# COSMOS: Making Robots and Making Robots Intelligent
# Lecture 9: Control design and implementation of `Robobrain`

Jorge Cortés and William B. Dunbar

October 27, 2005

**Abstract**

In this lecture, we restate the model for `Robobrain` and the IR wall-based sensing algorithm. We then define, simulate and analyze a feedback controller designed to make `Robobrain` track a signal near the wall.

## Contents



## 1 `Robobrain` Model

The schematic image of an enlarged `Robobrain` robot sitting the middle of a room, given in Figure 1. In Figure 1, the three triangle shapes on the left side of the robot represent three IR sensors, and the single triangle shape represents the front IR sensor. The discrete-time dynamic model of the robot dynamics is given by

$$\left.\begin{aligned} x_{k+1} &= x_k + \Delta u_k \cos(\theta_k) \\ y_{k+1} &= y_k + \Delta u_k \sin(\theta_k) \\ \theta_{k+1} &= \theta_k + \Delta v_k \end{aligned}\right\} \tag{1}$$

If we have designed a feedback controller specifying $(u_k, v_k)$, we convert these into the desired angular velocities using the previously derived equations:

$$\omega_R(t_k) = \frac{u_k}{r} + \frac{w v_k}{4\pi r}, \quad \omega_L(t_k) = \frac{u_k}{r} - \frac{w v_k}{4\pi r},$$

where $w$ is the width of the robot, as shown in Figure 1, and $r$ is the radius of each wheel.

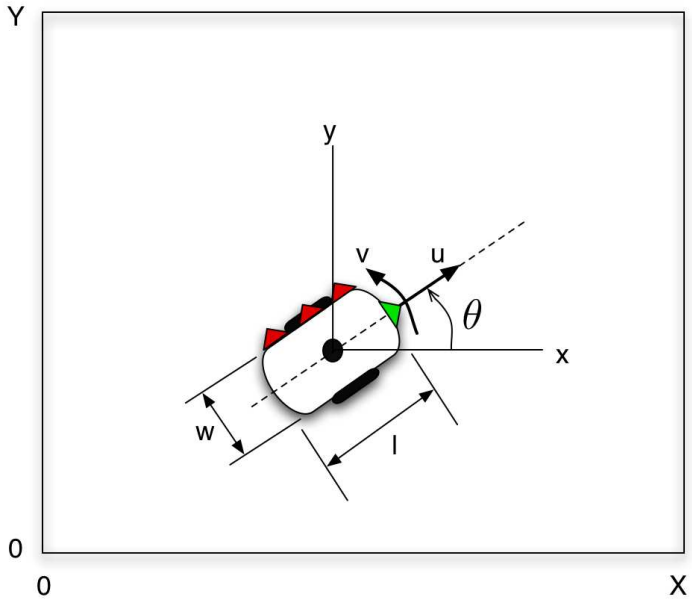


Figure 1: Schematic drawing of an enlarged Robobrain robot operating in a room. The border in the drawing represents the room walls, and the coordinate system shows that the lower left corner is denoted $(x, y) = (0, 0)$ and the upper right corner is denoted $(x, y) = (X, Y)$, with $X$ and $Y$ given by the dimensions of the room. The triangles depict the IR sensors used to determine distance and heading relative to the wall.

## 2 Wall-Based IR Sensing

The objective of the control design is for the robot to approach any wall in any room or hall way and, upon detecting the wall using IR sensors, turn to track a specified separation distance $d_{\text{sep}}$ relative to the wall. To do so, the robot will use IR sensors to keep track of distance and heading relative to the wall. For now, assume that $d_{\text{sep}}$ is constant. Later, we will make it time varying, e.g., a sinusoid.

As defined in the previous lecture, the measured distances to the wall at any time $t_k$ from each sensor as follows: $d_f(t_k)$ is measured from the front, $d_{l,f}(t_k)$ from the left front, $d_{l,b}(t_k)$ from the left back, and $d_{l,m}(t_k)$ from the left middle. The distances are shown pictorially in Figure 2. From the left sensor distance measurements, we define the left average distance as

$$d_l(t_k) = \frac{1}{3}\left(d_{l,b}(t_k) + d_{l,m}(t_k) + d_{l,f}(t_k)\right)$$

at any time. The following algorithms summarize how the IR distance measurements are used to compute the configuration variables $x_k$ and $\theta_k$, assuming a straight long hallway.

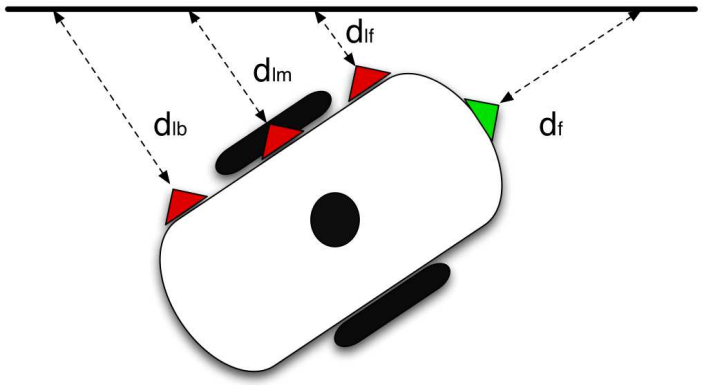


Figure 2: Schematic of robobrain sensing a nearby wall, where the triangles depict the IR sensors used to determine distance and heading relative to the wall. The distances measured are $d_f$ from the front sensor, $d_{l,f}$ from the left front sensor, $d_{l,b}$ from the left back sensor, and $d_{l,m}$ from the left middle sensor.

**Name:** `Robobrain` **sensor initialization algorithm**
**Goal:** Initialize position of `Robobrain` to follow a wall

1: Go forward with control $u = v_{\text{nom}}$ and $v = 0$ until $d_f \approx 2d_{\text{sep}}$. The positive scalar $v_{\text{nom}}$ is the nominal wall following velocity, defined to be some percentage of the maximum velocity of a wheel.
2: Turn clockwise slowly in place with control $u = 0$ and $v = -0.05v_{\text{nom}}$ until all three left sensors registers wall measurements. Continue to turn slowly, keeping track of the values of the left sensors. As soon as $d_{l,b} = d_{l,m} = d_{l,f}$, stop.
3: Reset time to be $t_0 = 0$ and define $\theta_0 = \pi/2$ and $x_0 = d_l(t_0) + w/2$. Note that $d_l(t_0) = d_{l,b}(t_0) = d_{l,m}(t_0) = d_{l,f}(t_0)$.

**Name:** `Robobrain` **sensor update algorithm**
**Goal:** Initialize position of `Robobrain` to follow a wall

For all $k = 1, 2, 3, ...$, the configurations $x_k$ and $\theta_k$ are given as follows:
1: set $\theta_k = \theta_0 + \alpha_k$, where

$$\alpha_k = \frac{1}{3}\left\{ \arctan\left[\frac{d_{l,b}(t_k) - d_{l,f}(t_k)}{l}\right] + \arctan\left[\frac{d_{l,m}(t_k) - d_{l,f}(t_k)}{l/2}\right] + \arctan\left[\frac{d_{l,b}(t_k) - d_{l,m}(t_k)}{l/2}\right]\right\}$$

where $l$ is the length of `Robobrain`.
2: set $x_k = \left[d_l(t_k) + \frac{w}{2}\right]\cos(\alpha_k)$.

**Task 2.1** Write a program that converts the three left scalar distance measurements $d_{l,f}$, $d_{l,m}$ and $d_{l,b}$ into the state values $x$ and $\theta$ using the equations given above. Call the program `sensorupdate.m`.

# 3 Control Design

We are now in a position to begin to design controllers so that `Robobrain` will track a desired separation distance from a wall. For now, we will present you with a control. The discrete proportional-derivative wall following control is given by

$$u_k = v_{\text{nom}}, \quad v_k = k_p\,(x_k - d_{\text{sep}}) + k_d \frac{x_k - x_{k-1}}{\Delta},$$

An example simulation where this controller is employed is shown in Figure 3. Note that this

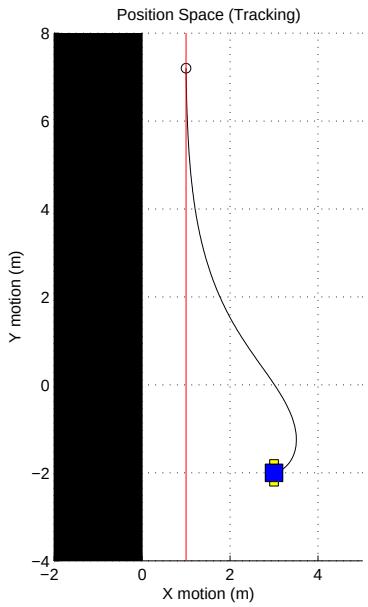


Figure 3: Simulation of feedback control for keeping a desired separation from the wall at a constant speed. Initial condition is $(x_0, y_0, \theta_0) = (3, -2, 0)$, nominal wall following velocity is $v_{\text{nom}} = 1$ and desired separation is $d_{\text{sep}} = 1$. Control gains are $k_p = 1/3$ and $k_d = 1$. Sample period is $\Delta = 0.05$.

controller ignores $y$. For that matter, we have not even discussed the possibility of sensing $y$.

**Task 3.1** Can $x$ and $y$ be controlled independently? Can we be content with just controlling $x$? Should we be?

**Task 3.2** Modify the file **robobrain.m** that you created in the previous lecture. Rename the file and function to be **robobrainControl.m**. Implement the feedback controller defined above. This new function should ask for the initial configuration $x_0, y_0, \theta_0$ of `Robobrain`, the time step $\Delta$, the number of iterations $N$, the desired separation distance $d_{sep}$ and the control gains $k_p$, $k_d$. The function should output a vector with components $x, y, \theta$ and the elapsed time. The first line of the file should then by something like

function [x,y,theta,time] = robobrainControl (x0,y0,theta0,delta,N,kp,kd,dsep)

Generate a plot of $y$ vs. $x$ using the same parameters defined in the caption of Figure 3.

**Task 3.3** Go to `www.soe.ucsc.edu/~jcortes/cosmos`, download the file `robot_movie.m` and save it in your working directory. In order for this program to work, you need to have `robobrainControl.m` working well. The function `robot_movie.m` outputs a visualization of the feedback controller. Type help `robot_movie.m` to understand the syntax and use the program to get cool movies of `Robobrain` working.