Object Recognition and Path Smoothing Robot, Phase 2
Test Plan

University of Washington Tacoma, School of Engineering and Technology

Authors:
Ammon Dodson ammon0@uw.edu
Alex Marlow alexmarlow117@gmail.com
Jake McKenzie jake314@uw.edu
Distribution: Matthew Tolentino
Document state: draft
Modified: November 11, 2018

Contents

1 Revision History

- v0.1: In this version of the testplan we defined the scope of the problem with the introduction, test items and approach.
- v0.2: In this version of the testplan we defined the test items, schedule and approach.

2 Introduction

2.1 Purpose

This test plan describes the testing approach and overall framework that will drive the testing of the ORPS-Robot version 0.1 – The Object Recognition and Path Smoothing robot. The document introduces:

Test Strategy: These are the rules the test will e based on including the givens of the project (e.g.: start / end dates, objectives, assumptions); description of the process to set up a valid test (e.g.: entry / exit criteria, creation of test cases, specific tasks to perform and scheduling)

Execution Strategy: describes how the test will be performed and process to identify and report problems, and to fix and implement fixes.

Test Management: process to handle the logistics of the test and all the events that come up during execution (e.g.: communications, escalation procedures and risk mitigation)

2.2 Project Overview

The ORPS-Robot will be a platform for validating the research of Michael McCourt and a scheme for exploring object recognition via OpenCV with Robot Operating System, which is a powerful framework for writing robot software. There will be a demonstration of Simultaneous Localization and Mapping (SLAM). Together this will demostrate a "finder robot" with applications in search and rescue and threat detection. Additionally there will be beacon triangulation and/or GPS to fuse additional location information into the SLAM or finder functionality.

2.3 Audience

Collaborative robotics software development for research in control systems for path smoothing. Collaboration in academic research is usually thought to mean equal partnership between two academic faculty members who are pursuing mutually interesting and beneficial research. In our case we are creating a platform which will serve Dr. McCourt research where deep understanding of the control system in place is not require on our part and a deep understanding of the robot are not require on the part of Dr. McCourt.

Creating truly, robust, general-purpose robot software is hard. As undergraduates using robot operating system framework allows us to encompass solving robotics problems in real-world variations in complex tasks and environments that no single individual, laboratory, or institution could hole to create completely on their own from scratch. The audience for this device is us, as it serves our education.

Applications in path smoothing and object recognition used in ORPS-Robot project are heavily used in semi-autonomous and autonomous vehicles. The knowledge gain in applying these skills in the ROS ecosystem should serve us to gaining a greater knowledge of the systems and practices of both control systems and object recognition.

3 Test Items

3.1 Hardware Test Items

These will have more descriptions in later revisions.

360° LiDAR for SLAM & Navigation

Rasberry Pi 3 Model B

32-bit ARM Cortex-M7 OpenCR

DYNAMIXEL wheels

Li-Po Battery 11.1V 1,800mAh

Xbox 360 Controller

Marvelmind Sensors

3.2 Software Test Items

At this point in the process we do not knwo what the overall architecture of the software will look like but we can have a general plan of attack for writing software tests using ros-test. According to Paul Ammann's Introduciton to Software Testing, there are five levels of software testing:

Level 0 There is no difference between testing and debugging.

Level 1 The purpose of testing is to show correctness.

Level 2 The purpose of testing is to show that software does not work.

Level 3 The purpose of testing is not to prove anything specific, but to reduce the risk of using the software.

Level 4 Testing is a mental discipline that helps all IT professionals develop higher-quality software.

For our purposes levels 0 through 2 will be used heavily in this process while level 3 and 4 will only be used when needed. To accomplish the task in level 0 we will be making heavy use of first GDB (C debugger) and PDB (Python debugger) along with Valgrind. Levels 1 and 2 will be tackled, hopefully, by making heavy use of *rostest* which is an extension of roslaunch that enables roslaunch files to be used as text fixtures. Due to complex behaivors involved in this project we need to write tests and move on from functionality. When we introduce new functionality to the to ORPS-robot it will need to pass the old tests we wrote for it. if everything is functioning properly there should be no need to re-write our old tests. This will likely only be done with software strict aspects of the project as hardware is subject to change as the implementation changes over time. These tests, at least of the framing of this writing, will be to test the linking of different scripts and files written in the overall project.

3.3 System Test Items

For the 360° LiDAR for SLAM & Navigation we will need to consult three seperate resources for testing the LiDAR. The first will be a github wiki page entitled "How to use rplidar" which is just a walkthrough of how to get rplidar ros packarge working. While this just gives a short introduction to rplidar, a more up to date tutorial on quickly getting rplidar up and running can be found on Service Engineering Research Area website under "From unboxing RPLIDAR to running in ROS in 10 minutes flat". The first goal with LiDAR appears to be establishing the range of the device as this what determines the quality of the information used in navigation. Given more time there will also involve be static and dynamic tests based on scanning calibration, speed of the LiDAR, vehicle speed, laser position feedback. Further testing will involve the use of a tutorial on husarion.com entitled

"SLAM navigation". According to the Bachelor's Thesis of Felix Feik from Technical University of Munich, the most difficult factor in indoor mapping using LiDAR is that the laser sensor always has the position or your map building and pose estimation will fail. The LiDAR needs a consistent, relative motion to map a building correctly. Glass doors and other visible objects will not be accounted for by the LiDAR as the signal will not penetrate glass. Long story short, to use SLAM properly we will need to test take into account the limitations of the device and leverage the libraries provided by robot operating system to do so.

As for testing our controller that will make heavy use of the joy package in ROS which is a library for generic Linux joysticks.

TALK ABOUT MARVELMIND eventually

4 Approach

Our tests will be derived from the requirements and specifications of our project and will follow the following methodological approach:

> Acceptance Testing: Assess system (software & hardware) interaction with respect to user needs. For example, making sure the controller is responsive or camera is functioning adequately.
>
> System Testing: Assesses system interaction to architecture design and overall behaivor. For example a series of rostests that test the linking of each submodule we write for the overall project.
>
> Integration Testing: The continous conjoining of different software artifacts to serve the overall project development. For example, a test that ensures two individual modules are linked properly.
>
> Unit Testing: Assesses system interaction with respect to implementation of specific subsystems. For example, a test that ensures the controller is functioning properly standalone through the joy package.

While these levels are emphasised in terms of when they are applied, it is more important to distinguish the types of faults listed above. They are all a continious part of the design and build process. At the end of the day there is an entire spectrum of testing, and much of it is hard to distinguish from development itself. Most tests look like unit tests and are quite simple. Other tests are more subtle, complex and critical. For such tests we need to use the tools laid out in this document to check our implemetation. In the testing process the standard behaivoral model should be employed, where execution of a program or task is respented by a behaivor. A behaivor is sequence of states while a state is an assignment of values of variables. Our program should be tested and modeled by a set of behaivors we define, representing possible executions.

Safety when testing can be specified using two things:

- The set of possible initial states.
- A next-state relation, describing all possible successor states of any state.

The above is a mindset that integrates the testing process into the development process.

5 Fail Criteria

6 Testing Deliverables

7 Roles

8 Schedule

In the testing of the different components of the final system there are always possibilities for unpredictable delays in the schedule, caused by either the testing or by the design. To stop these delays from having an effect on the overall project completion time there are a couple safeguards in place to ensure the project remains on schedule.

The first safeguard in place is the breaks in between quarters. In the estimation for completing specific tasks, the breaks between quarters was not included as optional time. This allows for two, $2-3$ weeks of time, which are completely clear of other curricular activities, to catch up on the schedule or to get ahead.

If the breaks in between quarters is not enough time to handle the delays the first task besides stretch goals which will be allowed to be shortened or removed, is the testing of the radar sensor. The completion of the Marvel Mind positional sensors is necessary for the testing of Dr. McCourt's algorithm, so the robot's primary purpose will still be able to function without a completely tested radar system.

Testing the controller and robot movement functionality is scheduled for the first two weeks after the completion of getting the controller and the robot running. The current predicted date for those two weeks is Jan 7^{th} to Jan 21^{st}.

The testing of the Marvel Mind positional sensors is scheduled for the first 4 weeks after the completion of the installation of both the Marvel Mind sensors and the radar sensor. This time segment is currently supposed to be April 1^{st} to April 28^{th}.

The testing of the radar sensor will be concurrent with the testing of the Marvel Mind sensors, however the marvel Mind sensors will take priority. If due to any past delays, the sensor testing is not completed by the time the algorithm testing should start, the testing will move on to that of the algorithm, and the radar testing will be fit in if there is any extra time.

On completion of the Marvel Mind sensors testing, we will be able to test our implementation Dr. McCourt's algorithm. This is scheduled for the 4 weeks after the completion of the Marvel Mind sensor testing. The current expected dates are April 28^{th} to March 26^{th}.

If all the previous tasks are completed before March 26^{th}, we will attempt to complete the image recognition stretch goal. At this point there would only be this optional task at hand, so the testing would last as long as it could before project presentation day. The

decision of showcasing this feature will depend on how well it is running by that day.

9 Testing Risks and Mitigation

10 Approvals