CS 360:  
Advanced Artificial Intelligence  
Class #2: Introduction to Robotics  

Daniel M. Gaines  

Objectives:  

• Know what robots are made of  
• Know how sensors and effectors work  
• Know how to build our robots  
• Know how to program our robots
Review of Previous Classes

- Introduced field of robotics and relation to course
  - history
  - uses
  - challenges
Overview of Today’s Class

• Look in more detail at what robots are made of
  – sensors
  – effectors
  – computation

• Introduction to the robots we’ll use
What Are Robots Made Of?

- 6 major subsystems
  1. Structure
  2. Sensors
  3. Effectors
  4. Power
  5. Computation
  6. Information

---

1Dave Miller of KISS Institute
1. Structure\textsuperscript{2}

- Carries forces exerted by robot and environment
- Effectors attached
- Holds sensors

\textsuperscript{2}Dave Miller of KISS Institute
• Examples:

wheeled platform

LEGO
2. Sensors

• Allow robot to sense the world
  – sensors convert physical phenomena into electrical signals that the microprocessor can read

• Two main classes
  – **propiroceptive**: report on current state of robot
    * position of joints
    * position of robot
    * orientation
    * hunger
  – **external**: report on state of world
    * amount of light
    * position of objects
    * temperature
    * wind
Example Sensor-Rich Robots

- Robart II:
  - mobile sentry robot
  - 5 foot tall
  - sensors include:
    - sonar sensors
    - infrared sensors
    - bump sensors
    - microwave motion sensors
    - surveillance camera
    - earthquake sensor
    - flood sensor
- rough terrain explorer
- six-legged, shoe-box sized
- over 60 sensors (23 motors and 11 computers)
- legs sensors allow it to detect and step over obstacles:
  - force sensors
  - touch sensors
  - color sensors
  - potentiometers
- chassis sensors include:
  - force-sensing whisker
  - gyroscope
  - pitch-and-roll sensor
  - near-infrared rangefinder
  - small camera
Comparison with Other Creatures

- Today’s robots impoverished compared to animals and insects

- Our skin is covered with pressure, heat, cold, pain and itch sensitive nerve endings
  - Each fingertip has about 100 pressure sensors

- Cockroaches have 30,000 wind-sensitive hairs on their legs
  - Can sense change in wind direction and change direction within 10 ms

- No robot today comes even close
Interfacing Sensors

• Need microprocessor to read sensor information
  – must convert physical phenomena measured by sensor into a number read by microprocessor

• Typical approach:
  – microprocessor measures voltages
  – sensor placed in a circuit such that different physical phenomena correspond to different voltages
  – e.g. bump switch can switch voltage to high or low depending on contact
  – for the MC68HC11:
    * voltage range: 0 - 5 V
    * converted into 256 discrete levels (i.e. 0-255)
Two types of sensors (from microprocessor perspective):

- **analog sensors**: voltage placed on sensor line proportional to amount of measured phenomenon
  * e.g. higher voltage may correspond to higher voltage
- **digital sensors**: voltage used to encode a number
  * e.g. 0 (low) for no contact, 1 (high) for contact
  * e.g. Sharp infra-red range finder encodes measured distance to object in an 8 bit number
Software drivers: code telling microprocessor how to read the sensor, examples:

- continually poll a particular pin waiting for voltage to change
- Sharp GP2D02 Digital Infra-Red Ranging System
  * send a signal to sensor indicating it should begin a measurement (set line to low)
  * sensor signals on another line that measurement is complete (sets line to high)
  * sensor then sends 8 bits on sensor line, processor must read in these bits
    - need to clock in the data
    - processor generates 8 pulses on one line
    - reads value sensor places on its output line between pulses
Photoresistor (Photocell)

- A resistor that changes resistance proportional to intensity of light
  - resistive element made from Cadmium Sulfide
- **Typical use:** light detector
  - used as a voltage divider
  - as light increases, resistance drops
  - as resistance drops, voltage drops \((V = IR)\)
Infrared Reflectance Sensor

- Technically, these are near-infrared devices
- Measures amount of infrared light received
  - often conveniently paired with an infrared transmitter
- **Typical uses:**
  - proximity detection
  - joint position detection
  - color detection
IR Range Sensor

- Measures distance to object with triangulation:

- **Typical use:** range finding, obstacle detection
Pyroelectric Sensor

- Sensitive to IR emitted by human skin
- **Typical use:** motion detection
Encoder

• Combination of:
  – IR transmitter
  – IR receiver
  – encoder wheel

• How it works:
  – transmitter aimed at receiver
  – teeth on wheel breaks light

• Typical uses:
  – odometer
  – speedometer
  – joint angle
  – direction
Bump Sensor

- Switches when contacted/released

**Typical uses:**
- collision detection
- joint position detection
• Combination of **transducer** (speaker) and receiver
  – transducer emits an ultra-sonic **ping**
  – if an object is w/in range, the ping bounces back
  – receiver detects ping
  – **Time of Flight (TOF)** used to measure distance

• Range: typically 1 foot - 35 feet

• **Typical use:** range finding, obstacle detection
Redundant Sensors

• Good to use a variety of sensors for a particular purpose
  – e.g. sonar, IR proximity and bump for obstacle detection

• Increases robustness

• Can often extract info not available otherwise:
  – e.g. IR reflectance sensor color and range sensitive
  – Sharp's rangefinder color insensitive
  – together, can detect object color
    * rangefinder fixes distance
    * IR reflectance sensor detects color

\[ ^3 \text{although it is a little color sensitive} \]
Effectors

• Allow robot to affect the world
  – change its own state
  – change world state

• Examples
  – motors
  – grippers
  – arms
  – legs
  – speakers
  – lights
  :
**Locomotion for Wheeled Robots**

- Wheel arrangements impact the control of the robot

- **Differential drive:** two wheels on common axis, driven independently
  - usually a extra caster wheel(s) for support
  - mechanically simplest
  - allows robot to spin in place
  - turn by moving wheels at different speeds / directions
  - challenges:
    * motors differ
    * if one wheel is on different surface, they will move at different rates
• **Synchro drive:** all wheels point in same direction, driven together
  – all wheels steer and drive
  – to turn: all wheels rotate together about a vertical axis
  – simplest for control
  – overcomes problems of differential drive
  – mechanically complex

• **Car/tricycle drive:** drive wheels and steer wheel(s)
  – aka: **Ackerman steering**
  – overcomes problems of differential drives
  – to go straight: place steering wheels in neutral position
  – kinematics more difficult: may not be able to move to all points at any orientation
DC Motors

- Axel rotates when current applied
  - follow this link to find out why

- **Gears:** trade speed for torque

- Interfacing motors:
  - need to protect processor from high currents drawn by motors
  - need a way to change motor direction
  - need a way to change speed
H-Bridge

• Isolates processor from high currents of motors
  – processor can control motor while being isolated from high current

• Allows processor to control motor direction
  – switches control direction of current
  – processor can control switches
Motors Off
Motors On
Motors On: Reverse Direction
Controlling Speed

- One way to control speed is to change voltage
  - however, computer not good for changing voltage

- Alternative, **Pulse Width Modulation**
  - turn motors on and off for different periods of time
  - changes the **average voltage**
Servo Motors

• Turns to a specified position

• Composed of:
  – DC motor: provides rotation
  – limit stops: restricts rotation within limits
  – potentiometer: position feedback
  – integrated circuit: position control

• Three lines:
  1. power
  2. ground
  3. control input
4. Power

• Unsolved problem for mobile robots
  – much of our robot size and weight is taken up by batteries!

• Our power source: rechargeable NiCd batteries
5. Computation

- interprets sensors: perception
- controls effectors
- performs prediction/planning
- must be lower power
- must be integrated with sensors and effectors
The Handy Board
6. Information

- Internal
  - how to interpret sensors
  - how to control effectors
  - keep track of state and history

- External
  - world models
  - predictive models

- How to do it?
  - focus of this course
IC: Interactive C

• Language we’ll use to program our robots

• Features include:
  – ability to start and stop processes
  – ability to execute statements interactively
    ∗ useful for developing/debugging code
Example IC Program: Wander Program

1. Start up a process to check the sensors

2. Wait for Start button

3. Go forward

4. If stop button hit, stop

5. If right touch sensor is activated
   (a) back up
   (b) turn left

6. If left touch sensor is activated
   (a) back up
   (b) turn right

7. Go to Step 3
/* Constants */
#define MOTOR_RIGHT 0
#define MOTOR_LEFT 1
#define TOUCH_RIGHT 12
#define TOUCH_LEFT 9

/* Globals */
int stopped = 0;
int started = 0;
int touchedLeft = 0;
int touchedRight = 0;

/* If your robot arcs to the right, make drive_bias positive,
   arcs to the left require a negative correction. */
int drive_bias = 0;

void main ()
{
    start_process (monitorSensors());
    printf ("Press Start...
"));
    while (!started) { sleep (0.1); }
    start_process (wander());
}
int wander ()
{
    while (!stopped) {
        driveb (80, 0);
        if (touchedRight) {
            driveb (-50, 0); sleep (2.0); driveb (0, -50); sleep (2.0);
        }
        if (touchedLeft) {
            driveb (-50, 0); sleep (2.0); driveb (0, 50); sleep (1.0);
        }
    }
    ao();
}

void monitorSensors ()
{
    while (1) {
        touchedLeft = digital (TOUCH_LEFT);
        touchedRight = digital (TOUCH_RIGHT);
        stopped = stop_button ();
        started = start_button ();
    }
}
void driveb(int trans, int rot) /* Correct for motor bias */
{
    int rot_bias = (drive_bias * trans) / 100;
    motor(MOTOR_RIGHT, trans - (rot + rot_bias));
    motor(MOTOR_LEFT, trans + (rot + rot_bias));
}