IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2015

BEng Honours Degree in Computing Part III
BEng Honours Degree in Electronic and Information Engineering Part III
MEng Honours Degree in Electronic and Information Engineering Part III
BEng Honours Degree in Mathematics and Computer Science Part III
MEng Honours Degree in Mathematics and Computer Science Part III
MEng Honours Degrees in Computing Part III
MSc in Computing Science
MSc in Computing Science (Specialist)
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C333

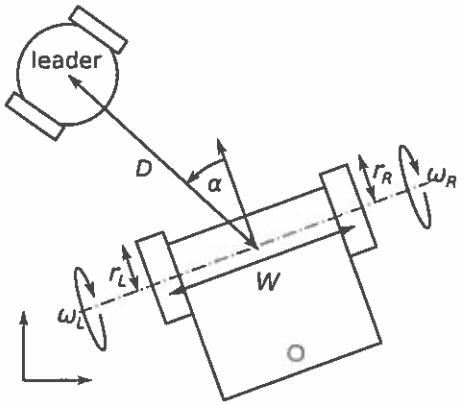
ROBOTICS

Wednesday 25 March 2015, 14:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 A differential drive robot is to be programmed using feedback control to follow an independently moving 'leader' vehicle. It has a vision system which can measure the relative angle α and distance D to the leader:



A low-level velocity controller is available which sets the rotation rates ω_L and ω_R of the left and right wheels respectively using the command `SetMotorVelocity(port, omega)`, where `port` is `left` or `right`. The wheels have radii r_L and r_R and the distance between them is W.

a Write a function `ComputeWheelRotations(v, thetadot)` in Python-like pseudocode which takes as arguments the desired forward velocity v and rotation rate $\dot{\theta}$ of the robot and returns appropriate ω_L and ω_R. The instantaneous forward velocity v and rotation rate $\dot{\theta}$ of a differential drive robot in terms of the linear velocities v_L and v_R of the two wheels are as follows:

$$v = \frac{v_R + v_L}{2}, \qquad \dot{\theta} = \frac{v_R - v_L}{W},$$

b Implement a proportional control law which aims to keep the measured relative angle α to the leader close to 0, and the measured relative distance D close to a user-settable reference distance r_D. This law will have the form of two independent proportional controllers: a relative angle controller `ComputeAngleControllerOutput(alpha)` should return a desired rotation speed $\dot{\theta}$, and a distance controller `ComputeDistanceControllerOutput(D, r_D)` should compute a desired forward velocity v.

c To specify the overall structure of the controller, draw a simple block diagram with the three functions you have implemented in boxes with arrows connecting them named with the variables which serve as inputs and outputs. The overall diagram should show how the wheel rotation rates ω_R and ω_L are computed from measurements α and D and the desired distance r_D.

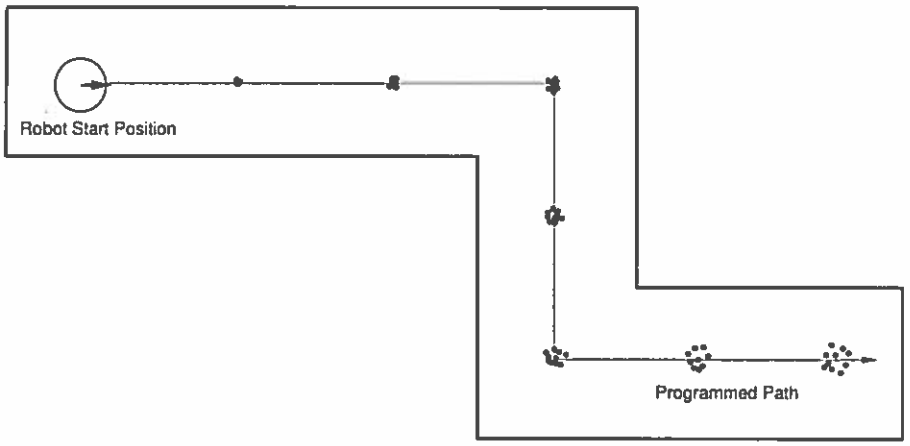
d In order to improve performance in the leader-following behaviour, it is decided to upgrate from proportional to `PID` control in the vision control loop. Modify the functions `ComputeAngleControllerOutput(Delta_theta)` and `ComputeDistanceControllerOutput(Delta_D, r_D)` to perform PID control, defining proportional, integral and differential gain constants for later tuning. Assume that the control functions are called at a constant rate of 20 Hz.

The four parts carry, respectively, 30%, 20%, 20%, and 30% of the marks.

2a The uncertain motion of a differential drive robot with no outward-looking sensors is modelled with the following expressions when commanded to make a forward motion of distance D or a pure rotation of angle α respectively:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + (D+e)\cos\theta \\ y + (D+e)\sin\theta \\ \theta + f \end{pmatrix} \quad , \quad \begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta + \alpha + g \end{pmatrix} .$$

Particles are used to represent its estimated location when programmed to move in steps through a set of waypoints defined along the centre of a corridor. After an initial calibration of terms e, f and g, the particles move as shown:



Draw three careful copies of this figure, showing how the evolution of particles would differ if e, f and g in turn were doubled in magnitude, highlighting and explaining with words the key differences. In each case, assume that the other two terms retain their original values. For clarity, please also note in words the main differences between the three figures.

b Using centimetre units for distances and radians for angles, the robot's motion can be programmed using position control with functions `DriveForward(D)`, (positive is forward), and `Rotate(alpha)` (positive is left). The robot is given an ordered list of (W_x, W_y) waypoints in the world coordinate frame. These are specified via a Python list `waypointList` of `Waypoint` objects, which have data members `Waypoint.x` and `Waypoint.y`. Write a clear and complete program in Python-like pseudocode such that the robot, starting from location $(x, y, \theta) = (0, 0, 0)$, will navigate to each of the waypoints in order as efficiently as possible by making steps of maximum 20cm. Assume here that the robot is very well calibrated and makes all commanded movements precisely. The angular orientation with which it reaches each waypoint does not matter. When it reaches each waypoint it should beep via the built-in function `Beep()`.

The two parts carry, respectively, 40%, and 60% of the marks.

3 A robot is equipped with a scanning sonar sensor which can be rotated to point at a horizontal angle α in degrees relative to the robot's forward direction ($\alpha = 0$ when the sonar points forward, and α is positive when it points to the left). The robot uses measurements z in cm from the sonar to update an occupancy map of its surroundings using the log odds form. Assume that the occupancy map has cell size 1cm and covers the area 0cm $\leq x <$ 500cm and 0cm $\leq y <$ 500cm.

a Write Python-like pseudocode for a function `UpdateOccupancyMap(x, y, theta, z, alpha)` to update the occupancy map in response to a sonar measurement z when the robot is at location (x, y, θ) and the sonar is at angle α. The occupancy map has been pre-allocated globally as a 2D array and the log odds value of the cell at (x, y) can be accessed at `occupancyMap[x][y]`. Assume that the robot is located somewhere near the middle of the occupancy map and that the maximum range of the sonar is less than the distance to the occupancy map's boundaries.

The half width of the sonar beam is 5° and the standard deviation of its depth measurements is 4cm. When it returns a depth measurement z, you should increase the log odds occupancy value of all cells whose centres lie within the beam width at a distance from the robot which is within plus or minus one standard deviation of the depth measurement; and you should decrease the occupancy value of cells within the beam at less than this distance. Choose sensible values for the amount of increase or decrease and explain their meaning.

b After some time exploring, the robot has created a high quality occupancy map of a room where the occupancy values of all cells within the reachable area have converged to confident values.

Explain with sketches how it would be possible, without needing to make any more measurements from the robot, to generate from the occupancy map a set of location signatures suitable for later place recognition (global localisation) of the robot within the room.

The two parts carry, respectively, 70%, and 30% of the marks.

4a A robot has a structured light depth sensor with minimum range 50cm and maximum range 4m. The sensor is precise but not very robust at close range, with Gaussian-distributed zero-mean uncertainty of standard deviation 1cm at its minimum range when it receives a good measurement but on 24% of measurements it reports random 'garbage' values between the minimum and maximum range. At greater range the precision decreases such that the standard deviation of good measurements is proportional to the square of ground truth distance; but becomes more reliable, with a 'garbage' measurement rate inversely proportional to depth.

Draw a suitable likelihood function $p(z|m)$ for the sensor in the form of three separate 2D 'slice' plots which show the probability of obtaining depth measurement z at ground truth distance $m =$1m, 2m, 3m respectively. Your plots should be carefully drawn to scale and annotated with the appropriate numerical values.

b The robot is to navigate within a closed walled area using Monte Carlo Localisation (MCL). Its 2D pose is specified with position and orientation (x, y, θ) relative to a standard coordinate frame. The area's walls are defined with a list of pairs of coordinates (A_x, A_y) and (B_x, B_y) which specify their endpoints. The depth sensor described above is located at the centre of the robot and facing forward. You are provided with the expression:

$$m = \frac{(B_y - A_y)(A_x - x) - (B_x - A_x)(A_y - y)}{(B_y - A_y)\cos\theta - (B_x - A_x)\sin\theta} .$$

for the forward distance to an infinite wall passing through (A_x, A_y) and (B_x, B_y) from a pose (x, y, θ).

Write in Python-like pseudocode a function `GetLikelihood(particle, z, walls)` which returns a floating point likelihood value for a particle given a sonar measurement `z`. Assume that `particle` is an object of the `Particle` class with data members `Particle.x`, `Particle.y` and `Particle.theta`, and that the walls are represented by a Python list `walls` of `Wall` objects, each of which has data members `Wall.Ax`, `Wall.Ay`, `Wall.Bx` and `Wall.By`. Use centimetre units throughout.

The two parts carry, respectively, 40%, and 60% of the marks.