Robot Morphologies, Sensors, and Motors

Fall 2011

Prof. Holly Yanco
Robot morphology

• Some possibilities
  – Humanoid
  – Trashcan robots
  – Reconfigurable
  – Shape shifting
  – Biped
  – Quadruped
  – …
Why does shape matter?

- Shape influences expectations for interaction
- Differing capabilities with differing robot shapes
- Need to select shape to fit a task
Locomotion

- Terrestrial
- Aquatic
- Airborne
- Space
Locomotion: Wheels

• Differential Drive
  – Two wheels mounted on a common axis controlled by separate motors

• Synchronous Drive
  – Usually three wheels, all of which can be driven and steered

• Ackerman Steering
  – Car drive: front steering wheels turn on separate arms
Locomotion: Tracks

- Usually two treads
- May have ability to change shape
Locomotion: Legs

- Varying numbers of legs: 1, 2, 4, 6, etc.
- Usually more difficult to control than wheeled robots: need to consider gait and balance
- Can traverse areas that wheels can not
Sensors

• Also known as transducers
• Measure internal or external state
• Internal state, also called proprioception
  – Battery level
  – Wheel rotations
  – Position of gripper
• External state:
  – Location of objects with respect to the robot
  – Color of objects
Sensors

• Active
  – Emit energy into the environment to make observations
  – E.g., infrared, sonars
  – Don’t confuse active sensors (devices that emit energy) with active sensing (e.g., pan-tilt camera)

• Passive
  – Receive already existing energy from environment
  – E.g., bump sensor, video camera (without additional lighting)
Sensors

• Sensors which measure same form of energy and process it in similar ways form a modality

• Modalities refer to the raw input
  – Sound
  – Pressure
  – Temperature
  – Light
Types of Data Returned

• Range (distance to objects)
• Absolute position (e.g., latitude and longitude)
• Environmental (e.g., temperature or color of objects)
• Inertial (e.g., acceleration)
Sensors: Inertial Sensors

• Accelerometers
  – Measure acceleration in a single direction
• Gyroscopes
  – Measure change in orientation
• Compasses
  – Measure position relative to magnetic north
• Inclinometers
  – Measure orientation of the gravity vector
Sensors: Global Positioning

• Global Positioning Sensors (GPS) use satellite readings to determine location
• Differential GPS (DGPS) uses a base station to eliminate the intentional error in the satellite signals
Ranging Sensors: Sonar

- Sends an acoustic pulse and listens for echo
- Can determine how far away an obstacle is based upon time of flight and speed of sound
- Can have problems with specular reflections
Ranging Sensors: Infrared

• Fast and inexpensive proximity sensors
• Emit an infrared pulse and detect the reflected signal
• Infrared can be washed out by bright light
Ranging Sensors: Laser Ranging

• Uses one of the following methods
  – Triangulation (relationship between outgoing light beam, incoming ray and its position on the film plane)
  – Time-of-flight (similar to sonar or infrared)
  – Phase-based (based on difference between phase of emitted and reflected signals)

• Low-power ranges a few meters
• High-power can range over a kilometer
Sensors: Cameras

• Images from cameras must be processed in order for the robot to use the information
  – Edge detection
  – Tracking movement
  – Finding particular colors
  – Distance of objects from the camera
• We’ll talk more about vision later in the term
Problems with Sensors

• Noisy
• Return an incomplete description of the environment
• Can not be modeled completely

• We will come back to these issues throughout the term, particularly when we talk about mapping
Constructing a Sensor Suite

Consider the following for each sensor:

1. Speed of operation
2. Cost
3. Error rate
4. Robustness
5. Computational requirements
6. Power, weight and size requirements
Attributes of a Sensor Suite

Consider for entire suite:

1. Simplicity
2. Modularity
3. Redundancy
   - A sensor might be noisy or inaccurate by itself
   - False positives / False negatives
   - Complementary sensors can be used to provide disjoint information about a percept
Fusing Sensor Data:
Sensor Fission

```
sensor → percept → behavior → action
sensor → percept → behavior → action
sensor → percept → behavior → action

combination mechanism → action
```
Fusing Sensor Data: Action-Oriented Sensor Fusion

(sensor) → percept → fusion → percept → behavior → action
Motors
CBC Motor Ports

Motor ports 0 (left motor) and 3 (right motor)
Plugging in DC Drive Motors

- DC drive motors are the ones with two prong plugs
- The CBC has 4 drive motor ports numbered 0 & 1 on the left and 2 & 3 on the right
- When a port is powered it has a light that glows blue for one direction and red for the other
- Plug orientation order determines motor direction, but by convention, blue is forward and red reverse
CBC Motor Test Screen

- **Power**
- **Velocity**
- **Position**

**Target Speed**: 0

**Target Position**: 0

**Position**: 0

**Clear**

**GO**

Lecture 3, Slide 26
CBC Motor Commands

\texttt{mav()}, \texttt{ao()}, \texttt{off()}

- \texttt{mav(motor\#, vel)}; \quad [\text{mav=move at velocity}]
  - \textit{motor\#} is the motor port (0-3) being used
  - \textit{vel} is the rotational speed of the motor measured in ticks per second (-1000 to 1000)
  - the amount of rotation per tick depends on the kind of motor
  - the motor runs at the set speed until turned off or commanded otherwise

- \texttt{ao()}; turns off all motor ports

- \texttt{off(motor\#)}; turns off the specified motor port
Motor Position Counter

- As a DC motor runs, the CBC keeps track of its current position in ticks
  - `get_motor_position_counter(motor#)`;
    is a library function that returns this value for `motor#`
  - `clear_motor_position_counter(motor#)`;
    is a library function that resets the `motor#` counter to 0
  - You can see the current value of the counter for a motor on the `motors..test` and `Sensor Ports` screens
Motor Polarity

• Plug the drive motors into CBC motor ports 0 and 3 (corresponding to simbot DB1)
  – Motor port numbers are hard to see; they are numbered 0-3 starting from the black button side

• Check motor polarity
  – Manually rotate each motor and observe its power light (it will glow red or blue as you rotate the motor)
  – If a motor does not turn in the direction you want to correspond to forward (blue), reverse its plug
Servo Motors

- A servo is a motor designed to rotate to a specified position and hold it.
- To help save power, servo ports by default are not active until enabled.
- A command is provided in the CBC library for enabling (or disabling) all servo ports:
  - `enable_servos();` activates all servo ports
  - `disable_servos();` de-activates all servo ports
- `set_servo_position(2,925);` rotates servo 2 to position 925
  - Position range is 0-2047
  - You can preset a servo’s position before enabling servos so it will immediately move to the position you want when you enable servos.
  - Default position when servos are first enabled is 1024
- `get_servo_position(2);` provides the current position of servo 2
  - Works only when servos are enabled
CBC Servo Motor Ports

servo ports 0 and 1; servo ports 2 and 3
Plugging in Servos

- Servo motors (brown/black-red-yellow cables with 3 prong receptacle) plug into the CBC servo ports
- The CBC has 4 servo ports numbered 0 & 1 on the left and 2 & 3 on the right
- Plug orientation order is, left to right, brown/black-red-yellow when the CBC is oriented so the screen can be read (or follow the labeling: - + S; the yellow signal wire goes in S)
CBC Servo Test Screen
Sensor and Motor Manual

• For further detail about motors, consult the Sensor and Motor Manual on the course web site
Using Servos

• The CBC library functions for enabling (or disabling) all servo ports
  – `enable_servos();` activates all servo ports
  – `disable_servos();` de-activates all servo ports
• `set_servo_position(2, 925);` rotates servo 2 to position 925
  – You can preset a servo’s position before enabling servos so it will immediately move to the position you want when you enable servos
  – Default position when servos are first enabled is 1024
• `get_servo_position(2);` provides the current position of servo 2
  – Works only when servos are enabled
• The CBC Servo Test screen can be used to center a servo or determine what position values to use once the servo is installed on a bot
**while** Loop Operating a Servo

- A loop is a program construction used to repeat program steps until some condition is met.
- Suppose we want to have a servo move from position 200 to position 1800 in steps of 100
  - we could write 18 `set_servo_position` commands to do this, or
  - this can be done with a **while** loop:
    ```
    set_servo_position(2,200); // move servo 2 to position 200
    sleep(.1); // give it time to move
    while (get_servo_position(2) < 1800)
    {
        // move servo 2 in steps of 100
        set_servo_position(2,get_servo_position(2)+100);
        sleep(.1); // give it time to move
    }
    ```
while Loop in a Program

Run in the simulator

```c
int main() {
    // preset servo 2 position
    set_servo_position(2, 200);
    enable_servos();  // turn on servos
    sleep(.1);  // pause while it moves

    while (get_servo_position(2) < 1800) {
        // move servo 2 in steps of 100
        set_servo_position(2, get_servo_position(2) + 100);
        sleep(.1);  // pause while it moves
    }

    disable_servos();
}
```

Lecture 3, Slide 37
CBC Sensors
CBC Sensor Ports
Analog and Digital

analog ports (0-7) and digital ports (8-15)
## CBC Sensor Ports Screen

<table>
<thead>
<tr>
<th>Analog Port</th>
<th>Value</th>
<th>Digitals</th>
<th>Value</th>
<th>Accelerometer</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1022</td>
<td>8</td>
<td>0</td>
<td>X</td>
<td>-9</td>
</tr>
<tr>
<td>1</td>
<td>1021</td>
<td>9</td>
<td>0</td>
<td>Y</td>
<td>-7</td>
</tr>
<tr>
<td>2</td>
<td>1022</td>
<td>10</td>
<td>0</td>
<td>Z</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>1022</td>
<td>11</td>
<td>0</td>
<td>Voltage</td>
<td>7.673</td>
</tr>
<tr>
<td>4</td>
<td>1022</td>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1023</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1022</td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1022</td>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor</th>
<th>Value</th>
<th>Power</th>
<th>Value</th>
<th>Position</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-17</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
CBC Sensor Ports Screen

• Go to the Sensor Ports screen
  – The Sensor Ports tab is Under the Sensors/Motors tab on the opening screen

• Plug the two sensors to be used for this activity into analog ports on the CBC
  – Plug the light sensor into the same port you chose in Activity B-a2
  – Plug the reflectance sensor into any other available analog port

• When you point the reflectance sensor towards the IR light sensor you should see a low value for its port

• If you aim the reflectance sensor at the table and move it across the table edge its value will change
digital()

- For a digital sensor such as a button sensor plugged into digital port 8, `digital(8);` will provide the current value of the sensor
  - A digital sensor’s value is 0 if off and ≠0 if on
Sensors for Activity
Using the CBC and sensors

• **IR light sensor**
  – Analog sensor (pull up)
  – Plug into any port 0-7

• **Reflectance sensors**
  – Analog sensor (pull up)
  – Plug into any port 0-7
  – Has an IR emitter and an IR detector
  – Light source for this activity
analog10()

- For an analog sensor such as a light or reflectance sensor plugged into analog port 2, \texttt{analog10(2)} will provide the current value of the sensor
  - An analog sensor provides a range of values
  - The \texttt{analog10} function gives values from 0-1023
Analog and Floating Analog Sensors
IR Reflectance Sensors

- An IR reflectance sensor has an emitter producing an IR beam and an IR light sensor that measures the amount of IR reflected when the beam is directed at a surface.
- There are two reflectance sensors in the Botball kit.
- The CBC library function `analog10` returns an amount that measures the amount of light reflected as a value between 0 and 1023.
- A dark spot reflects less IR, producing a higher reading.
IR Reflectance Sensor Behavior

Amount of reflected IR depends on surface texture, color, and distance to surface (higher values mean less IR indicating a dark surface or a drop off)
Sample code

- Plug a reflectance sensor in port 0 and a light sensor in port 2

```c
int main() {
    while (a_button() == 0) {
        printf("reflectance: %d ", analog10(0));
        printf("light: %d\n", analog10(2));
        sleep(1);
    }
}
```
Analog and Floating Analog Sensors

- For an analog (e.g., light sensor or reflectance sensor) or a floating analog sensor (e.g., ET sensor or sonar) plugged into analog port 3, \texttt{analog10(3)}; will produce the current value of the sensor.
- An analog sensor’s value will be in the range from 0 to 1023.
- The CBC can change each analog port to be either analog (pull up resistor) or floating analog (no pull up).
  - \texttt{set_each_analog_state(0,0,0,1,0,0,0,0,0)}; sets port 3 to analog float and the rest analog.
Optical Rangefinder “ET”

- Floating analog sensor
- Connect to ports 0-7
- Access with library function `analog10(port#)`
  - You can also use `analog(port#)` for lower resolution
- Low values (0) indicate large distance
- High values indicate distance approaching ~4 inches
- Range is 4-30 inches. Result is approximately $1/d^2$. Objects closer than 4 inches will produce values indistinguishable from objects farther away
Optical Rangefinder

Position Sensing Device (PSD)

Lens

Focused IR Beam

(high value)

(low value)

(low value)
Ultrasonic Rangefinder (Sonar)

- Timed floating analog sensor.
- Connect to ports 0-7
- Access with function `analog(port#)`
  - Assumes `port#` has been set to floating!
- Returned value is distance in inches (not fractions of inches) to closest object in field of view
- Range is approximately 10-250 inches
- **Important:** When first powered up, make sure there is nothing within 24 inches (60cm) so sonar can self calibrate.
Ultrasonic Sensors

- Puts out a short burst of high frequency sound
- Listens for the echo
- Speed of sound is \(~300\text{mm/ms}\)
- `analog(port#)` times the echo, divides by two and multiplies by speed of sound
- The sonar field of view is an approximately 30° cone
Sample code

• Plug an ET sensor in port 1 and a sonar sensor in port 3

```c
int main() {
    set_each_analog_state(0,1,0,1,0,0,0,0,0);
    while(a_button()==0) {
        printf("ET: %d ",analog10(1));
        printf("Sonar: %d inches\n",analog(3));
        sleep(1);
    }
}
```
Sensor and Motor Manual

• For further detail about sensors, consult the Sensor and Motor Manual on the course web site.
Thursday’s Lab

• Backing up and turning after hitting an obstacle with the bumper (touch sensors)
• Line following (IR reflectance sensors)