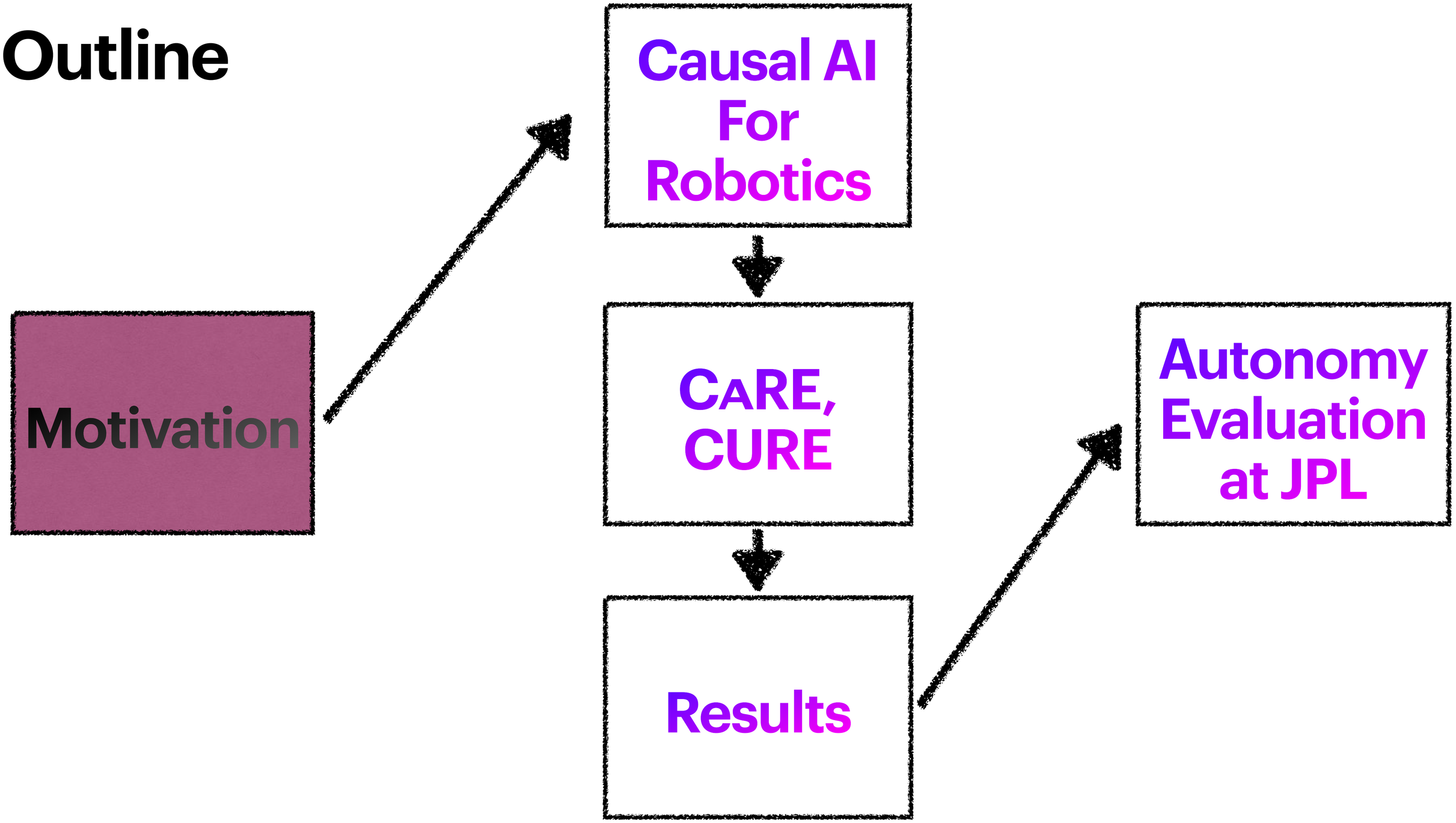
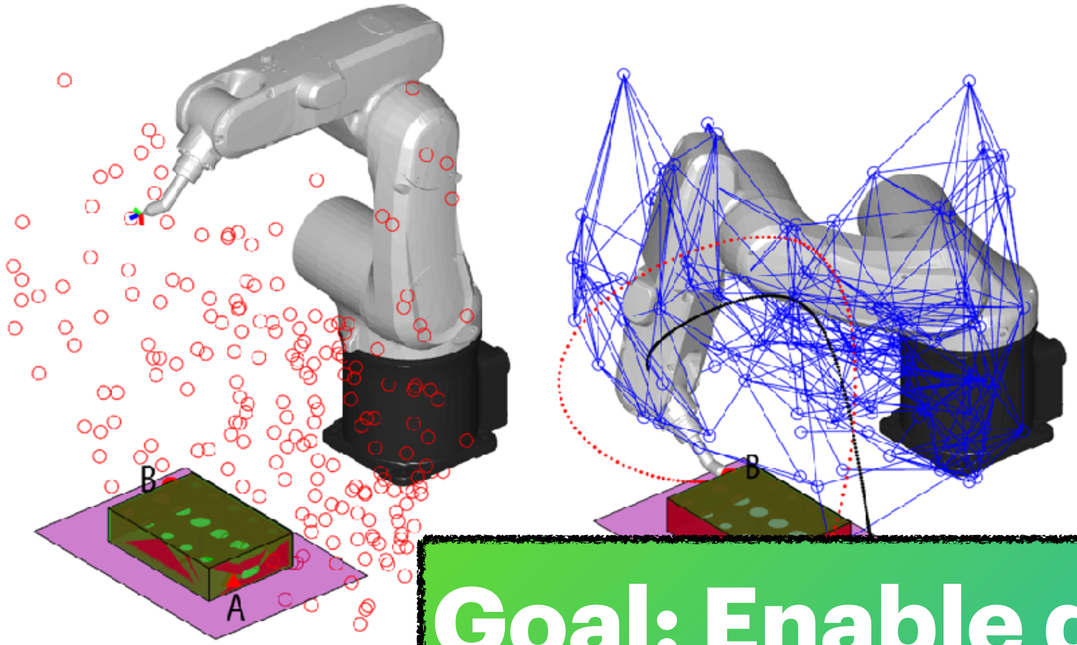
Performance Modeling, Debugging, and Optimization of Highly Configurable Robotic Systems with Causal AI

A Path towards Building Dependable Robotic Systems

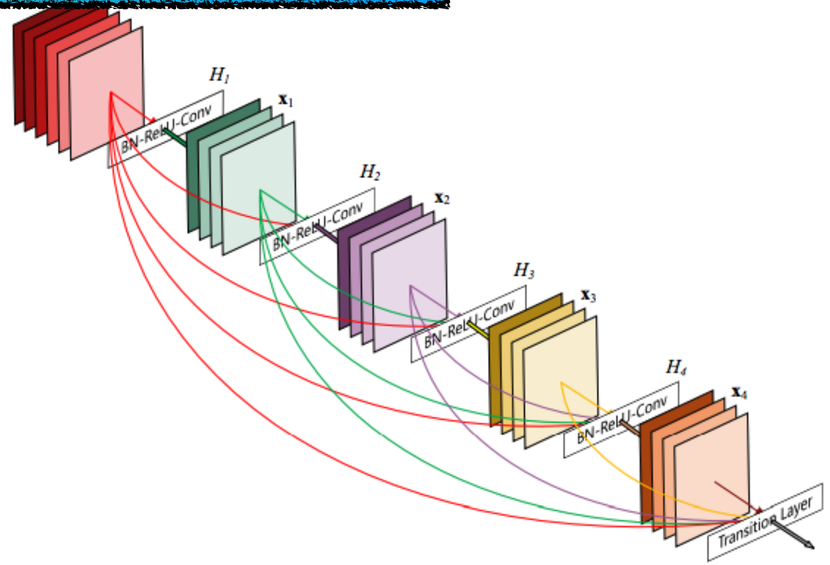
Md Abir Hossen
University of South Carolina

Outline


Causal AI For Robotics
Motivation
CaRE, CURE
Autonomy Evaluation at JPL
Results

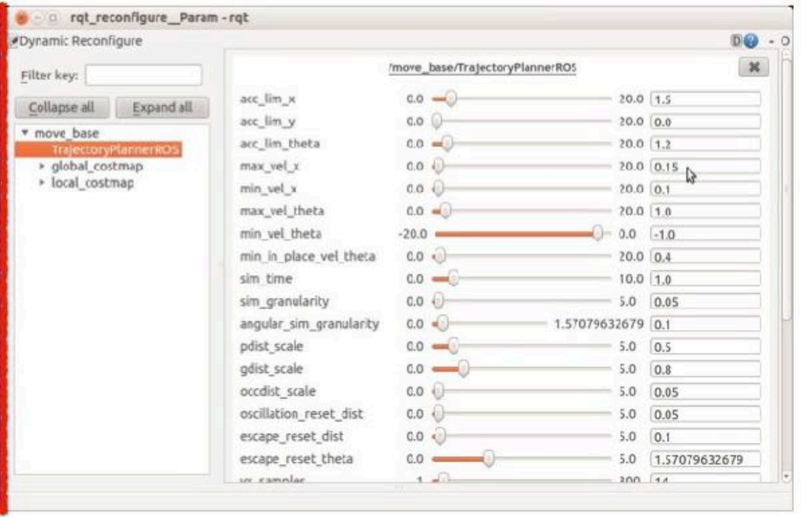


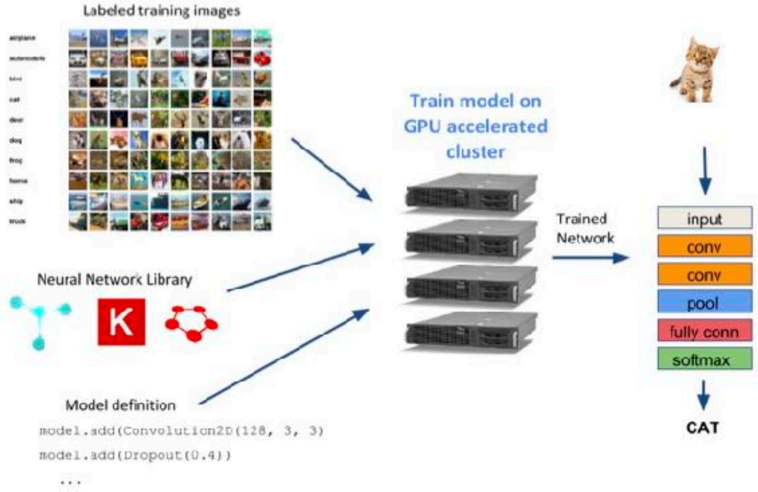
Goal: Enable developers/users to find the right quality tradeoff

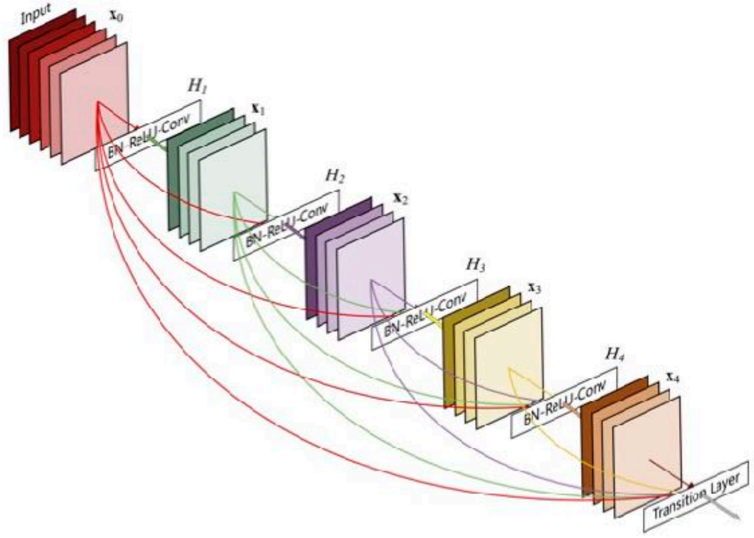


Today's most popular systems are built configurable

```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

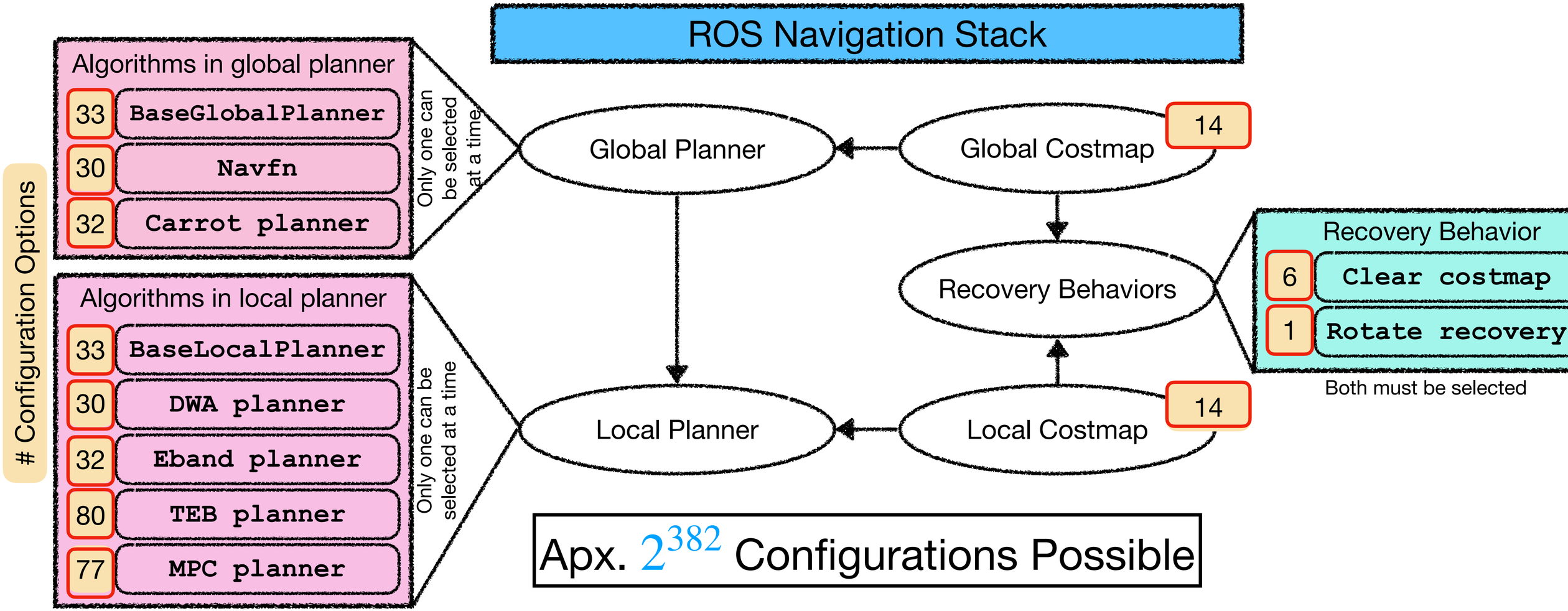




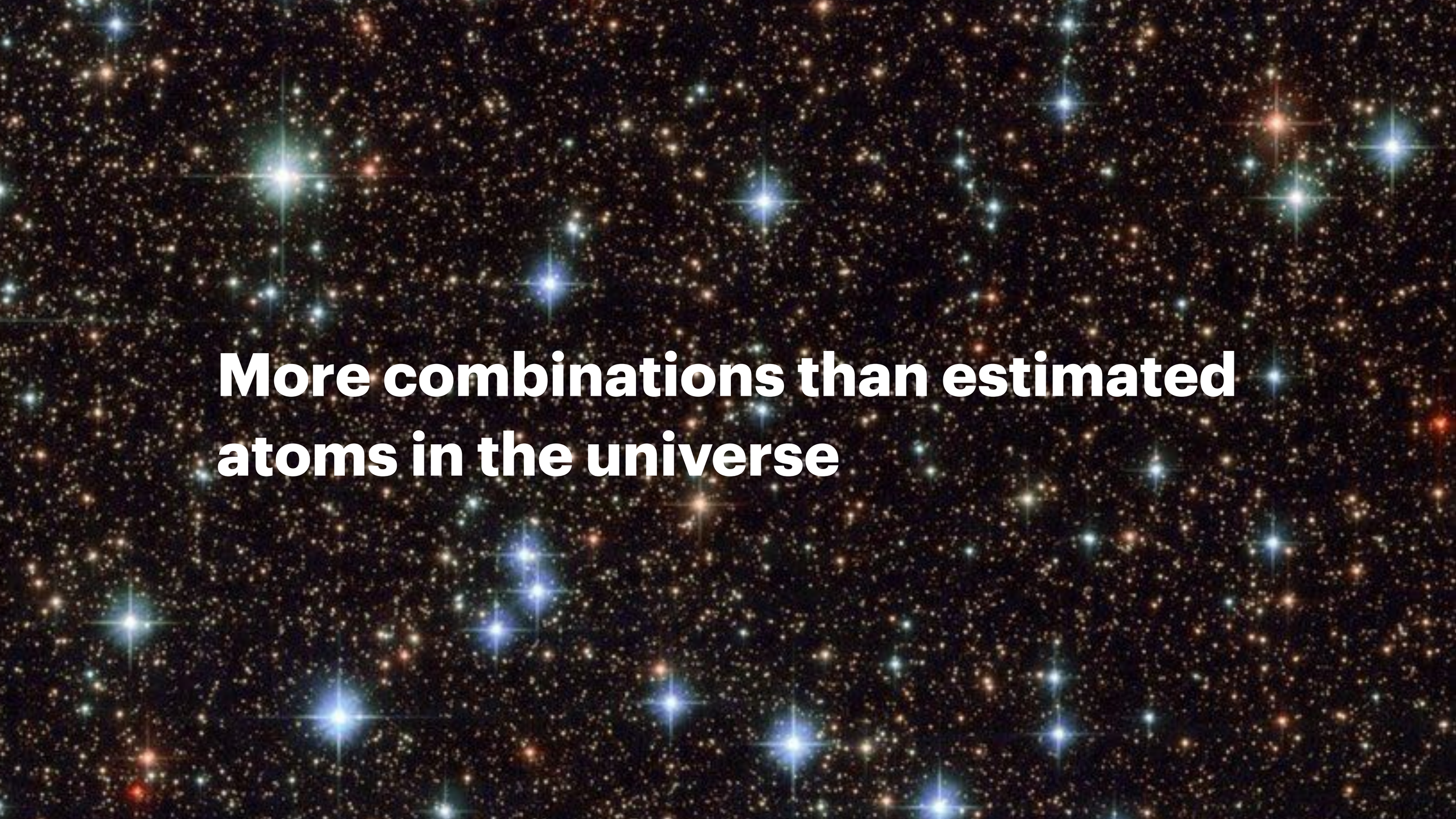


```yaml
# Goal Tolerance Parameters
yaw_goal_tolerance: 0.1
xy_goal_tolerance: 0.2
latch_xy_goal_tolerance: false

# Forward Simulation Parameters
sim_time: 2.0
sim_granularity: 0.02
angular_sim_granularity: 0.02
vx_samples: 6
vtheta_samples: 20
controller_frequency: 20.0

# Trajectory scoring parameters
meter_scoring: true # Whether the gdist_scale and pdist_scale parameters should assume that goal_distance and path_distance a
occdist_scale:  0.1 #The weighting for how much the controller should attempt to avoid obstacles. default 0.01
pdist_scale: 0.75  #     The weighting for how much the controller should stay close to the path it was given . default 0.6
gdist_scale: 1.0 #     The weighting for how much the controller should attempt to reach its local goal, also controls speed

heading_lookahead: 0.325  #How far to look ahead in meters when scoring different in-place-rotation trajectories
heading_scoring: false  #Whether to score based on the robot's heading to the path or its distance from the path. default fa
heading_scoring_timestep: 0.8   #How far to look ahead in time in seconds along the simulated trajectory when using heading s
dwa: true #Whether to use the Dynamic Window Approach (DWA)_ or whether to use Trajectory Rollout
simple_attractor: false
publish_cost_grid_pc: true

# Oscillation Prevention Parameters
oscillation_reset_dist: 0.25 #How far the robot must travel in meters before oscillation flags are reset (double, default: 0.
escape_reset_dist: 0.1
escape_reset_theta: 0.1
```

Configuration Space in Robotic Systems



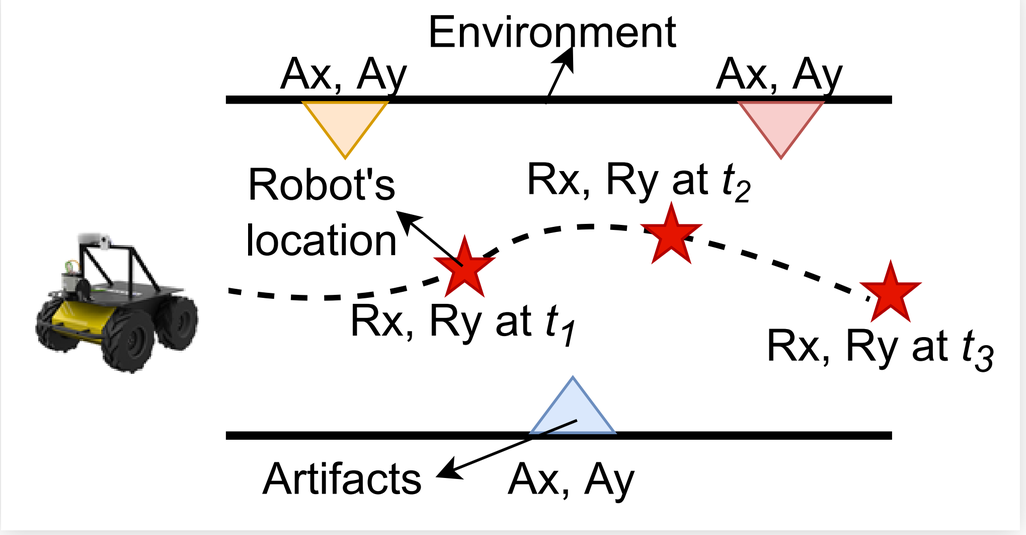
Complex interactions between options (intra or inter components) give rise to a combinatorially large configuration space.


More combinations than estimated
atoms in the universe

Motivating Scenarios

- Robot stops *0.5 m* away from the target location and transmits incorrect artifact locations to the control station.
- A cause for this fault might be a delay in data transformation (the sensor transmits data at a rate of *1 Hz* and the robot travels at *0.5 m/s).*
- As a result, when the *costmap* receives data from the sensor, it is a second old and the robot has already traveled *0.5 m* away from that position.

Delay in data transform results in a functional fault

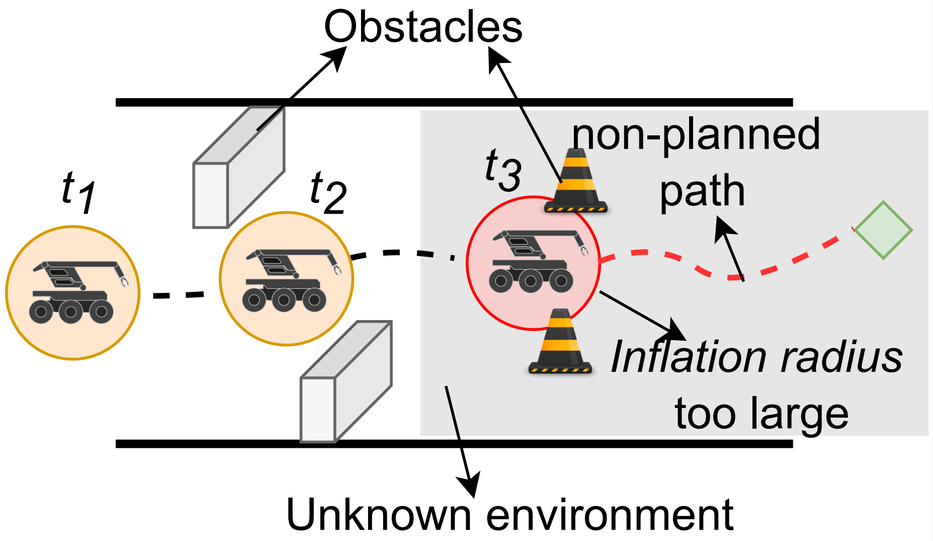


Demo

Motivating Scenarios

- At t_3 the robot encounters unique obstacles that too close together violating the **`inflation radius`**.
- Resulting in an indecisive robot that is stuck in place.

Change in environment results in an indecisive robot



Demo

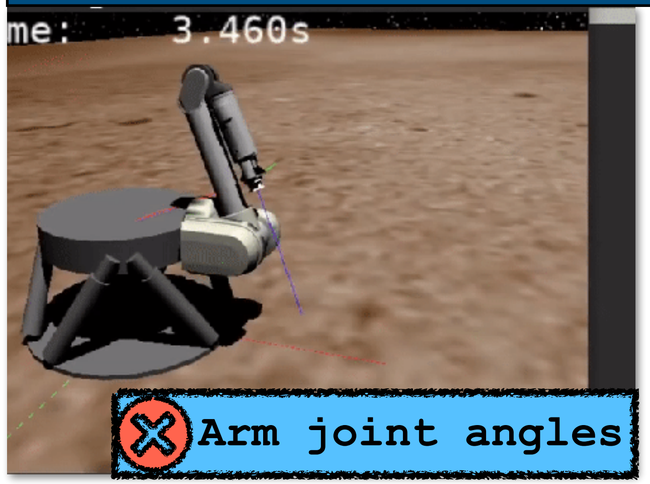
Motivating Scenarios

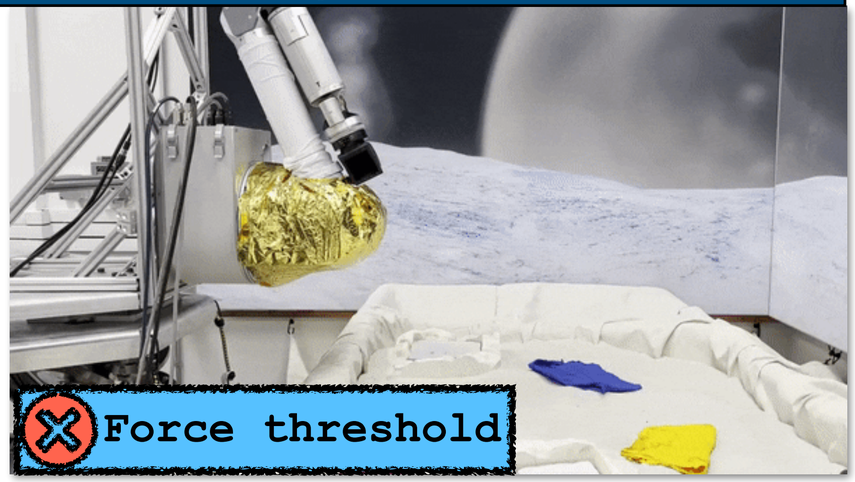
Turtlebot 3

Husky UGV

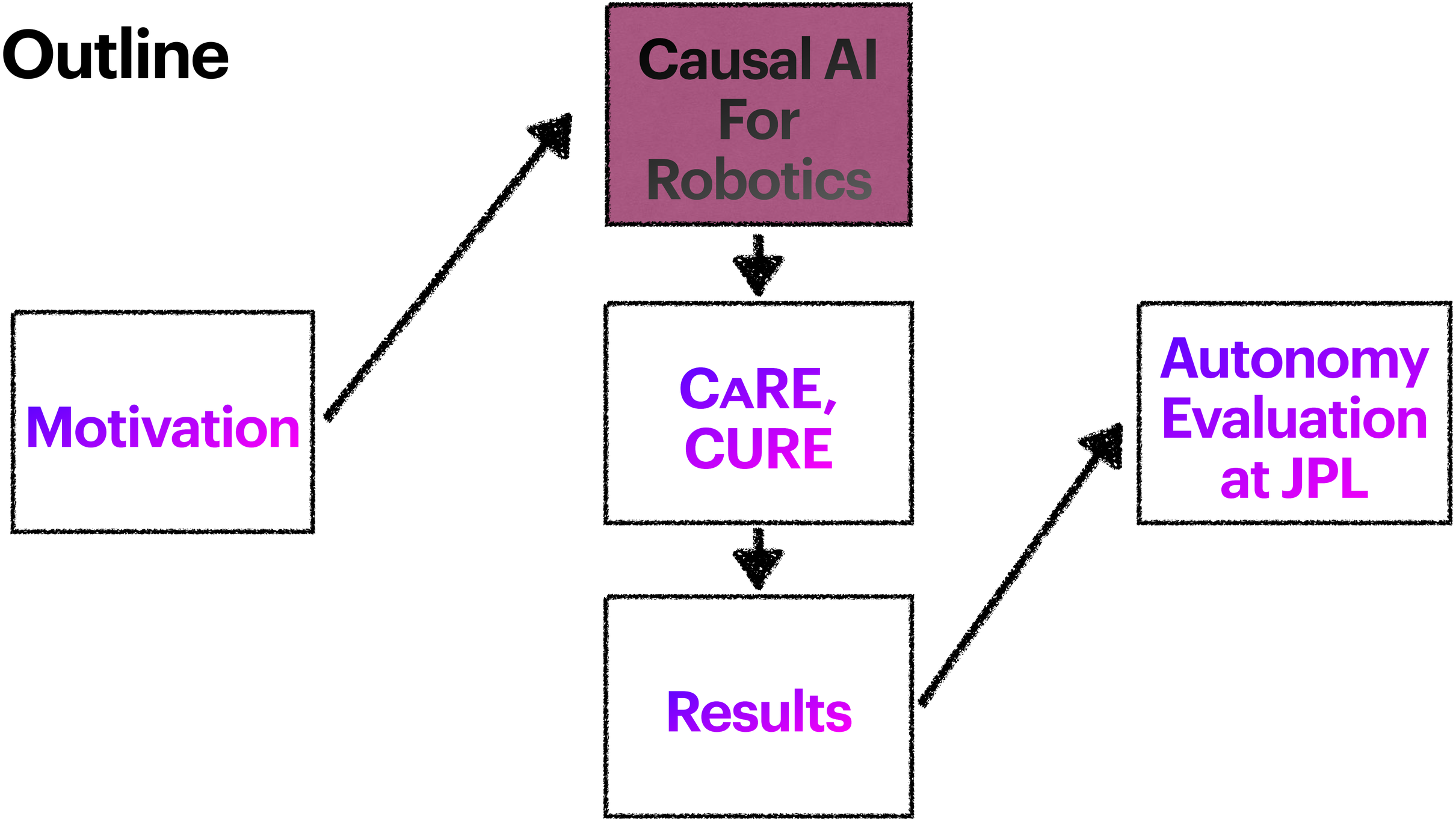
Husky UGV

Ocean World Lander Autonomy Testbed





Outline


Causal AI For Robotics
Motivation
CARE, CURE
Results
Autonomy Evaluation at JPL

Causal AI in Systems and Software

Robotics

IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 7, NO. 4, OCTOBER 2022

Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures

Maximilian Diehl, *Graduate Student Member, IEEE*, and Karinne Ramirez-Amaro, *Member, IEEE*

UAV

SWARMBUG: Debugging Configuration Bugs in Swarm Robotics

Chijung Jung
University of Virginia
Charlottesville, Virginia, USA
cj5kd@virginia.edu

Ali Ahad
University of Virginia
Charlottesville, Virginia, USA
aa5rn@virginia.edu

Jinho Jung
Georgia Institute of Technology
Atlanta, Georgia, USA
jinho.jung@gatech.edu

Sebastian Elbaum
University of Virginia
Charlottesville, Virginia, USA
selbaum@virginia.edu

Yonghwi Kwon
University of Virginia
Charlottesville, Virginia, USA
yongkwon@virginia.edu

Software Engineering

Causal Testing: Understanding Defects' Root Causes

Brittany Johnson
University of Massachusetts Amherst
Amherst, MA, USA
bjohnson@cs.umass.edu

Yuriy Brun
University of Massachusetts Amherst
Amherst, MA, USA
brun@cs.umass.edu

Alexandra Meliou
University of Massachusetts Amherst
Amherst, MA, USA
ameli@cs.umass.edu

Operating System

COZ: Finding Code that Counts with Causal Profiling

Charlie Curtsinger Emery D. Berger
School of Computer Science
University of Massachusetts Amherst
{charlie,emery}@cs.umass.edu

Big Data

Causality-Guided Adaptive Interventional Debugging

Anna Fariha
University of Massachusetts Amherst
afariha@cs.umass.edu

Suman Nath
Microsoft Research
Suman.Nath@microsoft.com

Alexandra Meliou
University of Massachusetts Amherst
ameli@cs.umass.edu

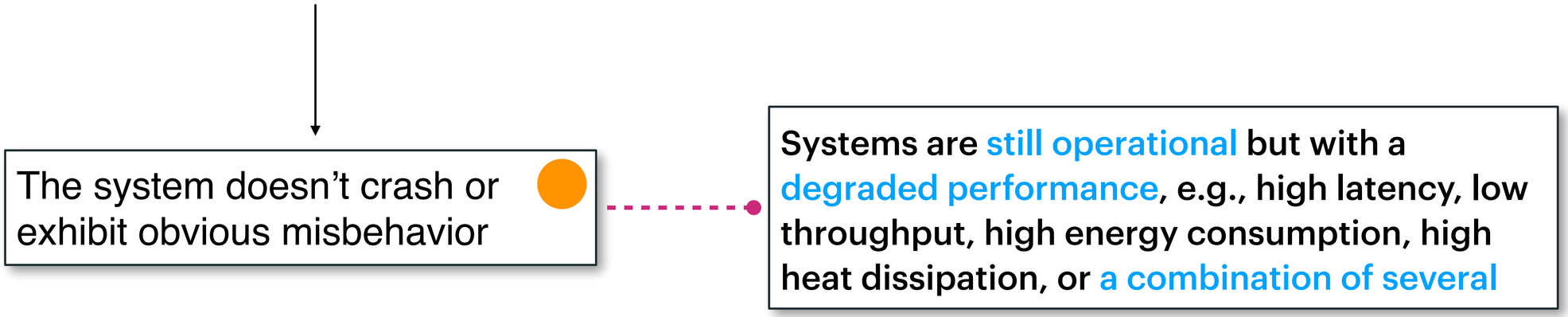
Database

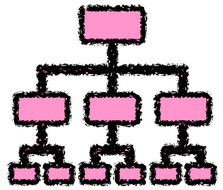
Demonstration of Inferring Causality from Relational Databases with CaRL

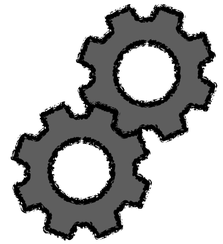
Moe Kayali, Babak Salimi, Dan Suciu
{kayali, bsalimi, suciu}@cs.washington.edu
University of Washington

Misconfiguration and its Effects

- Misconfigurations can elicit unexpected interactions between software and hardware
- These can result in non-functional faults
  - Affecting non-functional system properties like latency, throughput, energy consumption, etc.







My research goal is to improve performance by considering configuration level changes.

Motivating Example

https://github.com/ros-navigation/navigation2/issues/2439

Error while starting navigation using local planner(DWB) for hardwa

Closed **Soofiyan** opened this issue on Jul 7, 2021 · 5 comments

ros-navigation / **navigation2**

Soofiyan commented on Jul 7, 2021 • edited

Bug report

Required Info:

- Operating System:
  - Ubuntu 20.04
- ROS2 Version:
  - Foxy
- Version or commit hash:
  - Latest of foxy-devel branch(2nd July 2021)
- DDS implementation:

Steps to reproduce issue

```
This error is seen in simulation when the robot collides with obstacles(similar issue).
```

Actual behavior

Here the path is always failing and starts to spin, And the bot is not hitting any obstacle, and transformations are perfect. The robot is not even moving a little bit; it directly starts the recoveries servers as soon as a global planner is planned.

This user is trying to fix an error encountered by local planner. The local planner is not following the path planned by the global planner.

Expected behaviour

After giving a goal, the global planner is properly planning the path towards the goal. The expected behaviour is that the path must follow this path using a local planner.

The user expects the local planner to follow the path planned by the global planner.

Performance debugging aims at finding the root cause of the misconfiguration and fix it.

Motivating Example

ros-navigation / **navigation2**

SteveMacenski commented on Jul 7, 2021 • edited — Member

You can certainly tune the DWB controller into behavior where it has a hard time moving. Have you tuned the parameters of DWB and local costmap from the defaults in nav2_bringup? (not that those are perfect by any stretch either, but its a baseline we can use to test that hypothesis) The error also looks to be pretty straight forward: https://github.com/ros-planning/navigation2/search?q=Trajectory+goes+off, is your costmap reasonably sized? Please provide your configuration of DWB / local costmap. I also don't see a footprint in rviz, are you sure that is set correctly?

The developer is asking if the configuration of local planner was set correctly?

Soofiyan commented on Jul 7, 2021 — Author

Thank you for replying to my query. I have tuned this for almost 5-6 hours. Sometimes it is going towards the goal but still failing in the middle of the trajectory. 9/10 times it is still going into the recovery mode without hitting any obstacle, as mentioned earlier. If you can suggest any change in a parameter, it will be beneficial.
Here is my nav2_param file, which I am using

The user tuned the parameters for almost 5-6 hours, yet 9/10 times the robot is failing in the middle of the trajectory.

SteveMacenski commented on Jul 8, 2021 — Member

Please ask on ROS Answers -- I can't provide personalized tuning assistance to every user. If you identify a tangible bug or issue, I'm happy to discuss a resolution to that.

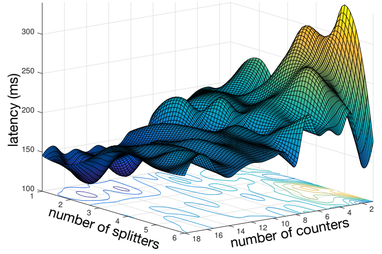
The developer concluded that he cannot provided personalized tuning assistance to every user.

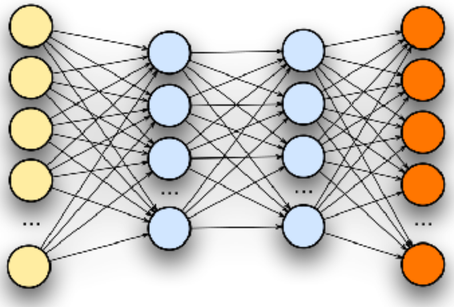
How to resolve these issues in an automatic manner?

Performance Influence Models

	Path dist bias	Transform tolerance	…	Traveled distance	…	Mission success
c_1	0.75	0.5	…	8.32 m	…	1
c_2	1.5	1.2	…	12.6 m	…	0
…	…	…	…	…	…	…
c_n	0.1	0.2	…	7.43 m	…	0

Observational Data





Black-box models

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon$$

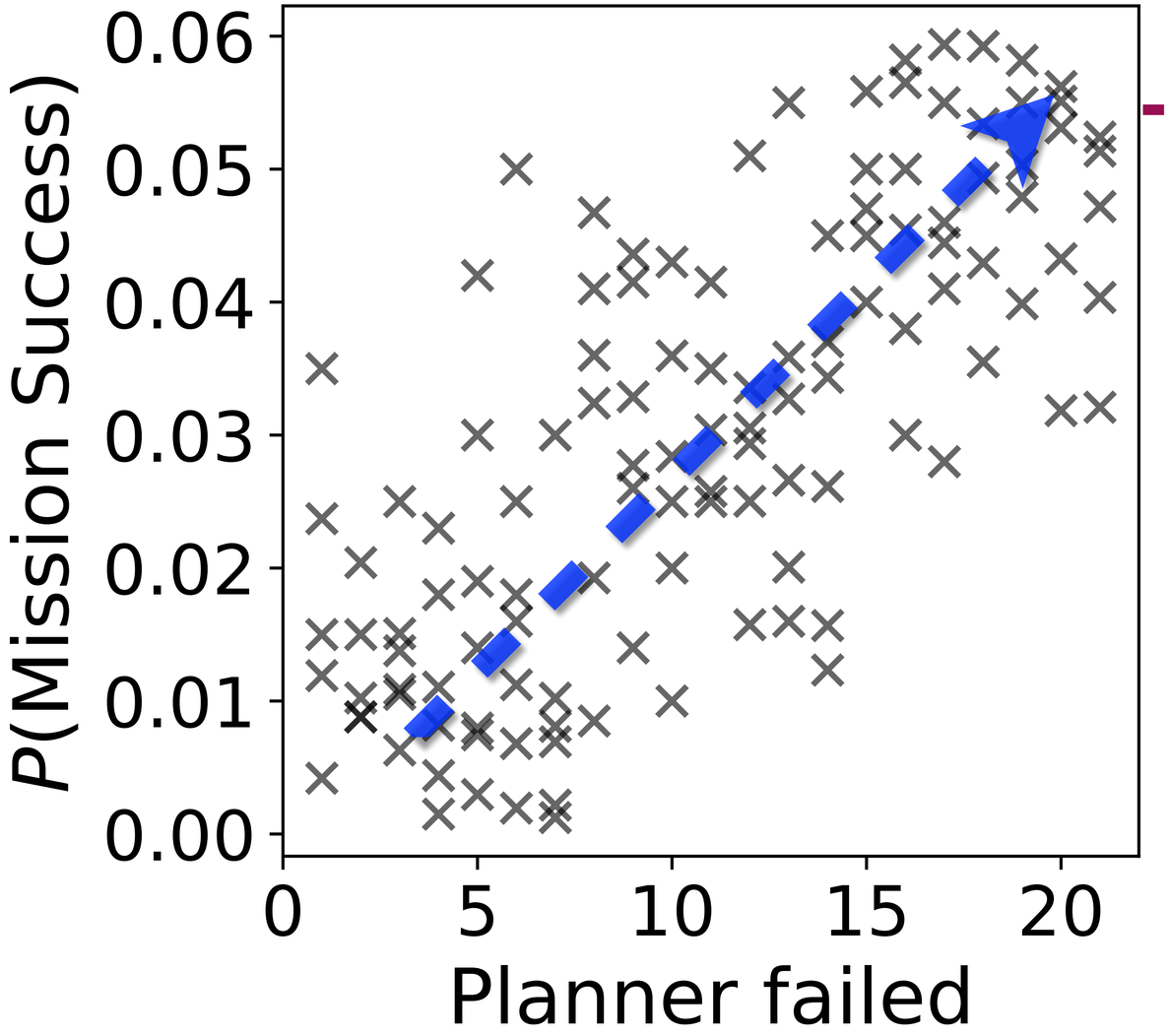
Regression Equation

These methods rely on statistical correlations to extract meaningful information required for performance tasks.

Challenges

- **Trial-and-error** requires non-trivial human efforts due to the large configuration space.
- **Even after finding the optimal fix**, the new fix is not guaranteed to function in different environments.
- **Performance influence models** suffer from several shortcomings:
  - Non-transferable across different robotic platforms and environments
  - Could produce Incorrect explanations
  - Could produce Unreliable predictions
  - Data collection is expensive from physical hardware

Incorrect Reasoning About The Robot's Behavior



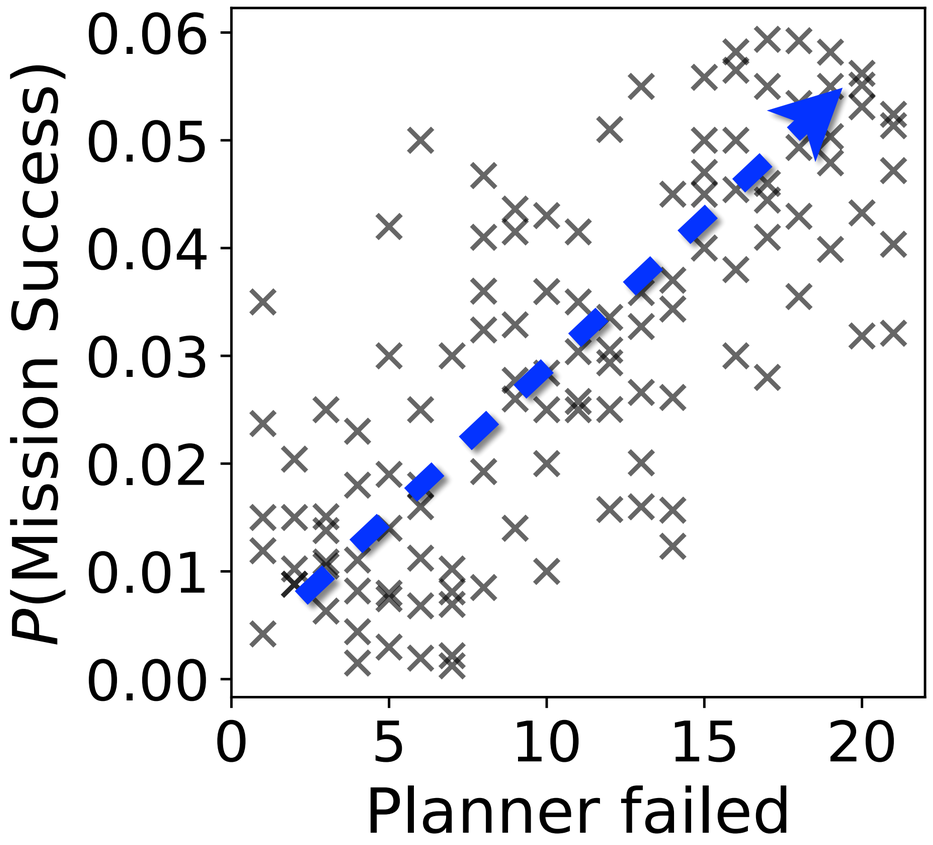
Increasing Planner failed increases $P(\text{Mission success})$

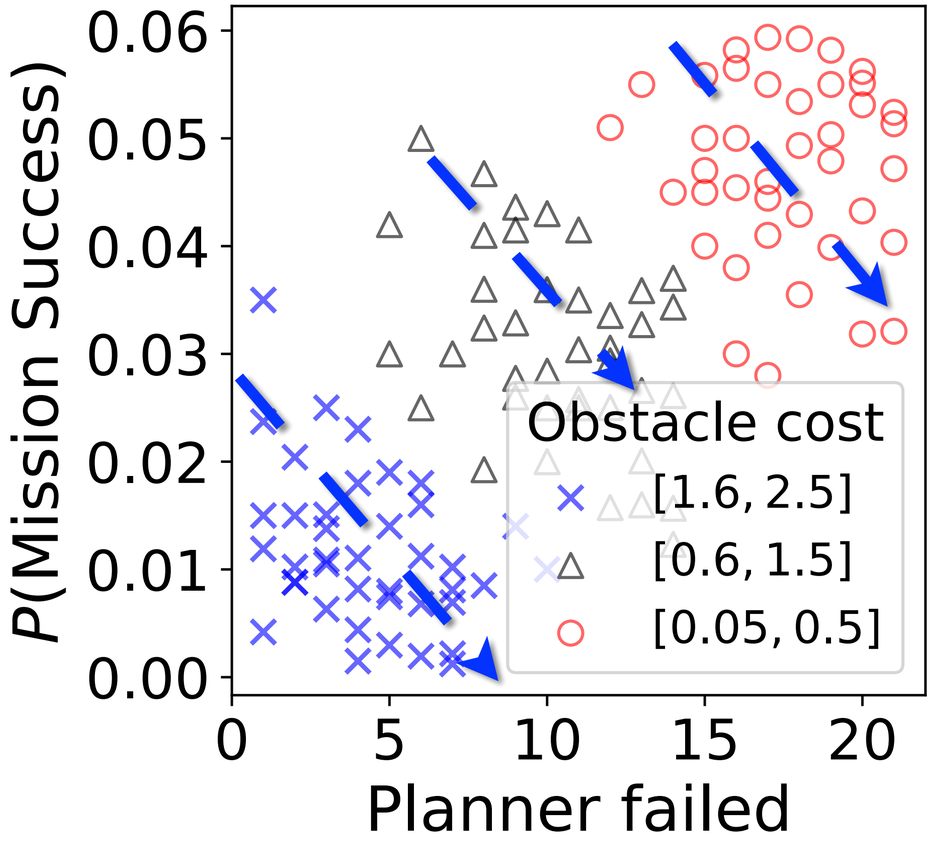
This is counter-intuitive

More Planner failed should reduce $P(\text{Mission success})$ not increase it

Purely statistical models built on this data will be unreliable.

Performance Influence Models might be Unreliable

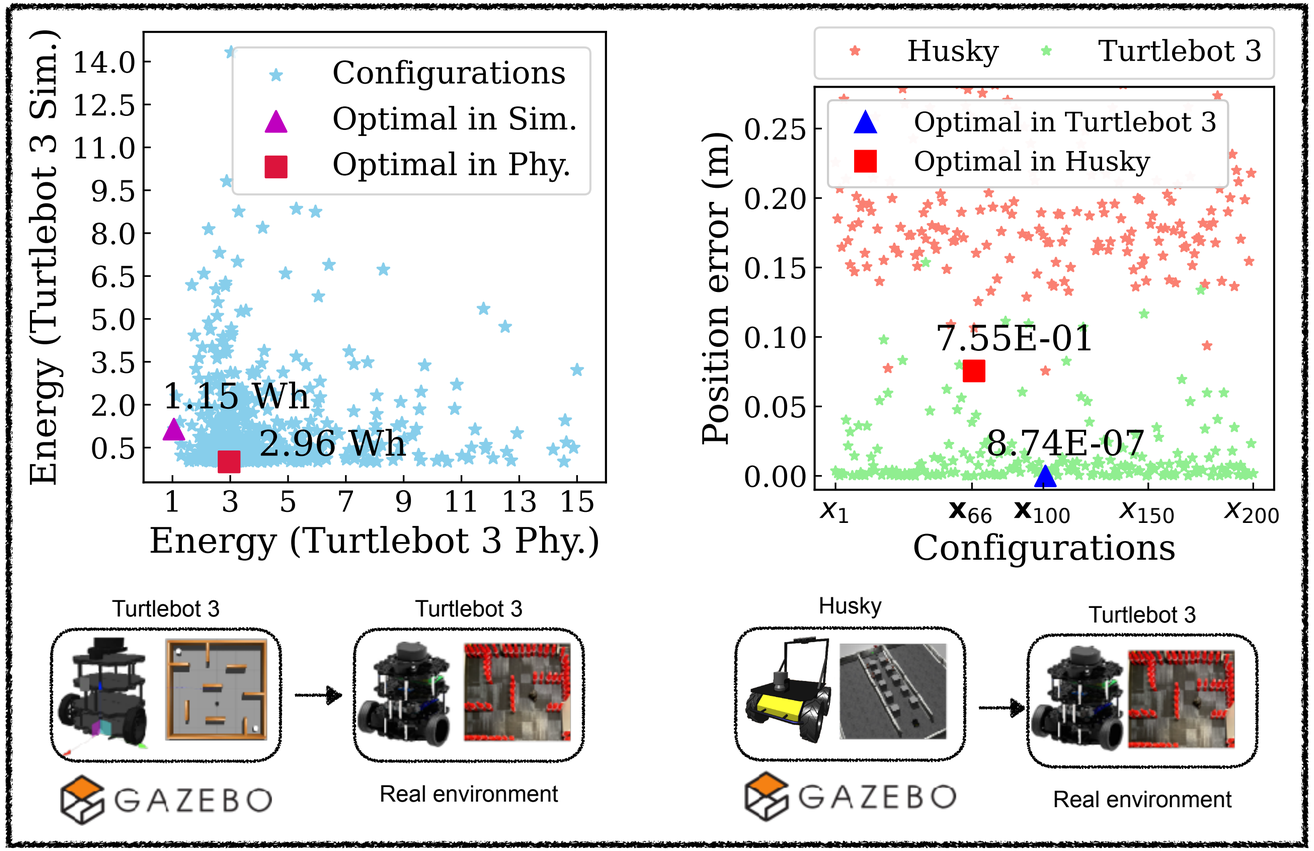




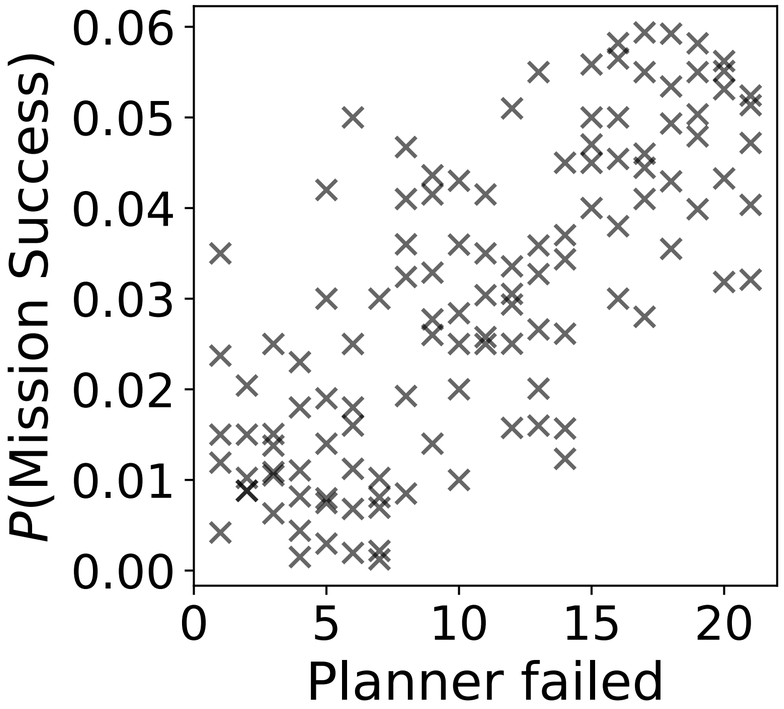
Segregating data on Obstacle Cost indicates that within each group increase of Planer Failed result in a decrease in P(Mission success).

Optimal configuration in one deployment environment does not remain optimal in another.

- Optimal configuration for *Turtlebot 3* in simulation differs from its physical counterpart (energy consumption is increased by 2.57x).
- Optimal configuration for *Turtlebot 3* is not suitable in *Husky* (significant increase in position error by 8.64×10^5 times).
- Machine learning models employed in the transfer learning approaches are vulnerable to spurious correlations.

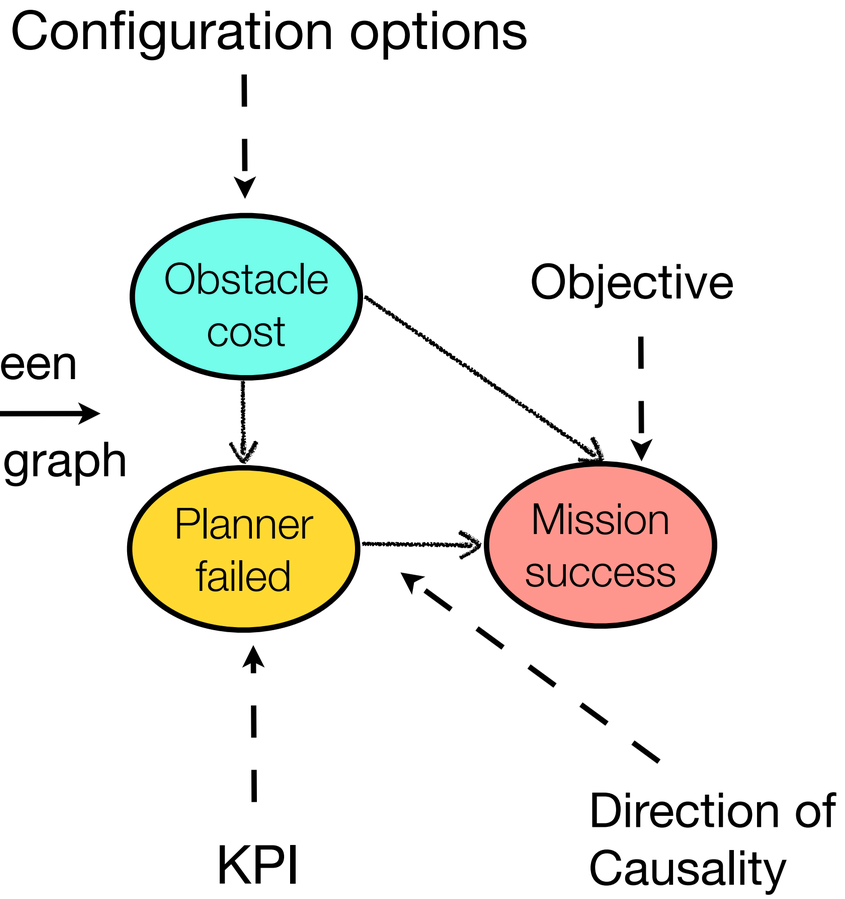


Our Key Idea: Building Causal Performance Model instead of Performance Influence Models



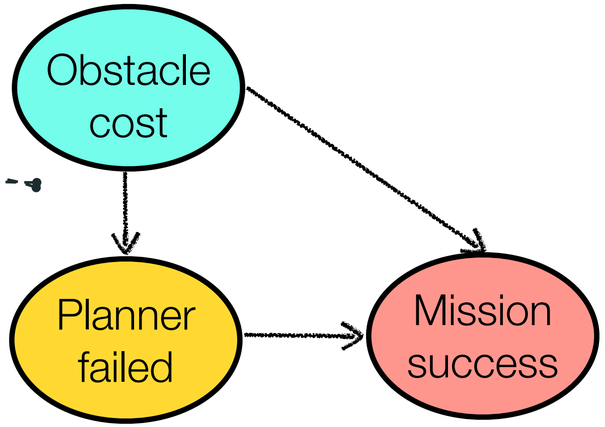
Expresses the relationships between

interacting variables as a causal graph



How to use Causal Models?

How to generate a causal model?



Inference

Given the causal graph $C_{\mathcal{G}}$, determine the configuration option in $\mathcal{X}$ which is the root cause for the observed functional fault characterized by performance objectives $\mathcal{Y}$ as follows:

$$\{v_i^*\} = \underset{v_i}{\text{argmax}} ACE(v_i, v_j^*)$$

where $\{v_i^*\} \subset \mathcal{X}$ is the set of root causes, $\{v_j^*\} \subset \mathcal{Y}$ are the performance objectives characterizing the functional fault, and *ACE* represents the average causal effect.

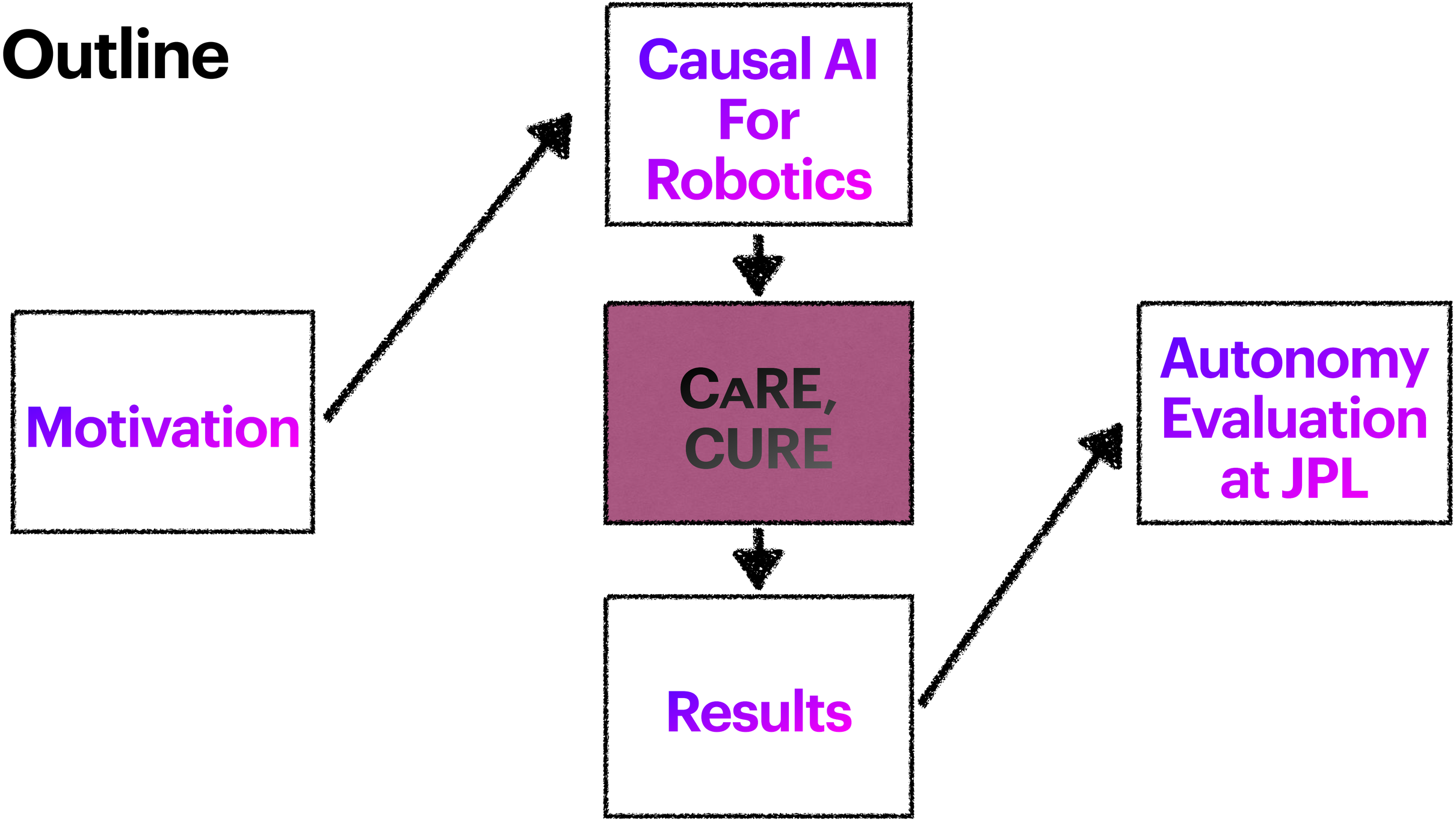
Learning

Given the observational data D, recover the causal graph $C_{\mathcal{G}}$ that encodes the dependency structure between $\mathcal{X}$, $\mathcal{S}$ and $\mathcal{Y}$ of $\mathcal{V}$ such that the following structural constraints are satisfied.

$$v_i \nvdash v_j \; \forall v_i \in \mathcal{X} \subset \mathcal{V}, \forall v_j \in \mathcal{Y} \subset \{\mathcal{V} \backslash \mathcal{X} \backslash \mathcal{S}\}$$

How to use the causal model for diagnosing misconfigurations?

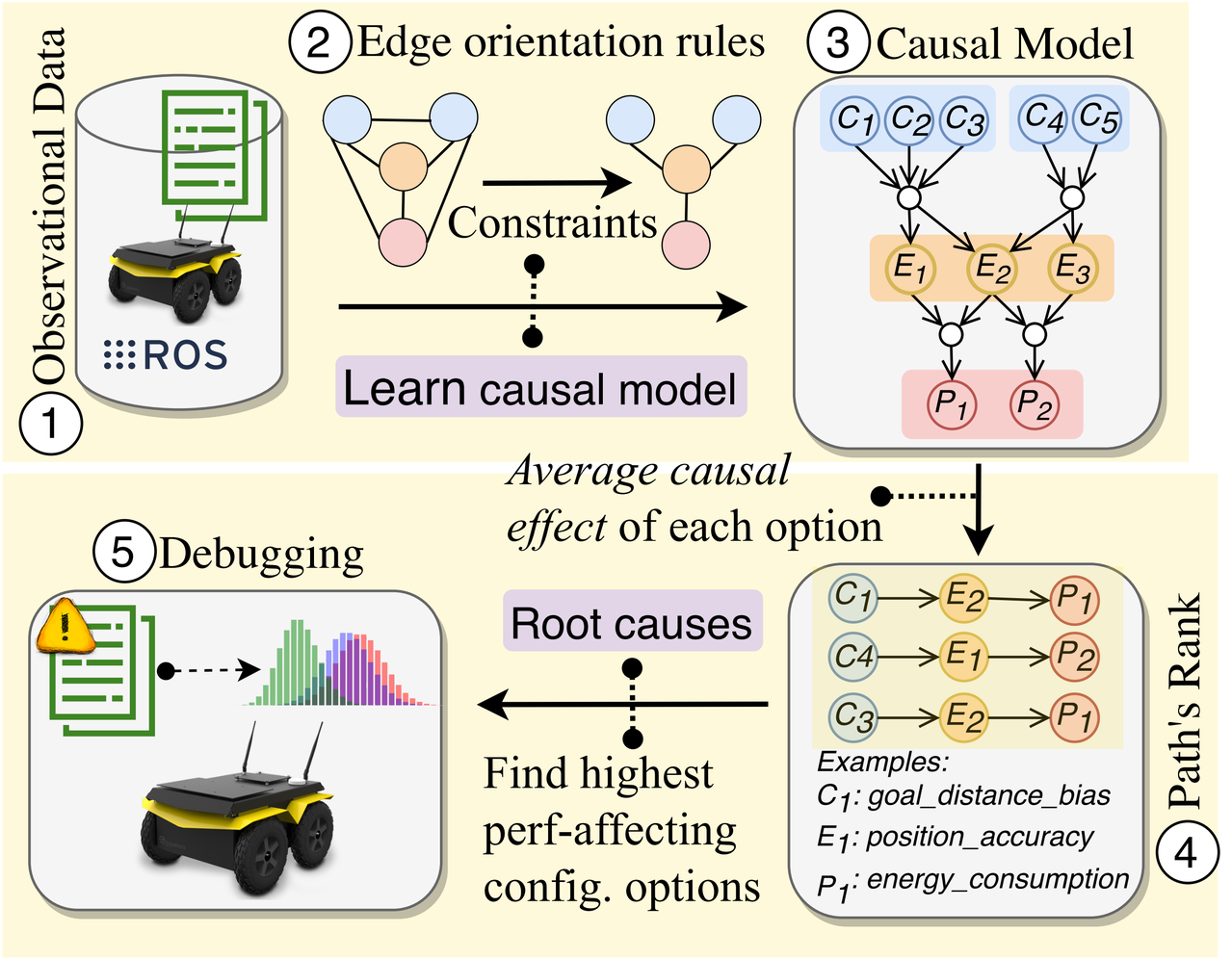
Outline


Causal AI For Robotics
Motivation
CaRE, CURE
Autonomy Evaluation at JPL
Results

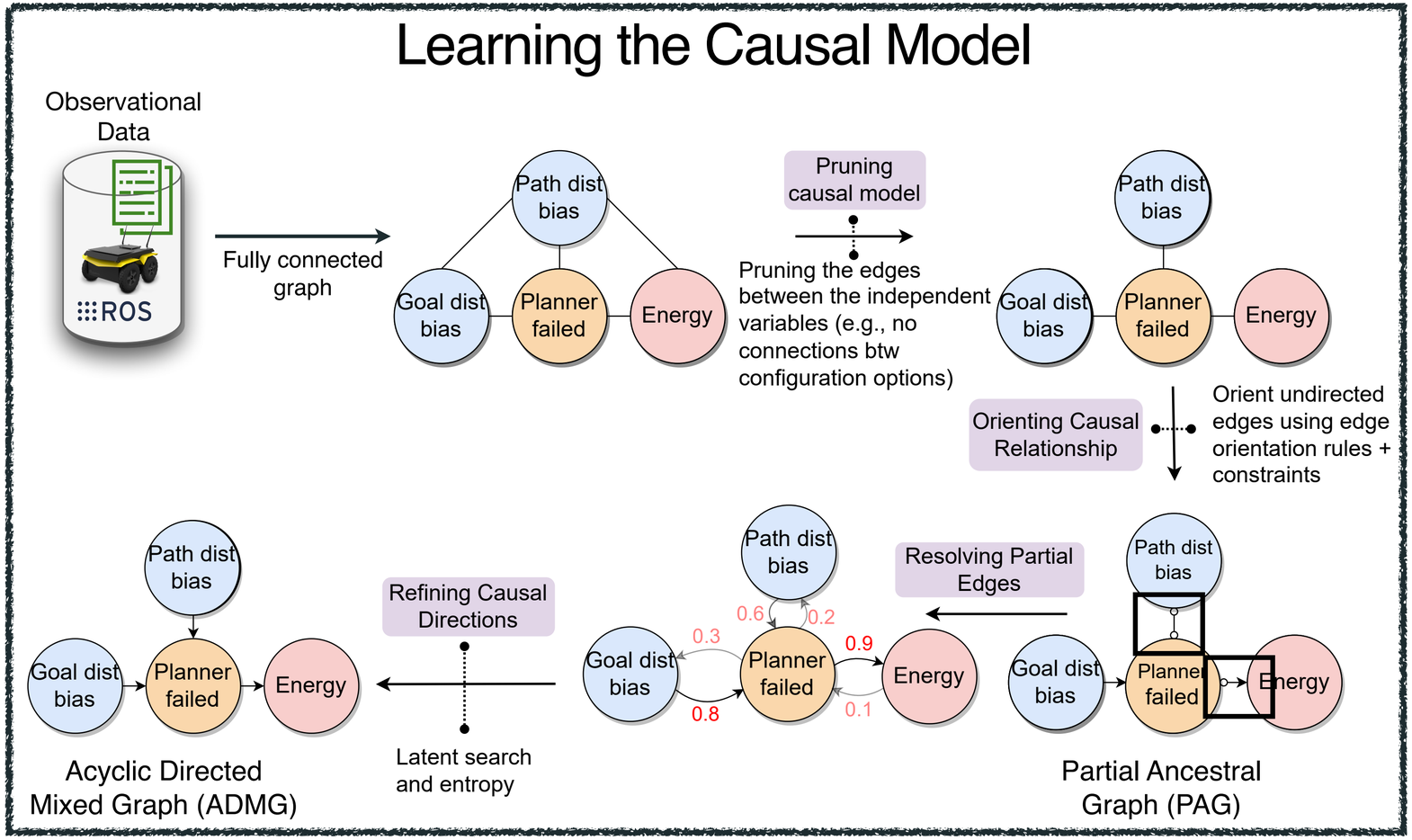
CaRE: End-to-end Pipeline

Stages

1. Learn causal model from observational data
2. Identify the causal paths
3. Average causal effect estimation



Learning and Estimating Causal Effects



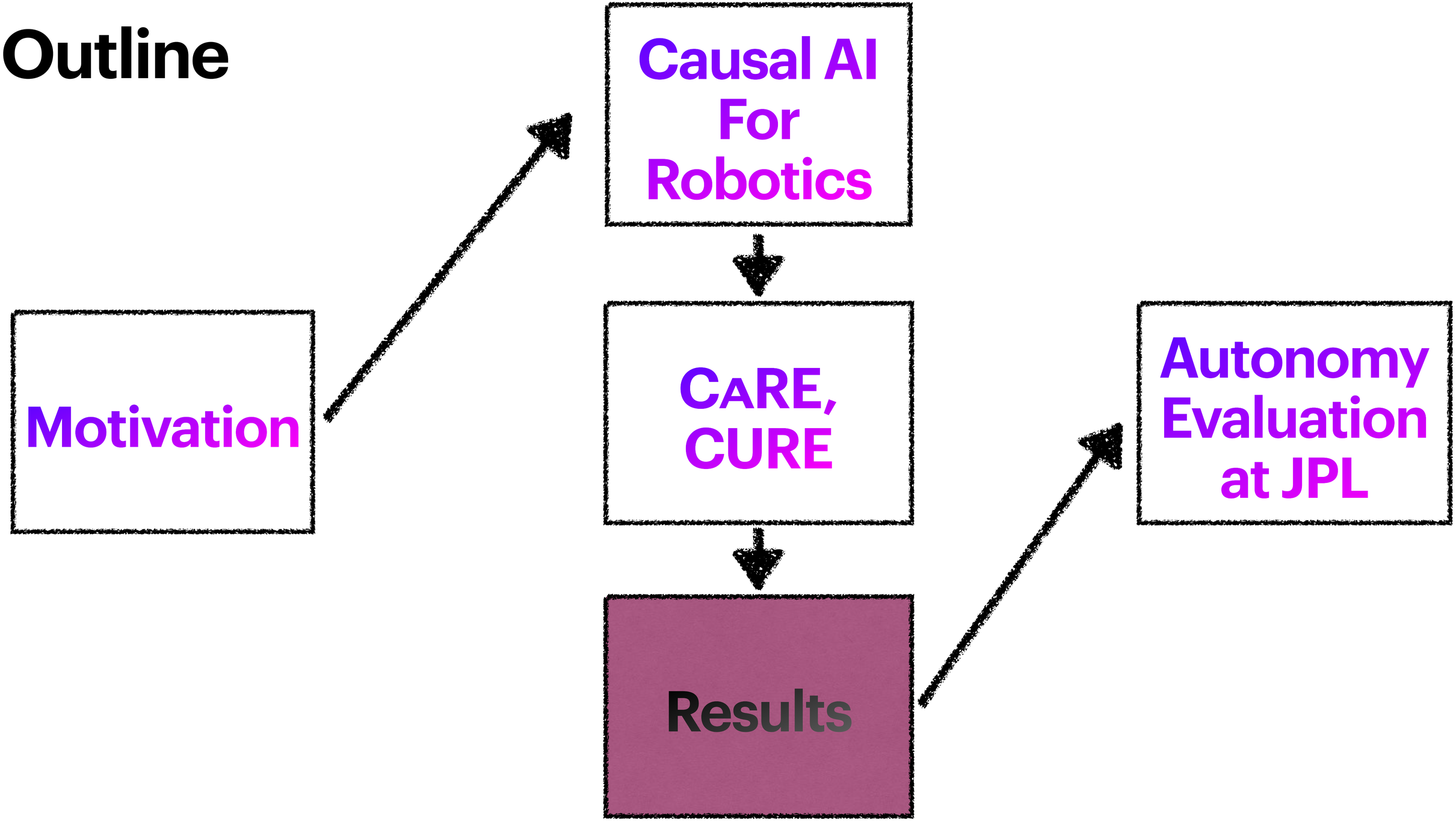
Causal Effect Estimation

- Use do-calculus to measure average causal effect

Estimate the probability of satisfying *energy* given *path_dist_bias*=20.0

$$P(Energy < 25Wh \mid do(Path\_dist\_bias = 20.0))$$

Outline


Causal AI For Robotics
Motivation
CaRE, CURE
Autonomy Evaluation at JPL
Results

Research Questions

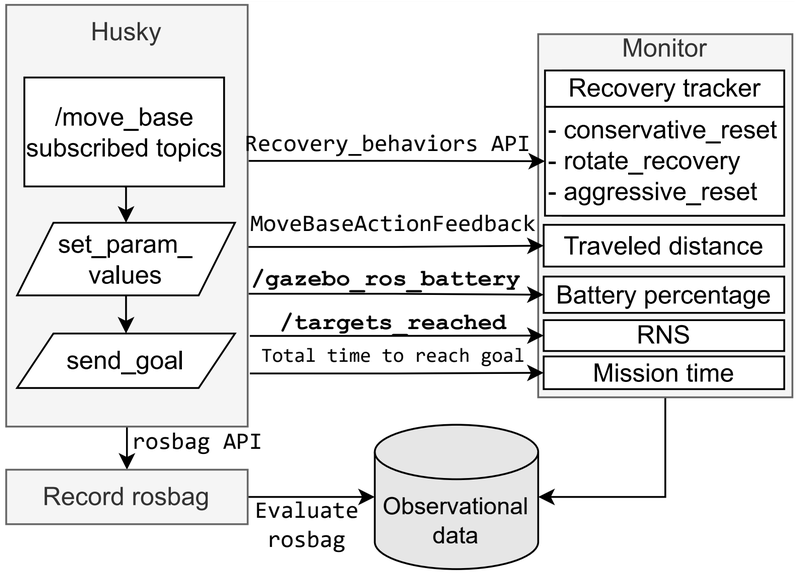
- **Research Question 1 (Accuracy).** To what extent are the root causes determined by CaRE the true root causes of the observed functional faults?

- **Research Question 2 (Transferability).** To what extent can CaRE accurately detect misconfigurations when deployed in a different platform?

Results

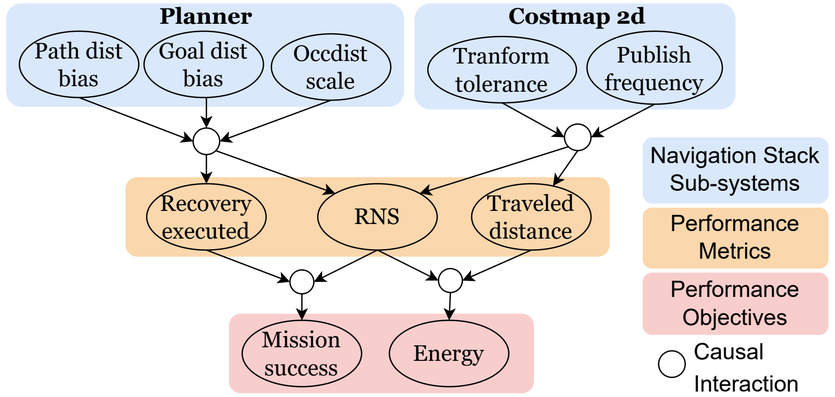
Experimental Setup

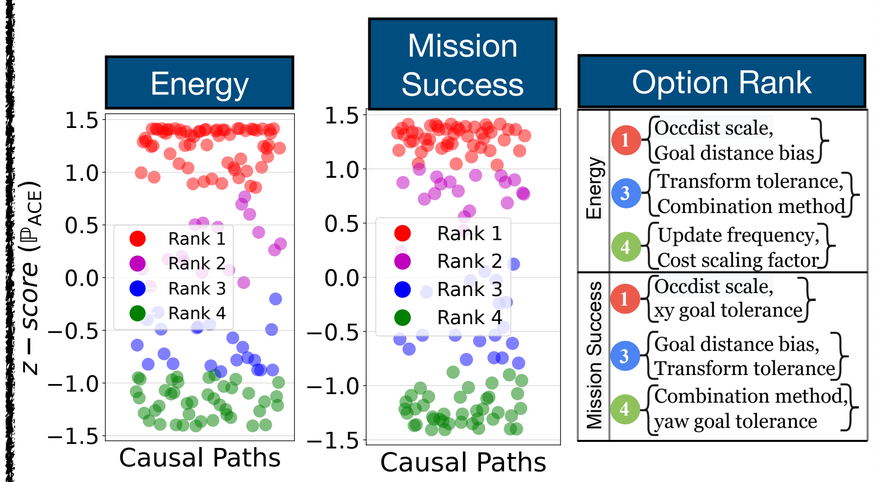
Observational Data Collection



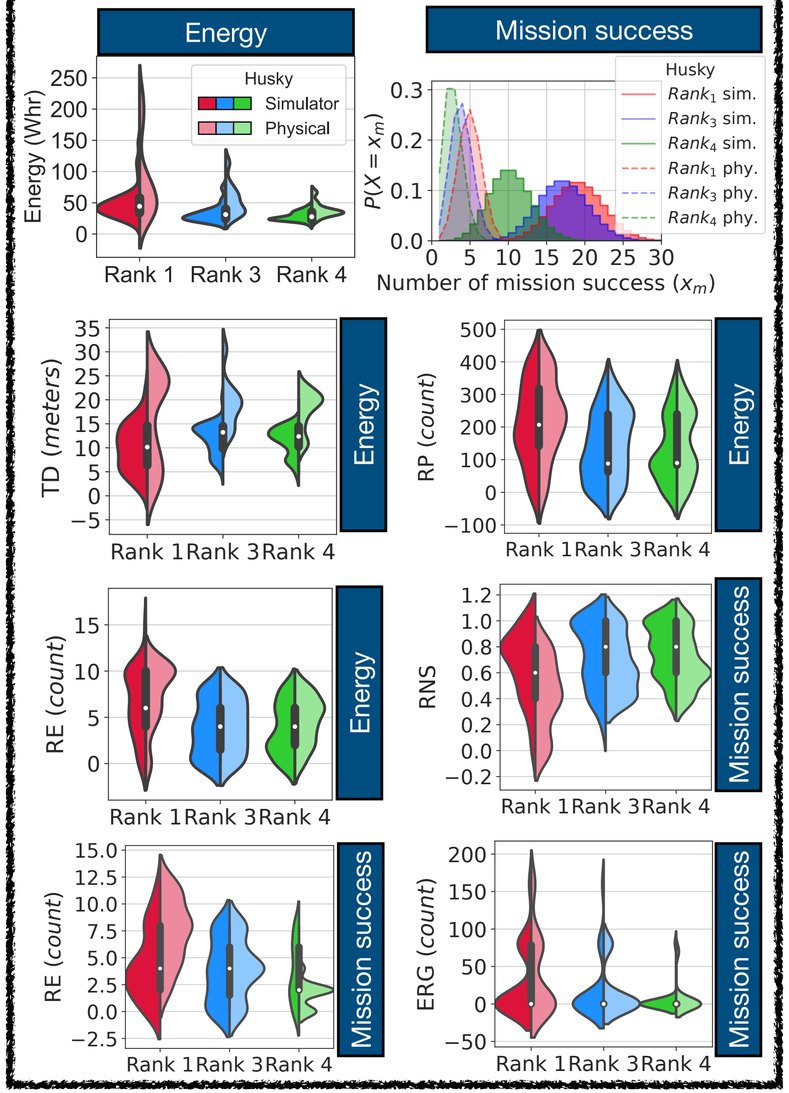
Applying CaRE

A partial causal model discovered in our experiments using *Husky* in simulation





Evaluation



Takeaway Configuration options which rank higher have the strongest influence on the performance objectives.

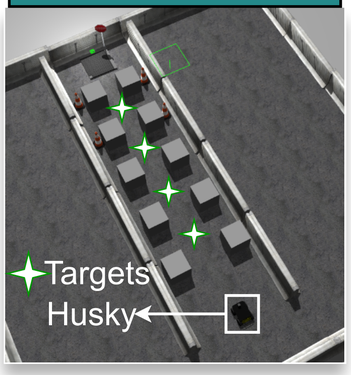
Results

Experimental Setup

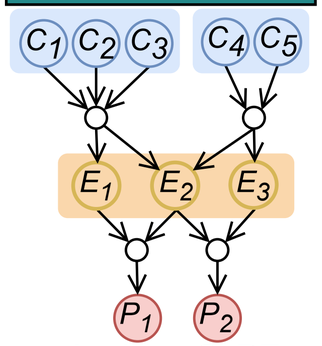
Learning the Causal Model from Source Platform and Environment

Husky

Gazebo Environment



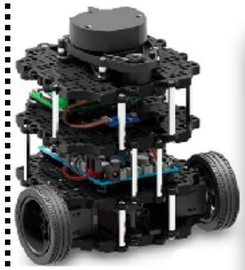
Learned Causal Model



Applying CaRE

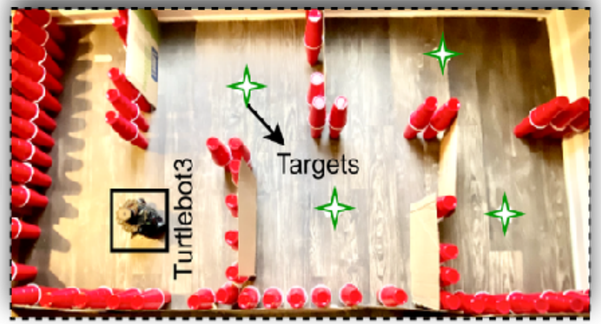
Reusing the learned causal model from source

Target Platform and Environment

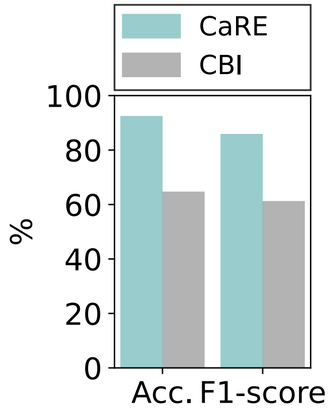


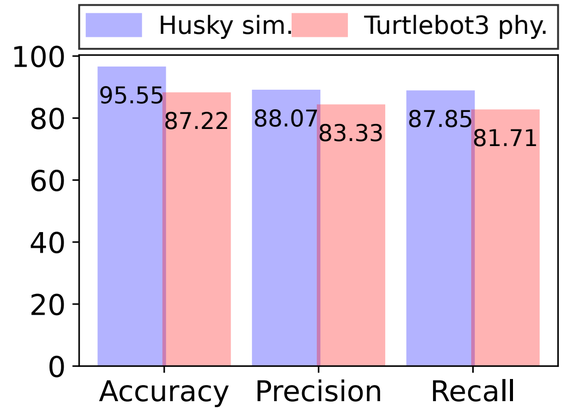
Turtlebot 3

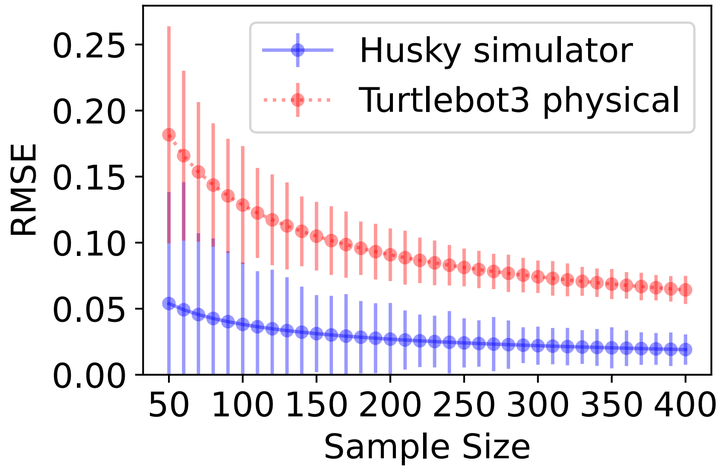
Real Environment



Evaluation







Takeaway

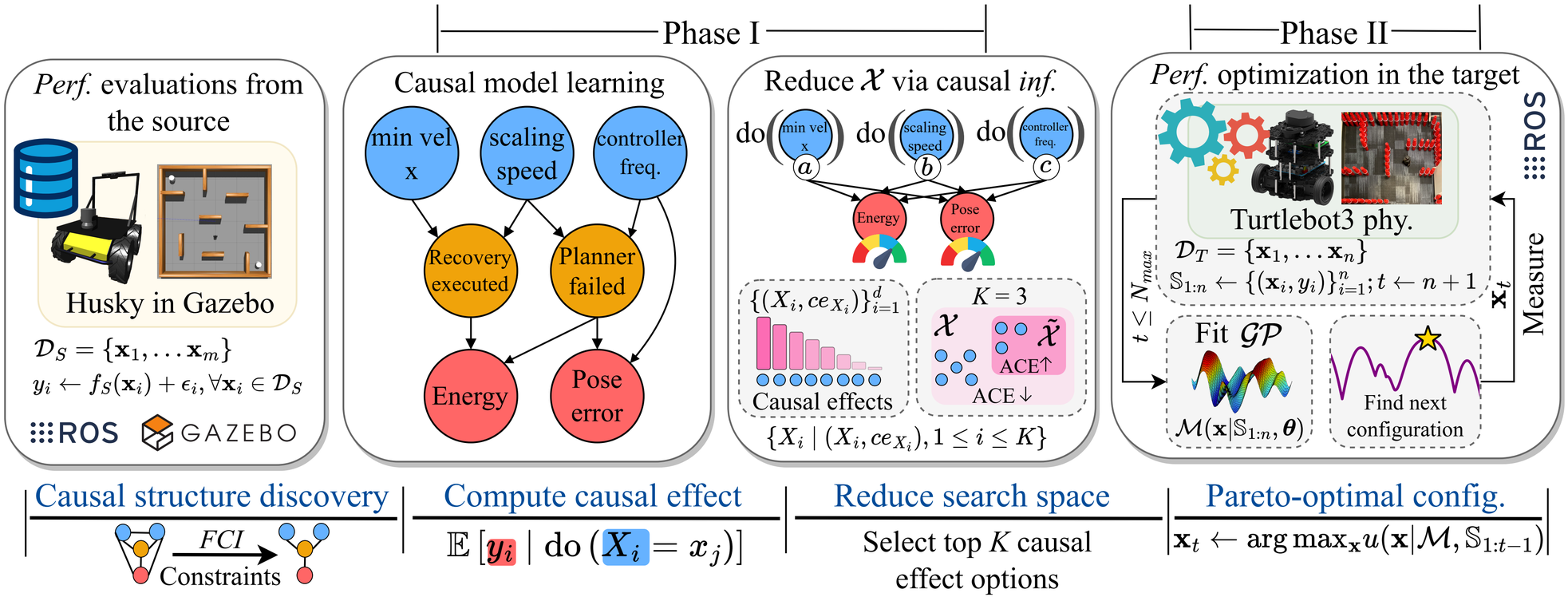
CaRE transfer reasonably well when reusing the causal model learned from source platform and environment and achieves higher accuracy than the baseline.

CURE: Simulation Augmented Auto-tuning in Robotics

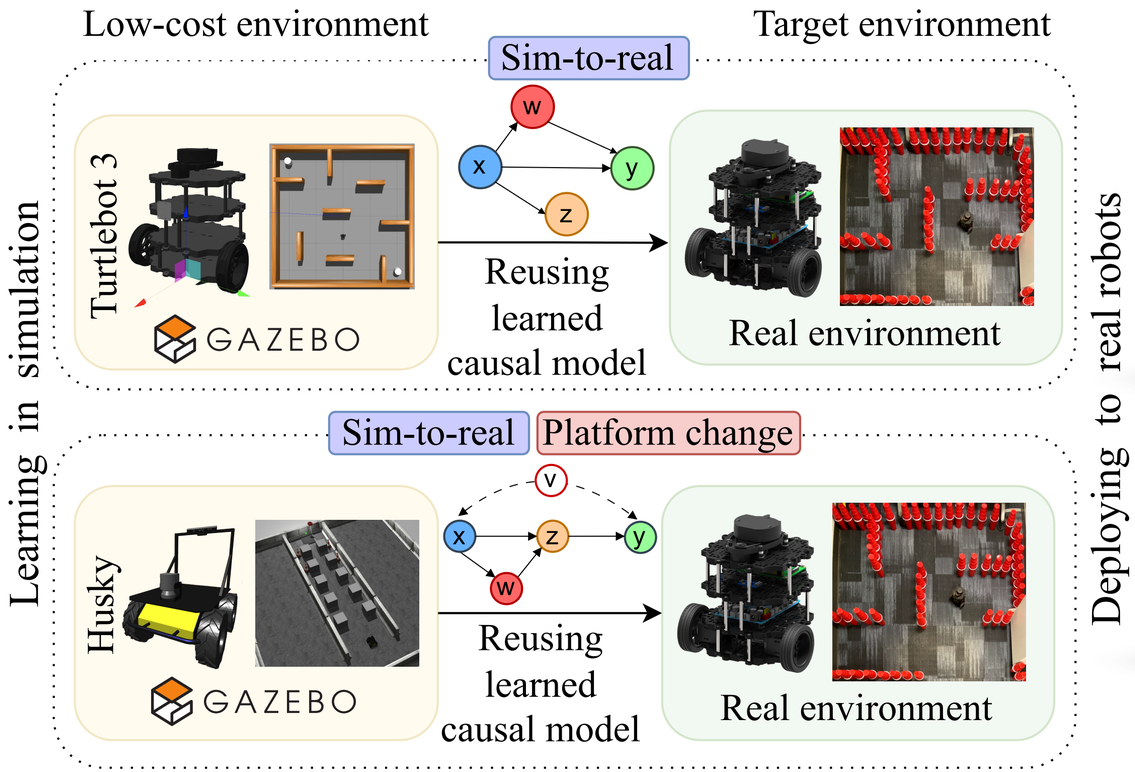
Goal

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}), s.t. : h(\boldsymbol{x}) \geq 0 \quad (1)$$

where $\boldsymbol{x}^* \in \mathcal{X}$ is one of the Pareto-optimal configurations and adhere to the safety constraints.



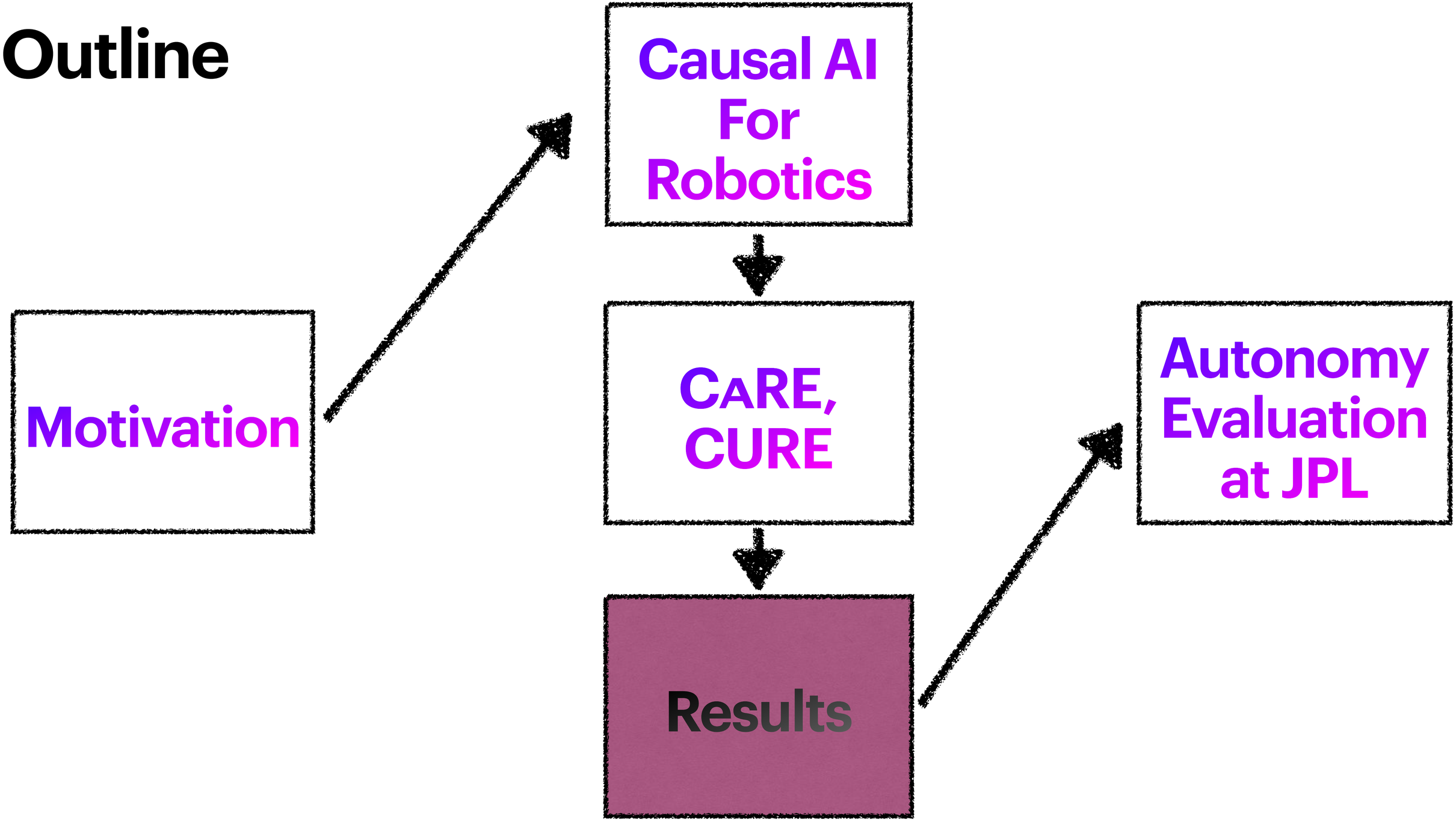
Experimental Setup



RQ1 (Effectiveness): How effective is CURE in (i) ensuring optimal performance; (ii) utilizing the budget; and (iii) respecting the safety constraints compared to the baselines?

RQ2 (Transferability): How does the effectiveness of CURE change when the severity of deployment changes varies (e.g., environment and platform change)?

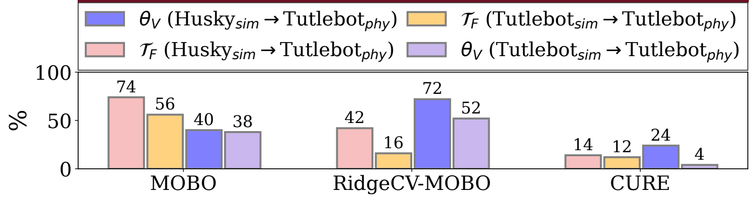
Outline


Causal AI For Robotics
Motivation
CaRE, CURE
Autonomy Evaluation at JPL
Results

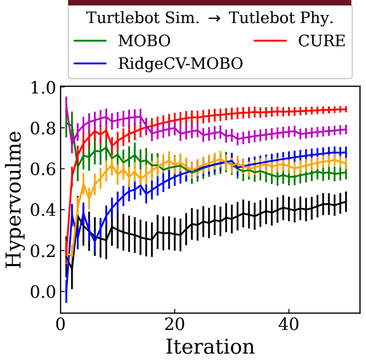
Results

Transferability and Effectiveness

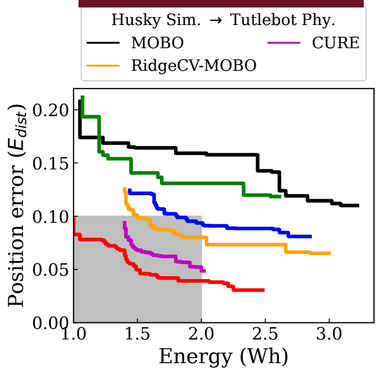
Constraint violation (θ_V) and task failure ($\mathcal{T}_F$) for different severity change.



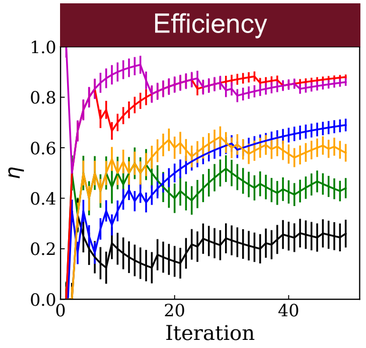
Hypervolume



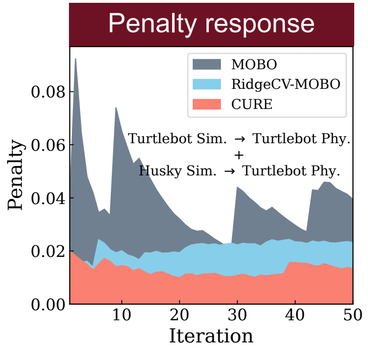
Pareto fronts



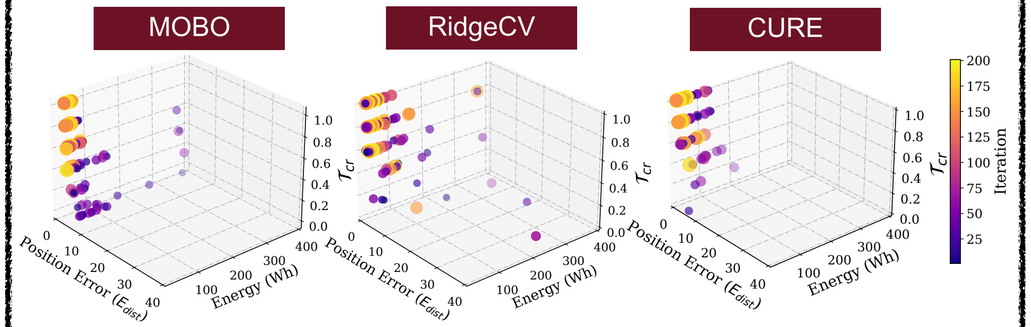
Efficiency



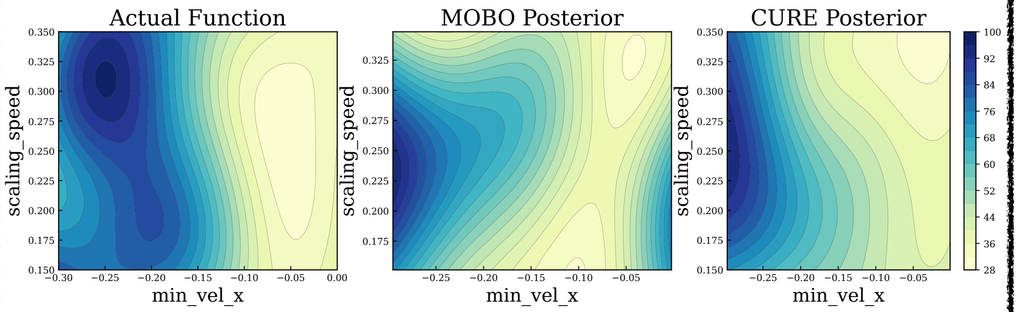
Penalty response



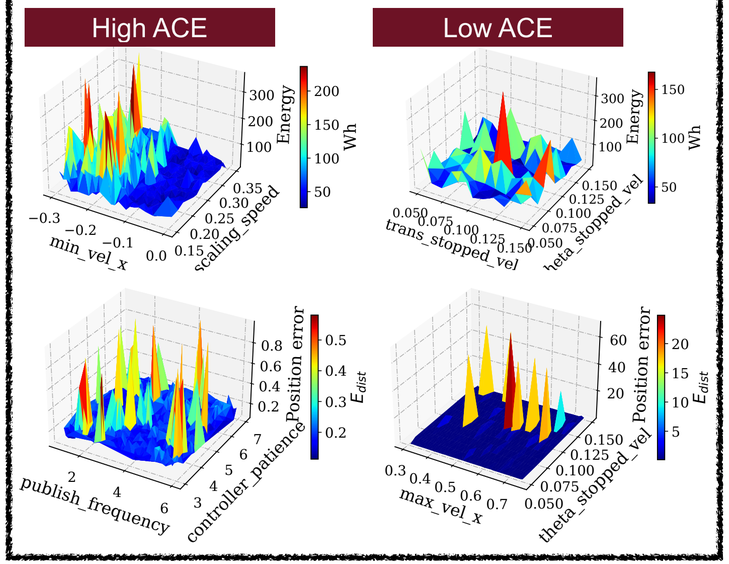
CURE demonstrates a denser surface response near the Pareto front and achieved higher task success rate ($\mathcal{T}_{cr}$) in fewer iterations.



CURE demonstrates a better understanding about the performance behavior compared to multi-objective Bayesian optimization (MOBO).



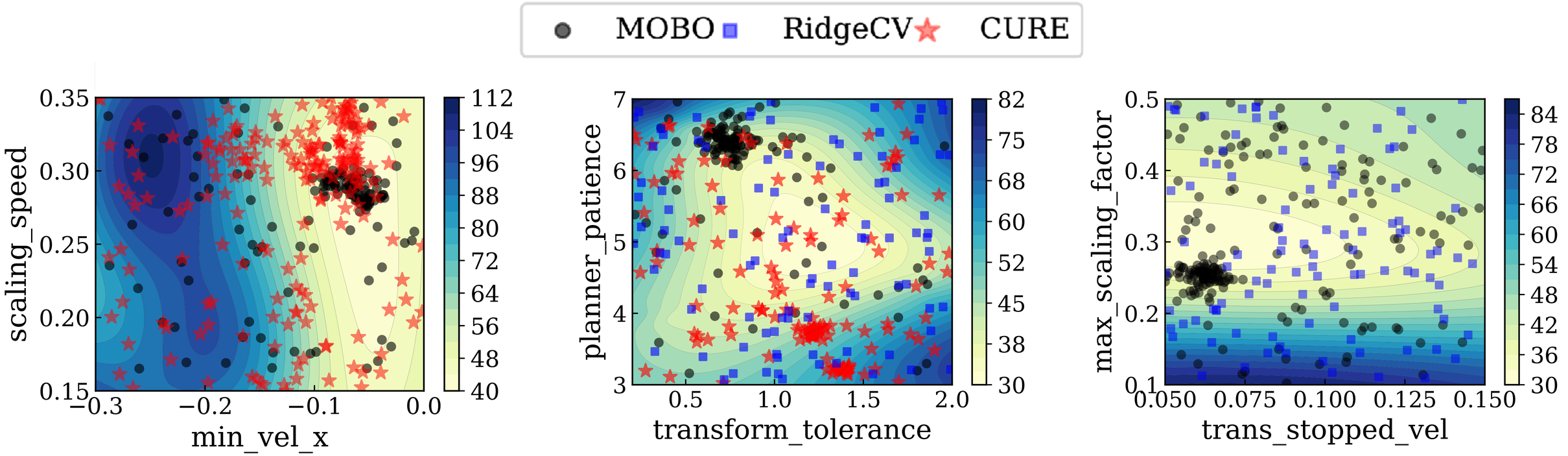
Pairwise interaction between configuration options.



Takeaway

- CURE finds a configuration that improves performance by 2x.
- CURE achieves this improvement with gains in efficiency of 4.6x when we transfer the knowledge learned from *Husky* in simulation to the *Turtlebot 3* physical robot.
- The learned causal model is transferable across similar but different settings, encompassing different environments, mission/tasks, and new robotic systems.

Why causal reasoning based optimization works better?



High ACE

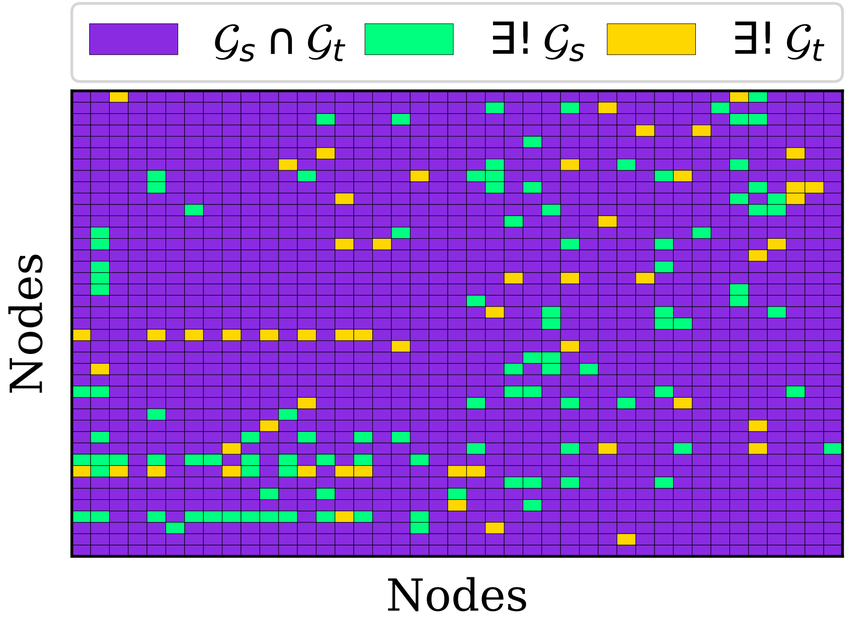
Median ACE

Low ACE

CURE's efficient budget utilization is attributed to a comprehensive evaluation of the core configuration options.

Why causal reasoning based optimization works better?

- Significant overlap between causal structures (common edges are represented as blue squares) developed in *Husky* ($\mathcal{G}_s$) and *Turtlebot 3* ($\mathcal{G}_t$).



Unique edges are represented as green and yellow squares in $\mathcal{G}_s$ and $\mathcal{G}_t$, respectively.

CURE leverages the knowledge derived from the causal model learned on the source platform.

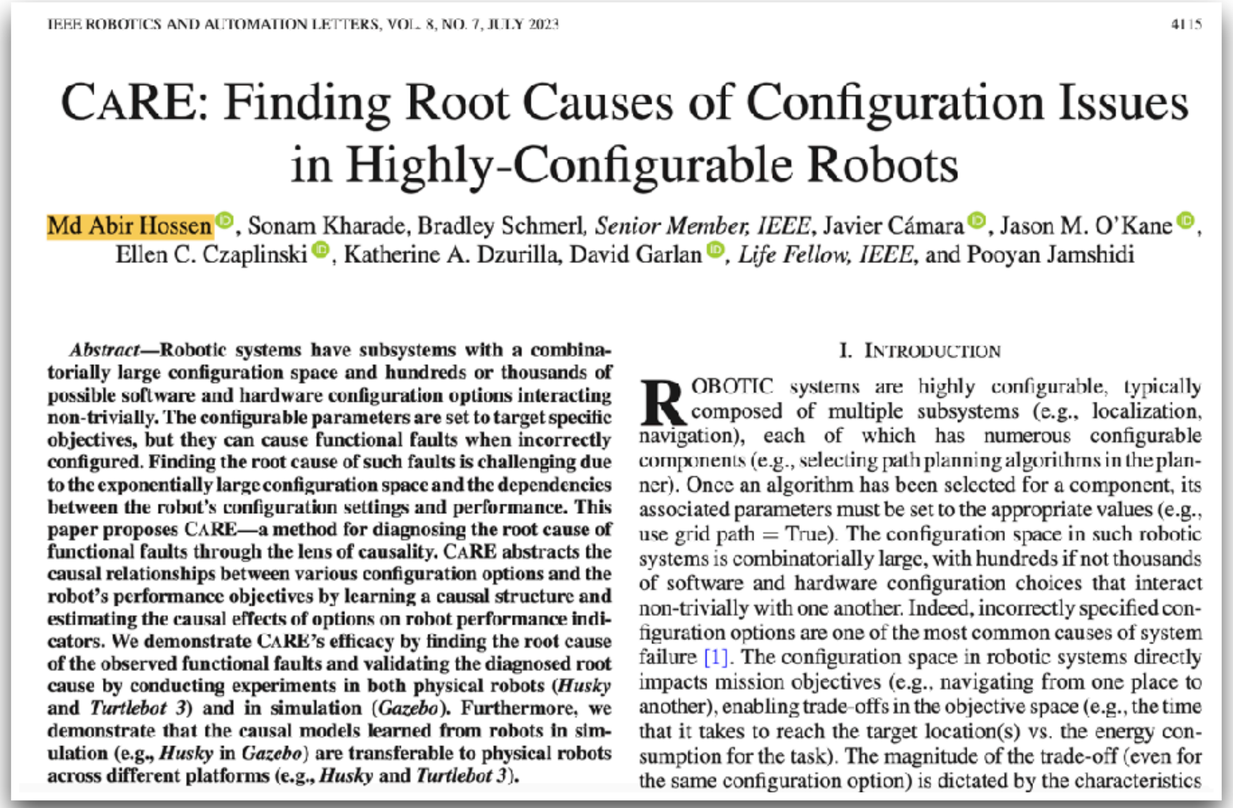
Limitations

- **Causal model error:** The NP-hard complexity of causal discovery introduces a challenge, implying that the identified causal model may not always represent the ground-truth causal relationships among variables.
    - Discrepancies between the causal structure discovered and the actual structures.

- **Potential biases when transferring the causal model:** Caution must be exercised when reusing the entire causal graph learned from the source platform.

Code and Data

IEEE Robotics and Automation Letters, 2023 —> Presented in IROS 2023



IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 8, NO. 7, JULY 2023 4115

CARE: Finding Root Causes of Configuration Issues in Highly-Configurable Robots

Md Abir Hossen, Sonam Kharade, Bradley Schmerl, *Senior Member, IEEE*, Javier Cámara, Jason M. O'Kane, Ellen C. Czaplinski, Katherine A. Dzurilla, David Garlan, *Life Fellow, IEEE*, and Pooyan Jamshidi

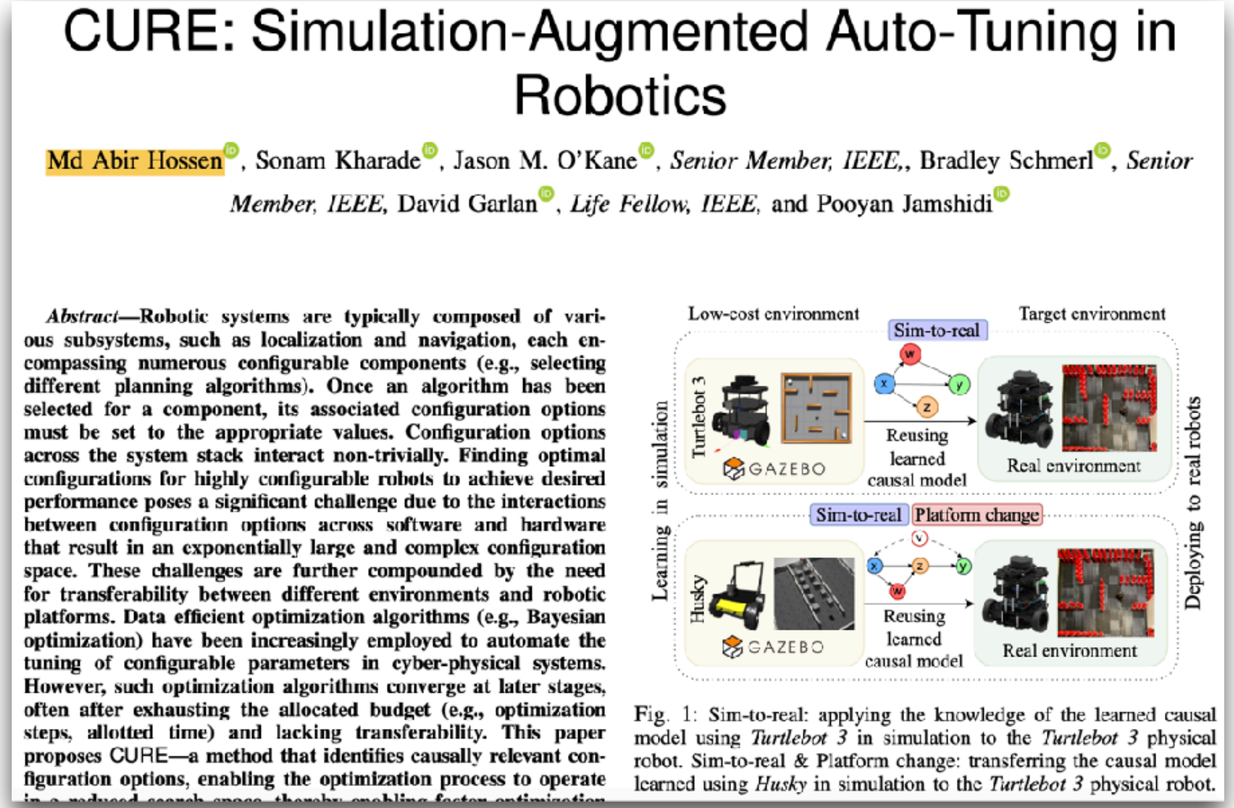
***Abstract*—Robotic systems have subsystems with a combinatorially large configuration space and hundreds or thousands of possible software and hardware configuration options interacting non-trivially. The configurable parameters are set to target specific objectives, but they can cause functional faults when incorrectly configured. Finding the root cause of such faults is challenging due to the exponentially large configuration space and the dependencies between the robot's configuration settings and performance. This paper proposes CARE—a method for diagnosing the root cause of functional faults through the lens of causality. CARE abstracts the causal relationships between various configuration options and the robot's performance objectives by learning a causal structure and estimating the causal effects of options on robot performance indicators. We demonstrate CARE's efficacy by finding the root cause of the observed functional faults and validating the diagnosed root cause by conducting experiments in both physical robots (*Husky* and *Turtlebot 3*) and in simulation (*Gazebo*). Furthermore, we demonstrate that the causal models learned from robots in simulation (e.g., *Husky* in *Gazebo*) are transferable to physical robots across different platforms (e.g., *Husky* and *Turtlebot 3*).**

I. Introduction

ROBOTIC systems are highly configurable, typically composed of multiple subsystems (e.g., localization, navigation), each of which has numerous configurable components (e.g., selecting path planning algorithms in the planner). Once an algorithm has been selected for a component, its associated parameters must be set to the appropriate values (e.g., use grid path = True). The configuration space in such robotic systems is combinatorially large, with hundreds if not thousands of software and hardware configuration choices that interact non-trivially with one another. Indeed, incorrectly specified configuration options are one of the most common causes of system failure [1]. The configuration space in robotic systems directly impacts mission objectives (e.g., navigating from one place to another), enabling trade-offs in the objective space (e.g., the time that it takes to reach the target location(s) vs. the energy consumption for the task). The magnitude of the trade-off (even for the same configuration option) is dictated by the characteristics

https://github.com/softsys4ai/care

IEEE Transactions on Robotics, 2024 (under review)



CURE: Simulation-Augmented Auto-Tuning in Robotics

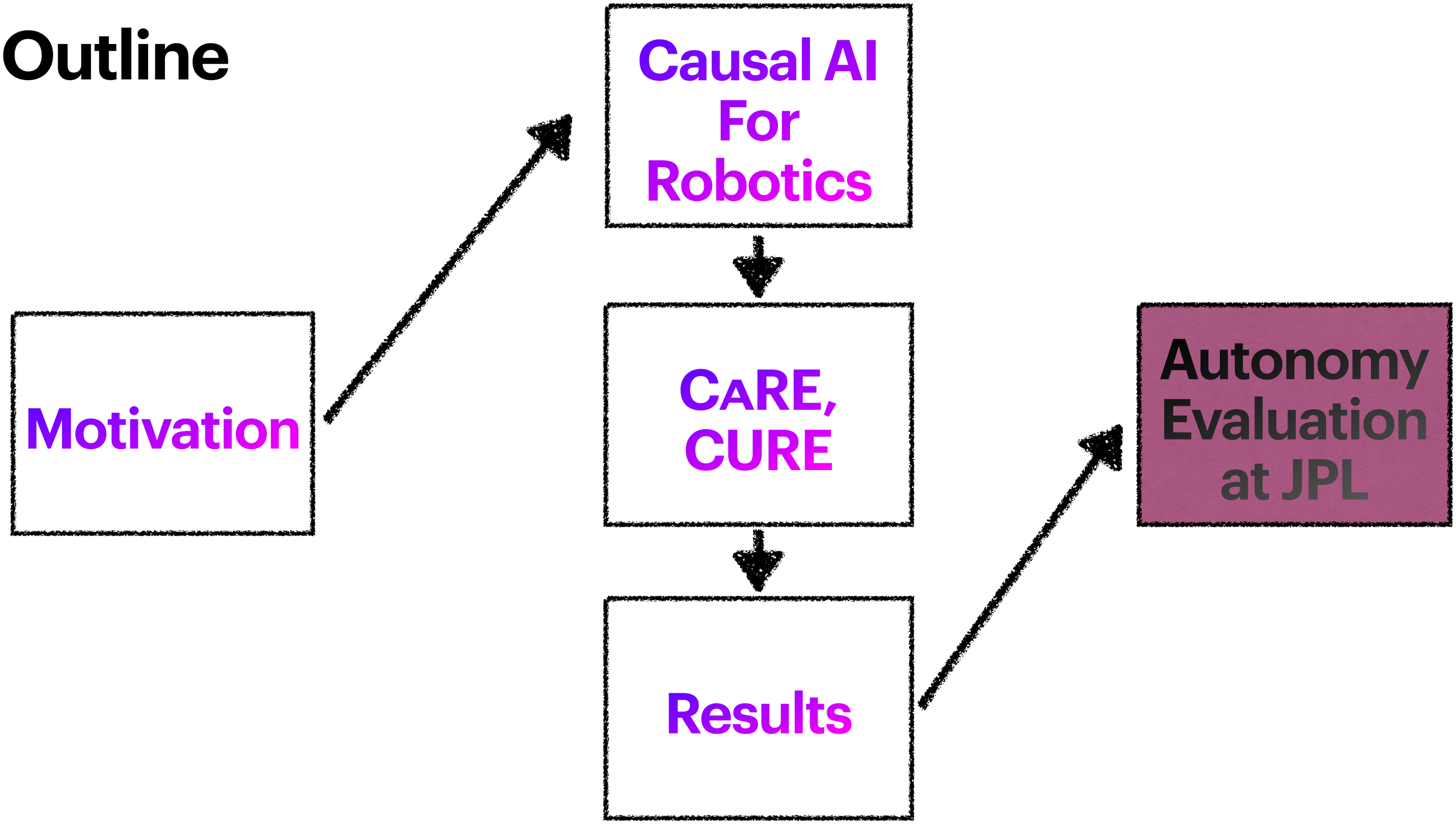
Md Abir Hossen, Sonam Kharade, Jason M. O'Kane, *Senior Member, IEEE*, Bradley Schmerl, *Senior Member, IEEE*, David Garlan, *Life Fellow, IEEE*, and Pooyan Jamshidi

***Abstract*—Robotic systems are typically composed of various subsystems, such as localization and navigation, each encompassing numerous configurable components (e.g., selecting different planning algorithms). Once an algorithm has been selected for a component, its associated configuration options must be set to the appropriate values. Configuration options across the system stack interact non-trivially. Finding optimal configurations for highly configurable robots to achieve desired performance poses a significant challenge due to the interactions between configuration options across software and hardware that result in an exponentially large and complex configuration space. These challenges are further compounded by the need for transferability between different environments and robotic platforms. Data efficient optimization algorithms (e.g., Bayesian optimization) have been increasingly employed to automate the tuning of configurable parameters in cyber-physical systems. However, such optimization algorithms converge at later stages, often after exhausting the allocated budget (e.g., optimization steps, allotted time) and lacking transferability. This paper proposes CURE—a method that identifies causally relevant configuration options, enabling the optimization process to operate**

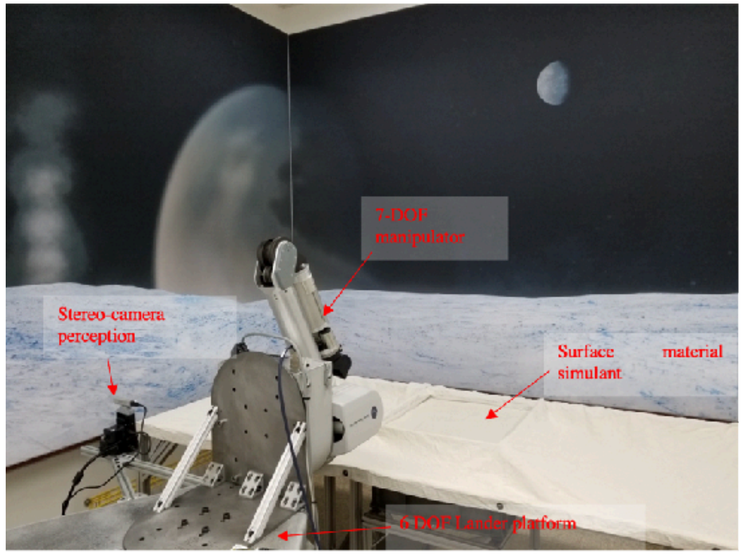
Fig. 1: Sim-to-real: applying the knowledge of the learned causal model using *Turtlebot 3* in simulation to the *Turtlebot 3* physical robot. Sim-to-real & Platform change: transferring the causal model learned using *Husky* in simulation to the *Turtlebot 3* physical robot.

https://github.com/softsys4ai/cure

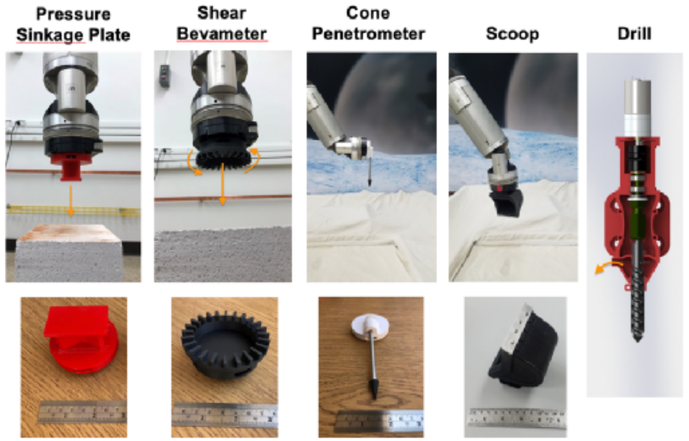
Outline


Causal AI For Robotics
Motivation
CaRE, CURE
Results
Autonomy Evaluation at JPL

Physical Space Lander Testbed at JPL

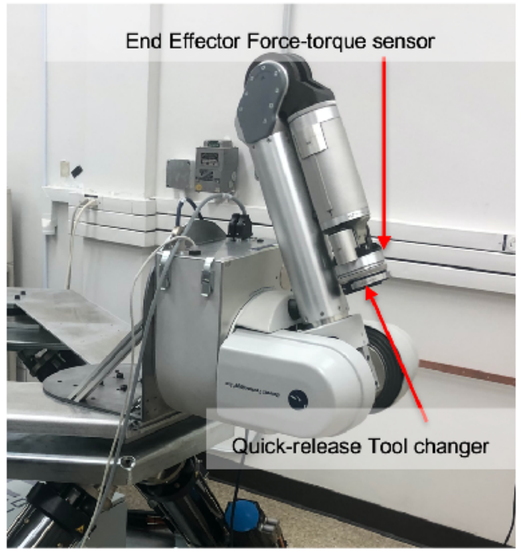


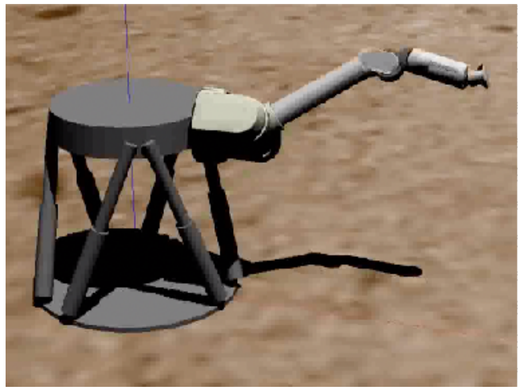
Testbed setup and major components



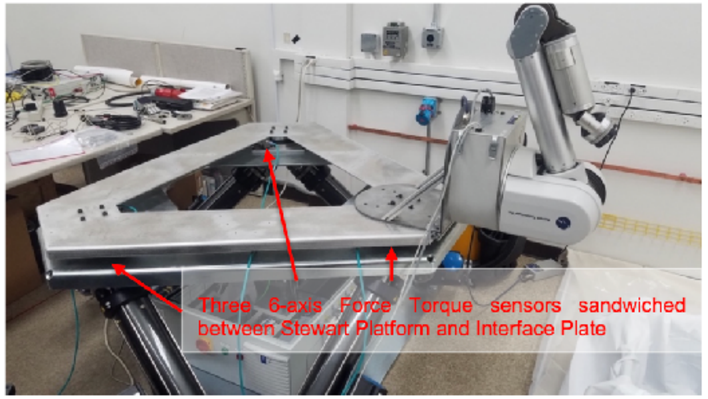
Modular instruments to be mounted on robot arm

Barrett WAM seven DOF manipulator arm mounted to lander with wrist FTS and tool changer





HITL simulator of lander and manipulator

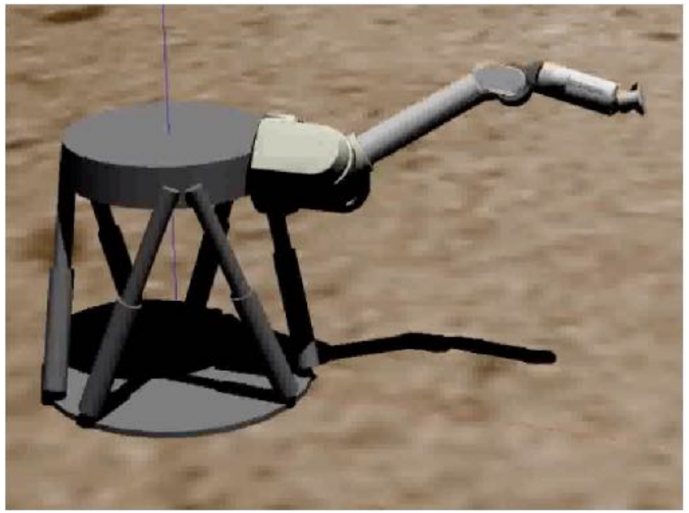


E2M Technologies six DOF Stewart Platform representing spacecraft lander

Physical Autonomy Testbed: https://www1.grc.nasa.gov/wp-content/uploads/2020_ASCE_OWLAT_20191028.pdf

Learning in Simulation for Transfer Learning to Physical Testbed

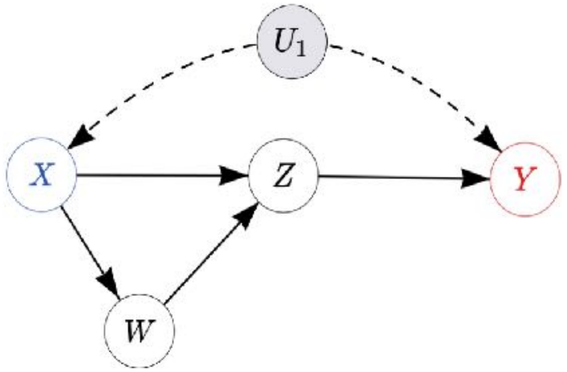
Simulation Environment



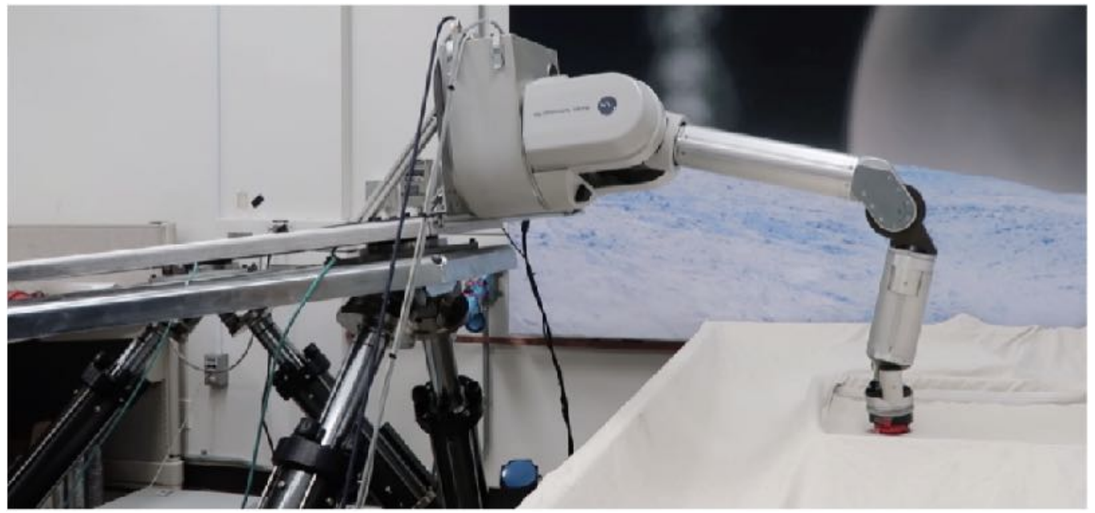
OWLAT-sim

Transfer

Causal Invariances



Physical Testbed



OWLAT

Demo

Extending CURE: Causal Multi-information Source Optimization

- Configuration: $\boldsymbol{x} = \langle X_1 = \mathbf{x}_1, X_2 = \mathbf{x}_2, \ldots, X_d = \mathbf{x}_d \rangle$
- Set of configurations: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$
- Target functions: $\mathbf{f}^{(t)} = \langle f_1^{(t)}, \ldots, f_m^{(t)} \rangle$ *s.t. q* constraints, $h_1(\boldsymbol{x}) \geq 0, \ldots, h_q(\boldsymbol{x}) \geq 0$
- Access to *k* bias and/or noisy sources: $S_1, \ldots, S_k$, and $\mathbf{f}^{(S_j)} = \langle f_1^{(S_j)}, \ldots, f_m^{(S_j)} \rangle$
- $\mathbf{f}^{(S_j)}$ approximates the target $\mathbf{f}^{(t)}$ with variable bias $\delta_j(\boldsymbol{x}) = \mathbf{f}^{(t)}(\boldsymbol{x}) - \mathbf{f}^{(S_j)}(\boldsymbol{x})$
- Intervention cost: $\mathscr{C} = \sum_{e=1}^{T^{(t)}} C_t(\boldsymbol{x}_e^{(t)}) + \sum_{j=1}^{k} \sum_{e=1}^{T^{(S_j)}} C_s(\boldsymbol{x}_e^{(s_j)})$

Our goal is to find a configuration $\boldsymbol{x}^*$ in the target environment that results in Pareto-optimal performance with minimal cost, $\mathscr{C}$:

$$\boldsymbol{x}^* = argmax_{\boldsymbol{x} \in \mathscr{X}} \left(\mathbf{f}^{(t)}(\boldsymbol{x}), -\mathscr{C} \right), \; s.t. \; h_l(x) \geq 0.$$

Causal Multi-information Source Optimization: Method

Causal GP (CGP)

$$f(x) \sim GP(m(x)), K(x, x')$$

$$m(x) = \mathbb{E}[Y \mid do(X = x)]$$

Causal prior

$$K(x, x') = K_{RBF}(x, x') + \sigma(x)\sigma(x')$$

where, $\sigma(x) = \sqrt{\mathbb{V}[Y \mid do(X = x)]}$

Variance estimated from observational data

Causal multi-source multi-output GP

$$\mathbf{f}^{(S_j)} \sim CGP(\mathbf{m}^{(S_j)}(x), K^{(S_j)}([x, i], [x', j]))$$

$$\mathbf{f}^{(t)} = \rho \mathbf{f}^{(S_j)} + \delta(x)$$

ρ is a scaling factor, aligns the $\mathbf{f}^{(S_j)}$ and $\mathbf{f}^{(t)}$

$$\delta(x) \sim CGP(\mathbf{m}_\delta(x), K_\delta([x, i], [x', j]))$$

Captures discrepency between $\mathbf{f}^{(S_j)}$ and $\mathbf{f}^{(t)}$

$$K[x, i], [x', j]) = K_{iputs}(x, x') \,.\, K_{objectives}(i, j) + \sigma(x)\sigma(x')$$

Covariance between two datapoints. Inputs x and x', and objectives i and j

“Ideas don’t come out fully formed, they only become clear as you work on them.”

- Mark Zuckerberg

- Your first paper is not your last paper.
- Don’t try to solve all research questions in a single paper.
- Research is an iterative process.

Lessons Learned

- Establishing clear and regular communication channels.
- Defined Roles and Responsibilities.
- Successful collaboration often requires adaptability to different working styles.

Trust your expertise—reviews are guidance, not gospel.

Awesome$^{\infty}$ Collaborators

Pooyan Jamshidi

UNIVERSITY OF SOUTH CAROLINA

Bradley Schmerl

Carnegie Mellon University

Jason M. O'Kane

TEXAS A&M UNIVERSITY

David Garlan

Carnegie Mellon University

Sonam Kharade

UNIVERSITY OF SOUTH CAROLINA

Javier Cámara

UNIVERSIDAD DE MÁLAGA

Ellen C. Czaplinski

Jet Propulsion Laboratory
California Institute of Technology

Katherine A. Dzurilla

Jet Propulsion Laboratory
California Institute of Technology

Thank you!