Introduction to Vision and Robotics

Second Assessed Practical

Autonomous Navigation and Task Execution by a Team of Robots[1]

1 Introduction

The aim of this practical is to control a small group of simulated robots such that a coordinated behaviour is produced. You will work in pairs of students and must submit one joint report (only one student of each pair needs to submit the report). The report must be accompanied by a short explanation of how the work was shared and how you think the (joint) mark should be distributed. You can choose the partner yourself and you will have to choose a different partner from the first practical assignment. Please contact michael.herrmann@ed.ac.uk immediately if you do not have a partner. You will be paired up with the nearest student who has sent the same request.

The work will be done using the Webots robot simulator. Webots runs at any DICE computer if it is a 64bit machine. The computers in the IVR lab (AT 3.01) are all 64bit. The current (previous) version is Webots 6.2 and runs actually on all machine, however, the new version (Webots 6.4.x) will run only on the 64bit machines. License numbers are limited, so, please, never stay connected to Webots when you are not using it. The course page http://www.inf.ed.ac.uk/teaching/courses/ivr/ gives a link to a project directory that can be used in Webots (see below).

2 The Task

The task involves cooperative task execution in autonomous robots. Each robot should use the same control program, but, due to the different sensory information each robot receives during the run, the behaviour of the robots can be different. If all goes well the robots will be able to (co-) operate in their environment and use the sensory information to solve their task.

The task consists in programming a robotic vacuum cleaner which is supposed to clean off virtual dust in a virtual flat. Robots that mow a lawn, harvest a field, survey an area or paint a large object could use a similar strategy. The robots can use distance and light sensors as well as an on-board camera. An important goal is that the robots move swiftly except where high precision is required or when it needs to wait for another robot. Try to optimize the trade-off between speed, accuracy and cooperativity in a way that is appropriate for different phases of the task execution.

a) Basic behaviours (solve all)

i. The robot should move through a room such that its path covers the floor area fully and efficiently. It should be able to avoid obstacles such as chairs or tables (the robot is not required to clean underneath the obstacles, but it can do this e.g. for tables). Technically, the path of the robot changes the texture of the floor which is clearly visible to a human, but it is difficult to detect this with the robot's camera.

ii. The robot should keep track of its position relative to a defined starting position by odometry. It should not directly use the x and y coordinates of the simulator, but can use the built-in odometry functions. The robot can make use of a map of the room

1 Given out on 3/11/2011

such that it is able to correct its position as soon as it encounters a wall. However, the map would not contain the positions of any movable obstacles.

iii. The robot should be able to perform “homing”, i.e. to return to a base position (e.g. the starting position after the job is done.

b) Coordinated Behaviour (solve all)

i. Two robots can solve the task in half the time if each of them cleans half of the room. Having solved task a) already, this would be a trivial task. Without implementing this solution, you may still discuss in the report about how the robots may find a unique way to split the floor into halves and how to decide which robot is cleaning which part. Also in the following, the robots may negotiate their respective roles in solving the task (roles may be: “near wall” and “near robot” or simply “left” and “right”).

ii. Implement instead a solution to the more interesting problem of having the two robots do the cleaning as a formation. The robots should move side-by-side on parallel paths whenever the environment permits. If one of the robots encounters an obstacle the other one should slow down or wait until the first one has circumvented the obstacle and catches up, but it should take care at the same time that it does not form itself an obstacle for the circumventive robot. The robot that eventually has successfully avoided the obstacle should then speed up to align again with the partner. What would happen if a robot while avoiding the other one, encounters itself an obstacle? In some cases it may be useful if the robots change roles (i.e. the “left” robot becomes the “right” robot and vice versa)

iii. Analyse how much faster two robots can clean the room as compared to a single robot. Use different configurations of obstacles. Would you describe your observation as “Many hands make light work” or rather as “Too many cooks spoil the broth”?

c) Advanced tasks (choose one)

i. A single robot is supposed to be too weak to move any objects out of its way. However, if two robots cooperate they will be able to do this. The robots may thus simplify the cleaning task by the (practically possibly less desirable) ability to cooperatively rearrange your furniture.

ii. Implement a vision based version of b), i.e. without using information on the other robot's position.

iii. Same as b) but with three or more robots.

iv. Design your own task of coordinating robotic behaviour using the present set-up. Discuss with the instructors whether your approach is realisable and acceptable: While almost all tasks will be acceptable most of them may require a lot of work to actually work.

Hints

- Files containing the environment in Webots and a simple controller can be found in this archive http://www.inf.ed.ac.uk/teaching/courses/ivr/practicals/ivr2011a2.tgz, see also the link at the course page. Unpack the archive using “tar xvfoz ivr2011a2.tgz” and go to the directory ivr2011a2 and start webots. Then load the world file vaccum6.2.wbt (in directory “worlds”).
- The robot is controller by a simple controller “vacuum6.2”. In the file “vacuum6.2.c” or “vacuum6.2.m” you can make changes to achieve a more useful robot behaviour.

You will need to compile and to build the controller (buttons above the right window) before is can be used by the robot.

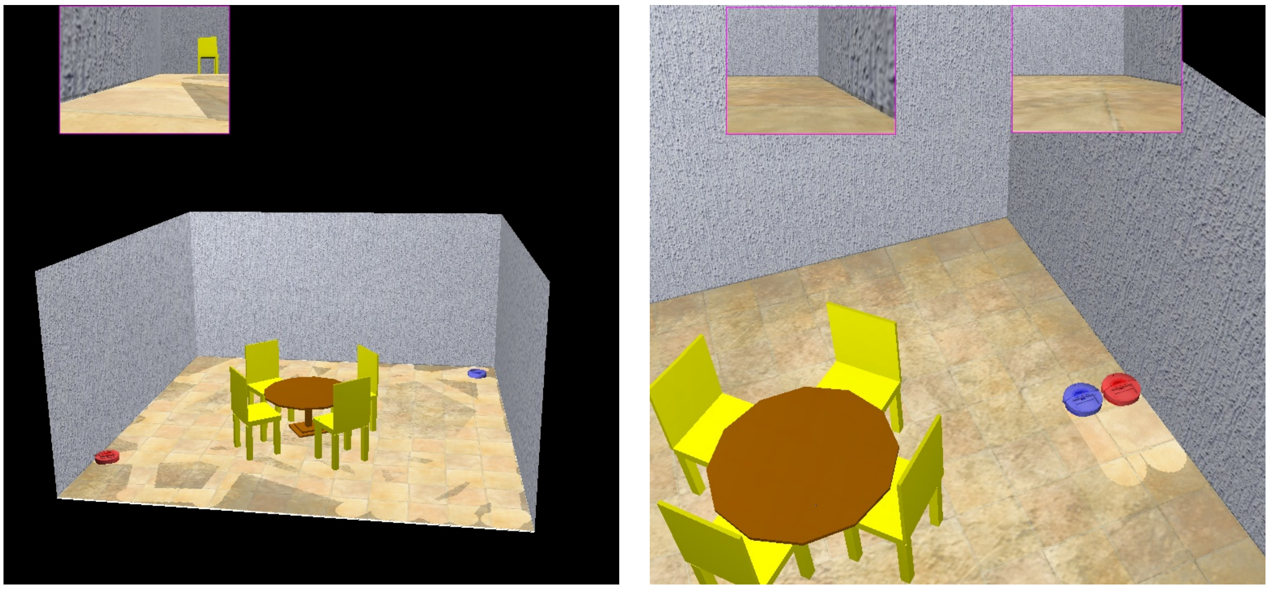

Figure: (left) Two robots cleaning the floor following random trajectories. The inset shows the view from one of the robots' camera. (right) Two robots setting out for coordinated cleaning action. The views from both cameras are shown on top.

- Using camera images: Taking a snapshot is possible by the command "wb_camera_get_image(camera)". You may find related command in the Webots documentation. You may also like to change the resolution of the camera, the position where it is shown in the world view or the orientation of the camera (e.g. to observe the other robot rather than the path).
- How to get more than one robot?
    - Simply use the copy-and-past buttons for the "differentialwheels" entry in the world tree on left of the Webots window.
    - Change name of the copy from VACU1 to VACU2 (or VACU3). Then move new robot to a different place than the first one.
    - If you want to change the color of the new robot then find the entry "create_red" in the VACU2 subtree and change it into "create_blue", "create_green", or "create".
    - You will need to save the world and to reload it (buttons above the center window) in order to use the new robot.
- The two robots should be exact copies of each other and should use the same control program. Due to the different sensory information each robot receives during the run, the behaviour of the robots can be different. The program can contain different parts which are processed only if the robot assumes a particular role.
- The control signals for the robots are to be determined based on information from the robot's sensors, i.e. only information that would be available to a real robot can be used. The robots can sense each other by distance sensors and on-board cameras but in principle they cannot directly transmit any data messages between each other. Nevertheless, the robots will need to communicate. In order to simplify

this we assume that each robot can obtain the **estimated** coordinates of the other one (see the robot controller file). The robots may also exchange behavioural messages such as moving a bit forward in order to indicate their moving intention or move back and forth to show that they are waiting etc. In principle, they may also communicate using their LEDs.

- In order to analyse the behaviour of the robots you may use the simulator coordinates. These can be obtained by the controller of the “supervisor” (see world tree or the *.wbt file) The robots should not use these values.
- Cleaning is not too interesting a task for the larger part of the runtime. Use the fast-forward button in Webots such that you can focus on the more challenging aspects.
- In order to coordinate the tasks (e.g. moving, homing, and obstacle avoidance) it is recommended to prioritize tasks (this could be implemented by the so-called *subsumption* architecture). In the control program this can be implemented by nested if-statements. Start with a graphical representation of the program structure.
- You should approach the tasks in several stages. Take a sensible approach to the system design: THINK about what components you need; PLAN the tasks to be done, how they will be done, by whom and in what order; IMPLEMENT; and TEST. Your final mark will be based on how well you explain your approach to the task and evaluate the capability of your program. If your control algorithm isn’t reliable, you won’t fail if you can provide a sensible explanation of what you attempted and the limitations and problems in your report.

3 Writing the report

The report should be a concise description of what you did, why, and what happened. The entire report should be no more than 1500 words (excluding appendices). Please stick to this limit, e.g. by moving descriptions of details to the appendix or by using figures. Program listings are not required since you are asked to submit your code separately. It should contain the following sections:

a) Introduction: and overview of the main ideas used in your approach.

b) Methods: Give a functional outline of your code (include your full commented code in an Appendix) and explain how each part of it is meant to work. Where suitable, justify your decisions, e.g. why you used one method rather than another, what you tried that didn’t work as expected, etc.

c) Results: You should provide some actual quantitative data, from repeated trials on how well your algorithm controls the robot. This can include data on tests that you have performed in preparation of the task, trajectories of the robot(s) in the actual task, comparison between different strategies. Try to represent the data graphically. Well documented failure will get more marks than unsupported claims of success (well-documented success will get even more marks!).

d) Discussion: Assess the success of your program, and explain any limitations, problems or improvements you would make.

4 Live demonstration

You will have to demonstrate your program working in Webots with simulated robots.

The demonstration session for this second practical will take place on Friday 25/11/2011 from 10am onwards. Note that the environments where the robots will have to perform its tasks will not be exactly the same as during your preparations. It will have a similar size, the objects will be the same but will be placed in different positions.

5 Submission

Your submission should be a single PDF file and should be submitted electronically by 4pm Thursday 24/11/2011. The command to use for on-line submission is:

```
submit ivr 2 FILENAME
```

where FILENAME is the name of your file, e.g. report.pdf. You are asked to submit your code in a separate zipfile. This assignment is estimated to take 12 hours work per student. You must do this assignment in pairs and assign credit in your final report. Try to distribute the work evenly such that you can contribute what you are good at. The assignment will be marked as follows:

Issue	Percentage
Demo: Single robot behaviour (A: all tasks)	10%
Demo: Robot team behaviour (B: all tasks)	20%
Demo: Robot team behaviour (C: one task)	10%
Report: Program design	20%
Report: Results	20%
Report: Clarity	20%

6 Demonstrators

Vladimir Ivan (s0972326@sms.ed.ac.uk), Simon Smith (artificialsimon@ed.ac.uk), Efstathios Vafeias (E.Vafeias@sms.ed.ac.uk) are the demonstrators for IVR. They will be in the lab at the agreed demonstration sessions. You can also email them when course equipment or software is not working, but not if your software has bugs (that's your problem!). For general questions about this practical please contact Michael Herrmann (michael.herrmann@ed.ac.uk, Tel 651 7177)

7 Plagiarism

This assignment is expected to be in your own words and code. Short quotations with proper, explicit attribution) are allowed, but the bulk of the submission should be your own work. Use proper citation style for all citations, whether traditional paper resources or web-based materials. Before submitting please acknowledge any additional sources of code that you use and ensure that your submission follows the school policy on plagiarism: http://www.inf.ed.ac.uk/teaching/plagiarism.html.