Robotics Computation

This wiki space is a home for documentation concerning robotic hardware and software integration at NPS. The goal is to generate self-serve solutions to common problems faced by those interested in developing "robotic" systems - where "robotic" refers to any computational system that interacts with the environment. Below is a growing number of community-generated pages that describe potential solutions based on the NPS-specific tools.

Guidelines

This wiki is a work in progress. It requires that members of the community (you) contribute their knowledge in the form of working examples of NPS-specific solutions. If you contribute your experience it will likely save future members lots of time as they try to get things working. Everyone in the NPS community that is logged into the wiki system should be able to edit these pages and add new pages. Here are some rough guidelines/thoughts for adding new material:

- **Collaboration**: The reason to document this work as a wiki, rather than a static document, is to allow collaboration. If you find bugs, better solutions, unclear instructions or anything that would benefit the community, please add it to the wiki. This only works if we are both consumers and creators of the content.
- **Audience**: The audience for these pages is someone familiar with the individual software tools (MATLAB, ROS, etc.) but trying to link them together into a system. Write the document that you wish you had at the start of your project.
- **Goldilocks**: Try to include enough detail so that the reader could reproduce your work, but not so much that you are recreating what already exists. For example, we aren't adding to the existing general MATLAB/ROS/Linux documentation (there is lots of that out there), instead we are providing specific instructions for an individual implementation.
- **Troubleshooting**: If you have problems or questions, please use the Questions link on the left panel.
  If you want to help answer these questions, please "watch" the Questions page.

Getting Started

There is some foundational knowledge that will help you make sense of these examples. A minimal getting started scenario might be as follows

- A basic familiarity with the Linux command line interface. The first 5 sections of Learning the Shell (user feedback would be good here! If you find a tutorial you like, please share.)
- The ROS Beginner Level Tutorials
- Mathworks Getting Started with Robotics System Toolbox
- Git (Optional). It would be good to get a high-level idea of how version control (e.g., git) works. This is not strictly necessary to get started, but it will help resolve some of the mystery. This seems like a reasonable first tutorial http://www.sitepoint.com/git-for-beginners/

Software Tools

This page is not a resource for these individual software tools, this documentation (tutorials, APIs, online courses, etc.) is mature and available elsewhere. For each of the software tools we do provide pages with community-recommended external resources.

- Ubuntu Linux
- ROS
- MATLAB/Simulink
    - Mathworks ROS Toolbox: Tips and Tricks
    - MATLAB on Linux - Windows Shortcuts
    - Simulink Pacer Block
    - Simulink ROS Idioms
    - Standalone ROS Nodes from MATLAB/Simulink
- Git - Software version control repositories
    - Setting up a ROS package from Git

Concepts and Utilities

- Local Coordinate Frames
- Rosbag: Post-Processing

NPS Development Environments

The most difficult program is often "hello world". To write and run the hello world program often requires a number of ever-changing, platform specific steps to setup a development environment that allows for writing the first few lines of functional code. For robotic applications the number of computational tools is rapidly expanding. These pages document an NPS-specific set of known functional development environments.

- Ubuntu 14.04 (trusty) + MATLAB 2015b + ROS Indigo Igloo: Includes a number step-by-step instruction on setting up an NPS development environment, including setup scripts.
- Initial Setup: User Account and Connecting to NPS Network: Instructions for a new user with admin access to an existing Ubuntu installation. Adding a user, setting up the network, etc.
- The development environment is instantiated in the CAVR Robotics Computation Lab on the mezzanine of the CAVR lab.
- A Development Environment under evaluation for use onboard the Pioneer mobile robots is the CarPC w/Ubuntu+ROS. A guide for setting up the dev environment on this hardware can be found here: CarPC setup: SSH server+auto logon to Campus Wi-Fi+Ubuntu+ROS . This dev environment should allow for the onboard CarPC to automatically join the Campus Wi-Fi network (NPS Wireless) on bootup and start an SSH server for remote logon.

Recipes

The key challenge of robotic computation is getting a bunch of individual hardware and software bits to work together in as an integrated system. The pages below are working examples of specific collections of tools that have been successfully developed here at NPS.

Hardware Recipes

The proliferation of powerful, cost-effective computational solutions has made robotics applications available to a much broader audience of users. On the other hand this ever increasing wide variety of possibilities makes it challenging for new users to select suitable platform. Below are a few hardware tools that are particularly suitable for NPS users.

- ODROID

Aerial Platforms

- ARDrone, MATLAB, ROS and Piksi RTK-GPS module
- ODROID using Simulink/ROS for feedback control using GPS/IMU (Microstrain) and servos: Simple feedback control example.
    - ROS Sensor, MATLAB Control, Multiple Machines: Example of having a sensor, connected to a Linux machine which talks to MATLAB on a windows machine.

Ground Platforms

- Husky Checkout
- Pioneer
    - Car PC Setup for Pioneers
    - Pioneer P3AT Checkout: Connecting to P3AT via serial, driving via keyboard and visualization from a single PC.
    - Pioneer P3AT Wireless Joystick Control: Using two computers (onboard and offboard) to control a mobile P3AT via joystick.
    - Two Pioneers: Leader and Follower: Running two pioneer robots simultaneously.
- Turtlebot3: Main page

Marine Platforms

- Clearpath Robotics Kingfisher USV

Sensor/Actuator/Input Recipes

- April Tags with ROS
- UM7 IMU
    - UM7 Setup: Calibration, KF settings, etc.
- Joystick Control: Wired Control of Pioneer P3AT
- GPS: Garmin GPS18x PC
- ThingMagic RFID
    - ThingMagic USB Pro RFID Reader, Pioneer Integration
    - ThingMagic USB Pro RFID Reader, Turtlebot3 Integration
- Hokuyo UTM-30LX Scanning Laser Rangefinder

Simulation Recipes

- Simple waypoint control of Husky in Gazebo. Illustrates three equivalent control implementations: Simulink, MATLAB and Python.
- Multiple Husky robots in Gazebo - control via interactive twist server in rviz
    - Written up on Clearpath Robotics Blog
- Multiple Husky robots in Gazebo with Simulink control
- Pioneer 3AT in Gazebo
- Two Pioneers in Gazebo
- Adding Kingfisher/Heron Model to Gazebo
- Pioneer in Gazebo, GPS/IMU Navigation, MATLAB OOP Waypoint Guidance
- Pioneer in Gazebo, GPS/IMU Navigation, MATLAB OOP Line Following Guidance

MATLAB/Simulink

- USV Heading Control using ROS and Simulink

Projects

- Pioneer Preparation for Spring 2017
- MultiRobot Control Projects 17-3
- Multirobot Control Final Projects 19-3

Miscellaneous

- MultiMaster

Contact

- Email list: coming soon...

Recent space activity

Rascon, Andreina (LT)

Robotics Computation Lab updated Feb 05, 2020 • view change

McClain, Richard (Rick) (LT)

April Tags with ROS updated Oct 16, 2019 • view change

Van Houtte, Lee (CIV)

Kingfisher Checkout updated Oct 01, 2019 • view change

Bingham, Brian (CIV)

Robotics Computation Lab updated Jun 18, 2019 • view change

Search and Reacquire commented Jun 14, 2019

Space contributors

- Rascon, Andreina (LT) (18 days ago)
- McClain, Richard (Rick) (LT) (130 days ago)
- Van Houtte, Lee (CIV) (145 days ago)
- Bingham, Brian (CIV) (250 days ago)
- Tsatsanifos, Theodoros FORNATL, GR (256 days ago)
- ...