

Lab-1

PID control of a robot via position feedback

Name: _____ Date: _____

1. Objectives:

- Experiment with a PID controller using Simulink
- Implement a controller in a practical complex system.
- Gain intuition about how each of the PID control parameters affects the system performance.

2. Background:

In this lab, you will implement a controller to control a robot via position feedback. Specifically, the controller, which runs on a server, reads the position of the robot as the feedback signal to the PID control, the PID control will determine the motor to drive the robot to the target location.

Prior to the lab you will need to complete section 5 (prelab assignment) using the Simulink model attached on Canvas. These questions are all qualitative, however they will be important to complete prior to the lab in order to get the best results from the “real world” PID testing. Sections 3-4 will be setup for you in lab (you will not have to do this) however it is worth looking over to better understand our experiment.

The control flowchart is shown in Fig. 1. By changing the PID control parameters used in the experiment, you are expected to understand how PID works and gain intuition about how these control parameters affect the performance of the system.

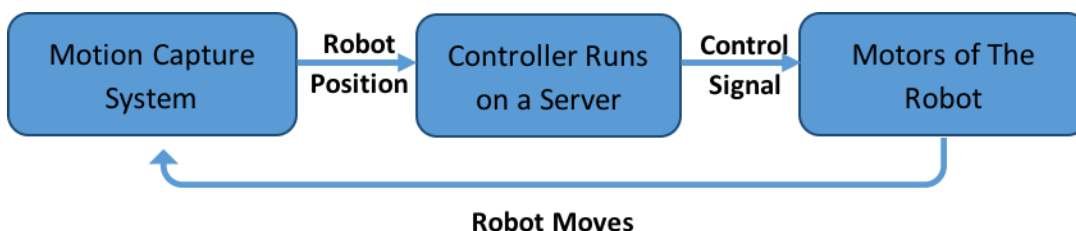


Fig. 1. The control flow

Two demos of this lab are provided as below for your information. The demos can also be found in the local disk drive T: .

Demo#1 <https://youtu.be/B-uWmT8-oTg>

Demo#2 https://youtu.be/nLZd2_aDr4g

3. System Setup and its working principle:

a. Major equipment

- *Server*

This desktop collects the position data from the motion capture system, processes the data by the controller programmed in Matlab, and sends out the motor control commands to the robot via the communication module.

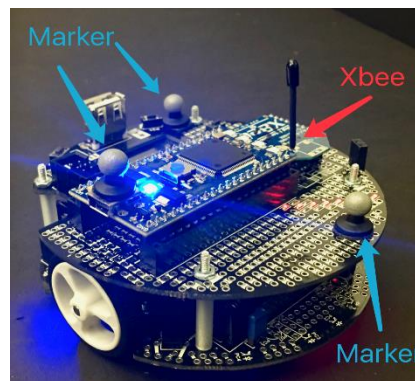
- *Motion Capture System*

This *OptiTrack* motion capture system has four high resolution cameras for capturing the position of the robot. This system also sends the position data to the server.



- *M3PI Robot*

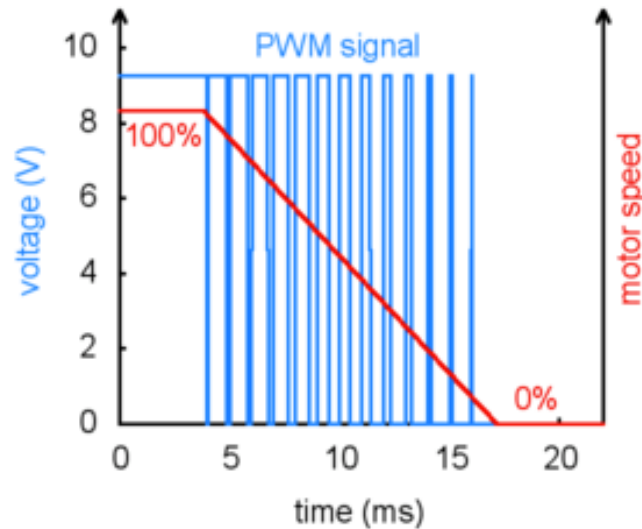
It is a two-wheel robot. A Xbee module on it is to receive commands from the server. Also, tracking markers for motion capture are installed on the body of the robot.



b. The speed control of the motor

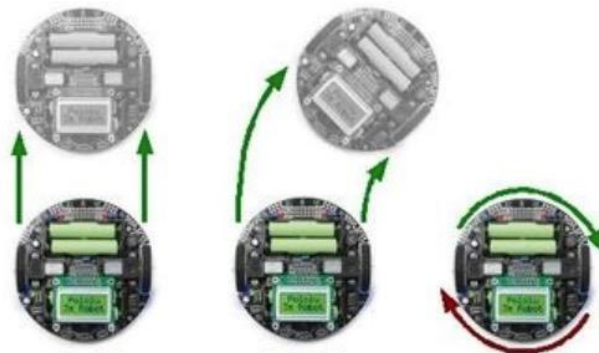
The motor driver switches between the “forward” and “brake” states rapidly to control its speed. Since the motor voltage is a series of pulses of varying width, this method of speed control is called pulse-width modulation (PWM). An example series of PWM pulses is shown in the graph below: as the size of the pulses decreases from 100%

duty cycle down to 0%, the motor speed decreases from full speed down to a stop. In this lab, the control signal to the motor is the PWM value.



c. Turning with a different drive

The robot has an independent motor and wheel on each side, which enables a method of locomotion called differential drive. It is also known as a “tank drive” since this is how a tank drives. Turning with a differential drive is accomplished by running the two motors at different speeds. The difference in speeds determines how sharp the turn will be, and spinning in place can be accomplished by running one motor forward and one backward. The following figure demonstrates the effect of various motor settings.



In this lab, **we only let the robot go straight either forward or backward in one axis**. Hence, the control commands sent to the two motors are always identical.

d. The Motive Software

Motive is a software platform which works with our four camera motion capture system. It analyzes the real time images captured by the cameras, generates the location of the markers on the robot, and sends the data to the server.

e. The PID Controller in Matlab

The PID controller of the control system in this lab is written as shown below:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt}$$

In Matlab, it is expressed as:

```
% error entering the PID controller
error(idx+1) = target - feedback(idx);
% error of proportional term
prop(idx+1) = error(idx+1);
% derivative of the error
der(idx+1) = (error(idx+1) - error(idx))/dt;
% integration of the error
int(idx+1) = (error(idx+1) + error(idx))*dt/2;
% the sum of the integration of the error
I(idx+1) = sum(int);
% the three PID terms
PID(idx+1) = Kp*prop(idx) + Ki*I(idx+1) + Kd*Der(idx);
```


Please note that:

The **feedback(idx)** is the location of the robot from the motion capture system,

The **PID(idx+1)** is the PWM control signal sent to the motor of the robot.

This controller currently only controls one axis for simplicity.

4. Procedure and Tasks:

- a. Turn on the computer and login. Turn on the power for the motion capture system. Make sure all the instruments are in their proper positions and the robot and camera field platform area is cleared for safety.
- b. Open motive software and perform the following steps:
 - Create new project
 - Perform calibration process with the following two tools, the general steps are described as below.
 - From the Camera Preview pane, clear existing masks by clicking .
 - Open the *Calibration pane*, and use *Mask Visible* to mask extraneous reflections that cannot be removed from capture volume.
 - Collect calibration samples: Wandering.
 - Calculate and review the calibration results.
 - Set the Ground Plane.

The link below has the instruction video and detailed setup steps.


(<http://wiki.optitrack.com/index.php?title=Calibration>)



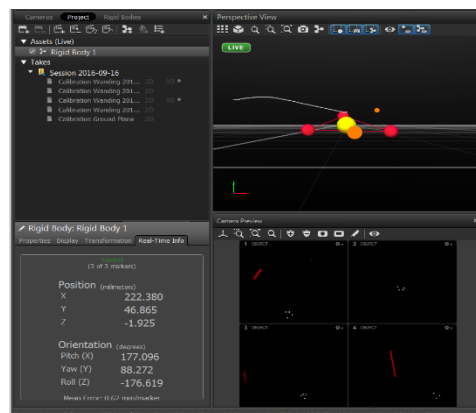
Winding Tool



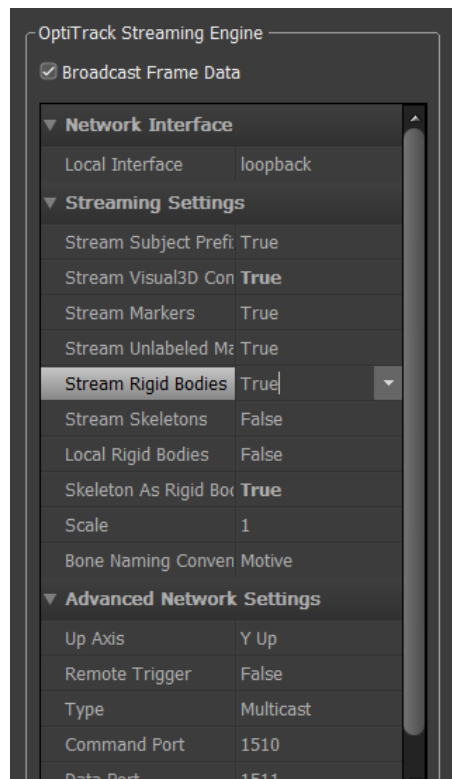
Ground Plane

- c. Create the rigid body (the 3 makers) of the robot, the general steps are as below.
- Select the associated rigid body markers in the 3D view pane
 - Right-click on the perspective view to access the context menu.
 - Go to Create from Selected Markers under Rigid Body. This can also be done by using the Create Rigid Body  button in Project Pane while the markers are selected.
 - When a rigid body is created, the markers will be colored and interconnected to each other. A newly created rigid body will be displayed under Assets in the Project pane.
 - The link below has the detailed instruction on rigid body.

(http://wiki.optitrack.com/index.php?title=Rigid_Body_Tracking)



- d. Start streaming the data of the rigid body to the server.



- e. Make sure the comm. transceiver is connected onto the server (connect if not already done so).
- f. Run the experiment:
Power on the robot if not on (make sure your robot is fully charged before start, and please pay special attention and check with your TF on charging the robot)

Start MATLAB, open – T:\es158\es158-mcs-v3\run.m

The PID parameters can be set in *setPID.m*

The parameters including max speed limitation, sample frequency, etc. can be modified in the code.

The program will run for 6 seconds after it starts. **Do not** stop the program or exit Matlab while the code is still running.

Now you are ready to play with the robot controller:

Open up *setPID.m*, change K_p , K_i , and/or K_d as below, and record the results and the observed robot behavior. The target destination is defined as the variable “desired” in *setPID.m*, change it, and record the results and compare the observed robot behavior again.

The dynamical model for our vehicle control problem is also simplified to be a second order dynamical system with position and velocity as the states:

The goal is to let the position $s(t)$ follow a reference signal $s_{\text{ref}}(t)$ by designing a PID controller

where $e(t) = s(t) - s_{\text{ref}}(t)$. Note that when $s_{\text{ref}}(t)$ is a constant s^* for all time $t \geq 0$, it means the objective is to let position reach a target value s^* .

- Page 7 of 13

- c. Based on the observation you had from part 5.a, how will you pick your best combination of these three parameters? And what parameters you will pick? Provide a few justifications how you make your choice. (Note: This is an open-ended question.)

- d. Now experiment with using the auto tuning PID function in Simulink (button right next to the PID constants box) and see what results you get from there. Qualitatively describe the differences that you had between the controllers you obtained and the auto tuning results by PID tuning. You can also play with PID tuning by changing the response time and observe the change of corresponding P, I, D values and the transient dynamics of the trajectory.
- e. An important factor of real-world control problems is external sources of noise that can affect our system. In this case we are simulating external noise by just adding random Gaussian "noise" into the measurement. Having an initially reasonable controller that mostly tracks our desired output, let us experiment with adding noise with increasing variance. What happens to our output as we increase our variance (on the sensor noise block)?

- f. Similarly to noise another non-ideal constraint of real-world feedback problems is sampling accuracy and speed. In the lab we use a motion capture (camera) system to take the measurements, use a server to compute the controller value, and send the control command to the vehicle. This feedback loop comes with a latency that prevents us from creating a truly ideal controller. Let's now try and experiment with the effects of this sampling and latency issue. Change the sampling rate of the analog to digital conversion block in our feedback loop and qualitatively describe the affects you see on the PID output and our tracking performance as you increase the latency.
- g. As you might recall from Pset 1 you were asked with graphing the difference between the output signal and a sinusoidal reference signal we wanted to track. Let us now recreate that here and try to better understand the results from those graphs. Let us first switch one of our step inputs to a sine wave (you can do this by deleting one of the wire from our step input signal and instead connecting the sine wave block to the sum block). Now test setting the frequency of that reference desired position higher and higher. Add a channel to our output scope that plots the difference between the desired output and actual output. What happens as you increase the frequency of the desired tracking sinusoidal signal? And what happens with the difference "error" signal?

6. Results from in-lab experiments: (attach graphs if any)
Here we only test reaching a target constant position.

Target Destination =

Kp	Ki	Kd	Time-to-target	Running behavior note
3.5	0.1	0.2		
2.5	0.1	0.2		
4.5	0.1	0.2		
3.5	0.2	0.2		
3.5	0.3	0.2		
3.5	0.3	0.4		
3.5	0.3	0.6		

Target Destination =

Kp	Ki	Kd	Time-to-target	Running behavior note
3.5	0.1	0.2		
2.5	0.1	0.2		
4.5	0.1	0.2		
3.5	0.2	0.2		
3.5	0.3	0.2		
3.5	0.3	0.4		
3.5	0.3	0.6		

Discuss what is each parameter's physical meaning? Their effects on the overall performance?

K_p –

K_i –

K_d –

Acknowledgement: The following external people were referenced and/or helped in preparation of this experiment: Jie You.