Lab 8

Closed Loop Control II

Author:

Jonas Lussi, Tarik Rifai, Franziska Ullrich, Naveen Shamsudhin, and Prof. Bradley J. Nelson

Institute of Robotics and Intelligent Systems

Date: 2019

Version: 2.0

Summary: The purpose of this week's lab is to implement position control using a closed loop controller. We will implement a PID controller and study the effect of proportional and integral control actions on a system. To close the control loop you will use position feedback from your previously written vision-based tracker. Finally, we will use your controller to move a microrobot to a predefined point in space. This week, you will learn how to:

- program your own PID controller
- use a vision-based tracker for feedback
- to do motion control of a magnetic microrobot

8.1 Background

This lab builds on the previous lab. We will use the same setup and include the microrobot immersed in silicone oil. This lab will focus on implementing and testing a PID controller. The feedback comes from the camera and will be used to accurately follow trajectories with the microrobot.

The general setup is depicted in Figure 8.1. A spherical permanent magnetic is glued onto the moving 2D stage. Another small magnetic object (for these purposes we call it "microrobot") moves in a container that is filled with silicone oil. The silicone oil has a higher viscosity than water and has damping characteristics. Thus, it stabilizes the motion of the microrobot. When the stage is actuated and the magnetic sphere is moved on a path, the microrobot will follow the sphere and can thus be manipulated. The motion of the microrobot is observed by a camera and tracked using computer vision. The tracked position acts as a feedback for the closed loop control.

8.2 Closed Loop System

In the previous week you have written the code to move the stage. This week we will use position feedback to run the 2D stage in closed loop. The output of the controller nr_{steps} is the number of steps that you feed into the stepper motor. The camera measures the position pos_{real} of the microrobot and gives $pos_{measured}$ as the feedback. Figure 8.2 shows the block diagram for the closed loop system.

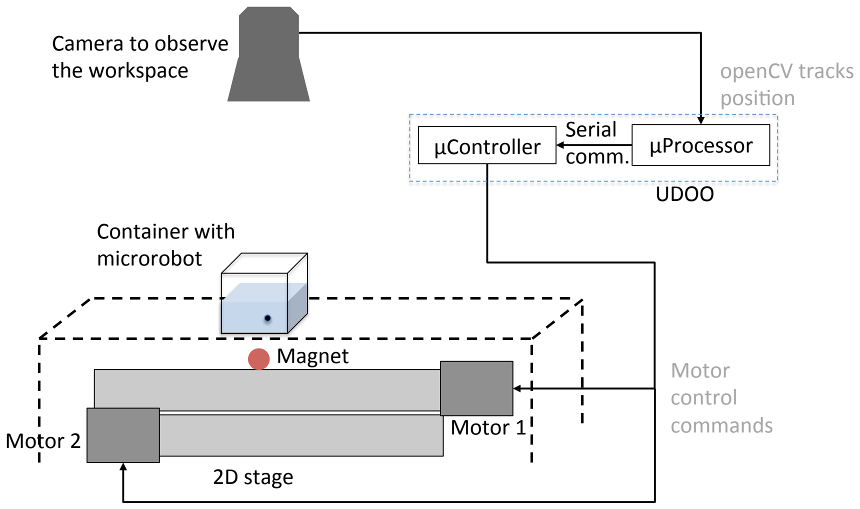

Figure 8.1: Setup for microrobotic application using PID control and visual feedback.

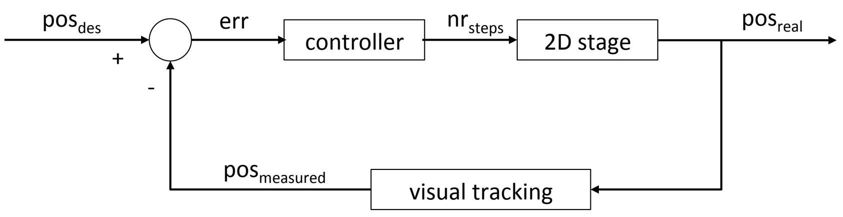

Figure 8.2: Closed loop system with position feedback from the camera.

8.3 PID Controller

The transfer function for an analog PID controller is:

$$T(s) = K_P + \frac{1}{s}K_I + sK_D$$

where K_P, K_I and K_D are the proportional, integral and derivative gains respectively.

Figure 8.3 shows the diagram for a PID controller with anti-windup. In this lab you are not required to implement the anti-windup loop, but you can add it if you want to. We have found good control parameters without the anti-windup.

8.3.1 Discrete Implementation

At any time t_n, the output of the controller is given by:

$$v(t_n) = P(t_n) + I(t_n) + D(t_n)$$

where

$$\begin{aligned} P(t_n) &= K_P \cdot e_n \\ I(t_n) &= I(t_{n-1}) + K_I \cdot \frac{e_n + e_{n-1}}{2} \cdot T_s \end{aligned}$$

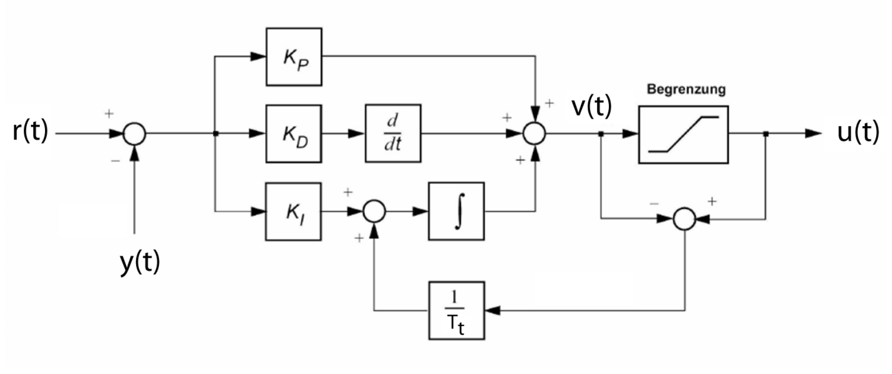

Figure 8.3: PID controller with windup protection.

$$D(t_n) \quad = \quad K_D \cdot \frac{e_n - e_{n-1}}{T_s}$$

Note: there are other possibilities to implement the PID controller. For example, a common choice for $I(t_n)$ is also

$$I(t_n) = I(t_{n-1}) + K_I \cdot e_n \cdot T_s \quad \text{or} \quad I(t_n) = I(t_{n-1}) + K_I \cdot e_{n-1} \cdot T_s.$$

The algorithm for PID control thus becomes:

Initialization

1. Set the values of K_P, K_I and K_D.
2. Set the initial values of I_{n-1}, the sample time T_s and e_{n-1}.

Controller algorithm

1. Input the error e_n.
2. Calculate $P(t_n) + I(t_n) + D(t_n)$ using the above equations.
3. Calculate the temporary output $v(t_n)$ using the above equation.
4. Check for saturation and calculate the controller output $u(t_n) = sat(v(t_n))$.
5. Output $u(t_n)$ (used as input to your motor)

Update controller variables

1. Update the value of the error by setting e_{n-1} equal to e_n.
2. Update the value of the position.
3. Update the value of the integral by setting I_{n-1} equal to I_n
4. Wait for the sampling interval to elapse.
5. Repeat the loop.

8.4 Step Response

Depending on the damping in a system and the controller gain that is used, the step response could be oscillatory (under-damped), critically damped or over-damped.

Figure 8.4 shows the step response for different gain values of $K_p > K_{cr}$, $K_p = K_{cr}$ and $K_p < K_{cr}$.

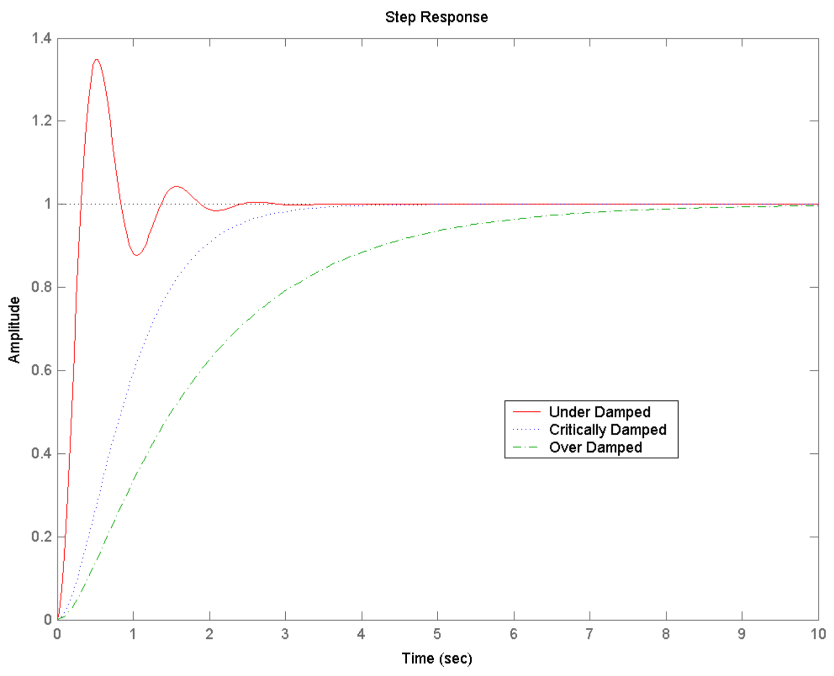

Figure 8.4: response for under damped, critically damped and over damped cases.

8.5 Prelab Procedure

Note: Prelab assignments must be done before reporting to the lab and must be turned in to the lab instructor at the beginning of your lab session. Additionally, you also have to upload your solution as a single PDF-file to moodle. Make sure to upload the file before your lab session starts, late submissions will not be corrected. The prelab needs to be handed in as a group. All the prelab tasks are marked with **PreLab Qx**.

We will write our own PID controller as part of the prelab. You should write your code in the files `closedLoopControl.c` and `closedLoopControl.h` from the previous lab.

1. Write a PID position controller function (called `PID`) that uses position feedback from the color tracker. See the specifications for the function in section 8.3. To get the timestamp of each iteration use the function `gettimeofday()` (use it directly to implement your own `tic()` and `toc()` functions). Make sure to include <sys/time.h>. The signature of the `PID` function is given below (and also in the file `closedLoopControl.h`). **(PreLab Q1)**

```
/*
Function PID uses a color tracker to control the movement of the stages. It can only move one stage
at a time.
Inputs:
int fd - serial port index
int targetPosition - coordinate of the target (in pixels)
CvCapture* capture - camera index
Outputs:
/
Return:
-1 - if there was an error
0 - otherwise

Notes:
- Target in pixels or mm?
*/

int PID (int fd, int targetPositionX, int targetPositionY, CvCapture* capture);
```

Make sure to hand in all the code (*closedLoopControl.c* and *closedLoopControl.h* files) you wrote to solve prelab question Q1 (print out and upload a single PDF-file).

8.6 Lab Procedure

Note: Questions marked with **PostLab Qx** have to be answered as part of **PostLab 08**.

You will need to use the files `closedLoopControl.c` and `closedLoopControl.h` from lab 07. Also, make sure that the camera is calibrated. You can redo the calibration or use the same value. The file `lab08.c` can be downloaded from moodle and should be used to select and execute a specific postlab task.

You will also need to use your Arduino program `lab07.ino` in order for the microcontroller to work properly.

8.6.1 PID Controller

1. In a first step, set the I and D values of your PID-controller to zero. Then, run the function `PID()` with a desired goal position of x = 5mm (no movement in y). The stop condition for the loop is when the desired position is reached within a given error tolerance. NOTE: the error tolerance must be given in pixel (e.g. 5 Pixels). **(PostLab Q1)**
2. Tune the K_P value of your P controller to find a "good" step response by plotting the position versus the timestamp. Plot out the step response (timestamp versus position). The easiest way to do that, is to write into a .txt file directly in the `PID()` function (similarly to `moveMotorRectangular()`). **(PostLab Q2)**

3. Now we will also use the I and D part of the PID controller. Tune the parameters, such that you once get a critically damped and an overdamped response. Plot the two step responses (versus timestamp) in one figure. Also try to get the step response for the underdamped system. You might not achieve to get an underdamped system. **(PostLab Q3)**

4. Rewrite your functions, such that it controls the x- and y-position simultaneously. **(Postlab Q4 - Bonus)**

8.6.2 Closed Loop Control

Now we will move the 2D stage using the PID controller.

1. Write code that uses the PID controller to move the spherical magnet along a square trajectory. Again, do this for a square trajectory with side length 5 mm and plot the resulting tracked positions of the magnet. **(Postlab Q5)**

2. What differences do you find when comparing the open loop control to closed loop control Which is better and why? Please explain. **(Postlab Q6)**

8.6.3 Microrobotic Application

1. Place the container with the microrobot on top of the moving magnet. Be careful not to spill any silicone oil as this might damage equipment.

2. Move the magnet on a square trajectory with side length 5 mm in open loop and in closed loop configurations. The microrobot will follow the large magnet on its path. Use the color tracker to track the microrobot. Plot the trajectory. **(Postlab Q7)**

3. Explain how tracking the spherical magnet differs with tracking the microrobot considering the precision of motion. Why do we place the microrobot in silicone oil with a viscosity higher than water? Please explain. **(Postlab Q8)**

8.7 Postlab and lab report

- Show your running system to the assistant.
- Upload a single PDF-file with your solution to moodle. The file should contain your answers (including plots, and code) to the postlab questions Q1-Q8 (make sure to include the complete files `lab08.c`, `closedLoopControl.c` and `closedLoopControl.h` and all plots from Q2, Q3, Q5 and Q7). Please upload your solution in time (before next lab session), late submissions will not be corrected.
- Print the PDF-file and hand it in at the beginning of the next lab session.
- Come prepared with the prelab procedure for the next lab.