G54ARS
Autonomous Robotic Systems
Lecture 2

Last week…

- What is an autonomous robot?
- Foundations of robotics
  - agents, simulation and intelligence
  - robots and their components
  - tasks in robotics & general purpose robots?
  - environments (structured, unstructured, and partially structured)
- Architectures and Behaviours
  - control models
    - the *sense-think-act* control cycle
    - model-based controllers
  - reactive robotics
    - behaviour-based controllers
    - other approaches, hybrid controllers & learning robots

Kahoot!

https://kahoot.it/

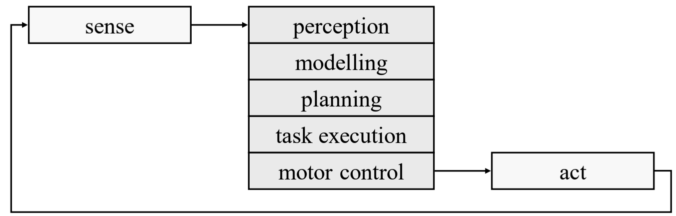
This week - Overview

- Background
  - the behaviour based approach revisited
  - behaviour arbitration
- Subsumption architecture
  - Brooks' assumptions about mobile robot design
  - levels of competence
    - example competences
  - layers of control
    - subsumption
  - structure of layers
  - Extensions / finite state machines
- Robot Hardware
  - Sensors
    - Sensor Characteristics
    - Sensor Types and categories
  - Actuators
    - Effectors and actuators
    - DC motors and Servo Motors
    - DC motors – how they work
    - Degrees of Freedom
- G54ARS Assignment

Background

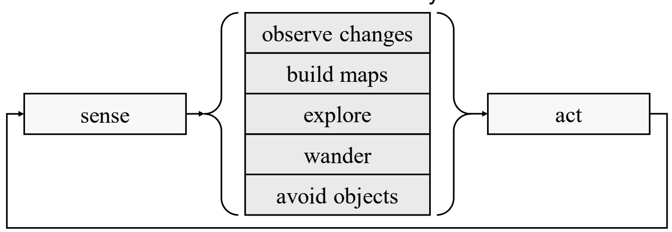
Model-based Approach to Control

- Note the sequential processing / flow of information



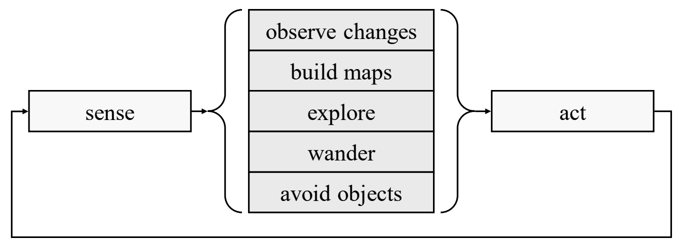
Recall - Behaviours

- The control system is broken down into a vertical stack of horizontal modules, or *behaviours*, that run in parallel
  – each behaviour has direct access to sensor readings and can control the robot's motors directly



Recall - Behaviours

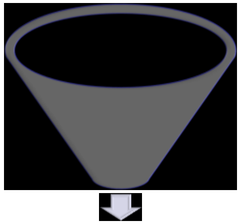
- Behaviour-based architectures enable multiple behaviours in parallel.
- **BUT :** **– how do we decide which behaviour to execute?**
  – how do we coordinate multiple behaviours?



Behaviours Coordination

- There are two main approaches to behaviour coordination.

Behaviour Fusion



Fused Output

Behaviour Arbitration

Behaviour Coordination Continued

- **Behaviour Fusion:**
  - Combination of multiple behaviours into one single output behaviour.
  - Based on some form of aggregation (e.g., sum of motor outputs from all behaviours)
  - Challenge: a suitable aggregation method needs to be defined AND all behaviours need to be suitable for combination (e.g., same outputs)

Behaviour Coordination Continued

- **Behaviour Arbitration:**
  - Selection of ONE behaviour from the pool of behaviours as overall output behaviour.
  - Also referred to as "competitive behaviour coordination" as behaviours can be said to compete for the output.
  - Two main subtypes:
    - Fixed priority hierarchy (priorities are known/fixed at run-time)
    - Dynamic hierarchy (priorities change at run-time)

Behaviour Coordination Continued

- Behaviour Fusion or Behaviour Arbitration?
  - Both strategies have their respective advantages and disadvantages.
  - Behaviour arbitration excels at:
    - Complexity management
    - Simple design
    - Simple debugging
    - Good robustness
- In the labs we use Brook's subsumption architecture which is an example of behaviour arbitration using fixed behaviour priorities.

Subsumption Architecture

A Robust Layered Control System for a Mobile Robot

Rodney A. Brooks

IEEE Journal of Robotics and Automation
Vol. RA-2, No. 1, March 1986
(http://ieeexplore.ieee.org/document/1087032/)
→ Remember you can access articles when not at the university by logging in via Shibboleth and your normal username/password

Brooks' Assumptions - 1

- Complex and useful behaviour need not necessarily be a product of an extremely complex control system (remember Braitenberg vehicles)
- Things should be simple
  - if a single module is getting too big: rethink the design
  - unstable or ill-conditioned solutions are not good
- Ability to wander in real environments (populated by other agents, e.g., humans) is crucial
- The real world is three dimensional
  - the robot must model its environment in 3D
- Absolute coordinate systems are a source of error
  - relational maps are more useful to a mobile robot (think landmarks)

Brooks' Assumptions - 2

- The real world is not constructed of exact shapes
  - models may be useful, but should not be used alone
- Sonar data, while easy to collect, does not lead to rich descriptions of the world
  - visual data is much better
  - But sonar data may be useful for fast responses (e.g., Obstacle Avoidance)
- For robustness, the robot must be able to function when one or more of its sensors fail(s) or start(s) giving erroneous readings
  - recovery should be quick (self-calibrating)
- Robots should be autonomous and self-sustaining (able to survive weeks without human assistance)

Levels of Competence

- Autonomous Robotics is a complex challenge: "Divide et Impera":
  - Any (real) robotics problem must be broken down (decomposed) into a number of sub-problems
- Brooks chose to decompose problems vertically
  - the problem is 'sliced' as a vertical stack on the basis of the desired external manifestations of the control system
- A number of *levels of competence* are defined
  - an informal specification of a desired class of behaviours for a robot over all environments it will encounter
- A higher level of competence implies a <u>more specific desired class of behaviours</u>

Example Competences

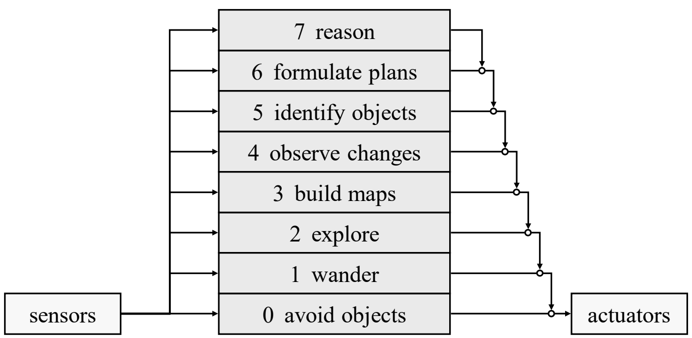
7	reason about the behaviour of objects in the world and modify plans accordingly
6	formulate and execute plans that involve changing the state of the world in some desirable way
5	reason about the world in terms of identifiable objects and perform tasks related to certain objects
4	notice (observe) changes in the "static" environment
3	build a map of the environment and plan routes from one place to another
2	explore the world by seeing places in the distance that look reachable and heading for them
1	wander aimlessly around without hitting things
0	avoid contact with (stationary or moving) obstacles

notes
each level of competence includes as a "subset" each earlier level of competence
higher levels of competence provide additional constraints on the lower levels

Layers of Control

- The key idea of levels of competence is that layers of a control system can be built to correspond to each level
  - a control system can be built to achieve level zero competence (avoid obstacles)
    - it is programmed, debugged and then fixed in operation
  - another layer is then added (first-level control)
    - it is able to examine data from the level-0 system
    - it is able to inject output, to suppress the level-0 output
    - level-0 is unaware of the suppression and continues to run
- This is called *Brooks' subsumption architecture*

Subsumption



Control is layered, with higher level layers subsuming the roles of lower level layers **when they wish to take control**. The system can be partitioned at any level, and the layers below form a complete operational control system.

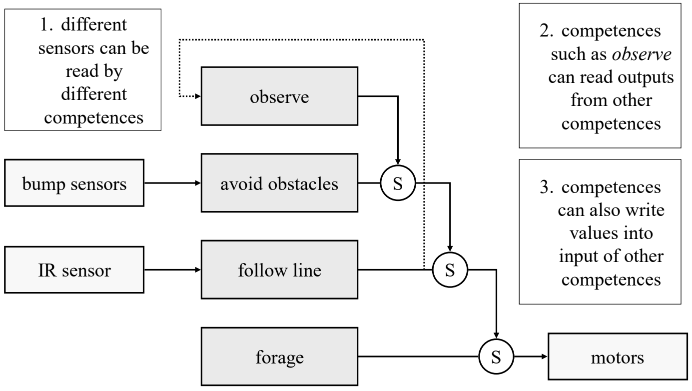
Subsumption Architecture Properties

- Multiple Goals
- Multiple Sensors
- Robustness
- Extensibility

Structure of Layers

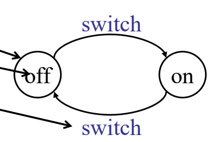
- How is each level built?
  - does each have to be structured in horizontal manner?
- This is true to a certain extent, but the key difference is that not all desired perceptions need to be processed by each competence
  - different decompositions can be used for different sensor-set task-set pairs
- Layers can be built on a set of small processors
  - each one sends (small) messages to others
  - no message acknowledgement is required
  - there is no need for central control (or synchronisation)

Extensions



Finite State Machines

- In Brooks' original work, each module was implemented as a finite state machine
  - augmented with some instance variables
- A finite state machine has
  - a set of states
    - a start state; (one or more stop states)
  - a set of symbols
  - a set of transitions
- For example
  - an FSM for a light switch



- An FSM can be implemented in C
  - using a *state* variable and **switch** statement

Robot Hardware

Sensors

- Sensors are physical devices that measure physical quantities (e.g., light, distance, etc.)
- Some examples:

Sensing Technology:	**Physical Quanitity:**
• Bump sensors, switches	- Contact
• Ultrasound, Infra Red, Laser Range Finder	- Distance
• Photocells, Cameras	- Light Level
• Thermal, Infra Red	- Temperature
• Microphones	- Sound Level

Sensors & Uncertainty

- As physical devices operating in the real world, sensors are subject to noise and errors.
- *Uncertainty* refers to the fact that a robot cannot be certain about the properties of the environment or even its own state in respect to this environment.
- There are many sources of uncertainty in robotics, including:
  - Sensor noise and errors, sensor limitations (e.g., max distance)
  - Effector and actuator noise (e.g., a bent axle)
  - Changing/Dynamic environments

Sensors & Complexity

- Sensors vary in the amount of information they produce.
  - A bump sensors produces one bit of information (on/off). *(By the way: Bit comes from Binary digit)*
  - A camera produces a huge amount of information. E.g., consider outdated VGA (640x480 pixels) which results in 307200 pixels. We are not talking about color yet… *(A phone camera like on the iPhone 8 has 12 million pixels…)*
- More information is not always a good thing as it results in higher processing requirements.
- Signal-to-Symbol problem:
  - symbols (e.g., "Tom") are useful (e.g., "find Tom").
  - But – sensors provide data, not symbols
  - Translation from data to symbols is HARD and requires computation (e.g., how do you know it is Tom?)
  - Thus often symbols are not used (remember reactive robotics?)

Sensor Exercise

- How would you detect Tom?
  - (Note: Cameras are obvious but highly complex, so try other means)
  - Some approaches: temperature, movement (what moves is alive, well, unless it's a robot ☺), color (clothes, skin), distance,…
- These are examples – what is important is that there are many ways to achieve the task and they all have their specific combination of advantages/drawbacks.

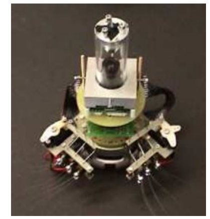
Sensor Types

- Active Sensors
  - Generate an active signal and measure the interaction of this signal with the environment.
  - E.g.,: active light sensor, others?

- Passive Sensors
  - Measure a physical property of the environment
  - E.g.: thermal sensor, others?
- Let's look at some categories of both types…

Switches

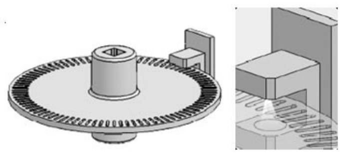
- Switches are either open or closed.
  - We noted – 1 bit of information.
- Many types of switches exist on robotics:
  - Bump Switch
  - Limit Sensors
  - Whiskers
- Switches can be used to gather a wealth of information, e.g.,:
  - Shaft/Position Encoders



Source: http://weburbanist.com

Shaft/Position Encoders

- Shaft Encoders detect the rotation of a shaft by counting the number of times an optical switch is closed/opened.
- Can Shaft encoders be used to measure direction of rotation?



Source: http://www.vexforum.com

Light Sensors

- Light sensors are based on photocells.
  - Photocells convert light into a voltage.
  - The voltage (analogue) is converted to digital by an A2D converter.
- Many different types of light sensors exist, measuring different properties of light:
  - Light intensity (e.g., as in the Lego case)
  - Differential intensity (differences in light)
  - Break-beam sensors
- Light sensors can either be active or inactive
  - Why is calibration often required when light sensors are used?
  - How can light sensors be calibrated?

Distance Sensors

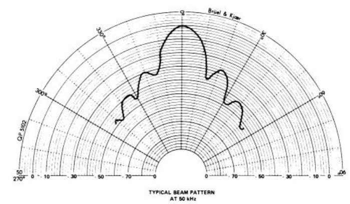
- Most distance sensors work by measuring Time of Flight (ToF) of the signal travelling at speed *S*.

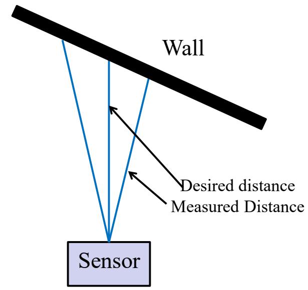
$$Distance = (ToF * S)/2$$

- Ultrasonic Sensor Example: Sound travels at 340.29 m/s (at about 15 degrees Celsius, etc.)
  - If ToF = 0.005s, Distance = $(0.005 * 340.29)/2 \approxeq 0.85$m
- Other distance sensors include:
  - Laser Range Finder, e.g., by SICK (based on ToF)
  - Infra-red distance sensors, e.g.: Sharp GP2Y0A21 (based on IR reflection)

Ultrasonic Sensors in Detail

- Ultrasonic Sensor or Sonar (*originally an acronym for SOund Navigation And Ranging*)
- Typical Beam Pattern of a sonar sensor
- Challenges?





Actuators
Less sensing, more doing!

Actuators

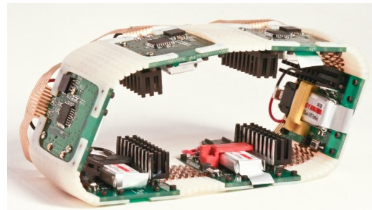
- Some Vocabulary:
  - **Effector**:
    - any device that has an effect on the environment. E.g.: grippers, wheels, arms, etc.
  - **Actuators**:
    - from *to actuate*, i.e. to put into motion. The mechanism that drives an effector. E.g.: DC motors, hydraulics, etc.
  - **Passive Actuation**:
    - energy for the actuator/effector is delivered through interaction of the effector with the environment (E.g., a helium-filled balloon)
  - **Active Actuation**:
    - energy for effector is actively generated by the actuator (E.g., a DC motor)

Passive Actuation Example

- 4-legged passive-dynamic walker:
- http://www.youtube.com/watch?v=rhu2xNIpgDE

Active Actuators

- Electric Motors
- Hydraulics
- Pneumatics
- Shape memory polymers (SMPs)



MIT Shape-memory alloy robot. Source: http://web.mit.edu/newsoffice/2010/actuators-slideshow-0406.html

A closer look at motors

- **DC (direct current) motors** are the most common type of actuator in robotics
  - DC motors produce a continuous rotational motion (useful for example in a fan).
  - In robotics, a more controlled motion is generally desirable and servo-motors are employed.
- **Servo motors** are used to move the drive-shaft of the motor to a specific position.
  - Servo motors are DC motors with additional components (gears, a position/shaft encoder (i.e. a sensor!) and electronics to control the motor)
  - Servo motors allow accurate position control – but it makes the actuators also very *stiff*.

DC Motors – How they work

- Motor speed ω is proportional to induced voltage (energy carried by charge) *V*:

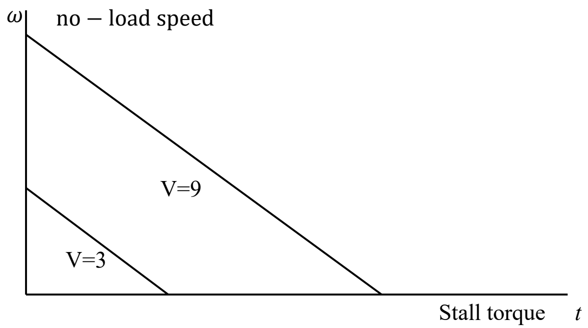
$$\omega = k_V * V$$

- Motor torque t is proportional to applied current (rate of flow of charge) *I*:

$$t = k_I * I$$

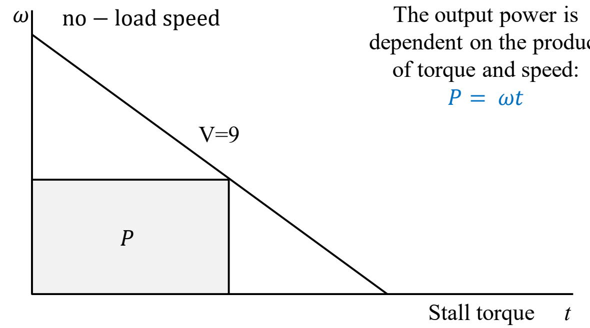
- Motors have a maximum speed (the no-load speed) and a maximum torque (the stall torque)
- DC motors have commonly high speed (9000+ RPM) but low torque.

Speed – Torque Relationship



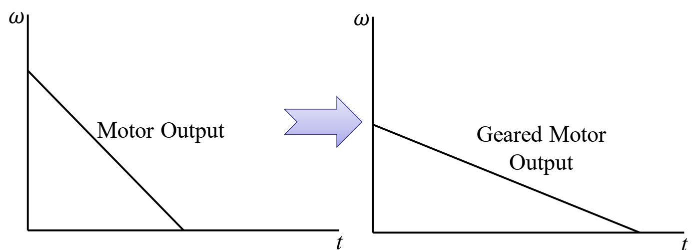
Speed – Torque Relationship

The output power is dependent on the product of torque and speed:
$P = \omega t$



Motor Gearing

- In robotics we generally need low speed and high torque, i.e. to make use of DC motors we need *gearing* to alter the relationship between torque and speed:



Gearing – Basics

- The Linear Speed v at the edge of a gear with radius r is: $v = \omega r$
- The Force F at the edge of a gear with radius r is: $F = t/r$
- Two meshed gears have the **SAME** linear speed, i.e. for Gears 1 and 2, we have $v_1 = v_2$

- What is the output speed ω of gear 2, knowing the output speed of gear 1?
  $$\omega_2 = (r_1/r_2)\omega_1$$
- What is the output torque t of gear 2 knowing the output torque of gear 1?
  $$t_2 = (r_2/r_1)t_1$$
- $\boldsymbol{r_2/r_1}$ **is the Gear Ratio**

Gear 1 Gear 2

Source: Wikipedia

Gearing continued

- Using gears, different combinations of speed and torque can be achieved using the same motor.
- Gearing up, for example:
  - $r_1 = 2, r_2 = 1$
  - Ratio: 1:2
  - Halves the torque, doubles the speed
- Gearing down, for example:
  - $r_1 = 1, r_2 = 2$
  - Ratio: 2:1
  - Halves the speed, doubles the torque
- Many different types of gears, see for example: http://en.wikipedia.org/wiki/Gear

Note: "Gearing up" – Input gear has more teeth than output gear

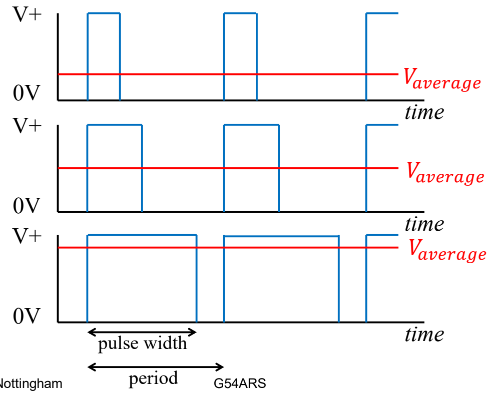
"Gearing down" – the reverse is true.

DC Motor Control

- The speed of DC motors is generally controlled using PWM (Pulse Width Modulation)
- The direction of rotation is controlled using an H-bridge (implemented in electronics)

PWM

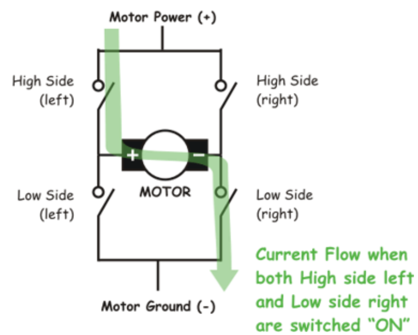
- $\omega = k_V V_{average}$



H-bridge

- See source: http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/index.html

High Side Left	High Side Right	Lower Left	Lower Right	Quadrant Description
On	Off	Off	On	Motor goes Clockwise
Off	On	On	Off	Motor goes Counter-clockwise
On	On	Off	Off	Motor "brakes" and decelerates
Off	Off	On	On	Motor "brakes" and decelerates



We have motion!
Which way do we apply it?

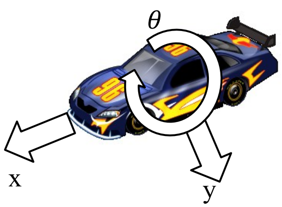
Degrees of Freedom (DOF)

- DOF = ?
  - In mechanical terms, the DOF is the number of variables that determine the state of a physical system.
  - Informally, think about in what directions a system can move.
- In 3D space an object has generally six DOFs, split into two groups:
  - Translational DOFs (*x, y, z*)
  - Rotational DOFs (*roll, yaw, pitch*)
- Effectors can have 1 or more DOFs, e.g., a motor (0 DOFs makes no sense…)

DOFs Examples

- A car has three DOFs (x, y and orientation (theta))
  - two Controllable DOFs (CDOFs)
  - 1 uncontrollable DOF (UDOF))
- A helicopter has 6 controllable DOFs

I know this is not a helicopter!

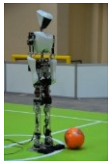


http://en.wikipedia.org/wiki/Degrees_of_freedom_%28mechanics%29

DOFs continued

- If the total DOFs of a system (TDOF) is smaller than the controllable DOFs of a system, this can make control complicated (think about parallel-parking a car).
- Three main possibilities are differentiated:
  - Holonomic: CDOF = TDOF
    - e.g., a helicopter
  - Nonholonomic: CDOF < TDOF
    - e.g., a car
  - Redundant: CDOF > TDOF
    - e.g., the human arm:
      - seven degrees of freedom (shoulder: 3, elbow: 1, wrist: 3)
      - But in 3D space there are at most 6 DOFs! → ability to move arm/its parts in different ways to the same position, i.e. redundancy.

Putting it all together…



- Virginia Tech's "Charli-2" humanoid robot
  - # of Servo Motors?
  - # of DOFs?
  - Sensors?
  - http://www.youtube.com/watch?v=kmeJvkN4ntl
- Boston Dynamics' Atlas
  - E.g.: https://www.youtube.com/watch?v=6zpuHr7t8xl

Assignment

- Assignment - see on Moodle:
  - http://moodle.nottingham.ac.uk/pluginfile.php/3495658/mod_resource/content/11/G54ARS%20Assignment%2017-18.pdf

Summary

- Background
  - The behaviour based approach revisited
  - Behaviour arbitration
- Subsumption architecture
  - Brooks' assumptions about mobile robot design
  - levels of competence
  - layers of control
  - structure of layers
  - extensions; finite state machines
- Sensors
  - Sensor Characteristics
  - Sensor Types and categories
- Actuators
  - Effectors and actuators
  - DC motors and Servo Motors
  - DC motors – how they work
  - Degrees of Freedom
- Next week: PID Control & DARPA Grand Challenge