CoSMo 2012 Robotics Tutorial : Robot Control

Date: August 16th, 2012 **Instructor:** Brenna Argall

Design an "open-ended" *trapezoidal controller* to estimate the position changes of a wheeled mobile robot, with the following inputs, outputs and constant parameter values.

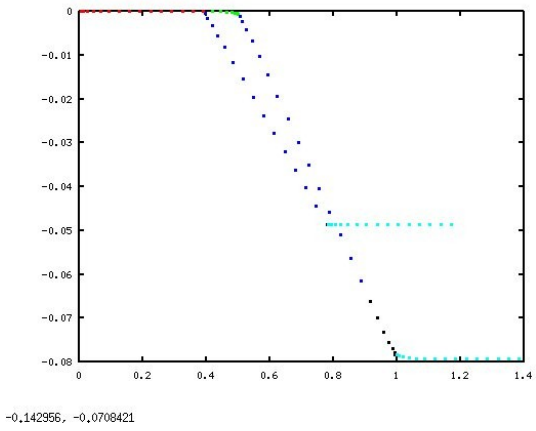
$$\text{Input}:[\nu,\omega,t] \quad \text{Output}:[dx,dy,d\theta]$$

$$\dot{\nu}=1\frac{m}{s} \quad \dot{\omega}=1\frac{rad}{s} \quad dt=\frac{1}{15}s$$

Test your controller with the following sequence of commands, starting each time from zero translational and rotational speeds. Plot your output.

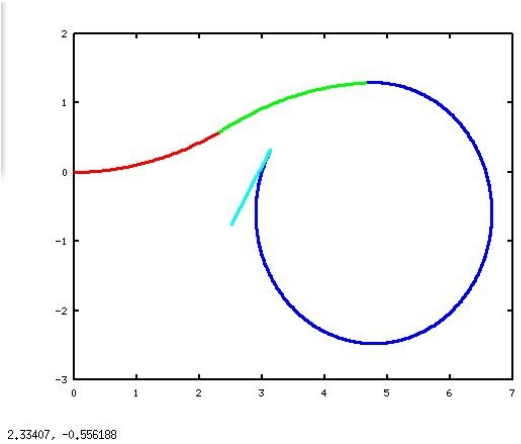
$$[\nu=0.5,\ \omega=0,\ t=1]$$
$$[\nu=0,\ \omega=\frac{-1}{2\pi},\ t=1]$$
$$[\nu=0.5,\ \omega=0,\ t=1]$$
$$[\nu=0,\ \omega=\frac{1}{2\pi},\ t=1]$$
$$[\nu=0.5,\ \omega=0,\ t=1]$$

Repeat the test sequence, now using the ending speed of the previous command as the starting speed of the next command. Plot your output.



Test another sequence of commands:

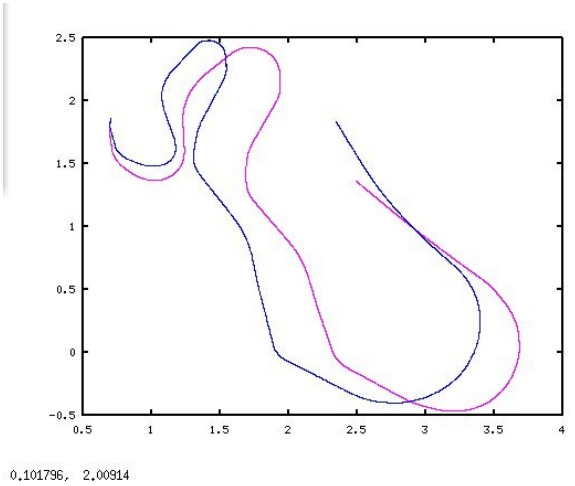
$$[\nu=0.5,\ \omega=0.1,\ t=5]$$
$$[\nu=0.5,\ \omega=-0.1,\ t=5]$$
$$[\nu=0.3,\ \omega=\frac{-1}{2\pi},\ t=33]$$
$$[\nu=-0.5,\ \omega=0,\ t=3]$$

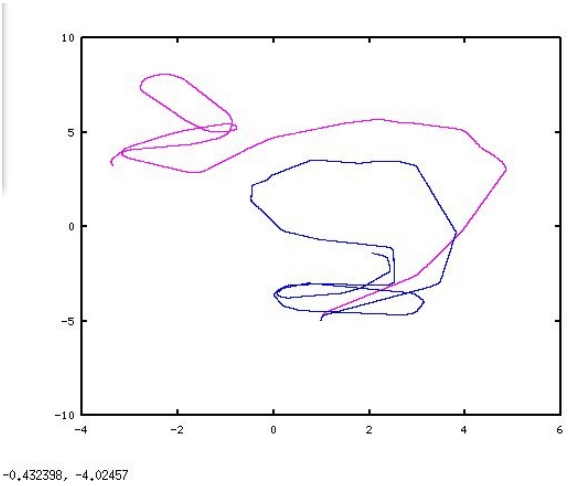


Now add some actuation noise to your controller. Play around with different noise amounts, and repeat the first command sequence above, multiple times. Record the (x,y) positions (for use in the robot learning section).

Now try your controller with a real robot dataset. The file odometry_0.dat contains data of the form: $[t \ \nu \ \omega]$ (Note that the time is absolute, not relative.) This is essentially a "dead-reckoning" model. Compare this to the ground truth data, recorded from a Vicon system, located in the file ground_truth_0.dat: $[t \ x \ y \ \theta]$, by plotting both results. How do they compare – how accurate is this dead reckoning model for this robot platform? What might be some causes for these discrepancies?

Also test the more challenging dataset (odometry_1.dat, ground_truth_1.dat). How does your dead reckoning model compare on the two datasets? What might be a reason for the difference in performance?





CoSMo 2012 Robotics Tutorial : Robot Learning

Date: August 16th, 2012 **Instructor:** Brenna Argall

Locally Weighted Linear Regression (LWR) is a straightforward "lazy" learning approach, that keeps around all of the training data. Minimizing the least squared error is be found by solving the following equation:

$$\hat{y}_q = x_q (Z'Z)^{-1} Z'v$$

x_q: query point (within the input space)
W: diagonal matrix, where $w_{ii} = \sqrt{K(d_i)}$
$K()$: kernel function $K(d_i) = e^{-|x_i - x_q|}, x_i \in X$
$Z = WX$
$v = Wy$

Implement a LWR algorithm, and test it on your (x,y) data recorded above (to learn the mapping $x \rightarrow y$). Divide your data into a training set and a testing set (e.g. 50% in each), and make sure to randomize your training data. Test your algorithm on the testing set, and plot the results. Also plot your training set; the test set predictions should roughly follow the form of the points in the training set. Hint: Threshold your kernel function, so that only training data points with high kernel activations are included within the equations above.

In the robot datasets of the previous section, errors in orientation are present. The dataset lrn_0.dat has computed errors in heading from a comparison to the ground truth dataset, and is of the form $[\omega \ e_\theta]$. Use your LWR algorithm to learn a mapping from $\omega \rightarrow e_\theta$, and use this to correct the predictions of your controller. Plot the results, using the odometry_1.dat dataset. Hint: It can help to normalize your inputs and outputs to within [0,1].

A lot of work goes into building a "good" learning dataset. The dataset lrn_1.dat is of the form $[d\theta \ e_\theta]$, which at first glance we would expect to represent a similar relationship to that captured in the lrn_0 dataset. Use your LWR algorithm to learn a mapping from $d\theta \rightarrow e_\theta$, and use this to correct the predictions of your controller. Plot the results. How do they compare to those of dataset lrn_0?