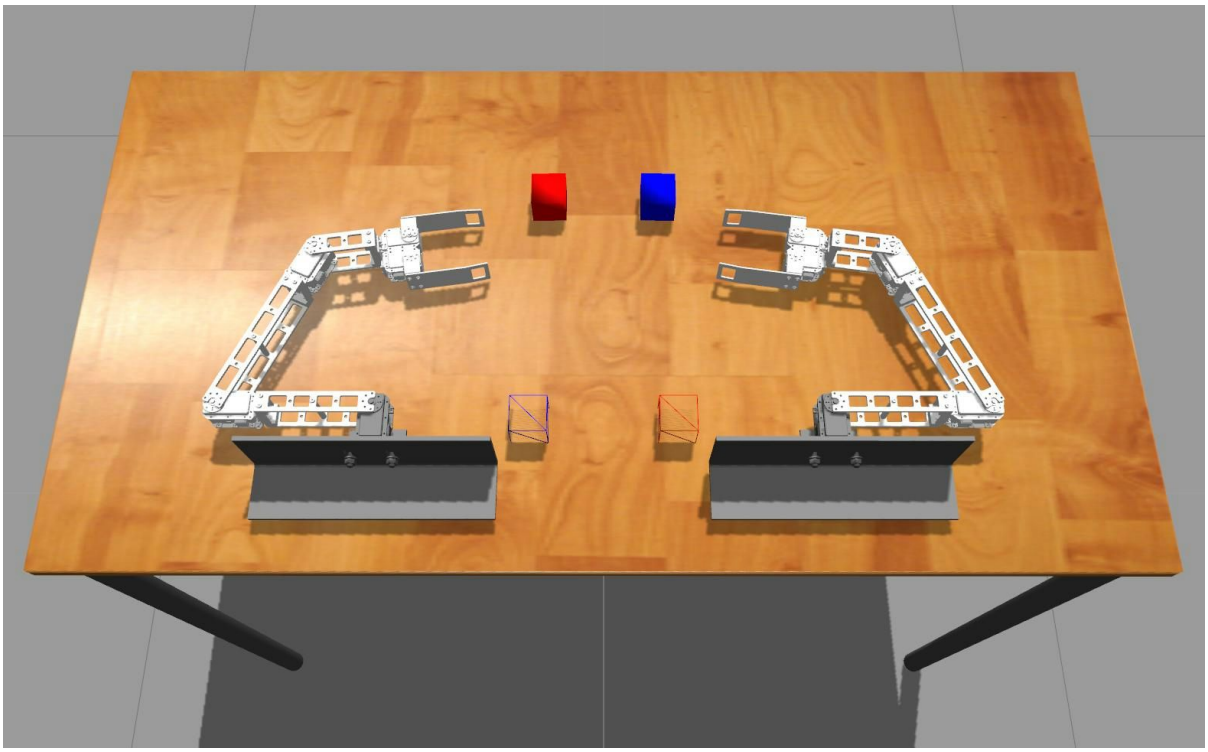
Combined Task and Motion Planing for Dual Arms in Pick/Place Problems



Full State Domain

In this example we define 2 arm 2 boxes and a goal position for each box. The mission is to manipulate the boxes using the two arms to its destination. The Full State Domain (FSD) and its solution functions as our worst case scenario in order to have a comparison case for other solutions. The FSD represent each state as a full description of the world i.e it includes all of the DoF.

The solution of the FSD relies on a discrete world such as “Manhattan Distance” principle. This way we constraints the motion of the boxes in a way that it can only move up, down, left, right, twist clockwise or counter-clockwise (one unit motion). It is important to note that any motion of the box is done by the arm, hence translated directly to the joints angles of the arm.

The Planning Domain Definition Language (PDDL) is as follows:

Domain:

Types: Grippers(Lgripper,Rgripper) // attached to a locations
// is attached to an arm with static base

Arms(Larm,Rarm) // left/right robotic arms

Boxes(B1,B2) // attached to a locations
// may attached to gripper

Positions((x_1,y_1,ϕ_1),(x_2,y_2,ϕ_2),...) // $x_i = i\frac{W}{N_x}$ $i = 1..N_x$
// $y_j = j\frac{H}{N_y}$ $j = 1..N_y$
// $y_k = -\pi + k\frac{2\pi}{N_\phi}$ $k = 1..N_\phi$

Predicates:

(Occupied	?P - position)	// is P occupied by any box
(Empty	?G - gripper)	// is gripper empty or not
(Obstructs	?B - box ?A - arm)	// is box and arm are in colission
(Obstructs	?A1 - arm ?A2 - arm)	// is arm1 and arm2 are in colission
(GripperAt	?G - gripper ?P - position)	// is gripper is at position
(BoxAt	?B - box ?P - position)	// is gripper holding the box

Initial State:

BoxAt(B1.home,B1)
BoxAt(B2.home,B2)
GripperAt(Rgripper,Rgripper.home)
GripperAt(Lgripper,Lgripper.home)

Goal State:

BoxAt(B1,B1.target)
BoxAt(B2,B2.target)

Actions:

- MoveGripperOnly(gripper,position) // Moves the gripper by 1 unit distance
  - Precon: Empty(gripper)
    ¬Occudied(position)
    ∀boxes ¬Obstucs(gripper.arm)
    ∀arms ¬Obstucs(gripper.arm)
  - Effect: GripperAt(gripper,position)

- MoveGripper&Box(Box,gripper,position) //Moves the gripper+object by 1 unit distance
  - Precon: ¬Empty(gripper)
    ¬Occupied(position)
    ∀boxes ¬Obstucs(gripper.arm)
    ∀arms ¬Obstucs(gripper.arm)

Effect: Occupied(position)
BoxAt(B1,position)

- Grasp(gripper) // close the gripper
  Precon: Empty(gripper)
  Effect: ¬Empty(gripper)

- Release(gripper) // open the gripper
  Precon: ¬Empty(gripper)
  Effect: Empty(gripper)

Separate Planning Domain

In this example we define 2 arm 2 boxes and a goal position for each box. The mission is to manipulate the boxes using the two arms to its destination. We define the Separate Planning Domain (SPD) as a planning sceme that distincts between the two planning phases. The SPD is representing each state as a partial state of the world due to pre defined positions on the map (although we treat the map with continuous parameters).

The solution of the SPD relies on the task planning flow with predicates/conditions that respects (or not) the motion plan. The main principle is to apply an motion action which calculated based on the "current" world state. This way we can use any "off the shelf" task and motion planners (apart).

The Planning Domain Definition Language (PDDL) of the SPD is as follows:

Domain:

Types:	gripper(Lgripper,Rgripper)	// attached to a locations
		// is attached to an arm with static base
	arm(Larm,Rarm)	// left/right robotic arms
	box(b1,b2)	// attached to a locations
		// may attached to gripper
	pose((x_1,y_1,ϕ_1),(x_2,y_2,ϕ_2),...)	// $x\in[0,L]$, $y\in[0,L]$, $\phi\in[-\pi,\pi]$

Predicates:

(Occupied	?p - pose ?b - box)	// is p occupied by b
(Occupied	?p - pose ?g - gripper)	// is p occupied by g
(Occupied	?g - gripper ?b - box)	// is g occupied by b
(Empty	?g - gripper)	// is gripper empty or not
(Obstructs	?b - box ?mp - motionplan)	// is box and mp are in colission **TBD**
(Obstructs	?a - arm ?mp - motionplan)	// is arm and mp are in colission **TBD**
(ExistMP	?p1 - pose ?p2 - pose)	// is exist motion plan from p1 to p2

Initial State:

Occudied(b1.start,b1)
Occudied(b2.start,b2)
Occudied(g1.start,g1)
Occudied(g2.start,g2)

Goal State:

Occudied(b1.target,b1)

Occudied(b2.target,b2)

Actions:

- MoveGripperOnly(gripper,pose) // Moves the gripper to a new pose

  Precon: Empty(gripper)

  ExistMP(gripper.pose,pose)

  ∀ boxes ¬Obstucs(box,mp) // during motion

  ∀ arms ¬Obstucs(arm,mp) // during motion

  Effect: Occupied(pose,gripper)

- MoveGripper&Box(box,gripper,pose) //Moves the gripper+box to a new pose

  Precon: Occupied(gripper,box)

  ExistMP(gripper.pose,pose)

  ∀ boxes ¬Obstucs(box,gripper.arm)

  ∀ arms ¬Obstucs(arm,gripper.arm)

  Effect: Occupied(pose,box)

- Grasp(gripper) // close the gripper

  Precon: Empty(gripper)

  Effect: ¬Empty(gripper)

- Release(gripper) // open the gripper

  Precon: ¬Empty(gripper)

  Effect: Empty(gripper)

Optional Solution for task planning:

- ❏ Any Graph Search

  Optional heuristics: Number of steps to goal
- ❏ And/Or trees

Optional Solution for motion planning:

- ❏ A*
- ❏ RRT
- ❏ PRM

Optional Heuristics: minimal path for all arms