Robot Unit Testing

*depending on purpose
SFB125: Cognition-Guided Surgery

Perception → Interpretation → Action

Knowledge-base

Institute for Anthropomatics and Robotics (IAR)
Intelligent Process Control and Robotics Lab (IPR)
Conceptual system architecture

Medical image processing

Cognition-guided control

Haptic control

Ethernet

ROS.org

GAZEBO Simulation

Collision free path planning
Safety supervision
Nullspace optimization

Multi-mode robot control

Patient

Environment

Automatic
Hands-on
Telemanip
Preliminary remarks

- Not about a specific technical solution, it is about *testing methodology* (i.e. systematic analysis and best practices) *for robotics*.
  - Example implementation uses ROS, rostest and Gazebo.

- Automated testing for robots insofar it is *specific to robots*.


- An Answer to: When is the robotics simulation's realism *sufficient*?
Robot Unit Testing: The big picture

Real Robot

High level interface

Object Detection

Inverse Kinematics

Desc

J

J'

Interpolation

Preprocessing

RGB'

RGB

ΔJ'

Real World

Low level interface

Sensors

Actuators

Simulation

Low level interface

Sensors

Actuators

Robot Unit Testing
Benefits of RUT

- A novel level of *automated* test coverage for robots.

- *All* software components of the real robot are tested together with their *interaction*.

- Writing RUTs often requires *less effort* than good coverage through traditional unit tests.

- Drawback: Test times are significantly longer (setup and teardown).
Requirements for RUT

- Robotics simulator
  - Support for relevant sensors and actuators.
  - *Sufficient* accuracy with respect to relevant properties.
  - World state accessible through (external) interface.

- Robot software
  - Possibility to reroute sensor input and actuator output at start time.
  - Interfaces to tap into communication of components.
  - Robot description reusable for definition of simulated robot.
On sufficiency

- Here a qualitative criterion that is task/test specific

- Properties $P$ of real $r$ and simulated $s$ entities for test $t$

\[ \forall P \in \mathcal{P} : P(s) \iff P(r) \]
\[ D = \sum_{P \in \mathcal{P}} d_P(s, r) \]

- $d_{\text{surface}}(s,r,t)$ ↓
- $d_{\text{volume}}(s,r,t)$ ↑
- $d_{\text{kinematics}}(s,r,t)$ ↑
- $d_{\text{dynamics}}(s,r,t)$ ↓
**RUT costs**

- Code for test cases (create input, pass to interface, check results)
  - If it is hard to create input data for your robot and pass it to its high level interface, testing is not the issue to work on.
  - If it is hard to get ground truth, use another simulator.

- Simulation models
  - Models for collision checking and visualization should be available.

- Simulated actuators with an appropriate interface
- Simulated sensors with an appropriate interface
  - Gazebo, MORSE, V-Rep provide most 'standard' devices
  - For special devices or very low d_p significant effort required.
RUT for ROS: Example scenario
**RUT for ROS**

*Complete launch file:*

```xml
<launch>
  <include file="$(find ipr_models)/launch/lab_lwrs.launch"/>

  <test test-name="test_reachability_lwrs" pkg="ipr_models"
    type="test_reachability_lwrs.py" time-limit="300"/>
</launch>
```

Two test cases (using python rosunit and unittest):

```python
def test_joint_interface_reachable2(self):
    # Close to joint limits
    self.joint_interface_reachable('lwr1', [rad(-169.5), ...

def test_joint_interface_unreachable1(self):
    # Beyond joint limit
    self.joint_interface_unreachable('lwr1', [rad(-170.1), ...
```
RUT for ROS
Outlook:

- Develop generic Robot Unit Testing framework for ROS & Gazebo
- Connect with Continuous Integration Server (jenkins)
- Write more tests (e.g. for self-calibration procedures)
Outlook: Integrate Advanced ROS Network Introspection

http://wiki.ros.org/arni
Summary: Robot Unit Testing (RUT)

- A methodology as theory and practice of automated robot testing
- Simulation must not be perfect!
- Available realism can be exploited by test design!

- Example implementation based on rostest and Gazebo

- The entry barrier to RUT is low, the potential benefits high: (Robot Unit) **Test it!**

*depending on purpose*
Thank you for your attention.

Questions?

andreas.bihlmaier@kit.edu