FrankNarm Development Manual

1. Abstract

The main goal of this project was to construct an arm capable to write inverse kinematics for to serve as a controller testing platform and educational demonstrations. This project originally started with 2 separate 5 degree of freedom (DOF) arms that began to malfunction. These arms were combined to create one 7 DOF arm. For this reason the project has been dubbed the FrankNarm project.

1. Systems Engineering
   1. Deliverables:

Due to the specific nature of this constantly evolving project, the deliverables outlined were generic. They are as follows:

* Develop a platform to test and develop inverse kinematics code one
* Develop a low cost platform to use in evaluating controller features
* Portable to be used for demonstrations

1. Procedure
   1. Hardware development
      1. Initial Setup

As stated the above, two 5 DOF arms were the original operation equipment of this project. One had underpowered and broken servos, the other had a broken servo control board and faulty joints. Once the arms were combined, it was realized the arm had become too long and heavy for the base servo’s to lift itself. Springs were added to provide a counter force to allow the robot arm to lift itself. To fix the issues with the broken servo boards, an Adafruit servo control board paired with a Raspberry-Pi was used to allow for both direct servo control and to allow for the computations necessary for inverse kinematics to be done by the arm’s system itself.

3.2.2 Prototype Development:

Once the arm was functional, all of the support hardware will be packaged into a plastic project box to allow for easy transport. An Ethernet cable will be used to connect the arm for to the box. A USB Hub will be necessary to coonect all of the internal hardware. Instructions on the construction of the box are as follows:

**COLEMAN WRITE INSTRUCTIONS ON BUIILDING THE CASE HERE!!!!!!**

* 1. Software Development

Coleman, I don’t get mace’s notes about this. Change these bullets into sentecnes when you get a chance:

.     Phase 1 -- Configure Raspian for i2c Devices

* sudo nano /etc/modules
* Add to end of file:
  + i2c-bcm2708&nbsp;
  + i2c-dev
* sudo apt-get install python-smbus
* sudo apt-get install i2c-tools
* sudo nano /etc/modprobe.d/raspi-blacklist.conf
* Comment out:
  + blacklist spi-bcm2708
  + blacklist i2c-bcm2708
* Detect connected i2c devices
  + sudo i2cdetect -y 1

If 40 appears under column 0 and row 40, the arm is connected and able to receive input. If not, check the wiring.

* 1. Cartesian Control Development (Inverese kinematics)
     1. Inverse Kinematics Function

An inverse kinematics function applied to a robotic arm would allow the end of the robot to be controlled by Cartesian control. To obtain the inverse kinematics function, they following steps were followed:

1. obtain DH parameters of robotic arm (Introduction to Robotics, John J Craig)
2. using DH parameters, obtain Jacobian as function of joint angles (via Differentiation Method)
3. build inverse kinematics function:
   1. take in current joint angles, and use to build jacobian matrix for current configuration
   2. take in wanted change (delta x, delta y, delta z, delta roll, delta pitch, delta yaw)
   3. use damped least squares to solve deltaS = J \* deltaTheta (<http://math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf>)
   4. solves for delta theta: change in joint angles
   5. add delta thetas to current angles
   6. update joint positions
      1. Inetgration code

**Ben I have no idea how you guys are doing this to write about it. Here is the bullets Mike has given me so far about it.**

To utilize inversekinematics:

ensure InverseKinematics.py and ForwardKinematics.py (available at <https://github.com/NASA-rdt/raptor/tree/sixDOF-Mike>) are in your path

import both

InverseKinematics.goTo(current\_angles,deltas,plot, damping\_coeff, which\_method)

current\_angles is 7 entry vector with angles in radians of each joint, with first entry as lowest and 7th as closest to endeffector

deltas is a 6 entry vector or wanted changes (x, y, z, roll, pitch, yaw) in inches and radians. Entries should be small (< 0.4 inches, or < pi/8 radians)

plot defaults to 0: set = 1 if you want a plot of current arm positions

damping\_coeff is the damping coefficient for the damped least squares method. Default is .03, set if you desire a different one

which\_method designates the IK method. Defaults to 0 (damped least squares): set = 1 for pseudoinverse (legacy)

InverseKinematics.goTo() returns a 7 entry vector with new joint angles in radians, numbered the same way as current\_angles, such that looping:

>>>>theta = InverseKinematics.goTo(theta,delta)

Appendixes:

Appendix A: Risk Chart

Appendix B: Cost Chart

Appendix C: Inverse Kinematics Function Diagram

Appendix D: Hardware Layout

Appendix C

