

# 1 Motion Control Loop

This is a description of the calculations used to control the motion of a differential drive robot using a PID control loop. The tutorial [1] served as a starting point, along with the PID control discussion at Wikipedia [2].

## 1.1 Basic Kinematics

Let  $c_{L_n}, c_{R_n}$  be the measured encoder counts for the left and right encoders, respectively, at samples  $n$ , and  $D$  be the distance between wheels from left to right. Then the incremental distance the wheels travel can be calculated from the encoder counters as

$$d_{L_n} = (c_{L_n} - c_{L_{n-1}}) \frac{C_w}{c_{rev}} \quad (1)$$

$$d_{R_n} = (c_{R_n} - c_{R_{n-1}}) \frac{C_w}{c_{rev}} \quad (2)$$

where  $C_w$  is the wheel circumference and  $c_{rev}$  is the number of encoder counts per revolution.

Time is sampled with slight irregularity, but this can be accommodated by using the high-resolution clock in the Linux kernel to record sample times  $t_{i_n}$  to near microsecond accuracy on each encoder, then use successive differences to obtain  $\delta_{i_n}$ . Actual (perceived) velocity is then calculated by

$$v_{L_n} = d_{L_n} / \delta_{L_n}, \quad (3)$$

$$v_{R_n} = d_{R_n} / \delta_{R_n}. \quad (4)$$

in distance units per second. The location and robot pose  $(x, y, \theta)$  are estimated by

$$x_n = \frac{\delta_n}{2} (v_{R_n} + v_{L_n}) \cos(\theta_n) + x_{n-1} \quad (5)$$

$$y_n = \frac{\delta_n}{2} (v_{R_n} + v_{L_n}) \sin(\theta_n) + y_{n-1} \quad (6)$$

$$\theta_n = \frac{\delta_n}{D} (v_{R_n} - v_{L_n}) + \theta_{n-1} \quad (7)$$

where the heading is measured positive counter-clockwise relative to the positive x-axis in the global coordinate system (e.g., a heading of zero with a equal positive wheel velocities will drive positively

along the x-axis. This can be computed more accurately in terms of distance because the time estimate is not involved:

$$x_n = \frac{1}{2}(d_{R_n} + d_{L_n})\cos(\theta_n) + x_{n-1} \quad (8)$$

$$y_n = \frac{1}{2}(d_{R_n} + d_{L_n})\sin(\theta_n) + y_{n-1} \quad (9)$$

$$\theta_n = \frac{1}{D}(d_{R_n} - d_{L_n}) + \theta_{n-1} \quad (10)$$

## 1.2 PID Loop Equation

Two independent variables can be used to control the robot: the destination  $P_d = (x_s, y_s)$  and speed  $q_{v_d} = (q_l + q_r)/2$ . These setpoints and the current pose  $(x_n, y_n, \theta_n)$  are used to calculate the next set of motor speed commands.

The PID control equations are

$$\epsilon_{i,n} = (\theta_{d_n} - \theta_n) \quad (11)$$

$$I_n = \rho I_{i,n-1} + \epsilon_{i,n} \delta_n \quad (12)$$

$$u_n = K_p \epsilon_{i,n} + K_i I_{i,n} + K_d (\epsilon_{i,n} - \epsilon_{i,n-1}) / \delta_n \quad (13)$$

where the error  $\epsilon_{i,n}$  is the difference between the desired heading and the current heading (pose),  $I_{i,n}$  is the integrated error with leakage  $0 \leq \rho \leq 1$ , and  $K_p$ ,  $K_i$  and  $K_d$  are the loop control parameters.

The PID output  $u_n$  should tend to correct for heading errors both in sign and magnitude, but is not bounded by the limits of the motor speed control's inputs, nor does it in any way determine the absolute average speed. It would be perfectly content to find itself pointing in exactly the right direction but stationary and thus never get there at all. There are a variety of ways to accomodate a desired speed, but simply adding steering offsets to it to arrive at individual motor speed settings creates difficulties when both the speed is high and the heading error great. Limiting creates discontinuities in the derivatives which are unsettling to the PID.

I am taking a different approach which uses two bounded entire functions to control the individual

motors speeds, this providing continuous response to changes in PID output. The first maps the PID output through an arctangent, which provides a bounded range of outputs to unbounded inputs, is monotonically increasing, and maps 0 to 0. The second uses a gaussian to decrease the desired average speed as the motor speed difference increases:

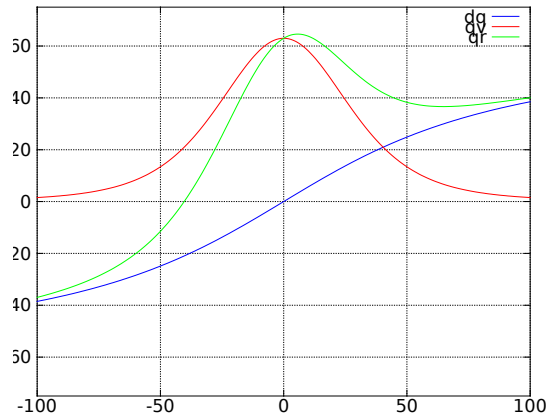
$$d_q = d_{q_{max}} \operatorname{atan}(u_n/u_0) \quad (14)$$

$$q_v = q_{v_d} e^{-(d_q/d_{q_0})^2} \quad (15)$$

$$q_r = q_v + d_q \quad (16)$$

$$q_l = q_v - d_q \quad (17)$$

The speed differential  $d_q$  depends on an additional  $u_0$  parameter which controls the sensitivity of the rate of change to PID output  $u_n$ . The absolute average speed  $q_v$  depends on an additional  $d_{q_0}$  parameter which must be adjusted to prevent the sums  $q_r$  and  $q_l$  from ever exceeding  $q_{max}$  in magnitude. The figure shows the interaction.



Tuned properly, this results in high errors requiring high rates of turn being performed in place ( $q_v = 0$ ) rather than while moving, and a continuous transition between that and turning more slowly while moving.

### 1.3 Calibration

There are several parameters which need to be calibrated, most notably the wheel circumference and spacing. Tests have been devised to separate these.

#### 1.3.1 Wheel Circumference

If the robot is driven in a straight line, the distance it travels together with the number of revolutions the wheels make can be used to compute the wheel circumference through equations (1) and (2).

#### 1.3.2 Wheel Separation

If the motors are set to fixed speeds that differ one side to the other, the robot will move in a circle whose circumference is directly related to the wheel separation.

Is this independent of circumference? It might be!

## References

- [1] *Experiment I; Kinematics of a car-like mobile robot; Mobile robot basic - differential drive*, Institute of Informatics VII; Julius Maximilian Univeritat Wurzburg, Updated 22 March 2003.
- [2] Wikipedia PID Control article [http://en.wikipedia.org/wiki/PID\\_controller](http://en.wikipedia.org/wiki/PID_controller)