Ευφυή Αυτόνομα Συστήματα

Αναφορά Εργασίας εξαμήνου

Κωνσταντίνος Αγαθόπουλος 4119 - Δήμητρα Αγγελίδου 4200

Υλοποίηση του Αλγορίθμου i-Bug

Αξιοποιώντας τις πηγές που ήταν συνημμένες στην εκφώνηση της εργασίας καθώς και υλικό από το μάθημα, βασίσαμε το project μας σε τέσσερις κλάσεις εκ των οποίων η μία είχε αποκλειστικά το ρόλο του εκτελέσιμου αρχείου (*Main.java*) που κάνει generate το περιβάλλον του Simbad με βάση την κλάση *Env.java*. Εκεί αρχικοποιούμε και περιγράφουμε όλα τα αντικείμενα του περιβάλλοντος: τη λάμπα, το ρομπότ και τα διάφορα εμπόδια.

I. Το Περιβάλλον

Προγραμματίσαμε τις πηγές φωτός έτσι ώστε να εκπέμπει μια μόνο λάμπα (το light1), προκειμένου να μην αλλοιώνονται οι μετρήσεις των αισθητήρων, και την τοποθετήσαμε σε ύψος 2μ από το έδαφος. Έχουμε επίσης τις συναρτήσεις για τέσσερα διαφορετικά είδη εμποδίων που μας βοήθησαν στις δοκιμές αργότερα. Τέλος, τοποθετούμε στον κόσμο το ρομπότ μας, του οποίου η συμπεριφορά και οι ιδιότητες περιγράφονται στην κλάση *MyRobot.java*.

II. Ο Πράκτορας

Αυτή η κλάση επεκτείνει την κλάση Agent του Simbad, οπότε περιλαμβάνει και τη συνάρτηση που θα τρέχει σε κάθε βήμα της προσομοίωσης (*performBehavior()*). Εδώ αρχικοποιείται το αντικείμενο MyRobot με 8 sonars περιμετρικά του σκελετού του και τρεις αισθητήρες φωτός: δύο μπροστά - αριστερά και δεξιά αντίστοιχα - και έναν στο κέντρο του ρομπότ, πίσω από τους άλλους δύο, για τον καθορισμό της κατεύθυνσης του ρομπότ όταν αυτό βρίσκεται στη διεύθυνση της λάμπας.

Από τα οκτώ sonars στον εντοπισμό εμποδίων χρησιμοποιούμε τα μπροστινά πέντε, που παρατηρήσαμε ότι επαρκούν. Με δοκιμές καταλήξαμε ότι η εμβέλεια 1 είναι κατάλληλη για την ομαλή εκτέλεση του προγράμματος.

III. Ο Αλγόριθμος

1) Let $i_L = h_i(x)$.
2) Apply u_{ori} and then u_{fwd}.
3) If $h_i(x) = 1$, then terminate; the tower was reached.
4) If $i_L \neq h_i(x)$, then let $i_H = h_i(x)$.
5) If $h_t(x) = 0$, then go to Step 1.
6) Apply u_{fol}.
7) If $h_i(x) > i_H$ then go to Step 1.
8) Go to Step 6.

Για την εκτέλεση του iBug χρειαζόμαστε δύο μεταβλητές για την ένταση, που ενημερώνονται υπό συνθήκες με την τιμή έντασης του κεντρικού αισθητήρα (*cIntensity*): τις i_L και i_H. Η i_L παίρνει την τιμή της *cIntensity* κάθε φορά πριν προσανατολιστεί το ρομπότ. Η i_H αντιθέτως, μόνο αν το ρομπότ μετακινηθεί μετά τη u_{fwd}, αφού προσανατολιστεί.

Επιπλέον αποθηκεύουμε τις τρεις τελευταίες - πιο πρόσφατες - ενδείξεις του κεντρικού αισθητήρα στις μεταβλητές intensity1, intensity2 και intensity3 αντίστοιχα ώστε να ξέρουμε πότε το ρομπότ εντοπίζει τοπικό μέγιστο κατά την περιφορά των εμποδίων.

Για να περιγράψουμε τις συμπεριφορές του ρομπότ, στις ιδιότητες του προσθέσαμε και το RobotStatus που καθορίζει σε τι κατάσταση είναι σε κάθε βήμα της προσομοίωσης. Οι πιθανές συμπεριφορές - καταστάσεις είναι τέσσερις: *ORIENTATION* (u_{ori}), *FORWARD* (u_{fwd}), *FOLLOWING* (u_{fol}) και *END*. Σε κάθε εκτέλεση της performBehavior() η πρώτη συνθήκη που ελέγχουμε είναι αν το ρομπότ έφτασε κάτω από τη λάμπα, δηλαδή αν η ένταση του κεντρικού αισθητήρα ξεπέρασε ένα threshold (*GOAL)* που με πειραματισμό καταλήξαμε να θέσουμε ίσο με 0.784. Τότε το στάτους γίνεται *END* και το ρομπότ τερματίζει την κίνηση του ακόμα και αν οι συναρτήσεις των παραπάνω συμπεριφορών είχαν θέσει άλλο status για την επόμενη επανάληψη. Διαφορετικά ελέγχουμε σε ποια από τις άλλες τρεις καταστάσεις βρίσκεται το ρομπότ και καλείται η αντίστοιχη συνάρτηση από το αρχείο Helpers.java όπως περιγράφουμε παρακάτω:

➔ *ORIENTATION*

Αρχικά ενημερώνουμε την i_L. Έπειτα καλούμε τη συνάρτηση *orientate* που δέχεται το στιγμιότυπο του ρομπότ και τις τρεις εντάσεις (αριστερά, δεξιά, κέντρο) και προσανατολίζει το ρομπότ ώστε να κοιτάζει τη λάμπα. Αν οι δύο εντάσεις έχουν μεταξύ τους διαφορά πάνω από 10^{-3} συμπεραίνουμε ότι ο πράκτορας δεν είναι στραμμένος προς τη λάμπα οπότε δίνουμε ταχύτητα περιστροφής ίση με

```
sigmoid(ένταση_αριστερού_αισθητήρα - ένταση_δεξιού_αισθητήρα)*0.5
```

η οποία παρατηρήσαμε μετά από δοκιμές ότι μας δίνει την πιο ομαλή περιστροφή. Διαφορετικά, αν οι πλαϊνές εντάσεις είναι ίδιες αλλά ο κεντρικός αισθητήρας δέχεται μεγαλύτερη, σημαίνει ότι το ρομπότ έχει πλάτη στον στόχο οπότε δίνουμε σταθερή ταχύτητα περιστροφής αριστερόστροφα ώστε στο επόμενο βήμα της προσομοίωσης να ικανοποιεί την πρώτη συνθήκη. Τέλος όταν βεβαιωθούμε ότι το ρομπότ έχει προσανατολιστεί αλλάζουμε το RobotStatus σε *FORWARD.*

➔ *FORWARD*

Μόλις αλλάξει η κατάσταση από *ORIENTATION* σε *FORWARD* καλείται η *goForward* που δέχεται επίσης το στιγμιότυπο του ρομπότ και τις ενδείξεις των sonars. Εκεί ελέγχουμε αν κάποιο από τα sonars υπ' αριθμόν 0,1,2,6 ή 7 που βρίσκονται μπροστά και στο πλάι του ρομπότ εντοπίζουν εμπόδιο. Αν όχι, δίνουμε σταθερή ταχύτητα μπροστά και αλλάζουμε το στάτους σε *ORIENTATION* πάλι. Διαφορετικά αλλάζουμε την κατάσταση σε *FOLLOWING* για να ξεκινήσει η περιφορά του εμποδίου.

➔ *FOLLOWING*

Εδώ καλείται η *circumNavigate* με παραμέτρους το instance του πράκτορα, τα sonars και μια λογική τιμή που καθορίζει αν το ρομπότ θα κάνει περιφορά σύμφωνα με τη φορά του ρολογιού (true) ή ανάποδα (false). Εμείς με βάση την εκφώνηση της δίνουμε την τιμή false. Η περιφορά του εμποδίου γίνεται βρίσκοντας κάθε φορά το διάνυσμα που είναι κάθετο στην προβολή του ρομπότ στο σημείο όπου εντοπίζεται εμπόδιο από τα sonars. Όταν η circumNavigate το υπολογίσει, δίνει ανάλογες ταχύτητες κίνησης και περιστροφής ώστε το ρομπότ να ακολουθήσει αυτό το διάνυσμα. Έπειτα συγκρίνουμε την τρέχουσα *cIntensity* με την i_H. Αν είναι μεγαλύτερη σημαίνει ότι έχουμε πλησιάσει τη λάμπα. Αν ισχύει ταυτόχρονα αυτή η συνθήκη και έχουμε και τοπικό μέγιστο (*intensity2* > *intensity1* και *intensity2* > *intensity3*) τότε το ρομπότ προσπαθεί πάλι να κινηθεί προς τη λάμπα οπότε θέτουμε το στάτους σε *ORIENTATION.*