

DOI: doi:10.5281/zenodo.7529716
Code: <https://github.com/softsys4ai/care>

This appendix provides additional information regarding CARE. We describe the steps to reproduce the results reported in §IV using CARE. The source code and data are provided in a publicly accessible GitHub repository, allowing users to test them on any hardware once the software dependencies are met.

A. Profiling the Observational Data

To collect the observational data, we develop a profiling tool, *Reval*, which currently supports *Husky* and *Turtlebot-3*. Note that, the following steps are optional since we provide the previously measured observational Data. Additionally, Ubuntu-20.04, and *Ros Noetic* are prerequisites for using *Reval*.

1) *Software Dependencies*: Install the software dependencies for *Reval* using the following commands:

```
$ apt install ripgrep
$ pip install pandas
$ pip install tqdm
$ pip install tabulate
```

Also, the dependencies can be automatically installed using the following commands:

```
$ git clone https://github.com/softsys4ai/Reval.git
$ sh Reval/requirements.sh
```

Download and install *ros_readbag.py* using the following commands:

```
$ wget https://raw.githubusercontent.com/
ElectricRCAircraftGuy/eRCaGuy_dotfiles/master/
useful_scripts/ros_readbagfile.py
$ sudo chmod +x ros_readbagfile.py
$ mkdir -p ~/bin
$ mv ros_readbagfile.py ~/bin/ros_readbagfile
$ ln -si "${PWD}/ros_readbagfile.py" ~/bin/ros_readbagfile
```

Note that, if this is the first time ever creating the `~/bin` directory, then log out and log back in to your Ubuntu user account to cause Ubuntu to automatically add your `~/bin` directory to your executable PATH.

2) *Dependencies for Husky simulator*: Install the dependencies using the following commands:

```
$ apt-get install ros-noetic-husky-simulator
$ apt-get install ros-noetic-husky-navigation
$ apt-get install ros-noetic-husky-desktop
```

3) *Dependencies for Turtlebot-3 physical*: Install the dependencies using the following commands:

```
$ mkdir -p ~/Reval/src/turtlebot3
$ git clone -b noetic-devel https://github.com/
ROBOTIS-GIT/DynamixelSDK.git
$ git clone -b noetic-devel https://github.com/
ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone -b noetic-devel https://github.com/
ROBOTIS-GIT/turtlebot3.git
```

4) *Installation and Running*: For building *Reval* from source use the following commands:

```
$ git clone -b husky https://github.com/softsys4ai/Reval.git
$ source /opt/ros/noetic/setup.bash
$ cd ~/Reval && catkin build
```

For *Turtlebot-3* use `-b turtlebot3`.

For execution, use the following commands,

```
$ source devel/setup.bash
$ python reval.py
```

Additionally, the following arguments can be used during execution:

```
optional arguments:
-h, --help            show this help message and exit
-v, -viz              turn on/off visualization of gazebo and rviz
                      (default: On)
-e, -epoch            number of data-points to be recorded
                      (default: 1)
```

For example, "python reval.py -v off -e 10"

B. Running CARE

CARE is implemented by integrating and building on top of several existing tools:

- *causal-learn* for structure learning.
- *ananke* and *causality* for estimating the causal effects.

1) *Installation*: To build CARE from source, use the following commands:

```
$ git clone https://github.com/softsys4ai/care.git
$ cd ~/care && pip install -r requirements.txt
```

2) *Data*: All the datasets required to run experiments are already included in the `./care/data` directory. Additionally, the `./care/observational_data` directory contains the required observational data to train the causal model.

C. Experiments

We run the following experiments to support our claims.

1) *E1: Root-cause Verification Experiment*: We first train the causal model using the observational data, and compute the ranks of the causal paths (the path's ranks are provided in the `./care/result/rank_path.csv` file). We conduct 50 trials for each rank and recorded the energy, mission success, and evaluation metrics both in *Husky* simulator and physical robot. We provide the result of the trails in the `./care/result/exp` directory. To train the causal model and compute the causal path's rank, execute the following command:

```
$ python run_care_training.py
```

To reproduce the results (§IV-B), we provide several functions in the `care_rootcause_viz.py` script.

2) *E2: Transferability Experiment*: We reuse the causal model constructed from the *Husky* simulator to determine the root-cause of the functional faults in the *Turtlebot-3* physical robot. The following command runs the experiment:

```
$ python run_care_inference.py
```

The list of root causes for different ranks are printed in the terminal along with the accuracy, precision, and recall. To reproduce the results (§IV-C), we provide the `care_transferability_viz.py` script that produces the *RMSE* plot in the `./care/fig` directory.

D. Using CARE with external data

CARE can be applied to a different robotic system given the observational data as a `pandas.DataFrame`. For example, update the `run_care_training.py` script as follows:

```
# read the observational data
df = pd.read_csv('observational_data.csv')
# read all columns
columns = df.columns
# Manipulable variables (configuration options)
manipulable_variables = ['option_1', 'option_2']
# Non-manipulable variables (evaluation metrics)
non_manipulable_variables = ['metric_1', 'metric_2']
# Performance objective (energy)
perf_objective = ['objective_1', 'objective_1']
```

Additionally, we added [instructions](#) to define values for configuration options, changing experimental environment, and define a mission specification during observational data collection.

APPENDIX

E. Configuration Options in ROS nav core

TABLE IV: Configuration options in base local planner

Parameters	Configuration options	Values/Range
Robot Configuration	acc_lim_x	0 - 5
	acc_lim_y	0 - 5
	acc_lim_theta	0 - 6
	max_vel_x	0 - 1
	min_vel_x	-0.1 - 0.2
	max_vel_theta	0 - 1
	min_vel_theta	-1 - 0
	min_in_place_theta	0 - 0.5
	escape_vel	-0.2 - 0
	holonomic_robot	true
Goal Tolerance	y_vels	-0.3 - 0.3
	yaw_goal_tolerance	0.1 - 3
	xy_goal_tolerance	0.1 - 0.4
	latch_xy_goal_tolerance	true, false
Forward Simulation	sim_time	1 - 2
	sim_granularity	0.015 - 0.03
	vx_samples	1 - 30
	vtheta_samples	1 - 30
Trajectory Scoring	controller_frequency	10 - 20
	meter_scoring	true, false
	pdist_scale	0.1 - 1
	gdist_scale	0.5 - 1.5
	occdist_scale	0.01 - 0.05
	heading_lookahead	0.325
	heading_scoring	true, false
	heading_scoring_timestep	0.8
	dwa	true, false
	publish_cost_grid	true, false
Oscillation Prevention	oscillation_reset_dist	0.05
Global Plan	prune_plan	true, false

TABLE V: Configuration options in global planner

Configuration options	Values
allow_unknown	0
default_tolerance	false
use_dijkstra	true
use_quadratic	true
use_grid_path	false
old_navfn_behavior	false
lethal_cost	253
neutral_cost	50
cost_factor	3
publish_potential	true
orientation_mode	0
orientation_window_size	1
outline_map	true

TABLE VI: Configuration options in costmap 2d

Configuration options	Values/Range
footprint_padding	0.01
update_frequency	4 - 7
publish_frequency	1 - 4
transform_tolerance	0.2 - 2
resolution	0.05
obstacle_range	5.5
raytrace_range	6
inflation_radius	1 - 10
cost_scaling_factor	1 - 20
combination_method	true, false
stop_time_buffer	0.1 - 0.3

F. Configuration Setting for Root Cause Verification

TABLE VII: Configuration setting for Energy

Rank 1		Rank 3		Rank 4	
Options	Values	Options	Values	Options	Values
Cost_scaling_factor	10	Cost_scaling_factor	10	Cost_scaling_factor	2 - 20
update_frequency	4	update_frequency	4	update_frequency	1 - 7
publish_frequency	3	publish_frequency	3	publish_frequency	3
transform_tolerance	0.5	transform_tolerance	0.2 - 2	transform_tolerance	0.5
combination_method	0	combination_method	0, 1	combination_method	0
pdist_scale	0.75	pdist_scale	0.75	pdist_scale	0.75
gdist_scale	0.5 - 4	gdist_scale	1	gdist_scale	1
occdist_scale	0.01 - 2	occdist_scale	0.1	occdist_scale	0.1
stop_time_buffer	0.2	stop_time_buffer	0.2	stop_time_buffer	0.2
yaw_goal_tolerance	0.1	yaw_goal_tolerance	0.1	yaw_goal_tolerance	0.1
xy_goal_tolerance	0.2	xy_goal_tolerance	0.2	xy_goal_tolerance	0.2
min_vel_x	0	min_vel_x	0	min_vel_x	0

TABLE VIII: Configuration setting for Mission Success

Rank 1		Rank 3		Rank 4	
Options	Values	Options	Values	Options	Values
Cost_scaling_factor	10	Cost_scaling_factor	10	Cost_scaling_factor	10
update_frequency	4	update_frequency	4	update_frequency	4
publish_frequency	3	publish_frequency	3	publish_frequency	3
transform_tolerance	0.5	transform_tolerance	0.2 - 2	transform_tolerance	0.5
combination_method	0	combination_method	0	combination_method	0, 1
pdist_scale	0.75	pdist_scale	0.75	pdist_scale	0.75
gdist_scale	1	gdist_scale	0.5 - 4	gdist_scale	1
occdist_scale	0.01 - 2	occdist_scale	0.1	occdist_scale	0.1
stop_time_buffer	0.2	stop_time_buffer	0.2	stop_time_buffer	0.2
yaw_goal_tolerance	0.1	yaw_goal_tolerance	0.1	yaw_goal_tolerance	0.05 - 1
xy_goal_tolerance	0.01 - 1	xy_goal_tolerance	0.2	xy_goal_tolerance	0.2
min_vel_x	0	min_vel_x	0	min_vel_x	0