6.836 Embodied Intelligence–Research Assignment 1

Massachusetts Institute of Technology

Due: Thursday, February 20, 2003

This is a **graduate class**, not an undergraduate class. As such, the research assignments are much more open ended; there will not be much hand holding, and there definitely will not be great levels of clarification of precisely what is wanted. It is up to you to figure out something reasonable that gets at the underlying issues, and at the same time do some thinking. Mechanical problem set problem/solution matching is not what we are looking for here. The purpose of these research assignments is to get you thinking about some of the underlying issues. The problems require some thinking but not much writing. I don't expect more than a page for any of them.

Please read the whole assignment before you start. Some of the questions will require knowledge of material covered in lecture 2. However, you can start learning the tools and making progress on the first questions now.

During lecture we looked at exquisitely simple vehicles which Braitenberg used to illustrate concepts such as psychology, cognition, and free will. In this research assignment, we will consider some only slightly more complex vehicles. We will explore issues related to their control in a simulated environment.

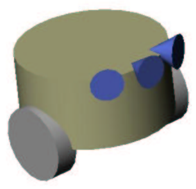
The Simulator.

For your convenience we provide a simple simulated environment with a vehicle and some obstacles in it. It runs on **Matlab** with **Simulink** and can be downloaded as an archive file from the course webpage by following the link from
http://www.ai.mit.edu/courses/6.836/handouts/handouts.html.
The page also has pointers to some Matlab and Simulink resources. To run Matlab on athena, type at the prompt: `athena% add matlab` and then `athena% matlab &`. Start Simulink inside Matlab by typing `simulink` at the prompt.

The archive consists of the following files. `MainR.m` is the main script that runs the simulator, calling `DrawObstacle.m` and `DrawRobot.m`. Other useful files are `detectobs.m` which contains code simulating the robot's sensor readings and `runover.m` which computes whether the robot has crashed into an object. Finally, the file `stestX.mdl` contains the Simulink model of the vehicle's control. We have attached the printout to this research assignment. In the following problems, you will add to that model. We are using the building-blocks world of Simulink specifically in order to get away from sequential programming and let you play with designing modular control systems.

You may, but are not required to, modify the simulator in any way that would augment its realism or functionality. You are also free to choose another simulator or build your own if you like, provided that it has the functionality required for this assignment. State any assumptions you are making about the virtual world, e.g., friction, obstacle and sensor characteristics, etc.

Problem 1. Consider a Braitenberg-like vehicle with two actuators and an array of three range sensors in front. The sensors point 0°, −30° and 30° from straight ahead. They each return the range to the nearest obstacle in that direction (up to some maximum value). In our simulator, the sensors have a range of ±15° from their orientation and a maximum sensing distance of 100 pixels.



These settings are of course arbitrary and you may modify them in interesting ways. The motors are actuated through velocity control. In this problem, we would like you to build a two-output (one for each actuator) neural network controller for this vehicle to achieve the two behaviors described below. Note that your network should contain linear sum neurons with either linear or nonlinear transfer functions. There is a pointer to a neural network tutorial online linked from the course webpage, in case you need to refresh your memory.

Note: While Simulink provides neural networks building blocks, you may be better off writing your own code in Matlab and calling that function from a `MATLAB Fcn` *block, or using a simple* `Fcn` *block.*

1a (1 point) The vehicle wanders around randomly, perhaps bumping into things. You can assume that the controller has access to a wire that produces a random number periodically.

1b (2 points) Now make your network learn obstacle avoidance for the vehicle. Use the sensors provided. You may do so by backpropagation or any other neural network learning algorithm. It could be easiest to train your network off-line by collecting data from robot movement, writing a short script to label instances, running your algorithm on that data and finally plugging the resulting weights back into the simulator.

Problem 2. (2 points) Comment on the complexity of your solution to problem **1**, including the number of trials it took your vehicle to learn its control function. Conduct some robustness tests. You may write your own test cases, add more obstacles, etc. Hand in your thoughts and a description of results.

Problem 3. Consider the same simulated Braitenberg-like vehicle in the same virtual environment. You will now build a network of augmented finite state machines (AFSMs from Chapter 2, or the lecture of February 14th) to control the vehicle. Again, provide two outputs: one for each actuator.

3a (1 point) As in **1a**, build an AFSM network to make the vehicle wander around randomly, perhaps bumping into things. You can assume that AFSMs have access to a wire that produces a new random number periodically. Fully specify each AFSM in Simulink.

3b (2 points) Now add an obstacle avoidance layer to the network in **3a**. Use the sensors provided. The new layer should only interact with the existing random-move layer through suppression and inhibition. Note that there is of course no learning involved.

Problem 4. (2 points) Comment on the complexity of your solution to problem **3**. Run on the same test cases as in **2**. Which approach do you think is better: AFSMs or neural networks? Why? Would your answer change if the two approaches were applied to a different kind of control problem?

Note: There may be more than one correct answer - or no correct answer - to the open-ended questions on this and other research assignments. We are looking for an understanding of the issues and trade-offs of the two approaches.

Project Ideas

It's never too early to start thinking about a term project for this class. If you wish to do a project that expands on the concepts encountered in this problem set, here are a couple of directions you could follow. For example, you could build a subsumption controller for a mobile robot performing a task that is substantially more complicated than the one in this research assignment. Since you may not have available (and we can't provide) a physical robotic platform for such a project, you could consider using a realistic physical simulator instead. You probably don't want to spend your time writing your own, but if you use somebody else's, note that it should be substantially more realistic than ours.

Another project idea could be to test several robotic control approaches (such as for instance, subsumption, a rule-based system, a machine learning system) on a set of tasks that may not be as complicated but would highlight the uses and drawbacks of these different frameworks.