ALBiR unit 5: visual tracking and feedback dynamics

Learning objectives:

There are many important sensorimotor systems that facilitate the smooth execution of locomotion. These almost reflexive control can be characterised using system identification approach from control engineering. Here, we characterize a simple visual tracking system as an example to demonstrate how we describe the sensorimotor dynamics. The same procedure is often used to understand animal's sensorimotor control or verify any controller designed for a robotic system.

In-class activities:

1. Lecture in modelling sensorimotor control in animals

This introduces two engineering approaches to study sensorimotor control in animal locomotion: 1) PID framework; 2) system identification. Instead of deriving the equations of motion based on the first principles as in the case of passive dynamic walker, we can attempt to model the input-output correlations in a sensorimotor system via some tools. We will explore how these models help us understand the dynamics of a biological system.

2. Set up the PixyCam system [study-group]

Last week you have assembled the pixyBot and worked on the motor tuning. This week we will explore the one onboard sensor: the pixyCam. In order to reduce the computational load and to allow you to focus on the navigation control, we use a colour-based machine vision module for the robot. In addition, we implemented pan control for the camera module, which enables a variety of active vision implementations. You will set up the system and test it during the study group and there will be a group homework for characterising the camera control system.

Homework 1: Tuning a visual tracking system [group coursework]

Setup the visual tracking test as outline below. Each group must submit a 2-page concise report on BlackBoard including the following:

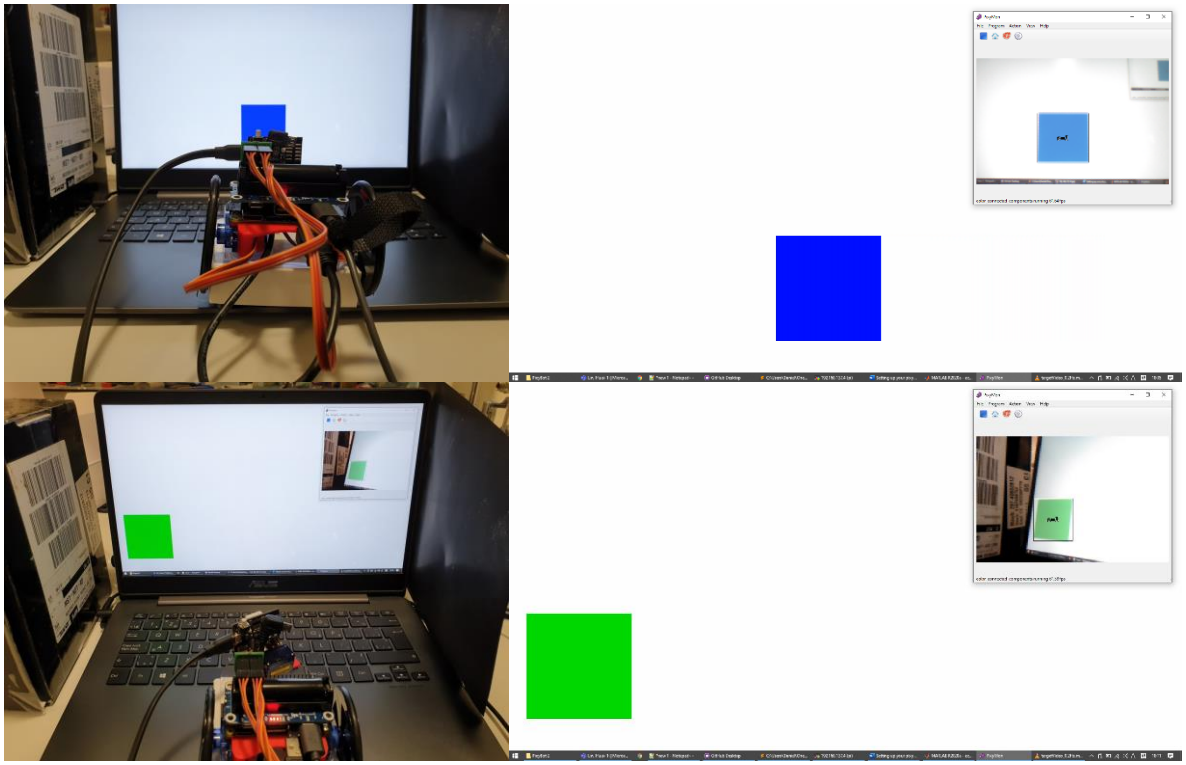
- Explaining the experimental setup and how you extract the phase and amplitude of the visual tracking system.
- Use a Bode plot to describe the system frequency response.
- Relax the tracking criteria and aim for smoothness. This can be done by tweaking the servo control code given. Produce another Bode plot to compare with the standard one.
- Discuss what you can conclude from the result and what are key factors in this feedback control loop.

Experimental setup

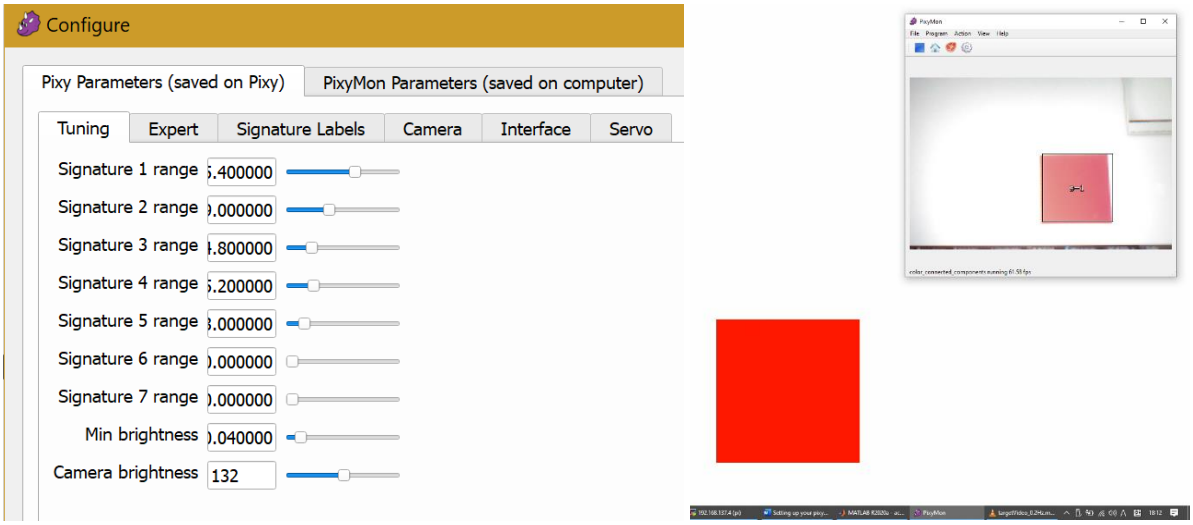
Open any target videos we provide (e.g. targetVideo_0.1Hz.mp4) and you will see 3 different colour squares. Now you want to 'teach' the pixyCam to recognize the red as signature 1, blue as signature 2 and green as signature 3. You may want to pause at a video frame where you a have clear target. Follow the "Setting up your pixyBot" document to find out how to do this.

You must properly shut down the pixyBot before plugging/unplugging the pixyCam. Use the PixyMon App on your PC to configure the pixyCam and make sure to properly shut down the app before unplugging pixyCam from your computer. Otherwise, the settings might not get saved.

The pixyCam should be positioned approximately 15 cm away from your screen, depending on your screen size (we used a 14-inch laptop screen). The moving squares should be roughly centred vertically on the PixyMon monitor. The entire screen does not need to be visible in the monitor at any given time.



Position the pixyRacer in front of your screen so it has to rotate ~30 degrees in either direction to see the edges of the screen



To improve the robustness of the detection of the coloured targets, adjust the range for each signature until your targets are recognised robustly without detecting false positives. Change the camera brightness and other settings to fine tune further.

Basic visual tracking tuning curves

Run exercise_2.py, making sure to edit the input argument to the call to tuningPad.measure with the frequency of the video you are playing from targetVideos.zip. The python script records the time of each datapoint, the angular error between the camera heading and the target angle, and pixyCam servo commands for 10 cycles. This record is then saved as calibrationCurveXHz_Y.csv, where the three rows correspond to the three variables. You can easily load this into MATLAB for visualisation. Fill the table below with a suitable number of tested frequencies across the provided range and compute the servo and visual error signal amplitude and phase difference in MATLAB. If you want to test different frequencies to provided videos, you can generate them using the script makeTargetVideo.m

Input frequency	Servo tracking amplitude RMS	Visual error amplitude RMS	Notes
0.1			
0.2			
...			
3			

…. add rows as you see fit. This is just an example. Watch the example video on blackboard to see what successful target tracking looks like.

Bode plot

To visualize the system response in the frequency domain, it is convenient to plot the gain (system response compared to the input signal) and the phase (system response timing relative to the input signal) as functions of input signal frequencies.

Hint: The gain is the amplitude (of servo position vs time) that you recorded normalised by the amplitude if the camera was always tracking perfectly, and the phase is the phase lag between these 2 signals. You will need to calculate the perfect tracking angles. To do this, you will need to know that the field of view of the camera is 50° and the camera has 316 pixels in the horizontal direction.

Smooth tracking adjustment

To use the pan angle as a proxy for steering control later, it is advantageous to enforce certain target tracking dynamics in the system. One quality is smoothness which can be enforced by only insisting

the target in the field of view under certain criteria. How would you implement a smooth tracking controller? Repeat the same characterization you did for the standard tracker with your modified one.

Homework 2: Lane following behaviour [group coursework]

Test and make sure **at least** two robots within the team can perform the basic lane following behaviour reliably. There is nothing to submit for this homework but you should get this ready for next week. We have provided a straight-line paper track with three coloured lines and the starter codes exercise_3.py and laneFollower.py. You will need to modify the laneFollower.follow function in the commented blocks. Try level 1 first then try level 2 once level 1 performance is robust. There are several provided helper functions in laneFollower.py, make sure you read the comments to understand how to use them.

Motor tuning, pixyCam tuning and lane following logic may feel like three distinct parts of the pixyBot, but the robustness of each and every one of the systems is crucial in achieving successful lane following, and eventually a successful autonomous robot race.