[CSE10_S23] Robotics & Computer Vision

Εργασια 1

Maze-Solving Robot

Αντρέας ‘

Ο στόχος αυτού της εργασίας είναι να δημιουργήσει μια προσομοίωση για την κίνηση ενός ρομπότ μέσα σε έναν λαβύρινθο χρησιμοποιώντας τον προσομοιωτή Webots. Η προσομοίωση περιέχει μια πίστα ή ορθογώνια αρένα με μέγεθος 3x3 και έναν λαβύρινθο με 18 τοίχους διαφορετικών μεγεθών και 14 εμπόδια σε τυχαία σημεία του χώρου που πρέπει να αποφεύγει το ρομπότ. Το ρομπότ που χρησιμοποιείται στην προσομοίωση έχει τους κατάλληλους αισθητήρες για να ανιχνεύει τους τοίχους και τα εμπόδια στο λαβύρινθο και έναν ελεγκτή στην Python που εφαρμόζει έναν αλγόριθμο επίλυσης λαβύρινθου που επιτρέπει στο ρομπότ να περιηγείται στον λαβύρινθο αυτόνομα και να φτάσει στο τέλος του λαβύρινθου στο το συντομότερο δυνατό χρόνο.

Σχεδιασμός και Υλοποίηση: Το ρομπότ μας που λύνει λαβύρινθο σχεδιάστηκε για να περιηγείται σε μια ορθογώνια αρένα με διαστάσεις 3m x 3m. Η αρένα είχε 18 τοίχους, 7 βαρέλια λαδιού και 7 χαρτόκουτα, καθιστώντας την ένα δύσκολο περιβάλλον για πλοήγηση. Το πρώτο βήμα ήταν η δημιουργία της προσομοίωσης, οπού δημιούργησα τον κόσμο - την αρένα. Θα δημιούργησα μια ορθογώνια αρένα με διαστάσεις 3μ x 3μ. Πρόσθεσα τοίχους για να δημιουργήσουμε έναν λαβύρινθο με τουλάχιστον 18 τοίχους διαφορετικών μεγεθών. Οι τοίχοι πρέπει να είναι διατεταγμένοι με τέτοιο τρόπο ώστε να υπάρχει μόνο μία είσοδος και μία έξοδος στον λαβύρινθο. Θα πρόσθεσα επίσης 14εμπόδια, 7 βαρέλι λαδιού και 7 χαρτόκουτο, σε τυχαία σημεία του λαβύρινθου που πρέπει να αποφύγει το ρομπότ.

Χρησιμοποίησα το ρομπότ E-puck που παρέχεται στον προσομοιωτή Webots για την περίπτωση μας, με τον controller "my_ap". Το ρομπότ E-puck διαθέτει μια σειρά από αισθητήρες που είναι κατάλληλοι για πλοήγηση στο λαβύρινθο, συμπεριλαμβανομένων αισθητήρων εγγύτητας, αισθητήρων φωτός και κωδικοποιητών τροχών, για να μπορεί να ανιχνεύει και να πλοηγείται στο περιβάλλον του. Το ρομπότ μπορεί να ελεγχθεί χρησιμοποιώντας κώδικα Python που εκτελείται στον controller του ρομπότ.

Αλγόριθμος: Επέλεξα να εφαρμόσουμε τον αλγόριθμο που ακολουθεί τον τοίχο για να πλοηγηθεί στον λαβύρινθο δηλαδή κανόνα του δεξιού χεριού. Αυτός ο αλγόριθμος περιλαμβάνει τη διατήρηση του ρομπότ κοντά στον αριστερό ή τον δεξιό τοίχο του λαβύρινθου και την παρακολούθηση του μέχρι το ρομπότ να φτάσει στο τέλος του λαβύρινθου. Ο αλγόριθμος παρακολούθησης τοίχου είναι ένας δημοφιλής αλγόριθμος για ρομπότ που λύνουν λαβύρινθο λόγω της απλότητας και της αποτελεσματικότητάς του. Επίσης το robot μας αποφεύγει τα εμπόδια που βρίσκει μπροστά του καθώς εκτελεί τον αλγόριθμο wall-follower. Επιπρόσθετα επέλεξα αυτόν τον αλγόριθμο γιατί λειτουργεί καλά σε ορθογώνιες αρένες με τοίχους που είναι σχετικά ίσιοι και συνεχείς. Επιπλέον, επιτρέπει στο ρομπότ να εξερευνήσει ολόκληρο τον λαβύρινθο αποφεύγοντας τα εμπόδια, εφόσον ο λαβύρινθος έχει μόνο μία είσοδο και έξοδο.

Η υλοποίηση του αλγορίθμου παρακολούθησης του τοίχου περιλάμβανε πολλά βήματα. Πρώτα, το ρομπότ προγραμματίστηκε να κινείται προς τα εμπρός μέχρι να συναντήσει έναν τοίχο. Μόλις το ρομπότ εντόπισε έναν τοίχο, θα έστριβε αριστερά ή δεξιά, ανάλογα με το ποια πλευρά του ρομπότ ήταν πιο κοντά στον τοίχο. Το ρομπότ είναι προγραμματισμένο να ακολουθεί τον τοίχο μέχρι να φτάσει σε μια γωνία ή διασταύρωση. Στη γωνία ή στη διασταύρωση, το ρομπότ θα χρησιμοποιούσε τους αισθητήρες του για να καθορίσει ποια κατεύθυνση θα στρίψει στη συνέχεια. Αν υπήρχε τοίχος στα αριστερά, το ρομπότ θα στρίψει δεξιά. Αν υπήρχε τοίχος στα δεξιά, το ρομπότ θα στρίψει αριστερά. Αν υπήρχαν τοίχοι και στις δύο πλευρές, το ρομπότ θα συνέχιζε ευθεία. Το ρομπότ θα συνεχίσει να ακολουθεί τον τοίχο μέχρι να φτάσει στο τέλος του λαβύρινθου. Μόλις το ρομπότ φτάσει στο τέλος του λαβύρινθου, θα σταματήσει και δείξει ότι είχε λύσει τον λαβύρινθο.

Η διαδικασία περιελάμβανε τον προγραμματισμό του ρομπότ να ακολουθεί τον τοίχο και να στρίβει σε διασταυρώσεις για να εξερευνήσει ολόκληρο τον λαβύρινθο αποφεύγοντας τα εμπόδια. Συνολικά, το ρομπότ, λύνει λαβύρινθο και είναι επιτυχές στην πλοήγηση στον λαβύρινθο και στο τέλος.

Επίσης έχτρα πρόσθεσα στον κώδικα μου «να κινείτε το ρομπότ μου με wall-follower για 7 λεπτά (420 seconds) και μέτα τα 7 λεπτά καλείτε άλλος κώδικά second_code όπου το ρομπότ μου κινείτε με obstacles avoid»

Directory: ...\my_project\controllers\my_ap

Code:

```
"""my_ap controller."""

# You may need to import some classes of the controller module. Ex:
# from controller import Robot, Motor, DistanceSensor
from controller import Robot, DistanceSensor, Motor

def run_robot(robot):
# get the time step of the current world.
```

```
timestep = int(robot.getBasicTimeStep())
max_speed = 6.28
max_time = 420 # 7 minutes in seconds

# Enable motors
left_motor = robot.getDevice('left wheel motor')
right_motor = robot.getDevice('right wheel motor')

left_motor.setPosition(float('inf'))
left_motor.setVelocity(0.0)

right_motor.setPosition(float('inf'))
right_motor.setVelocity(0.0)

#enable proximity sensor
prox_sensors = []
for ind in range(8):
sensor_name = 'ps' + str(ind)
prox_sensors.append(robot.getDistanceSensor(sensor_name))
prox_sensors[ind].enable(timestep)

# Initialize variables
elapsed_time = 0
is_time_up = False

# Main loop:
# - perform simulation steps until Webots is stopping the controller
while robot.step(timestep) != -1 and not is_time_up:
# Read the sensors:
for ind in range(8):
print('ind: {}, val: {}'.format(ind,prox_sensors[ind].getValue()))

# Process sensor data here.
left_wall = prox_sensors[5].getValue() > 80
left_corner = prox_sensors[6].getValue() > 80
front_wall = prox_sensors[7].getValue() > 80

left_speed = max_speed
right_speed = max_speed

if front_wall:
print("Turn right in place")
left_speed = max_speed
right_speed = -max_speed
else:
if left_wall:
print("drive forward")
left_speed = max_speed
right_speed = max_speed
else:
print("Turn left")
```

```
left_speed = max_speed/8
right_speed = max_speed
if left_corner:
print("too close drive right")
left_speed = max_speed
right_speed = max_speed/8

# Enter here functions to send actuator commands, like:
# motor.setPosition(10.0)
left_motor.setVelocity(left_speed)
right_motor.setVelocity(right_speed)

# Update elapsed time
elapsed_time += timestep/1000.0
if elapsed_time >= max_time:
is_time_up = True

# Enter here exit cleanup code.
if is_time_up:
from second_code import run_robot as second_code
second_code(robot)

if __name__=="__main__":
# create the Robot instance.
my_robot = Robot()
run_robot(my_robot)
```