6.141/16.405J: Robotics systems and science Lecture 2: Motor Control

Lecture Notes Prepared by Daniela Rus

EECS/MIT

Spring 2009

http://courses.csail.mit.edu/6.141/

Challenge: Build a Shelter on Mars

This week we will see

- Why is robot control hard
- Basic Control: Bang-Bang, P, D, I, PID
- DC Motors
- Stepper motors
- PWM control
- Encoders

Today:

- The role of *control*
- Open-loop (feed-forward) control
- Closed-loop (feedback) control
- Basic controllers:
  - P
  - PD
  - PID

The Role of Control

- Many tasks in robotics are defined by *achievement* goals
  - Go to the end of the maze
  - Push that box over here
- Other tasks in robotics are defined by *maintenance* goals:
  - Drive at 0.5m/s
  - Balance on one leg

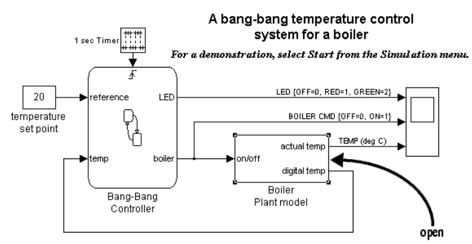
The Role of Control

- *Control theory* is generally used for low-level maintenance goals
- General notions:
  - output = Controller(input)
  - output is control signal to actuator (e.g., motor voltage/current)
  - input is either goal state or goal state error (e.g., desired motor velocity)
- Controller is stateless

Bang-bang control

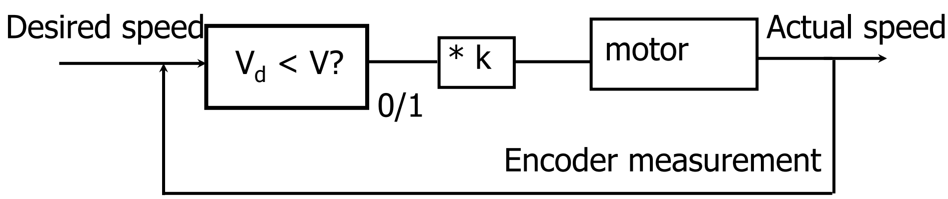
- Discrete on/off
- Example: furnace:
  goal temp = 70
  when temp < 70
  BANG! Heat;

  when temp > 70
  BANG! Stop the heat



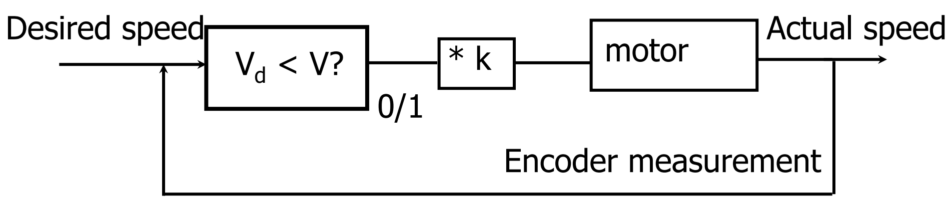
Example source: Mathworks

Bang-bang control



O(t) =
O(t) =

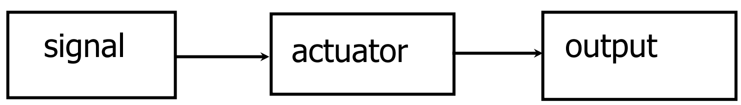
Bang-bang control



O(t) = k if $v(t) < V_d$
O(t) = 0 otherwise

Motor Control: Open Loop

- Give robot task with no concern for the environment
- Applications: ???
- Open loop: signal to action
- Not checked if correct action was taken
- Example: go forward for 15 secs, then turn left for 10 secs. Issues?

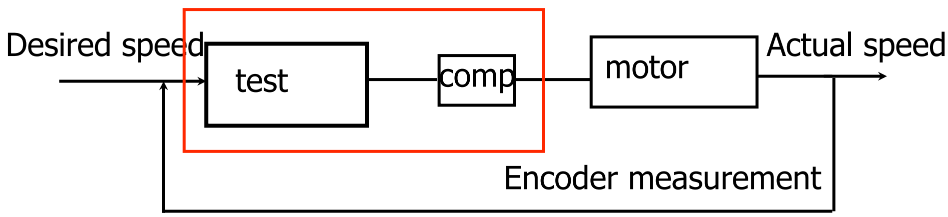


Open loop (feed-forward) control

- Open loop controller:
  - output = FF(goal)
- E.g.: motor speed controller (linear):
  - V = k * s
  - V is applied voltage on motor
  - s is speed
  - k is gain term (from calibration)
- Weakness:
  - Varying load on motor => motor may not maintain goal speed

Motor Control: closed loop

- A way of getting a robot to achieve and maintain a goal state by constantly comparing current state with goal state.
- Use sensor for feedback



Feedback Control

- Feedback controller:
  - output = FB(error)
  - error = goal state - measured state
  - controller attempts to minimize error
- Feedback control requires sensors:
  - Binary (at goal/not at goal)
  - Direction (less than/greater than)
  - Magnitude (very bad, bad, good)

Example: Wall Following

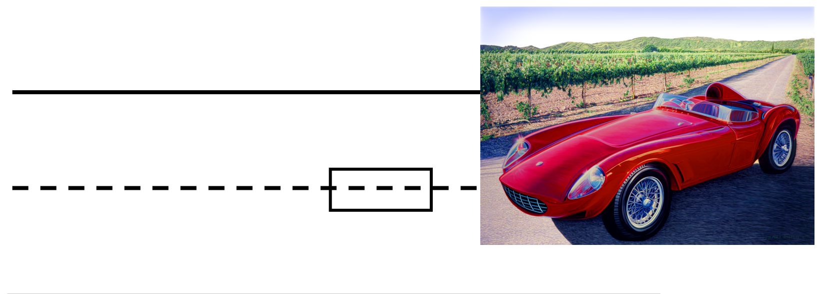
- How would you use feedback control to implement a wall-following behavior in a robot?
- What sensors would you use, and would they provide magnitude and direction of the error?
- What will this robot's behavior look like?

Motor Control: PID

- *Control theory is the science that studies the behavior of control systems*
- *CurrentState - DesiredState = Error*
- Three main types of simple linear controllers:
  - P: proportional control
  - PD: proportional derivative control
  - PID: proportional integral derivative control
- All use direction and magnitude of error

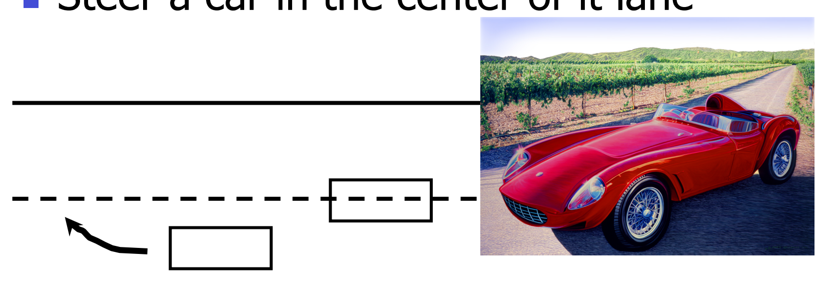
Example: driving

- Steer a car in the center of a lane



Example: driving

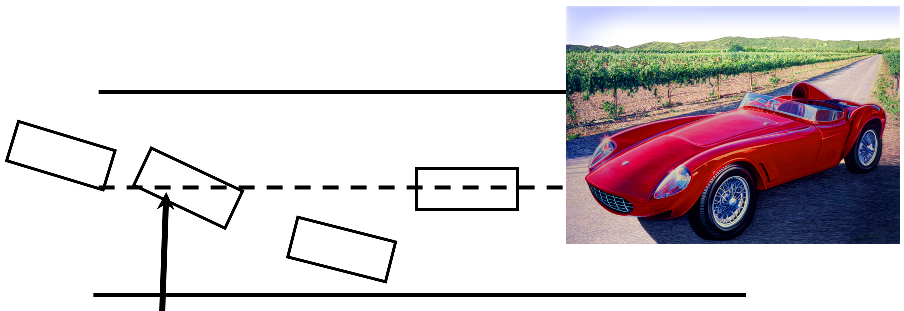
- Steer a car in the center of it lane



Observed error: speed of lateral movement

Example: driving

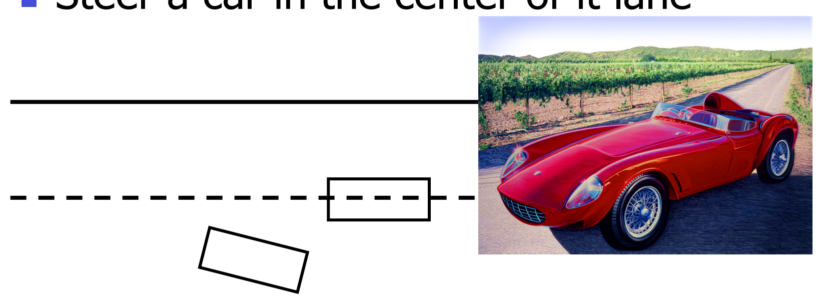
- Steer a car in the center of it lane



Error is zero but how is the car pointed?
What will this do to the car?
P controller is happy on line independent of orientation!

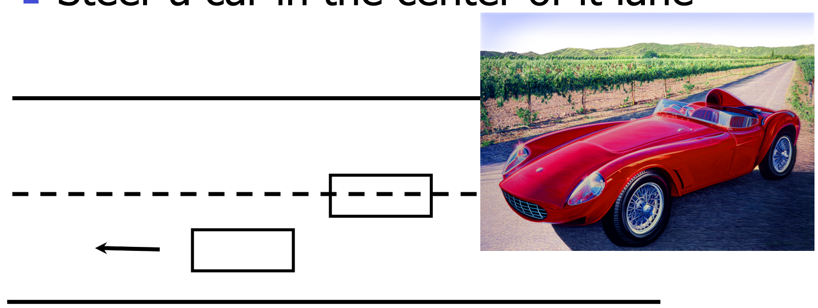
What if respond ~ rate of change ?

- Steer a car in the center of it lane



Example: driving

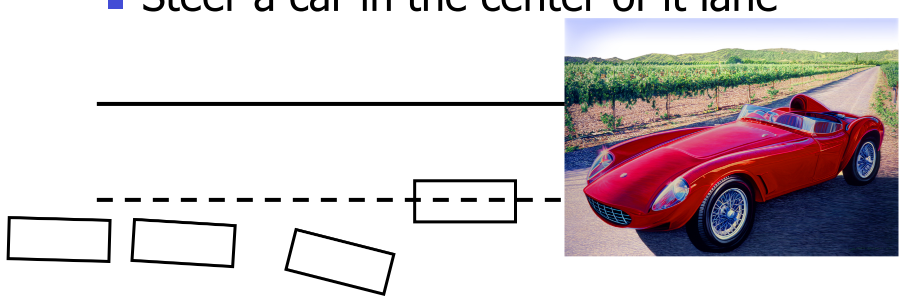
- Steer a car in the center of it lane



What is the observed rate of error?
Other error?

What if respond ~ rate of change ?

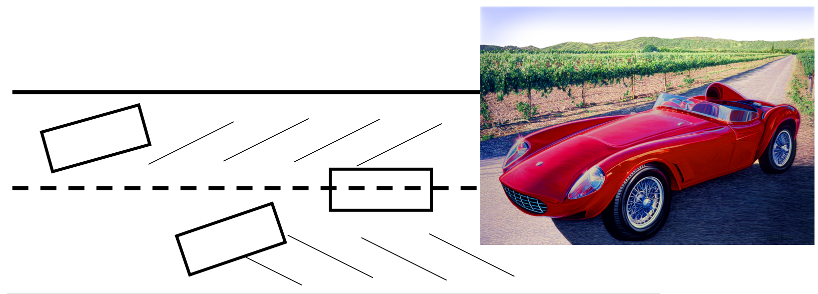
- Steer a car in the center of it lane



D controller is Happy on any parallel line!

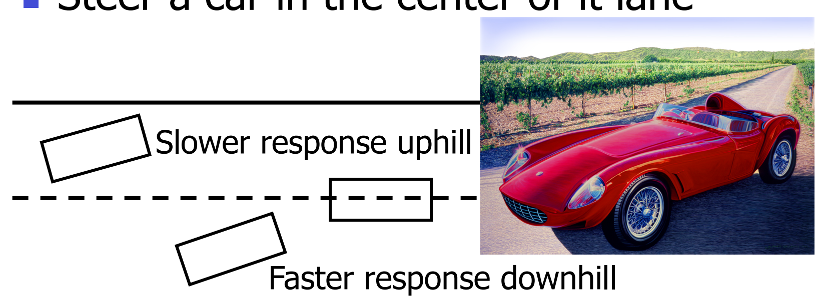
What if Road Sloped ?

- Steer a car in the center of its lane



What if Road Sloped ?

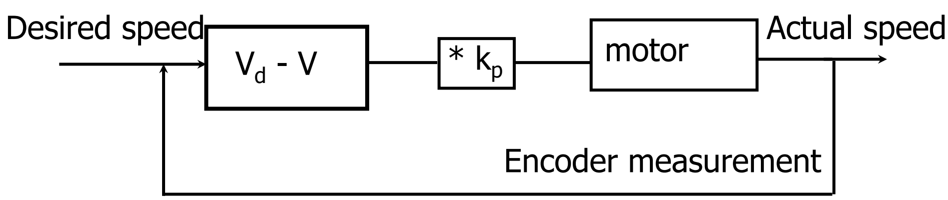
- Steer a car in the center of it lane



Gravity contributes a steady-state error

Proportional Control

- Act in proportion to the error
- A proportional controller has an output o proportional to its input i:
  $o = K_p * (V_d - v) = K_p * i = K_p * err(t)$
- Kp is a proportionality constant (gain)



Gains

- How do we decide how much to turn, or how fast to go?
- Parameters (K_p) are called gains
- Determining the right gains is difficult
- It can be done
  - analytically
  - empirically
  - trial and error

Damping

- Damping is the process of systematically decreasing oscillations
- A system is *properly damped* if it does not oscillate with increasing magnitude, i.e., if its oscillations are either avoided, or decrease to the desired set point within a reasonable time period

Integral Control

- Act in proportion to the *accumulated* error
- An integral controller has an output o proportional to the integral of its input i:

  $o = K_i * \int i(t)\, dt$
- Ki is a proportionality constant
- Integral control is useful for eliminating steady-state errors

Derivative Control

- Act in proportion to the *rate of change* of the error
- A derivative controller has an output o proportional to the *derivative* of its input i:

$$o = K_d * di/dt = K_d * derr(t)/dt$$

- Kd is a proportionality constant

PD Control

- PD control combines P and D control:

$$o = K_p * i + K_d * di/dt$$

- P component minimizes error
- D component provides damping
- Gains Kp and Kd must be tuned together

PID control

- PID control combines P and D control:

  $o = K_p * i + K_i * \int i(t)\,dt + K_d * di/dt$

  $o = K_p * err + K_i * \int err(t)\,dt + K_d * derr/dt$
- P component minimizes instantaneous error
- I component minimizes cumulative error
- D component provides damping
- Gains must be tuned together

Ziegler-Nichols Tuning Method

Exploration: set the plant under P control and start increasing the gain until loop oscillates

Note critical gain K_C and oscillation period T_C

	K_P	K_I	K_D
P	$0.5K_C$		
PI	$0.45K_C$	$1.2K_P/T_C$	
PID	$0.5K_C$	$2K_P/T_C$	$K_PT_C/8$

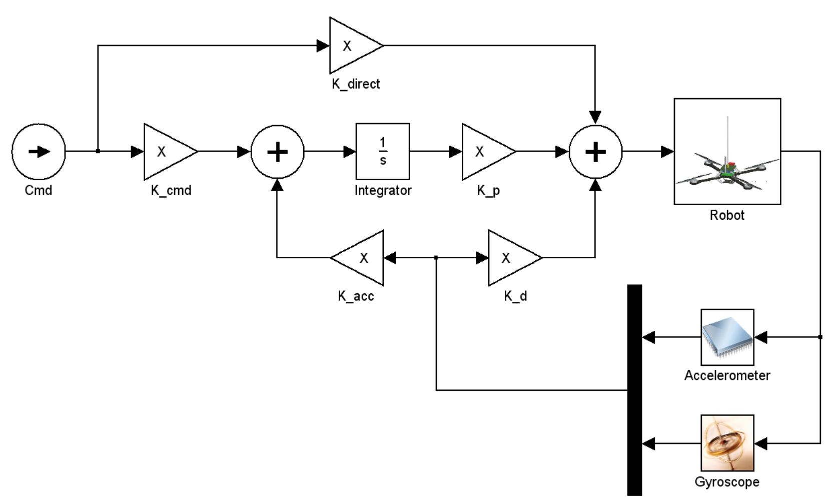
Z & N developed rule using Monte Carlo method

Rule useful in the absence of models

Implementation Issues

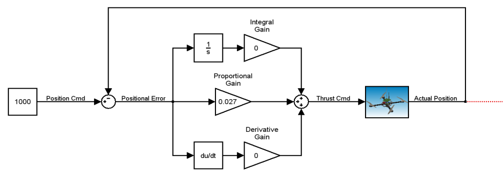
- How do we approximate $K_i * \int err(t)\, dt$ to implement an I controller?
- How do we approximate $K_d * derr/dt$ to implement a D controller?

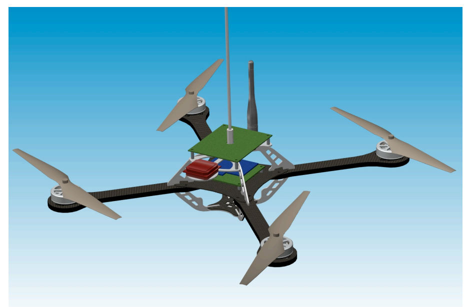
Quad-Rotor Controller



Example UFO Control Using Ziegler-Nichols Method

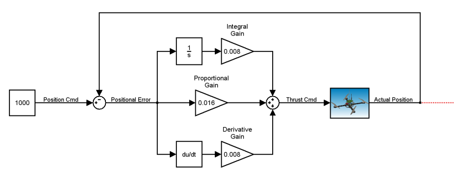
- Integral and Derivative gains are set to zero
- Proportional gain is increased until system oscillates in response to a step input
- This is known as the critical gain K_C, and the system oscillates with a period P_C

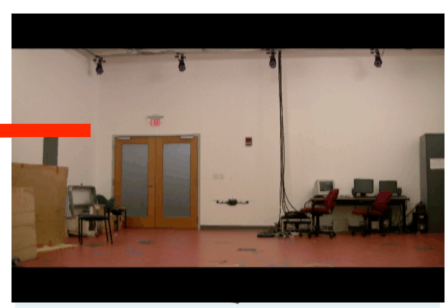




Calculating PID parameters

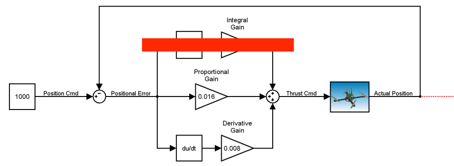
- $K_P = 0.6*K_C$
- $K_I = 2*K_P / T_C$
- $K_D = 0.125*K_P*T_C$
- $T_C = 4$
- The Ziegler-Nichols Method is a guideline for experimentally obtaining 25% overshoot from a step response.





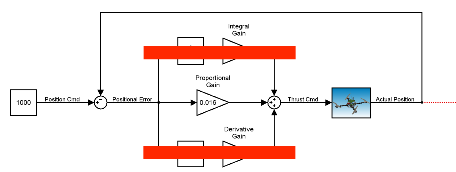
What does K_I do?

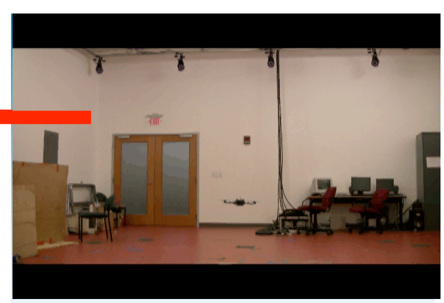
- For this system, K_I is crucial in eliminating steady state error primarily caused by gravity.
- The PD system overshoot and damping are still reasonable.



What does K_D do?

- Reasonable K_D helps minimize overshoot and settling time.
- Too much K_D leads to system instability.
- P system has unacceptable overshoot, settling time, and steady state error. However, it is stable.





Control summary

	Control type	Feedback	Pro/Con
Bang-bang	discreet	environment	Simple/ Discreet
Open loop	Control law	no	Simple/may be unrepeatable
Closed loop	P, I, D	yes	Continuous/ Tune Gains