High-Level Planning for Dual Arm Object Passing Tasks

Hamal Marino, Mirko Ferrati, Alessandro Settimi, Carlos Rosales, and Marco Gabiccini

***Abstract*— In this paper, we design a simple methodology for high-level planning, corroborated with low-level planning and execution, to solve a task having specified a priori a set of *interaction primitives*. We provide a realistic implementation focusing on the simple task of moving an object from a given initial to a desired final configuration, possibly exploiting the interaction between the object and the tabletop, as well as direct handoff between two hands when convenient. The interaction primitives are the different ways in which the object can be grasped and moved around, possibly different for each end-effector, and the planning step is carried out considering the table itself as an end-effector (with the *non-movable* attribute) with its own set of primitives. This results in a sequence of actions that are then translated into collision-free paths and, subsequently, low level commands for the robot. Experimental results with a bi-manual robotic platform show the viability of our approach, which can be well extended to multi-arm systems.**

Keywords: Dual Arm Manipulation; High-Level Planning; Bimanual Object Passing; Handoff

I. Introduction

In recent years, many approaches have been developed to allow a robot to execute a task by giving only high-level commands, and plan both the sequence of actions needed to complete the task, as well as the low-level sequence of movements to perform each single action.

In [1] a couple of robots demonstrates the capability of translating a natural language plan for making pancakes (found on the web), into a sequence of movements: one robot goes around the kitchen looking for ingredients, while the other is fixed at a table and prepares the pancake as soon as everything is placed inside its reachable workspace by the first robot.

In [2], a long-term planning strategy is devised such that, when used to displace an object, it also reasons about moving obstacles out of the way.

In [3], predefined grasp primitives are attached to objects which implement functional categories in a framework similar to object-specific reasoning introduced earlier in [4]; it uses PDDL language [5] and a semantic planner to decompose a given task into a series of subtasks.

H. Marino and A. Settimi are with the Dep. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, and with the Res. Center "E. Piaggio", University of Pisa, 56122 Pisa, Italy (hamal.marino@centropiaggio.unipi.it, alessandro.settimi@for.unipi.it)

M. Ferrati and C. Rosales are with the Res. Center "E. Piaggio", University of Pisa, 56122 Pisa, Italy (mirko.ferrati@gmail.com, carlos.rosales@for.unipi.it)

M. Gabiccini is with the DICI and Res. Center "E. Piaggio" University of Pisa, 56122 Pisa, Italy, and with the Dep. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova (phone: +39-050-221.80.77, fax: +39-050-221.80.65, email: m.gabiccini@ing.unipi.it)

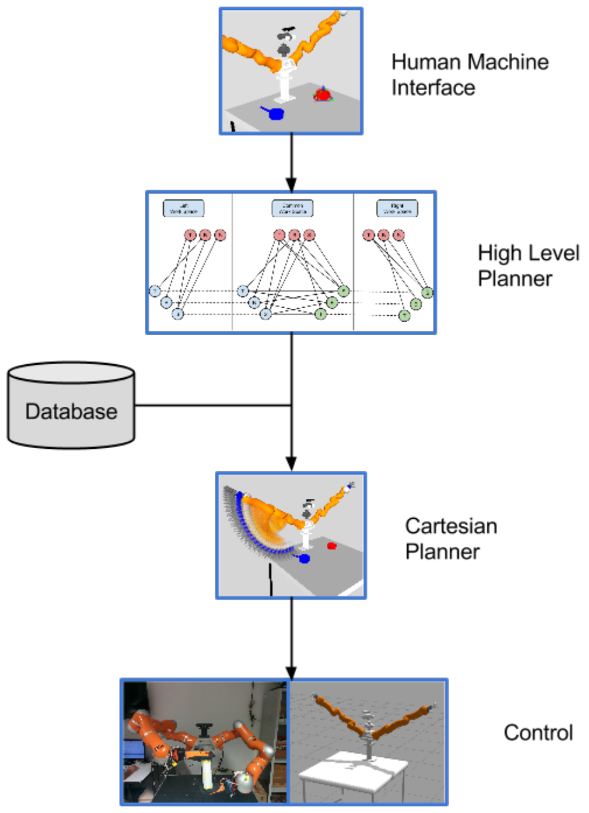

Fig. 1: Overview of the complete system.

[6] uses a hybrid task and path planner for positioning an object from a given initial to a desired final configuration, using pick and place actions that involves both arms of a bimanual robot. For all of these actions, the outcome is an abstract list of subtasks. The geometric level describing how a robot should actually perform each subtask can be carried out using probabilistic path planning methods, such as RRT [7] or PRM [8] to find a feasible, non-colliding path to each needed configuration.

[3] and [6] also use a geometric backtracking algorithm which allows for task-level replanning when a collision-free path has not been found, going back in the semantic plan, as much as needed, to find an alternative route to the goal.

Although all the previous are very promising approaches, we think they lack the possibility to include handoffs in the planning, which could be rather useful (if not necessary) in many situations.

This aspect is tackled instead in [9], where grasps are generated online once the object is perceived, as well as a way of performing the handoff. The disadvantage of this

Preprint submitted to 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 5, 2015.

approach is that it is limited to computing a single handoff transition. Therefore, the approach cannot generalize to more complex configuations when more than a single regrasping is needed.
This is not the case for [10] which, using pre-computed grasps and allowed grasp transitions (i.e., ways to perform the handoff), is capable of planning for multiple handoffs using an A* algorithm. The algorithm uses custom heuristics including distance of the object from its goal position and total number of handoffs penalties; the direct use of geometric information, although reduced to 3D, in the A* planner makes it computationally heavy. We believe that a better trade-off between speed and optimality can be obtained by implementing a high-level planner to decide which sequence of handoffs have to be performed.
Motivated by the idea of adding handoffs in a planning scheme which first considers task level planning and then geometric planning, we describe, in this paper, a methodology to solve a task given a set of *interaction primitives*.

We include, among the primitives, the generic action of passing the object from one end-effector to another, where the notion of end-effector is extended to consider also fixed, known environmental constraints when it is possible to exploit them: we will define them as *non-movable* end-effectors.

Possible grasps for each end-effector, as well as transitions between pairs of grasps, are pre-computed and stored in a database, avoiding an excessive increase in computational burden for the planning step.

Such a database can be generated via experiments, e.g. using a motion tracking system (PhaseSpace, [11]) to record hand and object pose during a grasp performed by a human "wearing" the robotic hand, simulations [12], or using diverse grasp synthesis algorithms, e.g. [13].

After highlighting the problem, we focus on the description of the proposed approach (Sec. II), and provide an in-depth view on the database structure (Sec. III), as well as the high-level planning strategy (Sec. IV). More on the low-level execution (Sec. IV-E) and other implementation details (Sec. V) follow. Experimental results (Sec. VI) and conclusions (Sec. VII) close the paper, discussing interesting extensions.

II. Problem Statement

The problem we look into to illustrate our methodology is "moving an object from a given initial pose to a desired target pose". Instead of specifying a way to achieve this result, we define a series of *interaction primitives* which are the ways in which an object can be manipulated from each end-effector. Given this simple specification, a sequence of actions may be needed in order to accomplish the task: a semantic planning step is used first, resulting in a feasible sequence of actions to be performed in order to reach the goal. A low-level planner has then to find a collision-free joint space trajectory for the robot in order to obtain the desired motion.

Specifically, we treat primitives that are: (i) handoffs, and (ii) picking/placing an object, incorporating (with the *movable* flag set to false) the environment such as a support surface in the high-level planning. It is worth remarking that a picking/placing primitive can be considered rightfully as a handoff between a movable end-effector (the hand) and a non-movable one that can apply only specific grasps to objects (see Fig. 2).

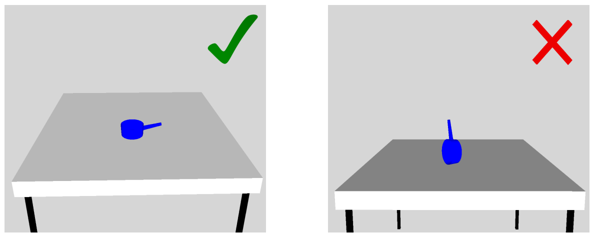
Fig. 2: A pot on a table. On the left a feasible bottom grasp, on the right an unfeasible side grasp.

Our working assumptions are that:

- vision gives to the planner all the necessary information, handling object recognition and pose estimation with sufficient accuracy;
- all objects appearing in the scene are either known or considered as fixed obstacles;
- information about the desired target configuration is given from a higher-level input, e.g. from a user (see Fig. 3).

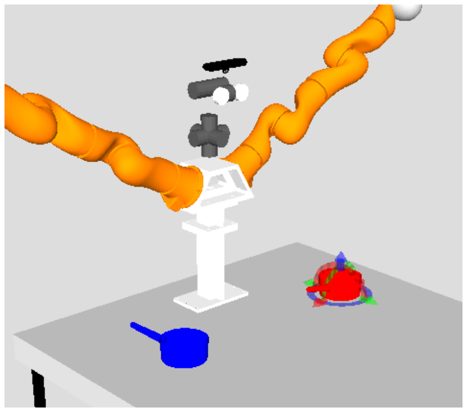
Fig. 3: The scene in Rviz. The blue object (left) represents the initial object configuration which is assumed to be given by the vision system; the red one (right) represents the goal selected by the user.

III. Database Description

The algorithm we devise relies on key information stored in a database in order to perform the planning step. A closer look at the tables therein is taken in the following. A very simple, exemplary database is shown in Fig. 4.

A. *Object Table*

All known objects are included in the database; for each of them, various object geometric information such as multiple 3D views which can be used for object recognition and pose estimation using a 3D vision library are stored.

B. *Grasp Table*

A grasp is defined as a Cartesian trajectory of the end-effector expressed in the object frame with the corresponding joint trajectory if any. A good estimate of the final object pose within the hand is relevant to accomplish the final and intermediate goals.

Then, a grasp is associated with a single object, however, an object can have more than one grasp associated.

The generation of this table can be done in several ways. It can be produced either using simulation-based synthesis approaches [12] as well as

For each object, a series of grasps are stored for each kind of end-effector (e.g. a left hand and a right hand; different hands can be considered, too). Each grasp consists of a cartesian trajectory of the end-effector expressed in object frame, as well as final in-hand object pose.
Specifically, we use simulated synthesis of the grasps as in [12].

C. *End-Effectors, Workspaces, and Reachability Tables*

Each end-effector is an entity which can act on an object: there are both "movable" end-effectors (such as robot hands) and "non-movable" ones (such as a fixed surface in the environment which can be exploited). From a semantic planning perspective, there is no distinction between movable and non-movable end-effectors, both types can interact with an object in specific ways, and can exchange the object with specific grasp transitions.
In order to generate the semantic workspaces table, the environment has to be divided in smaller regions (with a grid procedure or other segmentation algorithm), each one associated to a semantic workspace. These workspaces are semantically adjacent if their corresponding regions are adjacent in the 3D space. Both geometric and adjacency information is stored.
A workspace is considered to be reachable with an end-effector only if its associated region can be reached dexterously (i.e., with good mobility); this could be checked using, e.g., Capability Maps [14].

D. *Allowed Grasp Transitions*

We define possible transitions among different grasps with an associated *interaction primitive*, when there is no kinematic overlap between two different end-effectors performing the grasps at the same time.

The transitions we consider in this work are:

- pick and place actions when a non-movable end-effector is involved
- a sequence of grasp-ungrasp actions when both end-effectors are movable

To check whether, given a pair of grasps, a transition between them could exist, it is possible to check for kinematic feasibility (absence of collisions between the end-effectors) when both grasps are performed at the same time; alternatively, [15] developed an automatic algorithm for directly synthesizing grasps which allow for in-air re-grasping for fully actuated hands.

Grasp_id	EE_id	Name	Allowed_transitions
g1	e1	TopTable	g5 , g6 , g8 , g9
g2	e1	SideTable	g5 , g8
g3	e1	BottomTable	g4 , g5 , g7 , g8
g4	e2	TopLeftArm	g3, g8 , g9
g5	e2	SideLeftArm	g1 , g2 , g3 , g7, g9
g6	e2	BottomLeftArm	g1 , g7 , g8
g7	e3	TopRightArm	g3 , g5 , g6
g8	e3	SideRightArm	g1 , g2 , g3 , g4 , g6
g9	e3	BottomRightArm	g1 , g4 , g5

EE_id	Reachable Workspaces	Movable	Name
e1	w1,w2,w3	FALSE	Table
e2	w1,w2	TRUE	LeftArm
e3	w2,w3	TRUE	RightArm

Workspace_id	Adjacency	Geometry	Name
w1	w2	[x1,y1...xn,yn]	Left
w2	w1,w3	[x1,y1...xn,yn]	Middle
w3	w2	[x1,y1...xn,yn]	Right

Fig. 4: Example of a DataBase overall structure.

IV. High-Level Action Planning

The high-level semantic planning involves four distinct phases:

- construction of a graph with states and transitions
- conversion of cartesian information about current and target configurations to the semantic domain
- planning on a graph of a minimum cost path from initial to target state
- conversion of the semantic plan into a cartesian plan, which can then be executed from the robot

A *state* S is a possible (end-effector, grasp, workspace) tuple.

A. *Graph Construction*

Using the full DB structure as illustrated in Sec. III, a graph is generated. For the examplary content illustrated in Fig. 4, the generated graph can be seen in Fig. 5.
Each node of the graph represents a state $S_{e,g,w}$ of the object, where e is the end-effector which grasps the object with the grasp g in the workspace w. The arcs represent two different types of transitions:

1) from S_{e,g,w_i} to S_{e,g,w_j}: change of workspace with the same grasp, where w_i and w_j must be adjacent (in the sense of Sec. III-C) and e must be movable. This is represented by the dashed lines in Fig. 5.
2) from $S_{e_i,g_k,w}$ to $S_{e_j,g_l,w}$: change of end effector inside a certain workspace, where a transition between grasp g_k and g_l must exist (as described in Sec. III-D). This is represented by the continues lines in Fig. 5.

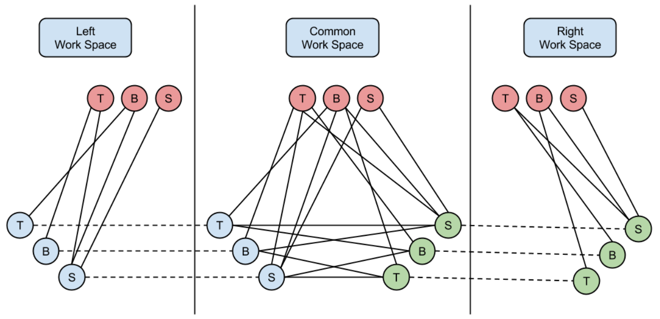

Fig. 5: Graph generated from the exemplary DB 4: 3 end-effectors, 3 workspaces, and 9 grasps. Left hand grasps, right hand grasps and table grasps are respectively represented with cyan, green and red circles while grasp types are indicated with T,B,S (top, bottom and side) inside the circles.

B. *Cartesian-to-Semantic Conversion*

Given the workspaces geometry associated with semantic workspace definitions, semantic states S_i and S_t representing initial and target object configurations are obtained as follows:

1) the workspaces containing the required poses are found (w_i, w_t)
2) by assumption, we consider only the non-movable end-effectors present in each of the workspaces (e_i, e_t)
3) a search for grasps (g_i, g_t) that fulfill the initial and final pose is performed.

This search involves all possible grasps associated to the end-effectors (e_i, e_t), which are checked for similarity w.r.t. the initial and target poses of the object; given the "non-movable" condition, this check is performed considering invariance to translations, and rotations around an axis normal to the end-effectors.

C. *Planning on Graph*

Using the graph generated in Section IV-A, Dijkstra algorithm [16] is used to find the high-level shortest path from S_i to S_t. We show in Fig. 6 a possible plan generated by hand, while in Fig. 7 the solution found by Dijkstra for the same case.
Another plan with different initial and target states that uses the table as intermediate end-effector in order to minimize path length is shown in Fig. 8. Each shortest path represents a set of workspace/grasp semantic transitions, which need to be converted in a sequence of cartesian commands for the robot.

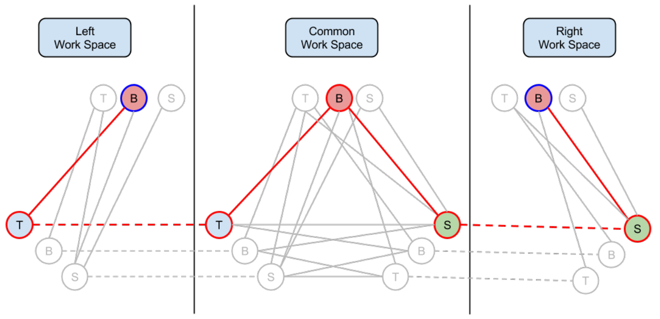

Fig. 6: Possible planning on the graph generated previously: highlighted are the initial and final states (in blue) and the path in between (red).

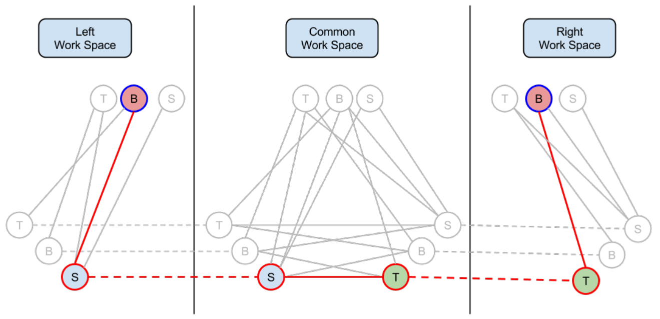

Fig. 7: Planning given by Dijkstra algorithm, the transition on the table is avoided because it would require more grasp transitions for the task.

D. *Semantic-to-Cartesian Conversion*

The high-level sequence of transitions obtained from the planning is simplified, by eliminating the transitions which are not useful, e.g. an end-effector moving through multiple workspaces $S_{e,g,w_1}...S_{e,g,w_n}$ becomes a single cartesian command to the final workspace w_n location. Next, each non-fixed workspace location of the object (initial and target configurations are given) are converted into a cartesian pose for the object such that all the end-effectors involved in that transition can reach it.
For this purpose, when one of the involved end-effectors

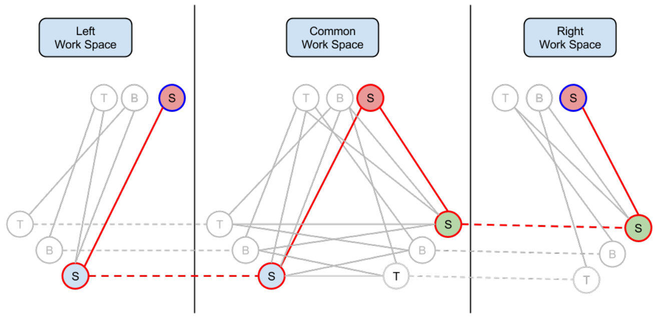

Fig. 8: Planning given by Dijkstra algorithm, the transition on the table is used because it is the shortest one in this example of initial and final states.

is not movable (e.g., the table), the position of the object is computed only considering possible rotations around the local axis aligned with the table normal (z-axis in world frame). Instead, when both end-effectors are movable, an initial guess of the object pose is considered to be where the end-effectors are mostly aligned with a "preferred" direction, and a local IK search is performed until a feasible robot configuration is found. Finally, each arc in the semantic plan is associated to low-level commands, in particular a change of workspace generates a *move* command for an end-effector, while a change of grasp generates a sequence of *grasp-ungrasp* commands for the end-effectors involved.

Summarizing, the performed steps before execution are:

- eliminating redundant semantic steps,
- translating semantic workspaces into cartesian poses,
- computing reachable object poses for each transition considering involved end-effectors,
- using the transition primitive information to plan the actions needed (i.e., *move*, *grasp-ungrasp*).

E. Execution of the Plan

The execution of the commands requires the use of a low level Inverse Kinematics planning. If more than a single end-effector *move* command has to be performed, they are performed simultaneously computing collision-free trajectories that involves the various end-effectors. This can be efficiently done e.g. using a sample-based random planner such as RRT [7]. Grasp trajectories stored in the database (Sec. III-B), which are in a frame local to the object, are translated into world coordinates and executed as well.

V. Implementation Details

Both simulated and real environment have been used to perform dual-manipulation experiments with the Vito robot[1]. The implementation of the planning algorithm, as well as the instructions to download and install the simulation package are available online as open source software[2]. A high-level scheme of the full pipeline of the experiment is depicted in Fig. 9.

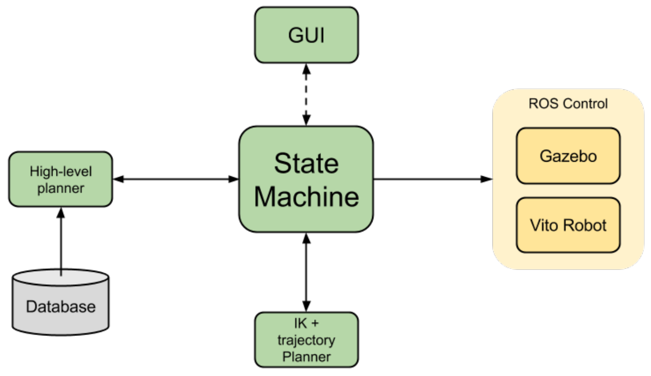

Fig. 9: High-level scheme of the full experiment pipeline with highlighted specific software blocks.

As shown in Fig. 9 the core of the framework is represented by the State-Machine, its internal structure is reported in Fig. 10. In the various states, the State-Machine communicates with different modules depending on its needs (e.g. in the *Getting Info* state it asks the vision for the starting position and the GUI for the target position).

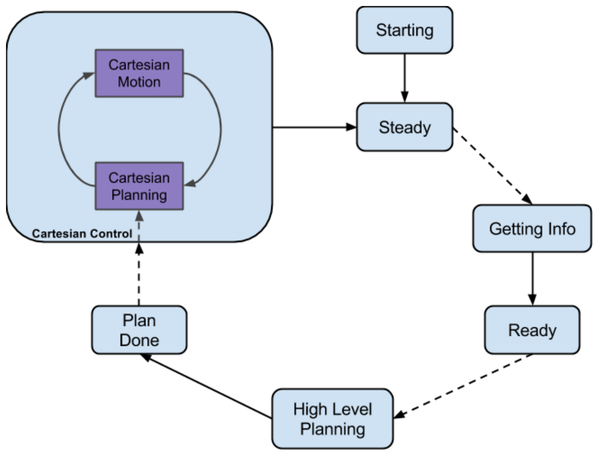

Fig. 10: The *State-Machine* structure is reported. Only the transition representing the main execution flow are in figure.

The user can, through the *GUI*, send commands to the *State-Machine* to perform a dual manipulation task. In particular, the dashed arrows in Fig. 10 represent the input from the user that can hence:

- get the object start position from the vision and set the desired final position for it,
- ask to an high level plan for the desired task,
- use the cartesian plan corresponding to the semantic generated one to perform the task with the robot (simulated or real one).

[1]http://www.pacman-project.eu/software

[2]https://bitbucket.org/dualmanipulation/dualmanipulation

The *Cartesian* control state has a sub-state-machine that iteratively plan cartesian movements and execute them up to completion of the plan generated in the *High Level Planning* state.

The experiment software, written in C++, runs on two Linux Ubuntu 14.04 PCs, using ROS [17] as a middleware for communication between vision, high- and low-level planning, and control.

VI. Experimental Results

A. Set-Up

The robot used for the validation of the approach is the Vito robot in Centro "E. Piaggio" (see Fig. 11, Fig. 12, Fig. 13).

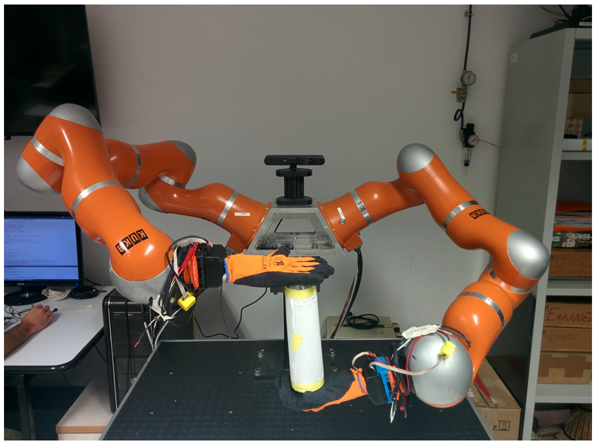
Fig. 11: Vito robot.

Two Kuka Light-Weigth Robots (LWR) mounted on a fixed torso have been equipped with a left and a right Pisa-IIT Soft-Hands.

B. Performed Experiments

In the proposed experiments a cylinder has been used. While the starting position (blue) is given by the vision system, the target position (red) is chosen by the user using the GUI[3].

In the first experiment (Fig. 12) the object starting pose is in the workspace of the right arm and the user selects the goal position in the workspace of the left arm with a similar orientation. The high level planner leads to the following behavior:

1) the robot picks up the cylinder from the top with the right hand,
2) the object is brought in the center of the common workspace,
3) an aerial handoff is performed with the left hand grasping the object from the bottom,
4) the right hand realeases the object and the robot puts the cylinder in the target position, inside the left workspace.

[3]Videos regarding the robot Vito performing the described experiments will be available at *https://www.youtube.com/user/centroepiaggio*.

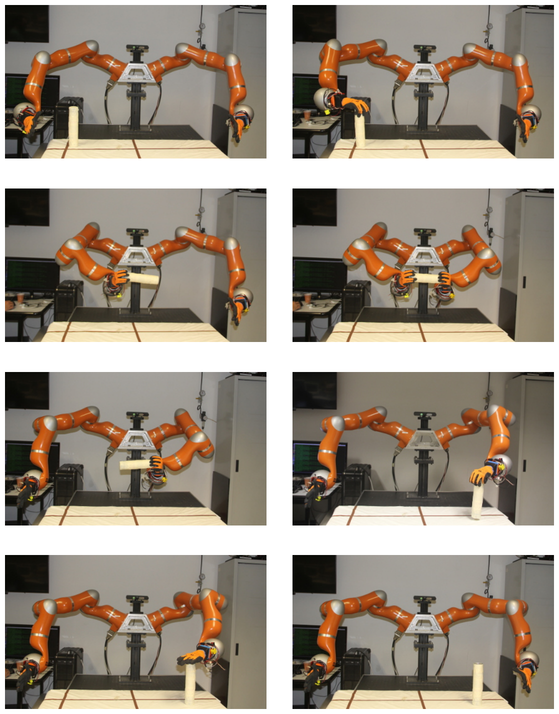
Fig. 12: Experiment 1

Concerning the second experiment (Fig. 13), the cylinder is positioned again in the right arm workspace, this time laying on a side; the user selects the goal position in the workspace of the left arm with a similar orientation. The resulting behavior is slightly different:

1) the robot performs a side grasp on the object with the right hand,
2) then, it brings the cylinder to the common workspace, and the table is used as temporary support (i.e. the table performs a side grasp),
3) with another side grasp the robot uses the left arm to grasp the object and brings it to the target workspace.

For the sake of clarity, the reader can refer to Fig. 8 for a simplified semantic plan, which conceptually corresponds to the results depicted in Fig. 13.

VII. Conclusions

In this work we proposed a multi layer planner to solve complex object passing tasks using both handoffs and picking/placing primitives, executed by multiple end-effectors. The use of a high level semantic graph coupled with low level cartesian planner and precomputed grasp transitions allows to handle complex plans without incurring in heavy computational load. The implementation was tested on a

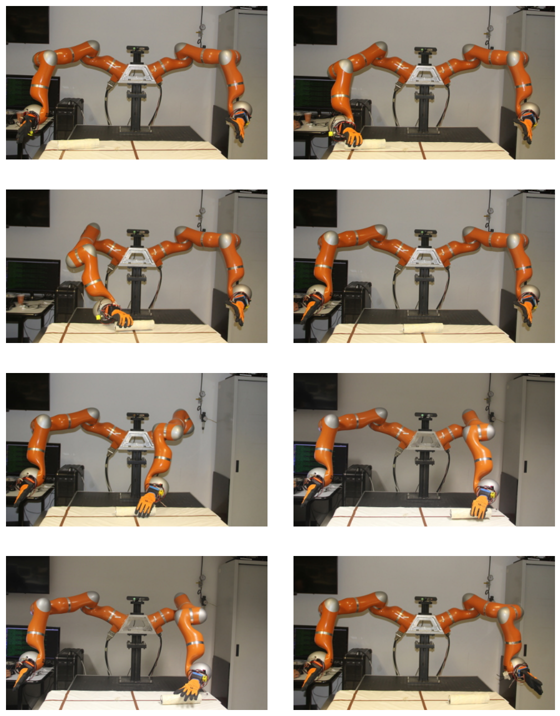

Fig. 13: Experiment 2

dual-arm robot using a cylinder with few associated grasps in order to test both the planner and the whole system in configurations that can be easily solved by a human, but that can still provide a testbed for the overall integration.

Ongoing development aims at adding arc costs which considers grasp and transition quality, in order to improve the planner decisions. Moreover, future works should extend the concept of "interaction primitives" to include dynamic primitives and extrinsic manipulation such as [18].

To allow for possible corrections during the procedure execution, we plan to include an Execution Monitoring System as suggested in [4], in order to check for correctness at every stage of the plan to be executed and considering replanning when needed.
Possible monitoring to check whether the object was successfully grasped could include visually checking the object position or reading current consumption from hand joint, thus understanding if the hand is applying force to the object or not.

Acknowledgments

This work is supported by the European commission projects PaCMan EU FP7-ICT 600918.

References

[1] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 529–536.

[2] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[3] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 429–435.

[4] L. Levison, "Connecting planning and acting via object-specific reasoning," Ph.D. dissertation, Citeseer, 1996.

[5] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. E. Smith *et al.*, "Pddl-the planning domain definition language," 1998.

[6] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system." Citeseer, 2012.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[9] J.-P. Saut, M. Gharbi, J. Cortés, D. Sidobre, and T. Siméon, "Planning pick-and-place tasks with two-hand regrasping," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4528–4533.

[10] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Proceedings of Robotics: Science and Systems*, 2014.

[11] Phase Space. (2015) Phase space motion capture. [Online]. Available: http://www.phasespace.com/

[12] M. Bonilla, E. Farnioli, C. Piazza, M. Catalano, G. Grioli, M. Garabini, M. Gabiccini, and A. Bicchi, "Grasping with soft hands," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, Madrid, Spain, November 2014.

[13] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees."

[14] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. Ieee, 2007, pp. 3229–3236.

[15] B. Balaguer and S. Carpin, "Bimanual regrasping from unimanual machine learning," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3264–3270.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[17] "ROS: Robot operating system," www.ros.org.

[18] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1578–1585.