Introduction to Vision and Robotics
Assessed Practical: Visually Guided Robot Navigation

October 22, 2010

1 Introduction

This practical develops a Matlab program that works in both the real and the Webot worlds. It uses images capture by a real or webot camera to:

1. Identify a Khepera robot, the boundaries of a small maze and two obstacles.
2. Navigate the robot through the maze.
3. Accurately park the robot on top of the final target circle.

You will work in pairs and must submit a joint report *but* this report must be accompanied by a short explanation of how the work was shared and how you think the (joint) mark should be distributed. 50:50 is OK if you shared the assignment fairly.

You should be able to normally access the IVR lab (AT 3.01) anytime with your swipe card. You need to get and return the Kheperas from the ITO between 9-5, Monday to Friday, unless you have agreed a special arrangement with them. Use the Kheperas in the IVR lab. The cameras will generally be available in the lab - do not take them elsewhere! There are a limited number of Khepera robots and arenas, so you cannot borrow one for more than 24 consecutive hours.

You can use Matlab and Webots at any time on other DICE machines, but there are license number limits on availability so *never* stay connected to the programs when you are not using them.

This practical will be run in two sessions (weeks 5-7, 8-10), with approximately equal numbers of teams in each session. The assignment will be essentially identical

for the two sessions. (You will have to submit your code, so don't plagiarise a first session group's code if you're in the second session! We'll check the code.)

2 The Task

The overall goal is to develop an algorithm that takes images of a scene and navigates a robot from a start to a goal location. Complicating the matter will be the fact that the images will have some perspective distortion and you will not know the exact camera position from where the images were taken. The practical task involves target recognition and visual servoing and will require use of both the Webots simulator and real camera images with the Khepera robot.

Figure 1a shows the map of the world within which the robot has to navigate. There are small dots in the alcoves where the Khepera starts and finishes. You could be asked to start in either of the locations and you have to navigate to the other. Note the black barriers and walls. The Khepera has to stay in the white region and not cross a black wall. There are 4 large exterior black dots that you should use for doing the homography mapping that corrects for the perspective distortion. You will want to identify these dots and estimate their image centre of mass. There are 2 smaller black dots at the start and finish positions. You can also use these smaller dots as part of the algorithm that identifies the correct dots for estimating the homography. The mazes are printed on A3 paper sheets, which can be found in the lab. There is a fine blue circle around the start and end points. Ignore this.

Figure 1b shows a typical real image of the scene layout, with the Khepera in the scene. Note that there are also obstacles blocking 2 of the 3 routes from the start to the finish positions. There will always be the 2 obstacles, but they could be placed in any 2 of the three routes. Note also that you cannot see the actual centre of the base of the Khepera. You will have to find a way to estimate this position from the observed Khepera. Hint: ignoring minor projective distortions, it lies on the vertical line through the Khepera centre of mass.

Figure 1c shows a typical webot world image of the scene layout, with the Khepera in the scene. Note that the real-world Khepera is bigger than shown here.

What you have to do is:

1. Given a random (but not unreasonable) placement of the webcam, capture an image of the scene.

2. Identify the maze (using background comparison to the scene without the maze).

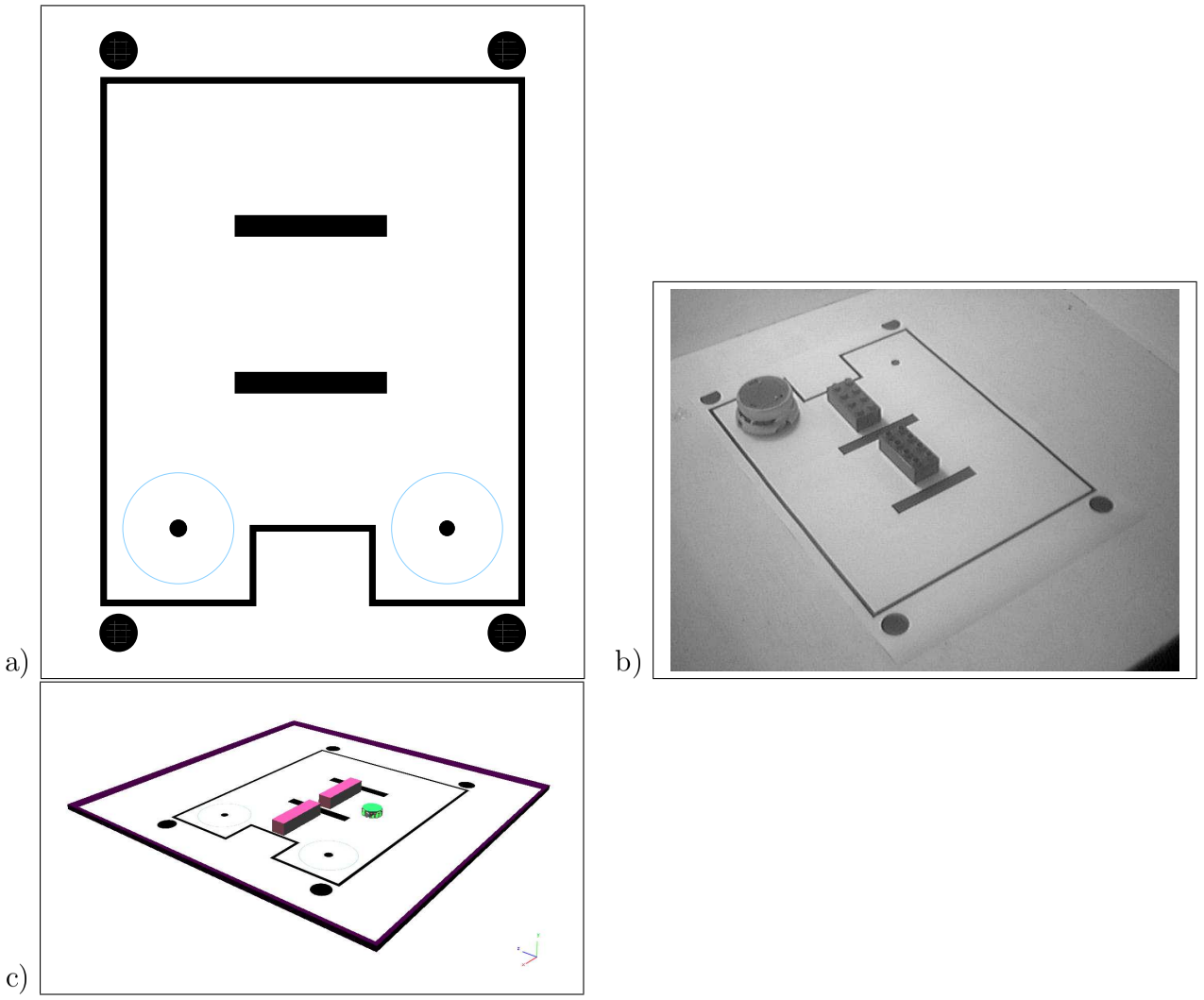


Figure 1: (a) Map as seen from above, (b) Real scene 1 with map and Khepera, (c) Webot scene with map and Khepera

3. Identify the robot and obstacles (using background comparison to the maze without the robot and obstacles).

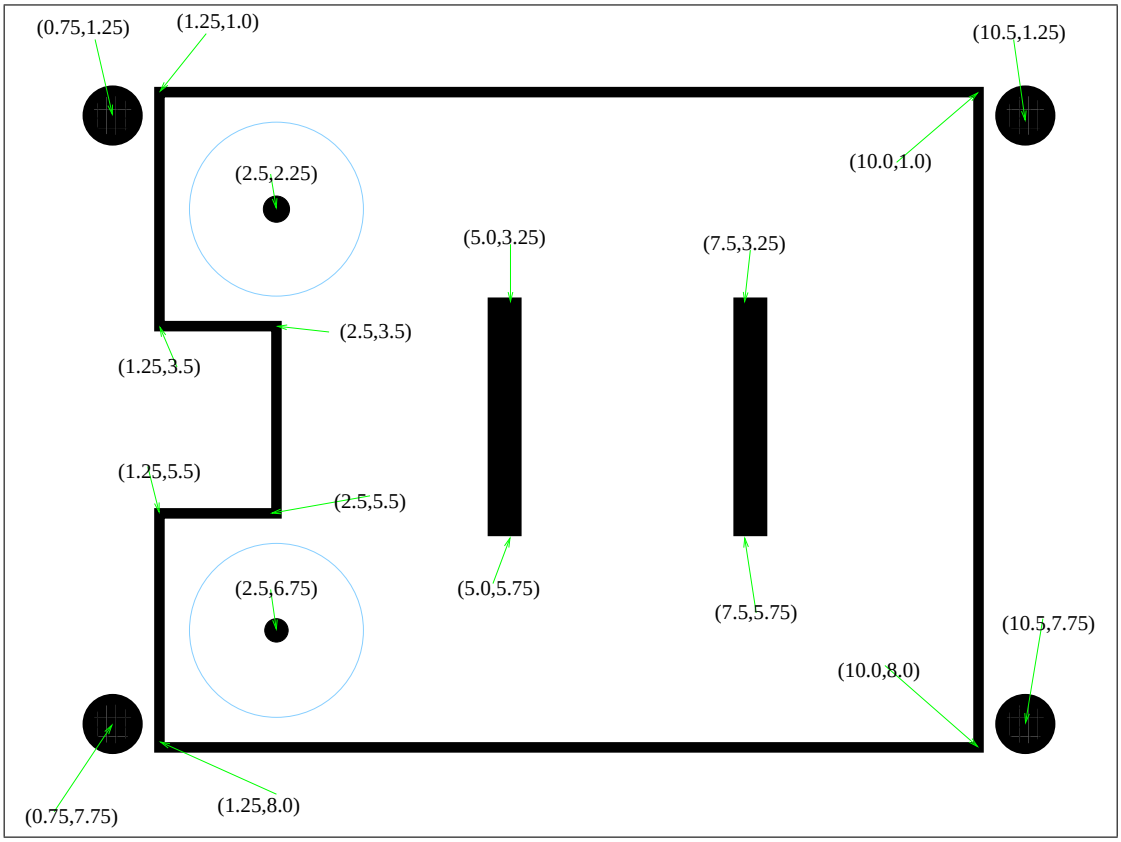
4. Using visual servoing, navigate the robot to move from the start point to the finish point (small black dots) without colliding with the black barriers or the Duplo blocks obstacles.

5. Servo the robot so that the final position is directly over the destination dot and within the faint circle.

You must analyse the real and webot scene images for the practical. You should produce algorithms that are insensitive to small changes in the camera position and orientation. To explore the insensitivity of your algorithm, in webots, move the camera to a new position and take a new image. In the lab, move the paper sheet around and take new webcam images.

Given an image, use colour and lightness to detect all of the circles and robot. Before you capture a real image, you should also capture an image of the background white box without the scene sheet or obstacles. Also capture an image of the scene with just the maze, before the robot and obstacles are added. You can mask away a small amount of the background at the edges of the image if there are highlights or shadows that give your algorithm problems.

When constructing the homography, you will need to know the image and model positions of a number of points. This image shows the model positions of the important points:



As well as the content given in lectures, you might find some of these pages from CVonline (`http://homepages.inf.ed.ac.uk/rbf/CVonline`) helpful:
Homography:

`http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG122`

Homogeneous coordinates:

`http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG25`

Homography transformation:

`http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG1366`

3 Files You Need

You will need these files (which can be downloaded from the IVR webpage):

`take_snap.m`	code that allows image capture by matlab of what's seen by the webot world camera
`addto_local_webots2010.rar`	extended webot controller software for practical

The `addto_local_webots2010.rar` file needs to be unpacked using

`unrar x addto_local_webots2010.rar`. Move the 2 webot world files from the unrar-ed worlds folder to `<YOUR WEBOT FOLDER>/local_webots/worlds`, and move the `scanner` and `tcp_ip_camera` folders from the unrar-ed controllers folder to `<YOUR WEBOT FOLDER>/local_webots/controllers`. `world2010.wbt` is the webots world to open from the pulldown menu.

4 Image Capture Details

Six robot 'arenas' will be set up in the IVR lab containing high white walls. There will also be a set of A3 papers printed with the maze to place in the arenas. Place your webcam so that it can see into the arena. You can use tape to hold the camera in a fixed position on one of the camera stands.

To grab an image in Webots, pull down the `File" -> "Take Screenshot...` option. This will create a PNG image that can be loaded into Matlab. Use the left/middle/right mouse buttons to move the camera into different views of the scene. You'll want multiple views for training your classifiers before using the robot in the scene. You should move the real or webot camera viewpoint so you get a variety of training data for the scenes.

You can also capture a view of the scene from the Matlab to webots interface using the function `take_snap.m`. The resulting file is saved under `/tmp/snapshot`. The overhead camera can be moved from the scene tree and its position/rotation params can be found under `\cam_bot Supervisor \children \Camera`. Initially, the camera is set directly above the arena.

On some machines, the captured screenshot may leave a black box where the pulldown menu was. If this is the case: 1) drag the webot window corner to make the panel larger, 2) use the right mouse button to drag the scene so it is not covered by the pulldown panel or 3) write your program to not use the leftmost portion of the webot images.

5 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 4000 words (excluding appendices). It should contain the following sections:

1. Introduction: an overview of the main ideas used in your approach.

2. Methods: Explain the vision and robotic techniques that you used. Then give a functional outline of how these ideas were implemented and the structure of your code. Include your full, commented, code in an Appendix. Do not print any code supplied from the IVR web pages. 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.

3. Results: You should provide some actual data, from repeated trials (with the camera or sheets in different positions) on how well your algorithm can: 1) identify scene features, 2) identify and maintain a correct trajectory through the maze and 3) park the robot at the goal. Show an example of your results for the real and webot scenes, including marked trajectories of the robot overlaying the ground floor image. Show an image of the scene where you overlay the key points found by the homography: start and end points, 4 control points, 8 grid corners, 4 obstacle corners. Well documented failure will get more marks than unsupported claims of success (well-documented success would be even better!).

4. Discussion: Assess the success of your program with regard to the reported results, and explain any limitations, problems or improvements you would make.

5. Code: the new Matlab code that you developed for this assignment. Do not include code that you downloaded from the course web pages. Any other code that you downloaded should be recorded in the report, but does not need to be included in the appendix.

Your final mark will be based on how well you explain your approach to the task and evaluate the capability of your Matlab program as well as the performance. If your navigation algorithm isn't reliable, you should provide a sensible explanation of what you attempted and the limitations and problems in your report.

6 Live Demonstration

On Friday Nov 5, 10:00-13:00, you will have to demonstrate your program working on the Webot and real scenes. We will be placing the cameras randomly (but not maliciously). You will be allocated to a demonstration time.

Submission

Your submission should be a single PDF file and should be submitted electronically by *10am Friday Nov 5*. The command to use for on-line submission is:

```
submit ai3 ivr cw1 FILENAME
```

where FILENAME is the name of your PDF file.

This assignment is estimated to take 25 hours work. You must do this assignment in pairs and assign credit in your final report. The assignment will be marked as follows:

Issue	Percentage
Program Design	20%
Report Clarity	20%
Experimental Results (in Report)	20%
Live demonstration Results (Webots)	20%
Live demonstration Results (Khepera)	20%

The live demonstration results will be based on a combination of scene feature detection (5%), suitable path planning (5%), collision avoiding navigation (5%) and accuracy when servoing the robot over the final target location (5%).

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`.