Αυτόνομοι Πράκτορες

Αναφορά 1ης Εργαστηριακής Άσκησης

Πέτρου Δημήτριος - 2018030070

Χανιά, Νοέμβριος 2021

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

1. Εισαγωγή

Στο πλαίσιο εξοικείωσης με το περιβάλλον προσομοίωσης Webots μελετήθηκε ο κώδικας του controller "nao_team_1". Ο συγκερκιμένος controller δίνει στον NAO μια σχετικά αυτόνομη συμπεριφορά ως προς τον εντοπισμό της μπάλας στο γήπεδο και την προώθηση της στον άξονα του τέρματος του αντιπάλου. Η συγγραφή του κώδικα που υλοποιεί τον controller έχει πραγματοποιήθει σε γλώσσα Java χάρη στην ευελιξία του Webots το οποίο προσφέρει στον προγραμματιστή πολλές επιλογές γλωσσών. Η κλάση `FieldPlayer.java`, η οποία κληρονομεί χαρακτηριστικά από την αφηρημένη κλάση `Player.java`, υλοποιεί την λειτουργικότητα που θα έχει ένα παίκτης NAO ο οποίος δρα εντός του γηπέδου (μη-τερματοφύλακας). Από την άλλη, η κλάση `GoalKeeper.java` υλοποιεί την συμπεριφορά του τερματοφύλακα και κληρονομεί επίσης την αφηρημένη κλάση `Player.java`. Ο controller αποτελείται και από άλλες κλάσεις, των οποίων ο ρόλος είναι να λειτουργούν σαν interface διασύνδεσης με τα διάφορα μοτέρ του ρομπότ, τις κάμερες, τους αισθητήρες υπερήχων, το γυροσκόπιο κτλ. Κύριο άξονα ενασχόλησης σε αυτήν την άσκηση αποτέλεσε η βελτίωση της συμπεριφοράς του FieldPlayer, ο καθορισμός ορισμένων παραμέτρων του 'κόσμου' του παιχνιδιού και η γενικότερη εξοικείωση με τον κώδικα και τα εργαλεία προσομοίωσης.

2. Κόσμος παιχνιδιού

Ο κόσμος στον οποίον αναπτύσσεται το ποδοσφαιρικό παιχνίδι είναι αντίγραφο του κόσμου που περιγράφεται στο αρχείο `robotstadium_nao_vs_robotis-op2.wbt`, από το αρχικό directory του Webots, με τις εξής διαφοροποιήσεις:

- Η κάμερα του NAO είναι προγραμματισμένη να αναγνωρίζει την μπάλα και το τέρμα στο περιβάλλον του γηπέδου χρησιμοποιώντας τα έντονα διακριτά χρώματα τους (μπάλα: rgb(240,140,50) τέρμα: rgb(140, 140, 15)). Το γήπεδο περιβάλλεται από ένα εικονικό βαςκγρουνδ για αισθητικούς λόγους, το οποίο είναι ορατό από την κάμερα του NAO. Ο κόσμος που παρέχεται από το Webots έχει ορισμένο `TexturedBackground:stadium`. Το συγκεκριμένο background περιέχει αποχρώσεις που ταιριάζουν με τα χρώματα της μπάλας ή του τέρματος, με αποτέλεσμα η κάμερα του NAO να αποπροσανατολίζεται και να θεωρεί ένα ασύνδετο σημείου του χώρου ως στόχο. Για τον λόγο αυτό στον νέο κόσμο οριστήκε να είναι `TexturedBackground:mountains` το οποίο περιέχει ουδέτερες αποχρώσεις.
- Ο αρχικός κόσμος του Webots υλοποιούσε αρχικά ένα παιχνίδι στο οποίο κάθε ομάδα διέθετε 4 παίκτες και έναν 1 τερματοφύλακα. Για να μειωθεί ο υπολογιστικός φόρτος ο νέος κόσμος διαμορφώθηκε ώστε κάθε ομάδα να διαθέτει 1 παίκτη και 1 τερματοφύλακα.

3. Κίνηση κλωτσιάς της μπάλας

Ο προγραμματισμός του nao_team_1 controller στην αρχική του έκδοση έκανε τον NAO να δίνει κίνηση στην μπάλα εξαιτίας της δικιάς του κίνησης και όχι με κάποιον χαρακτηριστικό τρόπο(πχ κλωτσιά). Εντός του directory "motions" υπάρχει αλληλουχία κινήσεων που κάνουν τον NAO να κλωτσά με το αριστερό του πόδι. Η αλληλουχία αυτή εισήχθει στον κώδικα ως εξής:

```
//FieldPlayer.java
import com.cyberbotics.webots.controller.*;

public class FieldPlayer extends Player {
    ...
    private Motion shootMotion;
    ...
    public FieldPlayer(int playerID, int teamID){
        ...
        shootMotion = new Motion("../../motions/Shoot.motion");
    }
}
```

Στην συνέχεια στο μέρος κώδικα όπου ο NAO είναι έτοιμος να κλωτσήσει την μπάλα έγινε η εξής τροποποίηση:

```
//FieldPlayer.java
import com.cyberbotics.webots.controller.*;

public class FieldPlayer extends Player {
    @Override public void run() {
        ...
        else{
            System.out.println("shooting !!!");
            playMotion(shootMotion);
        }
    }
}
```

Η κίνηση της κλωτσιάς αντικατέστησε μια αλληλουχία κινήσεων που μετακινούσε τον NAO μπροστά και παρέσυρε την μπάλα με την κίνηση του. Πλέον η συνθήκη, που αφορά στην απόσταση από την μπάλα και ελέγχεται πριν ο NAO κλωτσήσει, δεν είναι η κατάλληλη ώστε το πόδι του να είναι αρκετά κοντά στην μπάλα. Για αυτό το λόγο ο NAO αρχίζει να κινείται 'προσεκτικά' όταν βρίσκεται σε απόσταση μικρότερη από 0.22 και κλωτσάει την μπάλα όταν βρίσκεται σε απόσταση μικρότερο από 0.15 αν αυτό δεν έχει ήδη συμβεί.

4. Μέθοδος run() - Αναθεώρηση αναζήτησης μπάλας

Ο NAO έχει ως αρχικό στόχο να εντοπίσει την μπάλα εντός του χώρου του γηπέδου με σκοπό να κατευθυνθεί προς αυτή. Όταν η μπάλα είναι σε σχετικά μεγάλη απόσταση το FoV (Field of View) είναι διευρυμένο και η μπάλα εντοπίζεται εύκολα από την κάμερα χωρίς να χρειαστεί ο NAO να ανατιμήσει την στάση του. Ωστόσο σε περιπτώσεις που το ρομπότ έχει φτάσει αρκετά κοντά στην μπάλα ενδέχεται απότομα η κάμερα να μην είναι σε θέση να συνεχίζει να την κάνει track με αποτέλεσμα να είναι απαραίτητο να άλλαξει η θέση του. Το FoV σε αυτή την περίπτωση είναι αρκετά περιορισμένο σε σχέση με την μπάλα. Αρχικά ο controller ήθελε τον NAO να κάνει ένα `headScan()` με σκοπό να δει την μπάλα στον χώρο ευθεία μπροστά του από δεξιά μέχρι αριστερά.

Η στάση των χεριών του ρομπότ που καθορίζεται από τα motions που το μετακινούν πλαγιώς ή το περιστρέφουν, εμπόδιζε την κάμερα να έχει πλήρη εικόνα του δαπέδου μπροστά του. Με σκοπό την εξάλειψη αυτού του προβλήματος υλοποιήθηκε η μέθοδος `moveHandsAwayFromBody()` :

```
//FieldPlayer.java
import com.cyberbotics.webots.controller.*;

public class FieldPlayer extends Player {
    private void moveHandsAwayFromBody() {
      rightShoulderPitch.setPosition(2.0);
      rightShoulderRoll.setPosition(-0.6);
      rElbRoll.setPosition(1.5);
      rElbYaw.setPosition(0.0);

      leftShoulderPitch.setPosition(2.0);
      leftShoulderRoll.setPosition(0.6);
      lElbRoll.setPosition(-1.5);
      lElbYaw.setPosition(-1.5);
    }
}
```

Η πρόσβαση στο κάθε κινούμενο μέρος των χεριών του ρομπότ έγινε χρησιμοποιώντας την κλάση Motor.

Δοκιμάζοντας τον NAO σε δράση υπήρξαν περιπτώσεις που ένα βήμα πίσω δεν ήταν αρκετό ωστέ η κάμερα να βρει την μπάλα όταν αυτή δεν ήταν ορατή. Για τον λόγο αυτό προστέθηκε άλλη μια κίνηση προς τα πίσω πριν ο NAO ξεκινούσε να κοιτάζει στον χώρο γύρω του (μέσω headScan()) με σκοπό να ανιχνέυσει την μπάλα. Ο τελικός κώδικας ανάτιμησης της στάσης του ρομπότ σε περιπτώσεις που η μπάλα δεν ήταν ορατή στο άμεσο FoV διαμορφώθηκε ως εξής:

```
//FieldPlayer.java
import com.cyberbotics.webots.controller.*;

public class FieldPlayer extends Player {
    @Override public void run() {
        ...
       while (getBallDirection() == NaoCam.UNKNOWN) {
            System.out.println("searching the ball");
            getUpIfNecessary();
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            moveHandsAwayFromBody();
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            headScan();
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            playMotion(backwardsMotion);
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            playMotion(backwardsMotion);
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            moveHandsAwayFromBody();
            headScan();
            if (getBallDirection() != NaoCam.UNKNOWN) break;
            turnLeft180();
       }
        ...
    }
}
```

5. Ανίχνευση πτώσης

Για την ανίνχευση της πτώσης ο δεδομένος controller χρησιμοποιεί το γυροσκόπιο του NAO το οποίο δίνει πληροφορίες για την διεύθυνση του σε σχέση με τους 3 αξόνες [x,y,z]. Δεδομένου ότι μια πτώση συνεπάγεται ότι το ρομπότ είναι 'ξαπλωμένο' στο έδαφος είναι εφικτό να εκμεταλλευτούν οι αισθητήρες πίεσης που διάθετει στα πέλματα του, ώστε να ανιχνευθεί μια ενδεχόμενη πτώση. Όταν και τα δύο πέλματα δεν άπτονται του εδάφους σημαίνει ότι το ρομπότ έχει πέσει (...καθώς δεν μπορεί να πετάει). Με κατάλληλη προγραμματιστική πρόσβαση στους 3 αισθητήρες κάθε πέλματος υπολογίζεται η συνολική πίεση που 'διαβάζουν' η συνάρτηση `getUpIfNecessary()` τροποποιήθηκε ως εξής:

```
//Player.java

 protected void getUpIfNecessary() {
    double leftFootSensor[] = fsr[0].getValues(); //Left Foot sensors l[0] & l[1] & l[2]
    double rightFootSensor[] = fsr[1].getValues(); //Right Foot sensors r[0] & r[1] & r[2]

    double pressureReading = Math.abs(leftFootSensor[0]) + Math.abs(leftFootSensor[1]) +
                             Math.abs(leftFootSensor[2]) + Math.abs(rightFootSensor[0]) +
                    Math.abs(rightFootSensor[1]) + Math.abs(rightFootSensor[2]);
    if (pressureReading < 30){
      System.out.println("Oops I Fell! Lets stand up...");
      playMotion(standUpFromFrontMotion);
    }
 }
```