Funded by NSERC USRA

at UOIT

Supervisor: Dr. Scott Nokleby

2016

Implementation of ROS and MoveIt with a Light Weight Manipulator System

ANNA SAFONOV

Contents

Abstract 2

Introduction 3

Light-Weight Manipulator System 1.0 Structure 4

Overview of PhantomX Pincher, WidowX Robot Arm, and Clam Arm 5

Considerations in Structural Improvements for LWMS 8

Developing Control Stack for LWMS 10

Summary 15

List of Sources 16

Abstract

The research objective for this project was to investigate the implementation of ROS (Robotic Operating System) and MoveIt software for real-time remote control of LWMS (Light-Weight Manipulator System). This manipulator is intended as a part of a UAV platform for service works on high voltage lines and oil pipelines.

Research methods followed throughout this project included official ROS, MoveIt and Gazebo documentation and tutorials, as well as the rich, detailed, and informative discussions of the robotic community members on official ROS.org Answers forum, Google forum, GitHub Issues for relevant repositories, and Trossen Robotics forum.

At the end of this project, three major tasks were accomplished. Firstly, three ROS packages were created specifically for the configuration of the LWMS to allow simulation and motion planning in Gazebo, RViz, and MoveIt.

Secondly, ArbotiX control package (arbotix_python) was configured for the LWMS to allow low level control of manipulator joints via ArbotiX GUI.

Lastly, extensive research into the topic of real-time control, motion planning and inverse kinematics for robotic arm-like manipulator systems revealed that for full real-time control in MoveIt, LWMS-specific controllers and an interface between controllers and MoveIt will have to be written in C++ or Python. Several options are available to accomplish this. These options will be discussed in the report along with resources and tools available for support.

All implemented features were tested for performance and stability on Linux Ubuntu 14.04 using ROS Indigo, and current stable versions of MoveIt, Gazebo, and Rviz.

Introduction

The purpose of this project was to research implementation of ROS and MoveIt for real-time remote control of a light-weight manipulator system (LWMS from here on). The LWMS is a part of a larger project by Dr. Nokleby's Mechatronics and Robotic Systems Laboratory on developing an UAV system with a manipulator that can be used for repair works on high-voltage lines, oil and gas pipelines, and other field applications where dangerous conditions or poor accessibility would pose great risk for workers.

The LWMS was designed by a group of upper-year undergraduate students for 2015-2016 Capstone Design Project. It is a 6-DOF (degree-of-freedom) robotic arm with three types of Dynamixel servos in series, custom-made carbon fiber brackets and a two-gear reduction system to increase torque and weight-lifting capabilities. All design specifications and calculations are provided in the Capstone Report, with just a short overview in this section.

The LWMS has some unique features and functionality in the design. Robotic arms of similar size and functionality currently available on the market are usually made with a plastic skeleton (i.e. Trossen Robotics' PhantomX Pincher Arm and WidowX Robot Arm). The LWMS carbon fiber brackets decrease its overall weight and improve its durability, which is essential considering the conditions of its intended use. However, the current LWMS bracket design has weak points. Several suggested improvements will be discussed in later sections.

The three types of servos used in LWMS are AX12, AX18, and MX64 Dynamixel actuators. All actuator specifications may be found on the official Trossen Robotics website.

The manipulator is controlled via an ArbotiX-M microcontroller by Trossen Robotics. For Capstone Project, the microcontroller was loaded with a closed loop C++ program based on Bioloid firmware and software control stack. This pypose sketch enabled the manipulator to perform a series of preset demo poses. However, real-time control was not a specification for the Capstone design of this manipulator. Thus the main goal for this project was to investigate the implementation of remote real-time control for LWMS.

ROS was chosen as the framework for writing the control stack for LWMS because the UAV systems under development in the lab use ROS. Using a common framework will make integration of the quadcopter and the manipulator easy, providing a stable environment for control of both systems. MoveIt is the ROS-based software that provides motion planning capabilities, kinematics, control and navigation, with a user-friendly graphic interface, so the arm can be manipulated with precision to perform various tasks even if it is out of sight.

The following report is intended as a guide on what has been accomplished throughout this project, with analysis of the researched options into developing a stable control stack in ROS and MoveIt, detailed explanation of features that have been implemented and tested, limitations, and potential future for this project.

Light-Weight Manipulator System 1.0 Structure

Bottom to top:

1. J1: shoulder_pan_joint: AX12, ID 1
2. L1 link
3. J2: shoulder_pitch_joint: MX64, multi-turn, gear reduction mechanism, ID 2
4. L2 link
5. J3: elbow_pitch_joint: MX64, multi-turn mode, ID 3
6. L3 link
7. J4: elbow_roll_joint: AX18, ID 4
8. L4 link
9. J5: wrist_pitch_joint: AX18, ID 5
10. L5 link
11. J6: wrist_roll_joint: AX12, ID 6
12. L6 link
13. J7: gripper_joint: AX12, ID 7
14. L7: pincher gripper

Overview of PhantomX Pincher, WidowX Robot Arm, and Clam Arm

The Capstone Design Project Group did a brief overview of a few robotic manipulators available on the market that are of close configuration to specifications for the LWMS for the UAV project. However, during research for this project, several other options have been noted and their relevance to LWMS will be discussed below.

PhantomX Pincher

PhantomX Pincher is a 5 DOF manipulator arm by Trossen Robotics (See Table 1 and Table 2 below). It contains only one type of servo in all five of its joints, the Dynamixel AX12 actuator. This manipulator also comes will all the documentation, specifications, an ArbotiX M-microcontroller, as well as stable and regularly maintained firmware and ROS-compatible software packages. While there are numerous stable and maintained ROS packages to interface the PhantomX Pincher with RViz, Gazebo, and – most importantly for the goals of this project – MoveIt, the arm does not have the required design specifications. PhantomX Pincher was used at the early stages of this project as a learning tool for ROS, ArbotiX packages, and MoveIt.

With respect to this manipulator, of particular note are two ROS meta-packages from the TU Dortmund GitHub repository that provide control functionality based on official ArbotiX control package named arbotix_python, and an interface for MoveIt motion planning and inverse kinematics. The PhantomX Pincher and the LWMS share the same end effector, and the URDF description of the PhantomX was adapted for LWMS.

These two meta-packages as well as extensive documentation on them may be found at:

https://github.com/rst-tu-dortmund/phantomx_rst

https://github.com/rst-tu-dortmund/pxpincher_ros

WidowX Robot Arm

WidowX Robot Arm is also a product of Trossen Robotics, however it was not mentioned in the Capstone Design Report. This is a 6 DOF manipulator with a design and most of its specifications very similar to LWMS. The major difference between the two comes from the weight of WidowX (see Table 1), which is 600 g heavier than LWMS and significantly over the required weight specification. However, the most likely cause of this weight difference is the plastic brackets that form the links of WidowX Robot Arm. The LWMS has carbon fiber links, which compared to plastic, greatly reduce the weight of the manipulator and increase its durability.

WidowX also does not have a gear reduction mechanism, which may account for a slightly lower payload at full extension capability. However, the lower payload may also be the result of plastic links, which are not as strong carbon fiber. In addition, it uses MX28 actuators instead of AX18 as in LWMS. Compared to AX18, MX28 is about 20 g heavier and provides a continuous turn functionality, while AX18 is limited to a range between 0 and 300 degrees.

All in all, WidowX Robotic Arm and LWMS demonstrate very similar design and capabilities, including a specific shape of the second linking bracket in both manipulators. However, it seems like the authors of Capstone Design Project were not aware of this product at the time.

The WidowX also has extensive documentation, tests, and specifications available on the official Trossen Robotics website, along with the official ArbotiX firmware and software packages, which makes further development of this design convenient.

Clam Arm

Clam Arm is a project by a PhD student at Cornell Labs of Colorado University, David T. Coleman. Unfortunately, Clam Arm is a past project and lab's servers crashed, so no extensive documentation or adequate support is available at this time.

Clam Arm is a 7 DOF custom built robotic arm with four-finger end-effector. All of its code and related documentation, including MoveIt motion planning, inverse kinematics plugins and controllers based on a Dynamixel stack are open source and available on GitHub at https://github.com/davetcoleman/clam.

Initially, it was attempted to configure the available code to fit LWMS to see if all tested functionalities of Clam Arm may be transferred to LWMS. However, it became evident that a lot of maintenance and a thorough understanding of C++ programming for robotics, ROS and MoveIt would be required to revive this meta-package and configure it for LWMS. Though the repository has not been maintained since 2013 and is not currently functional, it was extensively used as a source of reference throughout this project, and as a guide to developing some ROS packages for LWMS (notably robot_desciprion ROS package) and adapting proper naming conventions in LWMS code and supporting files.

The known features and design details of Clam Arm are provided in Tables 1 and 2 below.

Table 1. Features of various robotic manipulators of same category as LWMS

Features	Specifications	PhantomX Pincher	WidowX Robot Arm	Clam Arm	LWMS
DOF (degrees-of-freedom)	6	5	6	7	6
weight (g)	<750	550	1330	N/A	732
horizontal reach (cm)	31	31	37	N/A	31
vertical reach (cm)	35	35	51	N/A	39
payload at full reach (g)	550	250	400	N/A	500

Table 2. Joint design of various robotic manipulators of same category as LWMS

Joint	PhantomX Pincher	WidowX Robot Arm	Clam Arm	LWMS
shoulder_pan	AX12	MX28	AX12	AX12
shoulder_pitch	AX12	MX64	EX106	MX64
elbow_pitch	AX12	MX64	RX64	MX64
elbow_roll		MX28	RX64	AX18
wrist_pitch	AX12		RX28	AX18
wrist_roll		AX12	RX28	AX12
gripper	AX12	AX12	AX12	AX12
gripper			AX12	

Considerations in Structural Improvements for LWMS

L2 Design

As discussed earlier, the design of the second link, L2, between MX64 servo ID 2 and MX64 servo ID 3, is similar in shape to the corresponding link design in WidowX Robot Arm. Such shape of the link fixes the servo ID 3 in a horizontal position with actuator rotating head turned 90 degrees counterclockwise. This limits the effective range of elbow_pitch joint to approximately 180 degrees, with L3 maximum and minimum positions at 90 and -90 degrees with respect to centered zero of the actuator. The Capstone Design Project group stated that there is no particular reason for this design and that it is not implemented for additional structural support. For this reason, changing L2 to a standard type bracket link was considered as an option, and an appropriate link was designed and modelled in Siemens NX 9.0.

However, after seeing same design implemented in WidowX Robot Arm for greater support of the elbow_pitch joint under the conditions of a payload, it seems that the L2 link of LWMS should be left as is currently designed, at least until some tests are done to determine the optimal shape. The range and the workspace of the manipulator is not affected by the limited range of motion of the elbow_pitch joint, so there is no practical reason for changing.

Horizontal Offset of shoulder_pitch Joint

In the current design, the shoulder_pitch joint is offset with respect to the shoulder_pan joint by approximately 3 cm to the left on the horizontal plane. This offset, although it does not affect the tested functionality of the manipulator, does shift the center of the mass of the manipulator slightly. This manipulator is designed to be mounted on top of a UAV, and a slight shift in center mass may not only bring additional complexity to the structure modelling and motion planning, but more seriously may lead to a disbalance during flight and operation.

Centering the two actuators over each other is technically a simple task and will reduce the complexity of the manipulator structure and its center mass.

Two-Gear Reduction System

LWMS has a two-gear reduction mechanism implemented at the second joint, the shoulder_pitch. This is arguably its most important difference in design from WidowX Robot Arm and other manipulators of the same type. The mechanism was implemented by the Capstone Design group to reduce torque, thereby increasing the payload handling capabilities of the arm at full extension.

At the moment, the gears in the design are plastic, and several teeth on both gears are already broken or worn out. For further testing and use of the manipulator, it would be beneficial to change the plastic gears to metal.

General Link Design Shortcomings

In the current implemented design of LWMS, each carbon fiber link consists of several brackets held together with epoxy. This is a point of weakness in the design, as the links tend to break along the epoxy seams during manipulator operation. It would be beneficial for the overall durability of the manipulator to redesign the links as wholesome carbon fiber structures.

Developing Control Stack for LWMS

The next section will be concerned with the main goal of this project – implementing real-time control for LWMS with ROS and MoveIt. Steps that have been researched and accomplished will be discussed in detail, issues will be analyzed, and possible future directions for this project will be suggested.

As discussed earlier, at the start of this project, the LWMS design was complete. It was built and tested few months prior, and had a Bioloid-based sketch on the microcontroller that enabled it to take on several static poses. No real-time control was available for the manipulator.

In the course of review of the existing code on the microcontroller, it was noted that one of the poses was not configured correctly and the second link of the manipulator (L2) collided with the base when moving into the flight pose, creating additional tension on the shoulder_pitch servo, and wearing out the teeth of the gears. The Bioloid code was deleted from the microcontroller but is available in the Capstone report for reference.

The next step was to establish a ROS compatible communication with the microcontroller, and that was accomplished by uploading a ROS sketch onto it via Arduino IDE. This ROS sketch provided a way of communication between the servos of the manipulator through the microcontroller and the Ubuntu 14.04 Linux with ROS Indigo that was used for development during this project.

ROS-ArbotiX Control Package - Python

After establishing a ROS-compatible way of communicating with the servos, the ROS-ArbotiX control meta-package, arbotix_python was used to try to move the arm via a GUI. It was noted then, that the servos were behaving in an unpredictable manner, covering ranges that were offset and/or reduced. The config yaml file for the package has been modified, however, something was missing. It was determined that the three different types of servos had different maximum ranges and number of ticks. For example, AX12 and AX 18 servos could only cover 0 to 300 degrees, while MX64 servos could cover 360 degrees and had an activated mutli-turn option. More details on the specification of the Dynamixel actuators used in the design of LWMS can be found on the Trossen Robotics website or the Robotis website.

To correct for varying capabilities of the actuators, the yaml file was edited to include the number of ticks for each joint. Also RoboPlus DynaManager software was used to reset proper IDs and baud rates for each actuator and center them. This diagnostic confirmed that all the actuators were functional and that issues with ranges of motion were due to incorrect yaml configuration and not due to any defects of the servos. The detailed instructions and support on how to use DynaManager may be found on the Trossen Robotics website.

Once the necessary edits were implemented into the arbotix_python config yaml file and the servo IDs and baud rates were reset, the ArbotiX package could be used to move each joint via an

ArbotiX GUI as intended. This package provides basic controllers for Dynamixel actuators, and is implemented in open-source packages for PhantomX Pincher and WidowX Robot Arm discussed earlier.

Custom Controllers Based on Dynamixel Interface Library – C++

The Clam Arm uses a custom-written controller package based on Dynamixel interface library, but it is unclear whether that library is up-to-date, because the Clam Arm repository is not functional and the attempt to configure Clam Arm controllers for LWMS was not successful.

ROS Dynamixel Control and Rosserial Stacks – C++

Other approaches to establish a real-time ROS control over the LWMS were attempted. The book *Learning ROS for Robotics Programming* features a whole section with instructions and tutorials on how to implement ros-dynamixel-control package to control Dynamixel servos via command line by posting to topics, and how to use rosserial stack to control Dynamixel servos via Arduino microcontrollers (Note: ArbotiX-M microcontroller is Arduino compatible, so in theory, rosserial can be applied to LWMS).

However, following these tutorials did not prove successful. The software was not able to identify the servos. A thorough search on online robotics forums and discussions, showed that other people had the same issue, yet no working solution was offered by the robotics community and the authors of the packages at the time. The online discussion suggested that the issue is either with the servos or with the way the packages were written. Most of the discussions were several years old and closed due to inactivity. This approach this approach does not provide a visual for the manipulator so it is not an optimal option for remote control, however, it remains open for further investigation.

ROS Dynamixel Controllers Stack

It was also attempted to use the ROS dynamixel_controllers stack from ROS.org to interface with the servos in real time via service-client model. The detailed tutorials and information about the stack are available on the official ROS website: http://wiki.ros.org/dynamixel_controllers

The package is available from the source on GitHub for further development: https://github.com/arebgun/dynamixel_motor

The first tutorial for the package “Connecting to the Dynamixel Bus” was followed, however connection was not established. The error was: `pan_tilt_port: No motors found.`

The package was configured for baud rate 1 000 000, a unique ID set for each servo using RoboPlus Dynamixel Manager (IDs 1-7). At this point, the IDs and baud rates for each servo were double checked with DynaManager.

Connection was configured to /dev/ttyUSB0 port and double checked using command `ls -la /dev/ttyUSB0*`.

The write rights were set using `sudo chmod a+rw /dev/ttyUSB0` to ensure proper two-way communication via the port.

The issue was very similar to the discussion on answers.ros.org/question/215063/no-motors-found/, where the arbotix_terminal node saw the motors and the arbotix_python package could be used to manipulate servos, but the dynamixel_controllers package could not find motors.

To ensure that the issue is not because the manipulator is connected to USB via microcontroller, the arm was reconnected via USB2AX, resetting port to `/dev/ttyACM0` in the configuration file, however, the node still could not find the motors.

Some of the answers on various discussions of similar issues dated in 2015, suggested that there was a malfunctioning batch of MX64 motors that was on the market at the time. There is a possibility that all the issues stem from such a defected motor in the manipulator, however, the servos were purchased in the beginning of 2016, and the first servo in series is AX12, which should be recognized even if the MX64 servos are defective. Moreover, the arbotix_python package worked as expected with the servos, thus they are unlikely to be defective.

If fixed, the dynamixel_controllers stack provides support for various Dynamixel actuators, including AX12, AX18, and MX64, and can be used to create JointPosition or JointTorque controllers for the LWMS based on available tutorials at ROS.org.

ROS Control Boilerplate – Future Direction

Meta-package ros_control is a generic rewrite of the packages for PR2 robot. It is a good generic package to use as a base for LWMS control. It takes joint state data as input from the servo encoders and an input set point, then it uses a PID controller to control output.

This meta-package provides effort, joint state, position and velocity controllers, as well as a number of interfaces such as Joint Command, (effort, velocity, and position), Joint State, Actuator State, Actuator Command, and force-torque sensor interface.

There is also a transmission interface that can be used for the gear mechanism in LWMS. A transmission tag will have to be implemented into the URDF. For now, a simpler solution was used, inverting the actuator motion at the level of yaml configuration in arbotix_python package.

A boilerplate for ros_control is available, written by David Coleman. It requires a few code blocks specific to LWMS written in C++ and should provide a good generic interface with the controllers.

Full documentation and tutorials are available at http://wiki.ros.org/ros_control .

There is also a tutorial on GitHub on how to create hardware interface for a custom robot: https://github.com/ros-controls/ros_control/wiki/hardware_interface.

This may be a useful resource for someone who takes up this project and has a solid understanding of writing code that implements ROS libraries and appropriate structs.

A tutorial on loading the controllers once they are written and starting them through service calls is described at:

http://wiki.ros.org/ros_control/Tutorials/Loading%20and%20starting%20controllers%20through%20service%20calls

Based on the PR2 robot, the process of writing a controller is described in detail here:

http://library.isr.ist.utl.pt/docs/roswiki/pr2_mechanism(2f)Tutorials(2f)Writing(20)a(20)realtime(20)joint(20)controller.html

This approach seems optimal for establishing a stable and flexible real-time control over LWMS. It was extensively researched but not implemented in this project because the sources provided above are specific to PR2, which bears no similarity in form or function to LWMS. Deeper understanding and experience in robotics code structure is required to use this tutorial as a reference for writing a controller for LWMS.

Once the controllers are written, a controller manager launch file can be used for specifying all configurations, loading and starting multiple controllers at the same time. The detailed instructions on writing a launch file are also provided in the above links.

Universal Robot Description File

Once various options for control of LWMS in real time were explored, the next step was to enable accurate simulation in RViz and MoveIt, which, once interfaced with controllers, will be used as a visualization tool for remote control.

The first step in simulating LWMS with available software tools was to create a URDF file (Universal Robot Description File) that would describe the structure of the robot to the software analyzing and simulating it in a language that this software understands. Official ROS tutorials on URDF were followed at http://wiki.ros.org/urdf/Tutorials and David Coleman's repository on Clam Arm was referenced for creating an actual robot_description package and using proper naming and structural convention.

Features were added to the URDF gradually as the topic was researched deeper, and a greater understanding of the structure and functionality of LWMS led to a more accurate description of the manipulator. The URDF was then parsed upon completion and tested in Rviz, Gazebo, and MoveIt.

For Gazebo, tutorials on the official http://gazebosim.org/tutorials website were followed to spawn the URDF into an empty Gazebo world and make sure that it is displayed correctly and simulated with reasonable physics.

A MoveIt configuration package was created for LWMS with the URDF following the ROS documentation on this topic at:

http://docs.ros.org/hydro/api/moveit_setup_assistant/html/doc/tutorial.html.

This robot_moveit_config package provides simulation and motion planning for the LWMS.

Finally, the URDF can be opened as a simulation in RViz and manipulated virtually via an RViz GUI.

In addition, robot_state_publisher node was written for the LWMS URDF and tested in RViz following the tutorial at http://wiki.ros.org/robot_state_publisher/Tutorials .

All code, scripts and instructions can be found on the lab's GitHub repository.

Summary

Four functional ROS packages were created for the LWMS in the course of this project:

1. robot_description package that contains URDF, XACRO, and DAE files describing the configuration and physical metrics of the manipulator.
2. robot_moveit_config package for simulated motion planning with MoveIt!
3. robot_state_publisher package that uses URDF to publish the state of the robot to tf.
4. robot_arbotix_control package – uses ArbotiX control stack and an edited yaml file specific for the manipulator to provide low-level control of the arm via ArbotiX GUI.

List of Sources

http://www.ros.org/

http://moveit.ros.org

http://gazebosim.org/

http://www.trossenrobotics.com/

Martinez, A., Fernandez, E. *Learning ROS for Robotics Programming*. Packt Publishing, 2016

https://github.com/davetcoleman/clam