Αναφορά Project-1- Robotics

ΤΕΧΝΟΛΟΓΙΕΣ ΕΥΦΥΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2021-2022

Ιωάννης Χαραλάμπους (1059685)

1. Περιγραφή Υλοποίησης

Στην παρούσα εργασία δημιουργήθηκε ένας ελεγκτής που συνδυάζει Behavior Trees με ένα low-level task space controller προκειμένου να λυθεί το Pick and Place πρόβλημα .
Χρησιμοποιήθηκε το αρχικό αρχείο tiago_pick_palce.py ως βαση στη λύση μου για αρχικοποίηση των ρομπότ και για έλεγχο αν λύθηκε το πρόβλημα 50 φορές.

Στην υλοποίηση μου έχω δημιουργήσει δύο κλάσεις και μία συνάρτηση για την λύση του προβλήματος.

```
class PITask:
```

είναι ο low-level task space controller ο οποίος ορίζεται για κάθε ακολουθία του Behavior Tree και έχει διαφορετικές παραμέτρους.

```
class ReachTarget(py_trees.behaviour.Behaviour):
```

ορίζει κάθε συμπεριφορά και εντολή που θα έχει ο κάθε controller για κάθε κίνηση του ρομπότ σε κάθεμία ακολουθία(sequence) του Behavior tree.

```
def change_base_pos(base_pose,root):
```

Η συνάρτηση αυτή αρχικοποιεί όλο το behavior tree και τις θέσεις που πρέπει να πάει το ρομπότ.Δέχεται ως όρισμα την θέση του box και την αρχικοποεί αναλόγως έτσι ώστε να μπορεί να γίνει αρχικοποίηση του δέντρου για 50 διαφορετικές τυχαίες θέσεις του box.

2. Κώδικας Υλοποίησης

Η αρχικοποίηση του ρομπότ και των αντικειμένων έγινε όπως ακριβώς και στο αρχείο tiago_pick_place.py που δόθηκε.

2.1 class PITask

```python
#controller
class PITask:
    def __init__(self, target, dt, Kp = 10., Ki = 0.1):
        self._target = target
        self._dt = dt
        self._Kp = Kp
        self._Ki = Ki
        self._sum_error = 0

    def set_target(self, target):
        self._target = target

    # function to compute error
    def error(self, tf):
        # 2 ways of computing rotation error in world frame
        # # 1st way: compute error in body frame
        # error_in_body_frame = rd.math.logMap(tf.rotation().T @ self._target.rotation())
        # # transform it in world frame
        # error_in_world_frame = error_in_body_frame @ tf.rotation().T
        # 2nd way: compute error directly in world frame
        rot_error = rd.math.logMap(self._target.rotation() @ tf.rotation().T)
        lin_error = self._target.translation() - tf.translation()
        return np.r_[rot_error, lin_error]

    def update(self, current):
        error_in_world_frame = self.error(current)

        self._sum_error = self._sum_error + error_in_world_frame * self._dt

        return self._Kp * error_in_world_frame + self._Ki * self._sum_error
```

Η κλάση αυτή ορίζει έναν task space controller ,υπολογίζει κάθε φορά το σφάλμα ανάμεσα στο μητρώο μετασχιματισμού της τωρινής θέσης με την επιθυμητή θέση του ρομπότ και το χρησιμοποιεί για να βρεί την κάθε ταχύτητα χρησιμοποιώντας τον έλεγχο με ανατροφοδότηση.

2.2 class ReachTarget

```
class ReachTarget(py_trees.behaviour.Behaviour):
    def __init__(self, robot, tf_desired, dt, link, sequence, name="ReachTarget"):
        super(ReachTarget, self).__init__(name)
        # robot
        self.robot = robot
        # end-effector name
        self.eef_link_name = link
        #use A,B,C,D,E,F, for every different movement sequence
        self.sequence=sequence
        # set target tf
        self.tf_desired = tf_desired
        # dt
        self.dt = dt

        self.logger.debug("%s.__init__()" % (self.__class__.__name__))

    def setup(self):
        self.logger.debug("%s.setup()->does nothing" % (self.__class__.__name__))
```

Η συγκεκριμένη κλάση είναι αυτή που μας ορίζει την συμπεριφορά κάθε ακολουθίας του behavior tree και ακολούθως ορίζει την κάθε κίνηση του ρομπότ.

Δέχεται ως ορίσματα το ρομπότ που θα χρησιμοποιήσουμε ,την επιθυμητή θέση που θέλουμε να φτάσει κάθε ακολουθία ,τον χρόνο όπου ασκούνται η ταχύτητες στο ρομπότ ,το όνομα του end-effector και το string sequence .

Το sequence παίρνει τις τιμές A,B,C,D,E,F,G και χρησιμοποιείται για να διαχωρίζονται οι λειτουργίες τις κάθε ακολουθίας κινήσεων του ρομπότ.

```
#initialize controller and change Kp, Ki parameters of controller so we can have movements with different speeds for each task
    def initialise(self):
        self.logger.debug("%s.initialise()->init controller" % (self.__class__.__name__))
        #controller parameters for robot movement going to basket
        if self.sequence in ['E']:
            self.Kp = 0.1
            self.Ki = 0.01
        #controller parameters for robot movement going to box
        elif self.sequence in ['A']:
            self.Kp = 1
            self.Ki = 0.1
        #controller parameters for every other movement
        else:
            self.Kp = 4.
            self.Ki = 0.1
        #initialize controller
        self.controller = PITask(self.tf_desired, self.dt, self.Kp, self.Ki)
```

Έδω γίνεται η αρχικοποίηση κάθε controller για κάθε ακολουθία κινήσεων.

Οι ακολουθίες έχουν διαφορετικές παραμέτρους όταν χρειάζομαι κινήσεις με διαφορετικές ταχύτητες

```
  #runs every sequence of the behavior tree
  def update(self):
      new_status = py_trees.common.Status.RUNNING
      # control the robot
      tf = self.robot.body_pose(self.eef_link_name)
      vel = self.controller.update(tf)
      jac = self.robot.jacobian(self.eef_link_name) # this is in world frame
      jac_pinv = np.linalg.pinv(jac) # np.linalg.pinv(jac) # get pseudo-inverse
      cmd = jac_pinv @ vel

      #these cases make up the different commands that we give to the robot
```

Στην update γίνεται η εκτέλεση κάθε ακολουθίας κινήσεων

Αρχικά υπολογίζονται οι εντολές που θα εισάγουμε στο ρομπότ μέσω του controller .

Στη συνέχεια ανάλογα με την ακολουθία κινήσεων στην οποία βρισκόμαστε περιορίζονται οι κινήσεις του ρομπότ δηλαδή οι εντολές ενώ χρησιμοποιούνται και κάποιες χειροποίητες

Sequence A : το σώμα του ρομπότ κινείται κοντά στο κουτί

```
#robot goes close to box
if self.sequence in ['A']:
    #eliminate any sideways or vertical body movement
    cmd[0:2]=0
    #eliminate any arm movement
    cmd[5:]=0
    #keep gripper open
    cmd[24]=0.01
```

Επιτρέπω τις κινήσεις του σώματος του ρομπότ στο δυσδιάστατο επίπεδο δάπεδο χωρίς να μπορεί να γύρει το σώμα του ή να κινηθεί οριζόντια. Επίσης μηδενίζω οποιαδήπονται εντολή στο χέρι του ρομπότ. Τέλος διατηρώ το γάντζο μόνιμα ανοικτό.

Sequence B : το χέρι του ρομπότ κινείται κοντά στο κουτί κ ο γάντζος το περιβάλει

```
#arm goes close to box and gripper goes in position  to grab
elif self.sequence in ['B']:
    #eliminate any body movement except of horizontal rotation
    cmd[0:2]=0
    cmd[3:6]=0
    #keep gripper open
    cmd[24]=0.01
```

Επιτρέπω τις κινήσεις του χεριού του ρομπότ και μηδενίζω τις εντολές για κίνηση του σώματος εκτός απο την περιστροφή γύρω από τον άξονα του . Επίσης διατηρώ το γάντζο μόνιμα ανοικτό.

Sequence C : Κλείνει το γάντζο

```
#short movement just for the gripper to grab box
elif self.sequence in ['C']:
    #eliminate any body movement
    cmd[0:6]=0
```

Επιτρέπω οποιαδήποτε κίνηση του χεριου του ρομπότ καθώς η ακολουθία αυτή διαρκεί λίγο.Μηδενίζω οποιαδήποτε κίνηση του σώματος.

Sequence D : το χέρι υψώνεται κρατώντας το κουτί.

```
#arm resets its position while holding box
elif self.sequence in ['D']:
    #eliminate any body movement
    cmd[0:6]=0
    cmd[16]=0
    #keep gripper close
    cmd[24]=-0.01
```

Μηδενίζεται κάθε εντολή για κίνηση του σώματος και μηδενίζεται και η ανύψωση του σώματος του ρομπότ μέσω της μέσης του που μπορεί να ανυψωθεί cmd[16]. Τέλος διατηρείται ο γάντζος κλειστός για να μην πέσει το κουτί.

Sequence E : Το σώμα φτάνει κοντά στο καλάθι με το γάντζο να βρίσκεται από πάνω του

```
#body moves close to basket
elif self.sequence in ['E']:
    #eliminate any sideways or vertical body movement
    cmd[0:2]=0
    #eliminate any arm movement
    cmd[5:]=0
    #keep gripper close
    cmd[24]=-0.01
```

Επιτρέπω τις κινήσεις του σώματος του ρομπότ στο δυσδιάστατο επίπεδο δάπεδο χωρίς να μπορεί να γύρει το σώμα του ή να κινηθεί οριζόντια. Επίσης μηδενίζω οποιαδήπονται εντολή στο χέρι του ρομπότ. Τέλος διατηρώ το γάντζο μόνιμα κλειστό.

Sequence F : Το χέρι τοποθετείται μέσα στο καλάθι για να σιγουρευτεί ότι το κουτί θα πέσει μέσα του

```
# arm goes in the basket to make sure the box goes in the basket
elif self.sequence in ['F']:
    #eliminate any body movement
    cmd[0:6]=0
    cmd[16]=0
    #keep gripper close
    cmd[24]=-0.01
```

Μηδενίζεται κάθε εντολή για κίνηση του σώματος και μηδενίζεται και η ανύψωση του σώματος του ρομπότ μέσω της μέσης του που μπορεί να ανυψωθεί και να χαμηλώσει cmd[16]. Τέλος διατηρείται ο γάντζος κλειστός για να μην πέσει το κουτί.

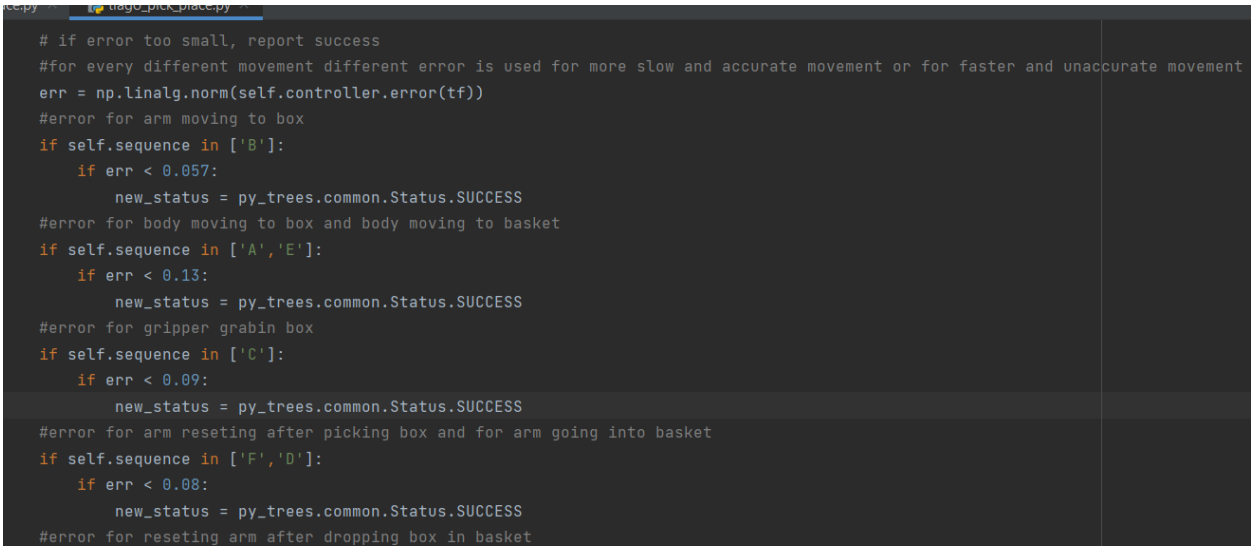
Sequence G : το χέρι υψώνεται κοντά στη θέση που είχε το ρομπότ στην αρχικοποίηση του και ο γάντζος ανοίγει μόνιμα αφήνοντας το κουτί στο καλάθι στην αρχή της ακολουθίας κινήσεων.

```
#arm drops box into basket and resets its position
elif self.sequence in ['G']:
    #eliminate any body movement
    cmd[0:6]=0
    cmd[16]=0
    #open gripper
    cmd[24]=0.01
#print(cmd)
self.robot.set_commands(cmd)
```

Μηδενίζεται κάθε εντολή για κίνηση του σώματος και μηδενίζεται και η ανύψωση του σώματος του ρομπότ μέσω της μέσης του που μπορεί να ανυψωθεί και να χαμηλώσει cmd[16]. Τέλος ο γάντζος ανοίγει μόνιμα.

Αφού προσαρμοστουν οι εντολές της κάθε ακολουθίας , εφαρμόζονται στο ρομπότ.

Στη συνέχεια ακολουθεί ο έλεγχος του σφάλατος κάθε ακολουθίας κινήσεων



```
# if error too small, report success
#for every different movement different error is used for more slow and accurate movement or for faster and unaccurate movement
err = np.linalg.norm(self.controller.error(tf))
#error for arm moving to box
if self.sequence in ['B']:
    if err < 0.057:
        new_status = py_trees.common.Status.SUCCESS
#error for body moving to box and body moving to basket
if self.sequence in ['A','E']:
    if err < 0.13:
        new_status = py_trees.common.Status.SUCCESS
#error for gripper grabin box
if self.sequence in ['C']:
    if err < 0.09:
        new_status = py_trees.common.Status.SUCCESS
#error for arm reseting after picking box and for arm going into basket
if self.sequence in ['F','D']:
    if err < 0.08:
        new_status = py_trees.common.Status.SUCCESS
#error for reseting arm after dropping box in basket
```

Κάθε ακολουθία έχει διαφορετικό σφάλμα προκειμένου να έχουμε μεγάλη ακρίβεια στη θέση που θέλουμε να φτάσει το αντικείμενο μας ή μικρή ακρίβεια αν θέλουμε να φτάσει πιο γρήγορα σε μία συγκεκριμένη θέση.

Το μικρότερο σφάλμα και επομένως η μεγαλύτερη ακρίβεια είναι όταν θέλουμε το χέρι να φτάσει στο κουτί και ο γάντζος να το περιβάλει. Ενώ το μεγαλύτερο σφάλμα το έχουμε όταν το σώμα του ρομπότ θέλουμε να φτάσει κόντα στο κουτί ή κοντά στο καλάθι.

```
    #error for reseting arm after dropping box in basket
    if self.sequence in ['G']:
        if err < 0.08:
            #after the last movement a new random robot position is calculated so that the robot goes after a new box position
            box_pt = np.random.choice(len(box_positions))
            box_pose = [0., 0., 0., box_positions[box_pt][0], box_positions[box_pt][1], box_size[2] / 2.0]
            box.set_base_pose(box_pose)
            #change_base_pos is called to initialize all movements in behavior tree with new box position
            change_base_pos(box.base_pose(),root)
            new_status = py_trees.common.Status.SUCCESS
    if new_status == py_trees.common.Status.SUCCESS:
        self.feedback_message = "Reached target"
        self.logger.debug("%s.update()[%s->%s][%s]" % (self.__class__.__name__, self.status, new_status, self.feedback_message))
    else:
        self.feedback_message = "Error: {0}".format(err)
        self.logger.debug("%s.update()[%s][%s]" % (self.__class__.__name__, self.status, self.feedback_message))
    return new_status

def terminate(self, new_status):
    self.logger.debug("%s.terminate()[%s->%s]" % (self.__class__.__name__, self.status, new_status))
```

Όταν ολοκληρωθεί η τελευταία ακολουθία κινήσεων sequence G και φτάσει το απαιτούμενο σφάλμα , τότε το κουτί παίρνει μία τυχαία καινούργια θέση και καλείται η συνάρτηση change_base_pos ώστε να αρχίσει η ακολουθία κινήσεων από την αρχή έχοντας ενημερωθεί για την καινούργια θέση του κουτιού.

Αυτή η υλοποίηση εξασφαλίζει ότι το ρομπότ θα ακολουθεί σχεδόν τις ίδιες ακολουθίες κινήσεων χωρίς να έχει μεγάλες αλλαγές στο σώμα του και κυρίως στη θέση του χεριού του ,καθώς αυτό φαινόταν να προκαλεί πολλά σφάλματα στις μελλοντικές λύσεις του pick and place προβλήματος (από τις 50 φορές που πρέπει να λυθεί το πρόβλημα.

Επίσης εξασφαλίζει ότι σε περίπτωση που το ρομπότ δεν βάλει το κουτί στο καλάθι θα ξαναπροσπαθήσει να το βάλει στις επόμενες ακολυθίες κινήσεων χωρίς επιπλοκές .

2.3 def change_base_pos

Στην συνάρτηση αυτή αρχικοποιείται το behavior tree και τις θέσεις target που πρέπει να φτάσει κάθε ακολουθία του ελεγκτή. Επίσης εισάγονται τα sequence : A,B,C,D,E,F,G έτσι ώστε να διαχωρίζονται οι ακολουθίες.

Δέχεται ως όρισμα τη θέση του κουτιού έτσι ώστε να μπορούμε να έχουμε 50 διαφορετικές υλοποιήσεις του ελεγκτή.

```
#remove children of root every time behavior tree is initialized
root.remove_all_children()
# Create sequence node (for sequential targets)
sequence = py_trees.composites.Sequence(name="Sequence")
```

Αρχικά αφαιρούνται όλα τα sequence του δέντρου από την προηγούμενη υλοποίηση του ελεγκτή και στη συνέχεια ορίζεται κάθε ακολουθία με την σειρά της

```
# Target A , target is close to box ,move robot body close to box
eef_link_name = "base_link"
tf_desired = base_pose

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0]-0.52, tf_desired.translation()[1], tf_desired.translation()[2]])

tf.set_rotation(tf_desired.rotation())
trA = ReachTarget(robot, tf, dt, eef_link_name, 'A',"Reach Target A")
# add target to sequence node
sequence.add_child(trA)
```

Η sequence A έχει ως στόχο να φτάσει το σώμα του ρομπότ κοντά στο κουτί με μια μικρή απόσταση , γιαυτό και έχει ως end effector το base_link του σώματος

```
# Target B , target is closer to box, move arm close to box
eef_link_name = "gripper_link"
tf_desired=base_pose

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0]+0.03, tf_desired.translation()[1], tf_desired.translation()[2]+0.1])
tf.set_rotation(tf_desired.rotation())

trB = ReachTarget(robot, tf, dt, eef_link_name, 'B', "Reach Target B")
# add target to sequence node
sequence.add_child(trB)
```

Η sequence B έχει ως στόχο να φτάσει το χέρι του ρομπότ κοντά στο κουτί με μια μικρή απόσταση και ο γάντος να περιβάλει το κουτί , γιαυτό και έχει ως end effector το gripper_link που είναι η αρχή του σώματος του gripper.

```
# Target C  , target is closer to box, gripper grabs box
eef_link_name = "gripper_left_finger_link"
tf_desired=base_pose

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0]-0.1, tf_desired.translation()[1], tf_desired.translation()[2]+0.1])
tf.set_rotation(tf_desired.rotation())

trC = ReachTarget(robot, tf, dt, eef_link_name, 'C', "Reach Target C")
# add target to sequence node
sequence.add_child(trC)
```

Η sequence C έχει ως στόχο λίγο πιο μποστά από το κουτί και στο ίδιο ύψος με τον προηγούμενο στόχο,ώστε ο γάντζος να πιάσει το κουτί με μία σχετικά γρήγορη κίνηση. Ως end effector έχουμε το gripper_left_finger_link ώστε να κλείσει ο γάντος.

```
# Target D   , target is higher from box, arm resets position while holding box
eef_link_name = "gripper_link"
tf_desired=base_pose

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0], tf_desired.translation()[1], tf_desired.translation()[2]+0.5])
tf.set_rotation(tf_desired.rotation())

trD = ReachTarget(robot, tf, dt, eef_link_name, 'D', "Reach Target D")
# add target to sequence node
sequence.add_child(trD)
```

Η sequence D έχει ως στόχο πιο ψηλά από το κουτί έτσι ώστε το χέρι να επαναφερθεί σε υψηλή θέση κρατώντας το κουτί με end effector τη βάση του gripper.

```
# Target E , target is basket ,body moves close to basket while holding box
eef_link_name = "base_link"
tf_desired=basket.base_pose()

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0]-0.52, tf_desired.translation()[1], tf_desired.translation()[2]])

tf.set_rotation(tf_desired.rotation())

trE = ReachTarget(robot, tf, dt,  eef_link_name, 'E',"Reach Target E")
# add target to sequence node
sequence.add_child(trE)
```

Η sequence D έχει ως στόχο να φτάσει το σώμα του ρομπότ κοντά στο κουτί με μια μικρή απόσταση , γιαυτό και έχει ως end effector το base_link του σώματος

```
# Target F  , target is closer to basket ,arm moves into basket while honding box
eef_link_name = "gripper_link"
tf_desired=basket.base_pose()

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0], tf_desired.translation()[1], tf_desired.translation()[2]+0.3])

tf.set_rotation(tf_desired.rotation())

trF = ReachTarget(robot, tf, dt,  eef_link_name, 'F',"Reach Target F")
# add target to sequence node
sequence.add_child(trF)
```

Η sequence F έχει ως στόχο να φτάσει το χέρι του ρομπότ κοντά στο κέντρο του καλαθιού και ο γάντος να βρίσκεται μέσα του, γιαυτό και έχει ως end effector το gripper_link που είναι η αρχή του σώματος του gripper.

```
# Target G ,target is higher to basket ,  arm drops box and resets position
eef_link_name = "gripper_link"
tf_desired=basket.base_pose()

tf = dartpy.math.Isometry3()
tf.set_translation([tf_desired.translation()[0], tf_desired.translation()[1], tf_desired.translation()[2]+0.5])

tf.set_rotation(tf_desired.rotation())

trG = ReachTarget(robot, tf, dt,  eef_link_name, 'G',"Reach Target G")
# add target to sequence node
sequence.add_child(trG)

# Add sequence to tree
root.add_child(sequence)
```

Η sequence G έχει ως στόχο πιο ψηλά από το κέντρο του καλαθιού έτσι ώστε το χέρι να επαναφερθεί σε υψηλή θέση αφήνοντας στην αρχή το κουτί με end effector τη βάση του gripper.

2.4 Εκτέλεση και έλεγχος αλγορίθμου

```
# Behavior Tree
py_trees.logging.level = py_trees.logging.Level.DEBUG

# Create tree root
root = py_trees.composites.Parallel(name="Root", policy=py_trees.common.ParallelPolicy.SuccessOnOne())

#initialize behavior tree
change_base_pos(box.base_pose(),root)



finish_counter = 0

# tick once
root.tick_once()
```

Αρχικοποιούμε για πρώτη φορά το δέντρο και εισάγουμε τον finish_counter που θα αυξάνεται κάθε φορά που το κουτί μπαίνει μέσα στο καλάθι.

```
for step in range(total_steps):
    if (simu.schedule(simu.control_freq())):
        # box basket positions for check
        box_translation = box.base_pose().translation()
        basket_translation = basket.base_pose().translation()

        #count every time box goes into basket
        if box_into_basket(box_translation, basket_translation, basket_z_angle):
            finish_counter += 1
            #after counting box into basket the box need to change position so it wont get counted again in the same movement sequence.
            #The final new random position is given when the arm resets its position in G sequence
            box_pt = np.random.choice(len(box_positions))
            box_pose = [0., 0., 0., box_positions[box_pt][0], box_positions[box_pt][1], box_size[2] / 2.0]
            box.set_base_pose(box_pose)

        # 50 times box goes into basket programm stops
        if (finish_counter >= 50):
            break

    if (simu.step_world()):
        break
    #runs every sequence of the tree
    root.tick_once()
```

Σε κάθε step του περιβάλοντος εκτελείται κάθε στιγμιότυπο των ακολουθιών του ελεγκτή με το root.tick_once() και ελέγχεται αν το κουτί είναι μέσα στο καλάθι και αν έχει εισαγθεί 50 φορές τότε σταματάει να τρέχει το πρόγραμμα.

Μέσα στον έλεγχο για τον αν το κουτί μπήκε στο καλάθι αφού αυξηθεί ο finish_counter το κουτί αλλάζει προσωρινά θέση παρόλο που δεν επαναφέρονται οι ακολουθίες του ελεγκτή με την change_base_pos .Αυτό γίνεται καθώς για κάθε step που το κουτί βρίσκεται μέσα στο καλάθι και το ρομπότ επαναφέρει το χέρι του σε υψηλή θέση ο finish_counter αυξάνεται με αποτέλεσμα η υλοποίηση να τρέχει μόνο μία φορά. Η πραγματική καινούργια θέση του κουτιού αλλάζει στο σφάλμα της τελευταίας ακολουθίας και ενημερώνεται. Είναι αναγκαίο το ρομπότ να επαναφέρει το χέρι σε υψηλή θέση καθώς αν δεν γίνει σε επόμενες υλοποιήσης το χέρι αχρηστεύεται.

2.5 Επιπλέον σχόλια για τη λειτουργικότητα του ελεγκτή

Ο ελεγκτής σε κάθε εφαρμογή του προκαλεί μία αρκετά συγκεκριμένη ακολουθία κινήσεων και είναι λειτουργικός σχεδόν σε όλες τις εφαρμογές του.

Σε περίπτωση όμως που το κουτί δεν προκύψει στο καλάθι στο τέλος της ακολουθίας κινήσεων το κουτί μεταφέρεται σε μία νέα τυχαία θέση ώστε ο ελεγκτής να προσπαθήσει να ξαναλύσει το πρόβλημα