologies

# Print and Fold

Each RoCo robot can be folded from a 2D pattern into a robot body!

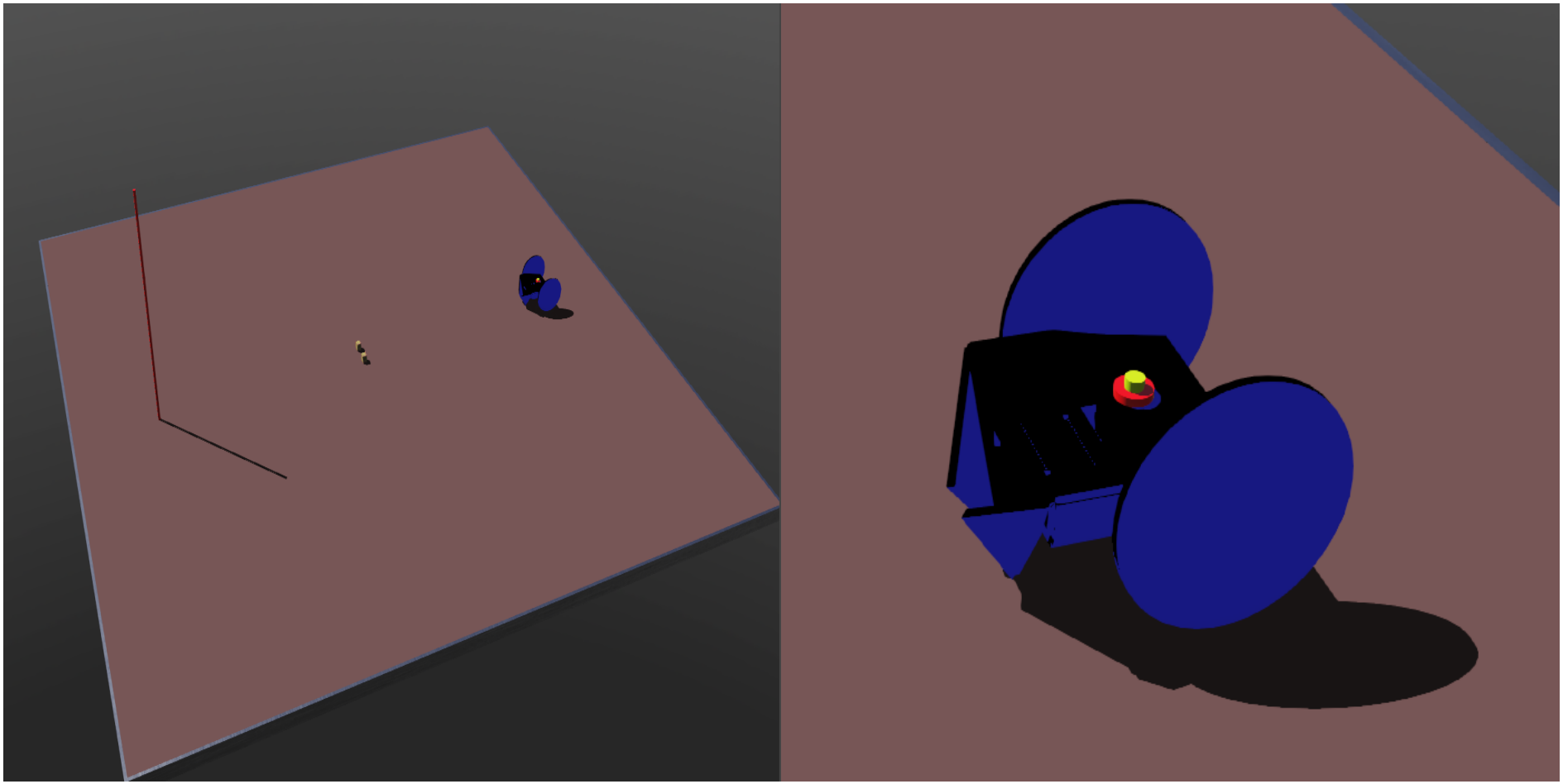




# Paperbot

A folded, functional Paperbot

# RoCo inputs to Webots (Physics Simulator)



# Environment Exploration Motivation

Roco allows us to explore the space of designs in a flexible modular way. We want something similar to explore environments.

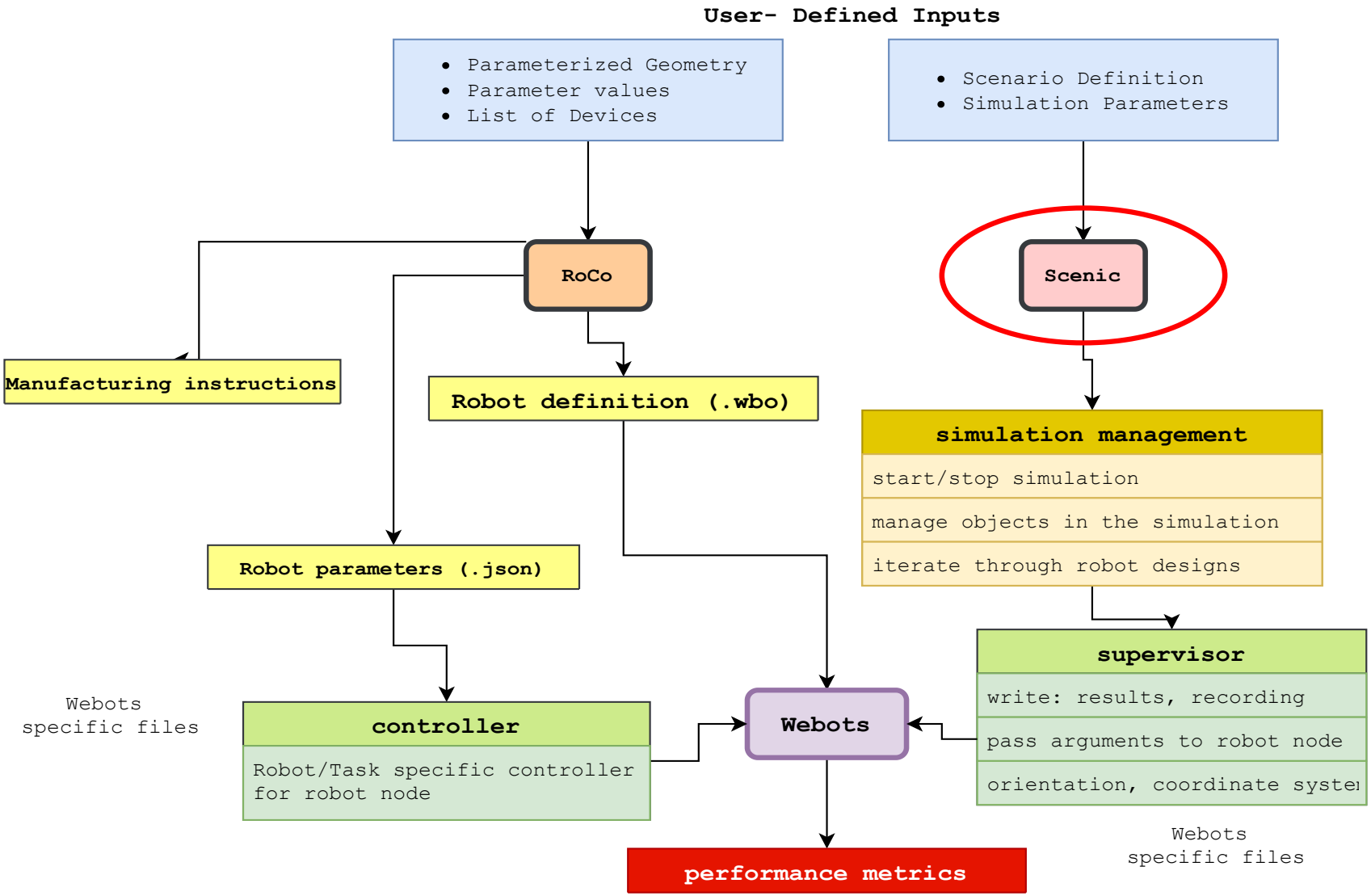
# Environment Exploration Motivation

Roco allows us to explore the space of designs in a flexible modular way. We want something similar to explore environments.

Ideally should:

- Quick to develop and tweak.
- Express not just one environment, but many with different variations
- Able to integrate with simulator to manage simulation and extract data

# Scenic in the pipeline



# Scenic Overview

- Scenic [2] is a probabilistic programming language for modeling the environments of cyber-physical systems

# Scenic Overview

- Scenic [2] is a probabilistic programming language for modeling the environments of cyber-physical systems
- A Scenic program defines a probabilistic distribution over physical objects and agents

# Scenic Overview

- Scenic [2] is a probabilistic programming language for modeling the environments of cyber-physical systems
- A Scenic program defines a probabilistic distribution over physical objects and agents
- Sampling yields concrete scenes of objects and agents which can be simulated to produce training or testing data

# Scenic Overview

- Scenic [2] is a probabilistic programming language for modeling the environments of cyber-physical systems
- A Scenic program defines a probabilistic distribution over physical objects and agents
- Sampling yields concrete scenes of objects and agents which can be simulated to produce training or testing data
- Can also dynamically monitor sampled simulations

# Scenic Overview

```
from gta import Car, curb, roadDirection

ego = Car

spot = OrientedPoint on visible curb
badAngle = Uniform(1.0, -1.0) * (10, 20) deg
Car left of (spot offset by -0.5 @ 0),
    facing badAngle relative to roadDirection
```

Figure 1: Scenic program describing a badly parked car and 4 scenes sampled from it, visualized in GTA simulator

# Design Pipeline: Scenic Environment

In design pipeline, Scenic is used to:

- Describe and generate environment

# Design Pipeline: Scenic Environment

In design pipeline, Scenic is used to:

- Describe and generate environment
- Record data at various points throughout simulation

# Design Pipeline: Scenic Environment

In design pipeline, Scenic is used to:

- Describe and generate environment
- Record data at various points throughout simulation
- Monitor the environment and terminate simulation when desired

# Speed Bumps Scenario

Toy scenario with 3 primary components:

- Place two speed bumps of variable size at a variable distance and fix Paperbot start location/target
- Require Paperbot remain in arena and terminate when near target or timeout is reached.
- At end of simulation, record test metadata.

# Speed Bumps Scenario

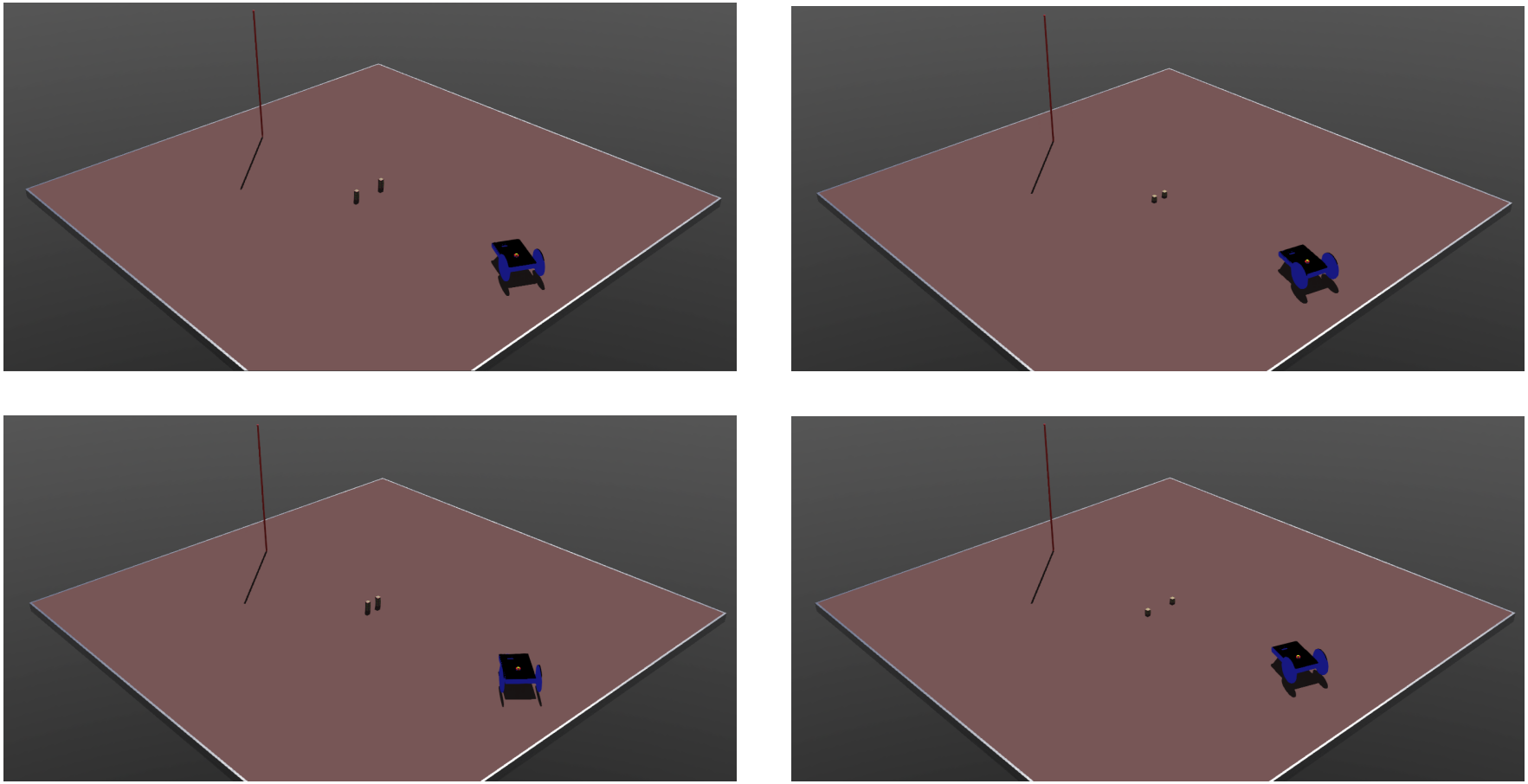


Figure 2: Various Instances of the Speed Bump Scenario in Webots

# Speed Bumps Scenario

Bare minimum scenario with very limited randomness. Designed to detect design choices that a human could also infer, such as...

# Speed Bumps Scenario

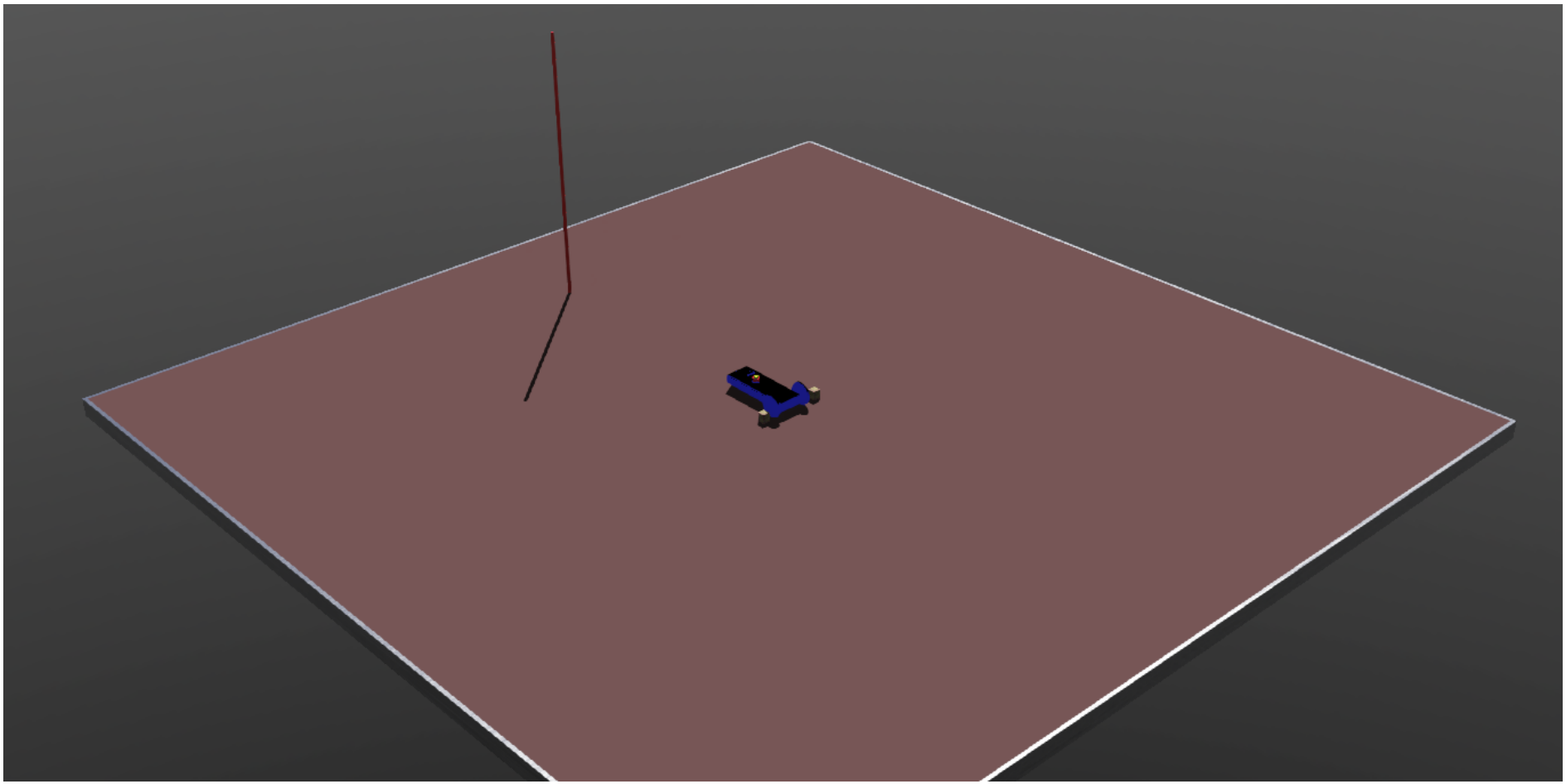


Figure 3: Thin robot can pass through speed bumps

# Speed Bumps Scenario

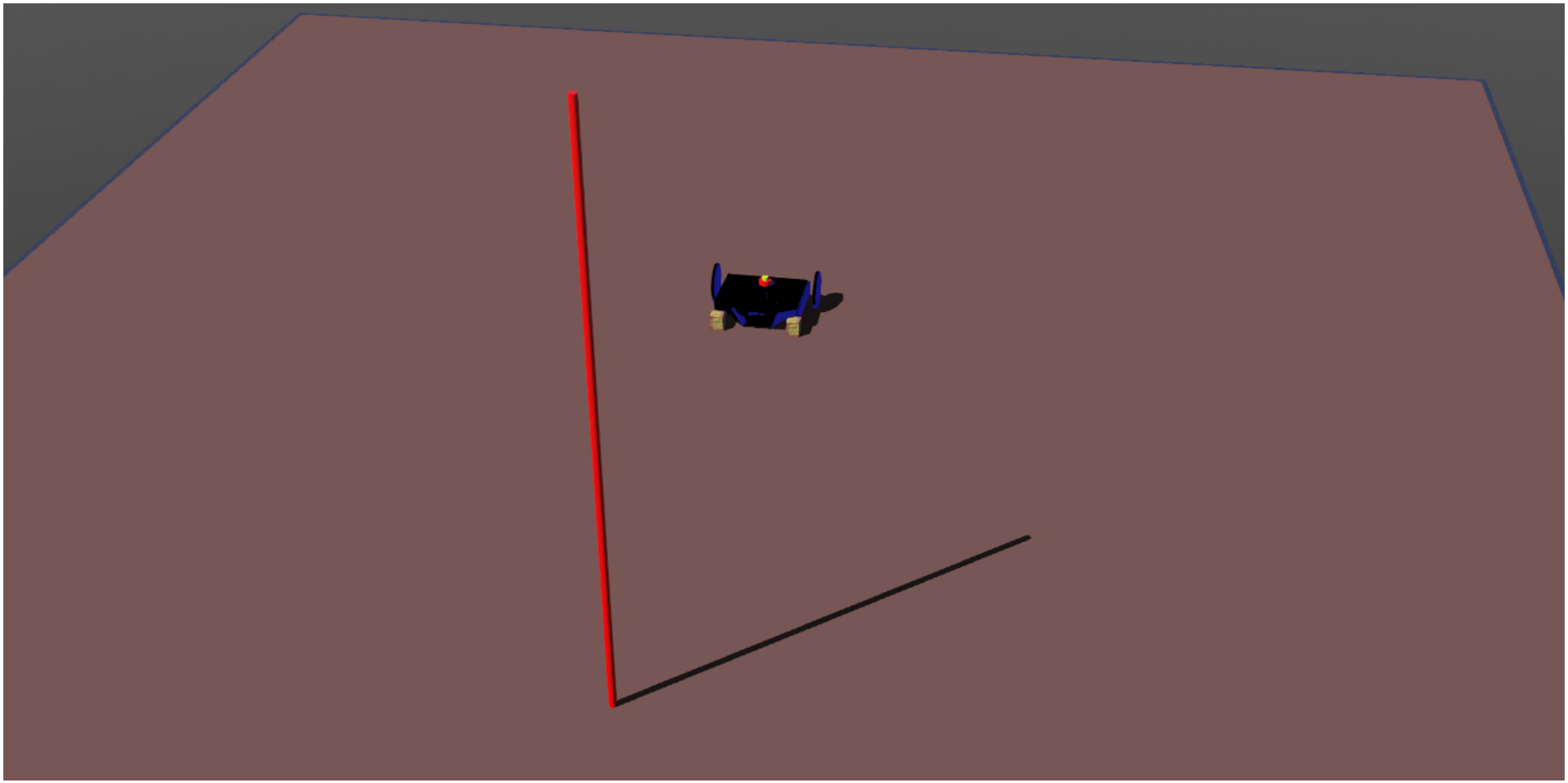


Figure 4: Wide robot will run into speed bumps

# Speed Bumps Scenario

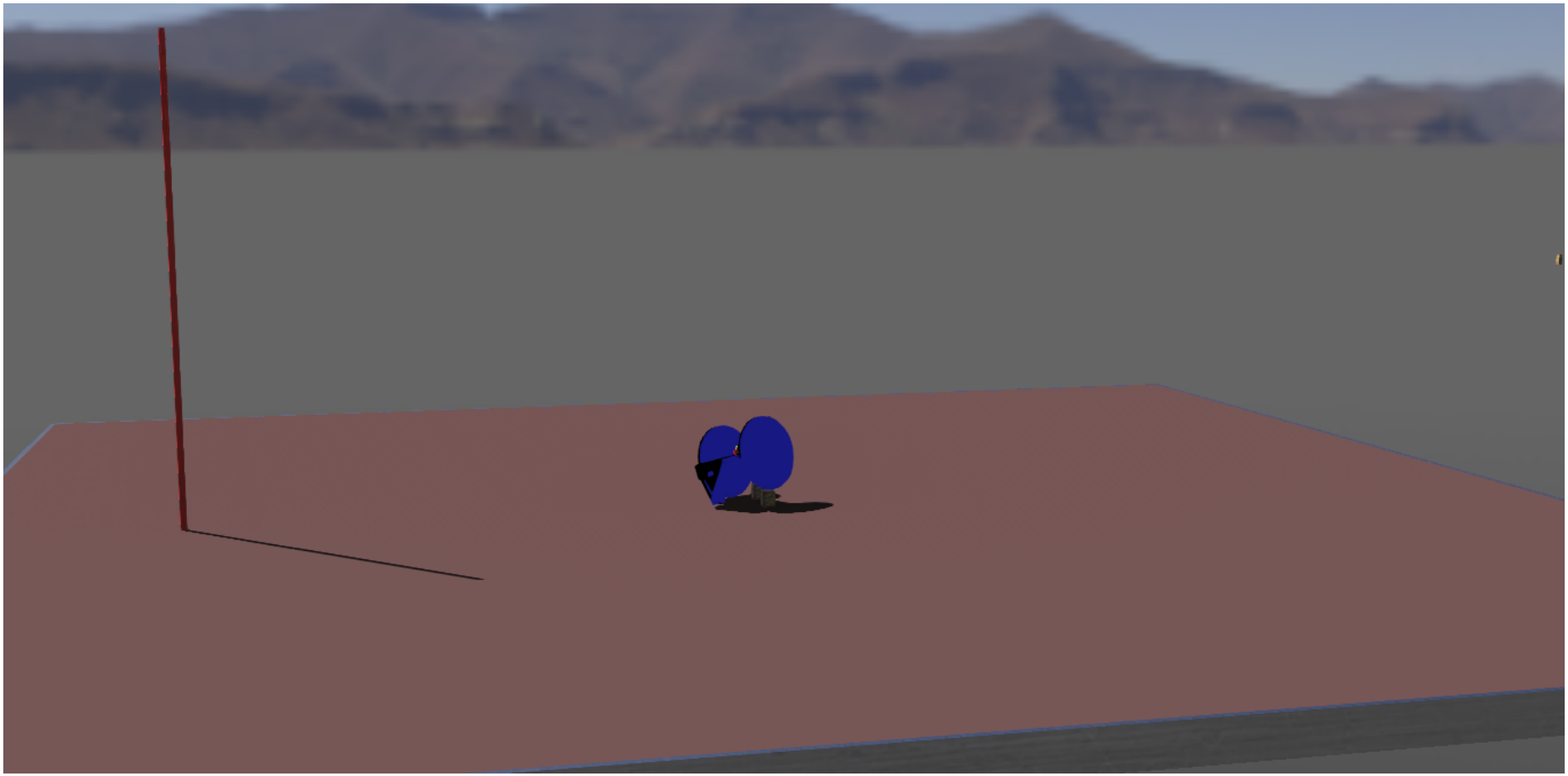


Figure 5: Big wheeled robot can roll over speed bumps

# Bumpy Field Scenario

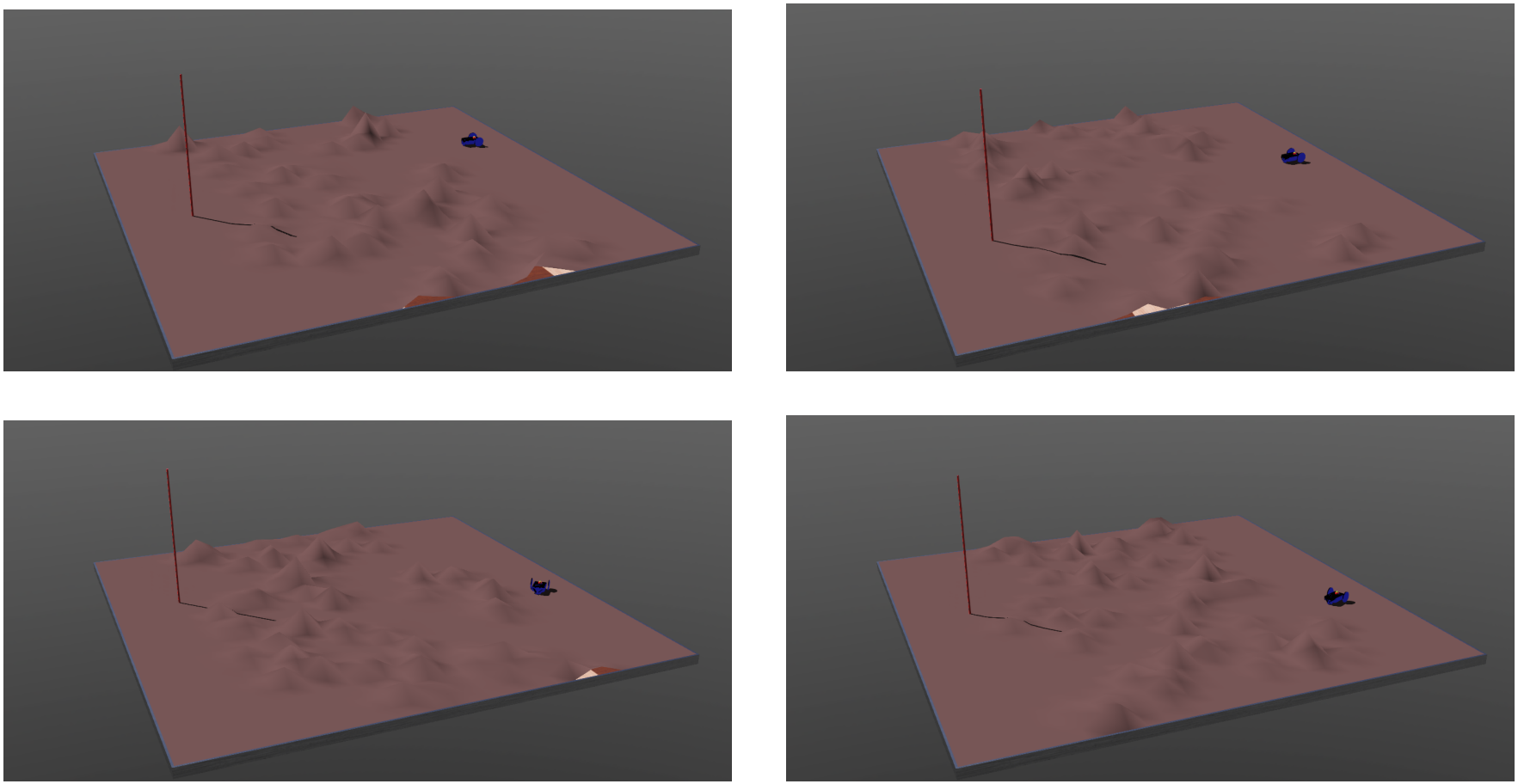


Figure 6: Various Instances of the Bumpy Field Scenario in Webots

# Bumpy Field Scenario

In the future, consider more chaotic scenario with substantial randomness, designed to detect features that a human could not immediately easily discern.

# Bumpy Field Scenario

In the future, consider more chaotic scenario with substantial randomness, designed to detect features that a human could not immediately easily discern.

Must balance:

- Wheel size for torque, speed, energy efficiency.
- Robot size and width to navigate through/over bumps

# Bumpy Field Scenario

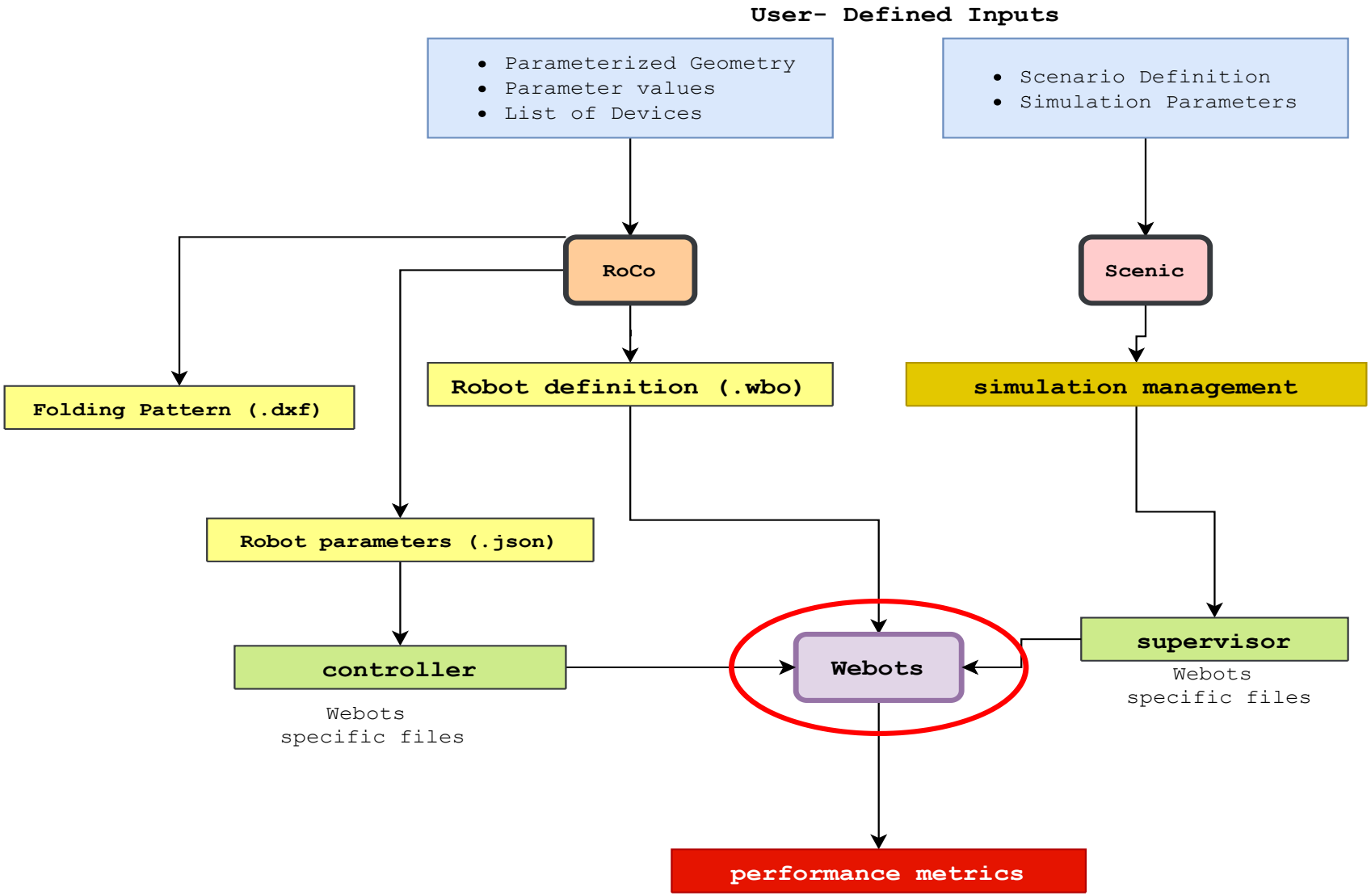
In the future, consider more chaotic scenario with substantial randomness, designed to detect features that a human could not immediately easily discern.

Must balance:

- Wheel size for torque, speed, energy efficiency.
- Robot size and width to navigate through/over bumps

The optimal parameters are not immediately obvious to a human, but can be explored via our pipeline.

# Webots in the pipeline



# Webots simulations

*Simulation Set 1*

**Objective** : Iterate through the design space of the robot in different scenarios in Webots [3]

- 2-Wheeled Car utilizing a differential-drive PID controller to reach a target
- Scenario has 2 speed bumps along the straight line path of the robot that does not allow cars with certain values of the geometrical parameters to pass through

# Simulations

Design Parameters

- Width (Wheelbase) range : (60,120) mm
- Height (Radius) range : (20,100) mm
- 5 simulations per design

Simulation environment parameters

- Simulation time limit = 40.0 s
- Distance between speed bumps range (0.05, 0.1, 0.15)m
- Height of speed bump range (0.025, 0.05) m

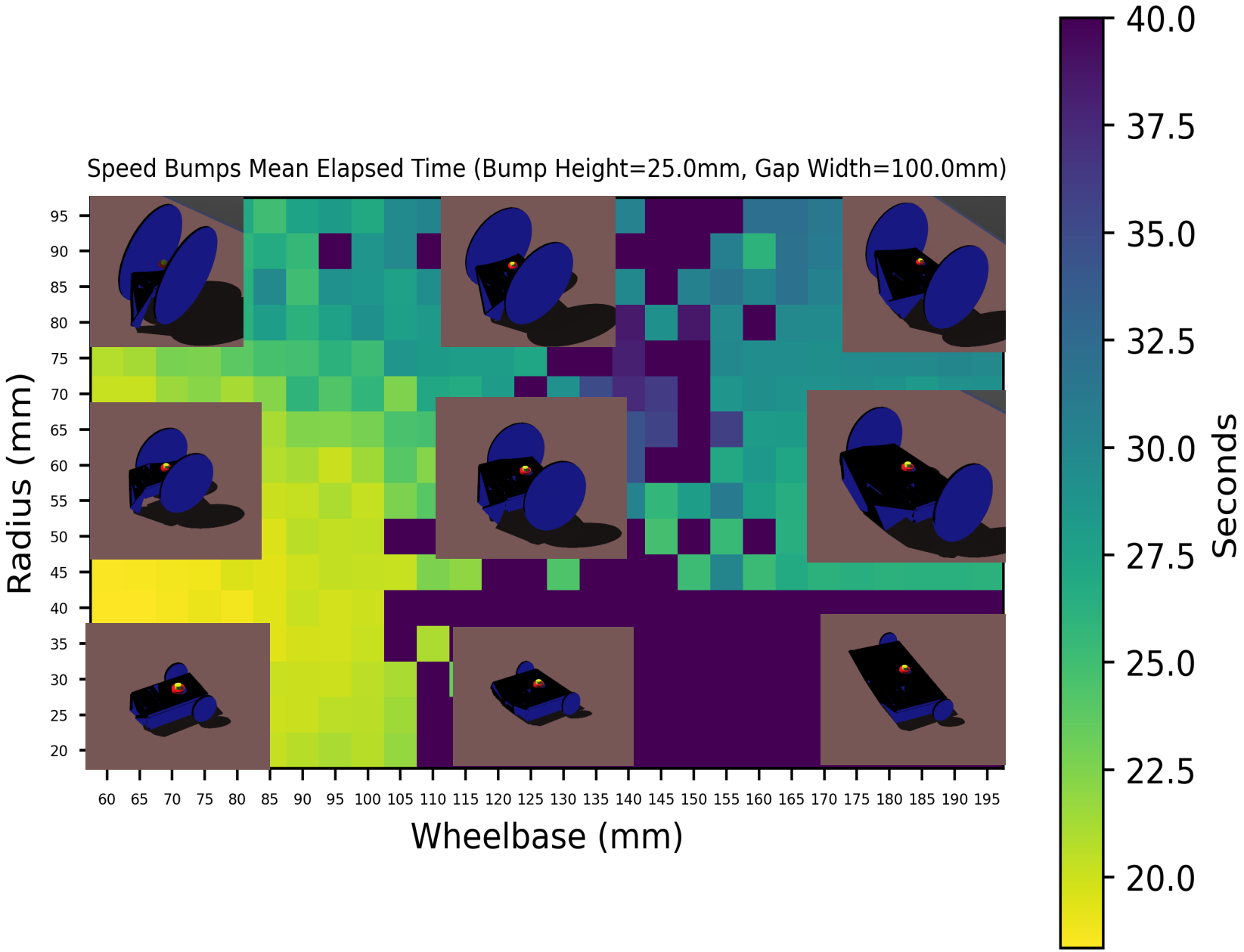
Test Results Metrics

- Distance to target: $< 0.15m \rightarrow$ Success
- Elapsed Time $< 40.0s \rightarrow$ Success (max simulation time)

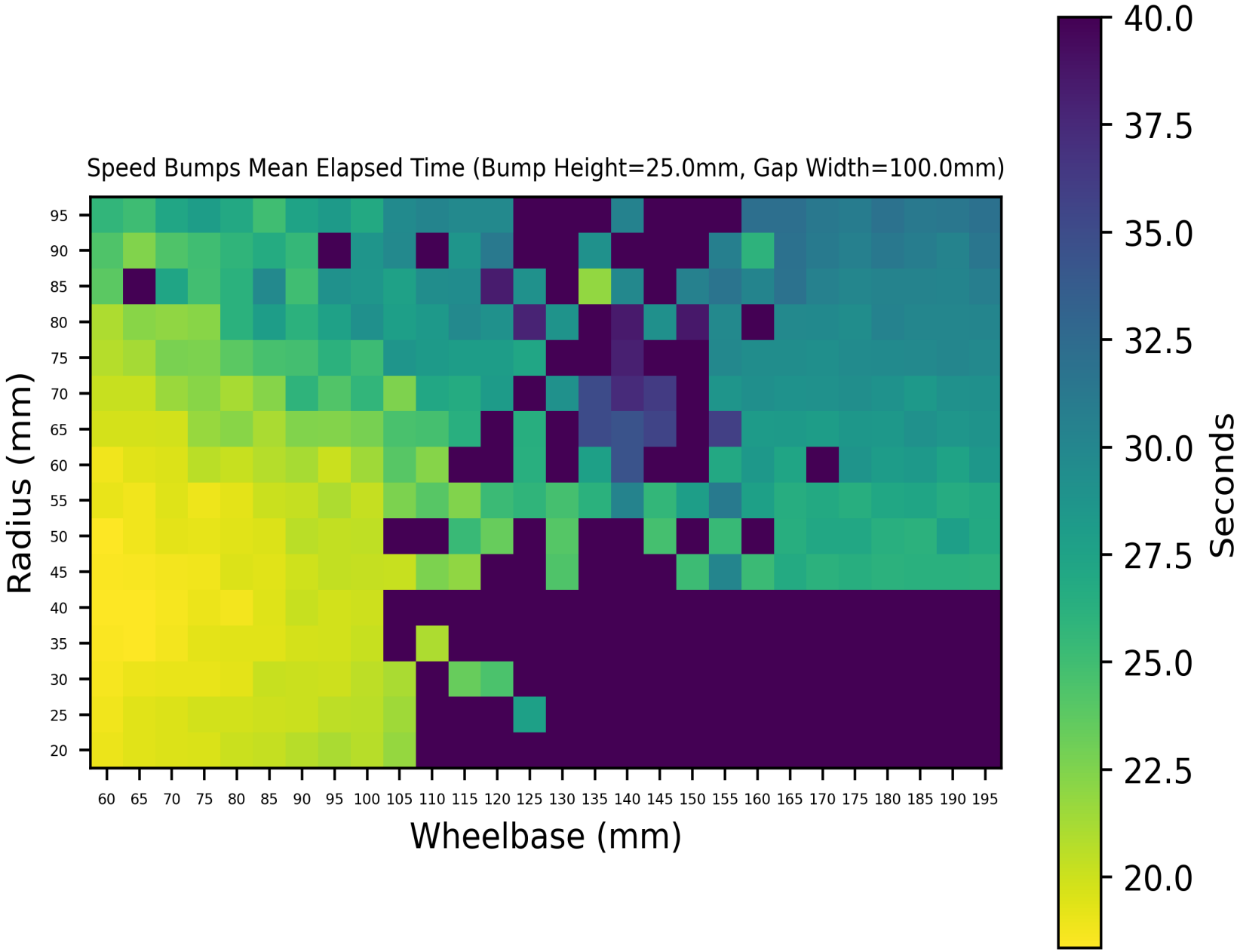
# Webots Simulations

https://www.youtube.com/watch?v=CqoAzELC3B0

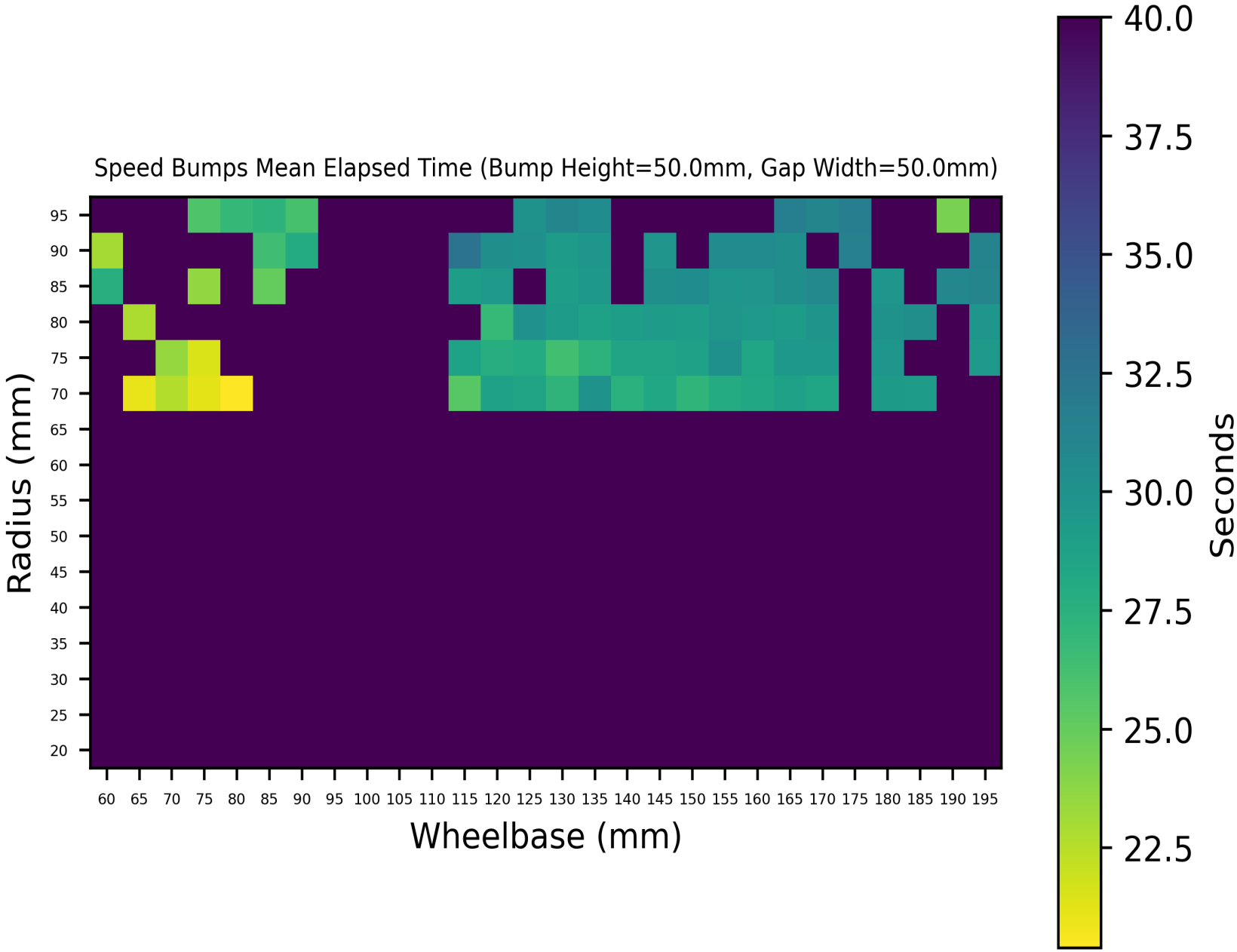
# Results: Radius of Wheel V/s Wheelbase Variation



1


Speed Bumps Mean Elapsed Time (Bump Height=25.0mm, Gap Width=100.0mm)
Radius (mm)
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195
Wheelbase (mm)
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
Seconds

2


Speed Bumps Mean Elapsed Time (Bump Height=50.0mm, Gap Width=50.0mm)
Radius (mm)
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195
Wheelbase (mm)
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
Seconds

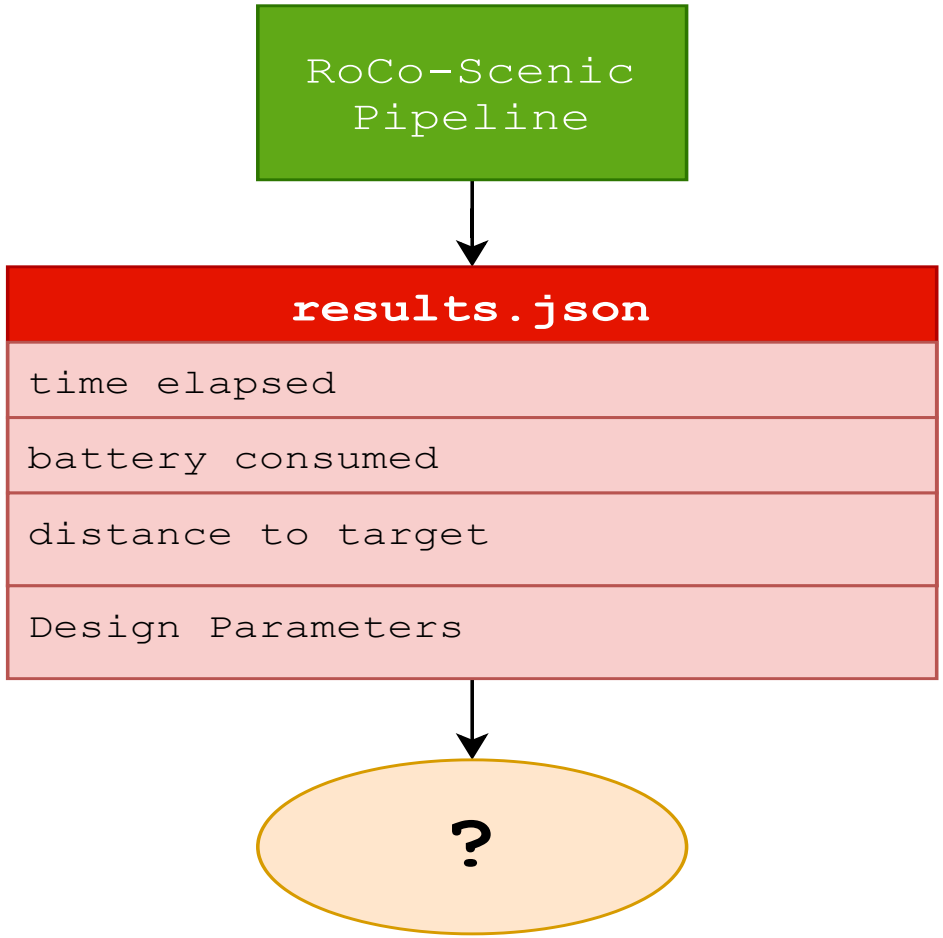
# Inferences

- These simulations were run keeping the length constant (100mm)
- They illustrate some regions of optimal parameters that might not be immediately obvious to a designer.

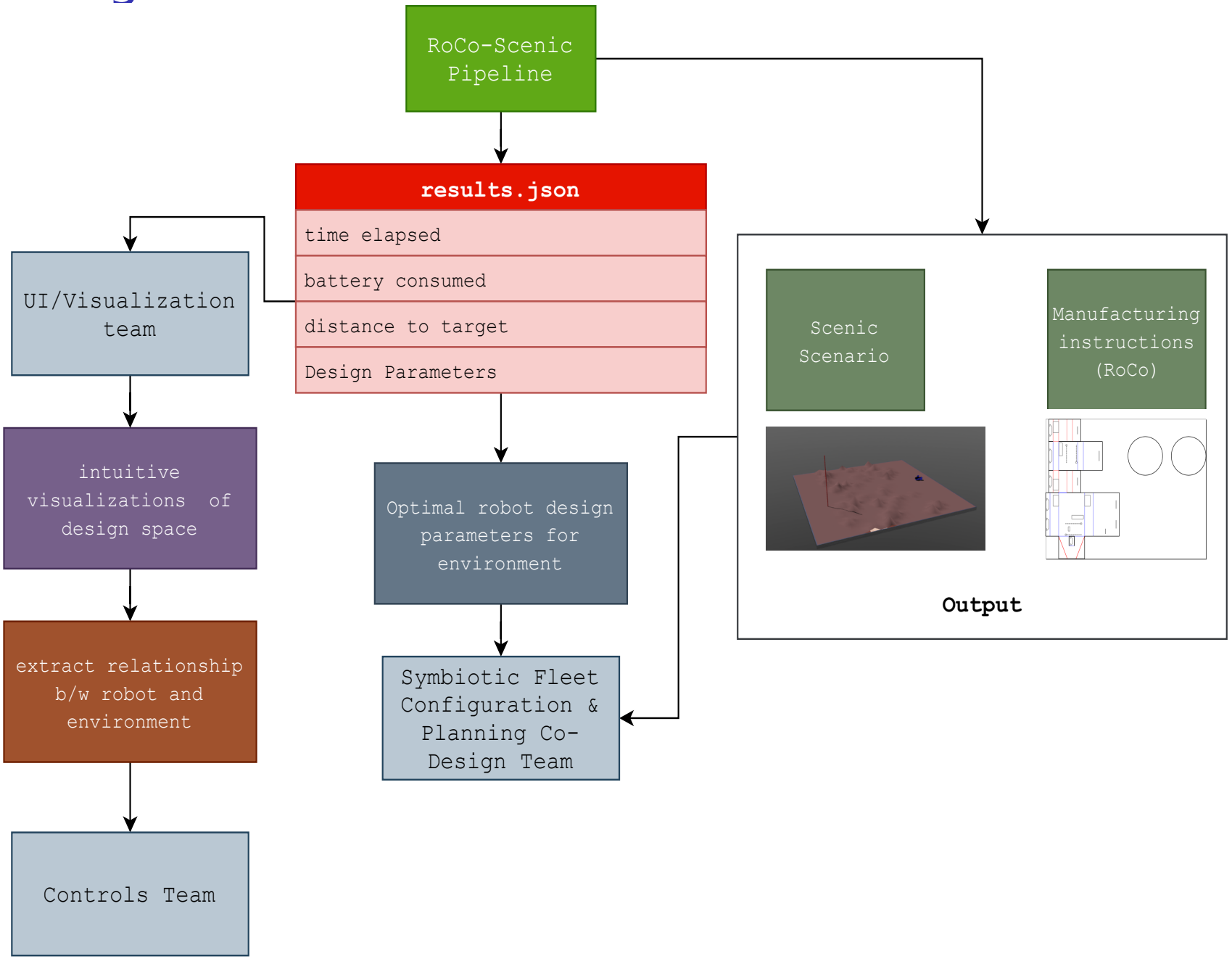
# Inferences

- Larger wheels would be thought of as uniformly good, except with a small wheelbase and length (proportionally) they often end up failing
- Yellow regions are the best , but even green ones could be useful, since they still succeed, and can help point toward useful combinations as well, depending on our error threshold, wrt the performance metrics used (slow and reliable vs fast and unreliable)

# Oracle generated results: What next?



# Oracle generated results: What next?



# References


[1] A. Mehta, N. Bezzo, P. Gebhard, B. An, V. Kumar, I. Lee, and D. Rus, “A design environment for the rapid specification and fabrication of printable robots,” in *Experimental Robotics*, pp. 435–449, Springer, 2016.

[2] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, “Scenic: a language for scenario specification and scene generation,” in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 63–78, 2019.

[3] Webots, “http://www.cyberbotics.com.”

Open-source Mobile Robot Simulation Software.