Acquired Intelligence & Adaptive Behaviour Simulated Robots
Lab Session 5

Goals:
1. To investigate a simple low-inertia wheeled robot in simulation.
2. To implement a number of elementary Braitenberg Vehicle architectures.
3. To investigate the interaction of multiple vehicles

Background:
In his 1984 book 'Vehicles: Experiments in Synthetic Psychology' Valentino Braitenberg proposes a number of hypothetical wheeled robots. He argues that even though the robots themselves were extremely simple, when embedded within a real environment, they would demonstrate quite complex, seemingly intelligent behaviours. Four of Braitenberg's Vehicles are of particular interest. Each has two ambient light sensors and two driven wheels. The control architectures of these vehicles are considered 'representation free' in that sensors are connected to motors (at the wheels) by direct innervation which could either be excitatory (positive) or inhibitory (negative). In the case of negative connections, motors were considered to turn without stimulation (i.e. via baseline innervation, a so-called 'bias'). Beyond this, connections could either be contra-lateral (crossing, left sensor to right motor and vice-versa) or ipsilateral (same-side). The four possible vehicles using this architecture are thus:

1. Aggressor: Contra-lateral, positive connections.
2. Coward: Ipsilateral, positive connections.
3. Lover: Ipsilateral, negative connections
4. Explorer: Contra-lateral, negative connections.

Task1:
MATLAB code for the class is available on Study Direct. It implements a simple low inertia wheeled robot in a simple 2D environment. In this session we will attempt to implement each of Braitenberg's vehicles (as described above) to demonstrate both; their behaviour and, the extent to which the simple simulation paradigm allows for the behaviours predicted by Braitenberg. You will need two files: init_sim.m and run_sim.m. Copy these into your home directory and open them both in MATLAB.

1. To begin, execute init_sim. This will set up the environment and draw our simple agent to the screen.
2. Now execute run_sim. This will integrate the simulation over 100ms and display the path taken by the agent. Note that you may repeat this stage to integrate further steps of 100ms.

Repeat the above stages a few times and get a feel for what happens. Now look at the code. Can you see why the agent behaves as it does? You may alter parameters and observe the effects. Now we're rolling (so to speak), alter the code to exemplify each of robots described above. Remember to keep track of your results. You can save the diagrams drawn by MATLAB.

Task 2:
Consider the following problems and and try and implement a solution to each:

a. Just one light source is boring.
We find that achieving photo-taxis is quite simple and soon enough we've implemented an agent which stops next to our light source (the 'lover'). On the other hand, we find implementing an 'aggressor' that repeatedly returns to the light source harder than we imagined and the best we can get is an agent that runs right over the light source once, only to end up sat in the dark too far away from the light to be attracted back towards it. In either case, we might find it helpful to introduce multiple light sources. Can you increase the complexity shown by these agents by adding more lights?

b. The agent often disappears off one or other side of the visible area.
If the environment is required to be infinite and non-periodic this might be exactly what we want, however most of the time having the agent disappear off 'into space' is a problem. Can you think of a way around this? Consider both the position of the agent, the contents of the environment and the way in which the sensors work.Simple patterns or stable configurations often emerge in a static environment. We find our agents are still somewhat boring. Lovers sit dotingly next to their favoured light source and aggressors continually bully theirs, going round and round in stereotypical loops over and over and over ... If this was the real world, something would have to give. The environment often responds to the behaviour of the agent. Specifically, we might consider the light sources as food. The abundance of these resources might then fluctuate, or even run out and have to be replaced. There are many ways that the depletion and replenishment of resource might me modelled in such a simplified environment and this may lead to a continually changing interaction between agent and environment. Can you implement a mechanism for resource depletion and replenishment?

b. Even robots have desires.
Our agent's are becoming ever more sophisticated. What will they think of next? Who knows, but the agents themselves? If you really want to push the limits of simple robot architectures, try implementing an internal 'battery' which runs down with time (or as the agent moves) and is replenished by the light sources in the environment. Have the agent's behaviour modulated by the state of its battery (maybe it will move faster as the battery dies?) and investigate how this affects the apparent complexity of the behaviour. Consider what might happen if the agent had multiple batteries and multiple competing behaviours!