Navigation Testing for Continuous Integration in Robotics

Jaime Pulido Fentanes[1], **Christian Dondrup**[2], **and Marc Hanheide**[1]

[1]Lincoln Centre for Autonomous Systems, University of Lincoln, UK
{jpulidofentanes,mhanheide}@lincoln.ac.uk
[2]School of Mathematical & Computer Sciences, Heriot-Watt University, Edinburgh, Scotland, UK
c.dondrup@hw.ac.uk

Abstract

Robots working in real-world applications need to be robust and reliable. However, ensuring robust software in an academic development environment with dozens of developers poses a significant challenge. This work presents a testing framework, successfully employed in a large-scale integrated robotics project, based on continuous integration and the fork-and-pull model of software development, implementing automated system regression testing for robot navigation. It presents a framework suitable for both regression testing and also providing processes for parameter optimisation and benchmarking.

Introduction

Robust long-term robot operation has many challenges that are yet to be overcome. Many of these challenges arise because in real-world deployments unexpected and unpredictable situations can harm robot performance. Some of these situations can only be dealt by constant software development and update, however, robotic systems are usually composed of many individual components, developed by multiple people, that particular in research environments, work *mostly individually*. This can lead to researchers modifying finely tuned parameters to fit one particular case but harming the robot's performance in other situations. It is for this reason, that systematic testing of the *integrated robot system* and test for *regressions* in the performance is critical. Previous deployments[1] have proven that navigation failures are among the most critical problems impacting a mobile robot's overall autonomy. Navigation performance is usually affected by the respective performance of many different components, such as mapping, robust localisation in dynamic environments, path planning, and motion control. Looking for instance at the probably most popular robot navigation package "ROS `move_base`"*, there are more than 50 parameters alone that affect navigation performance, even disregarding code changes and higher level autonomy functions. In an academic research environment where dozens of developers might be contributing to an integrated system, adopting established methodologies of software engineering and collaborative software development are key to ensure the high quality of the software required to facilitate long-term deployment (in the order of weeks and months) of robots in real-world environments.

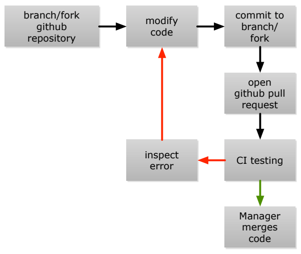
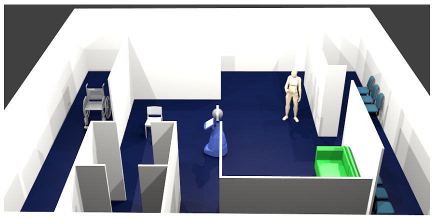


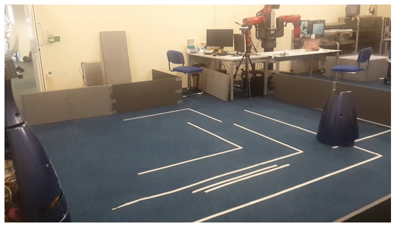
Figure 1. Work-flow.

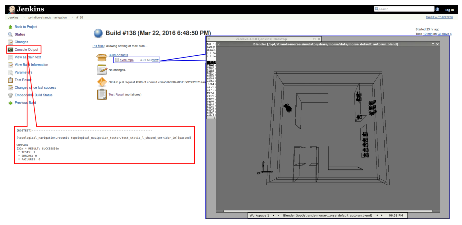
Continuous Integration and Testing in Robotics

The proposed testing regime has been established in the context of the STRANDS† project, which ambition as to develop autonomous mobile robots that run for weeks (a benchmark here was to run 120 days) without expert interventions[1]. As already proposed by earlier research on software engineering in robotics context[2], we adopted the paradigm of *continuous integration* (CI), successfully employed in many larger-scale software-intensive systems (e.g. by Lacoste et al.[3]) and also in robotics[4], facilitated via a dedicated JENKINS CI server‡. Replicating entire systems and testing them systematically has also been attempted by other researchers[5,6], however, the embedding of full navigation testing in the software development work-flow is still rare in the robotics community. The general

*http://wiki.ros.org/move_base
†http://strands-project.eu/
‡https://jenkins.io/


(a)


(b)



(c)

Figure 2. (a) Simulation environment. (b) Robot navigation testing arena at University of Lincoln. (c) Report page.

work-flow of software development captured in Fig. 1 follows the "fork and pull" model[7], which is well integrated with continuous integration. Every so-called pull request in our system undergoes full system testing in simulation (using the "Morse" simulator[8]) through a number of defined navigation test scenarios, build on top of the `rostest` framework[9], and our own topological navigation[10]. However, the same tests can also be run in a real-world "test arena", to verify software commits also with a real robot.

Navigation Test Scenarios

We propose an implementation of specific and repeatable navigation tests[§] as regression tests in a software development environment support by CI paradigms. The aim to minimise regression of navigation behaviour by software updates, but also can be used to develop and benchmark new navigation components or settings, with this purpose in mind we have developed a series of test scenarios. These test scenarios are executed in the continuous integration server every time there is a change to the *STRANDS navigation* stack. If one of these tests should fail, the system's maintainer is notified and can access a report page where they can see what exactly produced the failure and take measures to solve possible issues. Fig. 2c shows the report page on the continuous integration server, this report is comprised of a video report on which a video of the simulated test for this version of the software can be seen, and a link to the console output log where the results of the navigation tests can be found[¶]. It shall be noted that *critical* as well as *supplementary* scenarios are defined, with the first leading to rejection of any pull request of new code (a true critical regression), and the latter only being considered for performance analysis.

To successfully test the software the testing must be as realistic as possible, but at the same time tests have to be executed in a controlled environment where the conditions are as identical as possible every time. Additionally, tests should be run in an automated set-up where not only the robot should be able to move in the testing environment, but also the objects must move in a controlled manner. To achieve this we opted for the development of tests that could be performed in simulation and real-world system, with a transition as seamless as possible. Simulated tests are executed in a tailored environment (figure 2a) where all the scenarios can be tested and the objects and robot position are part of the test definition. On the other hand, real-world testing requires a human operator and supervisor. A typical set-up for this kind of testing is modifying an open space using some kind of panels for each test as seen in figure 2b, resembling the simulated test arena. The test also in real-world utilise the same test definition as in simulation allowing to run entire scenarios automatically at the push of a button also in real-world.

Discussion & Conclusion

Driven by demands of mid- to large-scale collaborative software development of robotic systems, we have proposed a framework for continuous integration focusing on automated testing of robot navigation. This framework has been productive in the EU STRANDS and ILIAD projects for more than 3 years, as has played an essential role to ensure robust system performance in real-world scenarios. The proposed work-flow has also been utilised to provide a framework for automated benchmarking of robotics algorithms[11], and will further be developed for testing full system capabilities, beyond navigation.

[§]See `https://github.com/strands-project/strands_navigation/tree/indigo-devel/topological_navigation/tests` for the specification of our tests.

[¶]`https://www.youtube.com/watch?v=2ROycER60t8`

References

1. Hawes, N. *et al.* The strands project: Long-term autonomy in everyday environments. *IEEE Robotics Autom. Mag.* **24**, 146–156 (2017). DOI 10.1109/MRA.2016.2636359.
2. Lier, F., Schulz, S. & Lütkebohle, I. Continuous Integration for Iterative Validation of Simulated Robot Models. In *Proc SIMPAR*, 101–112 (2012). DOI 10.1007/978-3-642-34327-8_12.
3. Lacoste, F. J. Killing the Gatekeeper: Introducing a Continuous Integration System. In *2009 Agile Conference*, 387–392 (IEEE, 2009). URL http://ieeexplore.ieee.org/document/5261054/. DOI 10.1109/AGILE.2009.35.
4. Mossige, M., Gotlieb, A. & Meling, H. Testing robot controllers using constraint programming and continuous integration. *Inf. Softw. Technol.* **57**, 169–185 (2015). URL http://www.sciencedirect.com/science/article/pii/S0950584914002080http://linkinghub.elsevier.com/retrieve/pii/S0950584914002080. DOI 10.1016/j.infsof.2014.09.009.
5. Ernits, J., Halling, E., Kanter, G. & Vain, J. Model-based integration testing of ros packages: A mobile robot case study. In *Mobile Robots (ECMR), 2015 European Conference on*, 1–7 (2015). DOI 10.1109/ECMR.2015.7324210.
6. Lier, F. *et al.* Towards Automated System and Experiment Reproduction in Robotics. In *Proc IEEE Conf on Intelligent Robots and Systems* (2016).
7. Gousios, G., Pinzger, M. & Deursen, A. v. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, 345–355 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2568225.2568260. DOI 10.1145/2568225.2568260.
8. Lemaignan, S. *et al.* *Simulation and HRI recent perspectives with the MORSE simulator*, vol. 8810 (2014).
9. Foundation, O. S. R. rostest - ros wiki (2015). URL http://wiki.ros.org/rostest. Accessed: 2016-03-21.
10. Pulido Fentanes, J., Lacerda, B., Krajník, T., Hawes, N. & Hanheide, M. Now or later? predicting and maximising success of navigation actions from long-term experience. In *International Conference on Robotics and Automation (ICRA)* (2015).
11. Marc Hanheide Tomáš Vintr Keerthy Kusumam Tom Duckett, T. K. Towards automated benchmarking of robotic experiments. In *ICRA Workshop on Reproducible Research in Robotics* (2017).