



FACULTAT D'INFORMÀTICA DE BARCELONA

ROBÒTICA

Highway driving

Jaume Pladevall (jaume.pladevall)

Tutoritzat per
Cecilio Angulo Bahon

24 de desembre de 2018

Sumari

1	Enunciat	1
2	Username	1
3	Codi	2
4	Variables latents	4
5	Performance	4

1 Enunciat



Highway Driving



Program a Lincoln MKZ autonomous car to drive as fast as possible on a crowded highway.

Difficulty level: **Master**
Number of participants: **148**
Robot: **Lincoln MKZ**
Programming language: **Python**
Programming interface: **Webots**
Minimum commitment: **a couple of days**

Codi del controlador en un fitxer .txt per la tasca Highway Driving de la plana <https://robotbenchmark.net/> i nickname al simulador.

Elements de valoració:

1. Controlador sense regles IF-THEN, només funcions parametritzades
2. Almenys definida una variable latent, i utilitzada a les funcions
3. Score per sobre de 1000m (ho anotaré en algun moment del dia 27 de desembre)

2 Username



Jaume Pladevall

Username: jaume

3 Codi

```

1  """Sample Webots controller for highway driving benchmark."""
2
3  from vehicle import Driver
4
5  # name of the available distance sensors
6  sensorsNames = [
7      'front',
8      'front right 0',
9      'front right 1',
10     'front right 2',
11     'front left 0',
12     'front left 1',
13     'front left 2',
14     'rear',
15     'rear left',
16     'rear right',
17     'right',
18     'left']
19  sensors = {}
20
21  maxSpeed = 110      # Maxima velocitat que se li permet obtenir al cotxe
22  maxSteer = 0.4      # Maxim steer que se li permet al cotxe
23  driver = Driver()
24  driver.setSteeringAngle(0.0) # go straight
25
26  # get and enable the distance sensors
27  for name in sensorsNames:
28      sensors[name] = driver.getDistanceSensor('distance sensor ' + name)
29      sensors[name].enable(10)
30
31  # Range of sensors
32  fR = sensors['front'].getMaxValue()
33  lR = sensors['left'].getMaxValue()
34  rR = sensors['right'].getMaxValue()
35  fR0R = sensors['front right 0'].getMaxValue()
36  fR1R = sensors['front right 1'].getMaxValue()
37  fR2R = sensors['front right 2'].getMaxValue()
38  fL0R = sensors['front left 0'].getMaxValue()
39  fL1R = sensors['front left 1'].getMaxValue()
40  fL2R = sensors['front left 2'].getMaxValue()
41  rLR = sensors['rear left'].getMaxValue()
42  rRR = sensors['rear right'].getMaxValue()
43
44
45  dRight = 0.58      # Distancia original amb la valla de la dreta
46
47  # Variables lantents
48  # El valor de steering cap a l'esquerra es negatiu i cap a la dreta positiu
49  dLeft = -0.40      # Distancia que ens hem apartat cap a l'esquerra
50  lastSteer = 0      # Valor de l'ultim steer realitzat pel cotxe
51
52  # Configuracions
53  alpha1 = 0.8 # Importancia de la deteccio esquerra per girar esquerra amb steering
54  alpha2 = 0.8 # Importancia de la deteccio dreta per girar dreta amb steering
55  alpha3 = 0.2 # Importancia de la deteccio del darrera esquerra
56  alpha4 = 0.2 # Importancia de la deteccio del darrera dret
57  beta = 0.5 # importancia de la separacio amb el lateral dret
58  phi = 0.35 # si no hi ha ningun al davant poder fer steering
59  gamma = 0.01 # velocitat de retorn a la posicio inicial

```

```

60  theta =0.3 # valor per reduir la variacio d'un steering amb el seu anterior, evitar
        canvis bruscos
61  # Velocity Hiperparameter
62  vh1 =3      # Front Left
63  vh2 =3      # Front Right
64  vh3 =-0.25 # Left
65  vh4 =0.25  # Right
66  vh5 =-0.5  # Rear Left
67  vh6 =-0.5  # Rear Right
68
69  while driver.step() !=-1:
70
71      # Distance value of sensors
72      fD =sensors['front'].getValue()
73      lD =sensors['left'].getValue()
74      rD =sensors['right'].getValue()
75      fR0D =sensors['front right 0'].getValue()
76      fR1D =sensors['front right 1'].getValue()
77      fR2D =sensors['front right 2'].getValue()
78      fL0D =sensors['front left 0'].getValue()
79      fL1D =sensors['front left 1'].getValue()
80      fL2D =sensors['front left 2'].getValue()
81      rLD =sensors['rear left'].getValue()
82      rRD =sensors['rear right'].getValue()
83      fLeft =fL0D/fL0R +fL1D/fL1R +fL2D/fL2R
84      fRight =fR0D/fR0R +fR1D/fR1R +fR2D/fR2R
85
86      speed =pow(maxSpeed, (fD/fR)) -fLeft*vh1 -fRight*vh2 -(lD/lR)*vh3 -(rD/rR)*vh4 -(rLD/
        rLR)*vh5 -(rRD/rRR)*vh6
87
88      driver.setCruisingSpeed(max(speed,0))
89      # brake if we need to reduce the speed
90      speedDiff =driver.getCurrentSpeed() -speed
91      driver.setBrakeIntensity(max(min(speedDiff /speed, 1), 0))
92
93      steer =(-1+fD/fR-phi)*((dLeft*gamma+(lD/lR)+fLeft)*alpha1 +(dRight-(rD/rR)-fRight)*
        alpha2 +(rLD/rLR)*alpha3 -(rRD/rRR)*
        alpha4)
94
95      steer =min(max(steer, -maxSteer), maxSteer)
96      steer =(steer-lastSteer)*theta
97      lastSteer =steer
98      dLeft =1 -(lD/lR)*beta
99      driver.setSteeringAngle(steer)
100
101      if (driver.step() %1000000==0):
102          '''
103          print 'Left: %(l).2f FrontLeft: %(fl).2f Front = %(f).2f FrontRight = %(fr).2f
            Right: %(r).2f' % {
104              'l': lD/lR,
105              'fl': fLeft,
106              'f': fD/fR,
107              'fr': fRight,
108              'r': rD/rR,
109          }
110          '''
111          #print("Steer: %.2f" % (steer))
112          #print("dLeft: %.2f" % (dLeft))
113          #print("Speed: %.2f" % (speed))
114          #print("SpeedDiff: %.2f" % (speedDiff))

```

4 Variables latents

En el codi he utilitzat 2 variables latents: *dLeft* i *lastSteer*.

dLeft: Consisteix en que a partir de la distància amb el lateral esquerra, la preferència i la rapidesa en que retornarà a la posició de confort. A on la posició de confort és aquella que permet adelantar més fàcilment, en aquest benchmark la posició és el centre, ja que es permet adelantar tan per la dreta com per la esquerra.

Obté el seu valor a partir de: $1 - (lD/lR) * beta$. A on *lD* és el valor del sensor esquerra, *lR* és el màxim valor del sensor esquerra, *beta* és un hiperparàmetre.

1. Si no hi ha cap obstacle a l'esquerra, el valor de *dLeft* serà negatiu.
2. Si hi ha un obstacle a l'esquerra, depenent de la seva distància, serà positiu o negatiu:
 - Si està molt aprop de l'obstacle, llavors serà positiu
 - Si està lluny però el detecta, dependrà del valor de *beta*.

Afecta al valor que pren *steer* depenent en les condicions en les quals es troba el cotxe. A on la formula que calcula el valor de *steer* que hi afecta el valor de *dLeft* és la següent:

$$steer = (-1 + fD/fR - phi) * ((dLeft * gamma + (lD/lR) + fLeft) * alpha1 + ...$$


1. Si el cotxe no té cap obstacle al davant ($(-1 + fD/fR - phi)$ serà positiu) i no hi ha cap obstacle a l'esquerra:
 - Si el valor de *dLeft* és positiu, tendirà a girar cap a la dreta.
 - Si el valor de *dLeft* és negatiu, tendirà a girar cap a l'esquerra.
2. Si el cotxe té un obstacle al davant ($(-1 + fD/fR - phi)$ serà negatiu) i no hi ha cap obstacle a l'esquerra:
 - Si el valor de *dLeft* és positiu, tendirà a girar cap a l'esquerra.
 - Si el valor de *dLeft* és negatiu, tendirà a girar cap a la dreta.
3. Si hi ha algun obstacle a l'esquerra l'efecte que tindrà sobre *steer* serà proporcional a la distància amb l'obstacle, com més aprop més gir cap a la dreta.

Tot el descrit abans no té en compte els altres atributs que efecten a *steer*.

lastSteer: Consisteix en evitar canvis bruscos i innecessaris en la direcció del cotxe (variable *steer*). Això ho faig recordant el *steer* anterior i a partir del hiperparàmetre *theta*. Així puc obtenir un valor que es trobi entre el *steer* anterior i l'actual. La formula és la següent:

$$steer = (steer - lastSteer) * theta$$

5 Performance

ranking	user	date	runs	performance	
1.	 Jaume Pladevall jaume	2018-12-24 19:43:48	43	1433.887 m.	<button>Run</button>