Robustness Testing

Ability to test systems under rare, off-nominal, extreme conditions

Environmental  Software  Hardware

When things break down and assumptions are violated, the system must continue to be safe!
Robustness Testing at NREC
Tested dozens of government and commercial systems since 2011
Robustness Testing at NREC

Built a suite of robustness tools that target different interfaces
Robustness Testing at NREC

Found and helped to fix bugs in over 30 systems that were tested

In Simulation
Message on /joy topic causes self-intersection

In Hardware
Spurious speed command causes speed limit violation

In The Wild
Test value caused self-intersection

With Human Harm
Spurious joint angle causes invalid state and stops safety system from functioning

months later, failure occurred on physical robot and caused irreparable damage
Robustness Testing at NREC

But almost all of this work was done under restricted funding
MOBSTA: Robustness Testing for ROS
Early Research Release
Example System: TurtleBot

Automatic Lidar Parking
MoBSTA: Robustness Testing for ROS

Three Ingredients for Robustness Testing

Nominal Data

Invariants

Mutations
TurtleBot Automatic Lidar Parking

Nominal Field Data Collection

• Setup an enclosed rectangular environment for the robot
• We vary the position of the reflective tape, used to indicate the parking spot
• We vary the starting position of the robot
• Collected a total of 10 log files
TurtleBot Automatic Lidar Parking

Invariants

Valid Cmd Vel

invariant_name: ValidNumbersInvariant
invariant_params:
velocity_topic: /cmd_vel

Command Timing

invariant_name: CorrectCommandsInvariant
invariant_params:
velocity_topic: /cmd_vel
joint_states_topic: /joint_states
bad_command_limit: 10
angle_goal_deg: 5
position_goal: 0.05
distance_goal: 0.5
angle_uncertainty_deg: 3
position_uncertainty: 0.2
parking_spot_x_1: -1.25
parking_spot_y_1: -0.623
parking_spot_x_2: -0.825
parking_spot_y_2: -0.754
MoBSTA supports a wider set of invariants

Node Crashes  Timeliness  Speed Limits

Turning Radius  Staleness  Unsafe Plans
TurtleBot Automatic Lidar Parking

Mutation: Lidar Intensities

topic: /scan:
  - message_intercept: /intensities

mutation:
  - mutation_type: Float32Array_AddToWholeArrayMutator
    mutation_chance: 1
    timeframe_begin: 0
    timeframe_end: -1
    mutation_args:
      valueToAdd: -100
      minArrayValue: 0
      maxArrayValue: 255
      arraySize: 259

sensor_msgs/LaserScan
  std_msgs/Header header
  float32 angle_min
  float32 angle_max
  float32 angle_increment
  float32 time_increment
  float32 scan_time
  float32 range_min
  float32 range_max
  float32[] ranges
  float32[] intensities
We support other realistic perturbations
Sensor inputs, control and planning data, configuration

<table>
<thead>
<tr>
<th>Dust on Lens</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera Distortion</td>
<td>Exposure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NavSatStatus status</td>
</tr>
<tr>
<td>Float64 latitude</td>
</tr>
<tr>
<td>Float64 longitude</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Camera Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>intrinsic parameters</td>
</tr>
<tr>
<td>local plane</td>
</tr>
<tr>
<td>world frame</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Look Ahead</td>
</tr>
<tr>
<td>Large Look Ahead</td>
</tr>
</tbody>
</table>
TurtleBot Automatic Lidar Parking
End-to-End Overview

- Command Timing
- Valid Cmd Vel
- NaN

Bag File
Mutation
Playback
TurtleBot Automatic Lidar Parking

Mutation exposes brittle lidar intensity handling

base_link

Nominal -100 +100

Failed to find parking spot.

Incorrect parking spot
TurtleBot Automatic Lidar Parking

Mutation exposes brittle lidar intensity handling

```
intensity_threshold = 100
...
for i in range(len(msg.intensities)):
    spot_intensity = msg.intensities[i] ** 2 * msg.ranges[i] / 100
    ...
    if spot_intensity >= intensity_threshold:
        ...
```
What’s Next?

Closed-Loop Robustness Testing via Enhanced Simulation

Efficient Search

AutoMOBSTA

Mature Tooling + Workflow Integration
Breaking Bots: Robustness Testing for ROS

We want to hear your thoughts! Do you do robustness testing?