A Methodology for Testing Mobile Autonomous Robots

IROS 2013

Jannik Laval, Luc Fabresse and Noury Bouraqqadi

car@mines-douai.fr
Outline

1. Context & Problem
2. Existing Approaches
3. Our Safe and Test-based Methodology
4. Report on an Experiment
5. Summary & Future Work
Context & Problem
Specifications / Requirements

Example:
the robot should move straight 4m with 3% of error at most
Context: Robots Quality Assurance

How to ensure that a robot complies with requirements

Specifications / Requirements

Example: the robot should move straight 4m with 3% of error at most
Context: Robots Quality Assurance

How to ensure that a robot complies with requirements using a safe, reusable, and repeatable method?

Specifications / Requirements

Example: the robot should move straight 4m with 3% of error at most
Existing Approaches
Existing Approaches

• Software Testing approaches
  • Koo Chung and al. @ ICCAS’07
  • Biggs @ ICRA’10
  • Liam and al. @ JSEA’10

• Formal Approaches
  • Bensalem and al. @ JOSER’11
  • …

but software only
Existing Approaches

• Simulation-based approaches
  • Seong and al. @ Automation’12

• Simulation + Hardware in the loop approaches
  • Petters and al. @ SIMPAR’08
  • Yen-Hung Chen and al. @ Joser’11
Our Test-based Methodology for Mobile Autonomous Robot
Specifications / Requirements for one product line

Reusable and repeatable tests written once

Safely run tests on robots
Ordering Tests for safety

• 3 Tests Dimensions
  • Sensing vs Sensing&Acting
  • Known vs Unknown environment
  • Static vs Dynamic environment
Ordering Tests for Safety

Test Level 1

Sensing in a Static Known environment

prerequisite for

Test Level 2

Sensing in a Dynamic Known environment

Test Level 3

Acting in a Static Known environment

Test Level 4

Acting in a Static Unknown environment
Acting in a Dynamic Known environment

Test Level 5

Acting in a Dynamic Unknown environment
Ordering Tests for Safety

Test Level 1

Sensing in a Static Known environment

Test Level 2

Sensing in a Dynamic Known environment

prerequisite for

Test Level 3

Acting in a Static Known environment

Test Level 4

Acting in a Static Unknown environment

Acting in a Dynamic Known environment

Test Level 5

Acting in a Dynamic Unknown environment
A Tool for Robot Testing

• BoTest
  • Implemented in Pharo as an extension of SUnit ([http://www.pharo-project.org](http://www.pharo-project.org))
  • Helps to write tests for robots
  • Support expressing dependencies between tests
  • Run tests in a specific order
Experiment
Example

the robot should move straight 4m with 3% of error at most

using its laser for collision detection
Test Level 1

Sensing in a Static Known environment

LaserTest
- testLaserWhenRobotAtBottomLeftOfTheBox
- testLaserWhenRobotAtBottomRightOfTheBox
- testLaserWhenRobotAtCenterOfTheBox
- testLaserWhenRobotAtTopLeftOfTheBox
- testLaserWhenRobotAtTopRightOfTheBox
- testLaserWhenRobotInsideTheBoxFacingObstacleAt2m
testLaserWhenRobotAtBottomLeftOfTheBox
| laser notification allDistances |
| self requestAction: 'Please, put the robot at', |
| the bottom left of the 1x1 box'. |

laser enableNotificationsEvery: 10.
nomination := self notificationOrNilFrom: laser.
allDistances := notification distances.
self assert: allDistances
areLessThan: 0.75
andGreaterThan: 0.25
Example of Laser Test

```plaintext
testLaserWhenRobotAtBottomLeftOfTheBox
  | laser notification allDistances |
  self requestAction: 'Please, put the robot at',
                      'the bottom left of the 1x1 box'.
  laser enableNotificationsEvery: 10.
  notification := self notificationOrNilFrom: laser.
  allDistances := notification distances.
  self assert: allDistances
               areLessThan: 0.75
               andGreaterThan: 0.25
```

Tester action requested to setup the robot pose
Example of Laser Test

```python
def testLaserWhenRobotAtBottomLeftOfTheBox:
    laser = robot.laserService.
    laser.enableNotificationsEvery: 10.
    notification = self.notificationOrNilFrom: laser.
    allDistances = notification.distances.
    self.assert: allDistances
        areLessThan: 0.75
        andGreaterThan: 0.25
```
Example of LaserTest

testLaserWhenRobotAtBottomLeftOfTheBox
| laser notification allDistances |
  self requestAction: 'Please, put the robot at',
                   ' the bottom left of the 1x1 box'.

laser enableNotificationsEvery: 10.
notification := self notificationOrNilFrom: laser.
allDistances := notification distances.

self assert: allDistances
       areLessThan: 0.75
       andGreaterThan: 0.25

Assertion automatically verified

Luc Fabresse
IROS 2013 @ Tokyo
Acting in a Static Known environment

```
**testStraightLineMotion**

| laser |

self requires: PBLaserTest.
self requires: #testEmergencyStop.

self requestAction: 'Put the robot in a corridor (1.3m wide)',
                   'with at least 5m free in front of it'.

laser enableNotificationsEvery: 10.
laser onNotificationDo: [:notif]
  (notif distances anySatisfy: [:each | each < 0.3]) ifTrue: [
   controller emergencyStop.
  ].
controller goTo: 4@0.
5 seconds asDelay wait.
self requestAssert: 'Did the robot travelled a distance',
                   'between 3.88m and 4.12m?'
```
Test Level 3

Acting in a Static Known environment

```
testStraightLineMotion
  | laser |
  self requires: PBLaserTest.
  self requires: #testEmergencyStop.
  self requestAction: 'Put the robot in a corridor (1.3m wide),
  ' with at least 0.5m free in front of it'.
  laser enableNotificationsEvery: 10.
  laser onNotificationDo: [:notif]
  (notif distances anySatisfy: [:each | each < 0.3]) ifTrue: [
    controller emergencyStop.
  ].
  controller goTo: 4@0.
  5 seconds asDelay wait.
  self requestAssert: 'Did the robot travelled a distance',
  ' between 3.88m and 4.12m?'
```
Test Level 3

Acting in a Static Known environment

**testStraightLineMotion**

```python
| laser |

- **self requires**: PBLaserTest.
- **self requires**: #testEmergencyStop.
- **self requestAction**: 'Put the robot in a corridor (1.3m wide)', 'with at least 1m free in front of it'.

- **laser := robot laserService.**
- **laser enableNotificationsEvery**: 10s
- **laser onNotificationDo**: [:notif]
  - **(notif distances anySatisfy**: [ :each | each < 0.3]) ifTrue: [controller emergencyStop.]

- **controller goTo**: 4@0.
- **5 seconds asDelay wait.**
- **self requestAssert**: 'Did the robot travelled a distance', 'between 3.88m and 4.12m?'
```

Tested code
Test Level 3

**Acting in a Static Known environment**

```plaintext

testStraightLineMotion

| laser |
self requires: PBLaserTest.
self requires: #testEmergencyStop.
self requestAction: 'Put the robot in a corridor (1.3m wide)',
                  'with at least 5m free in front of it'.
laser enableNotificationsEvery: 10.
laser onNotificationDo: [:notif]
   (notif distances anySatisfy: [:each | each < 0.3]) ifTrue: [
    controller emergencyStop.
]
controller goTo: 4@0.
5 seconds asDelay wait.
self requestAssert: 'Did the robot travelled a distance',
                  'between 3.88m and 4.12m?'
```

Assertion to be verified by the tester
Running a Test

Specifications / Requirements

the robot should move straight 4m with 3% of error at most

\[ \text{testStraightLineMotion} \ldots \]
Running a Test

the robot should move straight 4m with 3% of error at most

testStraightLineMotion...

Specifications / Requirements
Running a Test

Specifications / Requirements

the robot should move straight 4m with 3% of error at most

testStraightLineMotion...

1 run, 1 passes
Running a Test

Specifications / Requirements

The robot should move straight 4m with 3% of error at most

testStraightLineMotion...

1 run, 1 passes

1 run, 0 passes, 1 requirementFailure
Running a Test

Specifications / Requirements

the robot should move straight 4m with 3% of error at most

testStraightLineMotion ...

1 run, 1 passes

1 requirementFailure

1 run, 0 passes,
Running a Test

- testStraightLineMotion
  - testLaserWhenRobotAtBottomLeftOfTheBox
  - testLaserWhenRobotAtBottomRightOfTheBox
  - testLaserWhenRobotAtCenterOfTheBox
  - testLaserWhenRobotAtTopLeftOfTheBox
  - testLaserWhenRobotAtTopRightOfTheBox
  - testLaserWhenRobotInsideTheBoxFacingObstacleAt2m
  - testEmergencyStop

Requirements are executed first
Running a Test

- testStraightLineMotion
  - testLaserWhenRobotAtBottomLeftOfTheBox
  - testLaserWhenRobotAtBottomRightOfTheBox
  - testLaserWhenRobotAtCenterOfTheBox
  - testLaserWhenRobotAtTopLeftOfTheBox
  - testLaserWhenRobotAtTopRightOfTheBox
  - testLaserWhenRobotInsideTheBoxFacingObstacleAt2m
  - testEmergencyStop
Running a Test

- testStraightLineMotion

  - testLaserWhenRobotAtBottomLeftOfTheBox
  - testLaserWhenRobotAtBottomRightOfTheBox
  - testLaserWhenRobotAtCenterOfTheBox
  - testLaserWhenRobotAtTopLeftOfTheBox
  - testLaserWhenRobotAtTopRightOfTheBox
  - testLaserWhenRobotInsideTheBoxFacingObstacleAt2m
  - testEmergencyStop

1 run, 0 passes, 1 requirementFailure
Summary & Future Work
Summary

- Use robot tests for Quality Assurance and Maintenance

- Robot tests are:
  - apply on a “whole” (hardware+software) and “real” (not simulated) robot
  - reusable and repeatable
  - executed in a safe order according to their level
Future Work

- Automatic test dependencies extraction
- Improve tests automation
- Continuous integration for non-interactive tests
- Dedicated development tools
A Methodology for Testing Mobile Autonomous Robots

IROS 2013

Jannik Laval, Luc Fabresse and Noury Bouraqadi

car@mines-douai.fr