Τεχνολογίες Ευφυών Συστημάτων και Ρομποτική 2020-21
1η Υποχρεωτική Εργασία

Μπουλαφέντης Χρήστος ΑΜ:1059612 Δ' Έτος

1) Περιγραφή ελεγκτή και υλοποίησης:

Το πρόγραμμα μου χωρίζεται σε 5 διαφορετικά μέρη.:

- Το Initialisations, στο οποίο δημιουργώ την προσομοίωση, δημιουργώ τα γραφικά, φορτώνω τα ρομπότ και ορίζεται η αρχική θέση του ρομπότ.
- Το Controllers, που περιέχει τις κλάσεις των 2 Controllers που χρησιμοποιώ.
- Το Συναρτήσεις, στο οποίο ορίζονται οι συναρτήσεις που χρησιμοποιούνται.
- Το Μεταβλητές, στο οποίο αρχικοποιούνται μεταβλητές και πίνακες για αργότερη χρήση.
- Τέλος είναι το Main στο οποίο καθορίζεται η σείρα εκτέλεσης των εντολών και η επαναληπτική μέθοδος για την επίλυση του προβλήματος.

1. Όσον αφορά το πρώτο κομμάτι,το κομμάτι δηλάδη των Initialisations, ο κώδικας μου είναι σχεδόν ίδιος με αυτόν του αρχείου example.py που μας δώθηκε, εκτός από τον πίνακα (31) "tower_poles".

2. Για Controllers χρησιμοποίησα 2 διαφορετικά για να με καλύψουν στα είδη των κίνησεων που ήθελα να επιτελεί ο βραχίονας μου.
   Αρχικά, για την κίνηση του βραχίονα και την τοποθέτηση του end effector (panda_hand) στην σωστή θέση χρησιμοποίησα από το αρχείο 7-vel-task-separate.py τον Controller PITask, διότι μου έδινε μεγαλύτερη ελευθερία στον ξεχωριστό έλεγχο του rotation και translation που εισάγω ως στόχο. Αν και τελικά κατέληξα να μην χρειάζομαι αυτήν την ελευθερία μέσα στον Controller, διότι πραγματοποίησα τις επιθυμητές αλλαγές στο αντικείμενο Isometry3 πριν την εισαγωγή του στον Controller.
   Τον δεύτερο Controller τον χρειάστηκα για να μπορέσω να κίνησω τα "δάκτυλα" της δαγκάνας του βραχίονα. Για αυτήν την λειτουργία χρησιμοποίησα τον PIJoint Controller του αρχείου 5-pi.py, καθώς ήθελα να πραγματοποιεί κίνηση μονάχα των joints των "δακτύλων" του βραχίονα, χωρίς να υπάρχει κίνηση στα υπόλοιπα κομμάτια.

3. MoveArm(): Υπεύθυνη για την ενημέρωση του PITask Controller και την πραγματοποίηση της κίνησης.
   KleiseDagana(): Υπεύθυνη για το "κλείσιμο" της δαγκάνας με στόχο το "πιάσιμο" των δίσκων.
   AnoikseDagana(): Υπεύθυνη για το "άνοιγμα" της δαγκάνας με στόχο την απελευθέρωση των δίσκων.
   Katastasi(): Υπεύθυνη για τον εντοπισμό των θέσεων των δίσκων.
   CurPoleofDisk(): Υπεύθυνη για την επιστροφή του pole στον οποίο βρίσκεται ο δίσκος που προσπαθεί ο βραχίονας να μετακινήσει.
   AllagiStoxou(tf_desired, z_axis, anex): Τέλος, η συνάρτηση αυτή είναι υπεύθυνη για την εύκολη αλλαγή του στόχου για την μετακίνηση του βραχίονα.

4. Στο κομμάτι των μεταβλητών ορίζονται ένα πλήθος σταθέρων, μεταβλητών, πινάκων και αντικειμένων απαραίτητων για την λειτουργία του κώδικα, που θα αναλυθούν περαιτέρω στην συνέχεια.

5. Τέλος στο κομμάτι Main, που περιλαμβάνει ουσιαστικά το λειτουργικό κομμάτι του κώδικα θα περιγραφεί αναλυτικά παρακάτω.

2) Αναλυτική λειτουργία του κώδικα και λογική της δημιουργίας του:

Θα περιγράψω την λειτουργία του κώδικα ακολουθώντας την ροή της Main και παράλληλη επεξήγηση των συναρτήσεων και ό,τι άλλο χρειαστεί:

Πριν την Main καλείται η συνάρτηση next_move η οποία επιστρέφει τον δίσκο που πρέπει να μετακινηθεί και το αντίστοιχο pole. Για την συνάρτηση Katastasi που εισάγουμε σαν όρισμα θα αναφερθώ παρακάτω.

(247-249) Αρχικά, το πιο σημαντικό είναι η περιγραφή του σημείου στόχος στο οποίο θέλουμε να οδηγείται κάθε φορά ο end effector(panda_hand) του βραχίονα. Για τον λόγο αυτόν δημιούργησα το Isometry3 αντικείμενο stoxos, το οποίο θα περιέχει σε κάθε βήμα το επιθυμητό rotation και translation του βραχίονα. Σαν τιμές στο rotation αυτού του αντικειμένου έχω ορίσει:

- Z άξονας: Ήθελα σε κάθε κίνηση η δαγκάνα να είναι παράλληλη του ρομπότ tower, ώστε να αποφεύγονται συγκρούσεις και άλλα απρόβλεπτα αποτελέσματα κατά τις κινήσεις της δαγκάνας, επομένως εισήγαγα το x_rot[2]. (241) Η μεταβλητή αυτή είναι το τρίτο στοιχείο του πίνακα που περιέχει σε μορφή 1x3 το rotation του ρομπότ tower, μετά από μετατροπή του 3x3 rotation του Tranformation Matrix του tower_base_link. Στην τιμή αυτή προστίθεται το np.pi/2, ώστε να είναι όντως παράλληλη η δαγκάνα.
- Y άξονας: Στον άξονα αυτόν εισάγω την τιμή np.pi, ώστε να κοιτάει πάντοτε προς τα κάτω η δαγκάνα.
- X άξονας: 0.

Οι τιμές αυτές παραμένουν σταθερές καθ’ όλη την επίλυση του ρποβλήματος.

(220-226) Πρώτο βήμα της Main είναι η μετατροπή του αντικειμένου stoxos στην επιθυμητή θέση μετακίνησης του βραχίονα μέσω της συνάρτησης AllagiStoxou. Η συνάρτηση αυτή δέχεται 3 ορίσματα. Το tf_desired είναι το Transformation matrix του νέου σημείου στόχου. Το z_axis είναι η τιμή στον άξονα Z που θέλω να έχει το νέο Tranformation Matrix. Τέλος, η anex είναι Boolean μεταβλητή που καθορίζει αν το z_axis θα προστεθεί στην ήδη υπάρχουσα Z τιμή ή αν θα την αντικαταστήσει. Σαν tf_desired εισάγω σε κάθε βήμα το transformation matrix του tip του κάθε δίσκου που θέλω να μετακινήσω.

(157-182) Έπειτα, με την συνάρτηση MoveArm πραγματοποιείται η κίνηση του βραχίονα στο stoxos. Πρώτο βήμα είναι η δημιουργία του αντικειμένου controllerT της κλάσης PITask με όρισμα το stoxos που δημιουργήσαμε νωρίτερα. Σην συνέχεια πραγματοποιείται επαναληπτικά η διαδικασία μετακίνησης όπως ήταν στο αρχείο του Controller. Η διαφορά είναι πως μετά τον υπολογισμό σε κάθε βήμα του cmd, αν η μεταβλητή keep_closed είναι True, στην θέση 7 του cmd, που αντιστοιχεί στο “δάκτυλο” της δαγκάνας, εισάγεται μία τιμή, ώστε να διατηρείται κλειστή η δαγκάνα κατά την διάρκεια της μετακίνησης από μία σταθερή δύναμη. Επίσης,στο τέλος κάθε επανάληψης γίνεται έλεγχος για το σφάλμα της θέσης, όπως υπολογίζεται απο το update του Controller. Αν το σφάλμα είναι μικρότερο απο το μέγιστο επιτρεπτό, που ορίζεται στην μεταβλητή max_error_arm, τότε βγαίνει από τον βρόχο προκειμένου να μεταβεί στη επόμενη εντολή.

Οι παραπάνω διαδικασίες επαναλαμβάνονται μέχρι η δαγκάνα να βρεθεί σε θέση να “πιάσει” το tip του δίσκου. Τότε καλείται η συνάρτηση KleiseDagana. (123-138) Στην συνάρτηση αυτή δίνεται η εντολή στα “δάκτυλα” της δαγκάνας μέσω του PIJoint να κλείσουν τελείως. Όπως στην μετακίνηση του βραχίονα, στο τέλος της επαναληπτικής διαδικασίας πραγματοποιείται ένας έλεγχος. Αυτήν την φορά όμως ο έλεγχος είναι για την μεταβολή του σφάλματος μεταξύ 2 επαναλήψεων. Αν η μεταβολή του σφάλματος είναι μικρή,σημαίνει πως ο βραχίονας έχει βρει

εμπόδιο στο tip και δεν μπορεί να κλείσει παραπάνω. Ο έλεγχος αυτός γίνεται με βάση την μεταβλητή max_rithmall_dak. Μετά την πραγματοποίηση αυτών, η μεταβλητή keep_closed, που αναφέρθηκε νωρίτερα, παίρνει την τιμή True για να διατηρείται κλειστή η δαγκάνα.

Στην συνέχεια καλείται η συνάρτηση CurPoleofDisk. Για την εξήγηση αυτής της συνάρτησης, πρωτα θα εξηγηθεί η συνάρτηση Katastasi. (184-209) Η σύνάρτηση αυτή κάνει έλεγχο της θέσης των δίσκων και των poles και επιστρέφει την “Κατάσταση” στην οποία βρίσκεται το περιβάλλον.

Τα επόμενα βήματα έχουν ως στόχο την εύρεση της σωστής θέσης στην οποία θα οδηγηθεί ο βραχίονας ώστε να πραγματοποιηθεί σωστά η μετακίνηση του δίσκου. (211-218)Έτσι λοιπόν, στην συνάρτηση CurPoleofDisk γίνεται έλεγχος της κατάστασης, η οποία προκύπτει από την συνάρτηση Katastasi για να επιστραφεί το pole στο οποίο βρίσκεται ο δίσκος που προσπαθώ να μετακινήσω. Με την βοήθεια του όρου που μόλις υπολογίσαμε μπορούμε να υπολογίσουμε το διάνυσμα που δείχνει την απόσταση του κέντρου του δίσκου μέχρι το κέντρο του pole στο οποίο βρίσκεται, το οποίο εισάγουμε στο oros. Τέλος, υπολογίζεται το διάνυσμα που δείχνει την απόσταση του tip από το κέντρο του δίσκου. Αυτά συνδυάζονται μαζί με την θέση του pole στο οποίο πρέπει να μετακινηθεί ο δίσκος για τον ορισμό του translation του κενού αντικειμένου upologismos_stoxou, το οποίο θα μετατραπεί στον νέο στόχο.

Συνεχίζεται η διαδικασία αλλάγης στόχου και μετακίνησης μέχρι το σημείο που καλείται η συνάρτηση AnoikseDagana. (140-155)Ομοίως με την συνάρτηση KleiseDagana, χρησιμοποιείται ο PIJoint Controller για την μετακίνηση των δακτύλων της δαγκάνας. Αυτήν την φορά ο έλεγχος για την έξοδο από τον βρόγχο γίνεται και πάλι με τον έλεγχο του σφάλματος.

Όλα τα παραπάνω πραγματοποιούνται συνεχώς μέχρι η συνάρτηση Katastasi να ανιχνεύσει πως το περιβάλλον έχει φτάσει στην κατάσταση [[2, 1, 0], [], []] και επομένως έχει επιλυθεί το πρόβλημα. Αν δεν έχει επιλυθεί ακόμα, ενημερώνονται οι μεταβλητές disk pole από την συνάρτηση next_move και επαναλαμβάνεται η διαδικασία μετακίνησης του επόμενου δίσκου.

Πρόβλημα: Όταν προσπαθώ να κλείσω την προσομοίωση με το “X”, αν η συνάρτηση Katastasi() δεν επιστρέφει μια κατάσταση με όλους τους δίσκους σε ένα από τα poles, “κολλάει” η προσομοίωση και αναγκάζομαι να την κλείσω από το τερματικό.