UR3e OCE Stabilization System

Ben Anderson

November 2, 2022

Contents

1 Planning (Delete later)

Introduction section will summarize the body of the text,

I need a section detailing the interfaces of the UR3e robot and why I need to use the workaround that I have

I need a section detailing how my code works to optimize the strength of the input signal

I need a section detailing how all of these bits of code interact with each other (will be the hard bit)

Creative decision: I could either lead with the section detailing how all the components interact with each other, or I could lead with sections detailing individual components and close with a section summarizing their overlap. The tradeoff is the summary section will be missing a lot of context, but this could potentially be provided by linking forwards repeatedly. Mayhaps this is the move. I just need to learn how to link to a place further along in a document (section 2).

Okay that looks promising to me. Lets go with that.

2 Introduction

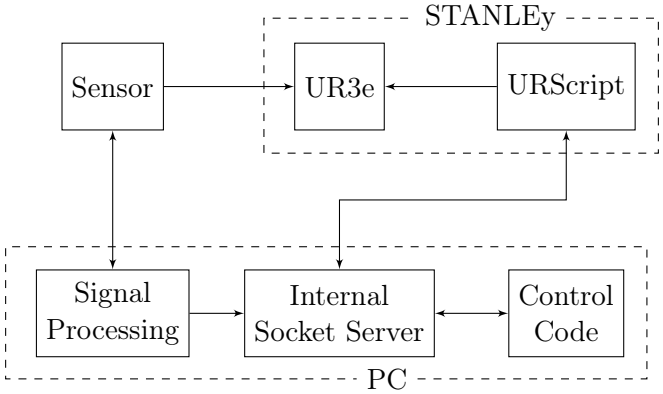
The focus of my work for the better part of the last year has been studying how to use a UR3e Cobot for the purposes of focusing and stabilizing a sensor at its focal length away from a target surface. This has involved a thorough examination of the API and programming tools that accompany the Universal Robots ecosystem, as well as creating systems for synchronizing the robotic control system with signal processing and sensor systems.

In doing so a number of systems are being used in ways for which they were not designed. The consequence of this is that my codebase resembles a kludgy pile of half-fixes and temporary-solutions that, technically, have achieved the objective, at the expense of stability and ease of use. In recent months more efforts have been made to improve the readability of the codebase and allow for easier changes later on, but the work is ongoing and the project cannot wait for me to complete this work.

The hope is with this documentation, others in the OCE project can gain enough of an understanding of how the system works to use it in my absence as well as modify or extend the platform I've built.

Due to the kludgy nature of the platform, it may not be suitable to read this guide linearly. The introduction section should hopefully give you an overview of how the components of this section communicate and interface with each other, and a more thorough description of the workings of each module are in the following chapters. If you are planning to operate the robot I would recommend reviewing the section on operating the robot through the control pendant (subsection 4.1) and the section on running the code (subsection 5.1).

3 Overview of Transudcer Alignment system



The sensor is a generic device that requires focusing and produces an output that is read by the signal processing package within the computer. The Stabilization and Transducer Alignment for Nearby Laser Elastography system (hencforth referred to as STANLEy) consists of a UR3e cobot from Unviersal Robots, and a script written in using the Universal Robots proprietary scripting language.
(Attempts are being made to reverse-engineer some of this languages functions to cut this box out of the control diagram, but the process will be very involved and is being put off in order to meet deadlines).

Physically, this system includes a robotic arm manipulator, a control box, and a teaching pendant. STANLEy interfaces with the sensor through a physical linkage at the end of the robotic manipulator, and with the PC through an ethernet cable.

The remaining control blocks occur within a nearby computer. The signal processing block refers to a program (typically in MATLAB or LabView) which recieves the data from the sensor and processes it for further use. *For the purposes of the control loop that focuses the sensor, the signal processing block must reduce the sensor input, no matter how complex, to a simple magnitude that should be maximized to bring the sensor into alignment.*
An internal socket server is spun up in a simple python program. This is a program that listens on a hard-coded IP-address and port within the computer for input sent from the signal processing block, the control code, and to the robot through the control box. More on the limitations of this system in the section on running the code (subsection 5.1)
The control code is run from a script named +TransducerHoming.py+, and associated libraries. For now, directly editing this file is the main interface with which we can modify how the homing sequence runs. In future I hope to expand the user interface as well as the command-line options to allow more on-the-fly modification of program parameters, but the duration off tests and the speed at which changes are made have made this an inefficient goal for some time.

Overall the system utilizes three discrete devices and at least three programming languages. I (Ben Anderson) am most directly involved with the programming of the control code and URScript. The top-down overview of each components' functions are pretty simple, but debugging/modifying them will require

4 Stanley and the URScript API

4.1 Teach Pendant

4.2 Client interfaces

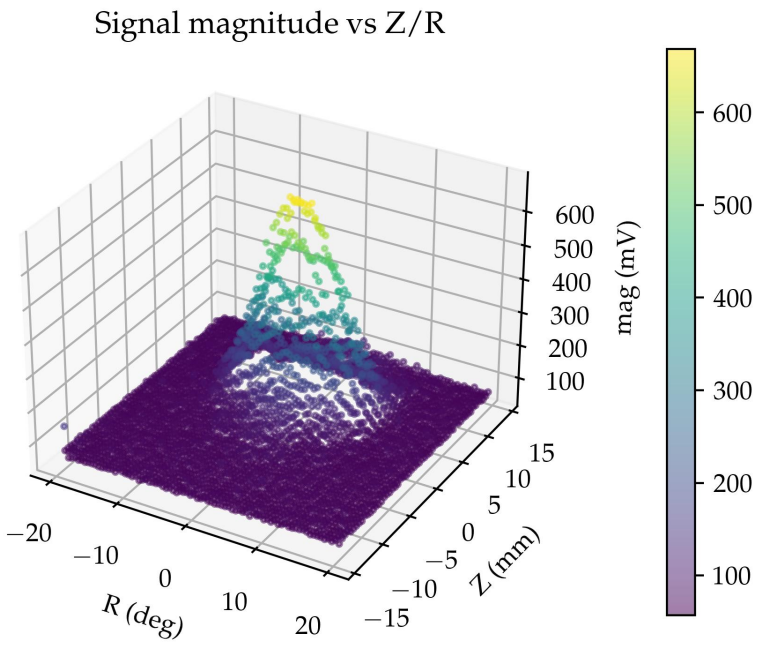
4.3 URScript API

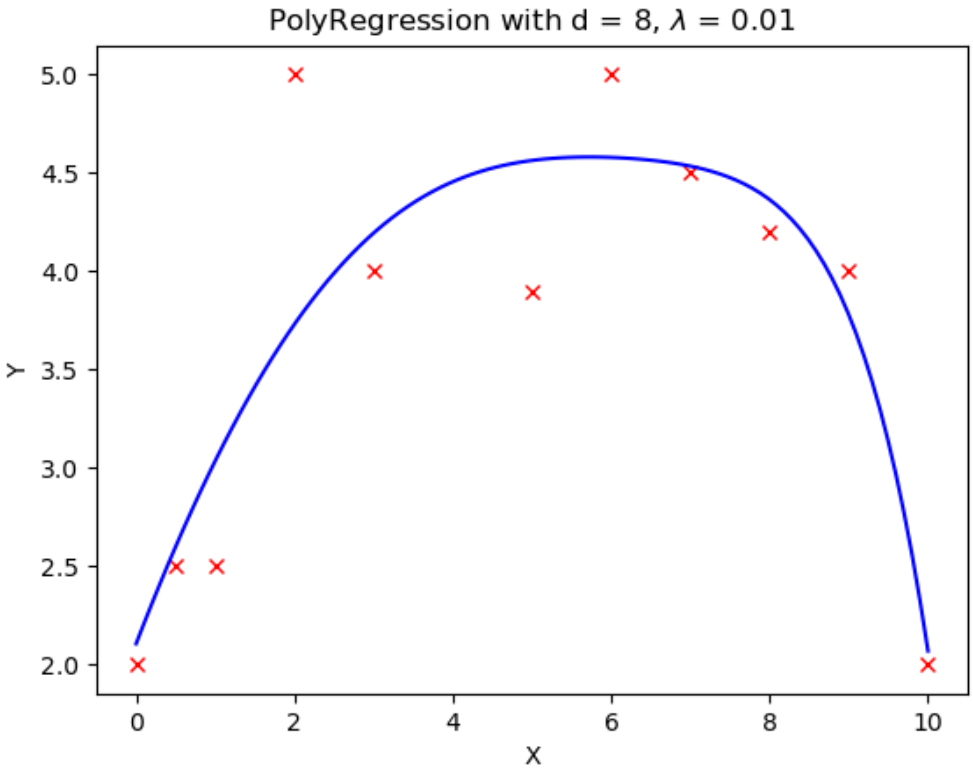
5 Python + the control loop

5.1 Startup guide

5.2 TransducerHoming.py

STANLEy -> Stabilization and Transducer Alignment for Nearby Laser Elastography


Signal magnitude vs Z/R
600
500
400
300
200
100
mag (mV)
−20
−10
0
10
20
R (deg)
15
10
5
0
−5
−10
−15
Z (mm)


PolyRegression with d = 8, λ = 0.01
5.0
4.5
4.0
3.5
3.0
2.5
2.0
Y
0
2
4
6
8
10
X